

Requel: A Collaborative Requirements Tool with Automated Assistance

Ronald Regan Jr.

A Thesis in the Field of Information Technology
for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

June 2009

Abstract

Poor requirements have been cited as the primary cause of software project failures and cost overruns. Fredrick Brooks tells us that “The hardest single part of building a software system is deciding precisely what to build.”

What causes poor quality requirements and why is it so hard to acquiring requirements that are understandable, unambiguous, precise, and complete? One of the key problems in identifying requirements is a lack of stakeholder involvement. All stakeholders, including users, managers, and developers, need to be involved during requirements acquisition. Another problem is the high level of detail necessary to implement software. Most people don’t think about the intricacies of the tasks they perform at a conscious level.

Requel is a requirements engineering system that supports collaboration among business and technical stakeholders and provides automated assistance to validate requirements and suggest improvements. It is a Web-based application to facilitate a distributed team of users. It supports collaboration with a semi-structured discussion and negotiation mechanism. It assists users by applying natural language processing to the requirements to detect and report ambiguity and complexity. It identifies potential significant terms in the requirements to assist in build a glossary and making the requirements more understandable and consistent.

Requel is open source software licensed under the GNU Public License and available from SourceForge at <http://requel.sourceforge.net/>.

Dedication

This thesis is dedicated to my wife Theresa, for all the love and support she has given that made this dream possible.

Acknowledgments

I would like to thank my thesis director, Dr. Bill Robinson, for his support throughout the thesis process. His guidance helped me get to the end when I would lose focus and stray off the path.

I would also like to thank all the people out there creating great open source software that made this project possible.

Table of Contents

Table of Contents.....	vi
List of Tables.....	xli
List of Equations.....	xliii
List of Figures.....	xliv
Chapter 1 Introduction.....	1
Software Requirements Engineering.....	3
 Acquisition.....	4
 Communication and Specification.....	9
 Analysis.....	11
 Methods and Techniques.....	15
Natural Language Processing.....	17
 Sentence Segmentation.....	17
 Morphology.....	17
 Spelling.....	18
 Parsing.....	18
 Word Sense Disambiguation.....	20
 Semantic Role Labeling.....	22
 Corpus Linguistics.....	23
Related Tools.....	23
Requirements Apprentice.....	24

<u>Requirements Assistant for Telecommunication Services (RATS)</u>	26
<u>UC Workbench</u>	26
<u>Procasor</u>	29
<u>gIBIS</u>	29
<u>EColabor and Tuiqiao</u>	30
<u>Feature Comparison</u>	30
<u>Summary</u>	31
<u>Organization of this Document</u>	31
<u>Chapter 2 System Requirements</u>	33
<u>Goals</u>	33
<u>Actors</u>	35
<u>Administrator</u>	35
<u>Automated Assistant</u>	35
<u>Project User</u>	36
<u>Use Cases</u>	38
<u>Create or edit a user account</u>	39
<u>Login to the system</u>	40
<u>Create a new project</u>	41
<u>Create or edit a stakeholder</u>	42
<u>Create or edit a goal</u>	45
<u>Add or edit a relationship between two goals</u>	46
<u>Create or edit a use case</u>	46
<u>Create or edit a scenario</u>	47

Annotate a requirements element with a note or issue.....	50
Add a position to an issue.....	51
Add an argument to a position.....	52
Select a Position as the resolution to an issue.....	53
Add or edit a term in the project glossary	54
Link two glossary entries.....	56
Generate a document for a project.....	57
Chapter 3 Design and Implementation.....	58
Architecture.....	58
User Interface Layer.....	59
Domain Layer.....	61
Service Layer.....	65
Problem Domain Model.....	65
User.....	66
Project User Role.....	67
System Admin User Role.....	67
Stakeholder.....	67
Stakeholder Permission.....	68
Project.....	68
Goal.....	69
Goal Relation.....	69
Story.....	70
Actor.....	70

<u>Scenario</u>	71
<u>Step</u>	71
<u>Use Case</u>	71
<u>Glossary Term</u>	72
<u>Document</u>	72
<u>Note</u>	73
<u>Issue</u>	73
<u>Position</u>	74
<u>Argument</u>	74
<u>External Components</u>	74
<u>Spring Framework</u>	74
<u>Java Persistence API and Hibernate</u>	76
<u>Echo2 Framework</u>	78
<u>Hibernate Validator</u>	79
<u>Java Architecture for XML Binding (JAXB)</u>	79
<u>Stanford Parser</u>	81
<u>OpenNLP</u>	81
<u>WordNet SQL builder</u>	82
<u>Implementation</u>	83
<u>Package Structure</u>	83
<u>Application Initialization</u>	84
<u>User Interface Framework</u>	86
<u>Requel User Interface</u>	92

Command Handler	93
Command Factories	96
Domain Object Proxy	96
Word Sense Disambiguation	101
Semantic Role Labeling	101
Chapter 4 User Guide	103
Getting Started	103
Working with Requirements	106
Creating a Project	107
Project Navigation	108
Creating and Editing Stakeholders	109
Creating and Editing Goals	114
Creating and Editing Stories	118
Creating and Editing Actors	120
Creating and Editing Scenarios	123
Creating and Editing Use Cases	125
Editing Scenario Steps	127
Generating Documents	130
Adding a Document Generator	132
Working with Terms	133
Discussing and Negotiating Requirements	135
Adding Notes and Issues	136
Working with Analysis Issues	140

<u>Requel Setup</u>	142
<u>System Requirements</u>	142
<u>Create a Database User</u>	142
<u>Configuration</u>	144
<u>Deploying to Apache Tomcat</u>	145
<u>Completing the Setup</u>	147
<u>User Administration</u>	148
Chapter 5 Summary and Conclusions	151
<u>Lessons Learned</u>	151
<u>Scope and Process</u>	151
<u>Architecture and Design</u>	153
<u>Technology</u>	154
<u>Natural Language Processing</u>	158
<u>Conclusion</u>	159
<u>Future Work</u>	160
<u>Natural Language Processing</u>	160
<u>Domain Knowledge</u>	161
<u>User Interface</u>	161
<u>References</u>	162
Appendix 1 Application Code	167
<u>Java Code</u>	167
<u>abstractannotation.java</u>	167
<u>abstractannotationcommand.java</u>	171

abstractappawareactionlistener.java	171
abstractappawarecontroller.java	172
abstractassistant.java	172
abstractcommand.java	174
abstractcommandfactory.java	175
abstractcomponent.java	175
abstractcomponentmanipulator.java	176
abstractcontroller.java	176
abstractdictionarycommand.java	178
abstractdictionarylemmatizerrule.java	178
abstrecteditcommand.java	178
abstrecteditorpanel.java	179
abstrectedittreenodefactory.java	186
abstrectedittreenodeupdatelistener.java	186
abstrecteditprojectcommand.java	187
abstrecteditprojectordomainentitycommand.java	188
abstractjparepository.java	189
abstractlistcomponentmanipulator.java	191
abstractnavigatortreenodefactory.java	192
abstractnavigatortreenodeupdatelistener.java	193
abstractnlptextwalker.java	193
abstractopennlptool.java	194
abstractpanel.java	195

<u>abstractpanelcontainerpanel.java</u>	196
<u>abstractpanelcontainerscreen.java</u>	197
<u>abstractprojectcommand.java</u>	197
<u>abstractprojectordomain.java</u>	198
<u>abstractprojectordomainentity.java</u>	204
<u>abstractrepository.java</u>	207
<u>abstractrequelannotationeditorpanel.java</u>	208
<u>abstractrequelcommandcontroller.java</u>	209
<u>abstractrequelcomponent.java</u>	210
<u>abstractrequelcontroller.java</u>	210
<u>abstractrequeleditorpanel.java</u>	211
<u>abstractrequelnavigatortable.java</u>	211
<u>abstractrequelprojecteditorpanel.java</u>	212
<u>abstractscreen.java</u>	213
<u>abstractsenserelationinfo.java</u>	214
<u>abstractsysteminitializer.java</u>	215
<u>abstracttextentity.java</u>	215
<u>abstractusercommand.java</u>	216
<u>abstractuserrole.java</u>	216
<u>actor.java</u>	219
<u>actor2actorimpladapter.java</u>	219
<u>actorassistant.java</u>	220
<u>actorcontainer.java</u>	220

actorcontainerstable.java	221
actoreditorpanel.java	223
actorimpl.java	229
actornavigatorpanel.java	231
actorselectorpanel.java	235
actorstable.java	238
addactorposition.java	242
addactortoactorcontainercommand.java	243
addactortoactorcontainercommandimpl.java	243
addactortoactorcontainercontroller.java	244
addglossarytermposition.java	245
addgoaltogoalcontainercommand.java	246
addgoaltogoalcontainercommandimpl.java	247
addgoaltogoalcontainercontroller.java	248
addstorytostorycontainercommand.java	249
addstorytostorycontainercommandimpl.java	249
addstorytostorycontainercontroller.java	250
addwordtodictionaryposition.java	251
adjectivematchingrule.java	252
adminuserinitializer.java	252
adverbmatchingrule.java	253
analysisinvokingcommandhandler.java	253
analyzableeditcommand.java	254

annotatable.java	254
annotation.java	255
annotationcommandfactory.java	255
annotationcommandfactoryimpl.java	257
annotationexistsexception.java	259
annotationreferertable.java	260
annotationrepository.java	262
annotationstable.java	264
appawarecontroller.java	268
applicationcontextcommandfactorystrategy.java	268
applicationexception.java	269
argument.java	270
argumenteditorpanel.java	271
argumentimpl.java	275
argumentpositionsupportlevel.java	278
assistantfacade.java	279
assistantuserinitializer.java	283
batchcommand.java	284
batchcommandimpl.java	284
buildwordnetdefinitionwordscommand.java	285
calculatelowdfrequencecommand.java	285
calculatelowdfrequencecommandimpl.java	285
category.java	286

changespellingposition.java	288
checkboxmanipulator.java	289
checkboxset.java	289
checkboxsetmanipulator.java	291
checkboxsetmodel.java	291
checkboxtreeset.java	293
checkboxtreesetmanipulator.java	294
checkboxtreesetmodel.java	295
classpathfilemanagerimpl.java	299
closepaneevent.java	300
collectionmanipulator.java	300
colocationsenserelationinfo.java	301
combinedlistmodel.java	302
combinedtextlistmodel.java	303
comboboxmanipulator.java	305
command.java	305
commandfactory.java	306
commandfactorystrategy.java	306
commandhandler.java	307
componentmanipulator.java	307
componentmanipulators.java	307
constituenttreedepthfinder.java	309
constituenttreeprinter.java	309

constraintviolationexceptionadapter.java	311
controller.java	311
convertsteptoscenariocommand.java	312
convertsteptoscenariocommandimpl.java	312
copyactorcommand.java	314
copyactorcommandimpl.java	314
copygoalcommand.java	316
copygoalcommandimpl.java	316
copyscenariocommand.java	318
copyscenariocommandimpl.java	318
copyscenariostepcommand.java	320
copyscenariostepcommandimpl.java	320
copystorycommand.java	322
copystorycommandimpl.java	322
copyusecusecasecommand.java	324
copyusecusecasecommandimpl.java	325
createdentity.java	326
databasecreationlistener.java	327
databaseinitializationlistener.java	328
databaseinitializer.java	329
databasespelldictionary.java	329
dateadapter.java	330
dateutils.java	331

defaultcommandhandler.java	332
defaulteditortreenode.java	333
defaulteventdispatcher.java	334
defaultpanelfactory.java	340
defaultpanelmanager.java	343
deleteactorcommand.java	348
deleteactorcommandimpl.java	349
deleteargumentcommand.java	350
deleteargumentcommandimpl.java	351
deletedentityevent.java	352
deleteteglossarytermcommand.java	352
deleteteglossarytermcommandimpl.java	352
deletegoalcommand.java	354
deletegoalcommandimpl.java	354
deletegoalrelationcommand.java	355
deletegoalrelationcommandimpl.java	356
deleteissuecommand.java	357
deleteissuecommandimpl.java	357
deletenotecommand.java	358
deletenotecommandimpl.java	358
deletepositioncommand.java	359
deletepositioncommandimpl.java	359
deletereportgeneratorcommand.java	361

deletereportgeneratorcommandimpl.java	361
deletescenariocommand.java	362
deletescenariocommandimpl.java	362
deletescenariostepcommand.java	364
deletescenariostepcommandimpl.java	364
deletestakeholdercommand.java	365
deletestakeholdercommandimpl.java	365
deletestorycommand.java	367
deletestorycommandimpl.java	367
deleteusecasecommand.java	368
deleteusecasecommandimpl.java	369
dependencyprimaryverbfinder.java	370
dependencyprinter.java	371
dependencysubjectfinder.java	372
dependencyverblerelationfinder.java	373
describable.java	374
dictionary.java	374
dictionarycommandfactory.java	375
dictionarycommandfactoryimpl.java	377
dictionaryinitializer.java	378
dictionaryphoneticcodeinitializer.java	380
dictionaryrepository.java	380
dictionarysqlinitializer.java	386

<u>dictionarysuffixexchanginglemmatizerrule.java</u>	388
<u>dictionizer.java</u>	390
<u>digesterruleloggingdecorator.java</u>	391
<u>documentationinitializationlistener.java</u>	392
<u>domainadminuserrole.java</u>	393
<u>domainobjectwrapper.java</u>	393
<u>domainobjectwrappingadvice.java</u>	395
<u>domainuserinitializer.java</u>	397
<u>downloadbutton.java</u>	397
<u>editactorcommand.java</u>	398
<u>editactorcommandimpl.java</u>	399
<u>editaddactortoprojectpositioncommand.java</u>	401
<u>editaddactortoprojectpositioncommandimpl.java</u>	402
<u>editaddwordtodictionarypositioncommand.java</u>	403
<u>editaddwordtodictionarypositioncommandimpl.java</u>	403
<u>editaddwordtogglossarypositioncommand.java</u>	404
<u>editaddwordtogglossarypositioncommandimpl.java</u>	404
<u>editannotationcommand.java</u>	406
<u>editargumentcommand.java</u>	406
<u>editargumentcommandimpl.java</u>	407
<u>editchangespellingpositioncommand.java</u>	409
<u>editchangespellingpositioncommandimpl.java</u>	409
<u>editcommand.java</u>	410

editdictionarywordcommand.java	410
editdictionarywordcommandimpl.java	411
editglossarytermcommand.java	411
editglossarytermcommandimpl.java	412
editgoalcommand.java	414
editgoalcommandimpl.java	415
editgoalrelationcommand.java	417
editgoalrelationcommandimpl.java	417
editissuecommand.java	420
editissuecommandimpl.java	420
editlexicalissuecommand.java	421
editlexicalissuecommandimpl.java	421
editmode.java	423
editnotecommand.java	423
editnotecommandimpl.java	424
editorcomponents.java	425
editortree.java	428
editortreecellrenderer.java	433
editortreedraganddropnodefactorydecorator.java	434
editortreemanipulator.java	434
editortreenode.java	436
editortreenodeabstractdecorator.java	437
editortreenodeactionbuttondecorator.java	439

editortreenodedraganddropdecorator.java	440
editortreenodefactory.java	441
editortreenodeupdatelistener.java	442
editpositioncommand.java	442
editpositioncommandimpl.java	442
editprojectcommand.java	444
editprojectcommandimpl.java	444
editprojectordomainentitycommand.java	447
editreportgeneratorcommand.java	447
editreportgeneratorcommandimpl.java	448
editscenariocommand.java	449
editscenariocommandimpl.java	449
editscenariostepcommand.java	451
editscenariostepcommandimpl.java	452
editsemlinkrefcommand.java	453
editsemlinkrefcommandimpl.java	453
editsensecommand.java	455
editsensecommandimpl.java	455
editstakeholdercommand.java	457
editstakeholdercommandimpl.java	458
editstorycommand.java	460
editstorycommandimpl.java	461
editsynsetcommand.java	463

editsynsetcommandimpl.java	463
editsynsetdefinitionwordcommand.java	464
editsynsetdefinitionwordcommandimpl.java	465
edittextentitycommand.java	466
editusecasecommand.java	467
editusecasecommandimpl.java	467
editusercommand.java	470
editusercommandimpl.java	471
editverbnetselectionrestrictioncommand.java	474
editverbnetselectionrestrictioncommandimpl.java	474
entityexception.java	476
entityexceptionactiontype.java	479
entityexceptionadapter.java	479
entityexistsexceptionadapter.java	479
entitylockexception.java	480
entityproxyinterceptor.java	481
entityvalidationexception.java	483
eventdispatcher.java	484
eventdispatcherexception.java	486
eventdispatchermultiexception.java	487
exceptionmapper.java	487
exceptionmappingcommandhandler.java	490
exportdictionarycommand.java	490

exportdictionarycommandimpl.java	491
exportprojectcommand.java	492
exportprojectcommandimpl.java	492
generatereportcommand.java	494
generatereportcommandimpl.java	494
genericpropertyvalueexceptionadapter.java	496
glossaryterm.java	497
glossaryterm2glossarytermimpladapter.java	497
glossarytermeditorpanel.java	498
glossarytermimpl.java	504
glossarytermnavigatorpanel.java	506
glossarytermreferable.java	510
glossarytermselectorpanel.java	513
glossarytermstable.java	516
goal.java	519
goal2goalimpladapter.java	519
goalassistant.java	520
goalcontainer.java	520
goalcontainerstable.java	521
goaleditorpanel.java	523
goalimpl.java	530
goalnameinuseexception.java	532
goalnavigatorpanel.java	532

goalrelation.java	536
goalrelationeditorpanel.java	537
goalrelationimpl.java	542
goalrelationtype.java	546
goalselectorpanel.java	546
goalselfrelationexception.java	550
goalstable.java	550
grammaticalrelation.java	554
grammaticalrelationimpl.java	555
grammaticalrelationtype.java	556
grammaticalstructurelevel.java	562
hashutils.java	562
identitylemmatizerrule.java	563
importdictionarycommand.java	564
importdictionarycommandimpl.java	564
importprojectcommand.java	565
importprojectcommandimpl.java	566
importsemcorcommand.java	568
importsemcorcommandimpl.java	568
importsemcorfilecommand.java	571
importsemcorfilecommandimpl.java	571
initappevent.java	572
integernlptextwalker.java	573

invalidstateexceptionadapter.java	573
issue.java	574
issueeditorpanel.java	574
issueimpl.java	581
jaxbannotatablepatcher.java	584
jaxbannotationgroupedbypatcher.java	584
jaxbcreatedentitypatcher.java	585
jaxbororganizedentitypatcher.java	586
jaxbususerrolepatcher.java	587
jpaannotationrepository.java	587
jpadictionaryrepository.java	590
jpaprojectrepository.java	604
jpauserrepository.java	610
labelmanipulator.java	613
lemmatizerrule.java	613
lexicalassistant.java	613
lexicalissue.java	623
lexicalmatchingrule.java	624
lexlinkref.java	625
lexlinkrefid.java	627
linkdef.java	629
loadwordnettaggedglossescommand.java	630
loadwordnettaggedglossescommandimpl.java	630

lockacquisitionexceptionadapter.java	634
logincommand.java	634
logincommandimpl.java	635
logincontroller.java	636
loginevent.java	636
loginfailedevent.java	637
loginokcontroller.java	637
loginokevent.java	638
loginscreen.java	639
logoutcontroller.java	645
logoutevent.java	646
logoutokcontroller.java	646
logoutokevent.java	647
mainscreentabbednavigation.java	647
messagehandler.java	648
morespecificwordsuggester.java	648
morphdef.java	649
morphref.java	649
morphrefid.java	650
namedentity.java	651
navigationevent.java	651
navigationinitializationcontroller.java	652
navigatorbutton.java	653

navigatable.java	654
navigatablecellvaluefactory.java	657
navigatablecolumnconfig.java	658
navigatableconfig.java	658
navigatablemanipulator.java	659
navigatablemodel.java	660
navigatablemodeladapter.java	661
navigatablepanel.java	661
navigatortree.java	662
navigatortreenode.java	664
navigatortreenodefactory.java	666
navigatortreenodeupdatelistener.java	666
navigatortreepanel.java	666
nlpnavigatorpanel.java	668
nlpnavigatortreenodefactory.java	668
nlppanelnames.java	669
nlpprocessor.java	670
nlpprocessorexception.java	670
nlpprocessorfactory.java	670
nlpprocessorfactoryimpl.java	672
nlptext.java	675
nlptextimpl.java	679
nlptextwalkerfunction.java	686

nosuchactorexception.java	686
nosuchannotationexception.java	687
nosuchentityexception.java	688
nosuchglossarytermexception.java	689
nosuchorganizationexception.java	690
nosuchpositionexception.java	690
nosuchprojectexception.java	692
nosuchroleforuserexception.java	692
nosuchuserexception.java	693
nosuchwordexception.java	694
note.java	695
noteeditorpanel.java	695
noteimpl.java	698
nounphrasefinder.java	699
nounphrasematchingrule.java	699
nullcomponentmanipulator.java	701
oddeventablecellrenderer.java	702
opennlpparser.java	703
opennlptagger.java	705
opennlptokenizer.java	708
openpanelevent.java	709
optimisticlockexceptionadapter.java	711
organization.java	711

organizationimpl.java	711
organizedentity.java	713
panel.java	713
panelactiontype.java	715
panelcontainer.java	715
paneldescriptor.java	716
panelfactory.java	716
panelmanager.java	716
parserexception.java	717
parserpanel.java	718
parsetag.java	719
partofspeech.java	725
partofspeechandsenseprinter.java	727
persistencecontexthelper.java	728
position.java	729
positioneditorpanel.java	729
positionimpl.java	735
predicateverbfinder.java	739
prepositionmatchingrule.java	739
primaryverbphrasefinder.java	740
project.java	741
projectassistant.java	741
projectcommandfactory.java	742

projectcommandfactoryimpl.java	745
projectimpl.java	749
projectmanagementpanelnames.java	752
projectnameinuseexception.java	753
projectnavigatorpanel.java	754
projectnavigatortreenodefactory.java	755
projectopenissuesnavigatorpanel.java	761
projectordomain.java	764
projectordomainentity.java	765
projectordomainentityassistant.java	766
projectoverviewpanel.java	769
projectrepository.java	775
projectset.java	777
projectteam.java	778
projectteamimpl.java	778
projectuserinitializer.java	780
projectusernavigatortreenodefactory.java	780
projectuserrole.java	783
propertycontainer.java	785
propertycontainersupport.java	785
reflectioncommandfactorystrategy.java	786
reflectivetetable.java	786
reflectivetablecellrenderer.java	787

reflectiveablecolumnmodel.java	788
reflectivetablemodel.java	788
reflectivetablesselector.java	789
reflectivetree.java	790
reflectivetreemodel.java	791
reflectivetreenode.java	791
reflectutils.java	793
removeactorfromactorcontainercommand.java	797
removeactorfromactorcontainercommandimpl.java	798
removeactorfromactorcontainercontroller.java	799
removeactorfromactorcontainerevent.java	800
removeannotationfromannotatablecommand.java	800
removeannotationfromannotatablecommandimpl.java	801
removegoalfromgoalcontainercommand.java	802
removegoalfromgoalcontainercommandimpl.java	802
removegoalfromgoalcontainercontroller.java	803
removegoalfromgoalcontainerevent.java	804
removestoryfromstorycontainercommand.java	805
removestoryfromstorycontainercommandimpl.java	805
removestoryfromstorycontainercontroller.java	806
removestoryfromstorycontainerevent.java	807
removeunneedlexicalissuescommand.java	808
removeunneedlexicalissuescommandimpl.java	808

replaceglossarytermcommand.java	810
replaceglossarytermcommandimpl.java	811
reportdownloadprovider.java	813
reportgenerator.java	814
reportgeneratoreditorpanel.java	814
reportgeneratorimpl.java	819
reportgeneratornavigatorpanel.java	820
repository.java	824
requelexception.java	825
requel mainscreen.java	826
requelupdatedentitynotifier.java	828
resloveissuecommand.java	829
resloveissuecommandimpl.java	830
resloveissuewithaddactorpositioncommandimpl.java	831
resloveissuewithaddglossarytermpositioncommandimpl.java	832
resloveissuewithaddwordtodictionarypositioncommandimpl.java	833
resloveissuewithchangespellingpositioncommandimpl.java	834
resourcebundlehelper.java	836
retryonlockfailurescommandhandler.java	838
scenario.java	840
scenarioassistant.java	841
scenariocontainer.java	842
scenarioeditorpanel.java	842

scenarioeditortreenodefactory.java	848
scenarioimpl.java	849
scenarionavigatorpanel.java	851
scenarioscenariostable.java	855
scenarioselectormanagepanel.java	858
scenariostepassistant.java	861
scenariostepeditor.java	863
scenariostepeditormodel.java	865
scenariostepeditortreenodefactory.java	866
scenariostepseditor.java	868
scenariotype.java	872
scenariousecasetable.java	873
screen.java	876
selectentityevent.java	876
selectorbutton.java	877
selectorbuttonmanipulator.java	880
selectortablepanel.java	880
semanticrelatednesssenserelationinfo.java	882
semanticrole.java	882
semanticrolecollector.java	885
semanticrolecollectorfunction.java	885
semanticrolelabeler.java	886
semanticrolelabelerexception.java	889

semanticroleprinter.java	890
semanticsimilaritysenserelationinfo.java	891
semcorfile.java	892
semcorinitializer.java	893
semcorsentence.java	895
semcorsentenceword.java	897
semlinkref.java	900
semlinkrefid.java	902
sense.java	903
senseid.java	906
senserelationinfo.java	907
sentencizer.java	907
setscreenevent.java	909
simplelemmatizer.java	909
simpleleskwsd.java	910
spellingchecker.java	912
spellingsuggester.java	913
stakeholder.java	913
stakeholdereditorpanel.java	914
stakeholderimpl.java	920
stakeholdernavigatorpanel.java	924
stakeholderpermission.java	929
stakeholderpermissionimpl.java	929

stakeholderpermissionsinitializer.java	932
stakeholderpermissiontype.java	933
staleobjectstateexceptionadapter.java	934
stanfordlexicalizedparser.java	934
stanfordnameentityrecognizer.java	937
stemmer.java	940
step.java	945
stepimpl.java	945
storiestable.java	948
story.java	952
story2storyimpladapter.java	952
storyassistant.java	953
storycontainer.java	953
storycontainerstable.java	954
storyeditorpanel.java	956
storyimpl.java	962
storynavigatorpanel.java	965
storyselectorpanel.java	969
storytype.java	972
stringnlptextwalker.java	973
subjectphrasefinder.java	973
synset.java	974
synsetdefinitionword.java	977

synsethypernymwalkcommand.java	980
synsethypernymwalkcommandimpl.java	981
syntaxmatchingcontext.java	982
syntaxmatchingrule.java	983
systemadminuserrole.java	984
systeminitializer.java	984
tabbedpanelcontainer.java	985
textcomponentmanipulator.java	986
textentity.java	987
textentityassistant.java	987
togglebuttonmodelex.java	988
typedassistant.java	988
uiframeworkapp.java	988
uiframeworkconfiguration.java	991
uiframeworkexception.java	992
uiframeworklogoutservlet.java	993
uiframeworkservlet.java	993
uimethoddisplayhint.java	993
uitypedisplayhint.java	994
unmarshallerlistener.java	995
unmarshallerlistener.java	996
untar.java	997
updatedentitynotifier.java	999

updateentityevent.java	999
usecase.java	1000
usecaseassistant.java	1000
usecaseeditorpanel.java	1001
usecaseimpl.java	1007
usecasenavigatorpanel.java	1010
usecaseselectorpanel.java	1014
user.java	1017
user2userimpladapter.java	1020
useradminnavigatorpanel.java	1020
usercollectionnavigatorTreeNodeFactory.java	1021
usercommandfactory.java	1023
usercommandfactoryimpl.java	1023
usereditorpanel.java	1024
userentityexception.java	1029
userimpl.java	1030
usernameinuseexception.java	1035
usernavigatorTreeNodeFactory.java	1036
userpropertyvalueexceptionadapter.java	1037
userrepository.java	1037
userrole.java	1038
userroleexistsexception.java	1039
userrolepermission.java	1040

userrolepermissionsinitializer.java	1042
userset.java	1042
usersetimpl.java	1043
verbmatchingrule.java	1045
verbnetclass.java	1046
verbnetframe.java	1047
verbnetframeref.java	1049
verbnetframesyntaxparser.java	1051
verbnetimporter.java	1053
verbnetrole.java	1053
verbnetroleref.java	1054
verbnetselectionalrestrictionsparser.java	1056
verbnetselectionrestriction.java	1057
verbnetselectionrestrictiontype.java	1058
word.java	1060
wordnet.java	1062
wordnetdefinitionwordsinitializer.java	1064
wordnethyponymcountinitializer.java	1066
wordnetsensekeyinitializer.java	1069
wordnetwsd.java	1071
workflowdisposition.java	1078
XSLT Code	1079
project2html.xslt	1079

<u>XML</u>	1087
<u>uiMainConfig.xml</u>	1087
<u>commandHandlerConfig.xml</u>	1088
<u>jpaConfig.xml</u>	1089

List of Tables

Table 1: Influence of Areas of Concerns.....	17
Table 2: Related Tools Feature Comparison.....	31
Table 3: Goals of the Requel System.....	35
Table 4: Stakeholder Roles, with Examples.....	37
Table 5: Consequences of the MVC Pattern.....	60
Table 6 User Properties.....	66
Table 7 Project User Role Properties.....	67
Table 8 Stakeholder Properties.....	68
Table 9 Stakeholder Permission Properties.....	68
Table 10 Project Properties.....	69
Table 11 Goal Properties.....	69
Table 12 Goal Relation Properties.....	70
Table 13 Story Properties.....	70
Table 14 Actor Properties.....	70
Table 15 Scenario Properties.....	71
Table 16 Step Properties.....	71
Table 17 Use Case Properties.....	72
Table 18 Glossary Term Properties.....	72
Table 19 Document Properties.....	72
Table 20: Note Properties.....	73

Table 21: Issue Properties.....	73
Table 22: Position Properties.....	74
Table 23: Argument Properties.....	74
Table 24: Requel Source Code Packages.....	84
Table 25: Step Node Controls.....	128
Table 26: Automated Task Positions.....	141

List of Equations

Equation 1: WordNet Information Content.....13

List of Figures

Figure 1: Example Constituent Parse.....	19
Figure 2: Example Dependency Parse.....	19
Figure 3: Example Passive Dependency Parse.....	20
Figure 4: Example Semantic Role Labeling.....	23
Figure 5: UC Workbench.....	28
Figure 6: Architecture Layers.....	58
Figure 7: Assistant Architecture.....	61
Figure 8: Language Component Architecture.....	64
Figure 9: Users, Roles and Stakeholders.....	66
Figure 10: Project Elements.....	68
Figure 11: Annotation Objects.....	73
Figure 12: Application Initialization.....	85
Figure 13: Database Initializers.....	86
Figure 14: UI Framework Screens and Panels.....	88
Figure 15: UI Framework Initialization.....	89
Figure 16: UI Framework Events.....	90
Figure 17: Event Processing Sequence of Login up to the EventDispatcher.....	91
Figure 18: Event Processing Sequence of Login after the EventDispatcher.....	91
Figure 19: Requel Main Screen.....	92
Figure 20: Main Requel Screen Components.....	93

Figure 21: Command Handlers.....	94
Figure 22: Command Handler Sequence.....	95
Figure 23: Command Factories.....	96
Figure 24: Domain Object Wrapping Advice in Command.....	99
Figure 25: Domain Object Wrapping Advice out of Command.....	100
Figure 26: Entity Proxy in Action.....	100
Figure 27: Login Screen.....	104
Figure 28: Main Screen.....	104
Figure 29: Projects Navigation Details.....	105
Figure 30: Edit User Screen.....	105
Figure 31: Edit User Screen with Errors.....	106
Figure 32: New Project Screen.....	107
Figure 33: Project Import File Selection.....	108
Figure 34: Project Navigation Tab.....	109
Figure 35: Stakeholder Navigator.....	110
Figure 36: Sorting Stakeholders.....	110
Figure 37: Edit Stakeholder Screen.....	111
Figure 38: Selecting a User for a Stakeholder.....	112
Figure 39: Stakeholder Permissions.....	112
Figure 40: Stakeholder Goals.....	113
Figure 41: Goal Selector.....	114
Figure 42: Goal Navigator Screen.....	115
Figure 43: New Goal Screen.....	116

Figure 44: Goal Referring Entities.....	116
Figure 45: Goal Relations.....	117
Figure 46: Goal Relation Editor Screen.....	117
Figure 47: Goal Relation "To Goal" Selection.....	118
Figure 48: Stories Navigation Screen.....	119
Figure 49: Story Editor Screen.....	119
Figure 50: Story Goals and Actors.....	120
Figure 51: Actors Navigator Screen.....	121
Figure 52: New Actor Screen.....	121
Figure 53: Referenced Goals.....	122
Figure 54: Scenarios Navigation Screen.....	124
Figure 55: New Scenario Screen.....	124
Figure 56: Scenario's Referring Scenarios.....	125
Figure 57: Use Case Navigation Screen.....	125
Figure 58: Use Case Edit Screen.....	126
Figure 59: Use Case Auxiliary Actors.....	127
Figure 60: Use Case Stories.....	127
Figure 61: Empty Scenario Step Editor.....	127
Figure 62: Empty Step Node.....	128
Figure 63: Select Scenario Screen.....	129
Figure 64: Scenario Steps Editor.....	129
Figure 65: Documents Navigator.....	130
Figure 66: Generating a Document.....	131

Figure 67: Requirements Document.....	132
Figure 68: Document Editor Screen.....	132
Figure 69: Terms Navigator.....	133
Figure 70: A Term with a Canonical Term.....	134
Figure 71: Term Editor.....	134
Figure 72: Canonical Term Selected.....	134
Figure 73: Replace Term.....	135
Figure 74: Annotations Section.....	136
Figure 75: New Note Editor.....	137
Figure 76: Shared Notes.....	137
Figure 77: New Issue Editor.....	138
Figure 78: Adding Positions.....	138
Figure 79: Position Editor.....	139
Figure 80: Argument Editor.....	139
Figure 81: Open Issues.....	140
Figure 82: Example Analysis Issue.....	140
Figure 83: Example Analysis Positions.....	141
Figure 84: Create the Database User.....	143
Figure 85: Extracting the db.properties file from the WAR File.....	144
Figure 86: Editing the Database Properties.....	145
Figure 87: Updating the WAR file.....	145
Figure 88: Apache Tomcat Manager.....	146
Figure 89: WAR File Deployment.....	146

Figure 90: Requel in the Tomcat Application Manager.....	147
Figure 91: Users Navigation Tab.....	148
Figure 92: User Editor.....	149
Figure 93: Username Already in Use.....	149
Figure 94: User Roles and Permissions.....	149

Chapter 1 Introduction

This thesis describes Requel, a collaborative tool that helps business and technical stakeholders to identify and elaborate a set of software requirements. Requel supports users with automated assistance that analyzes the requirements, identifies issues, and suggests solutions.

The primary objective of the tool is to facilitate the creation of high quality software requirements. More precisely, requirements should be:

- Understandable and unambiguous – all stakeholders will understand and agree on the expected behavior of the desired system; all terminology that may be ambiguous or unfamiliar to any of the stakeholders is clearly defined in a glossary.
- Precise and complete – there is enough explicit information in the requirements that a developer can implement them; a tester can verify that the implementation behaves correctly; and all the clients' needs are met.
- Well-organized – Relationships and dependencies between requirements elements are indicated clearly.

One of the key problems in identifying requirements is a lack of stakeholder involvement (Finkelstein, 1994, p. 4.) All stakeholders including users, managers, and developers need to be involved. Requel supports a collaborative environment for all stakeholders to participate in the process, including the following properties and features:

- Accessibility – Requel supports concurrent and distributed access through a rich Web interface in a Web browser so stakeholders can access it from anywhere.
- Process Adaptability – Requel is adaptable to different software engineering processes and activities by supporting requirements using a set of simple elements that includes goals, stories, actors, scenarios, and use-cases, but does not require using all element types.
- Extensibility and Customizability – Requel can interact with other tools through an import/export mechanism using the Extensible Markup Language (XML) and XML Stylesheet Language Transformations (XSLT). It supports customization of documentation through custom XSLT scripts.
- Issue discussion and negotiation – Requel supports a structured process for identifying and resolving issues through discussion of various candidate solutions with multiple arguments that support or oppose the solution to assist in making decisions.

The rest of this chapter has an overview of the requirements engineering domain and how Requel supports it; a description of the natural language processing tasks that Requel uses to perform analysis; a description of similar tools in the requirements engineering space and the approaches they use to solve problems similar to those proposed here.

Software Requirements Engineering

Poor requirements have been cited as the primary cause of software project failures and cost overruns. Fredrick Brooks said it best over 20 years ago in his article “No Silver Bullet: Essence and Accidents of Software Engineering” (Brooks, 1987)

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements...”

The crux of the problem lies in the level of detail and precision necessary to produce software. Brooks (1987) asserts that “the client does not know what he wants” and that “it is really impossible for a client to specify the exact requirements” because of the detail necessary.

The field of requirements engineering has grown out of the need to identify and solve the problems that make identifying requirements so difficult. As this project is concerned with collecting software requirements it is important to understand the goals and issues of requirements engineering to see what problems the tool is trying to solve.

There are many definitions for requirements and what the requirements engineering process entails that make it a confusing topic. Furthermore there are many names given to the process, such as elicitation, gathering, defining, and so forth that add to the confusion. In the context of this thesis “requirements” means “the functionality, qualities, and constraints imposed on a system to correctly support its purpose.” Given this definition the requirements engineering process is tasked with both defining the purpose in precise terms, and identifying the functionality and qualities that meet the purpose and the constraints imposed by the environment.

When working on a complex problem, decomposing it into simpler sub-problems is an effective approach. Many authors have decomposed the space of requirements engineering into various sub-areas. Finkelstein (1994) identifies seven areas of concern that reflect the basic structure in the requirements engineering process: context, groundwork, acquisition, modeling and specification, analysis, measurement, and communication and documentation. Nuseibeh and Easterbrook (2000) expand on Finkelstein's areas, combine some and extract others, although primarily focusing on the activities. Weigers (1999, pp. 38-39) organizes his best practices into seven categories: knowledge, requirements management, project management, elicitation, analysis, specification and verification.

Below are descriptions of the relevant areas of concern borrowing the concepts from Finkelstein and Weigers. For each area of concern there is a description of how Requel addresses it.

Acquisition

Acquisition is concerned with identifying the environment, stakeholders and requirements of the project (Finkelstein, 1994, pp. 3-4.) Acquisition is a continuous and iterative process of refinement. The earliest activities are concerned with understanding the problem from the point of view of the organizational and environmental setting (Mylopoulos & Castro, 2000, p. 2.) and why the system is needed (Yu 1997, p. 1.)

Requel supports environmental concerns as goals, notes and issues attached to the project during early analysis.

Stakeholders

Stakeholder analysis involves identifying the roles of the people and organizations that should be involved in the project, the individuals or organizations that fill those roles, and the responsibilities, capacities and relationships of those stakeholders (Finkelstein, 1994, p. 3.) Robinson and Volkov (1997, p. 1) extend analysis to include the stakeholders “which affect or are affected by the system” such as developers and managers. Before the stakeholders can be understood they must be identified. Sharp, Finkelstein, and Galal (1999) point out that there wasn’t a lot of help in the literature for identifying the stakeholders of a project, and propose a simple approach. They include examples of roles such as end-users, managers, developers, and regulatory bodies. They also describe categorizations that have been suggested by other authors that focus on the stakeholders’ relationship to the system being developed. Stakeholders may be identified as a project progresses; a project may start with a user heavy team and add technical stakeholders after a first pass to help elaborate technical details.

Beyond helping to identify stakeholders the roles of stakeholders also indicate their responsibilities in a project. Business stakeholders drive the business level and quality requirements and make decisions about the scope of the product. They include executives, managers, or staff from various disciplines like marketing, sales, services, and support. User stakeholders or surrogate users drive the user level requirements. Technical stakeholders include requirements engineers, domain experts and developers that elaborate functional requirements from the business, user and quality requirements. Finally there are supporting stakeholders that may not contribute to the requirements, but play a role in the process. Project managers organize the activities and measure progress and costs; quality

assurance engineers make sure the requirements can be verified as the system is implemented.

Requel supports identifying and tracking stakeholders, but it doesn't assist in identifying them by suggesting the type of people to include, which is left for future work. Two types of stakeholders are supported: individual stakeholders, which are system users, as well as non-user stakeholders to represent organizations or regulatory bodies. Stakeholders may have goals that define the environmental (business, technical, and legal) concerns particular to that stakeholder. Requel doesn't support explicit roles, but it does support different permissions for authorizing user stakeholders' ability to edit, annotate and delete the requirement elements as a way to restrict activity of users.

Requirements

Requirements are identified at various levels from high level goals to low level functions, and not in any particular order. Requirements are not gathered in a strictly top down approach because “people don't think that way” (Rich & Waters, 1990, p. 200.) The identification of one use case or goal may lead to others, or a new step in one scenario may indicate the need for a corresponding step in another. To facilitate working top-down, bottom-up or middle-out, Requel supports creating requirements elements in any order. For example a team may start by creating stories based on actual events and elaborate goals and use cases that abstract the main concepts in the stories.

Different requirements acquisition methods use different entities to describe the requirements. Agile methods, such as eXtreme Programming attempt to minimize documentation and focus on user stories to define requirements. A user story is a simple

narrative for the interaction of a user with the system. It is not intended to capture all the details of the interaction, but to be more of an overview of how a feature may work. Requel supports textual stories with a simple type of “success” or “exception” indicating if the story describes normal usage or a problem. Stories may have goal or actor associations and be used as “casual” use cases using Cockburn’s terminology (2000, p. 120.)

Goal oriented methods such as Eric Yu’s i* or Knowledge Acquisition in Automated Specification (KAOS) model requirements in graphs of goals and their relationships, where goals may support or conflict with each other. Weigers covers a variety of models for describing requirements, including qualities and functional requirements of the form “the system shall...” Using Lamsweerdem’s definition of a goal (2001, p. 2) as “an objective the system under consideration should achieve,” Weigers qualities and functional requirements can be considered as types of goals.

Requel uses goals in a general sense that includes answers to why, what and how questions concerning the system being developed. In this sense the goals include desired qualities, properties, constraints, features and functions.

Weigers (1999, p. 7) identifies three distinct levels of requirements: business, user and functional, as well as various nonfunctional or quality requirements. In this author’s opinion separating the goals into fine grained categories can be tedious for users and it doesn’t add significant value because the level of abstraction or intent of a goal is evident in the language used. For example “The system should be easy to use for non-technical users” is a soft goal that doesn’t lead to a specific function of the system, but dictates a quality of the system. On the other hand a goal like “The system needs to be accessible to users in multiple locations” dictates a mode of operation for the system as a whole. Finally,

goals like “The system must support an IBIS style discussion and negotiation of issues” dictate a feature and functionality that the system must support. Not categorizing the goals removes the burden of the user having to choose a category and the impact of the user choosing the wrong category.

More important than categorizing goals is identifying the relationships between them. Requel supports identifying a directed relationship from one goal to another with a quality indicating that the source goal supports or conflicts with the target goal. For example having the goal “a user must assign a level to a goal for it to be processed” conflicts with the goal that the system should be easy to use. The relationships help users understand how the goals impact or influence each other and identify areas where tradeoffs between conflicting goals are needed.

Scenario based methods such as Cooperative Requirements Engineering with Scenarios (CREWS) use structured scenarios to define processes and agents that invoke them. Methods such as the Object-Oriented Software Engineering (OOSE) or the Rational Unified Process (RUP) focus on use cases, which are typically composed of multiple scenarios. Scenarios in CREWS differ from use cases in that they specify only a single flow through a process. Multiple scenarios would be needed to describe a single use case with alternate and exceptional paths. There are a variety of formats for structuring use-cases in natural language. Cockburn (2000, ch. 11) covers textual use-cases from fully dressed formats with detailed scenarios for success and failure cases, to casual formats that present scenarios as narratives more like eXtreme Programming user stories. Requel supports scenario, use case, and story elements that can be used to represent scenarios and use cases of various methods. Each scenario can have multiple steps, where a step may be a

single statement or a reference to another scenario. Each step has a type: pre-condition, primary, alternative, exception, and optional. Pre-conditions aren't actually steps but represent conditions that must hold for the use case to start. The other types represent the different flows through a use case. A use case has a primary actor, a scenario, references to auxiliary actors, goals, and stories for example uses of the use case. As noted before stories have goals and actors and can be used as casual use cases.

Communication and Specification

Requirements engineering is fundamentally an exercise in communication and understanding. The customer stakeholders must communicate their needs so that the development stakeholders can understand what they truly need and determine what is feasible in the provided environment.

The requirements specification is an agreement between the customer and developers that defines in a precise way what the resulting implementation should do (Weigers 1999, p. 148). There are a variety of ways to structure the specification; both Weigers (1999, p. 153-154) and Cockburn (2000, p. 13-14) include outlines and descriptions of what should typically be included in a specification.

Given the varying documentation needs of projects, Requel supports customizable documentation generation using XML Stylesheet Language Transformations (XSLT) stylesheets. A project may have multiple stylesheets to generate documents with different levels of details and formats. It is also possible to create diagrams using a stylesheet that outputs Scalable Vector Graphics (SVG), an XML language for generating diagrams.

Discussion and negotiation are an important aspect of collaboration. To facilitate the elaboration of the requirements all of the requirements elements support annotation with issues. A stakeholder can add an issue such as a question or problem to a requirement element to start a discussion. Users can then propose solutions to the issue and argue the merits of each with arguments supporting or conflicting with the solution. Once all the proposed solutions have been discussed, one can be chosen as the resolution. Discussions can be referred to later to understand the rationale behind a requirement.

Dealing with informality is a key aspect of general knowledge acquisition characterized by domain specific terminology and ambiguity (Reubenstein, & Waters, 1991, p. 228.) A glossary defining the key terms of the project is an important tool in combating ambiguity and sharing understanding. This is particularly important when a term is used in different ways between the customer's domain and the software engineering domain.

Requel supports a glossary with terms and definitions. Terms that have the same meaning in a project may be linked together under a canonical term. It can then replace all occurrences of the subordinate terms with the canonical term to make the requirements more consistent. The automated assistance identifies terms in the text of the requirement elements that may be ambiguous and require definition. For each term the assistant adds an issue to a requirement element that the term may need to be added to the glossary with automated resolutions that add the term to the glossary and back link to the element.

Analysis

Analysis is concerned with validating the requirements to determine how correctly the requirements identify and communicate the needs of the stakeholders. Validation of requirements involves checking that requirements are:

- Accurate – the requirements are what the stakeholders want;
- Unambiguous – all stakeholders agree on what each requirement means; each term means only one thing in the context of the requirements.
- Understandable – all stakeholders know what each requirement means;
- Consistent – requirements don't contradict each other;
- Complete – all the requirements and only the requirements are documented and they are documented to a level of precision where they can be implemented and tested.

Some analysis may be automated, particularly analysis that can be modeled in rules, such as lexical analysis related to syntax or grammar. For example “I ran to the” is obviously incomplete and missing a noun, for example “store”. Requel supports limited lexical analysis using natural language phrase structure and dependency parsing to identify ungrammatical sentences. Analysis that is subjective, such as determining if a requirement is really what the client wants, requires manual analysis by a stakeholder.

Accurate

Accuracy is impossible to detect and measure automatically because it is subjective (Reubenstein & Waters, 1991, p. 229.) The only way to measure accuracy is to have the

customer stakeholders indicate that the requirements reflects what they want. Requel makes no attempt to analyze the accuracy of the requirements.

Unambiguous

Ambiguity can occur at different levels in natural language. Words can have different meanings and it isn't always clear which sense of a word is being used when the meanings are similar, or when there are specialized domain specific uses. The structure of sentences can also suffer from ambiguity particularly around conjunctions, called coordination ambiguity (Jurafsky & Martin, 2008, p. 433), and prepositional or adverbial phrases, called attachment ambiguity (Jurafsky & Martin, 2008, p. 432.) For example in the sentence "The man sat quietly with a bottle of wine and cigar on his table" there are multiple ambiguities. The adverb "quietly" may mean the man is being silent, that he is not moving, or that he was careful when moving into the seated position. The phrase "on the table" could refer to where the man sat, where the wine and cigar were placed or where only the cigar was placed. The pronoun "his" may refer to the man or to someone else identified in an earlier sentence. The last case is known as an anaphora reference meaning a reference to something that was introduced earlier in the text (Jurafsky & Martin, 2008, p. 696.)

Words with multiple meanings can be determined using a lexical database such as WordNet (Fellbaum 1998.) For example "bank" may refer to an institution for holding money, a container for holding money, or the sloped embankment of a river. Each meaning is a distinct entity in WordNet.

Ambiguity can also be caused by vagueness. For example, the word “vehicle” is more abstract than the word “automobile.” The requirement “there must be parking for six vehicles” can be interpreted differently by different readers without constraints on the type or dimensions of the vehicle.

The generality or abstractness of a word can be determined using a lexical database such as WordNet where the words are organized in a hierarchy of hypernym-hyponym relations where hyponyms have a more specific meaning than the hypernyms. For example “car” is a hyponym of “vehicle.”

Requel uses the WordNet hypernym-hyponym relations to calculate an information content value using an algorithm by Seco, Veale and Hayes (2004.) The basic idea is that words higher in the hierarchy convey less information than words lower in the hierarchy, otherwise the lower words wouldn’t be needed. The algorithm, represented formally in Equation 1, calculates a value that gets smaller as the count of the hyponyms of the word increases. In the equation the function $\text{hypo}(c)$ returns the count of all the words subsumed by the word c and the value max_{wn} represents the count of all words in WordNet with the same part of speech. The information content of each word in a sentence is compared to a threshold, by default 0.50, and words below the threshold are identified as potentially vague.

$$ic_{wn}(c) = 1 - \frac{\log(\text{hypo}(c) + 1)}{\log(\text{max}_{wn})}$$

Equation 1: WordNet Information Content

Using a glossary to explicitly define how words are used in a project can help disambiguate words, as long as the words are used consistently, which may be hard to detect; although Gale, Church, and Yarowsky (1992) found that typically only one sense of a word is used at a time.

“If a polysemous word such as sentence appears two or more times in a well-written discourse, it is extremely likely that they will all share the same sense.”

Understandable

Measuring understanding is impossible without the stakeholder explicitly indicating that they do or do not understand a specific requirement. Like ambiguity, using linguistic analysis can help determine the complexity of statements in a requirement, which may be an indication of its ability to be understood. Requel uses a very simple complexity measure based on the depth of the phrase structure of a sentence. A more rigorous method using parsing and lexical information is left for future work.

Consistent use of language is important in making requirements understandable. For example, using a single term per concept helps avoid confusion. Detecting terms that are likely to refer to the same concept is difficult and shares a lot of the same issues as identifying ambiguity. Requel does support identifying terms manually or automatically, and then manually indicating that they represent the same concept. Once terms are linked as being the same concept the system can then replace the duplicate term with the preferred term so that the concept is consistently referred to throughout the requirements.

Consistent

Requirements shouldn't contradict each other. Logical consistency can be measured from different perspectives, for example from simple logical contradictions to more subtle issues like ordering in time. Contradiction is likely to be the easiest to automatically measure by logically modeling the requirements and detecting contradictions in the logic. Given that Requel uses natural language to define the requirements, converting them to accurate logical models is not a trivial task and it is left for future work.

Complete

There are two aspects to completeness: all the requirements are known, and all the information about a particular requirement is known. In the later case with the use of semantic information and domain knowledge it may be possible to detect an incomplete requirement. This is another task that is very difficult. Requel supports some limited semantic analysis based on verb usage patterns using VerbNet (Schuler 2005), but there is not enough data to be effective and improvements are left for future work. Identifying missing requirements may be possible in a limited fashion by identifying goals that are not represented in any use cases, or by goals without relationships to other goals.

Methods and Techniques

Nuseibeh and Easterbrook (2000, p. 4) define six classes of techniques and methods: general data gathering techniques like surveys and interviews; group or team oriented techniques like focus groups and the Joint Application Design (JAD) method; prototyping; goal and scenario-based methods such as Knowledge Acquisition in automated Specification (KAOS) and Cooperative Requirements Engineering With

Scenarios (CREWS); cognitive techniques adapted from psychology and knowledge engineering such as protocol analysis to understand how a user thinks about a task; and contextual or ethnographic techniques where the tasks of a process are observed while a user performs them.

Christel and Kang (1992, p. 16) organize techniques and methods into the general requirements engineering areas of information gathering, analysis and validation and then rate them on how well they solve key elicitation issues as well as on their maturity and prescriptiveness. A few of the methods they include that are not already mentioned are Issue-Based Information System (IBIS), Controlled Requirements Expression (CORE), Soft Systems Methodology (SSM), Planning and Design Methodology (PDM) and Quality Function Deployment (QFD).

Requel is not tied to a specific method. It supports a variety of entity types that may be applied as needed by the process in use. There may be cases where the requirement elements structure does not lend itself to specific information. Requel supports adding notes to all the requirements elements to support this kind of information.

Table 1 summarizes the influence the areas of concerns have on Requel.

Area of Concern	Influence on Requel
Acquisition	Requel supports stakeholder identification and tracking. It supports a variety of elements used to define requirements including goals, actors, stories, scenarios, and use cases. Requel allows users to create requirements elements in any order or level of detail.
Communication and Specification	Requel supports the IBIS discussion and negotiation protocol and customizable documentation using XML and XSLT.
Analysis	Requel uses natural language processing to detect potential ambiguity and complexity, and to identify

	potential glossary terms and actors.
Methods and Techniques	Requel is not tied to a specific method and supports agile methods based on user stories; goal based methods; and scenario or use case based methods.

Table 1: Influence of Areas of Concerns

Natural Language Processing

A key feature of Requel is analyzing the requirements to identify issues and make suggestions to improve their quality. As the requirements are defined in natural language Requel must process the text of the requirements to collect lexical, grammatical and semantic information used by the analysis. This section gives an overview of the tasks that are performed to collect that information.

Sentence Segmentation

A sentence segmenter takes a string of text and breaks it into a set of sentences. This is non-trivial because there are multiple punctuation marks that can end a sentence, and the marks may be used for other functions. For example, a period “.” may end a sentence, separate digits in a decimal number, or end an abbreviation. A sentence segmenter is also known as a sentence boundary detector or sentencizer.

Morphology

Most words have multiple forms such as various tenses for verbs or plurals for nouns. To simplify working with multiple form words it is convenient to identify the normalized form of a word, called the “lemma.” Jurafsky and Martin define morphology as “the study of the way words are built up from smaller meaning-bearing units.” (2008, p. 47.)

Many words, called regular, follow a standard pattern for generating the different forms. For example, the verb “walk” derives its Past Tense by appending “-ed” and Present Participle by appending “-ing.” The knowledge that describes how the spelling of a word changes in its different forms are called orthographic rules (Jurafsky & Martin, 2008, pp. 45.) Irregular words don’t follow the standard patterns. For example, the verb “be” has the forms: am, are, is, was, were, be, being, been. Luckily, the number of irregular words is fixed and small enough to store efficiently in a database.

Spelling

Spell checking is important as a precursor to lexical analysis. Words spelled incorrectly can cause problems for the parser. The wrong tag may be assigned to a word, for example marking a noun as a verb, causing the parser to structure the sentence completely wrong.

Parsing

Parsing involves splitting the sentence into syntactic elements that are then grouped in various ways. The first part of parsing is identifying the words and punctuation, known as tokenization. Then each token is tagged with its part of speech such as noun, verb, determiner, or punctuation. Syntax parsers take a sentence of words and describe the relationships between the words. There are three common types of syntactic parsers: constituent, dependency, and link.

A constituent parser, also known as a phrase structure parser, organizes the words into a phrasal tree structure with terminal components like nouns and verbs and structural components like sentences and verb, preposition, and noun phrases (Jurafsky & Martin,

2008, pp. 385-387.) Figure 1 shows an example constituent parse of the sentence “An old man drove the red car quickly” visualized with the Stanford Parser Parse Visualization Tools (available at <http://ai.stanford.edu/~rion/parsing/index.html>.)

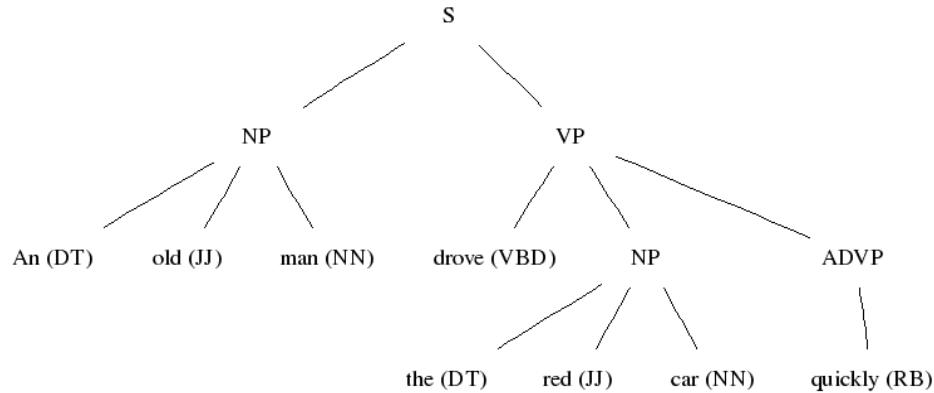


Figure 1: Example Constituent Parse

A dependency parser identifies the syntactic dependencies between the words of a sentence, for example the subject, direct object, and indirect object of a verb. Figure 2 shows a dependency parse of the same sentence as Figure 1.

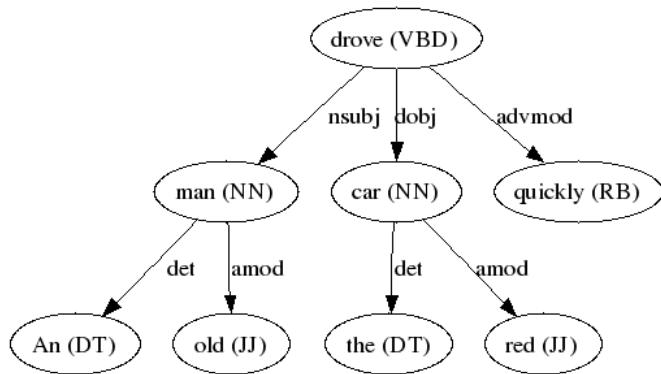


Figure 2: Example Dependency Parse

A dependency parse appears to have relationships similar to the roles assigned by a semantic role labeler, which gives the impression that it would be easy to map the dependencies to semantic roles. For example the subject becomes the agent and the indirect

object becomes the patient. Unfortunately the mapping isn't that simple, Figure 3 shows the dependency parse of the original example sentence converted to a passive form. In this case the car is the subject and the man is a prepositional object.

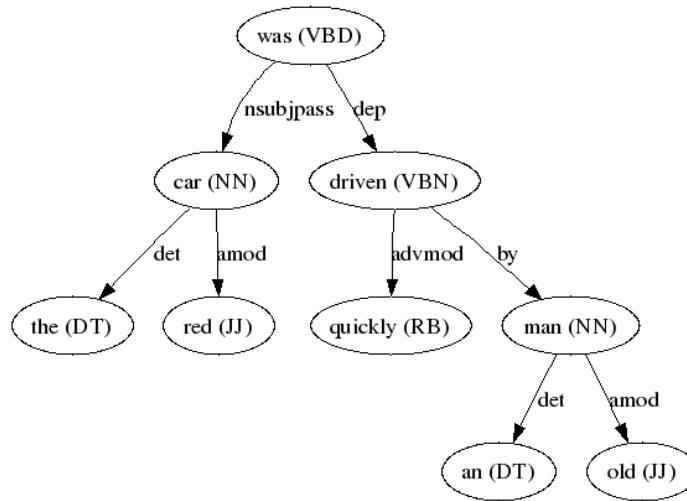


Figure 3: Example Passive Dependency Parse

Word Sense Disambiguation

Word sense disambiguation (WSD) is “the task of examining word tokens in context and determining which sense of each word is being used” (Jurafsky & Martin, 2008, p. 637.) WSD is necessary to achieve any amount of semantic understanding.

There are a few methods for performing WSD. Co-referencing is based on the idea that the different senses of a word tend to occur more frequently with specific senses of other words. By using a corpus where all the word senses are identified the relationship between senses of different words can be learned. A benefit of this approach is that it can identify relationships between words of different parts of speech. The downside of this approach is that you need a huge corpus to collect enough data to be useful. The SemCor

semantically tagged corpus identifies WordNet senses in about 350 of the Brown corpus files (Mihalcea, 1998.) Its coverage of words in WordNet that co-occur is very low.

Word similarity is based on the idea that the senses of polysemous words used together in sentences tend to be more semantically related than the other senses of the words. The semantic similarity of two words is calculated using a dictionary. There are a variety of methods for calculating word similarity.

Michael Lesk (1986) introduced the first algorithm based on this idea. In his algorithm the words used to define the multi-sense words in a sentence are compared and the senses that have the highest words in common are chosen as the correct sense. The main problem with the Lesk algorithms and its derivates is that the length of the word definitions is a key factor in its success; words with short definitions are less likely to have overlapping words (Jurafsky & Martin, 2008, p. 647.)

Another class of methods uses the hypernym-hyponym hierarchy of word relationships in a dictionary such as WordNet to calculate the similarity based on how close they are in the hierarchy (Jurafsky & Martin, 2008, p. 652-657; Simpson & Dao 2005; Banerjee 2002.) There are two predominant methods in this class; distance based methods compare the lengths of the paths between the different senses; information content based methods compare the information content of the lowest word in the hypernym-hyponym hierarchy that is an ancestor of both senses.

The main problem with semantic similarity methods is that the similarity can only be calculated for words in the same hierarchical relationship, most often the hypernym-hyponym relation, which only relates words with the same part of speech. Methods that use

a probability-based information content calculation also require a sense tagged corpus to calculate the probabilities, so the issue of word coverage becomes a problem again.

An information content calculation such as the one proposed by Seco, Veale and Hayes (2004) solve the corpus problem by calculating the information content using only the hypernym-hyponym relationships.

Semantic Role Labeling

Semantic role labeling is concerned with identifying the meaning of a sentence from the meaning of the words that compose it (Jurafsky & Martin, 2008, p. 670.) A semantic role labeler assigns semantic or thematic roles to the parts of a sentence with respect to a verb. Thematic roles are a more general class of roles appropriate for many verbs. For example the statement “I eat ice cream” can be represented using the VerbNet (Schuler, 2005) thematic roles as “verb: eat, agent: I, patient: ice cream.” Semantic roles tend to be verb specific. For example, using FrameNet (Ruppenhofer, Ellsworth, Petrucc, Johnson, & Scheffczyk, 2006) roles on the same sentence has the form “verb: eat, eater: I, eaten: ice cream.”

Figure 4 shows the semantic roles assigned to the example sentences from Figure 2 and Figure 3 generated using the semantic role labeling demo software from the Cognitive Computation Group at the Department of Computer Science, University of Illinois at Urbana-Champaign. (<http://l2r.cs.uiuc.edu/~cogcomp/srl-demo.php>) Note that the semantic parser assigns the “driver” role to the old man in both cases, unlike the syntactic relations assigned by the dependency parser in Figure 2 and Figure 3.

An	driver [A0]	the	thing in motion [A1]
old		red	
man		car	
drove	V: drive	was	
the	thing in motion [A1]	driven	V: drive
red		quickly	manner [AM-MNR]
car		by	driver [A0]
quickly	manner [AM-MNR]	an	
		old	
		man	

Figure 4: Example Semantic Role Labeling

Corpus Linguistics

A corpus isn't a process, but is important to many natural language processing tools. It is collection of texts used as an example of language use. Corpora typically contain annotated text with information related to syntax or semantics (Jurafsky & Martin, 2008, p. 130.) The Brown corpus is a million-word collection from 500 texts of different genres including newspapers, academic works, fiction, and nonfiction (Jurafsky & Martin, 2008, p. 85.) Many natural language processing tools use the Brown corpus as training data. The SemCor corpus is a subset of the Brown corpus with additional annotations identifying the sense of the verbs, nouns, adjectives and adverbs in the sentences (Mihalcea, 1998.)

Related Tools

There are many tools specific to, or contain features for requirements engineering. The International Council on Systems Engineering (INCOSE) has a Web site for the Requirements Management Tools Survey with a matrix of tools and supported features. Ian Alexander (2007) also maintains a Web site with a list of requirements related tools and summaries of their features.

Based on a review of the tool descriptions from both sites most of the tools focus on management issues such as traceability and change management, documentation and

visualization, and analysis of requirements. Borland's CaliberRM™, IBM Rational's® RequisitePro®, and Telelogic's DOORS® are the most popular commercial products that fall into the Requirements Management category.

There are very few tools in the requirements elicitation category. On Ian Alexander's site the tools "FeaturePlan" by Ryma Technology Solutions and Telelogic's "Focal Point" include features for requirements "collection" and customer involvement. The EColabor and Tuiqiao tools (Smith, 1997; Takahashi, Potts, Kumar, Ota, & Smith, 1996) work together to support elicitation and distributed collaboration.

There are a number of research projects that use processing of use cases written in a natural language like English for generating formal or semi-formal requirements models and for measuring the quality of the use cases. Tools such as Procasor and the Use Case Workshop fall into this category.

Requirements Apprentice

In the 1980's Rich and Waters (1987, 1990) among others started the Programmer's Apprentice (PA) research project at the MIT A.I. lab with the goal of understanding how expert programmers develop programs from requirements to implementation and how the tasks can be automated (Reubenstein & Waters 1991 p. 226). At that time they recognized it would be difficult to fully automate all the tasks of developing software and opted to create a system that would assist a software engineer by handling the simpler tasks and not get in the way when dealing with more complicated tasks (Reubenstein & Waters 1991 p. 226). The PA was distinct from other software engineering tools in that it was designed as an autonomous agent that cooperated with the developer through the development

environment instead of being a separate tool (Rich & Waters 1990 p. 2). Rich and Waters (1987, p. 3) identify three key areas that the PA project would entail: requirements acquisition, software design, and software implementation.

The Requirements Apprentice (RA) is one of the tools conceived as part of this research (Rich & Waters, 1986, 1987, 1990; Reubenstein & Waters 1991) and is the most relevant to this thesis project. The focus of the RA is on the transformation from informal requirements into a formal specification (Rich & Waters, 1990 p. 201.) A fundamental difference between the RA and other requirements tools of the time is that it focused on the transition from informal to formal requirements rather than the validation of requirements already in a formal representation (Reubenstein & Waters 1991 p. 227.) The authors noted that creating a formal specification was a difficult task that had been avoided by previous research. The RA is not intended to interact with end-users directly, but as a tool used by an analyst. This is mainly to avoid natural language processing issues that would arise from having to communicate with “naive” end-users in free-form text (Reubenstein & Waters 1991 p. 227.)

Requel borrows the idea that the assistant makes suggestions to the user instead of just acting on the input particularly because the assistant is processing natural language in free-form text and bound to encounter language it cannot fully process or processes incorrectly. The assistant never makes changes to the requirements without a user approving them. Users can ignore bad suggestions made by the assistant.

The RA uses a layered processing approach, which Requel also uses. Requel starts with simple processing like spell checking and syntax parsing and works its way to complex processing like word sense disambiguation and semantic role labeling. Requel

makes suggestions based on the level of processing it was successfully able to perform. For example, if the assistant was able to assign semantic roles to the text of a scenario step, then it can determine which text represents the actor and if that actor matches an actor defined in the requirements.

Requirements Assistant for Telecommunication Services (RATS)

The RATS tool was developed by Armin Eberlein as part of his PhD thesis on telecom service requirements (Eberlein, 1997.) It is conceptually similar to the Requirements Apprentice in that it is an expert system designed to assist a requirements engineer in creating a requirements specification using a knowledge base of telecom service domain models. RATS provides two type of assistance; passive assistance that validates the requirements against the domain knowledge and active assistance that gives suggestions on what the requirements engineer should do next based on the RATS method. The domain knowledge is implemented in Telos, a frame based logic system, using the ConceptBase knowledge base.

The original scope of this thesis included many ideas from the RATS system, in particular using the domain knowledge to suggest next steps, but due to time constraints proactive assistance was dropped.

UC Workbench

The UC Workbench is a tool for requirements analysts to create and validate use-cases for well formed-ness based on patterns of good quality use-cases (Ciemniewska, Jurkiewicz, Olek, & Nawrocki, 2007.) It uses Natural Language Processing (NLP) tools (specifically the Stanford dependency parser) to detect defects or “bad smells”, a term the

authors borrow from Kent Beck, defined as a surface indication of a deeper problem. In NLP terms “surface” indicates syntactic or grammatical information versus semantic information.

The authors codify the best practices described by Cockburn (2000), and Adolph, Bramble, and Cockburn (2002) into rules used to validate use cases at the specification level, use case level, and scenario step level. At the specification level they try to detect use-cases that have the same behavior in steps, only varying in the data objects being manipulated, trying to weed out duplicates. At the use-case level they validate the name, number of steps, and complexity. At the step level they validate the grammar, sentence complexity and detect missing actors.

The UC Workbench is implemented as an Eclipse Rich Client Platform (RCP) application. It has an interesting interface for editing use-cases that is similar to a WYSIWYG interface used by most modern word processors.

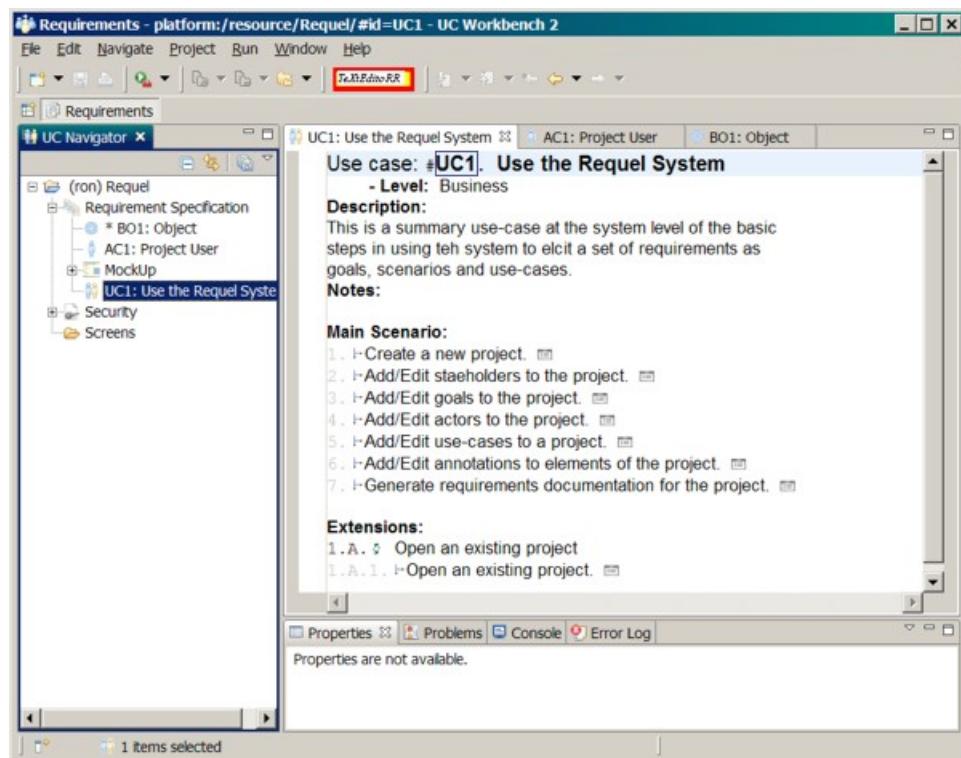


Figure 5: UC Workbench

A nice feature of the UC Workbench is that it can generate a Web-based form of the requirements document that stakeholders can navigate and review.

Requell is conceptually very similar to the UC Workbench, although the UC Workbench is intended for requirements engineers to use to create use cases and not as a collaborative tool for all stakeholders to identify and elaborate a set of requirements. The UC Workbench takes a more modest approach to natural language processing. The use cases are defined in a controlled subset of English grammar called FUSE (Nawrocki and Olek, 2005.) It expects the use case steps to have a simple structure with a subject, verb, direct object and prepositional phrase. The actor is always the subject of the sentence.

Procasor

The Procasor project is concerned with converting use cases in natural language to a more formal behavioral specification called Pro-Cases (Mencl 2004.) Vladimir Mencl runs the project at the Charles University in Prague, Czech Republic.

Procasor is similar to the UC Workbench in that it uses NLP to analyze use-cases at the step level, although it differs in how the analysis is done. It uses the Collins constituent parser to create a phrase structure tree and then analyses the structure. A limitation of using the phrase structure is that the tool requires sentences to be restricted to the form: subject - verb - direct object - preposition - indirect object (Mencl 2004.) More recently the team is working to reduce this limitation and support steps with leading conditions and compound sentences (Dražan & Mencl 2007.)

gIBIS

The gIBIS tool is not related to requirements engineering directly, but is a standalone tool that implements the IBIS method of discussion and negotiation. The goals of the tool include keeping track of the decision making process history, and having a collaborative environment for discussions. The authors of the tool make some suggestions on enhancements to the IBIS method (Conklin & Begeman, 1988.)

Requel models its discussion and negotiation features after the IBIS method with issues, positions and arguments. Requel also supports the “null” position concept, which basically indicates ignoring the issue; and picking a position as the resolution based on suggested improvements by the authors.

EColabor and Tuiqiao

Tuiqiao (Chinese for ‘elaboration’) is a single-user hypertext tool that implements the Inquiry Cycle method (Potts, Takahashi, & Anton 1994) for requirements analysis. It allows a user to manage requirements as text and scenarios, and add annotations such as questions, answers and reasons as part of a discussion about the requirements.

The interesting aspect of this tool relevant to this thesis is the collaboration through discussions as question and answer annotations and traceability of the annotations to give rationale to the requirements, which Requel adopts and mixes with the IBIS concepts. Other aspects that fell out of scope are the versioning of individual elements of the requirements document.

Feature Comparison

Table 2 summarizes and compares the key features of the related tools described above.

	Requel	UC Workbench	Procasor	Requirements Apprentice	RATS	EColabor & Tuiqiao	gIBIS
Collaborative Environment	x					x	x
Discussion & Annotations	x					x	x
Natural Language Processing	x	x	x				
Logical Modeling of Requirements				x	x		
Informal Requirements Specification	x	x				x	

Formal Requirements Specification			x	x	x		
-----------------------------------	--	--	---	---	---	--	--

Table 2: Related Tools Feature Comparison

Summary

This chapter described the key aspects of requirements engineering and how Requel addresses the related issues.

For acquisition Requel supports stakeholder identification and a variety of elements used to define requirements including goals, actors, stories, scenarios, and use cases. For communication and specification Requel supports the IBIS discussion and negotiation protocol and customizable documentation using XML and XSLT. For analysis Requel uses natural language processing to detect potential ambiguity and complexity, and to identify potential glossary terms.

Requel borrows design concepts and features from prior requirements and collaboration tools. Examples include the use of layered processing and the use of an assistant that suggests improvements, used in the Requirements Apprentice and RATS tools; the annotation and discussion concepts from the EColabor and gIBIS tools; and the natural language analysis used in the UC Workbench and Procasor tools.

Organization of this Document

The rest of this document is organized as follows. 2 defines the requirements of the Requel system as a set of stories, goals, actors and use-cases. 3 describes the design of the Requel system including the architecture, external tools used, and implementation details.

4is a comprehensive user and installation guide. 5concludes the document with what was learned and future work.

Chapter 2 System Requirements

This chapter gives a summary of the requirements for Requel as a set of goals, actors and use-cases with stories and scenarios as an example of what the requirements may look like being collected by the tool.

Goals

Goals include desired properties, constraints, features and functions that the system must support. Goals are primarily textual with associations between them. Goals have a name, which is a short one-line description of the goal, an optional text body which is a long description of the goal, and relationships to other goals that it supports or conflicts with. To help keep Requel simple goals don't have an explicit type or level. Below are a few select goals of the tool.

Goal	Description, Relationships, and Notes
Easy to use	<p>The tool must be useable by both technical and non-technical stakeholders. It must be simple for users to find and add information to the requirements.</p> <ul style="list-style-type: none">• Issue – This goal is vague and not easily testable.
Improve communication between stakeholders	<p>Requirements are fundamentally about communication and shared understanding. The customer stakeholders must describe what they need with a high level of detail so that developer stakeholders can implement it. This is a difficult task. The tool must support features that allow a dialog between the technical and non-technical stakeholders to elaborate the details so that all the stakeholders understand what will be built.</p>

Support discussion of requirements	<p>The system supports an IBIS style of discussion and negotiation of issues. Issues represent problems or questions about elements of the requirements. Issues have one or more positions that are possible solutions. Positions have zero or more arguments that support or conflict with the position as the solution to the issue.</p> <ul style="list-style-type: none"> • Supports “Improve communication between stakeholders.”
Don’t impose a process	<p>The system is useable with various requirements acquisition processes. Users can use any process that models the requirements in the available element types</p> <ul style="list-style-type: none"> • Conflicts with “Easy to use.”
Don’t impose top-down or bottom-up gathering of requirements	<p>Users are able to work at different levels and in different directions from abstract to concrete and vice-versa. For example a user can start with a functional requirement or scenario and then define the features, goals and stories that explain why a functional requirement or scenario exists. After a goal or story is defined the user may go back and modify a functional requirement based on ideas or discussions about the goals and add new ones.</p> <ul style="list-style-type: none"> • Supports “Easy to use.” • Supports “Don’t impose a process”
Remote access	The system supports users in a distributed environment. Users can access the system from remote sites as easily as if they were working locally.
Multi-user support	The system supports multiple users working concurrently on the same or different projects. If two users are working on the same data, changes by one user are not inadvertently over written by another user. Users will see changes by other users as they happen.
Requirements in natural language	<p>The system supports stories, goals, use cases and scenarios defined in natural language.</p> <ul style="list-style-type: none"> • Supports “Easy to use.” • Supports “Improve communication between stakeholders.”
Automated assistance	The system assists users by analyzing the requirements to identify issues and make suggestions for improving the requirements. The analysis identifies potential glossary terms, actors and domain objects from all of the requirements elements.

	<ul style="list-style-type: none"> • Supports “Easy to use.” • Issue – What types of “issues” does the analysis identify?
--	---

Table 3: Goals of the Requel System

Actors

Actors represent the users of the system, both human and other systems. Actors have a name, optional description, and goals to help refine the needs and constraints specific to this actor. There are two types of users of Requel: administrators and project stakeholders. The system has an assistant that is a pseudo-actor; it is part of the system, but acts like a user of the system.

Administrator

Administrators manage things like user accounts and system configuration.

Goals

- Create new accounts for users of the tool that will work on projects and/or be administrators.
- Add or change the system roles of users and their permissions. For example give a project user administrator privileges or authorization to create new projects.
- Edit user accounts to change personal information or reset their passwords.

Automated Assistant

The assistant is part of the system, but acts like a user of the system adding issues and positions to the requirements in a project based on its analysis.

Goals

- Assist users by adding issues to point out potential problems or enhancements to the requirements.

Project User

Project users use the system to create requirements and may be members of one or more projects in the system as stakeholders. Some project users may create new projects; others will only be able to work on projects when they are added by other users. Project users have project specific permissions for editing different types of requirements elements.

Goals

- Create and edit requirements on one or more projects.
- Discuss requirements by adding problems, questions and notes.
- Review open issues and add comments.
- Generate a requirements document.

Table 4 gives examples of the types of project users/stakeholders and the people that fill those roles. The people represent an example team working on a fictitious accounting system project.

Stakeholder Role	Description	Example Personas
Customer/Consumer (Business stakeholders, End-users)	These are users that are part of the customer organization that will use or have an interest in the system being specified. These users primarily define the “what and why” goals of the software and review and refine the use cases, although sometimes create them outright. Examples are end users and business management or customer proxies that act on behalf of end users or business management.	Theresa is an accountant for Bailey Pet Supply tasked with specifying the requirements for integrating the purchase order system with the accounting system using the system. Theresa has never used the requirements tool before and has no projects. Theresa will be an end-user of the resulting system. Buddy is the controller of Bailey Pet Supply; he is the sponsoring manager of the

		project with interests in, but is not a direct user of the resulting system.
Supplier/Developer	These users are part of the team that will deliver the software that meets the requirements. These users primarily review and discuss goals and use-cases to help refine them. Developers' interests are mainly concerned with getting as much detail as possible to be able to estimate the effort to build a system and the feasibility of a project given available technology.	Ron is a software developer contracted by Bailey Pet Supply to create the new purchase order system. Ron has not used the tool but is familiar with requirements engineering.
Analyst/ Requirements Engineer	The analyst actor represents the requirements analyst. This user works with the customer to review and refine goals and create use-cases. The analyst's interests are having the requirements reflect what the customer actually wants.	Jason is a requirements engineer tasked with elaborating the technical requirements for the system. Jason has lots of experience with the tool.
Project Manager	A Project Manager (PM) is a project user whose main objective is managing the team of users for one or more projects in the system. A PM may act on behalf of another stakeholder to ask or answer questions and make comments. The PM may be on the consumer or supplier team.	Rich is the project manager for the accounting system project assigned to answer questions that other team members have posted and to ask questions about requirements entered for the project.
Authority (non-user role)	Some application domains are subject to rules by consortiums or government agencies. These agencies typically will not be represented by a stakeholder user in the system.	FASB is the Financial Accounting Standards Board that sets rules on accounting practices.

Table 4: Stakeholder Roles, with Examples

Use Cases

Use cases represent functional requirements of the system in terms of how a user interacts with the system. In Requel use cases are a container that brings together simpler concepts including goals, stories, and scenarios. A use case has a name; an optional description; a primary actor that is the main participant interacting with the system; additional actors that the system may interact with to support the interaction with the primary actor; a set of goals that represent what the primary actor is trying to accomplish with the use case; a set of example stories including successful and exceptional cases; a scenario that may have alternate and exceptional paths.

Scenarios are semi-structured elements describing the interaction of an actor with the system. They are typically represented as a dialog between the actor and the system where the actor does something, the system reacts, the actor does something else, etc.

Stories are simple narratives of how a user interacts with the system. Stories can be used to describe how things work before the new system is in place, i.e. the current state of affairs, or how things will work when the new system is in place. Stories can describe negative cases, such as problems with the current system, or how the new system should behave in a bad situation. Stories are useful as rationale for understanding the scenario of the use case.

This section defines some of the use cases that Requel supports in varying levels of detail in the same way that it would be used to define requirements.

Create or edit a user account

An administrator creates user accounts; assigns roles such as administrator or project user; and grants permission for project users, such as the permission to create projects. Existing user accounts can be edited to change any personal information, roles and permissions.

Primary Actor: Administrator

Goals

- Create accounts for new users of the tool that will work on projects and/or be administrators.
- Add or change the system roles of users and their permissions. For example give a project user administrator privileges or authorization to create new projects.
- Edit a user's account to change personal information or reset their password.

Example Success Story

Eric gets an email from Rich requesting a user account for him. Rich needs to use the system as a project user and be able to create projects. Eric logs into the system, and because he is an administrator the system gives him the option to create new users. Eric chooses to create a new user account for Rich and the system displays an interface for entering a name, email address, phone number, username, password and the option to grant administrator and project user roles, and permission to create projects. Eric enters “rich” for the username and “rich123” as a temporary password. He adds the project user role and grants Rich permission to create new projects. Eric sends Rich an email with his username and password and tells rich to change his password when he first logs in.

Example Success Story

Eric gets a call from Ron that he forgot his password and needs it reset. Eric logs into the system and chooses “ron” from the list of users. In the user editing interface Eric enters a new password and saves it. Eric tells Ron to use “ron123” for his password and to change it when he next logs in.

Login to the system

All users must be authenticated by the system with a username and password to use it.

Primary Actor: Any User

Example Success Story

Theresa connects to the requirements tool via a Web browser from a link in an email message sent by Dave the system administrator. The system doesn’t detect an existing user session and presents the login screen. Theresa enters her username and password supplied in the email. The system verifies the username and password combination is correct and displays the user interface for project users containing the “Purchase Order System” project that Theresa is assigned to.

Example Exception Story

Ron connects to the requirements tool via a Web browser. The system doesn’t detect an existing user session and presents the login screen. Ron enters his username and an incorrect password. The system displays an error message that the username and password combination do not match a valid user. Ron calls Eric the I.T. guy and asks to reset his password.

Scenario

1. The user accesses the system through a Web browser.
2. The system determines the user doesn't have an active session and displays a login interface prompting the user for a username and password.
3. The user enters his or her username and passwords and submits them back to the system.
4. The system verifies the user supplied a correct username and password combination and displays the appropriate interface to the user based on his or her roles.
 - a. (exception) The system informs the user the username and password combination are not valid and redisplays the login interface.

Create a new project

A project is a container for managing a set of requirements for a particular system.

A project has a name; a description that can be used as an overview or scope; and a customer. The project name plus customer name must be unique in the system. Only authorized users can create projects.

Primary Actor: Project User

Example Success Story

Rich logs in to the system and it displays a list of his active projects and the option to create a new project. Rich chooses to create a new project and the system displays the “New Project” interface. Rich enters “Purchase Order System” for the project name, “Bailey Pet Supply” for the customer name, and leaves the description blank. The system creates the project and adds it to Rich’s list of active projects.

Example Success Story

Rich logs in to the system and it displays a list of his active projects and the option to create a new project. Rich chooses to create a new project and the system displays the “New Project” interface. Rich enters “Accounting System” for the project name, “Bailey

Pet Supply” for the customer name, and leaves the description blank. The system detects that a project with the name “Accounting System” already exists for the customer “Bailey Pet Supply” and displays a message that a project with that name already exists. Rich changes the name to “Purchase Order System”. The system creates the project and adds it to Rich’s list of active projects.

Scenario

1. (Pre-Condition) The user is authorized to create a new project.
5. The user chooses to create a new project.
6. The system displays the “New Project” interface with inputs for a name, description and customer. The customer input allows entering a new customer name or selecting an existing customer.
7. The user enters a project name, customer name and optionally a description.
 - a. (alternative) The user selects an existing customer.
8. The system verifies the project name plus customer name doesn’t already exist in the system, creates the project and adds it to the users list of current projects.
 - a. (exception) The system determines the combination of project name plus customer name is not unique and displays a message that the project name must be unique for a customer.
 - b. The user changes the name of the project to a name not already in use.
 - i. (alternative) The user changes the name of the customer to another customer name.

Create or edit a stakeholder

A project will have one or more stakeholders. Stakeholders may be users of the system or non-users that represent authorities such as a government regulatory agency. User stakeholders are granted permissions to edit various requirement elements and may have goals that represent that user’s interests. Non-user stakeholders will have goals that represent concerns or rules dictated by that authority.

Primary Actor: Project User

Example Success Story

After Rich creates the “Purchase Order System” project he needs to add Theresa as a stakeholder so she can add user requirements to the project. Rich selects the stakeholders interface for the project and chooses to add a new stakeholder. The system displays the “New Stakeholder” interface that includes a selector for existing system users and a list of permissions to grant to the user. Rich selects Theresa’s username and grants permission for her to create actors, stories, goals, and annotate any requirement element. The system creates the stakeholder and adds it to the stakeholders for the project. The next time Theresa logs in she sees the “Purchase Order System” in her list of current projects.

Example Success Story

Rich needs to make sure that the “Purchase Order System” meets the rules set by the Financial Accounting Standards Board. Rich decides that to distinguish these requirements from the user goals he will create a non-user stakeholder and add the rules as goals of that stakeholder. Rich selects the stakeholders interface for the project and chooses to add a new stakeholder. The system displays the “New Stakeholder” interface that includes an input field for entering the name of a non-user stakeholder. Rich enters “Financial Accounting Standards Board” for the name and saves. The system creates the stakeholder and adds it to the project. Rich edits the stakeholder and chooses the option to create new goals to add to the stakeholder. As Rich adds goals they appear on the stakeholder interface in the goal section. On the goal interface the stakeholder appears in the “referenced by” section.

Example Exception Story

Theresa wants to add her boss Buddy to the system so he can review the status of the project and make decisions related to the business rules the system should honor.

Theresa logs into the system and selects the stakeholders from the “Purchase Order System” project. The system displays the list of existing stakeholders but does not give her the option to add a new stakeholder because Rich didn’t grant her permission to create stakeholders.

Scenario

1. (Pre-Condition) The user is authorized to edit stakeholders in the current project.
9. The user chooses to create a new stakeholder.
 - a. (alternative) the user chooses to edit an existing stakeholder.
10. The system displays the edit stakeholder interface with fields for creating or editing user and non-user stakeholders.
11. The user selects an existing system user and chooses various permissions to grant to the user for the different requirements elements.
 - a. (alternative) The user enters a name for a non-user stakeholder.
12. The system verifies that a stakeholder doesn’t already exist for the user in the project, creates it, and adds it to the project.
 - a. (exception) The system detects that the user is already a stakeholder in the project and displays a message to the user that the selected user is already a stakeholder in the project.
 - b. (alternative) The system verifies that a non-user stakeholder doesn’t already exist for the supplied name in the project, creates it, and adds it to the project.
 - i. (exception) The system detects that a non-user stakeholder already exists in the project for the supplied name and displays a message to the user that the name is in use.

Create or edit a goal

Goals represent desired qualities, constraints, features and functions of the system.

Goals have a name; optional detailed description; and a set of relationships to other goals indicating a goal supports or conflicts with another goal.

Primary Actor: Project User

Example Success Story

Rich needs to add goals to the “Financial Accounting Standards Board” (FASB) stakeholder to indicate accounting practices the system must adhere to. Rich navigates to the FASB stakeholder and chooses to add a new goal. The system displays the “New Goal” interface with fields for a goal name and detailed description. Rich enters “Support FASB 133: Accounting for Derivative Instruments and Hedging Activities” for the goal name and the details of what the ruling requires in the description and saves it. The system creates the goal and adds it to the project and the FASB stakeholder’s set of goals, and returns Rich to the stakeholder edit interface. The system starts analysis of the goal in the background.

Scenario

1. (pre-condition) The user is authorized to edit goals in the current project.
13. The user chooses to create a new goal.
 - a. (alternative) the user chooses to edit an existing goal.
14. The system displays the edit goal interface with fields for the goal name and optional detailed description.
15. The user enters the name and detailed description.
16. The system verifies the goal name is unique to the project, creates the goal, adds it to the project, and starts analysis of the goal.
 - a. (exception) The system determines the goal name is already in use by another goal and displays a message to the user that the goal name is in use and must be changed.

Add or edit a relationship between two goals

The system supports relationships between goals such as one goal supporting or conflicting with another goal.

Primary Actor: Project User

Example Success Story

Theresa creates a new goal in the project “The system provides help to users based on the frequency that they use the system.” After creating the goal the system displays the edit goal interface with the option to add relationships from this goal to other goals. She decides that the new goal supports the goal “The system is easy to use for new users” and chooses to add a relationship between the two goals. The system opens the goal relationship editor with the new goal in the “from” field and a list of all the other goals in the “to” field. Theresa searches through the list and finds the goal she wants and selects it. She then selects “supports” from the list of available relation types and saves. The relation appears in the new goal’s table of relations to other goals.

Create or edit a use case

Use cases define the behavior of the system. A use case is a composite structure with a name, description, scenario and references to actors, goals, and stories. The name of the use case typically implies the primary goal of the primary actor, for example “create a use case.”

Primary Actor: Project User

Example Success Story

After a meeting with Theresa, Jason wants to create a use case to describe the process that she follows to add a purchase order to the system. Jason logs into the

requirements tool and selects the “Purchase Order Integration” project. He selects the use cases of the project and because he is authorized to add use cases, the system gives him the option to add a new one. Jason chooses to add a new use case and the system prompts him for a name, description, and primary actor. Jason enters “Create a purchase order” for the name and a short description of what a purchase order is for the description. Jason scans the list of actors and chooses “Purchase Clerk”. In the scenario section of the use case editor Jason sees the options to add a step or add a scenario, and he chooses to add a step. The system adds input fields to select the type of step; an empty text field for the description; the option to delete the step; and the option to add a sub-step. Jason adds a few steps and saves the use case.

Scenario

1. (Pre-Condition) The user is authorized to edit use cases in the current project.
17. The user chooses to create a new use case.
 - a. (alternative) the user chooses to edit an existing use case.
18. The system displays the edit use case interface with fields for the use case name, description, primary actor, and scenario.
19. The user enters a name and description, and selects the primary actor from a list of actors in the project.
20. (optional) The user edits the scenario (see the scenario use case.)
21. The system verifies the use case name is unique to the project, creates it, adds it to the project, and starts analyzing it.
 - a. (exception) The system determines the use case name is already in use by another use case and displays a message to the user that the name is in use by another user case and must be changed.

Create or edit a scenario

Scenarios are semi-structured elements representing a dialog between the actor and the system. Scenarios are composed of an ordered list of steps where a step may be a single

action or a reference to another scenario with its own set of steps. Both steps and scenarios have a type: primary, pre-condition, alternative, exception, and optional. Where primary means it is part of the primary flow of an interaction; alternative means it is an alternative but successful interaction; exception indicates a problem or error condition; and optional means it is a successful interaction, but one that is not required to take place. The pre-condition type isn't really a step, but indicates a condition that must be met for the scenario to start or continue.

Primary Actor: Project User

Example Success Story

Ron needs to create a use case describing how users will log into the system and what they see when they login. Ron logs into the requirements tool and chooses the use cases under the “Purchase Order Integration” project. Ron chooses to add a new use case and the system displays the use case interface. After filling in the use case details Ron starts working on the scenario. He chooses to add a step and the system displays an input to select the type and enter text. Ron enters “The system displays a login screen with a username and password.” He then chooses to add another step and enters “The user enters a username and password and submits.” Finally he adds a step that “the system verifies the username and password are valid and displays the main page.”

Ron realizes that he should start the scenario with a user action. The “Purchase Order Integration” system will be Web-based so Ron adds a new step “The user connects to the purchase order system with a Web browser.” The new step is at the bottom so Ron moves it to the top of the scenario and the system reorders the other steps below it.

Example Success Story

As Ron reviews the login use case for the “Purchase Order Integration” project he realizes that there isn’t any way of handling the case of a user that forgets their password. Ron adds a story describing what should happen and then edits the main scenario. The system lists the existing steps. Ron chooses the exception step where the system determines the username and password don’t match an expected pair and changes it to include that the system displays a “forgot password” option to the redisplayed login interface. Ron then chooses to add an alternative sub-step to the exception step that the user invokes the “forgot password” action. He then adds a series of steps describe the process for recovering from a forgotten password.

Scenario

1. (Pre-Condition) The user is authorized to edit scenarios in the current project.
22. The user chooses to create a new scenario.
 - a. (alternative) the user chooses to edit an existing scenario.
 - b. (alternative) the user chooses to create or edit a use case.
23. The system displays the edit scenario interface with fields for the name, type and sub-steps.
24. The user enters the name and chooses a type.
25. (optional) The user edits the steps of the scenario
 - a. (optional) The user adds a new step.
 - i. The system adds input components for the type and text after the last step in the scenario.
 - ii. The user selects a scenario from the list.
 - iii. The system adds the scenario as the last step in the original scenario.
 - c. (optional) The user moves a step from its current position to a new position in the scenario.

- i. The system inserts the step in the new location moving the original step in that location and all steps below that location down by one location.
- d. (optional) The user removes a step from the scenario.
 - i. The system removes the step from the list and moves all steps that followed the removed step up by one location.
- 26. The system verifies the scenario name is unique to the project, creates it, adds it to the project, and starts analyzing it.
 - a. (exception) The system determines the scenario name is already in use by another scenario and displays a message to the user that the name is in use and must be changed.

Annotate a requirements element with a note or issue

The system supports adding annotations such as notes that don't support discussion or issues that support discussion and resolution. Annotations can be added directly to a project, or to elements of the project such as goals, use cases, actors, scenarios, or steps in a scenario. Notes only have a text description; issues have a description, zero or more positions that represent potential solutions, and may have a "resolved by" position as the solution.

Primary Actor: Project User

Example Success Story

As Theresa defines the goal for reporting quarterly reports in the "Purchase Order Integration" project, she is not sure if old versions of the reports need to be kept for historical purposes. Theresa chooses to add an issue to the goal and the system displays the "New Issue" interface with a field to enter the text of the issue and an input to select if the issue needs to be resolved. In the text field Theresa enters "Does the system need to keep old versions of the report?" and saves. The system creates the issue and adds it to the goal. The issue also shows up in the open issues interface.

Example Success Story

Ron adds a goal to the “Purchase Order Integration” project with the name “Users can saerch for purchase orders by customer name” and saves it. The system invokes the assistant to analyze the goal. The assistant finds the word “saerch” that appears to be spelled incorrectly or is a word the assistant doesn’t know. The assistant adds an issue to the goal that the word “saerch” in the goal name is unknown to the assistant and may be a spelling error. The assistant adds positions to the issue to ignore the word, add the word to the dictionary, and with various words known to the assistant that are spelled similarly to the unknown word, such as “search.”

Add a position to an issue

Positions represent potential solutions to an issue. A position has a text description and a set of arguments.

Primary Actor: Project User

Example Success Story

Buddy is reviewing the open issues of the “Purchase Order Integration” system and finds the issue “Does the system need to keep old versions of the report?” that Theresa added. Buddy opens the issue and sees no positions. Buddy chooses to add a position and the system opens the “New Position” interface with inputs for entering the position text and adding arguments. Buddy enters “Yes, we may need to send out old versions of reports. If a purchase order is changed that impacts how the report looks, we need a copy of the original one” and saves the position. The position appears on the issue editor in the list of positions.

Example Success Story

Jason is reviewing the open issues of the “Purchase Order Integration” system and sees the position and argument that Buddy added to the issue “Does the system need to keep old versions of the report?” Jason edits the issue and adds a new position with the text “Save old versions of reports outside the system” and saves. The new position appears on the issue in the list of positions.

Add an argument to a position

Arguments are reasons why a particular position of an issue should or should not be chosen as the solution to an issue. Arguments have text description and a support level with values from “strongly against” to “strongly for.” Users can examine the arguments for the different positions to help make a decision of which position is most appropriate.

Primary Actor: Project User

Example Success Story

After Buddy creates a position for the issue “Does the system need to keep old versions of the report?”, he edits the position and chooses to add an argument. The system displays the “New Argument” interface with inputs for entering the argument text and selecting a support level. Buddy enters the text “This is required by the standard accounting practices”, chooses “Strongly For” as the support level, and saves. Buddy then goes and edits the position “Save old versions of reports outside the system” that Jason added and adds an argument with the text “It is easier to find documents if the system keeps track of them”, chooses “For” as the support level, and saves.

Example Success Story

Jason is reviewing the open issues of the “Purchase Order Integration” system and sees the position and argument that Buddy added to the issue “Does the system need to keep old versions of the report?” Jason edits the position and adds a new argument with the text “Saving the old versions of the report will add three days to the schedule”, selects “Neutral” for the support level, and saves.

Select a Position as the resolution to an issue

Issues that are marked as “must be resolved” remain in the open issues list until a position is chosen as the resolution.

Primary Actor: Project User

Example Success Story

After a stakeholder meeting where it was decided that the system will keep track of old versions of reports, Rich finds the issue “Does the system need to keep old versions of the report?” in the project open issues interface and edits it. In the issue editor Rich chooses to resolve the issue using the position that Buddy entered. The issue editor closes and the issue is removed from the open issues interface. Rich opens the original goal of the issue and sees that the issue is marked as resolved by Rich with the text of the position used to resolve it.

Example Success Story

Ron is reviewing the open issues of the “Purchase Order Integration” and sees an issue with the text “The phrase ‘purchase order’ is a potential glossary term, actor, or domain object/property” assigned to a goal and two stories. Ron chooses to edit the issue

and sees three issues: “Ignore this phrase.”, “Add ‘purchase order’ as an actor to the project”, and “Add ‘purchase order’ to the project glossary.” Ron resolves the issue with the position to add the text to the project glossary. The issue is closed and removed from the open issues interface. Ron opens the project glossary and sees the glossary term ‘purchase order’ and chooses to edit it. In the glossary term editor Ron sees the goal and two stories as “referrers” to the term. Ron opens one of the stories and sees the term listed under the terms referred to by the story.

Add or edit a term in the project glossary

Each project has a glossary of “significant” terms used in the text of the requirements elements. A term has the text of the term, such as an abbreviation, acronym, word, or phrase; a definition; and optionally a canonical term, which is the preferred term if there are multiple terms for the same concept.

Example Success Story

As Rich works on the requirements he determines that entering “Financial Accounting Standards Board” all over the place is tedious and makes the text longer. He decides that using the acronym FASB will be easier to type and read, but may not be known to non-accounting stakeholders so he should put it in the glossary. Rich selects the project glossary and chooses to add a new term. The system displays the “New Term” interface with inputs for the term, its description, and an option to select the canonical term. Rich enters “FASB” for the term and “The Financial Accounting Standards Board sets rules on accounting practices.” For the description and leaves the canonical term empty.

The system creates the term and starts analysis that looks for uses of “FASB” in the text of requirements and creates a reference back to the term in the glossary.

Rich then thinks that “Financial Accounting Standards Board” is already used in a bunch of places and he should change them for consistency. Instead of manually looking for all the places where the text occurs and editing each to replace it with “FASB”, he creates another term, this time putting “Financial Accounting Standards Board” for the term and selecting “FASB” as the canonical term. The system creates the term and starts analysis to find all uses of it.

After the analysis completes, Rich selects the option to replace the term “Financial Accounting Standards Board” with the canonical term “FASB”. The system updates the text of all the requirements elements that refer to the original term and replaces the text “Financial Accounting Standards Board” with “FASB”.

Example Success Story

Ron goes back to the “login scenario” he created earlier and sees the assistant added a bunch of issues related to terms. For example, “The text ‘a web browser’ is a potential glossary term.” Ron opens the issue and sees two positions: “do nothing”, and “Add ‘a web browser’ to the project glossary.” Ron chooses the “resolve” action for the second option. Back in the edit scenario interface the issue appears as resolved and in the list of referred terms “a web browser” appears. Ron chooses to edit the term and the system displays the “Edit Term” interface. He sees the “login scenario” in the referring entities list. Ron fills in the empty description and saves the term. The system updates the term, but because Ron

didn't change the name of the term, it doesn't start analysis to find references to the term in the other requirements, which happened when the term was first created.

Primary Actor: Project User

Goals

- Define terminology in an unambiguous way so that all stakeholders can understand it.

Link two glossary entries

Glossary entries may be cross-referenced to indicate synonyms or aliases. One term will be identified as the “canonical” term, which is the preferred term for the concept. For example, users may refer to “the system” in the requirements using the terms “the tool”, and “the application.” These terms can be edited in the glossary to set “the system” as the canonical term. Requel can then automatically replace the use of the alternate terms in the requirements’ text with the canonical term.

Primary Actor: Project User

Goals

- Identify the preferred term in the glossary for synonymous terms.
- Make the requirements more consistent by referring to a concept with a single term.

Example Success Story

As Ron reviews the glossary he notices that there are entries “The system” and “The application.” Ron reviews how the terms are used by looking at the requirement elements referenced by the terms and sees that the term “the system” is used mostly in the goals and scenarios, and the term “The application” is used mostly in the stories. Ron decides the term “The system” is preferred term and edits the term “The application” to make the term “The system” the canonical term and chooses the option to replace the term

“The application” with the canonical term. Requel updates the text of all the requirements elements that refer to the original term and replaces the text “The application” with the text “The system.”

Generate a document for a project

Users can generate project documentation by executing a report generator. Report generators hold an XSLT that is applied to an XML version of the requirements to generate a document. Each project can have multiple generators for different types of documents or reports.

Primary Actor: Project User

Example Success Story

Rich has a meeting with Buddy to review the requirements of the “Purchase Order Integration” system, but will not have access to the system for the meeting so he wants to print out a copy of the requirements. Rich logs into the system and selects the project. He navigates to the documents section and chooses the full requirements document. The system generates the document and opens it in a new Web browser window. Rich prints out the document and heads to the meeting.

Chapter 3 Design and Implementation

This chapter covers the implementation of Requel from the high level architecture to the component level details including design patterns and the technologies used.

Architecture

This section describes the architecture of Requel in terms of the patterns used and the rationale behind the choices made.

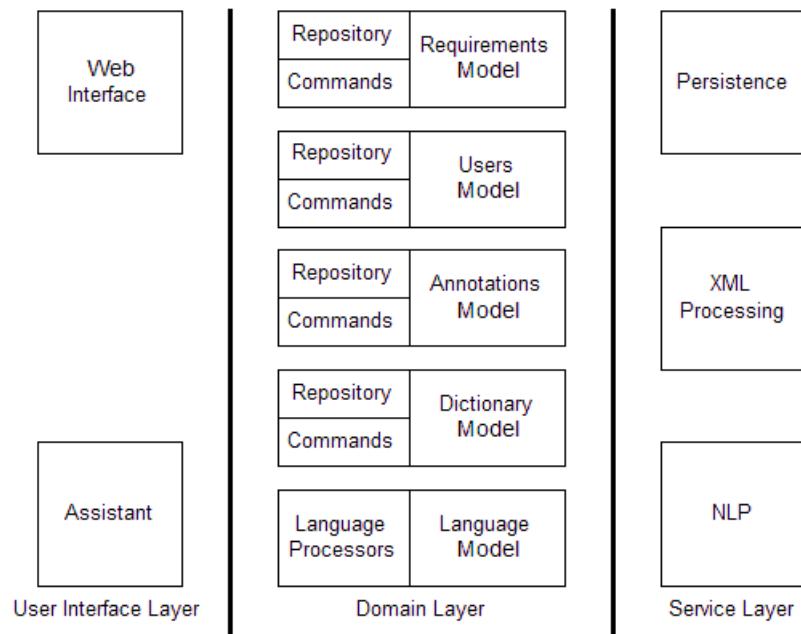


Figure 6: Architecture Layers

At the highest level of abstraction Requel uses a layered architecture (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 31-51) with three layers: user interface, domain, and services as shown in Figure 6. The layered architecture was chosen primarily

to insulate the core functionality from problems with the technology choices in the user interface and supporting services such as persistence.

User Interface Layer

This section describes the two components in the user interface layer: the Web interface for human users to access the system, and the assistant that analyzes the requirements.

Web Interface Component

The Web interface is structured primarily using the document-view variant of the model-view-controller architectural pattern (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 125-143.) The model is embodied in the interface to the domain layer and the objects that represent requirements and users. The screens and panels for displaying and manipulating the requirements represent the combined view-controller aspect of the pattern.

This pattern was chosen primarily for the separation of concerns between the user interface and the underlying domain model. There were multiple user interface frameworks under consideration and there was the potential that the selected one may have to be swapped out. Using this pattern mitigated the risk to only impacting the user interface.

Table 5 lists the consequences of the MVC architecture pattern from Pattern-Oriented Software Architecture (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 141-143) relevant to the Requel system and how it impacted the implementation.

Consequence	Impact
Multiple views of the same model.	The interconnectedness of the domain model

	requires that objects appear in different views in different ways. For example a goal appears in the goal navigator, goal selector, cross-reference tables for most of the other requirement elements, cross-reference tables for annotations and in an editor.
Synchronized views.	The objects in the domain model are highly intertwined with references between them so that one object may appear in multiple views such as navigation screens and cross-reference tables on the editors of other objects. When an object is updated, all the views of the object are updated immediately resulting in a good user experience.
Framework potential.	The Requel system uses the Echo2 framework, which is MVC based. As part of the Requel system a high-level panel and event framework was created on top of the Echo2 framework that made the development of the different screens easier.
Increased complexity.	While the panel and event framework increased productivity during development of the requirements specific panels, the framework itself is very complex and it took significant effort getting the panel management and event dispatching functionality working properly.
Potential for excessive number of updates.	This isn't a significant issue for Requel as there typically aren't that many open views at a time and propagation of changes only occur when a user initiates an action that persists a change, such as a save, add, remove, or delete.
Intimate connection between view and controller.	At the panel level this is very true. Each panel is a combination of the view and controller.
Close coupling of views and controllers to a model.	This is true at the panel level, for example adding, changing or removing a property on the Goal interface will most likely require a change to the views and controllers.

Table 5: Consequences of the MVC Pattern

Assistant Component

The automated assistant behaves just like the human users of the system; it reviews the requirements and creates notes and issues as needed. Given its behavior it makes sense

to have the assistant interact with the domain layer the same way the Web interface interacts with it.

The assistant component follows the Whole-Part pattern (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 225-242) where the assistant is composed of many assistant parts that support a specific type of analysis. The assistant uses a Façade (Gamma, Helm, Johnson, and Vlissides, 1995, pp. 185-193) to orchestrate the actions of the assistant parts. Figure 7 shows the whole-part relationship between the assistant façade and the specialized assistants. The project, use case, and scenario assistants are themselves composites that use other assistants to analyze the components of the corresponding requirement elements.

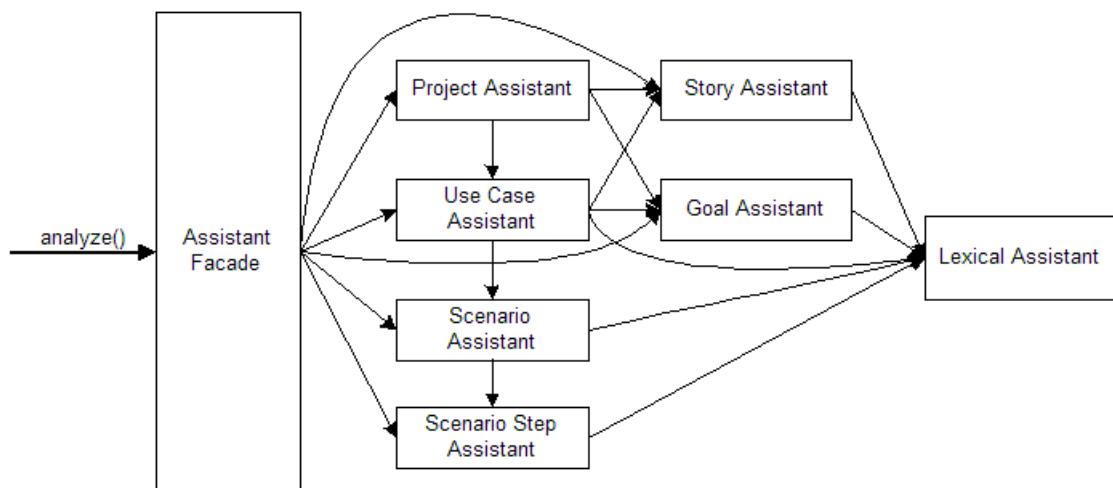


Figure 7: Assistant Architecture

Domain Layer

The domain layer is concerned with the requirements and supporting models and the actions that manipulate them. The users, requirements, and annotations components depicted in Figure 6 are an attempt to modularize the domain into logical areas of concern. The dictionary and language components insulate the assistant from the underlying natural

language processing tools and lexical data. Only the assistant component uses the dictionary and language components.

Domain Oriented Components

The requirements, users, annotations, and dictionary components share a similar internal structure with domain objects, a repository for finding specific objects, and a set of commands for manipulating the objects. The architecture of these components is based on the Domain Model, Repository, and Command Processor patterns.

The Domain Model pattern (Fowler, 2002, pp. 116-124) is the primary pattern dictating the structure of this layer. The requirements are modeled using the objects and relationships identified in the problem domain model. This pattern was chosen to support the rich user interface and automated analysis. The requirements elements have a lot of interdependencies that would be difficult to manage using a database record oriented approach. The downside to this approach is the complexity of the data management having to map the objects to and from the database, which is mitigated by the use of an object-relational mapping interface like the Java Persistence API.

The Repository pattern (Fowler, 2002, pp. 322-327) is used for hiding the details of the underlying persistence and querying mechanism. Each component has a repository interface for accessing the objects in that component.

The Command Processor pattern (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 277-290) is used to structure the actions on the object model into discrete units and provide a gateway between the user interface and domain layer. The primary concern for using this pattern is to isolate transactions from the user interface layer. The

user interface layer components create one or more commands and pass them to the domain layer through the command processor; the commands are processed in a transaction and returned to the user interface layer components back through the command processor.

Language Component

The language component is concerned with processing natural language into constituent parts and adding lexical and semantic information to convey the meaning of the original text in a form suitable for analysis. There are a variety of processes and tools that perform different aspects of the final solution from parsing, to word sense disambiguation, to semantic role labeling. The processes are independent, but some processes rely on the information identified by other processes. For example word sense disambiguation relies on the words and parts of speech identified by the parser.

The Blackboard pattern (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, pp. 71-290) has been used by various artificial intelligence applications, including language processing. It is ideal for the situation where different tools work independently to solve different parts of a problem. Figure 8 shows the architecture of the language component structured using the blackboard pattern.

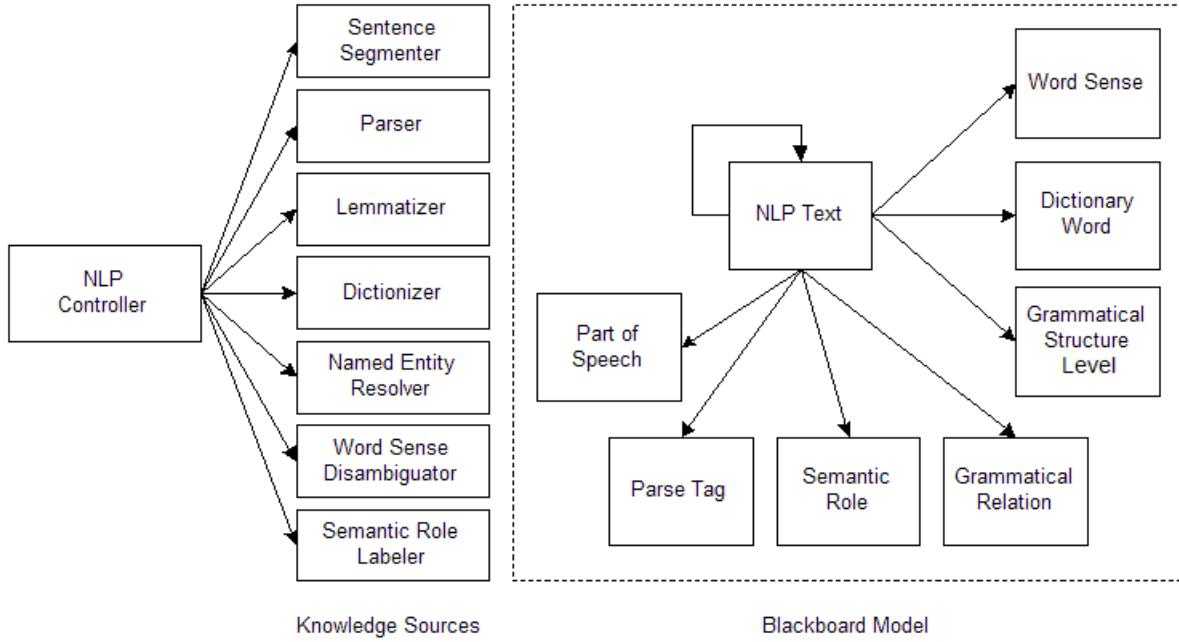


Figure 8: Language Component Architecture

The blackboard model represents the text being processed in a standardized form that is read and written to by the different knowledge sources. The NLP Text component is the primary data structure used to represent text from the paragraph level to the word level. The other components represent different pieces of knowledge attached to the NLP Text elements. For example the Part of Speech component identifies the part of speech of a word. The Parse Tag element identifies the constituent type of words, phrases, clauses, and sentences.

The knowledge sources represent the different processes applied to the text. Some are adapters to external tools that translate between the blackboard model and the tool specific representation. For example, different parsers use different sets of tags, typically depending on the corpus used to train or test them. The parser component will have to translate the tags to the ones used in the blackboard model.

The NLP Controller coordinates the execution of the knowledge sources. One of the consequences of the blackboard pattern is a lack of support for parallelism (Buschmann, Meunier, Rohnert, Sommerlad, and Stal, 1996, p. 94.) The controller must ensure access by the knowledge sources to the blackboard model is synchronized.

Service Layer

The service layer contains the components integrated with Requel to support the use cases implemented in the domain layer. The primary components in this layer are for natural language processing, XML processing used for import, export and document generation, and persistence of the domain models. Given that the service components are for the most part unrelated external tools, there is no specification for the architecture of them.

Problem Domain Model

This section defines the domain objects identified from the requirements and implemented in the domain layer components.

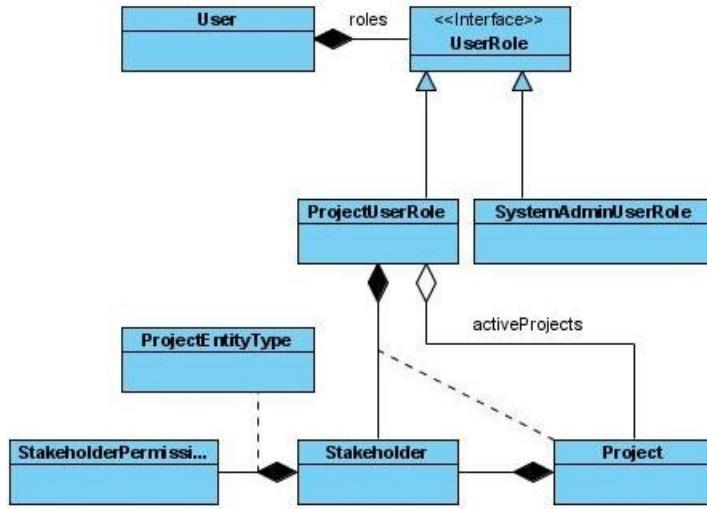


Figure 9: Users, Roles and Stakeholders

User

A user represents the users of the Requel system including project users and system administrators. It is conceivable that a user may have multiple roles in the system, such as system administrator and project user.

Username	A system-wide unique text identifier for the user.
Password	A string used to authenticate a user. This must be kept secret.
Name	The user's real name useful for identifying the user in documents or reports generated by the system.
Email Address	The user's email address.
Phone Number	The user's primary contact number. This may be included in generated documents or presented to other users of the system for contacting this user.
User Roles	The system level roles that the user has permission to use, such as project user, and system administrator.

Table 6 User Properties

Project User Role

The project user role represents a user's general project information and permissions. This is different from a user's project-specific "stakeholder" information, such as their role, team and permissions on a specific project.

User	The owning user. This is needed because the role contains user specific information.
Active Projects	A set of projects that this user has available.
User Role Permissions	A project user may have permission to create new projects.

Table 7 Project User Role Properties

System Admin User Role

The admin user role represents the system admin actor from the use cases. A user with this role has authority to create and edit other users. The role has no properties.

Stakeholder

Stakeholders are entities such as a person or organization with an interest and possible impact on a particular project. Stakeholders have goals that a project should satisfy. A stakeholder is specific to a project. All users of the system are stakeholders, although not all stakeholders are users. The role a stakeholder plays and their goals will vary from project to project.

User	If this is a user stakeholder, this references that user.
Name	If this is a non-user stakeholder, the name of the stakeholder.
Project	The project associated to the stakeholder.
Goals	Goals representing the interests of the stakeholder.
Permissions	Permissions the user has for accessing the requirement elements.

Table 8 Stakeholder Properties

Stakeholder Permission

Stakeholders are assigned operation level permissions such as grant, edit, or delete for the different types of requirement elements.

Entity Type	The type of requirement element the permission applies to. For example goal, story, actor, etc.
Permission Type	Grant, edit, or delete.

Table 9 Stakeholder Permission Properties

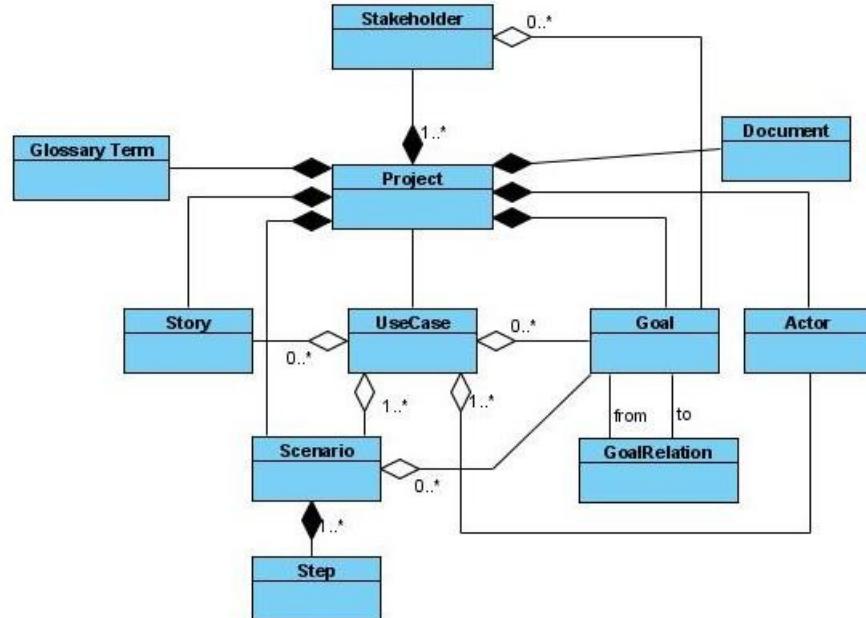


Figure 10: Project Elements

Project

A Project is a container for all the objects that make up a set of requirements and the ancillary objects created while eliciting the requirements, such as glossary terms, notes, and issues.

Name	A customer unique readable identifier for the project.
------	--

Customer	The organization the requirements are for. This is mainly for organizing projects.
Description	A summary or overview of the project.
Stakeholders	The stakeholders of the project
Goals	The goals of the project.
Stories	The stories of the project.
Actors	The actors of the project.
Scenarios	The scenarios of the project.
Annotations	Notes and issues of the project not specific to other elements of the project.
Glossary Terms	A glossary of significant terms of the project.
Documents	The XSLT stylesheets for generating different documents.

Table 10 Project Properties

Goal

Goals include desired properties, constraints, features and functions. Goals are primarily textual with relationships between them indicating support or conflicts.

Name	A goal has a short project unique name to make it easy find in a search and as a label in documents for cross-referencing.
Text	The text of the goal.
Relationships	The relationships this goal has with other goals.

Table 11 Goal Properties

Goal Relation

Goals may have relationships with other goals. Each relationship is directional from one goal to another. A relationship will indicate that a goal either supports or conflicts with the target goal.

From Goal	The goal that has the relationship with the “To Goal.”
To Goal	The goal that is the target of the relationship.
Relation Type	Specifies if the “From Goal” supports or conflicts with the “To Goal.”

Table 12 Goal Relation Properties

Story

Stories are descriptions of interactions between users and the system. They may describe desired interactions or how the system handles error conditions.

Name	A story has a short project unique name to make it easy find in a search and as a label in documents for cross-referencing.
Text	The text of the story.
Type	The type indicates if the story represents a successful or exception interaction.
Actors	A story my list the actors represented in the text.

Table 13 Story Properties

Actor

The actors represent the roles that users and other entities external to the system, such as other systems, play in the interactions that the system supports.

Name	An actor has a project unique name.
Description	A description of the role the actor plays in the system.
Goals	The goals the actor has in the system separate from the interaction specific goals in a use case.

Table 14 Actor Properties

Scenario

A scenario defines a sequence of steps used to complete an interaction with the system, or a common partial interaction. A scenario may be successful (meets all the goals) or unsuccessful (meets less than all the goals.)

Name	A scenario has a short project unique name to make it easy find in a search and as a label in documents for cross-referencing.
Use Cases	The use case(s) that use this scenario, which may be empty if scenario is exclusively used as a common step in other scenarios
Scenarios	The scenario(s) that use this scenario as a sub-step, which may be empty if this scenario is a top level scenario defining a complete interaction.

Table 15 Scenario Properties

Step

A step describes one side of a single interaction, from an actor or the system, in a scenario.

Text	The text describing the activity of this step.
Type	The type of step, primary, optional, exception etc.

Table 16 Step Properties

Use Case

Use cases define a specific behavior of the system through actors, goals, stories and a scenario.

Name	A use case has a short project unique name to make it easy find in a search and as a label in documents for cross-referencing.
Description	A summary or overview of the use case.
Primary Actor	The main actor in the use case.
Goals	The goals that the actors are attempting to accomplish by doing the use case.

Stories	Stories used to augment the use case with simple narratives.
Auxiliary Actors	Additional actors the system may need to interact with to complete the use case.
Scenario	The scenario of the use case. The scenario may contain multiple paths that are successful or exceptional.

Table 17 Use Case Properties

Glossary Term

A glossary term defines a single term or concept. Terms may be cross-referenced indicating that they refer to the same concept, such as “the system” and “the application.”

Name	A project unique string representing the term. It may be a single word or a phrase.
Definition	A text definition of the term.
Canonical Term	If two (or more) terms represent a single concept; the subordinate terms will reference the primary or preferred term. A term can only be subordinate to a single term so that the meaning will be unambiguous.

Table 18 Glossary Term Properties

Document

Documents represent the types of documentation that can be generated for the project. Each project can have specialized documentation.

Name	A project unique name identifying the type of document that will be generated.
Generator	The XSLT script that generates the document from the project elements.

Table 19 Document Properties

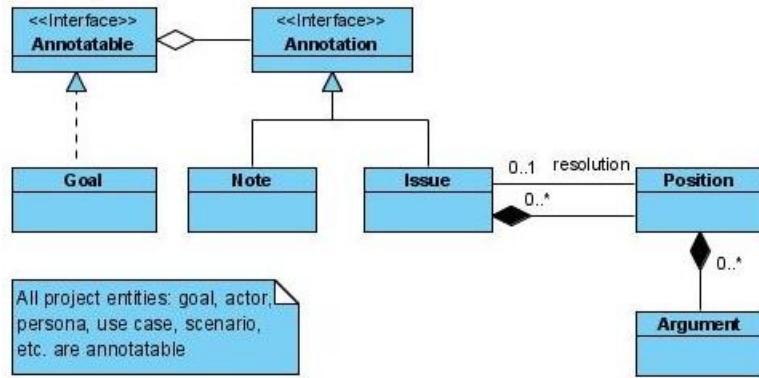


Figure 11: Annotation Objects

Note

A note is a simple annotation used to provide extra information to requirement elements. Notes don't have any discussion or resolution elements.

Text	The text of the note.
Annotated Entities	The requirement elements that refer to the note.

Table 20: Note Properties

Issue

An issue is an annotation that supports discussion through positions and arguments.

Text	The text of the issue.
Annotated Entities	The requirement elements that refer to the issue.
Positions	The potential solutions to the issue.
Resolution	The position that resolves the issue, if it is resolved.

Table 21: Issue Properties

Position

Positions represent the potential solutions to issues. They can be supported by a set of arguments to help identify the best solution.

Text	The text of the position.
Arguments	A set of arguments that argue for or against the position.
Issues	The set of issues that use this position.

Table 22: Position Properties

Argument

A position may have multiple arguments supporting or contradicting it. Collectively the arguments for all the positions should help the stakeholders agree on the deciding position.

Text	The text of the argument.
Position	The position the argument supports.
Support Rank	A rank from “strongly conflicts” to “strongly supports” to give arguments a relative weight.

Table 23: Argument Properties

External Components

This section describes the external tools and frameworks that Requel uses.

Spring Framework

The Spring Framework is an alternative or auxiliary container to the Java Enterprise Edition (JEE) platform. At its core is the Inversion of Control (IoC) container that creates and connects application objects together (Johnson, 2007, pp. 17-19.) This concept is also

known as “Dependency Injection” a term coined by Martin Fowler. In dependency injection an object is supplied with all the objects it needs through constructors or method parameters to perform an action. This differs from the “Service Locator” pattern where an object “asks for” the objects it needs through a locator interface (Fowler, 2004) or creates the objects it needs internally.

The biggest benefit of using the Spring Framework is in the configuration and plumbing of the application. It supports configuration through annotations on the classes and configuration files. Requel uses a mix of annotation and file based component mapping that makes configuring the system simple.

The Command Handler configuration is a good example of the power of dependency injection and the Spring Framework. During testing it was discovered that users working on the requirements could cause the analysis, running concurrently in the background, to fail due to database locking. Updating all the analysis code to catch locking exceptions and restart commands would make the code complicated and fragile. By creating a command handler decorator that intercepted the locking exceptions and re-executed the command, none of the commands or code that used the commands had to change. The only change was writing the decorator and changing the configuration of the command handler to use the new class. The Spring Framework injected the new handler in all the places where the original handler was used.

“One of the central tenets of the Spring Framework is that of non-invasiveness; this is the idea that you should not be forced to introduce framework-specific classes and interfaces into your business/domain model.” (Johnson, 2007, pp. 125.)

One of the main reasons the Spring Framework was chosen is that it doesn't require programming to an API to use it. The application code can focus on the problem domain and only a small amount of code is required to get the configured objects from the Spring Framework. Direct access to the Spring Framework's container API is limited to only a few places: the main application servlet, the database initialization listener, and in the command factories via a helper class that encapsulates the bean creation strategy.

The Spring Framework offers many other services; those used in this project include transaction management, Aspect-Oriented Programming (AOP), and thread pooling.

Java Persistence API and Hibernate

Object-Relational Mapping is a technique for combining the benefits of object-oriented programming and relational databases. Differences in representation of object and relational models and features of object and relational systems add complexity to this mapping. Hibernate is an open-source ORM tool that helps bridge the gap between the object and relational models. Hibernate supports the Java Persistence API, an official Java standard for managing object-relational mapping.

Hibernate is primarily concerned with transparent persistence of Java objects without the need for writing JDBC or SQL code to save and load them. Transparent means that it has minimal impact on the implementation of the application. Hibernate is designed to work with plain old java objects (POJOs), it doesn't require the classes of the objects that will be persisted to implement specific interfaces. Although you may need to include

the java.io.Serializable interface and implement hashCode() and equals() methods to get associations working properly.

One of the most powerful and at times frustrating features of Hibernate is the saving or loading of objects through reachability. What this means is that Hibernate traces through all the associations of an updated object to find other objects that have been changed and saves them as well. This was very useful for project importing where JAXB creates Java objects for all the requirements elements of a project and by making the project object persistent Hibernate saves all the objects associated to the project.

Another key feature of Hibernate is lazy loading, which means that when an object is loaded the objects it is associated with don't have to be loaded. Instead Hibernate creates a proxy representing the associated object and loads it from the database when it is accessed. This is extremely efficient especially in an application such as Requel that has a highly connected model. Without lazy loading all of the objects in a project would get loaded any time one of them was needed.

A problem with lazy loading is that the proxies standing in for the unloaded objects require an active session to load the data from the database. In stateless Web applications where pages are generated and all the objects released each time lazy loading doesn't cause problems. However, because Requel uses the Echo2 framework, which is not stateless, and the domain objects are used as the model objects in the user interface, lazy loading became a critical problem. In Requel's architecture the user interface and domain models are in separate layers. As soon as an object is passed from the domain layer to the user interface layer it loses its connection to Hibernate and the proxies no longer have a session to load the data. Hibernate allows attaching a disconnected object to another session for loading

lazy objects, but that requires having the user interface connect the object to Hibernate each time it needs to read properties, making the code complicated and fragile. Another problem is that even when the objects have the data already loaded the data becomes out dated and different user interface components end up with different versions of the objects.

Requel's solution is to add a proxy around each persistent object that caches the objects data, periodically refresh the cached objects data depending on how old the cache is, and traps accesses to Hibernate lazy loaded proxies and load them in the context of refreshing the whole object. More information is provided in the implementation section.

Echo2 Framework

The Echo2 framework is a rich Web client framework based on the MVC pattern (more accurately the Model-Delegate variant of the MVC pattern) and is very similar to the standard Java Swing API. It has many basic components such as text inputs, different types of buttons, lists, as well as more advanced components like tables and tabssets. The EchoPointNG project extends some of the Echo2 components and adds new ones, such as a tree component.

A major problem with Echo2 is that creating new components not built as composites of the existing components is complicated requiring a mix of Java and Javascript. Working with the Echo2 Javascript can be difficult. For example, there was a bug in the interaction of the tree component and drag-and-drop used in Requel that took days to debug. Echo2 uses the Javascript “eval” command as a form of reflection to dynamically call extension code. The Firebug Javascript debugger could not identify the code across the boundary and display the full call stack.

Another problem with Echo2 is dealing with restrictions of what components can be nested in other components. For example an Echo2 panel can only contain a single component. It requires using specialized layout components like grids, rows and columns to contain multiple children. However, grids, rows and columns cannot contain panels. The EchoPointNG extended panel does support multiple children components so it is used as the basis for Requel's panels.

Echo2 has its own XML based stylesheet language for styling the look of components. It is somewhat complicated to work with primarily because there is little documentation and the style properties are component specific. Also, only one stylesheet can be used at a time so all the styles for all panels must be contained in a single file. Despite these issues Requel is configured so that the styles of all the components are configurable using an Echo2 stylesheet.

Hibernate Validator

Validation of user input is important, but coding it by hand is tedious to implement and can make code fragile. The Hibernate Validator simplifies validation by allowing constraints to be defined as annotations on the properties of the domain objects and performing the validation in the persistence engine when an object is about to be saved.

Java Architecture for XML Binding (JAXB)

JAXB is a framework for working with XML data files with the promise of not having to understand XML or XML processing APIs (Ort & Mehta 2003.) It uses a binding compiler to generate a set of Java classes representing the XML data from an XML Schema (XSD) file or a schema generator to create an XML Schema from a set of Java

classes. It works for both input and output of XML data using an “Unmarshaller” to read the XML file and generate a graph of Java objects and a “Marshaller” to write out a graph of Java objects.

The initial JAXB 1.0 API was oriented towards the XML data and schema, and not the object model. It was really an API for wrapping an XML document in JavaBean objects and not a framework for importing and exporting a Java object model. Version 2.0 of the API added facilities for starting from the Java side. JAXB 2.0 adds a schema generator, a set of annotations for controlling the output of the XML, and the ability to add custom adapters that alter JAXB’s view of objects (Goncalves, 2007.)

The advantage of using annotations to control the mapping to XML with the ability to automatically generate a corresponding XSD file makes JAXB seem like an efficient and time saving approach.

JAXB has a few issues that made it a little hard to work with. The biggest short coming was with the Unmarshaller callback mechanism for doing pre and post processing of objects as they are loaded. JAXB supports a reflection-based method where it looks for before and after unmarshal methods on each object and a listener based method that puts the processing for all objects in a single class. The listener supports processing of objects just after they are created, but before they get filled in with data, and after they have been filled in and all child elements have been processed.

Requel required the ability to replace new object instances created during unmarshalling with existing persistent instances from the database, particularly for users, which may already exist in the system, and permission types, which are shared singleton objects, that are referenced in the XML file. Unfortunately the reflection-based method

only supports methods that take the Unmarshaller and the parent object as parameters, so there is no way to pass a repository through the Unmarshaller to load a persistent object. The listener method supports using a repository, but then the processing for all domain object types is handled by the single listener class. Requel's solution is a custom reflection-based method using a listener. The listener inspects each object for methods named "beforeUnmarshall" and "afterUnmarshall" with different parameter signatures and calls the methods it finds, passing the appropriate objects.

Stanford Parser

The Stanford parser is actually a collection of parsers that generates both constituent and dependency parses. It has a built in sentence detector, tokenizer and tagger, although the sentence detector seems to treat most periods as ending a sentence. The Stanford parser uses probabilistic context free grammars that may or may not include lexical information (Klein & Manning, 2003.) The novelty of the Stanford parser is that it uses separate models for the constituent and dependency parsing and then combines them into a final lexicalized parse (Klein & Manning, 2003.)

The only problems with the Stanford parser are that it uses a lot of memory to hold the models and the time to parse grows quickly relative to the length of the sentence being parsed.

OpenNLP

OpenNLP is a collection of natural language processing tools implemented in Java including a sentence segmenter, tokenizer, tagger, constituent parser, and named entity

recognizer. In the end only the sentence segmenter performed better than the corresponding Stanford tools and is the only component used.

The sentence detector uses a probabilistic machine learning method called maximum entropy to train a model from example sentences.

WordNet SQL builder

The WordNet SQL builder combines WordNet, VerbNet, and Extended WordNet data into a MySQL database.

WordNet (Fellbaum, 1998) is a lexical database containing over 140 thousand words, over 115 thousand concepts called synsets that give meaning to the words, and over 240 thousand semantic relationships between the concepts. The WordNet database is used for spell checking, word sense disambiguation, semantic role labeling, and vagueness checking.

VerbNet is a verb lexicon of a subset of the verb sense in WordNet that contains syntactic and semantic frames identifying the thematic roles of the words related to the verb in a sentence (Schuler 2005.) The frames are organized into classes of verbs that share similar uses and thematic roles. For example, below is the syntax part of one of the frames for the verb “accompany”. It shows that the noun phrase preceding the verb has the “Agent” role and the noun phrase following the verb has the “Theme” role.

```
<NP value="Agent"></NP><VERB/><NP value="Theme"></NP>
```

Some frames contain selection restrictions on the types of nouns appropriate for that frame. Examples include organization, animate, location, or human. The frames are

useful for semantic role labeling, but could also be used for word sense disambiguation to help select verb senses by the syntax of the sentence and category of the nouns filling the roles.

Extended WordNet takes the information in WordNet and processes it to generate the syntactic parse of each concepts definition, a logical representation of the meaning of the definition, and identify the sense of each word in the definition.

Implementation

Requel is a Web application based on the Java Enterprise Edition (JEE5) and Spring Framework platforms, intended to run in a Web container such as Tomcat. The Requel system is implemented using over 600 Java classes. This section describes the overall structure of the code and the implementation of some of the interesting aspects of the system.

Package Structure

Requel has three primary domain packages for the core application domain, natural language processing, and the user interface framework. Requel has two support packages for general command processing and repository support.

Package	Content
edu.harvard.fas.rregan.command	This packages contains the base interface for all commands, the base interface for command factories, the interface for the command handler, and the implementation of the default handler, a handler that does exception mapping from external exceptions to Requel specific exceptions, and a handler that catches database lock related exceptions and re-executes the command.
edu.harvard.fas.rregan.nlp	This package contains the classes for modeling the

	<p>processed text, adapters for external tools, and the implementation for a lemmatizer, word sense disambiguator and semantic role labeler.</p> <p>A sub-package contains the interface to the lexical database (WordNet, VerbNet, etc.)</p>
edu.harvard.fas.rregan.repository	<p>This package contains a base repository implementation for JPA with functions for reloading an object from the database, making an object persistent, and deleting objects.</p> <p>This package also contains the implementation of the domain object proxy and AOP advice used to wrap persistent objects so they can refresh themselves from the database and load lazy collections.</p>
edu.harvard.fas.rregan.requel	<p>This package contains the domain logic for managing users, working with requirements and projects, and working with annotations. Each domain has its own sub-package.</p> <p>There is a package that contains the user interface components for all the domains.</p>
edu.harvard.fas.rregan.uiframework	This package contains all the code for the custom user interface framework built on top of Echo2.

Table 24: Requel Source Code Packages

Application Initialization

The Requel system uses JEE ServletContextListeners to create the database, initialize the Spring Framework, and load data into the database when the application is first loaded. Figure 12 shows a sequence diagram describing the startup process.

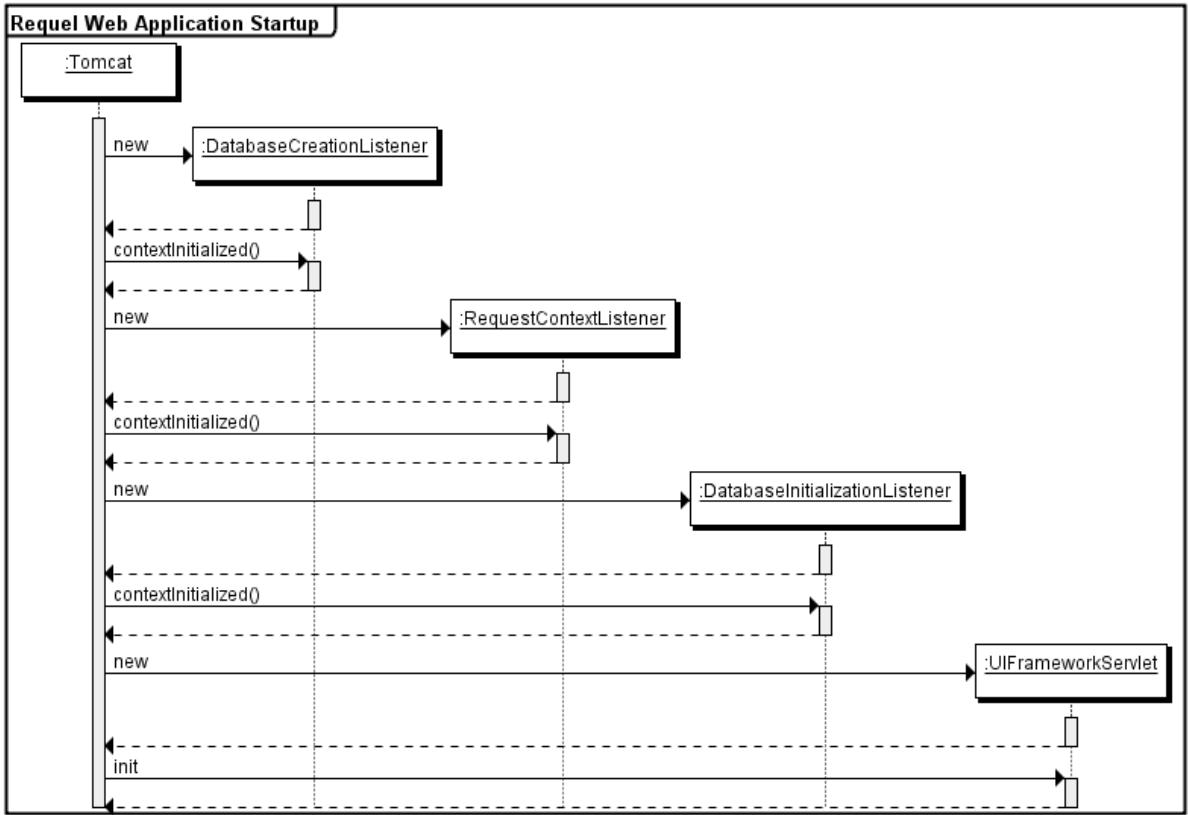


Figure 12: Application Initialization

The `DatabaseCreationListener` attempts to connect to the database specified for the system, and if the connection fails because the database doesn't exist, it creates the database.

The `RequestContextListener` is part of the Spring Framework. It loads the Spring configuration files and scans the classpath for classes annotated with Spring "component" annotations. It initializes Hibernate and the Hibernate Schema Update automatically generates the database schema and creates or modifies the tables and constraints.

The `DatabaseInitializationListener` gets the `DatabaseInitializer` object from the Spring Framework context. The `DatabaseInitializer` is configured with a set of `EntityInitializers` for loading the dictionary data, creating the default users, and initializing the user and stakeholder permissions. Figure 13 shows the classes involved in the process.

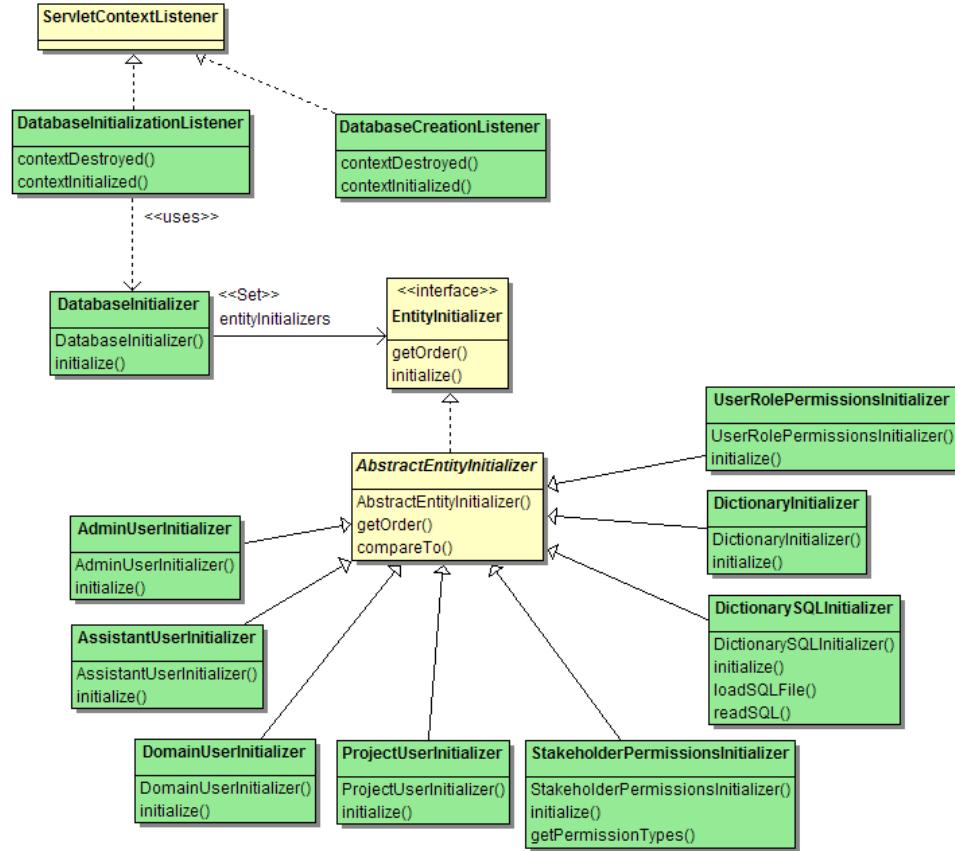


Figure 13: Database Initializers

Finally the UIFrameworkServlet is loaded and it initializes the Echo2 components.

User Interface Framework

A downside of the Echo2 framework is that it only provides low level components, which requires significant effort to create higher level constructs such as forms and navigation. Another issue, inherent in the observer pattern used for event passing, is that the controllers need to be explicitly wired together with the components they listen to. That is not such a big problem for components and controllers that are only concerned with activity internal to a panel or form, but when panels and forms need to interact, managing the relationships can add complexity and unwanted dependencies.

The UI Framework builds upon the Echo2 framework and provides higher level components including screens, panels and editors. The framework includes facilities for screen and panel management, entity and event type navigation, and event based message passing via a message broker. Figure 14 shows the core panel, screen and management classes of the framework. The key components are the Panel, PanelFactory, and PanelManager. A panel manager uses a panel factory to create new instances of a panel when the manager receives an open event, or to display an existing panel if the open event references the panel directly.

The UI Framework uses the Spring Framework to wire together the event dispatcher, screens, panels and controllers when a user first access the system as shown in Figure 15. The use of the Spring Framework and the event dispatcher allows the panels to be completely independent of each other. For example, an edit button on the goal navigator panel fires an open event for editing a goal. The panel manager receives the message and searches for an existing panel already open for editing the goal, and if one isn't opened it searches for a panel factory for creating a new panel for editing the goal. A new goal editor panel can be written and added to the application by simply changing the Spring Framework panel configuration to refer to the new panel. None of the existing panels need to change to use the new panel.

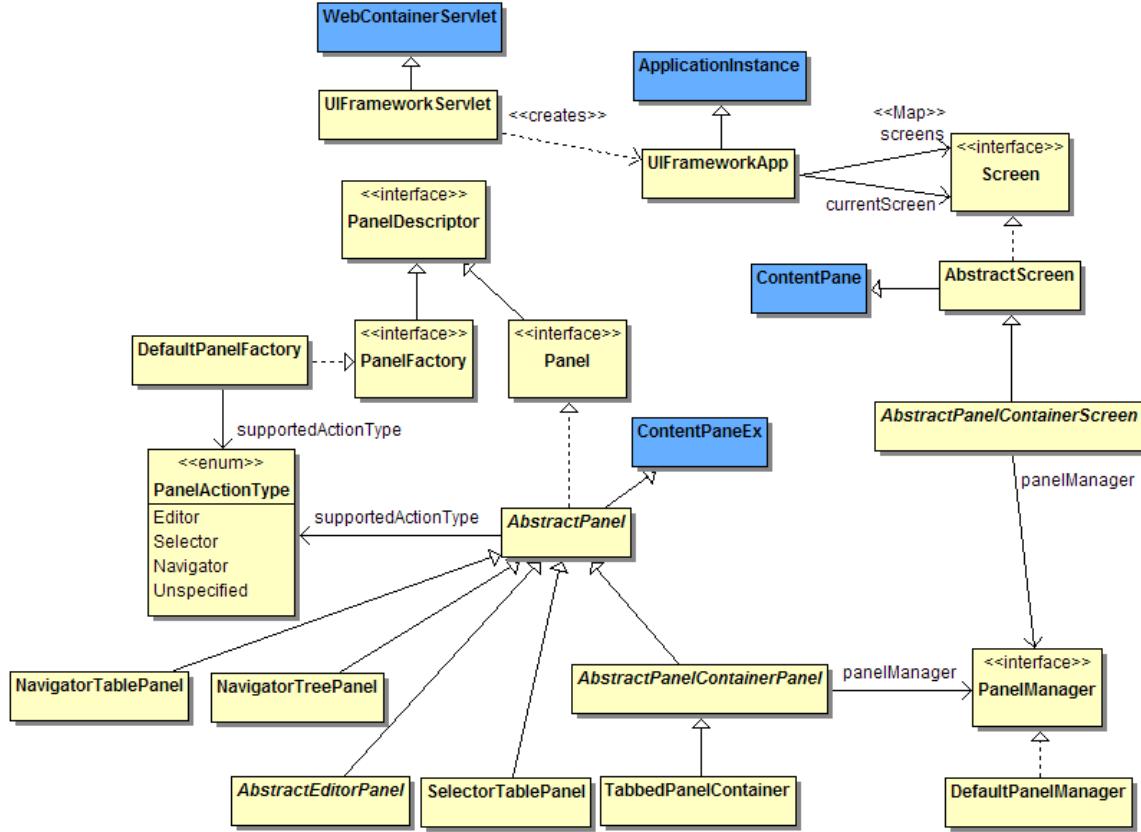


Figure 14: UI Framework Screens and Panels

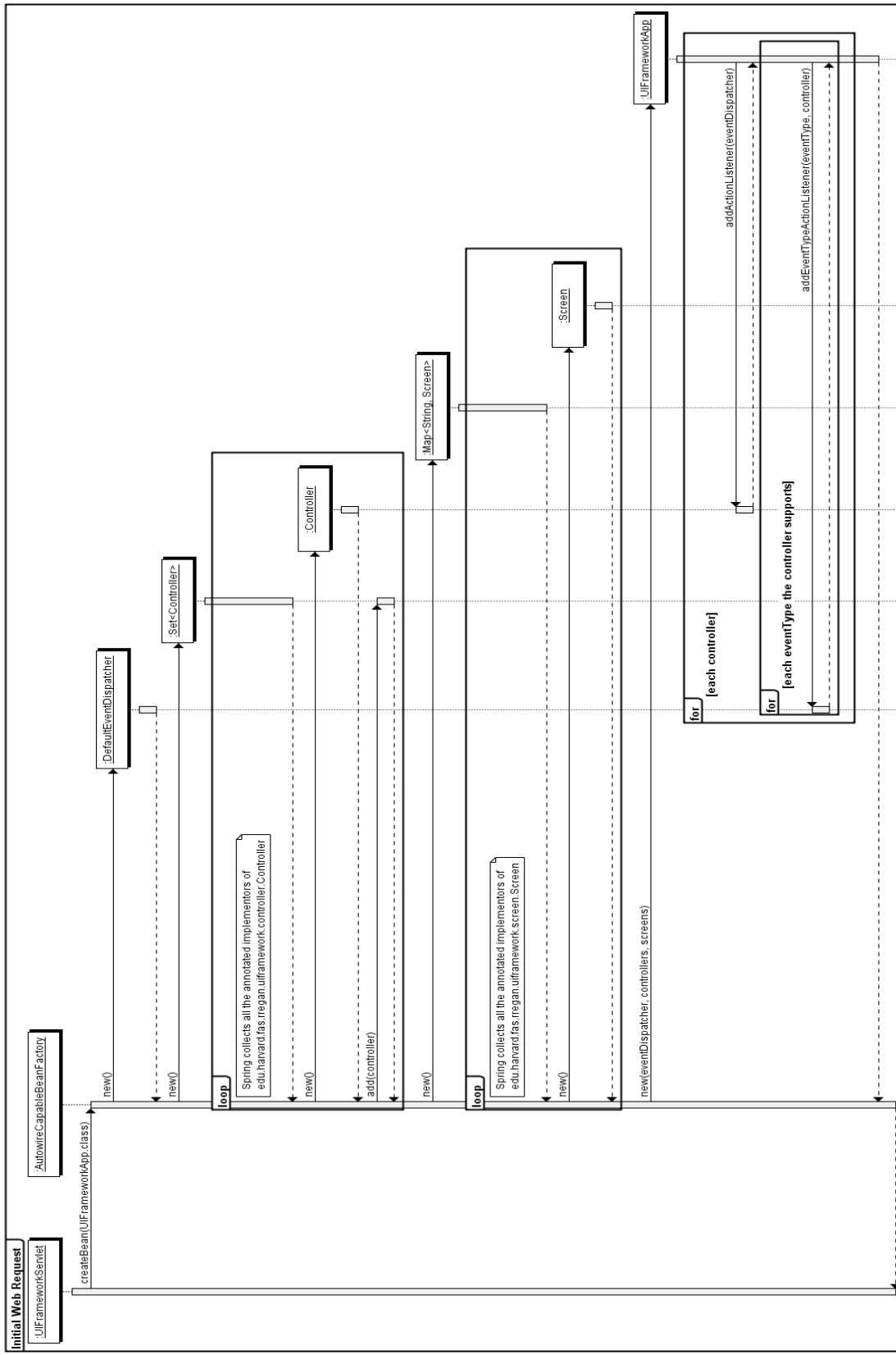


Figure 15: UI Framework Initialization

The Echo2 Framework uses event based message passing to indicate simple user interface component actions such as clicking a button, selecting an item in a list, or changing the contents of a text box. The UI Framework builds on this mechanism with a richer set of event messages for navigational events such as login, logout, open a screen, open an edit panel, open a panel to search/select an entity of a particular type, or to indicate that an entity has been changed or selected. Figure 16 shows the classes that make up the events and dispatching mechanism.

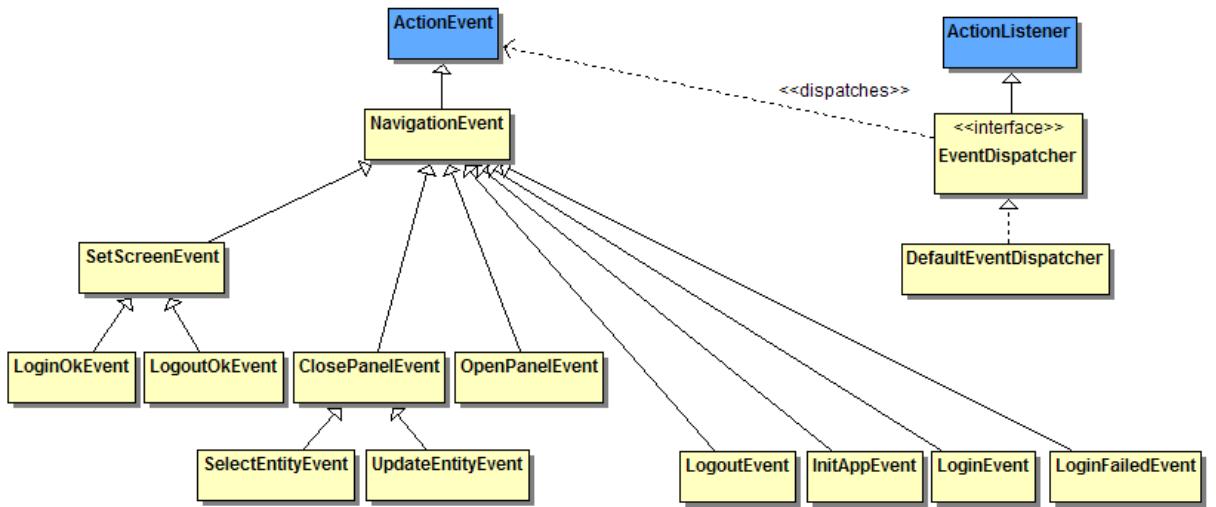


Figure 16: UI Framework Events

The framework uses both local and global events and listeners. The local events and controllers are for actions specific to a panel or composite UI component. For example a delete button on an editor is registered to a local controller on the panel. The listeners may be distinct classes such as the SubmitLoginListener or may be implemented directly by a panel. They are registered directly with the components that generate the events using methods such as addActionListener(). The listener may handle the event locally (within a single panel) by acting on the panel to change a data value on the panel or invoke a method

on the panel, for example the save method on an editor panel. A local listener may generate a message like a Login Event and pass it through the Event Dispatcher to a global level controller. Global controllers listen for navigational events fired by the event dispatcher. A Panel Manager is an example of a global controller that listens for requests to open or close panels. Having two levels of messages improves the performance of the event dispatcher by not having it deal with the messages passed between components on a panel. Figure 17 and Figure 18 shows the sequence of event processing from the local level to the global level for the login process.

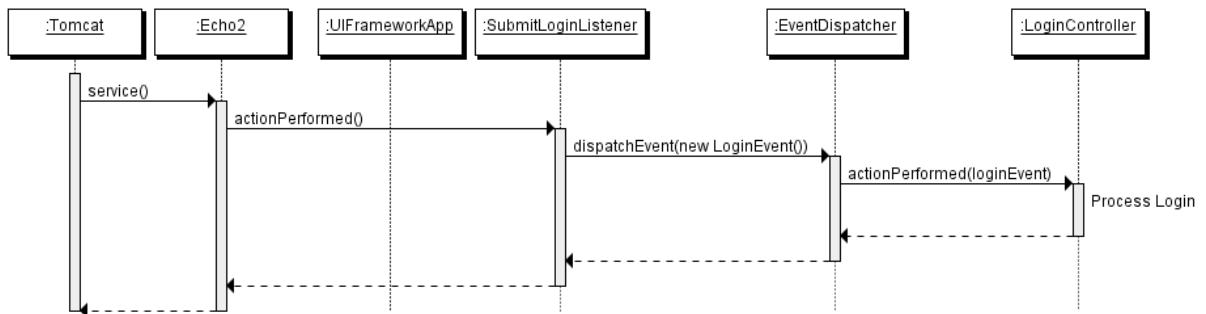


Figure 17: Event Processing Sequence of Login up to the EventDispatcher



Figure 18: Event Processing Sequence of Login after the EventDispatcher

Requel User Interface

The user interface builds on top of the UI framework and Echo2 to create the panels for navigating around and editing the domain objects. Figure 19 shows the main screen of the Requel application.

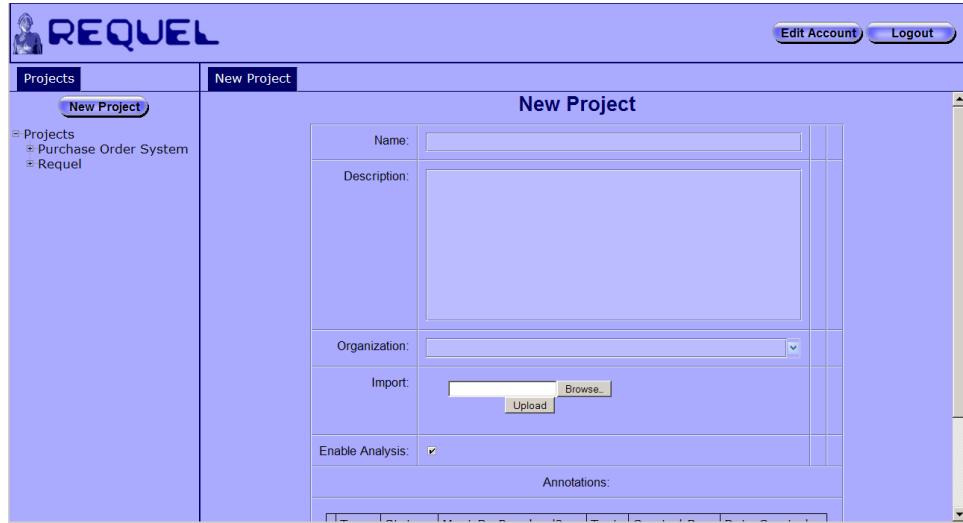


Figure 19: Requel Main Screen

Figure 20 shows the classes involved with the main screen and navigation. The tabbed navigation on the left side of the screen is managed by a MainScreenTabbedNavigation panel container. The panel contains one or more ProjectNavigatorPanels depending on the roles of the user. The trees in the panels are configured using TreeNodeFactories specific to the type of target object being navigated. For example the Projects navigator panel uses two factories to build different parts of the tree. The main target of the panel is the ProjectUser role and the ProjectUserNavigatorTreeNodeFactory creates nodes for each project the user is assigned. The ProjectNavigatorTreeNodeFactory creates the sub-nodes for each project in the tree.

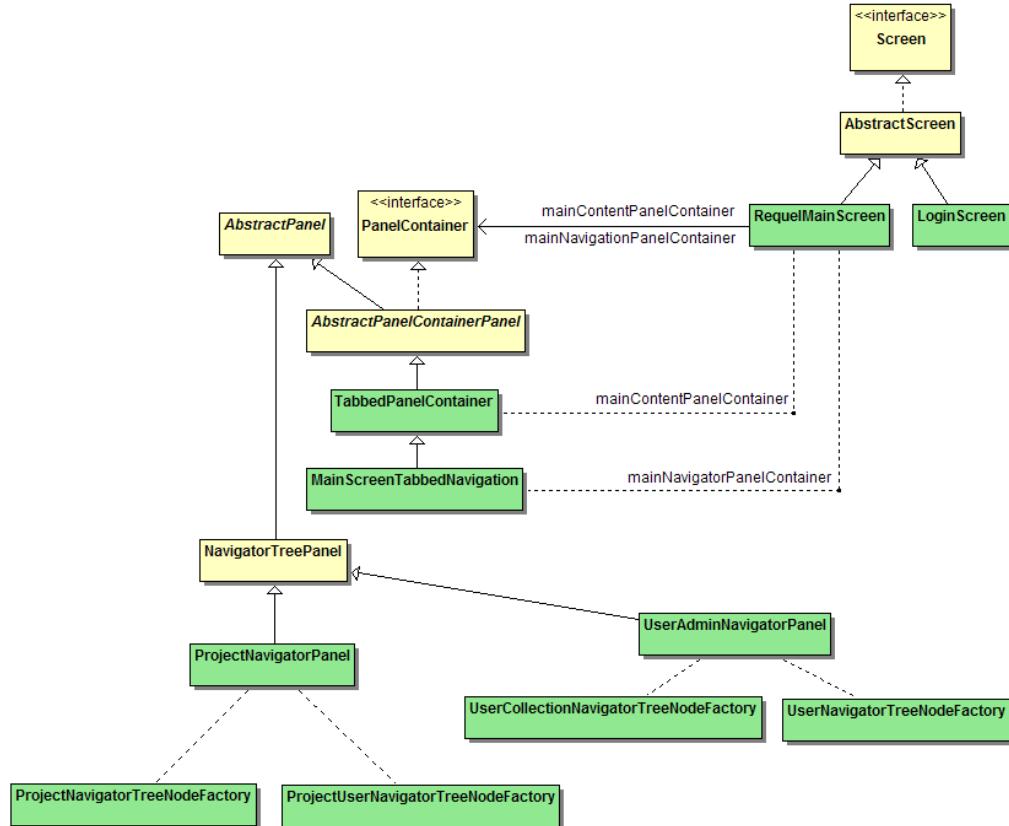


Figure 20: Main Requell Screen Components

The right side of the screen as shown in Figure 19 is the main content panel that uses a TabbedPanelContainer. This panel displays the domain object specific editors, selectors and navigator lists. The use of a TabbedPanelContainer allows multiple panels to be open at once so that users can change which panel is shown by clicking on the tab button at the top.

Command Handler

The command handler is the primary component in the Command Processor architectural pattern described in the architecture. The base handler is used to demarcate transaction boundaries; a transaction starts when the execute() method is invoked and ends

when it returns. Requel uses decorators to alter the functionality of the default command handler.

One decorator catches exceptions thrown during the execution of a command and transforms them into Requel specific entity exceptions. This is done to simplify handling of exceptions generated by different packages. For example, exceptions may be thrown originating from JDBC, Hibernate, JPA, or the Spring Framework. Handling each type of exception throughout the application would overly complicate the code.

Another decorator is used to detect command execution failures due to database locking issues and retry the execution of the command until it doesn't encounter a database locking related failure or a preset maximum number of retries is tried.

The final decorator is specific to commands for editing requirements. It detects if the resulting domain object of a command should be analyzed and invokes a special method on the command to start the analysis after the commands execute method completes successfully and the transaction ends.

Figure 21 shows the class structure of the command handlers. Each of the decorator handlers has a reference to the inner handler.

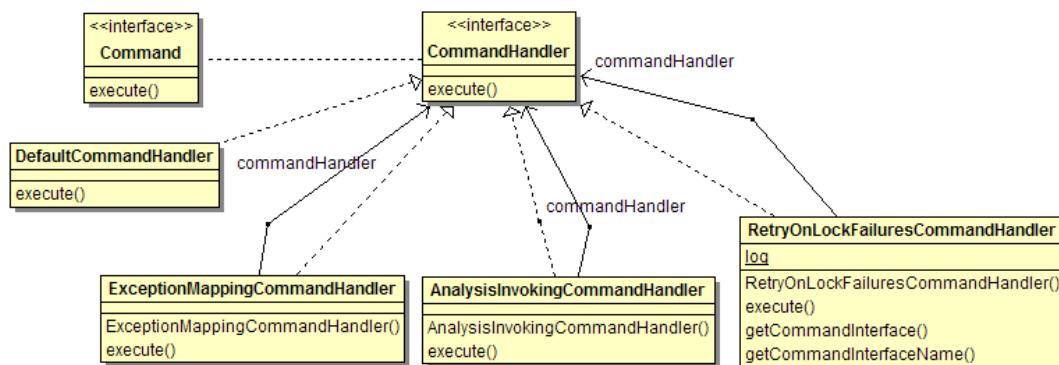


Figure 21: Command Handlers

Figure 22 shows the sequence of execution for an edit command from a panel through three handlers indicating the transaction part and exception handling.

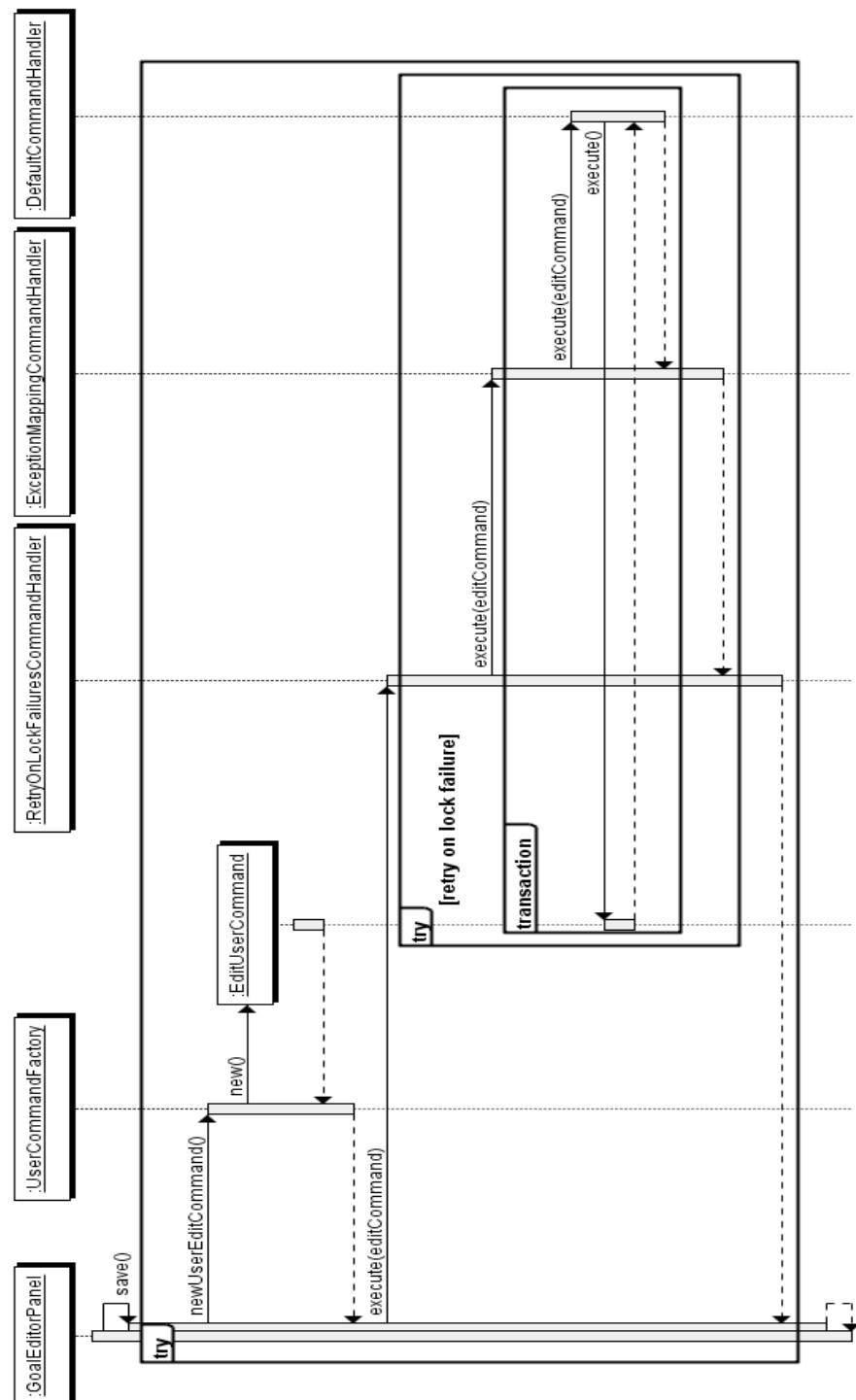


Figure 22: Command Handler Sequence

Command Factories

Spring manages the creation and injection of most of the components used in the system except for commands. This is because the commands are single-use objects used by the assistant or user interface components, which have a lifecycle that lasts for a user session or longer.

The command factories use the Spring Framework to create the command objects through the ApplicationContextCommandFactoryStrategy. Each time a factory method is called a new command object is created and returned. Figure 23 shows the class structure of the command factories with the UserCommandFactory as a concrete example.

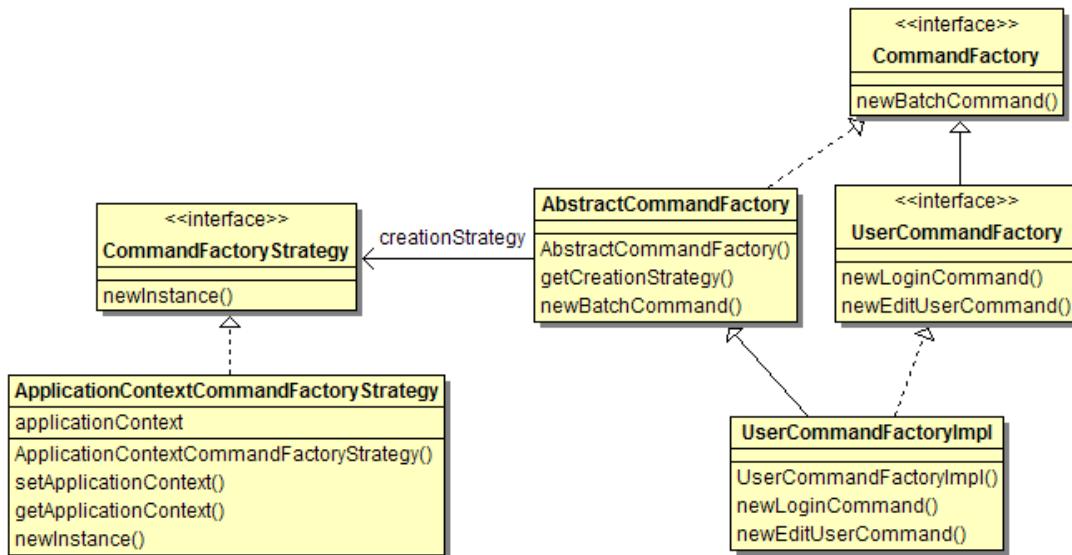


Figure 23: Command Factories

Domain Object Proxy

The domain objects of Requel, such as Users and Projects, are used as the model objects for most of the user interface framework panels. This ends up being a problem because the domain objects are persistent entities and when they get attached to the user

interface components they lose their connection to Hibernate. This leads to two problems. First, the objects become out of date as they sit attached to the user interface because changes made by other users or the assistant and saved to the database don't get forwarded to the disconnected objects. Second, Hibernate uses proxies to stand in for other associated entities so that they don't have to be loaded unless needed. When the connection to Hibernate is lost the proxies lose their ability to load the underlying entities.

Requel's solution to the problem is to add a proxy around each persistent object that keeps a reference to the persistent entity, periodically refreshing it to keep it up to date. When a method is called on the proxy to access a lazy loaded property, the call on the cached object is wrapped to catch lazy loading exceptions and retry loading in a transaction.

There were a lot of technical challenges in implementing the proxies. The largest challenge was how to get the domain objects wrapped in the proxies to begin with. Explicitly wrapping each object as it gets passed to the user interface layer is not practical. Custom entity loaders could have been added to Hibernate to wrap each object as it is returned, but always getting a proxy back is undesirable, especially in commands that are bound by a transaction. Requel's solution uses the Spring Framework's Aspect-Oriented Programming (AOP) features.

In AOP an aspect is a unit of functionality that is reused at multiple pointcuts. A pointcut consists of the "location", in Spring this is always a method call, and rules that determine if the aspect is invoked. The advice indicates how the aspect is applied, such as before, after or before and after a method call. Requel's pointcut indicates to intercept objects as they are returned by a repository or command object, and only if a method call

wasn't called from inside a command in a transaction. Requel's aspect is to wrap domain objects in a proxy, and the advice occurs after a "get" method is called to retrieve a domain object.

Figure 24 shows how the DomainObjectAdvice works inside a command and when returning an object from a command to a panel. Spring creates a proxy for the Command and Repository objects that uses the Spring AOPInterceptor to apply the DomainObjectAdvice on calls to the methods. In the top half of the sequence the command calls a get() method on a repository to get a domain object. The DomainObjectAdvice detects that the call is being made from inside a command execute() method, so it does not wrap the domain object in a proxy. In the bottom half of the sequence the panel calls a getResult() method on the command to return the domain object. This time the DomainObjectAdvice doesn't detect that the call is being made from inside a command execute() method, so it creates a new DomainObjectWrapper and EntityProxyInterceptor (the proxy) and returns the proxy to the panel.

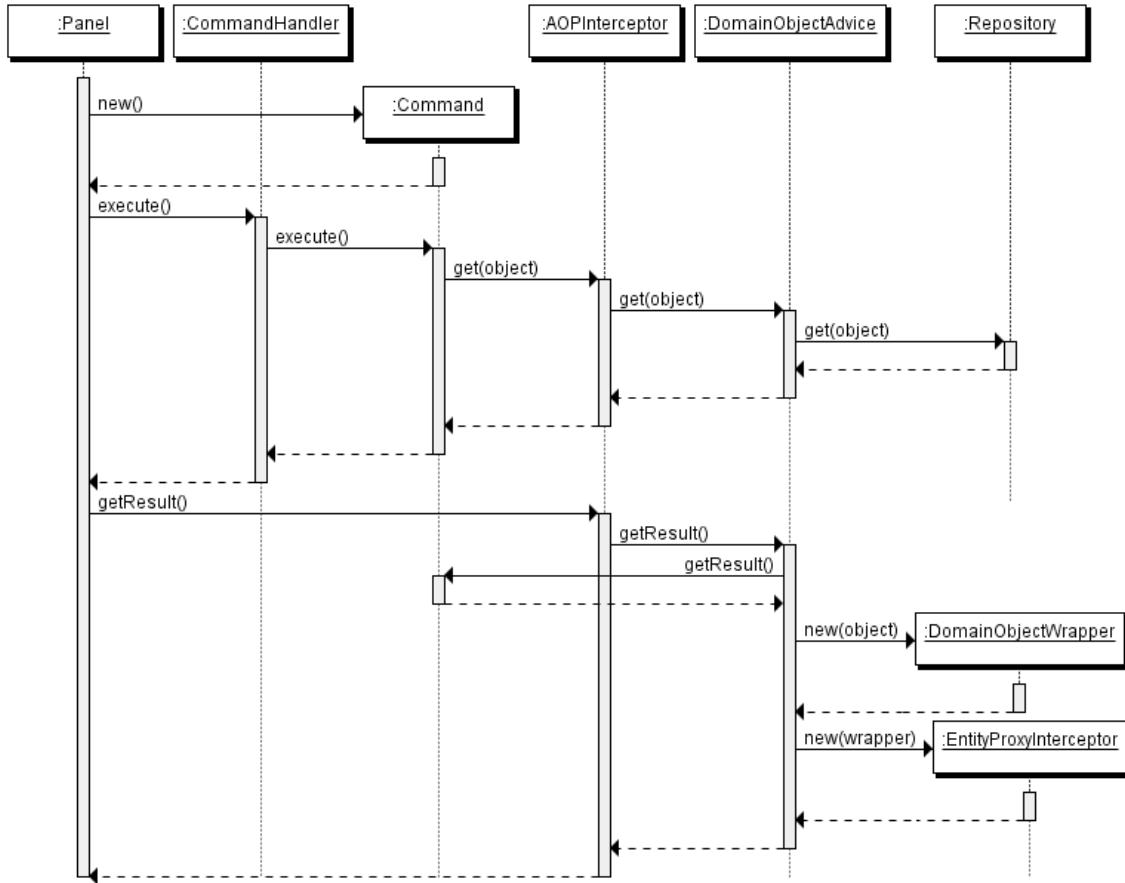


Figure 24: Domain Object Wrapping Advice in Command

Figure 25 shows the sequence where the panel calls the repository to load an object.

Spring creates a proxy for the Repository objects that applies the AOPInterceptor and DomainObjectAdvice on calls to its methods. The DomainObjectAdvice doesn't detect that the call is being made from inside a command execute() method so it creates a new DomainObjectWrapper and EntityProxyInterceptor (the proxy) and returns the proxy to the panel.

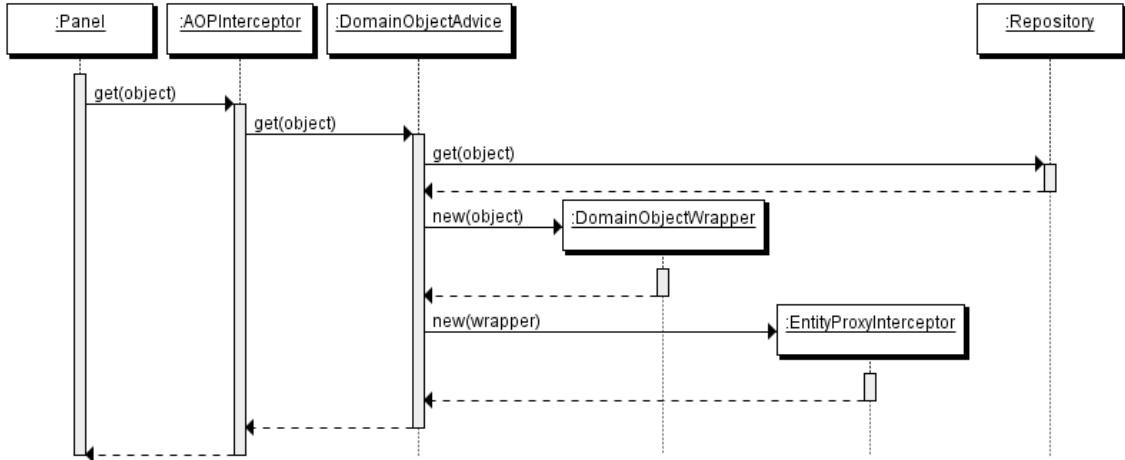


Figure 25: Domain Object Wrapping Advice out of Command

Figure 26 shows a user interface component accessing the property of an entity through the proxy. The proxy calls the EntityProxyInterceptor intercept method to get the property from the cached domain object. If the cached object is stale, for example it has been cached in the interceptor for a few minutes; the interceptor passes itself and the method to invoke to the PersistenceContextHelper. The helper retrieves a fresh copy of the entity from the database and processes the method invocation in a transaction. The helper caches the fresh copy of the entity in the interceptor, passes the results of the method through the DomainObjectWrapper to create proxies for new entities, and returns the value back through the interceptor to the user interface component.

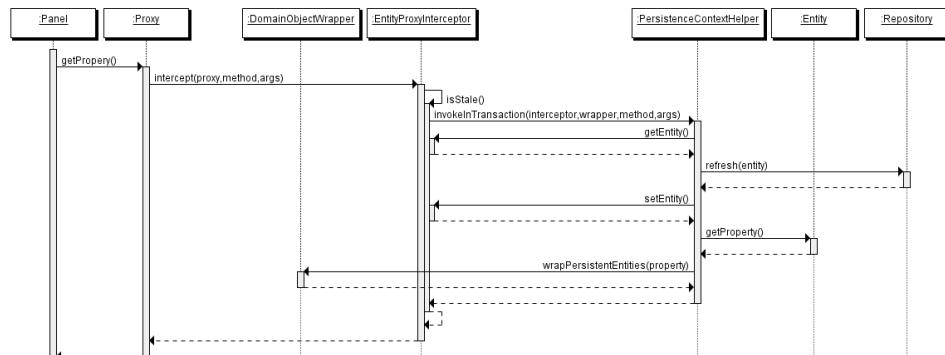


Figure 26: Entity Proxy in Action

Word Sense Disambiguation

Word Sense Disambiguation (WSD) is one of the most difficult problems in NLP. Requel uses the word similarity and collocation disambiguation techniques described in the Natural Language Processing section of Chapter 1. Requel also uses a relatedness method loosely based on the Lesk method described in Chapter 1. The relatedness method calculates the word similarity of the words that define the words being disambiguated. The intent of the relatedness method is to get around the caveat that word similarity works only for words in the same part of speech. By using a level of indirection nouns and verbs are compared based on the words that define them.

The word similarity uses the WordNet information content formula defined by Seco, Veale and Hayes (2004) shown in Equation 1 using the WordNet hypernym-hyponym relationship. In selected test cases the word similarity measure worked quite well on nouns.

The collocation method searches for word sense pair combinations from semantically tagged WordNet definitions and SemCor sentences. Senses that were found to be co-located were given a higher ranking than those without a collocation.

The word sense disambiguator performs poorly. The collocation method did not appear to improve the quality of the disambiguation at all, but had a huge performance penalty so it was disabled in the final version.

Semantic Role Labeling

Requel uses VerbNet frames to match a verbs usage and then assign the roles based on the phrase structure to role mappings in the “syntax” of the frame. If a sentence doesn’t

match a VerbNet frame Requel uses a simpler assignment based on the subject, direct object, indirect object, and preposition objects in dependencies identified by the parser.

In selected test cases of simply structured sentences the method worked fairly well, but due to the poor performance of the word sense disambiguator and more complex sentence structures in the sample requirements the roles were mostly assigned by the simpler fall back method using the dependencies.

Chapter 4 User Guide

This chapter describes how an administrator configures and installs the Requel system, and how users create and discuss requirements using the Requel system.

For administrators, the Requel Setup section describes how to prepare and install the Requel system in a Web server. The User Administration section describes creating and editing users, and assigning roles and permissions.

For system users, the Requel system should already be installed on a Web server and your administrator should have given you a URL to access it. The Getting Started section defines Web browser requirements, how to connect to the system, and how to login. The Working with Requirements section describes how to create a project and add requirements elements to it. The Discussing and Negotiating Requirements section describes adding notes and issues to requirements and negotiating resolutions to issues.

Getting Started

To get started you need a Web browser, the URL to the Requel system, and a username and password. Mozilla Firefox version 3.0 or later is the only Web browser with which the system is certified to work correctly.

To connect to the system, enter the URL of the application in the address bar of the browser. The system returns the login screen as shown in Figure 27. Enter the username and password supplied by your administrator and click the “Login” button.



Figure 27: Login Screen

If your account is setup correctly you will see the main screen as shown in Figure 28. On the left hand side of the screen is a tab labeled “Projects” with a tree containing the projects that you are assigned. If you have project creation authorization you will also see a “New Project” button above the projects.



Figure 28: Main Screen

Figure 29 shows a close up of the project navigation tab.

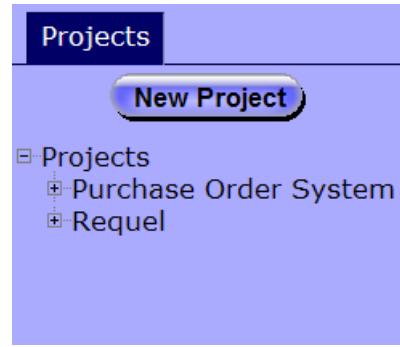


Figure 29: Projects Navigation Details

On the upper right hand of the screen, as shown in Figure 28, are a “Logout” button and “Edit Account” button. The logout button clears out your user session on the server and returns you to the login screen. The edit account button opens the “Edit User” screen for your system account as shown in Figure 30.

When you first login you should change the password assigned by the administrator to something more secure and fill in any missing personal data, such as your name, email address, and phone number. Unless you have system administrator privileges you will not be able to change your username.

A screenshot of the "Edit User" screen. The left sidebar shows the "Projects" navigation with a "New Project" button and a tree view of "Purchase Order System" and "Requel". The main area is titled "Edit User: project" and contains a form with fields for Username, Password, Retype Password, Name, Organization, Email Address, and Phone Number. The "Organization" field has "Requel" selected. At the bottom are "Cancel" and "Save" buttons.

Figure 30: Edit User Screen

After making changes to your account you must click the “Save” button at the bottom to submit the changes to the system. If everything is valid the edit user panel will close and your changes will be saved. If there are any problems with your data, such as the password and retype password fields don’t match, your email address is not formatted properly, or your phone number is not formatted properly, the system will display error messages next to the fields that need to be corrected as shown in Figure 31

Figure 31: Edit User Screen with Errors

Working with Requirements

This section shows you how to create a project and add requirements to it. Requel doesn’t impose any particular process for creating requirements, which makes it flexible, but the flexibility may make it harder for you to get started if you don’t have a process for gathering requirements in mind. If you don’t have a process in mind, you may want to start with stories to describe the behavior of the system.

In any case, to start you will need to create a project and the stakeholders that will work on the requirements.

Creating a Project

Requel organizes requirements into projects. If you have project creation authorization the “New Project” button appears at the top of the project navigation tab as shown in Figure 29. Clicking the button opens the “New Project” screen show in Figure 32.

Name:	<input type="text"/>
Description:	<input type="text"/>
Organization:	<input type="button" value=""/>
Import:	<input type="button" value="Browse"/> <input type="button" value="Upload"/>
Enable Analysis:	<input checked="" type="checkbox"/>
Annotations:	

Figure 32: New Project Screen

The new project screen supports creating a project from scratch or importing a project stored in an XML file.

To create a new project from scratch enter a name, organization, optional description, and click the save button. The combination of project name and organization are required to be unique. The system verifies the values are valid, creates the project, and adds it to the project navigation tab. If there are validation problems the system puts a message to the right of the affected fields.

To import a project from an XML file, click the “Browse” button in the “Import” field and use the file upload dialog to navigate to the file to import as shown in Figure 33. If you are importing a project that already exists in Requel, you must change the name by entering a new name in the name field to meet the uniqueness requirement described

earlier. Only the name can be changed when importing a project; values supplied in the description and organization fields are ignored.

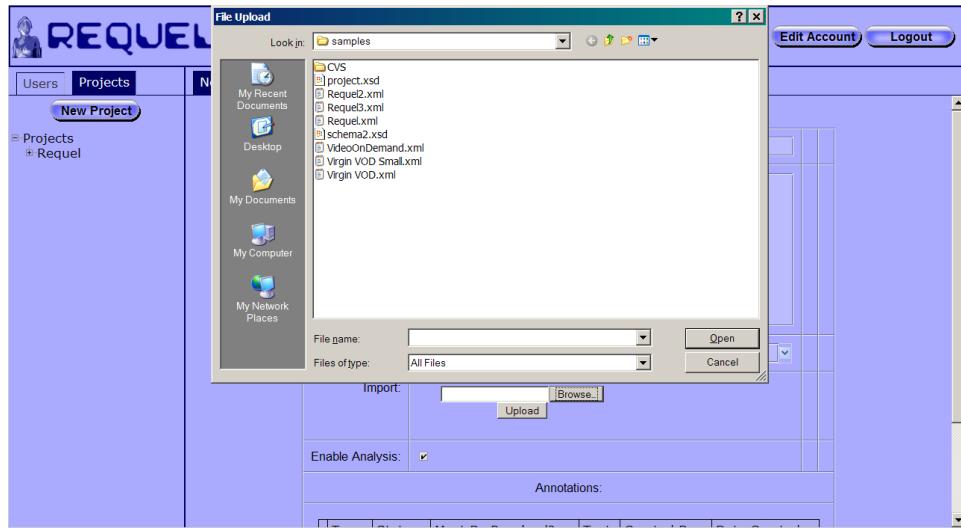


Figure 33: Project Import File Selection

When importing a project you have the option to turn off analysis using the “Enable Analysis” checkbox. This is useful for projects that were exported from the system and the elements have already been analyzed. Since the analysis uses a significant amount of processing power and can take a long time to complete. Projects created by hand or by another tool in the project XML format should have analysis enabled.

After selecting the file in the upload dialog, click the “Upload” button to upload the project file to the server. Then click the “Import” button at the bottom of the screen. The system creates a new project and adds it to the project navigation tab.

Project Navigation

Figure 34 shows a close up of the project navigation tab with the “Purchase Order System” project node in a closed state and the “Requel” project node in an open state. Clicking on the project name opens the “Project Overview” panel for editing the name,

description and organization of the project, as well as adding issues and notes to the project. Clicking on the + next to a project name in the project navigation tab expands the project node to display the links to the requirements elements.



Figure 34: Project Navigation Tab

Creating and Editing Stakeholders

The first task to do after creating a new project is adding stakeholders for the users that will work on the project. Clicking on the “Stakeholders” link under a project in the navigation tab opens the stakeholder navigator, which lists all the stakeholders on the project, ordered by username as shown in Figure 35.

The screenshot shows a web-based application interface titled "REQUEL". On the left, there's a sidebar with a tree view of project components: "Projects", "Purchase Order System", and "Requel" which is expanded to show "Stakeholders", "Goals", "Stories", "Actors", "Use Cases", "Scenarios", "Terms", "Reports", and "Open Issues". The main area is titled "Stakeholders: Requel". It contains a table with the following data:

	Name	User?	Team	Email Address	Phone Number	Created By	Date Created
Edit	admin	yes		rreganjr@acm.org		admin	2009-02-11 00:00
Edit	Analysis Assistant [assistant]	yes		rreganjr@acm.org		rreganjr	2009-01-04 00:00
Edit	Harvard University	no				rreganjr	2009-01-05 00:00
Edit	project	yes		rreganjr@acm.org		rreganjr	2009-01-05 00:00
Edit	Ron Regan [rreganjr]	yes		rreganjr@acm.org	781 645 1574	rreganjr	2009-01-04 00:00

Below the table are navigation buttons: "Close" and "Add". Above the table, there are navigation arrows: "◀◀", "1 of 1", and "▶▶".

Figure 35: Stakeholder Navigator

Users authorized to edit stakeholders will see an “Edit” link in the first column of each stakeholder entry in the table and an “Add” button below the table. If you are not authorized to edit stakeholders the first column contains a “View” link that opens the stakeholder editor, but with read-only access. Clicking on the column titles sorts the stakeholders by that column. The first click sorts them in ascending order alphabetically as shown in Figure 36.

The screenshot shows the same "Stakeholders: Requel" table from Figure 35, but the "Name" column is now sorted in ascending order. An upward-pointing arrow is visible next to the "Name" header. The data in the table remains the same as in Figure 35.

Figure 36: Sorting Stakeholders

The column indicates the sorting order with an arrow next to the column name. An arrow pointing up indicates ascending order and an arrow pointing down indicates descending order. Clicking on the column title, when it is sorted in ascending order,

changes the sorting to descending order. Note: stakeholders without a name, for example the admin and project stakeholders in Figure 36, appear at the end of the results.

There will always be at least two stakeholders, the person that created the project and the “Analysis Assistant” that the system uses to indicate issues and notes created by the system.

The screenshot shows the REQUEL application's stakeholder management interface. On the left, there's a sidebar with a user icon, the word 'REQUEL', and a 'Projects' button. Below 'Projects' is a 'New Project' button. Under 'Projects', there's a tree view with 'Purchase Order System' expanded, showing 'Requel' as a child node, which further branches into 'Stakeholders', 'Goals', 'Stories', 'Actors', 'Use Cases', 'Scenarios', 'Terms', 'Reports', and 'Open Issues'. The main content area is titled 'New Stakeholder'. It contains several input fields: 'Name' (text input), 'User' (dropdown menu), 'Team' (dropdown menu). Below these are sections for 'Goals' (with a table header 'Name', 'Created By', 'Date Created'), 'Annotations' (with a table header 'Type', 'Status', 'Must Be Resolved?', 'Text', 'Created By', 'Date Created'), and a list of existing stakeholders (with a table header 'Name', 'Created By', 'Date Created'). At the bottom are 'Cancel' and 'Save' buttons.

Figure 37: Edit Stakeholder Screen

Clicking the “Add” button on the stakeholder navigator screen opens the “New Stakeholder” screen shown in Figure 37. This screen is used for creating both user and non-user stakeholders for the project. To create a user stakeholder, leave the “Non-User Name” field empty and select a user from the “User” field, as shown in Figure 38.

New Stakeholder

Non-User Name:	<input type="text"/>
User:	<input type="text"/>
Team:	admin assistant project rreganjr

Figure 38: Selecting a User for a Stakeholder

Only administrators can create user accounts. If a person that you want to add to a project doesn't appear in the list they may not have a user account. You should ask the Requiel administrator to add an account for that person, with the "Project User" role.

You may optionally enter or select a team name in the "Team" field. The team is for documentation purposes only. Customer and Supplier are examples of team names.

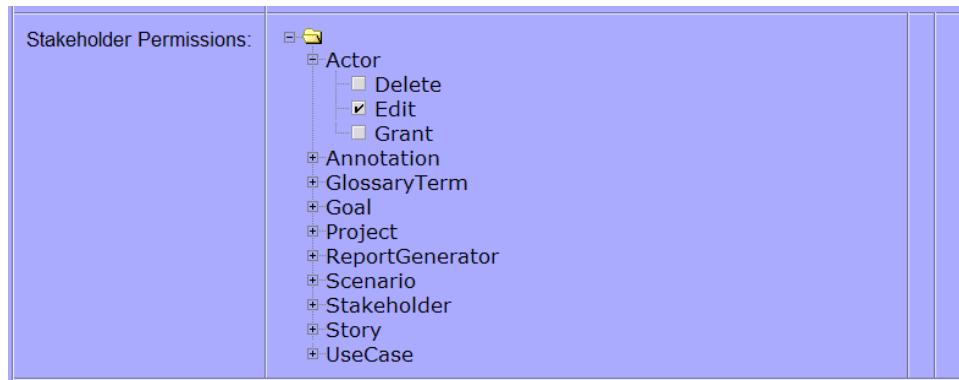


Figure 39: Stakeholder Permissions

User stakeholders can view all requirements elements on a project, but require explicit authorization to edit or delete requirements. The "Stakeholder Permissions" field contains a tree with nodes for each of the requirement element types. Each of the nodes has child nodes with permissions for actions such as "Delete", "Edit", and "Grant", and a checkbox as shown in Figure 39. Checking a box grants permission to the user to perform

that action on that element type; unchecking the box revokes the permission. For example in Figure 39 the checkbox for “Edit” under “Actor” is checked indicating the user can create and edit actors.

The “Grant” action permission allows a user with stakeholder “Edit” permission to grant permissions for that element type to other users. For example, if stakeholder “A” has stakeholder edit permission and story grant permission, he or she can edit stakeholder “B” and grant or revoke “Delete”, “Edit”, and “Grant” permission for actors to stakeholder “B”. If stakeholder “A” does not have grant permission for story elements, he or she can not grant or revoke permissions related to stories to stakeholder “B.”

Non-User Name:	Harvard University		
User:	<input type="button" value="▼"/>		
Team:	<input type="button" value="▼"/>		
Goals:			
	Name	Created By	Date Created
<input type="button" value="Edit"/> <input type="button" value="Remove"/>	Work is your own	rreganjr	2009-01-05
<input type="button" value="◀◀"/> <input type="button" value="1 of 1"/> <input type="button" value="▶▶"/>			
<input type="button" value="New"/> <input type="button" value="Find"/>			

Figure 40: Stakeholder Goals

Non-user stakeholders represent organizations or regulatory bodies that don’t have a user representative, but have interests or goals that the system must meet. Figure 40 shows a non-user stakeholder “Harvard University” with one goal named “Work is your own”; representing that projects submitted for homework are expected to be your own work. Non-user stakeholders are created using the same screen as user stakeholders, but instead of selecting a user, a name is entered in the “Non-User Name” field. Non-user

stakeholders may have a team. Stakeholder permissions don't make sense for non-user stakeholders.

After creating a stakeholder, goals can be added to it. Both user and non-user stakeholders can have goals. A new goal can be added by clicking the "New" button, which opens the "New Goal" screen, or an existing goal can be added to the stakeholder by clicking the "Find" button and selecting a goal from the goal selector panel. Figure 41 shows the goal selector panel with the selected goal highlighted.

Select Goal			
	Name	Created By	Date Created
	Automated Assistance	rreganjr	2009-01-04
	Collaborative Elicitation of Requirements	rreganjr	2009-01-04
	Discussion of Requirements	rreganjr	2009-01-04
	Easy to Use	rreganjr	2009-01-04
	Flexible Process and Deliverables	rreganjr	2009-01-04
	Improved Understanding	rreganjr	2009-01-04
	Notification of Project Changes	rreganjr	2009-01-04
	Requirements defined in Natural Language	rreganjr	2009-01-04
	Secure data access	rreganjr	2009-01-20
	Work is your own	rreganjr	2009-01-05

Figure 41: Goal Selector

The full goal can be viewed or edited by clicking on the "Edit" link in the first column of the goal table. The goal can be removed from the stakeholder by clicking the "Remove" link as shown in Figure 40. Removing a goal from a stakeholder does not delete it from the system.

Creating and Editing Goals

Goals are used to indicate desired properties, qualities, constraints, features and functions that the system must support. For example, "Easy to Use" is a high-level quality

goal; “Must run on Linux” is a technology constraint; “Support an IBIS style of discussion” is a feature.

Clicking on the “Goals” link in the project navigation tab opens the goal navigator screen shown in Figure 42. The goal navigator has a table with the name of each goal, who created it, and when it was created. If you are authorized to edit goals the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit goals the first column contains a “View” link that opens the goal editor for read-only viewing.

	Name	Created By	Date Created
Edit	Automatted Assistance	rreganjr	2009-01-04 00:00
Edit	Collaborative Elicitation of Requirements	rreganjr	2009-01-04 00:00
Edit	Discussion of Requirements	rreganjr	2009-01-04 00:00
Edit	Easy to Use	rreganjr	2009-01-04 00:00
Edit	Flexible Process and Deliverables	rreganjr	2009-01-04 00:00
Edit	Improved Understanding	rreganjr	2009-01-04 00:00
Edit	Notification of Project Changes	rreganjr	2009-01-04 00:00
Edit	Requirements defined in Natural Language	rreganjr	2009-01-04 00:00
Edit	Secure data access	rreganjr	2009-01-20 00:00
Edit	Work is your own	rreganjr	2009-01-05 00:00

Figure 42: Goal Navigator Screen

Clicking the add button opens the “New Goal” screen shown in Figure 43. Goals have a name, which is a short one-line description, and a text body that is the full description of the goal. The goal name must be unique in a project. After entering a name and text body, click the save button at the bottom of the screen to create the goal. If the name is unique the system creates the goal and adds it to the goal navigator table.

Figure 43: New Goal Screen

If the goal was created from another requirements element such as a stakeholder or use case, then the goal is also added to the goals field of that element. All the entities that refer to a goal are shown in the goal editor’s “Referring Entities” table shown in Figure 44.

Referring Entities:			
	Description	Created By	Date Created
Edit	Project: Requel	rreganjr	2009-01-04
Edit	Stakeholder: Harvard University	rreganjr	2009-01-05
Edit	UseCase: A user logs in to the system	rreganjr	2009-01-20

Figure 44: Goal Referring Entities

Upon saving the goal, the system initiates analysis of the goal as a background task referred to as the “Automated Assistant.” The assistant adds notes and issues to the goal as it analyzes it. During analysis of a goal the assistant identifies unknown or misspelled words; potential “significant terms” to add to the glossary; potential actors that don’t already exist in the project; and complex sentences that may be difficult to understand. You may work on any requirements elements as the assistant analyzes them; the assistant will

not interfere with your work. See the Working with Analysis Issues section for more information about the analysis.

Relations To Other Goals:				
	Relation Type	To Goal	Created By	Date Created
Edit	Supports	Improved Understanding	project	2009-03-23
◀◀ 1 of 1 ▶▶				
Add				

Figure 45: Goal Relations

Goals can have relationships with other goals listed in the goal editor’s “Relations to Other Goals” field, as shown in Figure 45. The relationships can be used to add traceability of goals from high level soft goals to specific functional requirements and as an indication of how goals interact with each other. Goals may support or conflict with each other. Conflicts help to identify places where tradeoffs may be needed, and to document the rationale behind the decisions made. Goal relations are directional, for example if goal “A” supports goal “B” there is a relationship from goal “A” to goal “B”, but not from goal “B” to goal “A”.

The screenshot shows the REQUEL application interface. At the top, there's a navigation bar with 'Edit Account' and 'Logout' buttons. Below that is a main menu bar with 'Projects', 'Goals: Requel', 'Edit Goal: Automatted Assistance', and 'New Goal Relation'. The 'New Goal Relation' tab is currently active. On the left, there's a sidebar titled 'Projects' with a 'New Project' button and a tree view of project components: 'Purchase Order System' and 'Requel' (which is expanded to show 'Stakeholders', 'Goals', 'Stories', 'Actors', 'Use Cases', 'Scenarios', 'Terms', 'Reports', and 'Open Issues'). The main content area is titled 'New Goal Relation'. It contains three dropdown menus: 'From Goal' (set to 'Automated Assistance'), 'To Goal' (empty), and 'Relation' (empty). Below these is a section for 'Annotations' with fields for 'Type', 'Status', 'Must Be Resolved?', 'Text', 'Created By', and 'Date Created'. At the bottom of the form are 'Cancel' and 'Save' buttons. A status bar at the very bottom indicates '◀◀ 1 of 1 ▶▶'.

Figure 46: Goal Relation Editor Screen

Clicking the “Add” button opens the “New Goal Relation” editor as shown in Figure 46. A goal relation has a “From Goal”, a “To Goal”, and the relation type. The editor defaults the “From Goal” selector to the goal in the editor where the add button was clicked. Both the “To” and “From” goal fields list all the goals in the project as possible options as shown in Figure 47. After selecting the “To Goal” and relation type, clicking save creates the relationship and adds it to the goal relations table of the original goal as shown in Figure 45.

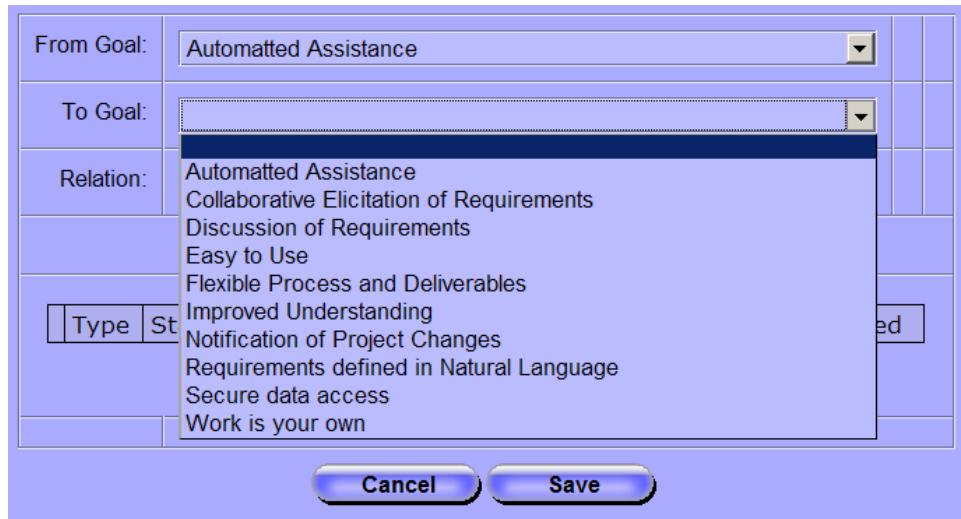


Figure 47: Goal Relation "To Goal" Selection

Creating and Editing Stories

Stories are simple narratives of how a user interacts with the system. Stories can be used to describe how things work before the new system is in place, i.e. the current state of affairs, or how things will be when the new system is in place. Stories can describe negative cases, such as problems with the current system, or how the new system should behave in a bad situation.

Clicking the “Stories” link in the project navigator tab opens the stories navigation screen as shown in Figure 48.

Figure 48: Stories Navigation Screen

The stories navigator has a table with the name, type, who created it, and when it was created. If you are authorized to edit stories the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit stories the first column contains a “View” link that opens the story editor in read-only mode.

Figure 49: Story Editor Screen

Clicking the “Edit” button for an existing story opens the story editor screen shown in Figure 49. A story has a short one-line name, a type of “Success” or “Exception” and a text body. The story name must be unique in the project. Clicking the “Save” button, located at the bottom of the screen, validates and creates or updates the story and initiates analysis as a background task. See the Working with Analysis Issues section for more information about the analysis.

Optionally, stories can have goals and actors assigned. The actors add traceability to the types of users appropriate to the story. The goals can be used to indicate the goals supported by the story or derived from the story. For both goals and actors there are “New” and “Find” buttons as shown in Figure 50. The “New” button is used to open an editor for creating a new actor or goal that gets assigned to the story. The “Find” button is used to open a selector, like the one in Figure 41, to choose an existing goal or actor.

The screenshot shows a software interface titled "Edit Story: Rich creates a new project". At the top left is a link "Stories: Requel". The main area is divided into two sections: "Goals:" and "Actors:". Each section contains a table with columns for Name, Created By, and Date Created. Below each table are "New" and "Find" buttons. The "Goals:" section shows one entry: "Collaborative Elicitation of Requirements" created by "rreganjr" on "2009-01-04". The "Actors:" section shows one entry: "Project User" created by "rreganjr" on "2009-01-05". Navigation arrows are present between the tables.

	Name	Created By	Date Created
Edit Remove	Collaborative Elicitation of Requirements	rreganjr	2009-01-04

	Name	Created By	Date Created
Edit Remove	Project User	rreganjr	2009-01-05

Figure 50: Story Goals and Actors

Creating and Editing Actors

Actors represent the types of users that will interact with the system. Actors may be human users or other systems. Clicking on the “Actors” link in the project navigation tab

opens the actor navigator screen shown in Figure 51. The actor navigator contains a table of the actors with columns for the name, description, who created it, and when it was created. If you are authorized to edit actors the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit actors the first column contains a “View” link that opens the actor editor for read-only viewing.

	Name	Description	Created By	Date Created
Edit	Automated Assistant	The automated assistant is part of the system, but interacts with the requirements part of the system like a project user adding issues and notes during analysis. The automated assistant will be a stakeholder on all projects.	rreganjr	2009-01-05 00:00
Edit	Interactive User		rreganjr	2009-01-20 00:00
Edit	Project User	Project users use the system to create requirements in projects.	rreganjr	2009-01-05 00:00
Edit	System Admin	System administrator users create other users and manage the system level user permissions.	rreganjr	2009-01-05 00:00

Figure 51: Actors Navigator Screen

Clicking the “Add” button opens the “New Actor” editor as shown in Figure 52.

Name:	<input type="text"/>			
Description:	<input type="text"/>			
Goals:				
<table border="1"> <tr> <td>Name</td> <td>Created By</td> <td>Date Created</td> </tr> </table>		Name	Created By	Date Created
Name	Created By	Date Created		

Figure 52: New Actor Screen

An actor has a name that must be unique in the project, and a description. The description can be used to describe the actor’s primary function or to describe an example

“persona” for a person or system that typifies the actor. For example the following is a persona for a user of the Requel system:

Theresa is an accountant for Bailey Pet Supply tasked with specifying the requirements for integrating the purchase order system with the accounting system using the system. Theresa has never used a requirements tool before and has no projects. Theresa will be an end-user of the resulting system.

Actors may have goals that identify the functions of the system relevant to the actor, or general interests that the actor has that the system must meet. Figure 53 shows the goal references of an actor in the Requel system.

Goals:				
	Name	Created By	Date Created	
Edit Remove	Easy to Use	rreganjr	2009-01-04	
Edit Remove	Identify potential actors of the system	project	2009-03-25	
Edit Remove	Identify potential glossary terms	project	2009-03-25	
Edit Remove	Identify sentences that may be difficult to understand	project	2009-03-25	

◀◀ 1 of 1 ▶▶

[New](#) [Find](#)

Figure 53: Referenced Goals

From the actor editor new goals can be added using the “New” button or existing goals can be added using the “Find” button. Goals attached to the actor can be removed by clicking the “Remove” link in the first column of the goals table. Goals that are removed from the actor are not deleted from the project.

Like other requirements element actors are analyzed when saved. See the Working with Analysis Issues section for more information about the analysis.

Creating and Editing Scenarios

Scenarios are semi-structured sequences of steps where the actor invokes an action on the system in one step, followed by a response by the system in the next step. Scenarios are composed of an ordered list of steps where a step may be a single action or a reference to another scenario with its own set of steps. Scenarios may be used as stand alone entities that describe an interaction more explicitly than a story, or in conjunction with a use case.

Clicking on the “Scenarios” link in the project navigator opens the scenarios navigator screen as shown in Figure 54. In addition to the standard name, created by, and date created columns, the scenario navigator has two additional columns: “Top Level” and “Type”. The top level column indicates if a scenario is used as a step in another scenario. Scenarios with a “Yes” value means the scenario is a top level scenario and not used in other scenarios. The type column indicates how the scenario is used, for example if it is used as an alternate, exceptional, or optional sequence of steps in another scenario.

If you are authorized to edit scenarios the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit scenarios the first column contains a “View” link that opens the scenario editor for read-only viewing.

Top Level	Name	Type	Created By	Date Created
Edit Yes	A user creates a new project	Primary	rreganjr	2009-01-20 00:00
Edit Yes	A user logs in to the system	Primary	rreganjr	2009-01-20 00:00
Edit No	The system verifies the project name and customer name are unique, creates the new project, adds the creating user as a stakeholder, and adds the project to the user's list of open projects.	Primary	rreganjr	2009-01-20 00:00
Edit No	The system verifies the user supplied a correct username and password combination, logs the successful login and presents the user with an interface to use the system.	Primary	rreganjr	2009-01-20 00:00
Edit No	The user chooses the help option.	Alternative	rreganjr	2009-01-20 00:00
Edit No	The user enters her username and password and submits them.	Primary	rreganjr	2009-01-20 00:00

◀◀ 1 of 1 ▶▶

[Close](#) [Add](#)

Figure 54: Scenarios Navigation Screen

Clicking the “Add” button opens the “New Scenario” editor as shown in Figure 55.

A scenario has a name, type, optional descriptive text, and a sequence of steps. The scenario name must be unique in a project. The type of scenario is most relevant for scenarios that are steps in other scenarios and is described in the Editing Scenario Steps section, which gives a description of how to add and edit the steps of a scenario.

New Scenario

Name:	<input type="text"/>
Type:	<input type="button" value="Primary"/>
Text:	<input type="text"/>

Figure 55: New Scenario Screen

The scenario editor lists the scenarios that use the scenario being edited in the “Referring Scenarios” table as shown in Figure 56. Clicking the edit link in the table opens the scenario editor for that scenario.

Referring Scenarios:			
	Name	Created By	Date Created
Edit	A user creates a new project	rreganjr	2009-01-20
Edit	A user logs in to the system	rreganjr	2009-01-20

◀◀1 of 1▶▶

Figure 56: Scenario's Referring Scenarios

Creating and Editing Use Cases

Use cases represent functional requirements in terms of how a user interacts with the system. In the Requel system a use case is a confluence of stories, goals, and actors with a scenario to describe the behavior of a specific function of the system.

Clicking on the “Use Cases” link in the project navigator opens the use case navigator screen as shown in Figure 57. The navigator includes columns for the use case name, primary actor, created by username, and date created. If you are authorized to edit use cases the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit use cases the first column contains a “View” link that opens the use case editor for read-only viewing.

Use Cases: Requel				
	Name	Primary Actor	Created By	Date Created
Edit	A user creates a new project	Project User	rreganjr	2009-01-20 00:00
Edit	A user logs in to the system	Interactive User	rreganjr	2009-01-20 00:00

◀◀1 of 1▶▶

[Close](#) [Add](#)

Figure 57: Use Case Navigation Screen

Clicking the “Add” button opens the “New Use Case” editor as shown in Figure 58. A use case has a name, which must be unique in a project, an optional description, and a primary actor. The primary actor input is a combo-box where the actor can be selected from the drop-down list, or a name can be entered. If a name is entered, when the use case is saved, a new actor is created, added to the project, and set as the primary actor. The description and other information of the actor can be edited later, by opening the actor editor through the actor navigation screen.

Figure 58: Use Case Edit Screen

A use case can contain a set of goals that represent what the primary actor is trying to accomplish with the use case. Goals are added and removed from use cases just like goals on other entities, such as on the actor editor screen as shown in Figure 53.

Most use cases only have a single primary actor, but in some cases additional actors are needed to show complex interactions where the system interacts with external services to support the user’s actions. Auxiliary actors are displayed in a table as shown in Figure 59. Like goals, a new actor can be created and added, or an existing actor can be added to the auxiliary actors of the use case.

Actors:			
	Name	Created By	Date Created
Edit Remove	Automated Assistant	rreganjr	2009-01-05
◀◀ 1 of 1 ▶▶			
New	Find		

Figure 59: Use Case Auxiliary Actors

Use cases can reference multiple stories as example of successful and exceptional interactions with the system. Stories are useful as rationale for understanding the scenario of the use case. Stories are displayed in a table as shown in Figure 60 and can be added and removed just like actors and goals.

Stories:			
	Name	Created By	Date Created
Edit Remove	Rich creates a new project	rreganjr	2009-01-20
Edit Remove	Theresa creates a new project	rreganjr	2009-01-20
◀◀ 1 of 1 ▶▶			
New	Find		

Figure 60: Use Case Stories

Editing the scenario of the use case is described in the Editing Scenario Steps section.

Editing Scenario Steps

The scenario step editor is used by both the scenario editor and use case editor screens. When creating a new scenario the steps editor appears with no steps and two buttons as shown in Figure 61.

Scenario Steps:	
Add Step	Add Scenario

Figure 61: Empty Scenario Step Editor

The “Add Step” button creates an empty step node as shown in Figure 62. New steps are added below any existing steps.



Figure 62: Empty Step Node

Each step has a box for editing the text, a drop down list for selecting the type, and a collection of buttons and icons for manipulating the step described in Table 25. The text of the step is limited to 255 characters. The types are pre-condition, primary, alternative, exception, and optional. A step with the pre-condition type isn’t really a step, but indicates a condition that must be met for a scenario to start or continue. The primary type indicates the step is part of the primary flow of the scenario; alternative means the step is an alternative to the primary flow, but part of a successful interaction; exception indicates a problem or error condition; and optional means the step is a successful interaction, but one that is not required to take place.

	The “slotted puzzle” icon is used to drag a step to a new location in the scenario by dropping it on the tabbed puzzle piece. A step can be dragged from and dropped to any sub-step level.
	The “tabbed puzzle” icon is the drop target for moving a step. Dropping a step on the target inserts that step in the location and moves the existing all subsequent steps down by one.
	The “X” button removes the step from the scenario and moves all subsequent steps up by one. If the step is a simple step with no sub-steps it is deleted from the project. If the step is a sub-scenario it is removed, but not deleted.
	The “sub-step” button adds a new sub-step to this step. If the step has other sub-steps the new one is added at the end of the sequence.

Table 25: Step Node Controls

The “Add Scenario” button shown in Figure 61 is used to add an existing scenario as a step in the current scenario. Clicking the button opens the “Select Scenario” screen as shown in Figure 63. Clicking on a scenario selects it and adds it as the last step in the scenario. Note: if the original scenario is a step or sub-step reachable through the selected scenario, then it would result in a cycle in the steps and it is not added.

Name	Top Level	Type	Created By	Date Created
A user creates a new project	Yes	Primary	rreganjr	2009-01-20
A user logs in to the system	Yes	Primary	rreganjr	2009-01-20
The system verifies the project name and customer name are unique, creates the new project, adds the creating user as a stakeholder, and adds the project to the user's list of open projects.	No	Primary	rreganjr	2009-01-20
The system verifies the user supplied a correct username and password combination, logs the successful login and presents the user with an interface to use the system.	No	Primary	rreganjr	2009-01-20
The user chooses the help option.	No	Alternative	rreganjr	2009-01-20
The user enters her username and password and submits them.	No	Primary	rreganjr	2009-01-20

Figure 63: Select Scenario Screen

When editing a scenario with existing steps, the steps are ordered in the sequence they should occur, with sub-steps below and to the right of their parent step as shown at the bottom of Figure 64.

Figure 64: Scenario Steps Editor

Sub-steps can be nested to any level. Each step that contains sub-steps has a “-” button to the left of all other controls when the sub-steps are visible. Clicking it hides the sub-steps. When a step’s sub-steps are hidden the step has a “+” button, which exposes the sub-steps when clicked.

Generating Documents

Once the requirements are in Requel you will want to generate a specification document to provide to the developers to build the system. Just as there are many processes for acquiring requirements, there are many formats for specifying requirements. Requel supports creating custom documentation using the XML Stylesheet Language Transformations (XSLT) language to transform the requirements from an XML format into a specification document. XSLT is a powerful language that can transform XML into a variety of structured or unstructured document formats such as HTML, PDF, and Microsoft Office documents using the Office Open XML format.

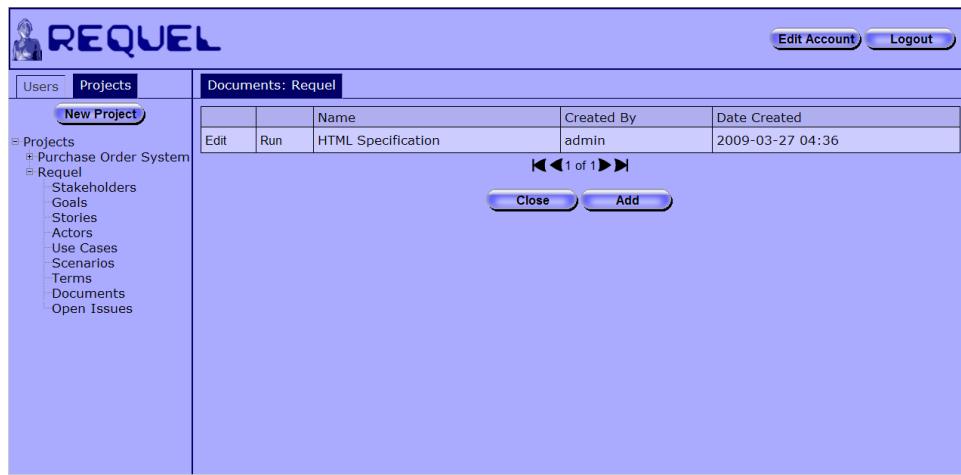


Figure 65: Documents Navigator

Clicking on the “Documents” link in the project navigation opens the documents navigator as shown in Figure 65. Each entry in the table represents an XSLT script for

generating a document. If you are authorized to edit documents (in the stakeholder permissions it is labeled as ReportGenerator) the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit documents the first column contains a “View” link that opens the document editor for read-only viewing.

The second column of the table holds a “Run” link that executes the XSLT script on the requirements to generate a document. The document is returned to the browser, which gives you the option to open it or save it as shown in Figure 66. The document will have the project’s name as the file name with an appropriate extension for the type of document as a suffix. For example, an HTML document for a project named “Requel” will be named “Requel.html”.

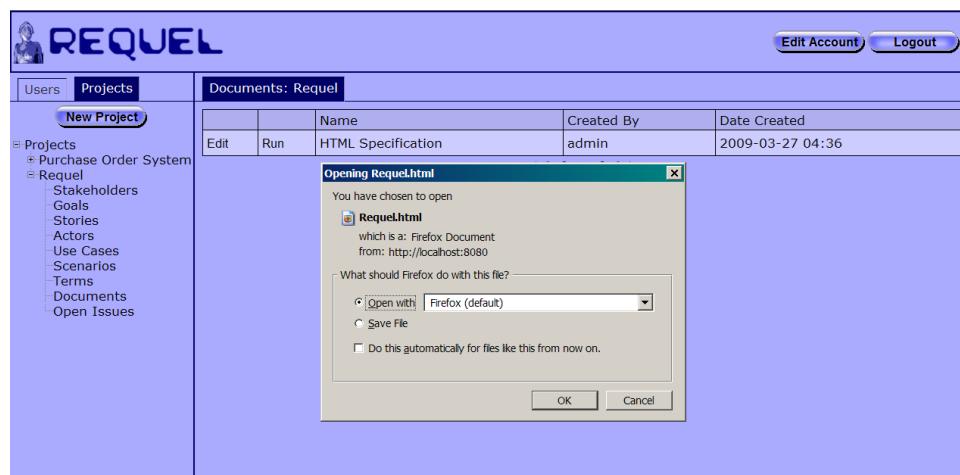


Figure 66: Generating a Document

The built-in “HTML Specification” document is added to all projects when they are created. It generates a simply formatted Web page with all requirements elements as shown in Figure 67. There is a section for each of the requirement element types: stakeholders, actors, goals, stories, use cases and scenarios. At the top of the document is a table of contents that links to each section and also to each element of the requirements.

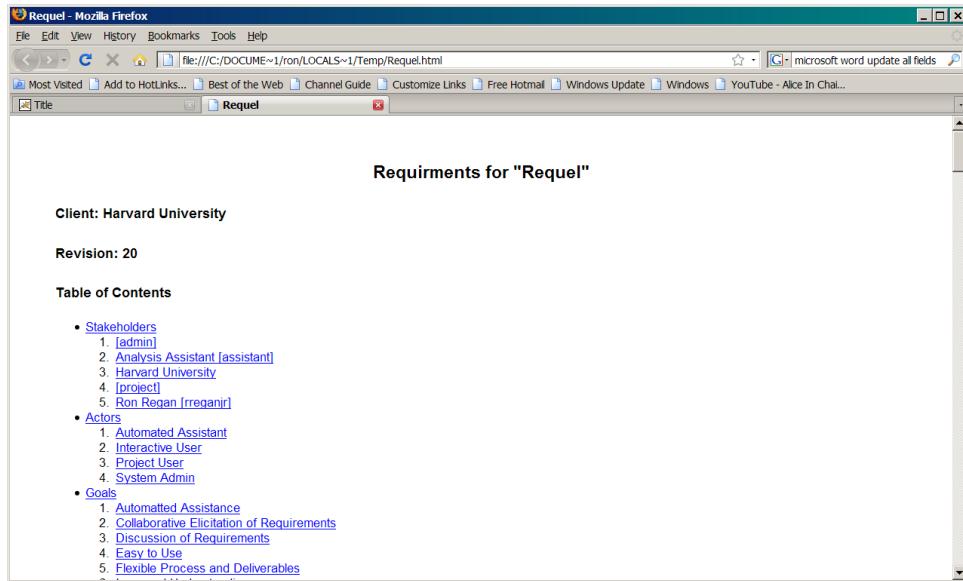


Figure 67: Requirements Document

Adding a Document Generator

Clicking the “Add” button opens the “New Document” editor as shown in Figure 68. A document has a name, which must be unique in the project, and the XSLT script text.

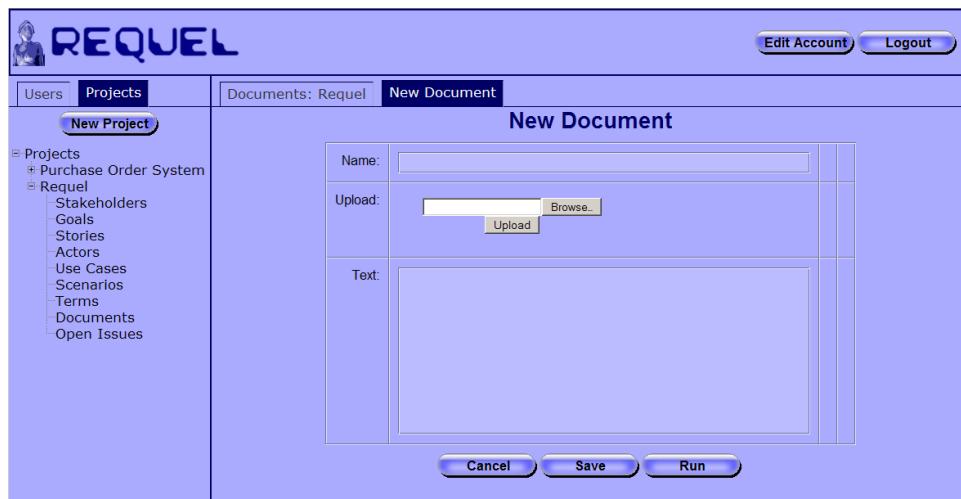


Figure 68: Document Editor Screen

An XSLT can be uploaded from your local system by clicking the “Browse” button, finding the file with the file upload dialog, and then clicking the “Upload” button. The

contents of the XSLT will be loaded in the “Text” field. You can edit the XSLT in the text box, but it is not an effective tool for more than simple text changes.

Working with Terms

Each project has a glossary of terms. Using a glossary to explicitly define how words are used in a project can help improve understanding by using a consistent and explicitly defined set of terms. Clicking on the “Terms” link in the project navigator opens the terms navigator screen shown in Figure 69. If you are authorized to edit terms (in the stakeholder permissions it is labeled as GlossaryTerm) the first column contains an “Edit” link and below the table is an “Add” button. If you are not authorized to edit terms the first column contains a “View” link that opens the term editor for read-only viewing.

	Name	Definition	Canonical Term	Created By	Date Created
Edit	The system	The system under development.		admin	2009-03-27 07:54
Edit	a project	a project encapsulates all the requirements.		admin	2009-03-27 08:22
Edit	the application		The system	admin	2009-03-27 08:40

Figure 69: Terms Navigator

The terms navigator shows the text of the term in the “Name” column and the full text of the definition. The canonical term column contains the name of another term if the term is not the preferred term for the concept. For example the term “the application” in Figure 70 has the term “The system” as the canonical term. Ideally there should only be one term per concept and the system has a feature to solve this problem described below.

Edit	the application	The system	admin	2009-03-27 08:40
------	-----------------	------------	-------	------------------

Figure 70: A Term with a Canonical Term

Clicking the “Edit” button on a term opens the term editor shown in Figure 71. The editor has text fields for the name and description and a selector button for choosing a canonical term.

Edit Term: The system		
Term:	<input type="text" value="The system"/>	
Description:	<input type="text" value="The system under development."/>	
Canonical Term:	<nothing selected> <input type="button" value="Select"/>	
Referring Entities:		
Description	Created By	Date Created

Figure 71: Term Editor

The selector button displays “<nothing selected>” when the term doesn’t have a canonical term. Clicking the “Select” button opens a selector screen that looks like the navigator screen. When a canonical term is selected the name of the term shows up to the left of the button as shown in Figure 72. When creating a new term or changing the name of the term, the system scans all the requirements and adds cross references between the elements and the term.

Canonical Term:	<input type="text" value="The system"/> <input type="button" value="Select"/>	
-----------------	---	--

Figure 72: Canonical Term Selected

As stated before, ideally there should only be one term per concept. In many cases different people use different terms to refer to the same thing when writing requirements

individually. To solve the problem Requel has function that will replace all the occurrences of a term with its canonical term in all the requirement elements. First, all the terms need to be added to the glossary. The automated assistant will probably create issues identifying them as potential glossary terms; resolving these issues will create the terms and add references to all the requirement elements that use them. For all the terms that you want to replace, edit them and select the preferred term as the canonical term. At the bottom of the term editor is a “Replace Term” button shown in Figure 73. Clicking the button causes the system to update all the occurrences of the term in the requirements elements with its canonical term.



Figure 73: Replace Term

Discussing and Negotiating Requirements

This section describes adding notes and issues to requirements, and negotiating resolutions to issues.

Discussion and negotiation are an important aspect of collaboration. For example a stakeholder adds an issue indicating a goal needs more details. Other stakeholders then propose solutions to the issue, and argue the merits of each with arguments supporting or conflicting with the potential solution. Once all the proposed solutions have been discussed, one can be chosen as the resolution.

Adding Notes and Issues

All the requirements elements support adding notes and issues. Each of the requirements editors has an “Annotations” table with the notes and issues as shown in Figure 74. If you are authorized to edit annotations the first column contains an “Edit” link and below the table are “Add Issue” and “Add Note” buttons. If you are not authorized to edit annotations the first column contains a “View” link that opens the note or issue editor for read-only viewing. Note: typically you will want all stakeholders to have “Edit” permission for annotations.

Annotations:							
	Type	Status	Must Be Resolved?	Text	Created By	Date Created	
Edit	Issue	Unresolved	Yes	The phrase "the requirements" is a potential glossary term, actor, or domain object/property.	assistant	2009-03-27	
Edit	Issue	Unresolved	Yes	The text "The system analyzes the requirements as they are added to a project and makes suggestions by adding issues to the elements." in the Text is complex and may be hard to understand.	assistant	2009-03-27	
Edit	Issue	Resolution: Change the word "Automatted" to "automated".	Yes	The word "Automatted" in the "Name" is not recognized and may be spelled incorrectly.	assistant	2009-03-27	

◀◀2 of 2▶▶

[Add Issue](#) [Add Note](#)

Figure 74: Annotations Section

The type column in the table indicates if the annotation is a note or issue. The status column will always have the value “Informational” for notes and either “Unresolved” or “Resolution: ...” for issues. When an issue is resolved the status will start with “Resolution:” followed by the text of the position that resolved the issue. The “Must be Resolved?” column will always have “No” for notes and either “Yes” or “No” for issues. The “Text” column contains the full text of the issue.

Clicking on the “Add Note” button opens the “New Note” editor as shown in Figure 75. A note only has the text of the note.

The screenshot shows the REQUEL application's 'New Note' editor. At the top, there are navigation links for 'Users', 'Projects' (which is selected), 'Terms: Requel', 'Goals: Requel', 'Edit Goal: Automated Assistance', and 'New Note'. Below this is a large text input field labeled 'Note:' containing placeholder text. Underneath is a section titled 'Referring Entities:' with a table. The table has columns for 'Description', 'Created By', and 'Date Created'. It contains one row with the description 'Goal: Automated Assistance', created by 'rreganjr' on '2009-01-04'. Navigation arrows at the bottom indicate this is the first of one entity. At the very bottom are 'Cancel', 'Save', and 'Delete' buttons.

Figure 75: New Note Editor

There isn't a way to find existing notes and add them to multiple requirement elements, but the system automatically detects adding notes with identical text and instead of creating a new note it will add the existing note to the requirement. Figure 76 shows a note shared by two goals, indicated in the “Referring Entities” table.

The screenshot shows the REQUEL application's 'Edit Note' editor. The 'Note:' field contains the text 'this is out of scope for phase 1.'. Below is a 'Referring Entities:' section with a table. The table has columns for 'Description', 'Created By', and 'Date Created'. It contains two rows, both of which are 'Edit' links. The first row is for 'Goal: Automated Assistance' created by 'rreganjr' on '2009-01-04'. The second row is for 'Goal: Discussion of Requirements' also created by 'rreganjr' on '2009-01-04'. Navigation arrows at the bottom indicate this is the first of one entity. At the very bottom are 'Cancel', 'Save', and 'Delete' buttons.

Figure 76: Shared Notes

Clicking on the “Add Issue” button opens the “New Issue” editor as shown in Figure 77. An issue has text, a must be resolved checkbox, and positions. The must be resolved property is only for documentation purposes.

Figure 77: New Issue Editor

After an issue is saved positions can be added by users authorized to edit issues by clicking the “Add” button below the positions table as shown in Figure 78.

Figure 78: Adding Positions

Positions represent potential solutions to an issue. The position editor has a text field to enter the text of the position and a table of arguments that argue for or against the position as a solution to the issue, as shown in Figure 79. After a position is saved arguments can be added by users authorized to edit issues by clicking the “Add” button below the arguments table.

Figure 79: Position Editor

Arguments are the reasons why a particular position of an issue should or should not be chosen as the solution. Arguments have text description and a support level with values from “strongly against” to “strongly for.” Clicking the add or edit button opens the argument editor shown in Figure 80. The argument editor has a field for entering the text of the argument and a drop down list with the support levels.

Figure 80: Argument Editor

All the open issues for all the elements of a project can be seen on the “Open Issues” screen as shown in Figure 81. The screen can be opened from the project navigator

“Open Issues” link. The issues screen has a “View” link for each issue that opens the issue editor. The issues table has a column labeled “Annotatables” that lists out all of the requirements elements that are assigned to the issue.

Figure 81: Open Issues

Working with Analysis Issues

Every time a requirement element is saved the system’s “Automated Assistant” analyses it and adds issues for potential problems or improvements. The issues will concern potential spelling errors or unfamiliar words, possible glossary terms and actors, complex sentences, and poorly structured scenario steps. Issues added during analysis will always have the “assistant” user as the creator as shown in the second column from the right in Figure 82.

Figure 82: Example Analysis Issue

The issues added by the assistant have specialized positions that automate the task described in the position. Figure 83 shows the positions of the issue “The phrase ‘a project’

is a potential glossary term, actor, or domain object” from Figure 82. The position “Ignore this phrase” resolves the issue without taking any action. The position “Add ‘a project’ to the project glossary” adds the term “a project” to the glossary and a reference to all the project elements that are assigned to the issue, when used to resolve the issue.

Positions:					
		Text ▾	Created By	Date Created	
Edit	Resolve	Ignore this phrase.	assistant	2009-03-27	
Edit	Resolve	Add "a project" to the project glossary.	assistant	2009-03-27	
Edit	Resolve	Add "a project" as an actor to the project.	assistant	2009-03-27	

◀◀ 1 of 1 ▶▶

Add

Figure 83: Example Analysis Positions

Table 26 lists the text of the special positions that invoke an automated task when used to resolve an issue.

Position Text	Action taken when the position is used to resolve the issue.
Add "..." to the project glossary.	Creates the term in the glossary and adds a reference to all the project elements that are assigned to the issue.
Add "..." as an actor to the project.	Creates a new actor in the project and adds a reference to all the project elements that are assigned to the issue.
Change the word "..." to "...".	Updates the name and text of all the requirement elements that are assigned to the issue to replace the original text with the new text.
Add "..." to the dictionary.	Adds the word to the systems global dictionary so that future occurrences of the word will not be identified as unknown.
Add the actor “...” to the use case.	Adds the actor to the auxiliary actors of the use case.
Change “...” to the primary actor.	Replaces the text in the scenario step with the name of the primary actor.

Table 26: Automated Task Positions

Requel Setup

This section describes the setup of the Requel system. The system is distributed as a Java Web Archive File (WAR) and requires only minimal configuration to setup.

System Requirements

Requel is a Web application implemented in the Java language and based on the Java Enterprise Edition (JEE5) platform, intended to run in a standard Web container. Requel requires the standard Sun Microsystems Java runtime version 6 update 4 or later to operate properly. Requel has dependencies on Sun's virtual machine and will not work with a third-party virtual machine. Requel is known to work with versions 5.5 and 6.0 of the Apache Tomcat Web container.

Requel stores user and project data in a standard SQL database. It is known to work with MySQL Server version 5.0. The database initialization scripts may have MySQL specific syntax and commands, so using a different database my require converting the SQL syntax of the files.

Create a Database User

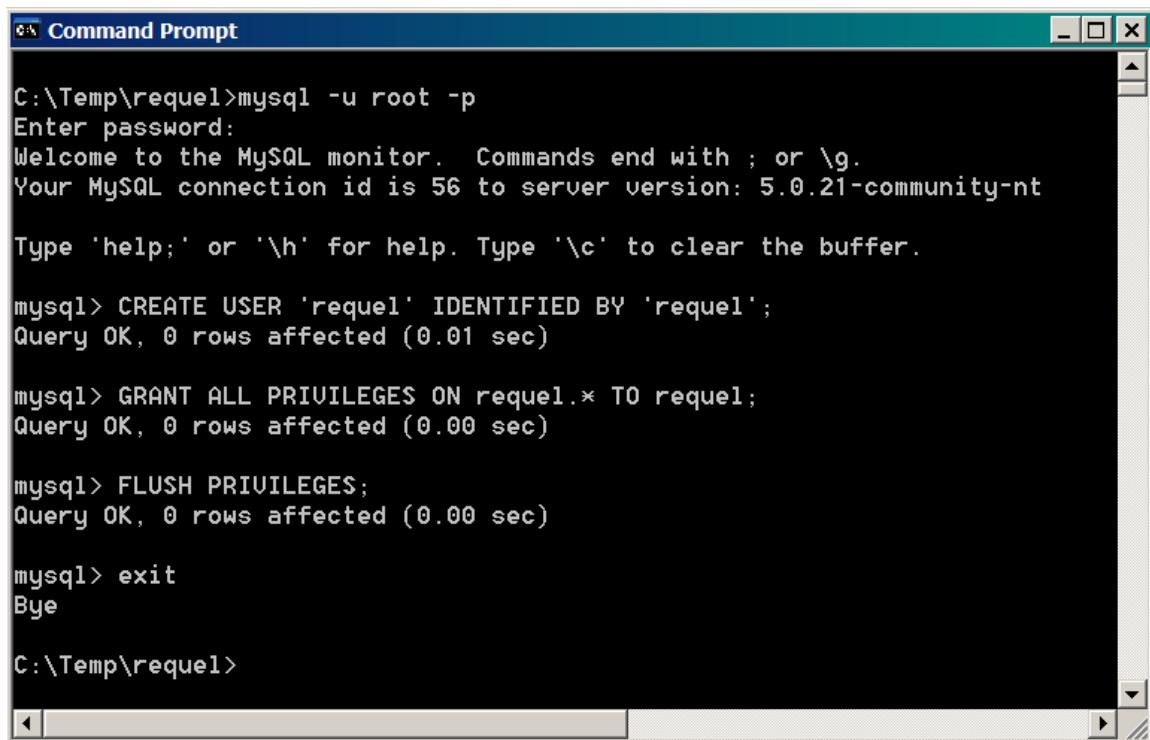
Before Requel can be installed a database user account must be created. Requel will use this account to create and initialize the database when first installed, and to update the requirements and user data during normal operation. It is important that the account has all privileges granted for the database that Requel will use.

The MySQL commands below create a user named ‘requel’ with the password ‘password’ and grants all privileges to that user for all objects in the database named

'requeldb'. The database does not need to be created ahead of time; Requel will create it on the first startup if it doesn't exist.

```
CREATE USER 'reque1' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON requeldb.* TO reque1;
FLUSH PRIVILEGES;
```

Figure 84 shows how to use the MySQL command line interface to login as the root user and create the Requel user. If the Web server and database server are on different physical servers, then in the create user and grant commands append the Web server hostname or IP address to the username. For example, `reque1@192.168.1.100`.



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The window contains the following MySQL commands:

```
C:\Temp\reque1>mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 56 to server version: 5.0.21-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE USER 'reque1' IDENTIFIED BY 'reque1';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON requeldb.* TO reque1;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye

C:\Temp\reque1>
```

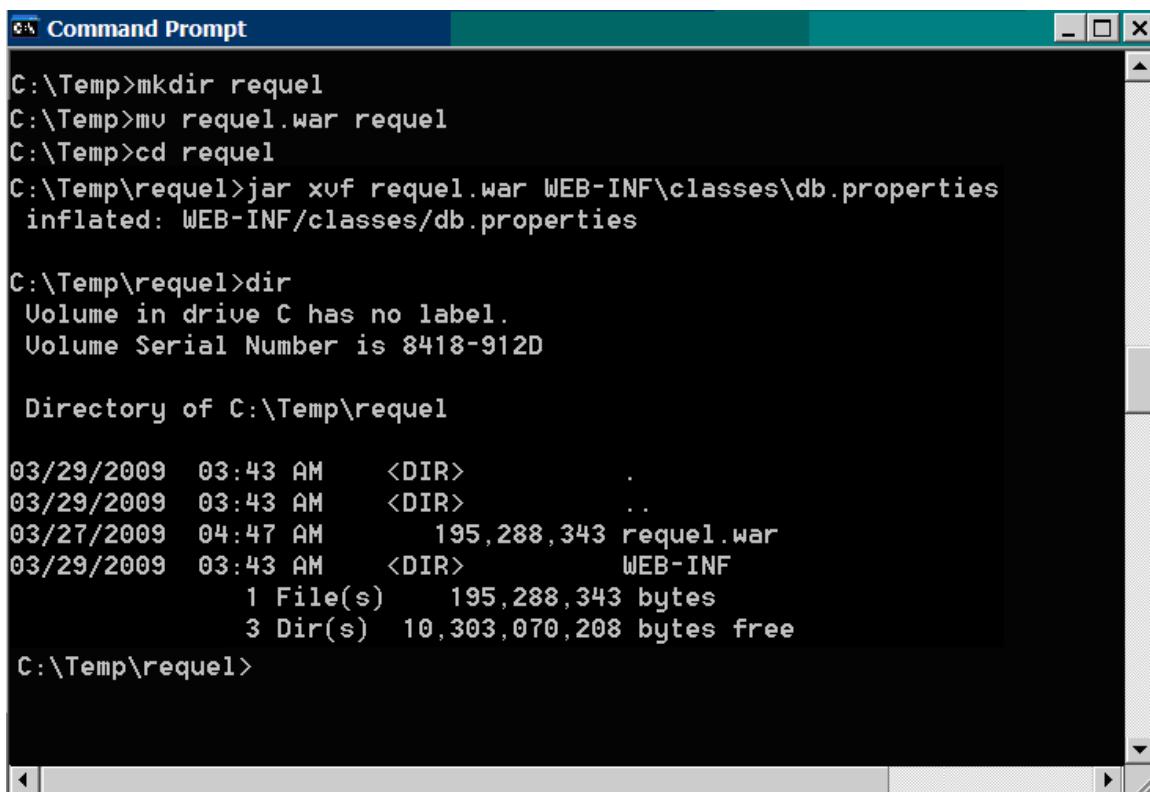
Figure 84: Create the Database User

Configuration

The Requel system requires minimal configuration to get running. The database settings are the only settings that must be configured before the system can be used and they should be configured before installing the WAR file in a Web server.

To configure the database settings the db.properties file must be unpacked from the WAR file. This can be done using the Java jar command as shown in Figure 85. Create a directory to hold the contents of the WAR file and copy the file into that directory. Use the jar command below to extract the file:

```
jar xvf requel.war WEB-INF\classes\db.properties
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The window contains the following command-line session:

```
C:\Temp>mkdir requel
C:\Temp>mv requel.war requel
C:\Temp>cd requel
C:\Temp\requel>jar xvf requel.war WEB-INF\classes\db.properties
inflated: WEB-INF/classes/db.properties

C:\Temp\requel>dir
Volume in drive C has no label.
Volume Serial Number is 8418-912D

Directory of C:\Temp\requel

03/29/2009  03:43 AM    <DIR>    .
03/29/2009  03:43 AM    <DIR>    ..
03/27/2009  04:47 AM    195,288,343 requel.war
03/29/2009  03:43 AM    <DIR>    WEB-INF
                1 File(s)   195,288,343 bytes
                3 Dir(s)  10,303,070,208 bytes free

C:\Temp\requel>
```

Figure 85: Extracting the db.properties file from the WAR File

When the jar command completes there will be a WEB-INF directory. Open the file WEB-INF\classes\db.properties in a text editor such as WordPad in Windows as shown in

Figure 86. You will need to set the db.username, db.password, db.server, db.port, and db.name properties. Note: the Requel database user must have database creation permission if the database specified in the db.name property doesn't already exist.

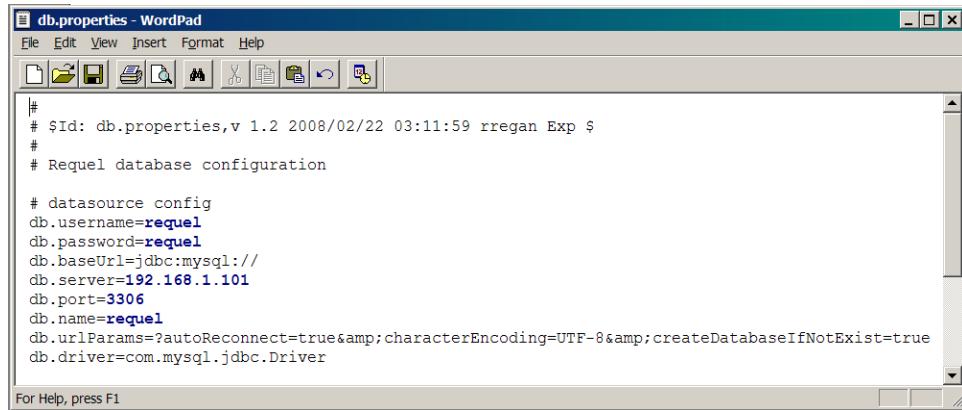


Figure 86: Editing the Database Properties

After saving the properties file, it must be added back to the WAR file. Use the jar command below to add the updated properties file to the war file as shown in Figure 87.

```
jar uvf requel.war WEB-INF\classes\db.properties
```

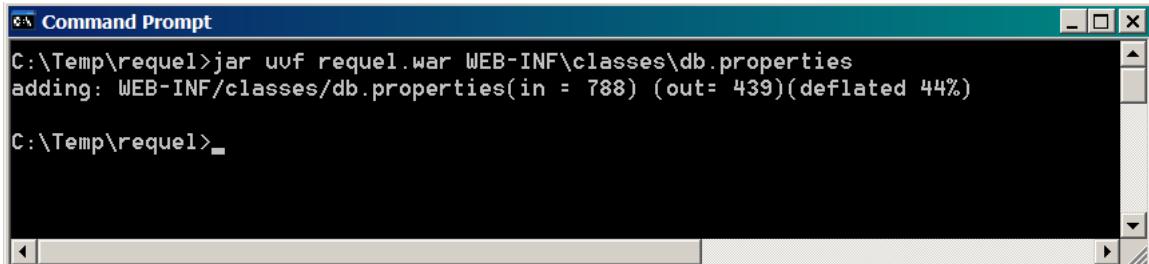


Figure 87: Updating the WAR file

The WAR file is now ready to be deployed to a Web server.

Deploying to Apache Tomcat

Apache Tomcat is a freely available open source JEE Web container. Use the Web application manger that comes with Tomcat to deploy the Requel WAR file. Open a Web

browser to the manager page of the Tomcat server. The server prompts you for a username and password and then displays the manager page as shown in Figure 88.

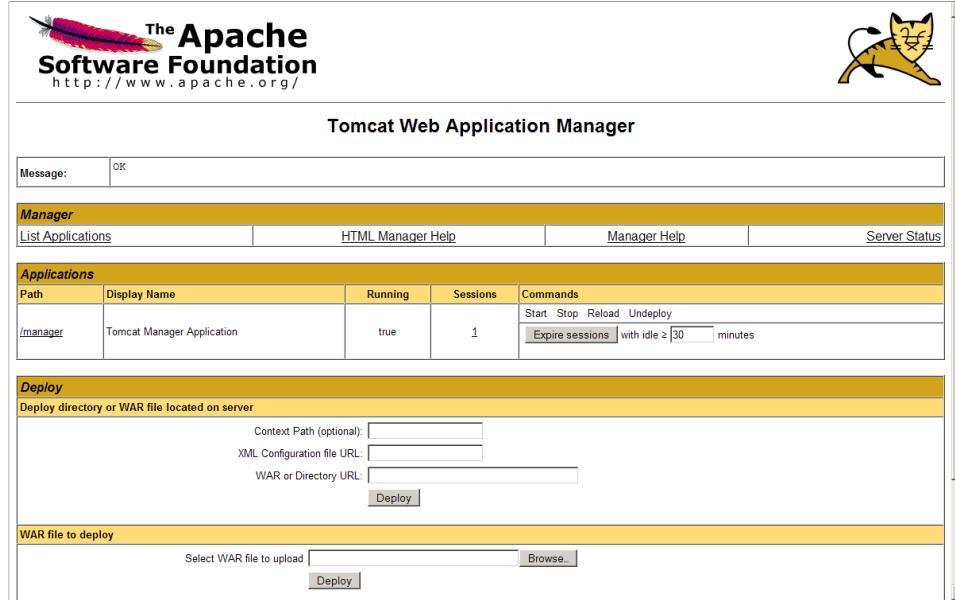


Figure 88: Apache Tomcat Manager

At the bottom of the page is the WAR file deployment form as shown in Figure 89. Use the browse button to locate the Requel WAR file with the updated database properties and then click the Deploy button to upload and install the application. It will typically take from between five and twenty minutes for Requel to create and initialize the database depending on the performance of the database server.

A screenshot of a deployment form titled "WAR file to deploy". It contains a "Select WAR file to upload" input field with a "Browse..." button and a "Deploy" button below it.

Figure 89: WAR File Deployment

When the deployment completes the application manager lists Requel in the applications section as shown in Figure 90. Clicking the “/requel” link in the “Path”

column of the table connects to Requel and the login screen should appear as shown in Figure 27. You should record the URL from the address bar of the Web browser to send to users for accessing the system.

The screenshot shows the Tomcat Web Application Manager interface. At the top, there is a message box containing "OK". Below it is a navigation bar with tabs: "Manager", "List Applications", "HTML Manager Help", "Manager Help", and "Server Status". The main area is titled "Applications" and contains a table with two rows. The first row represents the "Tomcat Manager Application" at path "/manager", which is running with 0 sessions. The second row represents the "requel" application at path "/requel", which is running with 1 session. Both rows have command buttons for Start, Stop, Reload, Undeploy, and an "Expire sessions" button with a value of 30 minutes. The "requel" row is highlighted with a green background.

Figure 90: Requel in the Tomcat Application Manager

Completing the Setup

The first thing to do when the system is ready for access is to login using the built-in administrator account and change the password. The username and password are both set to 'admin' when the system is initialized. See the User Administration section for instructions on changing the password. You should also change the password of the built-in project and assistant users.

The system is now ready to be used. The next step is creating accounts for the users to access the system. See the User Administration section for instructions on creating users.

User Administration

The primary task of system administrators is creating and managing users. System users with the system administrator role are authorized to edit user accounts for all users, and to add new users.

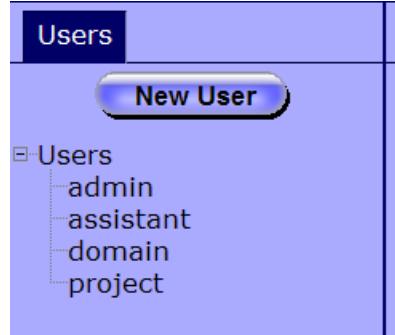


Figure 91: Users Navigation Tab

After logging in the left hand side of the screen will contain a tab labeled “Users” with a “New User” button for creating new users and tree containing all the users of the system as shown in Figure 91. Clicking the new user button opens the “New User” editor as shown in Figure 92. Clicking on a user name in the tree opens that user for editing. Each user must have a username, password, organization, email address, and at least one user role. The name and phone number are optional.

New User

Username:	<input type="text"/>	
Password:	<input type="password"/>	
Retype Password:	<input type="password"/>	
Name:	<input type="text"/>	
Organization:	<input type="text"/>	<input type="button" value="▼"/>
Email Address:	<input type="text"/>	
Phone Number:	<input type="text"/>	
User Roles:	<input type="checkbox"/> ProjectUserRole <input checked="" type="checkbox"/> createProjects <input type="checkbox"/> SystemAdminUserRole	

Cancel **Save**

Figure 92: User Editor

The user name must be unique. If the chosen name is already in use by another user, the system reports an error message to the right of the username field as shown in Figure 93.

Username:	<input type="text" value="project"/>	The username conflicts with an existing User
-----------	--------------------------------------	--

Figure 93: Username Already in Use

The user roles and permissions are used to authorize a user to access different features of the system. The “Project User Role” grants the user access to the project features of the system. The “Create Projects” permission under the project user role grants the user the ability to create new projects. The “System Admin User Role” grants the user access to the user administration function of the system. Figure 94 shows an example of a user with the project user role and permission to create new projects.

User Roles:

<input type="checkbox"/> ProjectUserRole	<input checked="" type="checkbox"/> createProjects	<input type="checkbox"/> SystemAdminUserRole
--	--	--

Figure 94: User Roles and Permissions

Roles and permissions can be changed after a user is created, but a user must always have at least one role. To disable a user's access to the system the password should be reset.

Chapter 5 Summary and Conclusions

This chapter summarizes the successes and failures of the project; the difficulties encountered; the conclusions of the original premise; and the direction for future work.

Lessons Learned

The key lessons learned in the project are:

- An iterative development process based on use cases, supports changing the scope with a minimum of wasted effort.
- Architecture and design patterns can be applied during development to fix structural problems as they are discovered.
- Natural language processing is a difficult task even with all the research and tools available for it.
- The scope of the proposed project was too broad.
- Too many unfamiliar technologies were used and a significant amount of time was spent solving technology specific problems.

The rest of this section elaborates what was learned.

Scope and Process

Identifying the use cases ahead of time and then using an iterative process to implement them, helped to gauge the effort needed to complete the project and allowed for adjustments to the scope as the project progressed. Implementing to use cases also allowed usable pieces of the system to be built and tested incrementally. When a use case was

dropped from the project to adjust the scope, no effort was wasted, because the work done in an iteration stayed focused on the use cases being implemented at that time.

The effort estimates of the good quality use cases, with detailed stories and scenarios were more accurate than the estimates based on incomplete or incorrect use cases. For example, the use cases for editing the requirements elements were implemented approximately on schedule. The scenarios for these use cases had detailed steps that described the required, optional, and exceptional cases. The implementation of these use cases closely matched the behavior described in the use case. The use cases for the automated assistance features took much longer than anticipated and many were dropped. This was partially due to technical problems, but also due to a lack of details or the wrong details in the use cases. For example, one of the use cases was “analyze a story.” In hindsight, this was a poor choice for a use case, because it broke the rule that a use case should define a complete task with expected results. The use case wasn’t defined in enough detail to determine if the implementation met the requirement; it was defined at too high of an abstraction level. A better use case would have been “identify the actors in the story” and it should have had examples of story text and the actors that were expected to be found.

The fact that there were over seventy use cases in the proposal should have been taken as a sign that the scope was too large. It probably would have been better to set a limit on the use cases during the proposal process, to keep the scope manageable.

In summary, I over specified the requirements, but an iterative process based on use cases allowed me to change the scope of the project as it progressed, without wasting effort implementing unnecessary functionality.

Architecture and Design

The architecture and design of the Requel system, as specified in the project proposal was sparse. It mainly consisted of an architectural component model, a class model for the UI Framework, and a few sequence diagrams showing the interaction of the architectural components for the login process. The proposal process had dragged on for more than a year and it was decided to follow a “Just in Time” design process.

A few design decisions were made at the start of development. The first was using the model-view-controller architectural pattern to mitigate the risk of having to change the user interface framework if technical problems arose with the Echo2 framework. The second was using the domain model pattern for modeling the requirements. This pattern was chosen primarily so that the domain objects could be used as the model objects in the user interface, with the expectation that it would make the design simpler. Both the user interface and business logic would use the same object model.

As the project progressed and technical problems were encountered, applicable design and architectural patterns were applied to solve them. For example, the layer and command processor patterns were applied to solve transactional issues between the user interface and domain components. Then the domain objects were wrapped in proxies to solve stale data and Hibernate lazy loading problems. Finally, the command processor handler was decorated to solve a database concurrency problem when a user and the automated assistant analysis were accessing the same objects.

The choice to use a particular pattern almost dictated the use of other patterns. For example, the use of the domain model pattern makes using the repository pattern almost a necessity. Technology choices also drive or are driven by pattern choices. For example,

object-relational mapping tools and the domain model pattern have a high affinity to each other.

In summary, I learned that the architecture doesn't need to be chosen before development begins. Applying patterns as needed to solve problems can result in a system with a high quality architecture.

Technology

This section describes the experiences with some of the technologies used to implement the Requel system.

Spring Framework

The Spring Framework offers a lot of functionality with a minimum of invasiveness in the code of a system. It makes the configuration and integration of the system components simple. A lot of code wasn't needed to be written to gain the benefits of using the framework. Some effort was spent up front learning how to use the framework, but I believe over the life of the project, it saved more time than it cost.

The basic concepts of Aspect-Oriented Programming (AOP) are fairly straight forward, although the terminology and intricacies of AspectJ, which is the basis for the Spring Framework implementation, were difficult to decipher. The Spring Framework implementation of AOP is proxy based and doesn't support the full features of AspectJ, which caused some problems getting the desired behavior of the entity proxies. Code was needed to simulate the unsupported control-flow based pointcut constraints, so that the entities returned by repository methods weren't proxied when called from within the execution of a command. Using AOP made the implementation of the entity proxy

completely separate from the domain model. I don't think I could have implemented the entity proxy functionality as cleanly without using AOP features.

Overall I had a positive experience using the Spring Framework.

Java Architecture for XML Binding

The promise of JAXB is the seamless mapping and transport of Java object to XML and vice-versa. The mapping capabilities are quite robust and the only short-coming encountered was the inability to have a referenced object embedded in one referring element, while having references to it embedded in other elements. For example, have a shared position object embedded in one issue, while the other issues only had a reference to it.

Other aspects of JAXB had issues that made it a little hard to work with. The schema generator modeled the XML document exactly to the Java implementation. For example, a class hierarchy where classes extended other classes would be modeled in XML schema as separate types. This made the ordering of elements in the XML document confusing if the XML needed to be edited directly. JAXB supports a mechanism for specifying the order of elements, but only for properties defined in a class and not inherited properties.

Dealing with Java interfaces is cumbersome with JAXB. Classes that referred to other classes through an interface required writing adapters that cast the objects between the implementing classes and their interfaces. It seems that there should be a way to describe this mapping declaratively instead of having to write code.

The biggest short coming was deficiencies in the Unmarshaller callback mechanism, for pre and post processing of objects as they are loaded. I would expect many applications using XML to import and export data, would also be persisting that data to a database. The case where an object in the XML document refers to an existing object in the database would be common, and require the ability to replace new object instances created during unmarshalling with existing persistent instances from the database. As described in 3it required significant effort to implement this capability.

The performance of JAXB was sufficient for importing and exporting the test requirements because the files were relatively small. Trying to use JAXB to import and export the WordNet data was unsuccessful because the huge amount of data made it too slow to be usable.

The final issue encountered was due to JAXB being bundled with the Java runtime. JAXB is being actively developed and enhanced, which is good, but by having it bundled in the runtime, caused deployment problems because a library bundled with the Java runtime requires special handling to overload with a different version. Requiring users of the system to alter the Java platform to support Requel may impact other applications in undesirable ways.

In summary, JAXB supports complex mapping between Java and XML and the declaration of the mapping as annotations has a time saving benefit, but there are still some kinks that need to be worked out.

Echo2 Framework

The decision to use the Echo2 framework was born out of frustrations using JSP and the desire to use Java for all parts of the implementation. Echo2 has an API very similar to Java's Swing framework, although it is clear to anyone that has used both APIs that there are many differences. Echo2 has restrictions on how container components can be nested in other container components. There are no layout-managers in Echo2; layouts are determined by special container components. Echo2 has a lot of layout and style restrictions based on the underlying Web browser capabilities. Both frameworks have threading issues to deal with, although they are for different reasons and require different approaches to solve.

The biggest problem using Echo2 for this project was that it is a low level API that focuses on components and offers no higher level features like navigation and panel management. It required a huge up front effort to create a UI framework on top of Echo2 to manage navigation and panels. There are also some quality issues with some of the components, such as the tab set. Panels with long titles take up the whole tab section, and the tabs extend off the screen if many panels are opened. Writing new components in Echo2 that aren't just composites of existing components is difficult and requires a mix of Java and Javascript.

On the positive side, Echo2 supports the creation of rich and very interactive applications comparable to desktop applications. The performance of Echo2 is quite fast and even when accessing an application over the Internet it feels like a local application.

In summary, using Echo2 allowed Requel to have a nice looking and interactive user interface, but it required a lot of effort to achieve.

WordNet SQL Builder

The WordNet SQL Builder was helpful mainly because it integrated the WordNet, Extended WordNet, and VerbNet data into a SQL database. WordNet is a text-based database; Extended WordNet and VerbNet are XML databases. A Java APIs exist for WordNet, but not for Extended WordNet or VerbNet. Writing code to access the Extended WordNet and VerbNet from XML and cross-referencing it with WordNet data would have been a huge undertaking.

The downside is that WordNet SQL Builder is only a database, (technically it is a Java tool to create the database, but I used a preloaded database,) and I had to implement a Java API to use it. The tables are in a highly normalized form that made it complicated to map to Java, particularly the associations between objects like words, senses, and synonym sets. I also had to make some extensions to cross-reference some of the VerbNet data with WordNet senses.

In summary, using WordNet SQL Builder simplified the integration of the different lexical databases and saved a lot of effort even though I had to implement a Java API to use it.

Natural Language Processing

Natural language processing is hard and the capabilities required by the project were not met. There are a lot of open source tools and databases, but many are standalone tools and not easy to integrate. An integration layer had to be created so that the information provided by different natural language tools could be used together. Creating this layer was a significant undertaking, although using the blackboard architectural pattern

made trying different tools easier than it would have been if the Requel components were implemented directly to the individual tools.

I was unable to find adequate tools for word sense disambiguation and semantic role labeling and had to create my own. The word sense disambiguation component performed worse than the standard baseline measure of always choosing the most commonly used sense (Jurafsky & Martin, 2008, p. 644.) While the semantic similarity function worked correctly on a small sample set of nouns, it only disambiguates words of the same part of speech and does much better at nouns than other parts of speech. The co-reference disambiguation using SemCor sentences and WordNet glosses didn't improve the results at all. The only way this method seems like it would work is to use a large domain specific sense tagged corpus.

The VerbNet based semantic role labeler performed correctly on a small sample set with simple sentences and explicitly specified word senses, but was ineffective on actual requirements. This was partially due to the poor performance of the word sense disambiguation, but mainly due to the limited verb coverage and syntax to semantic role mappings in VerbNet.

Conclusion

The original premise of the project was that writing high quality requirements is difficult and a tool that supports collaboration among the stakeholders and automated assistance would make it easier. Unfortunately I never got to test the theory with a team of stakeholders so I don't know if the discussion feature will improve collaboration.

Due to the difficulties with natural language processing the scope of automated assistance ended up being small. Identifying glossary terms and being able to replace duplicate terms for the same concept in the requirements text are probably the most useful. Despite the difficulties I encountered, I do believe that it is possible to implement automated assistance that would be useful, although not at the level of semantic understanding I originally anticipated.

I learned a lot about engineering a large scale project, the benefits of various architectural and design patterns, and the intricacies of a lot of new technologies. Aside from the analysis short-coming, Requel offers a lot of features for creating and elaborating requirements.

Future Work

This section touches on aspects of Requel that could be improved with some ideas on things to try.

Natural Language Processing

The word sense disambiguation and semantic role labeling tools need a lot of improvement. Using VerbNet may improve the word sense disambiguation; the verb sense may be found by determining which syntax frame best matches a given sentence. The selection restrictions on the types of nouns that fill the semantic roles in VerbNet may help narrow down which senses of a noun are appropriate.

The semantic role labeling may be improved by adding new frames to VerbNet and by trying to do partial matches between the syntax frames and sentences.

Domain Knowledge

The original scope of the project included a domain component with knowledge about concepts relevant to an application domain. At a minimum adding a domain lexicon of words and senses could improve the natural language processing performance.

On the requirements engineering side, having a domain specific glossary and example requirement elements could help jump start projects.

User Interface

Using the Echo2 framework for the user interface required a significant amount of effort. It would be interesting to try a framework like Flash or JavaFX to see if it would be more productive.

References

- Adolph, S., Cockburn, A., and Bramble, P. (2002) Patterns for Effective Use Cases. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
- Alexander, I. (2007) Requirements Tools Web page Available at <http://easyweb.easynet.co.uk/~iany/other/vendors.htm>
- Banerjee, S. (2002) Adapting the Lesk Algorithm for Word Sense Disambiguation to WordNet (Masters Dissertation, University of Minnesota, 2005) Retrieved June 4, 2008 from <http://www.d.umn.edu/~tpederse/Pubs/banerjee.pdf>
- Brooks, F. (1987) No Silver Bullet Essence and Accidents of Software Engineering. Computer 20, 4 (Apr. 1987), 10-19.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996) Pattern-oriented software architecture: a system of patterns. New York: John Wiley & Sons, Inc.
- Ciemniewska, A., Jurkiewicz, J., Olek, Ł., and Nawrocki, J. (2007) Supporting Use-Case Reviews. Business Information Systems vol. 4439/2007. Springer Berlin: Springer-Verlag. Draft Retrieved October 1, 2007 from http://www.cs.put.poznan.pl/olek/homepage/Research_files/07-BIS.pdf
- Christel, M. G. and Kang, K. C. (1992) Issues in Requirements Elicitation. Technical Report (CMU/SEI-92-TR-12) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University Retrieved July 9, 2007 from <http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr12.92.pdf>
- Cockburn, A. (2000). Writing Effective Use Cases. Boston, MA: Addison-Wesley.
- Conklin, J. and Begeman, M. L. (1988). gIBIS: a hypertext tool for exploratory policy discussion. ACM Trans. Inf. Syst. Vol. 6, issue 4 pp. 303-331. Retrieved September 12, 2007 from <http://www.cs.hut.fi/Opinnot/T-93.850/2005/Papers/gIBIS1988-conklin.pdf>
- Dražan, J. and Mencl, V. (2007). Improved Processing of Textual Use Cases: Deriving Behavior Specifications. In Proceedings of the 33rd Conference on Current Trends in theory and Practice of Computer Science. J. Leeuwen, G. F. Italiano, W. Hoek, C. Meinel, H. Sack, and F. Plášil (Eds.) Lecture Notes In Computer Science, vol. 4362. Berlin: Springer-Verlag. Retrieved February 2, 2008 from <http://dsrg.mff.cuni.cz/publications/DrazanMencl-ImprovedUC-SOFSEM2007.pdf>

Eberlein, A. (1997) Requirements Acquisition and Specification for Telecommunication Services. PhD thesis, University of Wales, Swansea, UK. Retrieved August 3, 2007 from
http://www2.enel.ucalgary.ca/People/eberlein/publications/PhD_Thesis.pdf

Fellbaum, C. (Ed.). (1998) WordNet: An Electronic Lexical Database, Cambridge, MA: MIT Press.

Finkelstein, A. (1994) Requirements Engineering: a review and research agenda. In Proceedings of the First Asia Pacific Conference on Software Engineering, New York, NY: IEEE CS Press, pages 10-19.

Fowler, M. (2002) Patterns of Enterprise Application Architecture. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Fowler, M. (2004) Inversion of Control Containers and the Dependency Injection pattern. Retrieved February 8, 2008 from <http://martinfowler.com/articles/injection.html>

Gale, W. A., Church, K. W., and Yarowsky, D. (1992) One sense per discourse. In Proceedings of the Workshop on Speech and Natural Language. Morristown, NJ: Association for Computational Linguistics. Retrieved March 6, 2009 from
<http://acl.ldc.upenn.edu/H/H92/H92-1045.pdf>

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Goncalves, A. (March 2007) Generate an XML Document from an Object Model with JAXB 2. DevX website article Retrieved January 22, 2008 from
<http://www.devx.com/Java/Article/34069>

Johnson, R., Hoeller, J., Arendsen, A., Sampaleanu, C., Harrop, R., Risberg, T., et al. (2007.) The Spring Framework - Reference Documentation Version 2.5. Retrieved February 8, 2008 from <http://static.springframework.org/spring/docs/2.5.x/spring-reference.pdf>

Jurafsky, D., and Martin, J. (2008). Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. Upper Saddle River, NJ: Prentice-Hall.

Kipper-Schuler, K. (2005) VerbNet: A broad-coverage, comprehensive verb lexicon. (Doctoral Dissertation, University of Pennsylvania, 2005) Retrieved March 6, 2008 from <http://verbs.colorado.edu/~kipper/Papers/dissertation.pdf>

Klein, D., and Manning, C. (2003) Fast Exact Inference with a Factored Model for Natural Language Parsing. In Advances in Neural Information Processing Systems 15

(NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10. Retrieved January 30, 2007 from <http://www-nlp.stanford.edu/~manning/papers/lex-parser.pdf>

Lesk, M. (1986) Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In V. DeBuys, (ed.) Proceedings of the 5th Annual international Conference on Systems Documentation. SIGDOC '86. New York: ACM Press, pp. 24-26.

Mencl, V. (2004) Deriving Behavior Specifications from Textual Use Cases, in Proceedings of Workshop on Intelligent Technologies for Software Engineering. Linz, Austria: Oesterreichische Computer Gesellschaft. pp. 331-341. Retrieved January 23, 2007 from <http://dsrg.mff.cuni.cz/publications/Mencl-DerivingBehSpec-WITSE04.pdf>

Mihalcea, R. (1998) SEMCOR - Semantically tagged corpus. Retrieved February 4, 2008 from <http://www.seas.smu.edu/~sanda/research/semcor.ps.gz>

Mylopoulos, J. and Castro, J. (2000) Tropos: A Framework for Requirements-Driven Software Development. In Brinkkemper, J. and Solvberg, A. (eds.), Information Systems Engineering: State of the Art and Research Themes. Lecture Notes in Computer Science, Springer-Verlag. Retrieved August 22, 2007 from http://www.troposproject.org/papers_files/Fossil.pdf

Nawrocki J., Olek Ł. (2005) Use-cases engineering with UC Workbench, in Zieliński, K. and Szmuc, T. (eds.), Software Engineering: Evolution and Emerging Technologies, volume 130, pages 319–329. Amsterdam: IOS Press. Retrieved October 1, 2007 from http://www.cs.put.poznan.pl/olek/homepage/Research_files/UCWorkbench.pdf

Nuseibeh, B. and Easterbrook, S. (2000) Requirements Engineering: A Roadmap. In A. Finkelstein (ed.) The Future of Software Engineering (pp. 35-46). (Companion volume to the proceedings of the 22nd International Conference on Software Engineering, ICSE'00). Los Alamitos, CA: IEEE Computer Society Press. Retrieved July 14, 2007 from <http://www.cs.toronto.edu/~sme/papers/2000/ICSE2000.pdf>

Ort, E. and Mehta, B. (March 2003) Java Architecture for XML Binding (JAXB) from Sun Developer Network website Retrieved January 22, 2008 from <http://java.sun.com/developer/technicalArticles/WebServices/jaxb/>

Potts, C., Takahashi, K., and Antón, A. (1994) Inquiry-Based Requirements Analysis. IEEE Software vol. 11-2 pp. 21-32. Retrieved July 14, 2007 from <http://www4.ncsu.edu/~ajanton/pubs/ieeeSW.pdf>

Reubenstein, H. B. and Waters, R. C. (1991) The Requirements Apprentice: Automated Assistance for Requirements Acquisition. IEEE Transactions on Software Engineering vol. 17 no. 3, pp. 226-240

Rich, C., and Waters, R. (1986). Toward a Requirements Apprentice: On the Boundary between Informal and Formal Specifications. (AI Memo AIM-907). Cambridge, MA: MIT, Computer Science and Artificial Intelligence Lab. Retrieved April 30, 2007 from <http://hdl.handle.net/1721.1/5516>.

Rich, C., and Waters, R. (1987). The Programmer's Apprentice Project: A Research Overview. (AI Memo AIM-1001). Cambridge, MA: MIT, Computer Science and Artificial Intelligence Lab. Retrieved June 15, 2007 from <http://hdl.handle.net/1721.1/6054>.

Rich, C., and Waters, R. (1990). *The Programmer's Apprentice*. New York, NY: ACM Press.

Robinson, W. N. and Volkov, S. (1997) A meta-model for restructuring stakeholder requirements. In Proceedings of the 19th international Conference on Software Engineering. ICSE '97. ACM Press, New York, NY, 140-149. Retrieved August 8, 2007 from <http://www.cis.gsu.edu/~wrobinso/papers/icse97.ps>

Ruppenhofer, J., Ellsworth, M., Petrucci, M., Johnson, C., and Scheffczyk, J. (2006) FrameNet II: Extended Theory and Practice Retrieved January 25, 2008 from http://framenet.icsi.berkeley.edu/index.php?option=com_wrapper&Itemid=126

Schuler, K. 2005 VerbNet: a Broad-Coverage, Comprehensive Verb Lexicon. Doctoral Thesis. University of Pennsylvania. Retrieved January 25, 2008 from <http://verbs.colorado.edu/~kipper/Papers/dissertation.pdf>

Seco, N., Veale T., and Hayes, J. (2004) An Intrinsic Information Content Metric for Semantic Similarity in WordNet. Technical Report. Dublin, Ireland: University College Dublin Retrieved October 8, 2008 from <http://afflatus.ucd.ie/Papers/ecai2004b.pdf>

Sharp, H., Finkelstein, A., and Galal, G. (1999) Stakeholder Identification in the Requirements Engineering Process. In Proceedings of the 10th international Workshop on Database & Expert Systems Applications, IEEE Computer Society, Washington, DC. Draft Retrieved September 5, 2007 from <http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/stake.pdf>

Simpson, T., and Dao, T. (2005, Oct 1). WordNet-based semantic similarity measurement. Retrieved April 4, 2008 from <http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx>

Smith J. D. (1997) EColabor: Collaborative Elaboration of Documents. NTT Multimedia Communications Laboratories. Berkeley Multimedia and Graphics Seminar. Retrieved July 3, 2007 from <http://bmrc.berkeley.edu/courseware/cs298/spring97/w5/slides.ppt>

Takahashi, K., Potts, C., Kumar, V., Ota, K., and Smith, J. D. (1996) Hypermedia Support for Collaboration in Requirements Analysis. In Proceedings of the 2nd international Conference on Requirements Engineering (ICRE '96) (April 15 - 18, 1996). ICRE. IEEE Computer Society, Washington, DC, 31.

van Lamsweerde, A. (2001) Goal-Oriented Requirements Engineering: A Guided Tour Retrieved July 15, 2007 from
<http://www.info.ucl.ac.be/Research/Publication/2001/RE01.pdf>

Weigert, K. E. (1999). Software Requirements. Redmond, WA: Microsoft Press.

Yu, E. S. (1997) Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In Proceedings of the 3rd IEEE international Symposium on Requirements Engineering (Re'97) (January 05 - 08, 1997). RE. IEEE Computer Society, Washington, DC, 226. Draft Retrieved August 21, 2007 from
<http://www.cs.toronto.edu/pub/eric/RE97.pdf>

Appendix 1 Application Code

Java Code

abstractannotation.java

```
/*
 * $Id: AbstractAnnotation.java,v 1.38 2009/02/16 10:10:07 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import java.io.Serializable;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.Lob;
```

```
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Any;
import org.hibernate.annotations.AnyMetaDef;
import org.hibernate.annotations.ManyToOne;
import org.hibernate.annotations.MetaValue;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.impl.ActorImpl;
import edu.harvard.fas.rregan.requel.project.impl.GlossaryTermImpl;
import edu.harvard.fas.rregan.requel.project.impl.GoalImpl;
import edu.harvard.fas.rregan.requel.project.impl.GoalRelationImpl;
import edu.harvard.fas.rregan.requel.project.impl.ProjectImpl;
import edu.harvard.fas.rregan.requel.project.impl.ProjectTeamImpl;
import edu.harvard.fas.rregan.requel.project.impl.ScenarioImpl;
import edu.harvard.fas.rregan.requel.project.impl.StakeholderImpl;
import edu.harvard.fas.rregan.requel.project.impl.StepImpl;
import edu.harvard.fas.rregan.requel.project.impl.StoryImpl;
import edu.harvard.fas.rregan.requel.project.impl.UseCaseImpl;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBAnnotationGroupedByPatche
r;
```

```

import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "annotations")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "annotation_type", discriminatorType =
DiscriminatorType.STRING, length = 255)
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public abstract class AbstractAnnotation implements Annotation,
Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private Object groupingObject;
    private String text;
    private String type;
    private Set<Annotatable> annotatables = new HashSet<Annotatable>();
    // private Set<Annotation> annotations = new TreeSet<Annotation>();
    private User createdBy;
    private Date dateCreated = new Date();
    private int version = 1; // start at 1 so hibernate recognizes the
new

    // instance as the initial value and not stale.

    protected AbstractAnnotation(String type, Object groupingObject,
String text, User createdBy) {
        setType(type);
        setGroupingObject(groupingObject);
        setText(text);
        setCreatedBy(createdBy);
        setDateCreated(new Date());
    }

    protected AbstractAnnotation() {
        // for hibernate
    }

    @Override
    @Transient
    public String getDescription() {
        return getTypeName() + ":" + getId();
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @XmlID
    @XmlAttribute(name = "id")
    @XmlJavaTypeAdapter(IdAdapter.class)
    public Long getId() {
        return id;
    }

    protected void setId(Long id) {
        this.id = id;
    }

    @Column(name = "annotation_type", insertable = false, updatable =
false)
    protected String getType() {
        return type;
    }

    protected void setType(String type) {
        this.type = type;
    }

    /**
     * @return An object used as the "owner" of a group of annotations.
     */
    @Any(metaColumn = @Column(name = "grouping_object_type", length =
255), fetch = FetchType.LAZY, optional = false)
    @AnyMetaDef(idType = "long", metaType = "string", metaValues =
{ @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class) })
    @JoinColumn(name = "grouping_object_id")
    @XmlTransient
    public Object getGroupingObject() {
        return groupingObject;
    }

    // this needs to be public for JAXB import
    public void setGroupingObject(Object groupingObject) {
        this.groupingObject = groupingObject;
    }

    @Version
    protected int getVersion() {
}

```

```

    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

/**
 * @return the entity that is annotated by this annotation.
 */
// TODO: it would be better if this wasn't dependent on the classes
being
// mapped.
@ManyToOne(fetch = FetchType.LAZY, metaColumn = @Column(name =
"annotatable_type", length = 255, nullable = false))
@AnyMetaDef(idType = "long", metaType = "string", metaValues = {
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.annotation.Annotation", targetEntity =
AbstractAnnotation.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.ProjectTeam", targetEntity =
ProjectTeamImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Goal",
targetEntity = GoalImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.GoalRelation", targetEntity =
GoalRelationImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCase",
targetEntity = UseCaseImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Scenario", targetEntity =
ScenarioImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Step",
targetEntity = StepImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Story",
targetEntity = StoryImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Actor",
targetEntity = ActorImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.GlossaryTerm", targetEntity =
GlossaryTermImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Stakeholder", targetEntity =
StakeholderImpl.class) })

```

```

@JoinTable(name = "annotation_annotatable", joinColumns =
{ @JoinColumn(name = "annotation_id") }, inverseJoinColumns =
{ @JoinColumn(name = "annotatable_id") })
public Set<Annotatable> getAnnotatables() {
    return annotatables;
}

protected void setAnnotatables(Set<Annotatable> annotatables) {
    this.annotatables = annotatables;
}

@Lob
@XmlElement(name = "text", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

@Override
public int compareTo(Annotation o) {
    AbstractAnnotation other = (AbstractAnnotation) o;
    int typeCompare = (getType() == null ? -1 :
getType().compareTo(other.getType()));
    return (typeCompare != 0 ? typeCompare :
getText().compareTo(other.getText()));
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    final int prime = 31;
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            int result = 1;
            result = prime * result + ((getType() == null) ? 0 :
getType().hashCode());
            result = prime * result + ((getText() == null) ? 0 :
getText().hashCode());
            tmpHashCode = new Integer(result);
        }
    }
    return tmpHashCode;
}

```

```

}

return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final AbstractAnnotation other = (AbstractAnnotation) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getText() == null) {
        if (other.getText() != null) {
            return false;
        }
    } else if (!getText().equals(other.getText())) {
        return false;
    }
    return true;
}

@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
public User getCreatedBy() {
    return createdBy;
}

protected void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

/**
 * This is for JAXB to patchup the type
 *
 * @see UnmarshallerListener
 */
public void beforeUnmarshal() {
    setType(getClass().getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship and swap
 * the
 * creator with an existing user.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
 * supplied.
 * @param annotatable
 * @see UnmarshallerListener
 */
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser,
    Object annotatable) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
    defaultCreatedByUser));
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBAnnotationGroupedByPatcher(this, (Annotatable)
annotatable));
}

/**
 * This class is used by JAXB to convert the id of an entity into an
 * xml id
 * string that will be distinct from other entity xml id strings by
 * the use
 * of a prefix.
 *
 * @author ron

```

```

/*
@XmlTransient
protected static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "ANN_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

abstractannotationcommand.java

```

/*
 * $Id: AbstractAnnotationCommand.java,v 1.5 2009/02/13 12:07:59
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditAnnotationCommand
;

/**
 * @author ron
 */
public abstract class AbstractAnnotationCommand extends
AbstractEditCommand implements
EditAnnotationCommand {

```

```

private Object groupingObject;
private Annotatable annotatable;
private String text;

protected AbstractAnnotationCommand(CommandHandler commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository repository) {
    super(commandHandler, annotationCommandFactory, repository);
}

protected Object getGroupingObject() {
    return groupingObject;
}

public void setGroupingObject(Object groupingObject) {
    this.groupingObject = groupingObject;
}

protected Annotatable getAnnotatable() {
    return annotatable;
}

public void setAnnotatable(Annotatable annotatable) {
    this.annotatable = annotatable;
}

protected String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

```

abstractappawareactionlistener.java

```

/*
 * $Id: AbstractAppAwareActionListener.java,v 1.1 2008/02/15 21:41:43
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import nextapp.echo2.app.event.ActionListener;

```

```

/**
 * @author ron
 */
public abstract class AbstractAppAwareActionListener implements
ActionListener {
    static final long serialVersionUID = 0;

    private final UIFrameworkApp app;

    protected AbstractAppAwareActionListener(UIFrameworkApp app) {
        this.app = app;
    }

    protected UIFrameworkApp getApp() {
        return app;
    }
}

```

abstractappawarecontroller.java

```

/*
 * $Id: AbstractAppAwareController.java,v 1.3 2008/05/01 08:10:17
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.controller;

import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * An abstract controller that makes the current UIFrameworkApp
available to
 * sub-classes.
 *
 * @author ron
 */
public abstract class AbstractAppAwareController extends
AbstractController implements
AppAwareController {

    private UIFrameworkApp app;

    protected AbstractAppAwareController() {

```

```

        super();
    }

    protected AbstractAppAwareController(UIFrameworkApp app) {
        super();
        setApp(app);
    }

    protected AbstractAppAwareController(EventDispatcher eventDispatcher,
UIFrameworkApp app) {
        super(eventDispatcher);
        setApp(app);
    }

    /**
     * This method is public so that the UIFrameworkApp can pass itself
to the
     * controller when initializing it.
     *
     * @see
edu.harvard.fas.rregan.uiframework.controller.AppAwareController#setAp
p(edu.harvard.fas.rregan.uiframework.UIFrameworkApp)
     */
    public void setApp(UIFrameworkApp app) {
        this.app = app;
    }

    protected UIFrameworkApp getApp() {
        return app;
    }
}

```

abstractassistant.java

```

/*
 * $Id: AbstractAssistant.java,v 1.8 2009/02/12 11:01:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import java.text.FieldPosition;
import java.text.MessageFormat;
import java.util.Locale;

import edu.harvard.fas.rregan.ApplicationException;

```

```

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditIssueCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditNoteCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * A base class for assistants that supports working with simple
annotations.
 *
 * @author ron
 */
public abstract class AbstractAssistant {

    private final AnnotationCommandFactory annotationCommandFactory;
    private final AnnotationRepository annotationRepository;
    private final CommandHandler commandHandler;
    private final ResourceBundleHelper resourceBundleHelper;

    protected AbstractAssistant(String resourceName, CommandHandler
commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository annotationRepository) {
        this.commandHandler = commandHandler;
        this.annotationCommandFactory = annotationCommandFactory;
}

```

```

this.annotationRepository = annotationRepository;
this.resourceBundleHelper = new
ResourceBundleHelper(resourceName);
}

protected AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

protected AnnotationRepository getAnnotationRepository() {
    return annotationRepository;
}

protected CommandHandler getCommandHandler() {
    return commandHandler;
}

protected Note addNote(Object groupingObject, User assistantUser,
    Annotatable thingBeingAnalyzed, String noteText) throws Exception {
    EditNoteCommand editNoteCommand =
annotationCommandFactory.newEditNoteCommand();
    editNoteCommand.setGroupingObject(groupingObject);
    editNoteCommand.setText(noteText);
    editNoteCommand.setAnnotatable(thingBeingAnalyzed);
    editNoteCommand.setEditedBy(assistantUser);
    editNoteCommand = commandHandler.execute(editNoteCommand);
    return editNoteCommand.getNote();
}

protected void addSimpleIssue(ProjectOrDomain projectOrDomain, User
assistantUser,
    ProjectOrDomainEntity thingBeingAnalyzed, String text) throws
Exception {
    EditIssueCommand editIssueCommand =
getAnnotationCommandFactory().newEditIssueCommand();
    try {
        Issue issue = getAnnotationRepository().findIssue(projectOrDomain,
thingBeingAnalyzed,
            text);
        editIssueCommand.setIssue(issue);
        editIssueCommand.setAnnotatable(thingBeingAnalyzed);
        editIssueCommand.setEditedBy(assistantUser);
        // don't mess up the existing properties when only adding an
        // annotatable
        editIssueCommand.setText(issue.getText());
        editIssueCommand.setMustBeResolved(issue.isMustBeResolved());
        editIssueCommand = getCommandHandler().execute(editIssueCommand);
    }
}

```

```

} catch (NoSuchAnnotationException e) {
    editIssueCommand.setGroupingObject(projectOrDomain);
    editIssueCommand.setText(text);
    editIssueCommand.setMustBeResolved(true);
    editIssueCommand.setAnnotatable(thingBeingAnalyzed);
    editIssueCommand.setEditedBy(assistantUser);
    editIssueCommand = getCommandHandler().execute(editIssueCommand);
    Issue issue = editIssueCommand.getIssue();
    addSimplePositionToIssue(projectOrDomain, assistantUser, issue, "Do
nothing.");
}
}

protected Position addSimplePositionToIssue(ProjectOrDomain
projectOrDomain,
    User assistantUser, Issue issue, String positionText) throws
Exception {
    EditPositionCommand editPositionCommand =
getAnnotationCommandFactory()
    .newEditPositionCommand();
    try {
        editPositionCommand.setPosition(getAnnotationRepository().findPosit
ion(projectOrDomain,
            positionText));
    } catch (NoSuchPositionException e) {
    }
    editPositionCommand.setIssue(issue);
    editPositionCommand.setEditedBy(assistantUser);
    editPositionCommand.setText(positionText);
    editPositionCommand =
getCommandHandler().execute(editPositionCommand);
    return editPositionCommand.getPosition();
}

protected void removeAnnotation(Annotation annotation, Annotatable
annotatable)
    throws Exception {
    RemoveAnnotationFromAnnotatableCommand command =
getAnnotationCommandFactory()
    .newRemoveAnnotationFromAnnotatableCommand();
    command.setAnnotatable(annotatable);
    command.setAnnotation(annotation);
    command = getCommandHandler().execute(command);
}

protected ResourceBundleHelper getResourceBundleHelper() {
    return resourceBundleHelper;
}

```

```

}

/**
 * @param locale
 * @throws ApplicationException
 *          if a resource bundle is not found for the locale
 */
public void setResourceLocale(Locale locale) {
    resourceBundleHelper.setLocale(locale);
}

private final StringBuffer messageBuffer = new StringBuffer(100);

protected String createMessage(String propName, String defaultMsg,
Object... arguments) {
    String msgPattern = getResourceBundleHelper().getString(propName,
defaultMsg);
    MessageFormat msgFormat = new MessageFormat(msgPattern);
    messageBuffer.setLength(0);
    return msgFormat.format(arguments, messageBuffer, new
FieldPosition(0)).toString();
}

```

abstractcommand.java

```

/*
 * $Id: AbstractCommand.java,v 1.1 2008/12/13 00:41:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.repository.Repository;

/**
 * @author ron
 */
public abstract class AbstractCommand implements Command {
    protected static final Logger log =
Logger.getLogger(AbstractCommand.class);

    private final Repository repository;

    protected AbstractCommand(Repository repository) {

```

```

        this.repository = repository;
    }

protected Repository getRepository() {
    return repository;
}
}

```

abstractcommandfactory.java

```

/*
 * $Id: AbstractCommandFactory.java,v 1.1 2008/12/13 00:40:57 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

/**
 * A base command factory that supplied the creation strategy to
 * implementation specific command factories.
 *
 * @author ron
 */
public class AbstractCommandFactory implements CommandFactory {
    private final CommandFactoryStrategy creationStrategy;

    protected AbstractCommandFactory(CommandFactoryStrategy
creationStrategy) {
        this.creationStrategy = creationStrategy;
    }

    protected CommandFactoryStrategy getCreationStrategy() {
        return creationStrategy;
    }

    @Override
    public BatchCommand newBatchCommand() {
        return (BatchCommand)
getCreationStrategy().newInstance(BatchCommandImpl.class);
    }
}

```

abstractcomponent.java

```

/*
 * $Id: AbstractComponent.java,v 1.5 2008/10/12 07:55:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.Locale;

import nextapp.echo2.app.Column;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public abstract class AbstractComponent extends Column implements
EditMode {
    static final long serialVersionUID = 0L;

    private ResourceBundleHelper resourceBundleHelper;
    private final EditMode editMode;

    protected AbstractComponent(EditMode editMode) {
        this.editMode = editMode;
    }

    protected AbstractComponent(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
        this(editMode);
        setResourceBundleHelper(resourceBundleHelper);
    }

    protected ResourceBundleHelper getResourceBundleHelper(Locale locale)
{
        resourceBundleHelper.setLocale(locale);
        return resourceBundleHelper;
    }

    protected void setResourceBundleHelper(ResourceBundleHelper
resourceBundleHelper) {
        this.resourceBundleHelper = resourceBundleHelper;
    }
}

```

```

protected EditMode getEditMode() {
    return editMode;
}

/***
 * This method should be overloaded to check permissions of the user
making
 * the edits and return true if the user only has read
permissions.<br>
 * use getApp().getUser() to get the user.
 *
 * @return
 */
public boolean isReadOnlyMode() {
    return editMode.isReadOnlyMode();
}

@Override
public boolean isStateEdited() {
    return editMode.isStateEdited();
}

@Override
public void setStateEdited(boolean stateEdited) {
    editMode.setStateEdited(stateEdited);
}

/***
 * @return
 */
public EventDispatcher getEventDispatcher() {
    return getApp().getEventDispatcher();
}

/***
 * @return
 */
public UIFrameworkApp getApp() {
    return UIFrameworkApp.getApp();
}

```

abstractcomponentmanipulator.java

```
/*
```

```

 * $Id: AbstractComponentManipulator.java,v 1.5 2008/10/13 22:58:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;

/**
 * @author ron
 */
public abstract class AbstractComponentManipulator implements
ComponentManipulator {

    public Object getOptionModel(Component component) {
        return null;
    }

    public void setOptionModel(Component component, Object optionModel) {
    }

    public Object getModel(Component component) {
        return null;
    }

    public void setModel(Component component, Object valueModel) {
    }
}
```

abstractcontroller.java

```

/*
 * $Id: AbstractController.java,v 1.8 2009/01/27 09:30:17 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.controller;

import java.util.Collections;
import java.util.HashSet;
import java.util.Set;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import org.apache.log4j.Logger;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * A base implementation for a Controller that contains the management
for
 * action listeners that listen for events from this controller
(typically the
 * event dispatcher) and the event types that this controller is
interested in
 * recieving (as an action listener itself.)<br>
 * Classes that extend this class should call
addEventTypeToListenFor() in the
 * constructor for each event this controller handles so that it will
be wired
 * up to receive those events from the event dispatcher.
 *
 * @author ron
 */
public abstract class AbstractController implements Controller {
    private static final Logger log =
Logger.getLogger(AbstractController.class);

    // TODO: use a weak hash so stale listeners are removed?
    private final Set<ActionListener> actionListeners = new
HashSet<ActionListener>();
    private final Set<Class<? extends ActionEvent>> eventTypesToListenFor
= new HashSet<Class<? extends ActionEvent>>();
    private EventDispatcher eventDispatcher = null;

    protected AbstractController() {
        super();
        log.debug("creating controller " + getClass().getName());
    }

    protected AbstractController(EventDispatcher eventDispatcher) {
        this();
        setEventDispatcher(eventDispatcher);
    }

    protected EventDispatcher getEventDispatcher() {
        return eventDispatcher;
    }

    protected void setEventDispatcher(EventDispatcher eventDispatcher) {
        this.eventDispatcher = eventDispatcher;
        if (eventDispatcher != null) {
            addEventListener(eventDispatcher);
        }
    }

    protected Set<ActionListener> getListeners() {
        return actionListeners;
    }

    public void addActionListener(ActionListener actionListener) {
        log.debug("adding listener: " + actionListener);
        getListeners().add(actionListener);
    }

    public void removeActionListener(ActionListener actionListener) {
        log.debug("removing listener: " + actionListener);
        getListeners().remove(actionListener);
    }

    protected void fireEvent(ActionEvent event) {
        log.debug("event: " + event);
        if (event != null) {
            for (ActionListener listener : getListeners()) {
                log.debug("sending event to listener " + listener);
                listener.actionPerformed(event);
            }
        }
    }

    protected void addEventTypeToListenFor(Class<? extends ActionEvent>
eventType) {
        eventTypesToListenFor.add(eventType);
    }

    protected void removeEventTypeToListenFor(Class<? extends ActionEvent>
eventType) {
        eventTypesToListenFor.remove(eventType);
    }

    /**
     * @return the set of event types that this command handles
     */
    public Set<Class<? extends ActionEvent>> getEventTypesToListenFor() {
        return Collections.unmodifiableSet(eventTypesToListenFor);
    }

    /*
     * @Override public int hashCode() { return getClass().hashCode(); }
    */
}

```

```

 * @Override public boolean equals(Object o) { if ((o != null) &&
 * getClass().equals(o.getClass())) { return true; } return false; }
 */
@Override
public String toString() {
    return getClass().getSimpleName() + "@" + super.hashCode();
}
}

```

abstractdictionarycommand.java

```

/*
 * $Id: AbstractDictionaryCommand.java,v 1.1 2008/12/13 00:39:55
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.command.AbstractCommand;
import edu.harvard.fas.rregan.nlp.DictionaryRepository;

/**
 * @author ron
 */
public abstract class AbstractDictionaryCommand extends
AbstractCommand {
    protected static final Logger log =
Logger.getLogger(AbstractDictionaryCommand.class);

    /**
     * @param dictionaryRepository
     */
    public AbstractDictionaryCommand(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }

    /**
     * @return
     */
    protected DictionaryRepository getDictionaryRepository() {
        return (DictionaryRepository) getRepository();
    }
}

```

}

abstractdictionarylemmatizerrule.java

```

/*
 * $Id: AbstractDictionaryLemmatizerRule.java,v 1.3 2009/01/26
10:19:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.lemmatizer;

import edu.harvard.fas.rregan.nlp.LemmatizerRule;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * Base class for lemmatizers that use the dictionary.
 *
 * @author ron
 */
public abstract class AbstractDictionaryLemmatizerRule implements
LemmatizerRule {

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     */
    public AbstractDictionaryLemmatizerRule(DictionaryRepository
dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }

    protected DictionaryRepository getDictionaryRepository() {
        return dictionaryRepository;
    }
}

```

abstracteditcommand.java

```

/*
 * $Id: AbstractEditCommand.java,v 1.1 2009/02/13 12:07:57 rregan Exp
$
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.

```

```
/*
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import edu.harvard.fas.rregan.command.AbstractCommand;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.user.User;

/***
 * @author ron
 */
public abstract class AbstractEditCommand extends AbstractCommand
implements EditCommand {

private final CommandHandler commandHandler;
private final AnnotationCommandFactory annotationCommandFactory;
private User editedBy;

/***
 * @param commandHandler
 * @param annotationCommandFactory
 * @param repository
 */
public AbstractEditCommand(CommandHandler commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository) {
super(repository);
this.commandHandler = commandHandler;
this.annotationCommandFactory = annotationCommandFactory;
}

/***
 * @see
edu.harvard.fas.rregan.requel.command.EditCommand#setEditedBy(edu.harv
ard.fas.rregan.requel.user.User)
 */
@Override
public void setEditedBy(User editedBy) {
this.editedBy = editedBy;
}
}
```

```
protected User getEditedBy() {
    return editedBy;
}

protected AnnotationRepository getAnnotationRepository() {
    return (AnnotationRepository) getRepository();
}

protected CommandHandler getCommandHandler() {
    return commandHandler;
}

protected AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}
```

abstracteditorpanel.java

```
/*
 * $Id: AbstractEditorPanel.java,v 1.40 2009/02/21 10:32:13 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Grid;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.layout.GridLayoutData;

import org.apache.log4j.Logger;

import echopointng.ComboBox;
import
edu.harvard.fas.rregan.uiframework.AbstractAppAwareActionListener;
import edu.harvard.fas.rregan.uiframework.MessageHandler;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import edu.harvard.fas.rregan.uiframework.panel.AbstractPanel;
```

```

import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public abstract class AbstractEditorPanel extends AbstractPanel
implementsEditMode, MessageHandler {
    private static final Logger log =
Logger.getLogger(AbstractEditorPanel.class);
    static final long serialVersionUID = 0L;

    /**
     * The property to set in the resource bundle for the panel's title
     * if the
     * entity being edited already exists. If this property does not
     * exist the
     * Panel.Title is used.
     */
    public static final String PROP_EXISTING_OBJECT_PANEL_TITLE =
"EditorPanel.ExistingObject.Title";

    /**
     * The property to set in the resource bundle for the panel's title
     * if the
     * entity being edited is new. If this property does not exist the
     * Panel.Title is used.
     */
    public static final String PROP_NEW_OBJECT_PANEL_TITLE =
"EditorPanel.NewObject.Title";

    /**
     * The style name to use in the Echo2 stylesheet to control the grid
     * layout
     * of the editor's input fields.
     */
    public static final String STYLE_NAME_FORM_GRID =
"EditorPanel.FormGrid";

    /**
     * The style name to use in the Echo2 stylesheet for controlling the
     * style
     * of labels on input fields added via addInput, but not
     * addMutliRowInput.
     */
    public static final String STYLE_NAME_FIELD_LABEL =
"EditorPanel.InputField.Label";

```

```

    /**
     * The style name to use in the Echo2 stylesheet for controlling the
     * style
     * of all types of input fields. Different types of fields will be
     * distinguished by the type element in the stylesheet.
     */
    public static final String STYLE_NAME_INPUT_FIELD =
"EditorPanel.InputField";

    /**
     * The style name to use in the Echo2 stylesheet for controlling the
     * style
     * of labels on input fields added via addMutliRowInput.
     */
    public static final String STYLE_NAME_MULTIROW_FIELD_LABEL =
"EditorPanel.MultiRowField.Label";

    /**
     * The name to use in the panel's properties file to set the label of
     * the
     * cancel button. If the property is undefined "Cancel" is used.
     */
    public static final String PROP_LABEL_CANCEL_BUTTON =
"EditorPanel.CancelButton.Label";

    /**
     * The name to use in the panel's properties file to set the label of
     * the
     * save button. If the property is undefined "Save" is used.
     */
    public static final String PROP_LABEL_SAVE_BUTTON =
"EditorPanel.SaveButton.Label";

    /**
     * The name to use in the panel's properties file to set the label of
     * the
     * copy button. If the property is undefined "Copy" is used.
     */
    public static final String PROP_LABEL_COPY_BUTTON =
"EditorPanel.CopyButton.Label";

    /**
     * The name to use in the panel's properties file to set the label of
     * the
     * delete button. If the property is undefined "Delete" is used.
     */

```

```

public static final String PROP_LABEL_DELETE_BUTTON =
"EditorPanel.DeleteButton.Label";

private final EditorComponents editorComponents;
private Grid formGrid;
private GridLayoutData formLayout;
private GridLayoutData fullRowLayout;
private GridLayoutData threeQuarterRowLayout;
private Row actionButtons;
private Button saveButton;
private Button cancelButton;
private Button deleteButton;
private Label generalMessage;
private Label titleLabel;
private boolean valid;
private boolean stateEdited;

protected AbstractEditorPanel(String resourceBundleName, Class<?>
supportedContentType) {
    super(resourceBundleName, PanelActionType.Editor,
supportedContentType);
    editorComponents = new EditorComponents(this);
}

protected AbstractEditorPanel(String resourceBundleName, Class<?>
supportedContentType,
    String panelName) {
    super(resourceBundleName, PanelActionType.Editor,
supportedContentType, panelName);
    editorComponents = new EditorComponents(this);
}

@Override
public void setup() {
    super.setup();
    fullRowLayout = new GridLayoutData();
    fullRowLayout.setColumnSpan(4);
    fullRowLayout.setAlignment(new Alignment(Alignment.CENTER,
Alignment.CENTER));
    threeQuarterRowLayout = new GridLayoutData();
    threeQuarterRowLayout.setColumnSpan(3);

    Row row = null;

    // title
    if (isShowTitle()) {
        row = new Row();
        row.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));
        titleLabel = new Label getTitle();
        titleLabel.setStyleName(STYLE_NAME_PANEL_TITLE);
        row.add(titleLabel);
        add(row);
    }

    // grid
    row = new Row();
    row.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));
    row.setInsets(new Insets(new Extent(10)));
    formGrid = new Grid(4);
    formGrid.setStyleName(STYLE_NAME_FORM_GRID);
    row.add(formGrid);
    add(row);

    formLayout = new GridLayoutData();
    formLayout.setAlignment(new Alignment(Alignment.RIGHT,
Alignment.TOP));

    // buttons
    actionButtons = new Row();
    if (isShowCancel() || isShowDelete() || isShowSave()) {
        actionButtons.setStyleName(STYLE_NAME_DEFAULT);
        actionButtons.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));
        if (isShowCancel()) {
            cancelButton = new
Button getResourceBundleHelpergetLocale().getString(
PROP_LABEL_CANCEL_BUTTON, "Cancel");
            cancelButton.setStyleName(STYLE_NAME_DEFAULT);
            actionButtons.add(cancelButton);
            cancelButton.addActionListener(new
AbstractAppAwareActionListener(getApp()) {
                static final long serialVersionUID = 0;
                public void actionPerformed(ActionEvent event) {
                    try {
                        cancel();
                    } catch (Exception e) {
                        // TODO: should this be propagated up?
                        log.error("Unexpected exception after cancel : " + e, e);
                    }
                }
            });
        }
    }
}

```

```

    });

}

if (isShowSave()) {
    saveButton = new
    Button(getResourceBundleHelpergetLocale()).getString(
        PROP_LABEL_SAVE_BUTTON, "Save"));
    saveButton.setStyleName(STYLE_NAME_DEFAULT);
    actionButtons.add(saveButton);
    saveButton.addActionListener(new
AbstractAppAwareActionListener(getApp()) {
    static final long serialVersionUID = 0;

    public void actionPerformed(ActionEvent event) {
        try {
            save();
        } catch (Exception e) {
            log.error("Unexpected exception after save: " + e, e);
            setGeneralMessage("Unexpected problem saving: " + e);
            // TODO: should this be propogated up?
        }
    }
});
}

// delete button
if (isShowDelete()) {
    deleteButton = new
    Button(getResourceBundleHelpergetLocale()).getString(
        PROP_LABEL_DELETE_BUTTON, "Delete"));
    deleteButton.setStyleName(STYLE_NAME_DEFAULT);
    actionButtons.add(deleteButton);
    deleteButton.addActionListener(new
AbstractAppAwareActionListener(getApp()) {
    static final long serialVersionUID = 0;

    public void actionPerformed(ActionEvent event) {
        try {
            delete();
        } catch (Exception e) {
            // TODO: should this be propogated up?
            log.error("Unexpected exception after delete : " + e, e);
        }
    }
});
}

add(actionButtons);
}

    }

    row = new Row();
    row.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));
    row.setInsets(new Insets(new Extent(10)));
    generalMessage = new Label();
    generalMessage.setStyleName(STYLE_NAME_VALIDATION_LABEL);
    row.add(generalMessage);
    add(row);
}

@Override
public boolean isStateEdited() {
    return stateEdited;
}

@Override
public void setStateEdited(boolean stateEdited) {
    this.stateEdited = stateEdited;
}

@Override
public void setGeneralMessage(String message) {
    generalMessage.setText(message);
}

protected Button addActionButton(Button button) {
    return addActionButton(button,
actionButtons.getComponents().length);
}

protected Button addActionButton(Button button, int index) {
    actionButtons.add(button, index);
    if ((button.getStyleName() == null) ||
"".equals(button.getStyleName().trim())) {
        button.setStyleName(STYLE_NAME_DEFAULT);
    }
    return button;
}

protected Button removeActionButton(Button button) {
    actionButtons.remove(button);
    return button;
}

@Override

```

```

public void dispose() {
    super.dispose();
}

/**
 * Add an input control with a label such that the label will be in
the
 * first column and the control will be in the second.
 *
 * @param <T>
 * @param fieldName
 * @param labelProperty
 * @param labelDefault
 * @param inputComponent
 * @param model
 * @return
 */
protected <T extends Component> T addInput(String fieldName, String
labelProperty,
    String labelDefault, T inputComponent, Object model) {
    String labelText =
getResourceBundleHelper(getLocale()).getString(labelProperty,
    labelDefault);
    Label label = new Label((labelText.endsWith(":") ? labelText :
labelText + ":")));
    return addInput(fieldName, label, inputComponent, model);
}

/**
 * @param <T>
 * @param fieldName
 * @param label
 * @param inputComponent
 * @param model
 * @return
 */
protected <T extends Component> T addInput(String fieldName, Label
label, T inputComponent,
    Object model) {
    editorComponents.addInput(fieldName, label, inputComponent, model);
    inputComponent.setStyleName(STYLE_NAME_INPUT_FIELD);
    // a hack to get a ComboBox text field to pickup the correct font
from
    // the stylesheet.
    if (inputComponent instanceof ComboBox) {
        ((ComboBox)
inputComponent).getTextField().setStyleName(STYLE_NAME_INPUT_FIELD);
}
}
}
label.setLayoutData(formLayout);
label.setStyleName(STYLE_NAME_FIELD_LABEL);
editorComponents.getHelp(fieldName).setStyleName(STYLE_NAME_HELP_LAB
EL);
editorComponents.getMessage(fieldName).setStyleName(STYLE_NAME_VALID
ATION_LABEL);
formGrid.add(label);
formGrid.add(inputComponent);
formGrid.add(editorComponents.getHelp(fieldName));
formGrid.add(editorComponents.getMessage(fieldName));

// By default expand the tree of a CheckBoxTreeSet
if (inputComponent instanceof CheckBoxTreeSet) {
    ((CheckBoxTreeSet) inputComponent).expandAll();
}
return inputComponent;
}

/**
 * Add multirow input field like a text area or composite of
components in a
 * table, etc.
 *
 * @param <T>
 * @param fieldName
 * @param labelProperty
 * @param labelDefault
 * @param inputComponent
 * @param model
 * @return
 */
protected <T extends Component> T addMultiRowInput(String fieldName,
String labelProperty,
    String labelDefault, T inputComponent, Object model) {
    String labelText =
getResourceBundleHelper(getLocale()).getString(labelProperty,
    labelDefault);
    Label label = new Label((labelText.endsWith(":") ? labelText :
labelText + ":")));
    label.setStyleName(STYLE_NAME_MULTIROW_FIELD_LABEL);
    return addMultiRowInput(fieldName, label, inputComponent, model);
}

/**
 * Add multirow input field like a text area or composite of
components in a
 */

```

```

* table, etc.
*
* @param <T>
* @param fieldName
* @param label
* @param inputComponent
* @param model
* @return
*/
protected <T extends Component> T addMultiRowInput(String fieldName,
Label label,
T inputComponent, Object model) {

editorComponents.addInput(fieldName, label, inputComponent, model);
inputComponent.setStyleName(STYLE_NAME_INPUT_FIELD);
inputComponent.setLayoutData(fullRowLayout);
label.setLayoutData(fullRowLayout);
label.setStyleName(STYLE_NAME_FIELD_LABEL);

editorComponents.getHelp(fieldName).setStyleName(STYLE_NAME_HELP_LAB
EL);
editorComponents.getMessage(fieldName).setStyleName(STYLE_NAME_VALID
ATION_LABEL);
editorComponents.getMessage(fieldName).setLayoutData(threeQuarterRow
Layout);

formGrid.add(label);
formGrid.add(inputComponent);
formGrid.add(editorComponents.getHelp(fieldName));
formGrid.add(editorComponents.getMessage(fieldName));
return inputComponent;
}

protected Component getInput(String fieldName) {
return editorComponents.getInput(fieldName);
}

protected <T> T getInputModel(String fieldName, Class<T> type) {
return editorComponents.getInputModel(fieldName, type);
}

/**
 * Replace the model of the named input with the new model.
*
* @param fieldName -
*           the name of the field
* @param model -

```

```

*           the new model.
*/
protected void setInputModel(String fieldName, Object model) {
editorComponents.setInputModel(fieldName, model);
}

/**
* Set the value of the named input field.
*
* @param fieldName -
*           the name of the input field.
* @param value -
*           the new value to set in the field.
*/
protected void setInputValue(String fieldName, Object value) {
editorComponents.setInputValue(fieldName, value);
}

/**
* @param <T> -
*           the type of the value to return.
* @param fieldName
* @param type -
*           the class of the type to return. Note: primatives are
not
*           supported, for example Boolean.class should be used
instead of
*           boolean.class.
* @return The value of the input field in the specified type
* @throws ClassCastException -
*           If the value can't be converted to the specified type.
*/
protected <T> T getInputValue(String fieldName, Class<T> type) {
return editorComponents.getInputValue(fieldName, type);
}

/**
* Clear the message text for all input fields.
*/
public void clearValidationMessages() {
editorComponents.clearValidationMessages();
}

/**
* Set the message text of the specified input field.
*
* @param fieldName

```

```

 * @param message
 */
public void setValidationMessage(String fieldName, String message) {
    editorComponents.setValidationMessage(fieldName, message);
}

/**
 *
 */
public void validateContent() {
    editorComponents.validateContent();
}

/**
 * Implement saving of the target object by overriding this method.
If the
 * user has invalid input throw a ValidationException. Overriding
 * implementations should call super.save() so that in the future
when
 * controls are self validating that code can be put here.<br>
 * The save method should fire one or more UpdateEntityEvents to
allow other
 * UI components to refresh any references to the updated entity.<br>
 * <code>getEventDispatcher().dispatchEvent(new
UpdateEntityEvent(this, entity));</code>
 */
public void save() {
    validateContent();
}

/**
 * This method doesn't need to be updated
 */
public void cancel() {
    clearValidationMessages();
    getEventDispatcher().dispatchEvent(new ClosePanelEvent(this));
}

/**
 * Implement deleting of the target object by overriding this
method.<br>
 * The delete method should fire one or more DeletedEntityEvent to
allow
 * other UI components to refresh any references to the deleted
entity and
 * close panels of deleted sub-components.<br>
 * <code>
 * getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
entity));
 * for (Entity subEntity : deletedSubEntities) {
 *     getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
subEntity));
 * }
 * </code>
 */
public void delete() {

}

/**
 * @return
 */
public boolean isValid() {
    return valid;
}

/**
 * @param valid
 */
public void setValid(boolean valid) {
    this.valid = valid;
}

protected boolean isShowTitle() {
    return true;
}

/**
 * This method should be overloaded to check permissions of the user
making
 * the edits and return true if the user only has read
permissions.<br>
 * use getApp().getUser() to get the user.
 *
 * @return
 */
public boolean isReadOnlyMode() {
    return false;
}

protected boolean isShowSave() {
    return !isReadOnlyMode();
}

```

```

protected boolean isShowCancel() {
    return true;
}

protected boolean isShowDelete() {
    return true;
}
}

```

abstraceteditortreenodefactory.java

```

/*
 * $Id: AbstractEditorTreeNodeFactory.java,v 1.3 2009/01/21 09:23:21
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import java.util.Locale;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.event.ActionListener;
import edu.harvard.fas.rregan.ResourceBundleHelper;

/**
 * A base factory for creating NavigatorTreeNode nodes for specific
types of
 * entity objects. This class manages and makes available the type of
entity
 * object for which the factory produces nodes.
 *
 * @author ron
 */
public abstract class AbstractEditorTreeNodeFactory implements
EditorTreeNodeFactory {

private final ResourceBundleHelper resourceBundleHelper;
private Class<?> targetClass;

protected AbstractEditorTreeNodeFactory(String resourceName,
Class<?> targetClass) {
    resourceBundleHelper = new ResourceBundleHelper(resourceName);
    setTargetClass(targetClass);
}

/**

```

```

 * Return the type of object this factory builds tree nodes for.
 *
 * @return
 */
public Class<?> getTargetType() {
    return targetClass;
}

protected void setTargetClass(Class<?> targetClass) {
    this.targetClass = targetClass;
}

protected ResourceBundleHelper getResourceBundleHelper(Locale locale)
{
    resourceBundleHelper.setLocale(locale);
    return resourceBundleHelper;
}

protected EditorTreeNode
addEditorTreeNodeActionButtonDecorator(EditorTree tree,
    EditorTreeNode decoratedNode, String labelText, String styleName,
    ActionListener actionListener) {
    Button button = new Button(labelText);
    button.setStyleName(styleName);
    button.addActionListener(actionListener);
    return new EditorTreeNodeActionButtonDecorator(decoratedNode, tree,
button);
}
}

```

abstraceteditortreenodeupdatelistener.java

```

/*
 * $Id: AbstractEditorTreeNodeUpdateListener.java,v 1.1 2008/10/15
09:20:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

/**
 * a NavigatorTreeNodeUpdateListener listens for update events for the
object
 * that the node is displaying, updates the nodes data object and
notifies the
 * tree to refresh the view when it changes. This abstract version
manages the

```

```

 * related objects and listening for events, it should be extended
with logic
 * for determining when the tree display is changed by changes to the
underlying
 * object and for updating the data stored in the node.
 *
 * @author ron
 */
public abstract class AbstractEditorTreeNodeUpdateListener implements
EditorTreeNodeUpdateListener {
static final long serialVersionUID = 0L;

private final EditorTreeNode navigatorTreeNode;
private final EditorTree tree;

protected AbstractEditorTreeNodeUpdateListener(EditorTreeNode
navigatorTreeNode,
EditorTree tree) {
super();
this.navigatorTreeNode = navigatorTreeNode;
this.tree = tree;
}

protected EditorTreeNode getNavigatorTreeNode() {
return navigatorTreeNode;
}

protected EditorTree getTree() {
return tree;
}
}

```

abstracteditprojectcommand.java

```

/*
 * $Id: AbstractEditProjectCommand.java,v 1.6 2009/03/30 11:54:29
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.command.EditCommand;

```

```

import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
public abstract class AbstractEditProjectCommand extends
AbstractProjectCommand implements
EditCommand {

private User editedBy;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
public AbstractEditProjectCommand(AssistantFacade assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

/**
 * @see
edu.harvard.fas.rregan.requel.command.EditCommand#setEditedBy(edu.harv
ard.fas.rregan.requel.user.User)
*/
@Override
public void setEditedBy(User editedBy) {
this.editedBy = editedBy;
}

protected User getEditedBy() {
return editedBy;
}

```

```
}
```

abstracteditprojectordomainentitycommand.java

```
/*
 * $Id: AbstractEditProjectOrDomainEntityCommand.java,v 1.14
2009/03/30 11:54:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.command;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.EditProjectOrDomainEntit
yCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
public abstract class AbstractEditProjectOrDomainEntityCommand extends
AbstractEditProjectCommand
    implements EditProjectOrDomainEntityCommand {

    private ProjectOrDomain projectOrDomain;
    private boolean analysisEnabled = true;
    private String name;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     */

}
```

```
        * @param commandHandler
     */
    public AbstractEditProjectOrDomainEntityCommand(AssistantFacade
assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public void setAnalysisEnabled(boolean analysisEnabled) {
        this.analysisEnabled = analysisEnabled;
    }

    protected boolean isAnalysisEnabled() {
        return analysisEnabled;
    }

    /**
     * @see
     * @see
edu.harvard.fas.rregan.requel.project.command.EditProjectOrDomainEntit
yCommand#setProjectOrDomain(edu.harvard.fas.rregan.requel.project.Proj
ectOrDomain)
     */
    @Override
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
        this.projectOrDomain = projectOrDomain;
    }

    protected ProjectOrDomain getProjectOrDomain() {
        return projectOrDomain;
    }

    @Override
    public void setName(String name) {
        this.name = name;
    }

    protected String getName() {
        return name;
    }
}
```

abstractjparepository.java

```
package edu.harvard.fas.rregan.repository.jpa;

import java.lang.reflect.Field;
import java.lang.reflect.Method;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.hibernate.Hibernate;
import org.hibernate.proxy.HibernateProxy;
import org.hibernate.proxy.LazyInitializer;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.repository.AbstractRepository;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;

/**
 * @author ron
 */
@Transactional(propagation = Propagation.REQUIRED)
public class AbstractJpaRepository extends AbstractRepository {

    @PersistenceContext
    private EntityManager entityManager;

    protected AbstractJpaRepository(ExceptionMapper exceptionMapper) {
        super(exceptionMapper);
    }

    protected EntityManager getEntityManager() {
        return entityManager;
    }

    @Override
    public <T> T persist(T entity) throws EntityException {
        try {
            entityManager.persist(entity);
            return entity;
        } catch (Exception e) {
            log.warn(e, e);
            throw convertException(e, entity.getClass(), entity,
                    EntityExceptionActionType.Creating);
        }
    }

    /**
     * Get the latest version of the supplied entity.
     */
    @Override
    public <T> T get(T entity) throws EntityException {
        if (entity != null) {
            try {
                return attach(entityManager, entity);
            } catch (Exception e) {
                log.warn(e, e);
                throw convertException(e, entity.getClass(), entity,
                        EntityExceptionActionType.Reading);
            }
        }
        return null;
    }

    /**
     * This is a hibernate specific implementation that loads proxied
     * properties
     * and collections of the supplied entity. NOTE: only a single level
     * is
     * loaded, for example the elements of a collection but not the lazy
     * properties of those elements. <br>
     * TODO: this is very expensive
     */
    @Override
    public <T> T initialize(T entity) throws EntityException {
        if (entity != null) {
            try {
                T attached = attach(entityManager, entity);
                // walk up the class hierarchy
                Class<?> entityType = attached.getClass();
                while (!entityType.getName().equals(Object.class.getName())) {
                    for (Field field : entityType.getDeclaredFields()) {
                        if (log.isDebugEnabled()) {
                            log.debug("initializing: " + attached + " field " +
                                    field.getName()
                                    + " " + field.getType());
                        }
                        field.setAccessible(true);
                        Hibernate.initialize(field.get(attached));
                    }
                    entityType = entityType.getSuperclass();
                }
            } catch (Exception e) {
                log.warn(e, e);
                throw convertException(e, entity.getClass(), entity,
                        EntityExceptionActionType.Initializing);
            }
        }
        return entity;
    }
}
```

```

    }

    Hibernate.initialize(attached);
    return attached;
} catch (Exception e) {
    log.warn(e, e);
    throw convertException(e, entity.getClass(), entity,
        EntityExceptionActionType.Reading);
}
}

return null;
}

/**
 * Save the state of the supplied entity to the database.
 */
@Override
public <T> T merge(T entity) throws EntityException {
    if (entity != null) {
        try {
            return entityManager.merge(entity);
        } catch (Exception e) {
            log.warn(e, e);
            throw convertException(e, entity.getClass(), entity,
                EntityExceptionActionType.Updating);
        }
    }
    return null;
}

@Override
public void delete(Object entity) {
    try {
        entityManager.remove(attach(entityManager, entity));
    } catch (Exception e) {
        log.warn(e, e);
        throw convertException(e, entity.getClass(), entity,
            EntityExceptionActionType.Deleting);
    }
}

@Override
public void flush() throws EntityException {
    try {
        entityManager.flush();
    } catch (Exception e) {
        log.warn(e, e);
        throw convertException(e);
    }
}
}

}

protected static <T> T attach(EntityManager entityManager, T entity)
{
    if (!entityManager.contains(entity) && (entity != null)) {
        T original = entity;
        // TODO: this is a hibernate specific hack for proxy related
        // problems
        if (HibernateProxy.class.isAssignableFrom(entity.getClass())) {
            LazyInitializer lazyInitializer = ((HibernateProxy) entity)
                .getHibernateLazyInitializer();
            entity = entityManager.find((Class<T>)
                lazyInitializer.getPersistentClass(),
                lazyInitializer.getIdentifier());
            if (entity == null) {
                entity = original;
                log.warn("reloading " + entity.getClass() + " "
                    + lazyInitializer.getIdentifier() + " returned null;");
            }
        } else {
            // the object may not be persisted, if it doesn't have an id,
            // don't load it from the db
            Object id = getId(entity);
            if (id != null) {
                entity = entityManager.find((Class<T>) entity.getClass(), id);
                if (entity == null) {
                    entity = original;
                    log.warn("reloading " + entity.getClass() + " " + id + " "
                        returned null);
                    // TODO: Something fishy is happening. in the
                    // hibernate AbstractBatcher getResultSet()
                    // ResultSet rs = ps.executeQuery();
                    // returns an empty result set, even though running
                    // the query in SQL ui returns the expected rows.

                    // maybe the entity hasn't been committed yet, return
                    // the original it may be that the entity was deleted,
                    // what should happen in that case?
                }
            }
        }
    }
    if (EntityProxyInterceptor.isEntityProxy(entity)) {
        log.warn("Entity proxy " + entity + " was expected to be a raw
entity.",
            new Throwable());
    }
}
}

```

```

entity = EntityProxyInterceptor.unwrap(entity);
}
return entity;
}

protected static Object getId(Object entity) {
if (entity != null) {
Class<?> entityType = entity.getClass();
do {
try {
Method getId = entityType.getDeclaredMethod("getId");
getId.setAccessible(true);
return getId.invoke(entity);
} catch (NoSuchMethodException e) {
entityType = entityType.getSuperclass();
} catch (Exception e) {
throw new RuntimeException("could not get the id of the entity " +
+ entity, e);
}
} while (entityType != null);
throw new RuntimeException("No way to get the id of the entity " +
entity);
}
return null;
}
}

```

abstractlistcomponentmanipulator.java

```

/*
 * $Id: AbstractListComponentManipulator.java,v 1.10 2008/10/13
22:58:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;
import nextapp.echo2.app.list.AbstractListComponent;
import nextapp.echo2.app.list.DefaultListSelectionModel;

```

```

import nextapp.echo2.app.list.ListModel;
import nextapp.echo2.app.list.ListSelectionModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class AbstractListComponentManipulator extends
AbstractComponentManipulator {

public <T> T getValue(Component component, Class<T> type) {
if (getModel(component).getSelectionMode() ==
DefaultListSelectionModel.MULTIPLE_SELECTION) {
Collection<Object> selections = createAppropriateCollection((Class)
type);
for (int i = 0; i < getModel(component).size(); i++) {
if (getModel(component).isSelectedIndex(i)) {
selections.add(getModel(component).get(i));
}
}
return type.cast(selections);
} else {
return
type.cast(getModel(component).get(getModel(component).getMaxSelectedIn
dex()));
}
}

private <T extends Collection<Object>> T
createAppropriateCollection(Class<T> type) {
if (Set.class.isAssignableFrom(type)) {
return type.cast(new HashSet<Object>());
} else if (List.class.isAssignableFrom(type)) {
return type.cast(new ArrayList<Object>());
}
return null;
}

public void setValue(Component component, Object value) {
getModel(component).clearSelection();
if (value instanceof Collection<?>) {
for (Object o : (Collection<?>) value) {
setSingleValue(component, o);
}
} else {

```

```

        setSingleValue(component, value);
    }

private void setSingleValue(Component component, Object value) {
    int index = getModel(component).indexOf(value);
    if (index > -1) {
        getModel(component).setSelectedIndex(index, true);
    }
}

public void addListenerToDetectChangesToInput(finalEditMode
editMode, Component component) {
    getComponent(component).getSelectionModel().addChangeListener(new
ChangeListener() {
    static final long serialVersionUID = 0L;

    public void stateChanged(ChangeEvent e) {
        editMode.setStateEdited(true);
    }
});
}

@Override
public CombinedListModel getModel(Component component) {
    return (CombinedListModel)
getComponent(component).getSelectionModel();
}

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setSelectionModel((ListSelectionModel)
valueModel);
    getComponent(component).setModel((ListModel) valueModel);
}

private AbstractListComponent getComponent(Component component) {
    return (AbstractListComponent) component;
}
}

```

abstractnavigatortreenodefactory.java

```

/*
 * $Id: AbstractNavigatorTreeNodeFactory.java,v 1.5 2008/05/06
09:15:41 rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.navigation.tree;

import java.util.Locale;

import edu.harvard.fas.rregan.ResourceBundleHelper;

/**
 * A base factory for creating NavigatorTreeNode nodes for specific
types of
 * entity objects. This class manages and makes available the type of
entity
 * object that the factory produces nodes for.
 *
 * @author ron
 */
public abstract class AbstractNavigatorTreeNodeFactory implements
NavigatorTreeNodeFactory {

    private final ResourceBundleHelper resourceBundleHelper;
    private Class<?> targetClass;

    protected AbstractNavigatorTreeNodeFactory(String resourceBundleName,
Class<?> targetClass) {
        resourceBundleHelper = new ResourceBundleHelper(resourceBundleName);
        setTargetClass(targetClass);
    }

    /**
     * Return the type of object this factory builds tree nodes for.
     *
     * @return
     */
    public Class<?> getTargetClass() {
        return targetClass;
    }

    protected void setTargetClass(Class<?> targetClass) {
        this.targetClass = targetClass;
    }

    protected ResourceBundleHelper getResourceBundleHelper(Locale locale)
{
        resourceBundleHelper.setLocale(locale);
        return resourceBundleHelper;
    }
}

```

```
}
```

abstractnavigatorTreeNodeupdateListener.java

```
/*
 * $Id: AbstractNavigatorTreeNodeUpdateListener.java,v 1.4 2008/03/07
10:37:27 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.navigation.tree;

/**
 * a NavigatorTreeNodeUpdateListener listens for update events for the
object
 * that the node is displaying, updates the nodes data object and
notifies the
 * tree to refresh the view when it changes. This abstract version
manages the
 * related objects and listening for events, it should be extended
with logic
 * for determining when the tree display is changed by changes to the
underlying
 * object and for updating the data stored in the node.
 *
 * @author ron
 */
public abstract class AbstractNavigatorTreeNodeUpdateListener
implements
    NavigatorTreeNodeUpdateListener {
    static final long serialVersionUID = 0L;

    private final NavigatorTreeNode navigatorTreeNode;
    private final NavigatorTree tree;

    protected AbstractNavigatorTreeNodeUpdateListener(NavigatorTreeNode
navigatorTreeNode,
        NavigatorTree tree) {
        super();
        this.navigatorTreeNode = navigatorTreeNode;
        this.tree = tree;
    }

    protected NavigatorTreeNode getNavigatorTreeNode() {
```

```
        return navigatorTreeNode;
    }
```

```
    protected NavigatorTree getTree() {
        return tree;
    }
}
```

abstractnlptextwalker.java

```
/*
 * $Id: AbstractNLPTextWalker.java,v 1.2 2009/01/26 10:19:00 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * An NLPProcessor that takes an NLPText and applies an
NLPTextWalkerFunction to
 * every node in the text and returns the results from the
NLPTextWalkerFunction's call to the end() method on the root node,
 * transforming them to type T from type F.
 *
 * @param <T> -
 *          the return type of the process method
 * @param <F> -
 *          the return type of the NLPTextWalkerFunction
 * @author ron
 */
public abstract class AbstractNLPTextWalker<T, F> implements
NLPProcessor<T> {

    private final NLPTextWalkerFunction<F> visitorFunction;

    /**
     * @param visitorFunction
     */
    public AbstractNLPTextWalker(NLPTextWalkerFunction<F>
visitorFunction) {
        this.visitorFunction = visitorFunction;
    }
}
```

```

@Override
public T process(NLPText text) {
    visitorFunction.init();
    return recursiveProcess(text);
}

private T recursiveProcess(NLPText text) {
    visitorFunction.begin(text);
    for (NLPText child : text.getChildren()) {
        recursiveProcess(child);
    }
    return transformResults(visitorFunction.end(text));
}

/**
 * This method must be overridden to transform the results of the
 * NLPTextWalkerFunction to the result type of the NLPTextWalker
process()
 * method.
 *
 * @param result
 * @return
 */
public abstract T transformResults(F result);
}

```

abstractopennlptool.java

```

/*
 * $Id: AbstractOpenNLPTool.java,v 1.2 2009/01/24 11:08:38 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.zip.GZIPInputStream;

import opennlp.maxent.GISModel;
import opennlp.maxent.io.BinaryGISModelReader;
import opennlp.maxent.io.GISModelReader;

import org.apache.log4j.Logger;

```

```

import edu.harvard.fas.rregan.nlp.NLPProcessor;

/**
 * Base class for OpenNLP tool wrappers with helpers for loading GIS
models.
 *
 * @author ron
 * @param <T> -
 *          the type of results returned by the process method
 */
public abstract class AbstractOpenNLPTool<T> implements
NLPProcessor<T> {
    private static final Logger log =
Logger.getLogger(AbstractOpenNLPTool.class);

    protected static GISModel readGISModel(String modelFile) throws
IOException {
        log.debug("loading GISModel file " + modelFile);
        InputStream modelInputStream =
Sentencizer.class.getClassLoader().getResourceAsStream(
            modelFile);
        if (modelFile.endsWith(".gz")) {
            modelInputStream = new GZIPInputStream(modelInputStream);
            modelFile = modelFile.substring(0, modelFile.length() - 3);
        }

        GISModelReader modelReader = null;
        if (modelFile.endsWith(".bin")) {
            modelReader = new BinaryGISModelReader(new
DataInputStream(modelInputStream));
        } else {
            log.error("unsupported GISModel file format " + modelFile);
            // is this the only supported type?
        }

        if (modelReader != null) {
            return modelReader.getModel();
        }
        throw new IOException("Could not read model file " + modelFile + ",
unknown type.");
    }
}

```

abstractpanel.java

```
/*
 * $Id: AbstractPanel.java,v 1.15 2008/09/13 00:33:43 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.util.HashSet;
import java.util.Locale;
import java.util.Set;

import nextapp.echo2.app.event.ActionListener;
import echopointng.ContentPaneEx;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * This extends ContentPaneEx so that multiple children can be added
 * to the
 * panel easily. Otherwise a separate column would need to be added
 * and that
 * would complicate extending this panel to add more children.
 *
 * @author ron
 */
public abstract class AbstractPanel extends ContentPaneEx implements
Panel {
    static final long serialVersionUID = 0L;

    private final PanelActionType supportedActionType;
    private final Class<?> supportedContentType;
    private final String panelName;
    private boolean initialized = false;

    private Object destinationObject;
    private Object targetObject;
    private final ResourceBundleHelper resourceBundleHelper;
    private final Set<ActionListener> actionListeners = new
HashSet<ActionListener>();
    private boolean loadingState;

    protected AbstractPanel(String resourceBundleName, PanelActionType
supportedActionType,
```

```
        Class<?> supportedContentType) {
    this(resourceBundleName, supportedActionType, supportedContentType,
null);
}

protected AbstractPanel(String resourceBundleName, String panelName,
        Class<?> supportedContentType) {
    this(resourceBundleName, PanelActionType.Unspecified,
supportedContentType, panelName);
}

protected AbstractPanel(String resourceBundleName, PanelActionType
supportedActionType,
        Class<?> supportedContentType, String panelName) {
    super();
    resourceBundleHelper = new ResourceBundleHelper(resourceBundleName);
    this.supportedActionType = supportedActionType;
    this.supportedContentType = supportedContentType;
    this.panelName = panelName;
}

protected ResourceBundleHelper getResourceBundleHelper(Locale locale)
{
    resourceBundleHelper.setLocale(locale);
    return resourceBundleHelper;
}

public PanelActionType getSupportedActionType() {
    return supportedActionType;
}

public Class<?> getSupportedContentTypes() {
    return supportedContentType;
}

public String getPanelName() {
    return panelName;
}

public Class<? extends Panel> getPanelType() {
    return this.getClass();
}

public String getTitle() {
    return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE, "");}
```

```

public boolean isInitialized() {
    return initialized;
}

public void setup() {
    initialized = true;
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    initialized = false;
}

/**
 * @return true if the panel is in the loadState() method, false
 * otherwise.
 */
public boolean isLoadingState() {
    return loadingState;
}

public Object getTargetObject() {
    return targetObject;
}

public void setTargetObject(Object targetObject) {
    this.targetObject = targetObject;
}

public Object getDestinationObject() {
    return destinationObject;
}

public void setDestinationObject(Object destinationObject) {
    this.destinationObject = destinationObject;
}

/**
 * @param actionListener
 */
public void addActionListener(ActionListener actionListener) {
    actionListeners.add(actionListener);
}

```

```

/**
 * @param actionListener
 */
public void removeActionListener(ActionListener actionListener) {
    actionListeners.remove(actionListener);
}

/**
 * @return
 */
public EventDispatcher getEventDispatcher() {
    return getApp().getEventDispatcher();
}

/**
 * @return
 */
public UIFrameworkApp getApp() {
    return UIFrameworkApp.getApp();
}

```

abstractpanelcontainerpanel.java

```

/*
 * $Id: AbstractPanelContainerPanel.java,v 1.4 2008/02/29 19:36:12
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import edu.harvard.fas.rregan.uiframework.PanelContainer;

/**
 * @author ron
 */
public abstract class AbstractPanelContainerPanel extends
AbstractPanel implements PanelContainer {

    private final PanelManager panelManager;

    /**
     * @param resourceBundleName
     * @param panelManager
     */

```

```

protected AbstractPanelContainerPanel(String resourceBundleName,
PanelManager panelManager) {
    super(resourceBundleName, PanelActionType.Unspecified,
Object.class);
    panelManager.setPanelContainer(this);
    this.panelManager = panelManager;
}

/**
 * The PanelManager manages the creation and state of panels
displayed in
 * the container while the PanelContainer manages how the panels are
 * displayed. The PanelManager will listen for navigation events from
the
 * EventDispatcher concerning panels in the container and call
methods on
 * the container to alter the display. Each container will have its
own
 * manager instance.
 *
 * @return the manager for this panel
 */
// This is public so that the NavigationEventHandlers
public PanelManager getPanelManager() {
    return panelManager;
}

```

abstractpanelcontainerscreen.java

```

/*
 * $Id: AbstractPanelContainerScreen.java,v 1.2 2008/02/29 19:36:08
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.screen;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.PanelContainer;
import edu.harvard.fas.rregan.uiframework.panel.PanelManager;

/**
 * @author ron
 */

```

```

public abstract class AbstractPanelContainerScreen extends
AbstractScreen implements PanelContainer {
    private static final Logger log =
Logger.getLogger(AbstractPanelContainerScreen.class);
    static final long serialVersionUID = 0;

    private final PanelManager panelManager;

    /**
     *
     */
protected AbstractPanelContainerScreen(String resourceBundleName,
PanelManager panelManager) {
    super(resourceBundleName);
    panelManager.setPanelContainer(this);
    this.panelManager = panelManager;
}

public PanelManager getPanelManager() {
    return panelManager;
}
}
```

abstractprojectcommand.java

```

/*
 * $Id: AbstractProjectCommand.java,v 1.7 2009/03/30 11:54:30 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.impl.command.AbstractUserCommand;

/**
```

```

* @author ron
*/
public abstract class AbstractProjectCommand extends
AbstractUserCommand {

    private final CommandHandler commandHandler;
    private final ProjectCommandFactory projectCommandFactory;
    private final AnnotationCommandFactory annotationCommandFactory;
    private final ProjectRepository projectRepository;
    private final AssistantFacade assistantManager;

    protected AbstractProjectCommand(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory,
        CommandHandler commandHandler) {
        super(userRepository);
        this.assistantManager = assistantManager;
        this.projectRepository = projectRepository;
        this.projectCommandFactory = projectCommandFactory;
        this.annotationCommandFactory = annotationCommandFactory;
        this.commandHandler = commandHandler;
    }

    protected ProjectRepository getProjectRepository() {
        return projectRepository;
    }

    protected AssistantFacade getAssistantManager() {
        return assistantManager;
    }

    protected ProjectCommandFactory getProjectCommandFactory() {
        return projectCommandFactory;
    }

    protected AnnotationCommandFactory getAnnotationCommandFactory() {
        return annotationCommandFactory;
    }

    protected CommandHandler getCommandHandler() {
        return commandHandler;
    }
}

```

abstractprojectordomain.java

```

/*
 * $Id: AbstractProjectOrDomain.java,v 1.48 2009/03/05 08:50:46 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.io.Serializable;
import java.util.Date;
import java.util.Deque;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.Lob;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;

```

```

import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.annotations.Where;
import org.hibernate.validator.NotEmpty;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectTeam;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "pods", uniqueConstraints =
{ @UniqueConstraint(columnNames = { "type", "name" }) })
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "type", discriminatorType =
DiscriminatorType.STRING, length = 255)
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public abstract class AbstractProjectOrDomain implements
ProjectOrDomain, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
}

```

```

private String name;
private String description;
private User createdBy;
private Date dateCreated = new Date();
private Set<Actor> actors = new TreeSet<Actor>();
private Set<Goal> goals = new TreeSet<Goal>();
private Set<Story> stories = new TreeSet<Story>();
private Set<UseCase> useCases = new TreeSet<UseCase>();
private Set<Scenario> scenarios = new TreeSet<Scenario>();
private Set<Stakeholder> stakeholders = new TreeSet<Stakeholder>();
private Set<ProjectTeam> teams = new TreeSet<ProjectTeam>();
private SortedSet<GlossaryTerm> terms = new TreeSet<GlossaryTerm>();
private Set<ReportGenerator> reportGenerators = new
TreeSet<ReportGenerator>();

private String type;
private int version = 1;

protected AbstractProjectOrDomain(String type, String name, User
createdBy) {
    setType(type);
    setName(name);
    setCreatedBy(createdBy);
    setDateCreated(new Date());
}

protected AbstractProjectOrDomain() {
    // for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return id;
}

protected void setId(Long id) {
    this.id = id;
}

@Version
@XmlAttribute(name = "revision", required = false)
protected int getVersion() {
    return version;
}

```

```

}

protected void setVersion(int version) {
    this.version = version;
}

@Column(name = "type", insertable = false, updatable = false)
protected String getType() {
    return type;
}

protected void setType(String type) {
    this.type = type;
}

@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.REFRESH }, fetch = FetchType.LAZY)
public User getCreatedBy() {
    return createdBy;
}

/**
 * @param createdBy
 */
public void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

@Column(nullable = false)
@NotEmpty(message = "a name is required.")
XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

XmlElement(name = "description", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@Lob
public String getText() {
    return description;
}

public void setText(String description) {
    this.description = description;
}

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

XmlElementWrapper(name = "glossary", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = GlossaryTermImpl.class)
@OneToMany(targetEntity = GlossaryTermImpl.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY)
@Sort(type = SortType.NATURAL)
@Override
public SortedSet<GlossaryTerm> getGlossaryTerms() {
    return terms;
}

protected void setGlossaryTerms(SortedSet<GlossaryTerm> terms) {
    this.terms = terms;
}

XmlElementWrapper(name = "actors", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = ActorImpl.class)
@OneToMany(targetEntity = ActorImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<Actor> getActors() {
    return actors;
}

protected void setActors(Set<Actor> actors) {
}

```

```

    this.actors = actors;
}

@XmlElementWrapper(name = "goals", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = GoalImpl.class)
@OneToMany(targetEntity = GoalImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<Goal> getGoals() {
    return goals;
}

protected void setGoals(Set<Goal> goals) {
    this.goals = goals;
}

@XmlElementWrapper(name = "stories", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = StoryImpl.class)
@OneToMany(targetEntity = StoryImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<Story> getStories() {
    return stories;
}

protected void setStories(Set<Story> stories) {
    this.stories = stories;
}

@XmlElementWrapper(name = "usecases", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = UseCaseImpl.class)
@OneToMany(targetEntity = UseCaseImpl.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<UseCase> getUseCases() {
    return useCases;
}

protected void setUseCases(Set<UseCase> useCases) {
    this.useCases = useCases;
}

}

@XmlTransient
@OneToMany(targetEntity = ScenarioImpl.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"projectOrDomain")
@Where(clause = "type like '%Scenario'")
@Sort(type = SortType.NATURAL)
public Set<Scenario> getScenarios() {
    return scenarios;
}

protected void setScenarios(Set<Scenario> scenarios) {
    this.scenarios = scenarios;
}

/**
 * This is a helper for JAXB to export all the steps and scenarios at
one
 * level and scenarios to use id references to identify the steps
 * internally. This allows for shared sub-scenarios and steps to be
exported
 * without duplication.
 *
 * @return a set of all the scenarios and steps in the project
 */
@XmlElementWrapper(name = "scenarios", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = StepImpl.class)
@Transient
public Set<Step> getAllScenariosAndSteps() {
    Set<Step> scenariosAndSteps = new HashSet<Step>(getScenarios());
    Deque<Scenario> scenariosToExamine = new
LinkedList<Scenario>(getScenarios());
    while (!scenariosToExamine.isEmpty()) {
        Scenario scenario = scenariosToExamine.pop();
        for (Step step : scenario.getSteps()) {
            scenariosAndSteps.add(step);
            if (step instanceof Scenario) {
                scenariosToExamine.add((Scenario) step);
            }
        }
    }
    return scenariosAndSteps;
}

```

```

@XmlElementWrapper(name = "stakeholders", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(name = "stakeholder", type = StakeholderImpl.class)
@OneToMany(targetEntity = StakeholderImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<Stakeholder> getStakeholders() {
    return stakeholders;
}

protected void setStakeholders(Set<Stakeholder> stakeholders) {
    this.stakeholders = stakeholders;
}

@XmlElementWrapper(name = "teams", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(name = "team", type = ProjectTeamImpl.class)
@OneToMany(targetEntity = ProjectTeamImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<ProjectTeam> getTeams() {
    return teams;
}

protected void setTeams(Set<ProjectTeam> teams) {
    this.teams = teams;
}

@Override
@XmlElementWrapper(name = "reports", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(name = "report", type = ReportGeneratorImpl.class)
@OneToMany(targetEntity = ReportGeneratorImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"projectOrDomain")
@Sort(type = SortType.NATURAL)
public Set<ReportGenerator> getReportGenerators() {
    return reportGenerators;
}

protected void setReportGenerators(Set<ReportGenerator>
reportGenerators) {

```

```

    this.reportGenerators = reportGenerators;
}

/**
 * @return all the entities of a project or domain in a single set.
 */
@Override
@Transient
@XmlTransient
public Set<ProjectOrDomainEntity> getProjectEntities() {
    Set<ProjectOrDomainEntity> projectEntities = new
HashSet<ProjectOrDomainEntity>();
    projectEntities.addAll(getActors());
    projectEntities.addAll(getGoals());
    projectEntities.addAll(getScenarios());
    for (Scenario scenario : getScenarios()) {
        projectEntities.addAll(scenario.getSteps());
    }
    projectEntities.addAll(getStakeholders());
    projectEntities.addAll(getStories());
    projectEntities.addAll(getUseCases());
    projectEntities.addAll(getGlossaryTerms());
    projectEntities.addAll(getReportGenerators());
    projectEntities.addAll(getTeams());
    return projectEntities;
}

/**
 * @return The interface of this object that is a subclass of
 *         ProjectOrDomain, basically getting the type of domain
 *         entity of a
 *         subclass.
 */
@Transient
public Class<?> getProjectOrDomainInterface() {
    Class<?> type = getClass();
    while (AbstractProjectOrDomain.class.isAssignableFrom(type)) {
        for (Class<?> face : type.getInterfaces()) {
            if (ProjectOrDomain.class.isAssignableFrom(face)) {
                return face;
            }
        }
        type = type.getSuperclass();
    }
    throw new RuntimeException("The class " + type.getSimpleName()
        + " is not a descendent of ProjectOrDomainEntity.");
}

```

```

@Override
public String toString() {
    return getProjectOrDomainInterface().toString() + "[" + getId() +
"]:" + getName();
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
            result = prime * result + ((getType() == null) ? 0 :
getType().hashCode());
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
proxy
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final AbstractProjectOrDomain other = (AbstractProjectOrDomain) obj;
    if (getName() == null) {
        if (other.getName() != null) {
            return false;
        }
    } else if (!getName().equals(other.getName())) {
        return false;
    }
}

```

```

    }
    if (getType() == null) {
        if (other.getType() != null) {
            return false;
        }
    } else if (!getType().equals(other.getType())) {
        return false;
    }
    return true;
}

/**
 * This is for JAXB to patchup the type
 *
 * @see UnmarshallerListener
 */
public void beforeUnmarshal() {
    setType(getClass().getName());
}

/**
 * This is for JAXB to patchup existing persistent objects for the
objects
 * that are attached directly to this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
supplied.
 * @see UnmarshallerListener
 */
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
    // update the references to goals
    for (Goal goal : getGoals()) {
        goal.getReferers().add(AbstractProjectOrDomain.this);
    }
}

/**
 * This class is used by JAXB to convert the id of an entity into an
xml id
 * string that will be distinct from other entity xml id strings by
the use

```

```

 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "POD_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

abstractprojectordomainentity.java

```

/*
 * $Id: AbstractProjectOrDomainEntity.java,v 1.50 2009/03/05 08:50:46
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl;

import java.io.Serializable;
import java.util.Date;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

```

```

import javax.persistence.ManyToOne;
import javax.persistence.MappedSuperclass;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.impl.AbstractAnnotation;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBAnnotatablePatcher;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@MappedSuperclass
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public abstract class AbstractProjectOrDomainEntity implements
ProjectOrDomainEntity, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;

```

```

private String name;
private ProjectOrDomain projectOrDomain;
private Set<Annotation> annotations = new TreeSet<Annotation>();
private SortedSet<GlossaryTerm> terms = new TreeSet<GlossaryTerm>();
private User createdBy;
private Date dateCreated = new Date();
private int version = 1; // start at 1 so hibernate recognizes the
new

// instance as the initial value and not stale.

protected AbstractProjectOrDomainEntity(ProjectOrDomain
projectOrDomain, User createdBy,
String name) {
setProjectOrDomain(projectOrDomain);
setCreatedBy(createdBy);
setName(name);
setDateCreated(new Date());
}

protected AbstractProjectOrDomainEntity() {
// for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
protected Long getId() {
return id;
}

protected void setId(Long id) {
this.id = id;
}

@Version
@XmlAttribute(name = "revision", required = false)
protected int getVersion() {
return version;
}

protected void setVersion(int version) {
this.version = version;
}

// NOTE: this is transient and must be over ridden in sub class so
that the
// behavior (required or uniqueness can be specified in the sub
class.
@Transient
@XmlTransient
public String getName() {
return name;
}

/**
 * @param name
 */
public void setName(String name) {
this.name = name;
}

@XmlTransient
@ManyToOne(targetEntity = AbstractProjectOrDomain.class, cascade =
{ CascadeType.REFRESH }, optional = false)
public ProjectOrDomain getProjectOrDomain() {
return projectOrDomain;
}

protected void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
this.projectOrDomain = projectOrDomain;
}

XmlElementWrapper(name = "annotations", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "annotationRef", type = AbstractAnnotation.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = AbstractAnnotation.class, cascade =
{ CascadeType.PERSIST,
CascadeType.REFRESH }, fetch = FetchType.LAZY)
@Sort(type = SortType.NATURAL)
public Set<Annotation> getAnnotations() {
return annotations;
}

protected void setAnnotations(Set<Annotation> annotations) {
this.annotations = annotations;
}

XmlElementWrapper(name = "glossaryTerms", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF

```

```

@XmlElement(name = "glossaryTermRef", type = GlossaryTermImpl.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = GlossaryTermImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@Sort(type = SortType.NATURAL)
public Set<GlossaryTerm> getGlossaryTerms() {
    return terms;
}

protected void setGlossaryTerms(SortedSet<GlossaryTerm> terms) {
    this.terms = terms;
}

@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
public User getCreatedBy() {
    return createdBy;
}

protected void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

/**
 * @return The interface of this object that is a subclass of
 *         ProjectOrDomainEntity interface, basically getting the
 *         type of
 *         domain entity of a subclass.
 */
@Transient
public Class<?> getProjectOrDomainEntityInterface() {

```

```

Class<?> type = getClass();
while (AbstractProjectOrDomainEntity.class.isAssignableFrom(type)) {
    for (Class<?> face : type.getInterfaces()) {
        if (ProjectOrDomainEntity.class.isAssignableFrom(face)) {
            return face;
        }
    }
    type = type.getSuperclass();
}
throw new RuntimeException("The class " + type.getSimpleName()
    + " is not a descendent of ProjectOrDomainEntity.");
}

@Override
public String toString() {
    return getProjectOrDomainEntityInterface().toString() + "[" +
getId() + "]:" + getName();
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result +
getProjectOrDomainEntityInterface().hashCode();
            result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
            result = prime * result
                + ((getProjectOrDomain() == null) ? 0 :
getProjectOrDomain().hashCode());
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }

```

```

if (obj == null) {
    return false;
}
// NOTE: getClass().equals(obj.getClass()) fails when the obj is a
proxy
if (!getClass().isAssignableFrom(obj.getClass())) {
    return false;
}
final AbstractProjectOrDomainEntity other =
(AbstractProjectOrDomainEntity) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}

if (getName() == null) {
    if (other.getName() != null) {
        return false;
    }
} else if (!getName().equals(other.getName())) {
    return false;
}
if (getProjectOrDomain() == null) {
    if (other.getProjectOrDomain() != null) {
        return false;
    }
} else if (!getProjectOrDomain().equals(other.getProjectOrDomain()))
{
    return false;
}
return true;
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */

```

```

public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser,
    Object parent) {
    setProjectOrDomain((ProjectOrDomain) parent);

    UnmarshallingContext.getInstance().addPatcher(new
JAXBAnnotatablePatcher(this));
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
}
}

```

abstractrepository.java

```

/*
 * $Id: AbstractRepository.java,v 1.2 2009/01/03 10:24:35 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;

/**
 * @author ron
 */
public abstract class AbstractRepository implements Repository {
    protected static final Logger log =
Logger.getLogger(AbstractRepository.class);

    private final ExceptionMapper exceptionMapper;

    protected AbstractRepository(ExceptionMapper exceptionMapper) {
        this.exceptionMapper = exceptionMapper;
    }

    /**
     * Proxies and lazy loading are not implemented in the simple
repository so
     * just return the supplied object.
     */
    @Override
    public <T> T initialize(T entity) throws EntityException {

```

```

    return entity;
}

public void flush() throws EntityException {
    // nothing to do
}

protected ExceptionMapper getExceptionMapper() {
    return exceptionMapper;
}

protected void addExceptionAdapter(Class<? extends Throwable>
exceptionType,
        EntityExceptionAdapter adapter, Class<?>... entityClasses) {
    getExceptionMapper().addExceptionAdapter(exceptionType, adapter,
entityClasses);
}

public RuntimeException convertException(Exception exception) {
    return getExceptionMapper().convertException(exception);
}

public RuntimeException convertException(Exception exception, Class<?
> entityType,
        Object entity, EntityExceptionActionType actionType) {
    return getExceptionMapper().convertException(exception, entityType,
entity, actionType);
}
}

```

abstractrequeulanotationeditorpanel.java

```

/*
 * $Id: AbstractRequeulanotationEditorPanel.java,v 1.4 2009/02/21
10:32:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requeul.ui.annotation;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requeul.annotation.Annotatable;
import edu.harvard.fas.rregan.requeul.annotation.Annotation;
import edu.harvard.fas.rregan.requeul.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requeul.project.GoalRelation;
import edu.harvard.fas.rregan.requeul.project.Project;
import edu.harvard.fas.rregan.requeul.project.ProjectOrDomain;

```

```

import edu.harvard.fas.rregan.requeul.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requeul.project.Stakeholder;
import edu.harvard.fas.rregan.requeul.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requeul.ui.AbstractRequeulEditorPanel;
import edu.harvard.fas.rregan.requeul.user.User;

/**
 * Base class for Annotation editors with access to an annotation
repository.
 *
 * @author ron
 */
public abstract class AbstractRequeulanotationEditorPanel extends
AbstractRequeulEditorPanel {

    private final AnnotationRepository annotationRepository;

    /**
     * @param resourceBundleName
     * @param supportedContentType
     * @param commandHandler
     * @param annotationRepository
     */
    public AbstractRequeulanotationEditorPanel(String resourceBundleName,
        Class<?> supportedContentType, CommandHandler commandHandler,
        AnnotationRepository annotationRepository) {
        super(resourceBundleName, supportedContentType, commandHandler);
        this.annotationRepository = annotationRepository;
    }

    /**
     * @param resourceBundleName
     * @param supportedContentType
     * @param panelName
     * @param commandHandler
     * @param annotationRepository
     */
    public AbstractRequeulanotationEditorPanel(String resourceBundleName,
        Class<?> supportedContentType, String panelName, CommandHandler
commandHandler,
        AnnotationRepository annotationRepository) {
        super(resourceBundleName, supportedContentType, panelName,
commandHandler);
        this.annotationRepository = annotationRepository;
    }
}

```

```

protected AnnotationRepository getAnnotationRepository() {
    return annotationRepository;
}

protected Annotatable getAnnotatable() {
    if (getTargetObject() instanceof Annotatable) {
        return (Annotatable) getTargetObject();
    }
    return null;
}

protected Object getGroupingObject() {
    // TODO: project classes are leaking into non-project package
    if (getAnnotatable() instanceof ProjectOrDomainEntity) {
        ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
getAnnotatable();
        return entity.getProjectOrDomain();
    } else if (getAnnotatable() instanceof ProjectOrDomain) {
        return getAnnotatable();
    } else if (getAnnotatable() instanceof GoalRelation) {
        GoalRelation entity = (GoalRelation) getAnnotatable();
        return entity.getFromGoal().getProjectOrDomain();
    }
    return null;
}

@Override
public boolean isReadOnlyMode() {
    boolean projectEntity = isProjectEntity(getAnnotatable());
    Stakeholder stakeholder = getUserStakeholder();
    if (stakeholder != null) {
        return !stakeholder.hasPermission(Annotation.class,
StakeholderPermissionType.Edit);
    }
    return projectEntity;
}

@Override
protected boolean isShowDelete() {
    Stakeholder stakeholder = getUserStakeholder();
    if (stakeholder != null) {
        return !stakeholder.hasPermission(Annotation.class,
StakeholderPermissionType.Delete);
    }
    return true;
}

```

```

protected Stakeholder getUserStakeholder() {
    User user = (User) getApp().getUser();
    Annotatable annotatable = getAnnotatable();
    if (annotatable != null) {
        Stakeholder stakeholder = null;
        if (annotatable instanceof Project) {
            Project project = (Project) annotatable;
            stakeholder = project.getUserStakeholder(user);
        } else if (annotatable instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity podEntity = (ProjectOrDomainEntity)
annotatable;
            if (podEntity.getProjectOrDomain() instanceof Project) {
                Project project = (Project) podEntity.getProjectOrDomain();
                stakeholder = project.getUserStakeholder(user);
            }
        }
        if (stakeholder != null) {
            return stakeholder;
        }
    }
    return null;
}

protected boolean isProjectEntity(Annotatable annotatable) {
    boolean projectEntity = false;
    if (annotatable != null) {
        if ((annotatable instanceof Project) || (annotatable instanceof
ProjectOrDomainEntity)) {
            projectEntity = true;
        }
    }
    return projectEntity;
}

```

abstractrequelcommandcontroller.java

```

package edu.harvard.fas.rregan.requell.ui;

import edu.harvard.fas.rregan.command.CommandFactory;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron

```

```

*/
public abstract class AbstractRequelCommandController extends
AbstractRequelController {

    private CommandFactory commandFactory;
    private CommandHandler commandHandler;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     */
    protected AbstractRequelCommandController(EventDispatcher
eventDispatcher,
        CommandFactory commandFactory, CommandHandler commandHandler) {
        super(eventDispatcher);
        setCommandFactory(commandFactory);
        setCommandHandler(commandHandler);
    }

    /**
     * @param commandFactory
     * @param commandHandler
     */
    protected AbstractRequelCommandController(CommandFactory
commandFactory,
        CommandHandler commandHandler) {
        super();
        setCommandFactory(commandFactory);
        setCommandHandler(commandHandler);
    }

    protected <T> T getCommandFactory() {
        return (T) commandFactory;
    }

    protected void setCommandFactory(CommandFactory commandFactory) {
        this.commandFactory = commandFactory;
    }

    protected CommandHandler getCommandHandler() {
        return commandHandler;
    }

    protected void setCommandHandler(CommandHandler commandHandler) {
        this.commandHandler = commandHandler;
    }
}

```

```

}

```

abstractrequelcomponent.java

```

/*
 * $Id: AbstractRequelComponent.java,v 1.4 2008/10/11 21:47:44 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requelandui;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requelanduser.User;
import
edu.harvard.fas.rregan.uiframework.panel.editor.AbstractComponent;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class AbstractRequelComponent extends AbstractComponent {
    static final long serialVersionUID = 0L;

    protected AbstractRequelComponent(EditMode editMode) {
        super(editMode);
    }

    protected AbstractRequelComponent(EditMode editMode,
ResourceBundleHelper resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
    }

    protected User getCurrentUser() {
        return (User) getApp().getUser();
    }
}

```

abstractrequelcontroller.java

```

/*
 * $Id: AbstractRequelController.java,v 1.4 2009/01/27 09:30:19 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requelandui;

```

```

import org.apache.log4j.Logger;
import
edu.harvard.fas.rregan.uiframework.controller.AbstractController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
/**
 * @author ron
 */
public abstract class AbstractRequelController extends
AbstractController {
    static final long serialVersionUID = 0;
    protected static final Logger log =
Logger.getLogger(AbstractRequelController.class);
protected AbstractRequelController() {
    super();
}
protected AbstractRequelController(EventDispatcher eventDispatcher) {
    super(eventDispatcher);
}
}

```

abstractrequeleditorpanel.java

```

/*
 * $Id: AbstractRequelEditorPanel.java,v 1.3 2008/12/13 00:41:59
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requelandui;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requelanduser.User;
import
edu.harvard.fas.rregan.uiframework.panel.editor.AbstractEditorPanel;
/**
 * Base class for Requel specific editors.
 *
 * @author ron
 */
public class AbstractRequelEditorPanel extends AbstractEditorPanel {

```

```

static final long serialVersionUID = 0L;
private final CommandHandler commandHandler;
protected AbstractRequelEditorPanel(String resourceBundleName,
Class<?> supportedContentType,
String panelName, CommandHandler commandHandler) {
super(resourceBundleName, supportedContentType, panelName);
this.commandHandler = commandHandler;
}
protected AbstractRequelEditorPanel(String resourceBundleName,
Class<?> supportedContentType,
CommandHandler commandHandler) {
this(resourceBundleName, supportedContentType, null,
commandHandler);
}
protected User getCurrentUser() {
return (User) getApp().getUser();
}
protected CommandHandler getCommandHandler() {
return commandHandler;
}
}

```

abstractrequelelnavigatortable.java

```

/*
 * $Id: AbstractRequelNavigatorTable.java,v 1.1 2009/01/08 06:48:46
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requelandui;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import
edu.harvard.fas.rregan.uiframework.panel.editor.AbstractComponent;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
/**
 * Base class for Navigator tables that get embedded in editors that
over rides
 * setEnabled() so that when an editor is in read-only mode the table
sorting

```

```

* and paging still works.<br>
* This is because the EditorComponents addInput() disables the input
component
* if the editor isReadOnlyMode() method return true.
*
* @author ron
*/
public class AbstractRequelNavigatorTable extends AbstractComponent {
    static final long serialVersionUID = 0L;

    protected AbstractRequelNavigatorTable(EditMode editMode) {
        super(editMode);
    }

    protected AbstractRequelNavigatorTable(EditMode editMode,
                                           ResourceBundleHelper resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
    }

    @Override
    public void setEnabled(boolean newValue) {
        // the table is always enabled, this gets called with true when
        // the editor is read-only, but that disables the paging buttons.
        super.setEnabled(true);
    }
}

```

abstractrequelprojecteditorpanel.java

```

/*
 * $Id: AbstractRequelProjectEditorPanel.java,v 1.3 2009/02/21
10:32:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelEditorPanel;

```

```

import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public class AbstractRequelProjectEditorPanel extends
AbstractRequelEditorPanel {
    static final long serialVersionUID = 0L;

    private final ProjectCommandFactory projectCommandFactory;
    private final ProjectRepository projectRepository;

    /**
     * @param resourceBundleName
     * @param supportedContentType
     * @param panelName
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */
    public AbstractRequelProjectEditorPanel(String resourceBundleName,
                                           Class<?> supportedContentType, String panelName, CommandHandler
commandHandler,
                                           ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
        super(resourceBundleName, supportedContentType, panelName,
commandHandler);
        this.projectCommandFactory = projectCommandFactory;
        this.projectRepository = projectRepository;
    }

    /**
     * @param resourceBundleName
     * @param supportedContentType
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */
    public AbstractRequelProjectEditorPanel(String resourceBundleName,
                                           Class<?> supportedContentType, CommandHandler commandHandler,
                                           ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
        super(resourceBundleName, supportedContentType, commandHandler);
        this.projectCommandFactory = projectCommandFactory;
        this.projectRepository = projectRepository;
    }
}

```

```

protected ProjectCommandFactory getProjectCommandFactory() {
    return projectCommandFactory;
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public boolean isReadOnlyMode() {
    Stakeholder stakeholder = getUserStakeholder(getTargetObject());
    if (stakeholder != null) {
        return !stakeholder.hasPermission(getSupportedContentType(),
            StakeholderPermissionType.Edit);
    }
    return true;
}

@Override
protected boolean isShowDelete() {
    Stakeholder stakeholder = getUserStakeholder(getTargetObject());
    if (stakeholder != null) {
        return stakeholder.hasPermission(getSupportedContentType(),
            StakeholderPermissionType.Delete);
    }
    return false;
}

protected Stakeholder getUserStakeholder(Object target) {
    User user = (User) getApp().getUser();
    Project project = null;
    Stakeholder stakeholder = null;
    if (target instanceof Project) {
        project = (Project) target;
        stakeholder = project.getUserStakeholder(user);
    } else if (target instanceof ProjectOrDomainEntity) {
        ProjectOrDomainEntity podEntity = (ProjectOrDomainEntity) target;
        if (podEntity.getProjectOrDomain() instanceof Project) {
            project = (Project) podEntity.getProjectOrDomain();
            stakeholder = project.getUserStakeholder(user);
        }
    }
    if (stakeholder != null) {
        return stakeholder;
    }
    return null;
}

```

```

}
```

abstractscreen.java

```

/*
 * $Id: AbstractScreen.java,v 1.5 2008/05/06 09:15:42 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.screen;

import java.util.LinkedList;
import java.util.List;
import java.util.Locale;

import nextapp.echo2.app.ContentPane;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * This is the base class for ui elements that encompass the whole
 * application
 * browser window.
 *
 * @author ron
 */
public class AbstractScreen extends ContentPane implements Screen {
    private static final Logger log =
        Logger.getLogger(AbstractScreen.class);
    static final long serialVersionUID = 0L;

    private final List<ActionListener> actionListeners =
        new LinkedList<ActionListener>();
    private final ResourceBundleHelper resourceBundleHelper;

    protected AbstractScreen() {
        this(AbstractScreen.class.getName());
    }

    protected AbstractScreen(String resourceName) {

```

```

super();
resourceBundleHelper = new ResourceBundleHelper(resourceBundleName);
}

public void addActionListener(ActionListener actionListener) {
    actionListeners.add(actionListener);
}

public void removeActionListener(ActionListener actionListener) {
    actionListeners.remove(actionListener);
}

protected ResourceBundleHelper getResourceBundleHelper(Locale locale)
{
    resourceBundleHelper.setLocale(locale);
    return resourceBundleHelper;
}

public void actionPerformed(ActionEvent e) {
    for (ActionListener listener : actionListeners) {
        listener.actionPerformed(e);
    }
}

public EventDispatcher getEventDispatcher() {
    return getApp().getEventDispatcher();
}

public UIFrameworkApp getApp() {
    // this isn't set until after the window is created and
    // ManagerApp.init() is complete.
    // return (ManagerApp) getApplicationInstance();
    return UIFrameworkApp.getApp();
}

public void dispose() {
    removeAll();
}

public void setup() {
}

```

abstractsenserelationinfo.java

```
/*
```

```

 * $Id: AbstractSenseRelationInfo.java,v 1.1 2008/12/14 11:36:17
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * @author ron
 */
public class AbstractSenseRelationInfo implements SenseRelationInfo {

    private final Sense sense1;
    private final Sense sense2;
    private final double rank;
    private final String reason;

    protected AbstractSenseRelationInfo(Sense sense1, Sense sense2,
                                         double rank, String reason) {
        this.sense1 = sense1;
        this.sense2 = sense2;
        this.rank = rank;
        this.reason = reason;
    }

    public Sense getSense1() {
        return sense1;
    }

    public Sense getSense2() {
        return sense2;
    }

    public double getRank() {
        return rank;
    }

    public String getReason() {
        return reason;
    }

    @Override
    public int hashCode() {
        return getSense1().hashCode() + getSense2().hashCode();
    }
}

```

```

}

@Override
public boolean equals(Object o) {
    if ((o instanceof AbstractSenseRelationInfo) &&
getClass().equals(o.getClass())) {
        AbstractSenseRelationInfo other = (AbstractSenseRelationInfo) o;
        return (getSense1().equals(other.getSense1()) &&
(getSense2().equals(other.getSense2())))
        || (getSense1().equals(other.getSense2()) &&
(getSense2().equals(other
    .getSense1())));
    }
    return false;
}

```

abstractsysteminitializer.java

```

/*
 * $Id: AbstractSystemInitializer.java,v 1.1 2009/01/26 10:19:05
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan;

import org.apache.log4j.Logger;

/**
 * Manages the ordering of entity initializers.
 *
 * @author ron
 */
public abstract class AbstractSystemInitializer implements
SystemInitializer {
    protected static final Logger log =
Logger.getLogger(SystemInitializer.class);

    private final int order;

    protected AbstractSystemInitializer(int order) {
        this.order = order;
    }

    @Override

```

```

        public int getOrder() {
            return order;
        }

        @Override
        public int compareTo(SystemInitializer o) {
            int orderCompare = (getOrder() - o.getOrder());
            int arbitraryCompare =
(getClass().getName()).compareToIgnoreCase(o.getClass().getName());
            return (orderCompare == 0 ? arbitraryCompare : orderCompare);
        }
    }

```

abstracttextentity.java

```

/*
 * $Id: AbstractTextEntity.java,v 1.10 2009/01/07 02:22:10 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl;

import javax.persistence.Lob;
import javax.persistence.MappedSuperclass;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.TextEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@MappedSuperclass
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public abstract class AbstractTextEntity extends
AbstractProjectOrDomainEntity implements
TextEntity {
    static final long serialVersionUID = 0L;

    private String text;
}

```

```
/***
 * @param projectOrDomain
 * @param name
 * @param text
 */
protected AbstractTextEntity(ProjectOrDomain projectOrDomain, User
createdBy, String name,
    String text) {
super(projectOrDomain, createdBy, name);
setText(text);

}

protected AbstractTextEntity() {
// for hibernate
}

/***
 * @see edu.harvard.fas.rregan.requel.project.TextEntity#getText()
 */
@XmlElement(name = "text", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@Lob
public String getText() {
    return text;
}

/***
 * @see
edu.harvard.fas.rregan.requel.project.TextEntity#setText(java.lang.Str
ing)
 */
@Override
public void setText(String text) {
    this.text = text;
}

/***
 * This is for JAXB to cleanup the text of extra whitespace at the
start and
 * end of the text.
 *
 * @see UnmarshallerListener
 */
public void afterUnmarshal() {
    if (getText() != null) {
        setText(getText().trim());
    }
}
```

}

abstractusercommand.java

```
/*
 * $Id: AbstractUserCommand.java,v 1.4 2008/12/13 00:41:57 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.request.user.impl.command;

import edu.harvard.fas.rregan.command.AbstractCommand;
import edu.harvard.fas.rregan.request.user.UserRepository;

/**
 * @author ron
 */
public abstract class AbstractUserCommand extends AbstractCommand {
    protected AbstractUserCommand(UserRepository userRepository) {
        super(userRepository);
    }

    protected UserRepository getUserRepository() {
        return (UserRepository) getRepository();
    }
}
```

abstractuserrole.java

```
/*
 * $Id: AbstractUserRole.java,v 1.12 2009/01/10 11:08:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user;

import java.io.Serializable;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import javax.persistence.CascadeType;
```

```

import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlType;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBUserRolePatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "user_roles")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "role_type", discriminatorType = DiscriminatorType.STRING, length = 255)
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public abstract class AbstractUserRole implements UserRole,
Serializable {
    static final long serialVersionUID = 0L;

    protected static final Set<Class<? extends UserRole>> userRoleTypes =
new HashSet<Class<? extends UserRole>>();
    // TODO: this is public for DatabaseInitializationListener, find a
way for
    // this not to be public
    public static final Map<Class<? extends UserRole>,
Set<UserRolePermission>> userRoleTypePermissions = new HashMap<Class<? extends UserRole>, Set<UserRolePermission>>();
}

```

```

private Long id;
private Set<UserRolePermission> userRolePermissions = new
HashSet<UserRolePermission>();
private String roleType;
private int version = 1; // start at 1 so hibernate recognizes the
new

// instance as the initial value and not stale.

/**
 * @param roleName
 */
protected AbstractUserRole() {
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
protected Long getId() {
    return id;
}

protected void setId(Long id) {
    this.id = id;
}

@Version
protected int getVersion() {
    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

@ManyToMany(targetEntity = UserRolePermission.class, cascade =
{ CascadeType.MERGE,
    CascadeType.PERSIST, CascadeType.REFRESH }, fetch =
FetchType.EAGER)
@JoinTable(name = "user_roles_permissions", joinColumns =
{ @JoinColumn(name = "user_role_id") }, inverseJoinColumns =
{ @JoinColumn(name = "user_role_permission_id") })
@XmlElementWrapper(name = "userPermissions", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef
protected Set<UserRolePermission> getUserRolePermissions() {
    return userRolePermissions;
}

```

```

}

protected void setUserRolePermissions(Set<UserRolePermission>
userRolePermissions) {
    this.userRolePermissions = userRolePermissions;
}

@Override
public void grantUserRolePermission(UserRolePermission permission) {
    getUserRolePermissions().add(permission);
}

@Override
public void revokeUserRolePermission(UserRolePermission permission) {
    getUserRolePermissions().remove(permission);
}

public boolean hasUserRolePermission(UserRolePermission permission) {
    return getUserRolePermissions().contains(permission);
}

@Transient
public Set<UserRolePermission> getAvailableUserRolePermissions() {
    return getAvailableUserRolePermissions(this.getClass());
}

// this is a hack so that roles can be searched by type
@Column(name = "role_type", insertable = false, updatable = false)
protected String getRoleType() {
    return roleType;
}

protected void setRoleType(String roleType) {
    this.roleType = roleType;
}

@Transient
public String getRoleName() {
    return getRoleName(this.getClass());
}

/**
 * @return The interface of this object that is a subclass of
UserRole
 *         interface, basically getting the type of the role of a
subclass.
 */

```

```

@Transient
public Class<?> getUserRoleInterface() {
    Class<?> type = getClass();
    while (UserRole.class.isAssignableFrom(type)) {
        for (Class<?> face : type.getInterfaces()) {
            if (UserRole.class.isAssignableFrom(face)) {
                return face;
            }
        }
        type = type.getSuperclass();
    }
    throw new RuntimeException("The class " + type.getSimpleName()
        + " is not a descendent of AbstractUserRole.");
}

@Override
public String toString() {
    return getUserRoleInterface().toString() + "[" + getId() + "]";
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((getRoleName() == null) ? 0 :
getRoleName().hashCode());
            tmpHashCode = new Integer(result);
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {

```

```

    return false;
}
final AbstractUserRole other = (AbstractUserRole) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getRoleName() == null) {
    if (other.getRoleName() != null) {
        return false;
    }
} else if (!getRoleName().equals(other.getRoleName())) {
    return false;
}
return true;
}

/**
 * This is for JAXB to patchup existing persistent objects for the
objects
 * that are attached directly to this object.
 *
 * @param userRepository
 * @see UnmarshallerListener
 */
public void afterUnmarshal(Repository userRepository) {
    UnmarshallingContext.getInstance()
        .addPatcher(new JAXBUserRolePatcher(userRepository, this));
}

/**
 * @param userRoleType
 * @return
 */
public static String getRoleName(Class<? extends UserRole>
userRoleType) {
    return userRoleType.getSimpleName();
}

/**
 * @param userRoleType
 * @return
 */
public static Set<UserRolePermission>
getAvailableUserRolePermissions(
    Class<? extends UserRole> userRoleType) {

```

```

    return
    Collections.unmodifiableSet(userRoleTypePermissions.get(userRoleType))
    ;
}

/**
 * @return
 */
public static Set<Class<? extends UserRole>> getAvailableUserRoles()
{
    return Collections.unmodifiableSet(userRoleTypes);
}
}
```

actor.java

```

/*
 * $Id: Actor.java,v 1.4 2008/09/06 09:31:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

/**
 * @author ron
 */
public interface Actor extends TextEntity, GoalContainer,
Comparable<Actor> {

    /**
     * @return The referers/users of the actor.
     */
    public Set<ActorContainer> getReferers();
}
```

actor2actorimpladapter.java

```

/*
 * $Id: Actor2ActorImplAdapter.java,v 1.1 2008/09/06 09:31:58 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;
```

```

import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import edu.harvard.fas.rregan.requel.project.Actor;

/**
 * Adapter for JAXB to convert interface Actor to class ActorImpl and
back.
 *
 * @author ron
 */
@XmlTransient
public class Actor2ActorImplAdapter extends XmlAdapter<ActorImpl,
Actor> {

    @Override
    public ActorImpl marshal(Actor actor) throws Exception {
        return (ActorImpl) actor;
    }

    @Override
    public Actor unmarshal(ActorImpl actor) throws Exception {
        return actor;
    }
}

```

actorassistant.java

```

/*
 * $Id: ActorAssistant.java,v 1.4 2009/01/23 09:54:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses stories and adds annotations with suggestions.
 *
 * @author ron
 */
public class ActorAssistant extends TextEntityAssistant {

```

```

    /**
     * @param lexicalAssistant -
     *          assistant for analyzing text for spelling, terms and
other
     *          word oriented analysis.
     * @param assistantUser -
     *          the user to use as the creator of the annotation
entities.
    */
    public ActorAssistant(LexicalAssistant lexicalAssistant, User
assistantUser) {
        super(ActorAssistant.class.getName(), lexicalAssistant,
assistantUser);
    }
}

```

actorcontainer.java

```

/*
 * $Id: ActorContainer.java,v 1.4 2008/09/06 09:31:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Comparator;
import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * An abstraction of something that contains a set of Actors.
 *
 * @author ron
 */
public interface ActorContainer extends Describable, CreatedEntity {

    /**
     * @return the actors referenced by this container
     */
    public Set<Actor> getActors();

    /**
     * Compare the objects that contain Storys by the description.
     */
}

```

```

public static final Comparator<ActorContainer> COMPARATOR = new
ActorContainerComparator();

/**
 * A Comparator for collections of Story containers.
 */
public static class ActorContainerComparator implements
Comparator<ActorContainer> {
    @Override
    public int compare(ActorContainer o1, ActorContainer o2) {
        return o1.getDescription().compareTo(o2.getDescription());
    }
}

}

actorcontainerstable.java

```

```

/*
 * $Id: ActorContainersTable.java,v 1.4 2009/01/08 06:48:45 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component to add to Panels of Actor container entity editors to
enable
 * editing of the Stories of the entity.
 *
 * @author ron
 */
public class ActorContainersTable extends AbstractRequelNavigatorTable
{
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(ActorContainersTable.class,
            new ActorContainersTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * ActorContainersTable to define the label of the Actor containers
field.
     * If the property is undefined the panel should use a sensible
default such

```

```

        * as "Actor Referers".
    */
public static final String PROP_LABEL_Actor_CONTAINERS =
"ActorContainers.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the view button in the Actor containers edit table column. If
the
 * property is undefined "View" is used.
 */
public static final String PROP_VIEW_Actor_CONTAINER_BUTTON_LABEL =
"ViewActorContainer.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the edit button in the Actor container edit table column. If
the
 * property is undefined "Edit" is used.
 */
public static final String PROP_EDIT_Actor_CONTAINER_BUTTON_LABEL =
>EditActorContainer.Label;

private Actor actor;
private final NavigatorTable table;

/**
 * @param editMode
 * @param resourceBundleHelper
 */
public ActorContainersTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
    super(editMode, resourceBundleHelper);
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    table = new NavigatorTable(getTableConfig());
    table.setLayoutData(layoutData);
    add(table);
}

protected Actor getActor() {
    return actor;
}

protected void setActor(Actor actor) {

```

```

        this.actor = actor;
        if (actor != null) {
            table.setModel(new NavigatorTableModel((Collection)
actor.getReferers()));
        } else {
            table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
        }
    }

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                ActorContainer actorContainer = (ActorContainer) model
                    .getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_VIEW_Actor_CONTAINER_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_EDIT_Actor_CONTAINER_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                    PanelActionType.Editor, actorContainer,
                    actorContainer.getClass(),
                    null, WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
                NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                openEditorButton.setLayoutData(rld);
                return openEditorButton;
            }
        }));
    tableConfig.addColumnConfig(new
        NavigatorTableColumnConfig("Description",
            new NavigatorTableCellValueFactory() {
                @Override

```

```

        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    ActorContainer ActorContainer = (ActorContainer) model
        .getBackingObject(row);
    return ActorContainer.getDescription();
}
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    ActorContainer ActorContainer = (ActorContainer) model
        .getBackingObject(row);
    return ActorContainer.getCreatedBy().getUsername();
}
});
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    ActorContainer ActorContainer = (ActorContainer) model
        .getBackingObject(row);
    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
    return formatter.format(ActorContainer.getDateCreated());
}
});
}

return tableConfig;
}

private static class ActorContainersTableManipulator extends
AbstractComponentManipulator {

protected ActorContainersTableManipulator() {
super();
}

@Override
public Object getModel(Component component) {
return getValue(component, Actor.class);
}
}

```

```

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getActor());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setActor((Actor) value);
}

private ActorContainersTable getComponent(Component component) {
    return (ActorContainersTable) component;
}
}

```

actoreditorpanel.java

```

/*
 * $Id: ActorEditorPanel.java,v 1.22 2009/03/23 11:02:56 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;

```

```

import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.command.CopyActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteActorCommand;
import edu.harvard.fas.rregan.requel.project.command.EditActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.ui.annotation.AnnotationsTable;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class ActorEditorPanel extends AbstractRequelProjectEditorPanel
{
    private static final Logger log =
Logger.getLogger(ActorEditorPanel.class);

    static final long serialVersionUID = 0L;

    /**
     * The name to use in the ActorEditorPanel.properties file to set the
     * label
     * of the name field. If the property is undefined "Name" is used.
     */
    public static final String PROP_LABEL_NAME = "Name.Label";

    /**
     * The name to use in the ActorEditorPanel.properties file to set the
     * label
     * of the actor type field. If the property is undefined "Actor Type"
     * is
     * used.
     */
    public static final String PROP_LABEL_STORY_TYPE = "ActorType.Label";

    /**
     * The name to use in the ActorEditorPanel.properties file to set the
     * label
     * of the text field. If the property is undefined "Text" is used.
     */
    public static final String PROP_LABEL_TEXT = "Text.Label";

    private UpdateListener updateListener;
    private Button copyButton;

    // this is set by the DeleteListener so that the UpdateListener can
    ignore
    // events between when the object was deleted and the panel goes
    away.
    private boolean deleted;

    /**
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */
    public ActorEditorPanel(CommandHandler commandHandler,
                           ProjectCommandFactory projectCommandFactory, ProjectRepository
                           projectRepository) {
        this(ActorEditorPanel.class.getName(), commandHandler,
             projectCommandFactory,
             projectRepository);
    }

    /**
     * @param resourceBundleName
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */
    public ActorEditorPanel(String resourceBundleName, CommandHandler
                           commandHandler,

```

```

    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    super(resourceBundleName, Actor.class, commandHandler,
projectCommandFactory,
    projectRepository);
}

/**
 * If the editor is editing an existing Actor the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Actor: {0}"<br>
 * Valid variables are:<br>
 * {0} - Actor name<br>
 * {1} - project/domain name<br>
 * For new Actor it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Actor"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    if (getActor() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Actor: {0}"));
        return MessageFormat.format(msgPattern, getActor().getName(),
getProjectOrDomain()
        .getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"New Actor"));
        return msg;
    }
}

```

```

@Override
public void setup() {
    super.setup();
    Actor actor = getActor();
    if (actor != null) {
        addInput(EditActorCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
TextField(),
        new StringDocumentEx(actor.getName()));
        addInput(EditActorCommand.FIELD_TEXT, PROP_LABEL_TEXT,
"Description", new TextArea(),
        new StringDocumentEx(actor.getText()));
        addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals", new
GoalsTable(this,
            getResourceBundleHelper(getLocale()), getProjectCommandFactory(),
            getCommandHandler(), actor));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABELGLOSSARYTERM,
            "Glossary Terms", new GlossaryTermsTable(this,
                getResourceBundleHelper(getLocale()), actor));
        addMultiRowInput("actorContainers",
ActorContainersTable.PROP_LABEL_ACTOR_CONTAINERS,
            "Referring Entities", new ActorContainersTable(this,
                getResourceBundleHelper(getLocale()), actor));
        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelper(getLocale()),
actor));
        copyButton = addActionButton(new
Button(getResourceBundleHelper(getLocale()).getString(
            PROP_LABEL_COPY_BUTTON, "Copy")));
        copyButton.addActionListener(new CopyListener(this));
        copyButton.setEnabled(!isReadOnlyMode());
    } else {
        addInput(EditActorCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
TextField(),
        new StringDocumentEx());
        addInput(EditActorCommand.FIELD_TEXT, PROP_LABEL_TEXT,
"Description", new TextArea(),
        new StringDocumentEx());
        addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals", new
GoalsTable(this,
            getResourceBundleHelper(getLocale()), getProjectCommandFactory(),
            getCommandHandler(), null));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABELGLOSSARYTERM,
            "Glossary Terms", new GlossaryTermsTable(this,

```

```

        getResourceBundleHelpergetLocale()), null);
    addMultiRowInput("actorContainers",
ActorContainersTable.PROP_LABEL_Actor_CONTAINERS,
    "Referring Entities", new ActorContainersTable(this,
        getResourceBundleHelpergetLocale()), null);
    addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
    new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
null);
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener, this);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
    }
}

    EditActorCommand command =
getProjectCommandFactory().newEditActorCommand();
    command.setActor(getActor());
    command.setActorContainer(getActorContainer());
    command.setEditedBy(getCurrentUser());
    command.setName(getInputValue(EditActorCommand.FIELD_NAME,
String.class));
    command.setText(getInputValue(EditActorCommand.FIELD_TEXT,
String.class));
    command = getCommandHandler().execute(command);
    setValid(true);
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener, this);
    }
    getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getActor()));
} catch (EntityException e) {
    if (e.isStaleEntity()) {
        // TODO: compare the original values before the user edited
        // to the current revisions values and if they are the same
        // then update the new revision with the user's changes and
        // continue, otherwise show the new changed value vs. the users
        // new values.
        String newName = getInputValue(EditActorCommand.FIELD_NAME,
String.class);
        String newText = getInputValue(EditActorCommand.FIELD_TEXT,
String.class);
        Actor newActor = getProjectRepository().get(getActor());

        setTargetObject(newActor);
        if (!newName.equals(newActor.getName()) || !
newText.equals(newActor.getText())) {
            setGeneralMessage("The actor was changed by another user and the
value conflicts with your input.");
            if (!newName.equals(newActor.getName())) {
                setValidationMessage(EditActorCommand.FIELD_NAME, "Your input '" +
newName
                    + "'");
                setInputValue(EditActorCommand.FIELD_NAME, newActor.getName());
            }
            if (!newText.equals(newActor.getText())) {
                setValidationMessage(EditActorCommand.FIELD_TEXT, "Your input '" +
newText
                    + "'");
                setInputValue(EditActorCommand.FIELD_TEXT, newActor.getText());
            }
        }
    }
}

```

```

        }
    } else {
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
newActor));
    }
} else if ((e.getEntityPropertyNames() != null)
    && (e.getEntityPropertyNames().length > 0)) {
    for (String propertyName : e.getEntityPropertyNames()) {
        setValidationMessage(propertyName, e.getMessage());
    }
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the actor: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
    try {
        DeleteActorCommand deleteActorCommand = getProjectCommandFactory()
            .newDeleteActorCommand();
        deleteActorCommand.setEditedBy(getCurrentUser());
        deleteActorCommand.setActor(getActor());
        deleteActorCommand =
getCommandHandler().execute(deleteActorCommand);
        deleted = true;
        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getActor()));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private ProjectOrDomain getProjectOrDomain() {
    if (getTargetObject() instanceof ProjectOrDomain) {
        return (ProjectOrDomain) getTargetObject();
    } else if (getTargetObject() instanceof ProjectOrDomainEntity) {

```

```

        return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
    }
    return null;
}

private ActorContainer getActorContainer() {
    if (getTargetObject() instanceof ActorContainer) {
        return (ActorContainer) getTargetObject();
    }
    return null;
}

private Actor getActor() {
    if (getTargetObject() instanceof Actor) {
        return (Actor) getTargetObject();
    }
    return null;
}

private static class CopyListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ActorEditorPanel panel;

    private CopyListener(ActorEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            CopyActorCommand copyActorCommand =
panel.getProjectCommandFactory()
            .newCopyActorCommand();
            copyActorCommand.setEditedBy(panel.getCurrentUser());
            copyActorCommand.setOriginalActor(panel.getActor());
            copyActorCommand =
panel.getCommandHandler().execute(copyActorCommand);
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null,
copyActorCommand.getNewActor()));
            panel.getEventDispatcher().dispatchEvent(
                new OpenPanelEvent(this, PanelActionType.Editor,
copyActorCommand
                    .getNewActor(), Actor.class, null));
        } catch (Exception e) {

```

```

        panel.setGeneralMessage("Could not copy entity: " + e);
    }
}

// TODO: it may be better to have different standardized update
listeners
// for different types of updated or deleted objects associated with
the
// input controls like an annotation table or referers.
private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ActorEditorPanel panel;

    private UpdateListener(ActorEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        Actor currentActor = panel.getActor();
        if ((e instanceof UpdateEntityEvent) && (currentActor != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            Actor updatedActor = null;
            if (event.getObject() instanceof Actor) {
                updatedActor = (Actor) event.getObject();
                if ((event instanceof DeletedEntityEvent) &&
                    currentActor.equals(updatedActor)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, currentActor));
                    return;
                }
            } else if (event.getObject() instanceof Goal) {
                Goal updatedGoal = (Goal) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if (currentActor.getGoals().contains(updatedGoal)) {
                        currentActor.getGoals().remove(updatedGoal);
                    }
                }
                updatedActor = currentActor;
            } else if (updatedGoal.getReferers().contains(currentActor)) {
                for (GoalContainer gc : updatedGoal.getReferers()) {
                    if (gc.equals(currentActor)) {

```

```

                        updatedActor = (Actor) gc;
                        break;
                    }
                }
            }
        } else if (event.getObject() instanceof UseCase) {
            // a use case has one-to-many and many-to-many references to
            // actors
            updatedActor = currentActor;
            if (event instanceof DeletedEntityEvent) {
                updatedActor.getReferers().remove(event.getObject());
            } else {
                UseCase useCase = (UseCase) event.getObject();
                if (useCase.getPrimaryActor().equals(updatedActor)) {
                    updatedActor = useCase.getPrimaryActor();
                } else if (useCase.getActors().contains(updatedActor)) {
                    for (Actor actor : useCase.getActors()) {
                        if (actor.equals(updatedActor)) {
                            updatedActor = actor;
                            break;
                        }
                    }
                } else if (updatedActor.getReferers().contains(useCase)) {
                    updatedActor.getReferers().remove(useCase);
                }
            }
        } else if (event.getObject() instanceof ActorContainer) {
            updatedActor = currentActor;
            if (updatedActor.getReferers().contains(event.getObject())) {
                if (event instanceof DeletedEntityEvent) {
                    updatedActor.getReferers().remove(event.getObject());
                } else {
                    ActorContainer actorContainer = (ActorContainer)
                    event.getObject();
                    for (Actor actor : actorContainer.getActors()) {
                        if (currentActor.equals(actor)) {
                            updatedActor = actor;
                            break;
                        }
                    }
                }
            } else if (event.getObject() instanceof Annotation) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if (currentActor.getAnnotations().contains(updatedAnnotation)) {
                        currentActor.getAnnotations().remove(updatedAnnotation);

```

```
        }
    } else if
(updatedAnnotation.getAnnotatables().contains(currentActor)) {
    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
        if (annotatable.equals(currentActor)) {
            updatedActor = (Actor) annotatable;
            break;
        }
    }
    updatedActor = currentActor;
}
if ((updatedActor != null) && updatedActor.equals(currentActor)) {
    String editedName =
panel.getInputValue(EditActorCommand.FIELD_NAME,
    String.class);
    if (!editedName.equals(updatedActor.getName())) {
        panel.setValidationMessage(EditActorCommand.FIELD_NAME,
            "The name has changed.");
    }
    panel.setInputValue(EditActorCommand.FIELD_NAME,
updatedActor.getName());
    String editedText =
panel.getInputValue(EditActorCommand.FIELD_TEXT,
    String.class);
    if (!editedText.equals(updatedActor.getText())) {
        panel.setValidationMessage(EditActorCommand.FIELD_TEXT,
            "The text has changed.");
    }
    panel.setInputValue(EditActorCommand.FIELD_TEXT,
updatedActor.getText());
    panel.setInputValue("glossaryTerms", updatedActor);
    panel.setInputValue("goals", updatedActor);
    panel.setInputValue("actorContainers", updatedActor);
    panel.setInputValue("annotations", updatedActor);
    panel.setTargetObject(updatedActor);
}
}
}
}
```

actorimpl.java

```
/*
 * $Id: ActorImpl.java,v 1.17 2009/02/12 11:01:35 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.annotations.AnyMetaDef;
import org.hibernate.annotations.ManyToOne;
import org.hibernate.annotations.MetaValue;
import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;
import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
```

```

import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "actors", uniqueConstraints =
{ @UniqueConstraint(columnNames =
"projectordomain_id", "name" ) })
@XmlRootElement(name = "actor", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "actor", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ActorImpl extends AbstractTextEntity implements Actor {
    static final long serialVersionUID = 0L;

    private Set<ActorContainer> referers = new
TreeSet<ActorContainer>(ActorContainer.COMPARATOR);
    private Set<Goal> goals = new TreeSet<Goal>();

    /**
     * @param projectOrDomain
     * @param createdBy
     * @param name
     * @param description
     */
    public ActorImpl(ProjectOrDomain projectOrDomain, User createdBy,
String name,
        String description) {
        super(projectOrDomain, createdBy, name, description);
    }

    protected ActorImpl() {
        // for hibernate
    }

    @Override
    @Column(nullable = false, unique = false)
    @NotEmpty(message = "a unique name is required.")
    @XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
    public String getName() {
        return super.getName();
    }

    /**
     * @author ron
     */
    @Transient
    @XmlTransient
    @ManyToMany(fetch = FetchType.LAZY, metaColumn = @Column(name =
"actorcontainer_type", length = 255, nullable = false))
    @AnyMetaDef(idType = "long", metaType = "string", metaValues = {
        @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class),
        @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCase",
targetEntity = UseCaseImpl.class),
        @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Goal",
targetEntity = GoalImpl.class),
        @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Story",
targetEntity = StoryImpl.class) })
    @JoinTable(name = "actor_actorcontainers", joinColumns =
{ @JoinColumn(name = "actor_id") }, inverseJoinColumns =
{
        @JoinColumn(name = "actorcontainer_type"), @JoinColumn(name =
"actorcontainer_id") })
    @Sort(type = SortType.COMPARATOR, comparator =
ActorContainer.ActorContainerComparator.class)
    public Set<ActorContainer> getReferers() {
        return referers;
    }

    protected void setReferers(Set<ActorContainer> referers) {
        this.referers = referers;
    }
}

```

```

@Override
@XmlElementWrapper(name = "goals", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
XmlElement(name = "goalRef", type = GoalImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = GoalImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "actor_goals", joinColumns = { @JoinColumn(name =
"actor_id") }, inverseJoinColumns = { @JoinColumn(name = "goal_id") })
@Sort(type = SortType.NATURAL)
public Set<Goal> getGoals() {
    return goals;
}

protected void setGoals(Set<Goal> goals) {
    this.goals = goals;
}

@Override
public int compareTo(Actor o) {
    return getName().compareToIgnoreCase(o.getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
supplied.
 * @see UnmarshallerListener
 */
public void afterUnmarshal(final UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
        defaultCreatedByUser));
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {

```

```

                // update the references to goals
                for (Goal goal : getGoals()) {
                    goal.getReferers().add(ActorImpl.this);
                }
            } catch (RuntimeException e) {
                throw e;
            } catch (Exception e) {
                throw new SAXException2(e);
            }
        });
    }
}

```

actornavigatorpanel.java

```

/*
 * $Id: ActorNavigatorPanel.java,v 1.3 2009/02/23 07:37:23 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowStyleData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;

```

```

import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
l;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * Panel for viewing and editing glossary terms for a project or
domain.
 */
public class ActorNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(ActorNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private ActorUpdateListener updateListener;
    private ProjectOrDomain pod;

```

```

    /**
     * Property name to use in the ActorNavigatorPanel.properties to set
the
     * label on the new Actor button.
     */
    public static final String PROP_NEW_ACTOR_BUTTON_LABEL =
"NewActorButton.Label";

    /**
     * Property name to use in the ActorNavigatorPanel.properties to set
the
     * label on the edit Actor button in each row of the table.
     */
    public static final String PROP_EDIT_ACTOR_BUTTON_LABEL =
>EditActorButton.Label";

    /**
     * Property name to use in the ActorNavigatorPanel.properties to set
the
     * label on the view goal button in each row of the table when the
user
     * doesn't have edit permission.
     */
    public static final String PROP_VIEW_ACTOR_BUTTON_LABEL =
"ViewActorButton.Label";

    /**
     */
    public ActorNavigatorPanel() {
        super(ActorNavigatorPanel.class.getName(), Project.class,
            ProjectManagementPanelNames.PROJECT_ACTORS_NAVIGATOR_PANEL_NAME);
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Actor Actor = (Actor) model.getBackingObject(row);
                    String editActorButtonLabel = null;
                    if (isReadOnlyMode()) {
                        editActorButtonLabel =
get ResourceBundleHelpergetLocale()).getString(
                            PROP_VIEW_ACTOR_BUTTON_LABEL, "View");
                    } else {
                        editActorButtonLabel =
get ResourceBundleHelpergetLocale()).getString(

```

```

    PROP_EDIT_ACTOR_BUTTON_LABEL, "Edit");
}

NavigationEvent openActorEditor = new OpenPanelEvent(this,
    PanelActionType.Editor, Actor, Actor.class, null,
    WorkflowDisposition.NewFlow);
NavigatorButton editActorButton = new
NavigatorButton(editActorButtonLabel,
    getEventDispatcher(), openActorEditor);
editActorButton.setStyleName(STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
editActorButton.setLayoutData(rld);
return editActorButton;
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Actor actor = (Actor) model.getBackingObject(row);
            return actor.getName();
        }
    });
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Description",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Actor actor = (Actor) model.getBackingObject(row);
            return actor.getText();
        }
    });
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
                return entity.getCreatedBy() == null ? "" :
entity.getCreatedBy()
                    .getUsername();
            }
        });
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    });
}

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "Actors: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROPERTY_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROPERTY_PANEL_TITLE,
    "Actors: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override

```

```

public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
        Alignment.DEFAULT));

    String closeButtonLabel =
    getResourceBundleHelpergetLocale()).getString(
        PROP_NEW_ACTOR_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
    getEventDispatcher(),
        closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);

    if (!isReadOnlyMode()) {
        String newActorButtonLabel =
    getResourceBundleHelpergetLocale()).getString(
        PROP_NEW_ACTOR_BUTTON_LABEL, "Add");
        NavigationEvent openActorEditor = new OpenPanelEvent(this,
    PanelActionType.Editor,
        getProjectOrDomain(), Actor.class, null,
    WorkflowDisposition.NewFlow);
        NavigatorButton newActorButton = new
    NavigatorButton(newActorButtonLabel,
        getEventDispatcher(), openActorEditor);
        newActorButton.setStyleName(STYLE_NAME_DEFAULT);
        buttonsWrapper.add(newActorButton);
    }

    add(buttonsWrapper);
}

```

```

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
        updateListener);
}
updateListener = new ActorUpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.class, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Actor.class,
                StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(pod.getActors());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

private static class ActorUpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ActorNavigatorPanel panel;

    private ActorUpdateListener(ActorNavigatorPanel panel) {
        this.panel = panel;
    }

    @Override

```

```

public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
        if (event.getObject() instanceof ProjectOrDomain) {
            updatedPod = (ProjectOrDomain) event.getObject();
        } else if (event.getObject() instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
                event.getObject();
            updatedPod = updatedEntity.getProjectOrDomain();
        } else if (event.getObject() instanceof Annotation) {
            if (!(event instanceof DeletedEntityEvent)) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                for (Annotatable annotatable :
                    updatedAnnotation.getAnnotatables()) {
                    if ((annotatable instanceof ProjectOrDomain)
                        && annotatable.equals(panel.getProjectOrDomain())) {
                        updatedPod = (ProjectOrDomain) annotatable;
                        break;
                    } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                        ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
                            annotatable;
                        if
                            (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                                updatedPod = entity.getProjectOrDomain();
                                break;
                            }
                        }
                    }
                }
            if (panel.getProjectOrDomain().equals(updatedPod)) {
                panel.setTargetObject(updatedPod);
            }
        }
    }
}

```

actorselectorpanel.java

```

/*
 * $Id: ActorSelectorPanel.java,v 1.4 2009/02/23 07:37:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

```

```

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class ActorSelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(ActorSelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the ActorNavigatorPanel.properties to set
     * the
     * * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CANCEL_BUTTON_LABEL";

    /**
     * @param projectRepository
     */
    public ActorSelectorPanel(ProjectRepository projectRepository) {
        super(ActorSelectorPanel.class.getName(), Project.class,
ProjectManagementPanelNames.PROJECT_STORY_SELECTOR_PANEL_NAME);
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();

        tableConfig.setRowLevelSelection(true);

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Actor actor = (Actor) model.getBackingObject(row);
        return actor.getName();
    }
}));
```

```

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Actor actor = (Actor) model.getBackingObject(row);
        return actor.getCreatedBy().getUsername();
    }
}));

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Actor actor = (Actor) model.getBackingObject(row);
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        return format.format(actor.getDateCreated());
    }
}));

        setTableConfig(tableConfig);
    }

    /**
     * Create a title for panel with dynamic information from the project
     * or
     * * domain, by default the title is "Select Actor"<br>
     * *
     * * @see Panel.PROP_PANEL_TITLE
     * * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
     */
    @Override
    public String getTitle() {
        return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Select Actor");
    }

    @Override
    public void dispose() {
        super.dispose();
        removeAll();
    }
}
```

```

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    updateListener = null;
}
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);
    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Actor.class,
StakeholderPermissionType.Edit);
        }
    }
}

}

return true;
}

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 *
 * @return an adapter to get the collection of items to select from
from the
 * target object.
 */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
    return new NavigatorTableModelAdapter() {
        private ProjectOrDomain targetObject;

        @Override
        public Collection<Object> getCollection() {
            return (Collection) targetObject.getActors();
        }

        @Override
        public void setTargetObject(Object targetObject) {
            this.targetObject = (ProjectOrDomain) targetObject;
        }
    };
}

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the Actor
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}
}

```

```

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ActorSelectorPanel panel;

    private UpdateListener(ActorSelectorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
                    event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
                        updatedAnnotation.getAnnotatables()) {
                        if (((annotatable instanceof ProjectOrDomain)
                            && annotatable.equals(panel.getProjectOrDomain())))
                            {
                                updatedPod = (ProjectOrDomain) annotatable;
                                break;
                            }
                        else if (((annotatable instanceof ProjectOrDomainEntity))
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
                            annotatable;
                        if
                            (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())))
                            {
                                updatedPod = entity.getProjectOrDomain();
                                break;
                            }
                        }
                    }
                }
            if (panel.getProjectOrDomain().equals(updatedPod)) {
                panel.setTargetObject(updatedPod);
            }
        }
    }
}

```

```

    }
}
```

actorstable.java

```

/*
 * $Id: ActorsTable.java,v 1.5 2009/01/08 06:48:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowHeaders;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.requel.ui.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component to add to Panels of actor container entity editors to
enable
 * editing of the actors of the entity.
 *
 * @author ron
 */
public class ActorsTable extends AbstractRequestNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(ActorsTable.class, new
ActorsTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * ActorsTable to define the label of the actors field. If the
property is
     * undefined the panel should use a sensible default such as
"Actors".
     */
    public static final String PROP_LABEL_ACTORS = "Actors.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     */
    * of the view button in the actor edit table column. If the property
is
    * undefined "View" is used.
    */
    public static final String PROP_VIEW_ACTOR_BUTTON_LABEL =
"ViewActor.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the remove button in the actor edit table column. If the
property is
     * undefined "Remove" is used.
     */
    public static final String PROP_REMOVE_ACTOR_BUTTON_LABEL =
"RemoveActor.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the edit button in the actor edit table column. If the property
is
     * undefined "Edit" is used.
     */
    public static final String PROP_EDIT_ACTOR_BUTTON_LABEL =
>EditActor.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the new actor button under the actors table. If the property is
     * undefined "New" is used.
     */
    public static final String PROP_NEW_ACTOR_BUTTON_LABEL =
>NewActor.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the find actor button under the actors table. If the property
is
     * undefined "Find" is used.
     */
    public static final String PROP_FIND_ACTOR_BUTTON_LABEL =
>FindActor.Label";

    private ActorContainer actorContainer;
}

```

```

private final NavigatorTable table;
private final NavigatorButton openActorEditorButton;
private final NavigatorButton openActorSelectorButton;
private final ProjectCommandFactory projectCommandFactory;
private final CommandHandler commandHandler;
private final AddActorToActorContainerController addActorController;
private final RemoveActorFromActorContainerController removeActorController;

/**
 * @param editMode
 * @param resourceBundleHelper
 * @param projectCommandFactory -
 *          passed to the add/remove actor to actor container
controller
 * @param commandHandler -
 *          passed to the add/remove actor to actor container
controller
 */
public ActorsTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper,
ProjectCommandFactory projectCommandFactory, CommandHandler
commandHandler) {
super(editMode, resourceBundleHelper);
this.projectCommandFactory = projectCommandFactory;
this.commandHandler = commandHandler;
ColumnLayoutData layoutData = new ColumnLayoutData();
layoutData.setAlignment(Alignment.ALIGN_CENTER);
table = new NavigatorTable(getTableConfig());
table.setLayoutData(layoutData);
add(table);

Row buttons = new Row();
buttons.setLayoutData(layoutData);

String buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_NEW_ACTOR_BUTTON_LABEL, "New");
openActorEditorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openActorEditorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
openActorEditorButton.setVisible(false);
buttons.add(openActorEditorButton);

buttonLabel =
getResourceBundleHelpergetLocale().getString(PROP_FIND_ACTOR_BUTTON_
LABEL,
"Find");

```

```

openActorSelectorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openActorSelectorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
openActorSelectorButton.setVisible(false);
buttons.add(openActorSelectorButton);

add(buttons);
}

protected ActorContainer getActorContainer() {
return actorContainer;
}

protected void setActorContainer(ActorContainer actorContainer) {
this.actorContainer = actorContainer;

if (actorContainer != null) {
table.setModel(new NavigatorTableModel((Collection)
actorContainer.getActors()));
if (!isReadOnlyMode()) {
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor,
getActorContainer(), Actor.class, null,
WorkflowDisposition.NewFlow);
openActorEditorButton.setEventToFire(openEditorEvent);
openActorEditorButton.setVisible(true);

ProjectOrDomain pod = null;
if (getActorContainer() instanceof ProjectOrDomain) {
pod = (ProjectOrDomain) getActorContainer();
} else if (getActorContainer() instanceof ProjectOrDomainEntity) {
pod = ((ProjectOrDomainEntity)
getActorContainer()).getProjectOrDomain();
}
if (pod != null) {
NavigationEvent openActorSelectorEvent = new OpenPanelEvent(this,
PanelActionType.Selector, pod, Project.class,
ProjectManagementPanelNames.PROJECT_ACTORS_SELECTOR_PANEL_NAME,
WorkflowDisposition.ContinueFlow);

openActorSelectorButton.setEventToFire(openActorSelectorEvent);
openActorSelectorButton.setVisible(true);
} else {
openActorSelectorButton.setVisible(false);
}

// use the the story table (this) as the destination because it

```

```

// is used as the source to the open panel events created above
if (addActorController != null) {
    getEventDispatcher().removeEventTypeActionListener(SelectEntityEvent.class,
        addActorController, this);
}
addActorController = new AddActorToActorContainerController(getEventDispatcher(),
    projectCommandFactory, commandHandler, actorContainer);
getEventDispatcher().addEventTypeActionListener(SelectEntityEvent.class,
    addActorController, this);

// use the the story table (this) as the destination because it
// is used as the source to the open panel events created above
if (removeActorController != null) {
    getEventDispatcher().removeEventTypeActionListener(
        RemoveActorFromActorContainerEvent.class,
        removeActorController, this);
}
removeActorController = new RemoveActorFromActorContainerController(
    getEventDispatcher(), projectCommandFactory, commandHandler,
    actorContainer);
getEventDispatcher().addEventTypeActionListener(
    RemoveActorFromActorContainerEvent.class, removeActorController,
    this);

}
} else {
    table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    openActorEditorButton.setVisible(false);
    openActorSelectorButton.setVisible(false);
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
                int row) {
                Row buttonsContainer = new Row();
                RowLayoutData buttonLayout = new RowLayoutData();
                buttonLayout.setAlignment(Alignment.ALIGN_CENTER);

```

```

                buttonLayout.setInsets(new Insets(5, 0));

                Actor actor = (Actor) model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_VIEW_ACTOR_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_EDIT_ACTOR_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(ActorsTable.this,
                    PanelEventType.Editor, actor, actor.getClass(), null,
                    WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                openEditorButton.setLayoutData(buttonLayout);
                buttonsContainer.add(openEditorButton);

                if (!isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_REMOVE_ACTOR_BUTTON_LABEL, "Remove");
                    NavigationEvent removeActorEvent = new RemoveActorFromActorContainerEvent(
                        ActorsTable.this, actor, actorContainer, ActorsTable.this);
                    NavigatorButton removeActorButton = new NavigatorButton(buttonLabel,
                        getEventDispatcher(), removeActorEvent);
                    openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                    openEditorButton.setLayoutData(buttonLayout);
                    buttonsContainer.add(removeActorButton);
                }
                return buttonsContainer;
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
            int row) {
            Actor actor = (Actor) model.getBackingObject(row);
            return actor.getName();
        }

```

```

        });

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Actor actor = (Actor) model.getBackingObject(row);
                return actor.getCreatedBy().getUsername();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Actor actor = (Actor) model.getBackingObject(row);
                DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                return formatter.format(actor.getDateCreated());
            }
        }));
}

return tableConfig;
}

private static class ActorsTableManipulator extends
AbstractComponentManipulator {

protected ActorsTableManipulator() {
    super();
}

@Override
public Object getModel(Component component) {
    return getValue(component, ActorContainer.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override

```

```

    public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getActorContainer());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setActorContainer((ActorContainer) value);
}

private ActorsTable getComponent(Component component) {
    return (ActorsTable) component;
}
}

```

addactorposition.java

```

/*
 * $Id: AddActorPosition.java,v 1.4 2009/01/19 09:32:25 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;
import
edu.harvard.fas.rregan.requel.annotation.impl.command.AnnotationComman
dFactoryImpl;
import
edu.harvard.fas.rregan.requel.project.impl.command.ResolveIssueWithAdd
ActorPositionCommandImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * This is a special position such that when it is chosen as the
resolution of

```

```

 * an issue via resolveIssue() the word is added as an actor to the
project.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.impl.AddActorPosition")
@XmlRootElement(name = "addActorPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "addActorPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class AddActorPosition extends PositionImpl {
    static final long serialVersionUID = 0L;

    // TODO: use spring to register the command for the positions, or
make the
    // position class implement the command
    static {
        AnnotationCommandFactoryImpl.addPositionResolverCommand(AddActorPosi
tion.class,
            ResolveIssueWithAddActorPositionCommandImpl.class);
    }

    /**
     * @param text
     * @param createdBy
     */
    public AddActorPosition(String text, User createdBy) {
        super(AddActorPosition.class.getName(), text, createdBy);
    }

    protected AddActorPosition() {
        super();
        // for hibernate
    }
}

```

addactortoactorcontainercommand.java

```

/*
 * $Id: AddActorToActorContainerCommand.java,v 1.1 2008/09/06 09:31:57
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;

/**
 * @author ron
 */
public interface AddActorToActorContainerCommand extends EditCommand {

    /**
     * Set the Story to add.
     *
     * @param actor
     */
    public void setActor(Actor actor);

    /**
     * @return the updated actor.
     */
    public Actor getActor();

    /**
     * Set the container this actor is being added to.
     *
     * @param actorContainer
     */
    public void setActorContainer(ActorContainer actorContainer);

    /**
     * @return the updated actor container.
     */
    public ActorContainer getActorContainer();
}

```

addactortoactorcontainercommandimpl.java

```

/*
 * $Id: AddActorToActorContainerCommandImpl.java,v 1.5 2009/03/30
11:54:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;

```

```

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.AddActorToActorContainerCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("addActorToActorContainerCommand")
@Scope("prototype")
public class AddActorToActorContainerCommandImpl extends AbstractEditProjectCommand implements AddActorToActorContainerCommand {

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     */
    @Autowired
    public AddActorToActorContainerCommandImpl(AssistantFacade
                                                assistantManager,
                                                UserRepository userRepository, ProjectRepository projectRepository,
                                                ProjectCommandFactory projectCommandFactory,
                                                AnnotationCommandFactory annotationCommandFactory,
                                                CommandHandler commandHandler) {
        super(assistantManager, userRepository, projectRepository,
              projectCommandFactory, annotationCommandFactory, commandHandler);
    }

    private Actor actor;
    private ActorContainer actorContainer;
}

```

```

@Override
public Actor getActor() {
    return actor;
}

@Override
public void setActor(Actor actor) {
    this.actor = actor;
}

@Override
public ActorContainer getActorContainer() {
    return actorContainer;
}

@Override
public void setActorContainer(ActorContainer actorContainer) {
    this.actorContainer = actorContainer;
}

@Override
public void execute() {
    Actor addedActor = getProjectRepository().get(getActor());
    ActorContainer addingContainer =
    getProjectRepository().get(getActorContainer());
    addedActor.getReferers().add(addingContainer);
    addingContainer.getActors().add(addedActor);

    // replaced the supplied objects with the updated objects for
    // retrieval.
    addedActor = getProjectRepository().merge(addedActor);
    addingContainer = getProjectRepository().merge(addingContainer);
    setActor(addedActor);
    setActorContainer(addingContainer);
}
}

```

addactortoactorcontainercontroller.java

```

/*
 * $Id: AddActorToActorContainerController.java,v 1.3 2008/12/13
 * 00:41:10 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

```

```

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import
edu.harvard.fas.rregan.requel.project.command.AddActorToActorContainer
Command;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * This controller is to be used in a actor container entity editor
where the
 * editor contains a ActorsTable and allows adding existing actors to
the
 * entity.
 *
 * @author ron
 */
public class AddActorToActorContainerController extends
AbstractRequelCommandController {
    static final long serialVersionUID = 0;

    private final ActorContainer actorContainer;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     * @param actorContainer
     */
    public AddActorToActorContainerController(EventDispatcher
eventDispatcher,
        ProjectCommandFactory commandFactory, CommandHandler
commandHandler,
        ActorContainer actorContainer) {
        super(eventDispatcher, commandFactory, commandHandler);
        this.actorContainer = actorContainer;
    }
}

```

```

    /**
     * FIXME: there is a flaw in that the select entity event doesn't
have an
     * expected destination so any select entity event with a actor will
trigger
     * this controller and may produce unintended results.
     */
@Override
public void actionPerformed(ActionEvent event) {
    if (event instanceof SelectEntityEvent) {
        SelectEntityEvent selectEntityEvent = (SelectEntityEvent) event;
        if (selectEntityEvent.getObject() instanceof Actor) {
            Actor actor = (Actor) selectEntityEvent.getObject();
            try {
                ProjectCommandFactory factory = getCommandFactory();
                AddActorToActorContainerCommand command = factory
                    .newAddActorToActorContainerCommand();
                command.setEditedBy(null); // TODO: need user?
                command.setActor(actor);
                command.setActorContainer(actorContainer);
                command = getCommandHandler().execute(command);
                fireEvent(new UpdateEntityEvent(this, null, command.getActor()));
                // TODO: this may not be needed because the update listeners for
things that
                // are actor containers will probably pickup the previous event
and get the
                // updated actor container from the actor.
                fireEvent(new UpdateEntityEvent(this, null,
command.getActorContainer()));
            } catch (Exception e) {
                // TODO: what can fail?
                log.error(e, e);
            }
        }
    }
}

```

addglossarytermposition.java

```

/*
 * $Id: AddGlossaryTermPosition.java,v 1.7 2009/01/19 09:32:25 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;
import edu.harvard.fas.rregan.requel.annotation.impl.command.AnnotationCommandFactoryImpl;
import edu.harvard.fas.rregan.requel.project.impl.command.ResolveIssueWithAddGlossaryTermPositionCommandImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * This is a special position such that when it is chosen as the
resolution of
 * an issue via resolveIssue() the word is added to the project
glossary.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition")
@XmlRootElement(name = "addGlossaryTermPosition", namespace = "http://
www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "addGlossaryTermPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class AddGlossaryTermPosition extends PositionImpl {
    static final long serialVersionUID = 0L;

    // TODO: use spring to register the command for the positions, or
make the
    // position class implement the command
    static {
        AnnotationCommandFactoryImpl.addPositionResolverCommand(AddGlossaryTermPosition.class,
            ResolveIssueWithAddGlossaryTermPositionCommandImpl.class);
    }

    /**
     * @param text
     * @param createdBy
     */
}

```

```

public AddGlossaryTermPosition(String text, User createdBy) {
    super(AddGlossaryTermPosition.class.getName(), text, createdBy);
}

protected AddGlossaryTermPosition() {
    super();
    // for hibernate
}
}

```

addgoaltogoalcontainercommand.java

```

/*
 * $Id: AddGoalToGoalContainerCommand.java,v 1.2 2008/05/01 08:10:16
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;

/**
 * @author ron
 */
public interface AddGoalToGoalContainerCommand extends EditCommand {

    /**
     * Set the goal to add.
     *
     * @param goal
     */
    public void setGoal(Goal goal);

    /**
     * @return the updated goal.
     */
    public Goal getGoal();

    /**
     * Set the container this goal is being added to.
     *
     * @param goalContainer
     */
}

```

```

public void setGoalContainer(GoalContainer goalContainer);

/**
 * @return the updated goal container.
 */
public GoalContainer getGoalContainer();
}

```

addgoaltogoalcontainercommandimpl.java

```

/*
 * $Id: AddGoalToGoalContainerCommandImpl.java,v 1.10 2009/03/30
11:54:28 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requiel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requiel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requiel.project.Goal;
import edu.harvard.fas.rregan.requiel.project.GoalContainer;
import edu.harvard.fas.rregan.requiel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requiel.project.command.AddGoalToGoalContainerCo
mmand;
import
edu.harvard.fas.rregan.requiel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requiel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requiel.user.UserRepository;

/**
 * @author ron
 */
@Controller("addGoalToGoalContainerCommand")
@Scope("prototype")
public class AddGoalToGoalContainerCommandImpl extends
AbstractEditProjectCommand implements
AddGoalToGoalContainerCommand {

```

```

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@.Autowired
public AddGoalToGoalContainerCommandImpl(AssistantFacade
assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory,
CommandHandler commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory, annotationCommandFactory, commandHandler);
}

private Goal goal;
private GoalContainer goalContainer;

@Override
public Goal getGoal() {
return goal;
}

@Override
public void setGoal(Goal goal) {
this.goal = goal;
}

@Override
public GoalContainer getGoalContainer() {
return goalContainer;
}

@Override
public void setGoalContainer(GoalContainer goalContainer) {
this.goalContainer = goalContainer;
}

@Override
public void execute() {
Goal addedGoal = getProjectRepository().get(getGoal());
GoalContainer addingContainer =
getProjectRepository().get(getGoalContainer());
addedGoal.getReferers().add(addingContainer);
addingContainer.getGoals().add(addedGoal);
}

```

```

    // replaced the supplied objects with the updated objects for
    retrieval.
    addedGoal = getProjectRepository().merge(addedGoal);
    addingContainer = getProjectRepository().merge(addingContainer);
    setGoal(addedGoal);
    setGoalContainer(addingContainer);
}
}

```

addgoaltogoalcontainercontroller.java

```

/*
 * $Id: AddGoalToGoalContainerController.java,v 1.3 2008/12/13
00:41:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import
edu.harvard.fas.rregan.requel.project.command.AddGoalToGoalContainerCo
mmand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * This controller is to be used in a goal container entity editor
where the
 * editor contains a GoalsTable and allows adding existing goals to
the entity.
 *
 * @author ron
 */
public class AddGoalToGoalContainerController extends
AbstractRequelCommandController {

```

```

static final long serialVersionUID = 0;

private final GoalContainer goalContainer;

/**
 * @param eventDispatcher
 * @param commandFactory
 * @param commandHandler
 * @param goalContainer
 */
public AddGoalToGoalContainerController(EventDispatcher
eventDispatcher,
    ProjectCommandFactory commandFactory, CommandHandler
commandHandler,
    GoalContainer goalContainer) {
    super(eventDispatcher, commandFactory, commandHandler);
    this.goalContainer = goalContainer;
}

/**
 * add the goal to the container via an AddGoalToGoalContainerCommand
*/
@Override
public void actionPerformed(ActionEvent event) {
    if (event instanceof SelectEntityEvent) {
        SelectEntityEvent selectEntityEvent = (SelectEntityEvent) event;
        // the destination of the event should be the object that created
this
        // controller. how can we check?
        if (selectEntityEvent.getObject() instanceof Goal) {
            Goal goal = (Goal) selectEntityEvent.getObject();
            try {
                ProjectCommandFactory factory = getCommandFactory();
                AddGoalToGoalContainerCommand command = factory
                    .newAddGoalToGoalContainerCommand();
                command.setEditedBy(null); // TODO: need user?
                command.setGoal(goal);
                command.setGoalContainer(goalContainer);
                command = getCommandHandler().execute(command);
                fireEvent(new UpdateEntityEvent(this, null, command.getGoal()));
                // TODO: this may not be needed because the update listeners for
things that
                // are goal containers will probably pickup the previous event
and get the
                // updated goal container from the goal.
                fireEvent(new UpdateEntityEvent(this, null,
                    command.getGoalContainer()));
            }
        }
    }
}

```

```
        } catch (Exception e) {
            // TODO: what can fail?
            log.error(e, e);
        }
    }
}
```

addstorytostorycontainercommand.java

```
/*
 * $Id: AddStoryToStoryContainerCommand.java,v 1.2 2008/09/06 09:31:57
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;

/**
 * @author ron
 */
public interface AddStoryToStoryContainerCommand extends EditCommand {

    /**
     * Set the Story to add.
     *
     * @param story
     */
    public void setStory(Story story);

    /**
     * @return the updated story.
     */
    public Story getStory();

    /**
     * Set the container this Story is being added to.
     *
     * @param storyContainer
     */
    public void setStoryContainer(StoryContainer storyContainer);
```

```
    /**
     * @return the updated Story container.
     */
    public StoryContainer getStoryContainer();
}
```

addstorytostorycontainercommandimpl.java

```
/*
 * $Id: AddStoryToStoryContainerCommandImpl.java,v 1.6 2009/03/30
11:54:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import
edu.harvard.fas.rregan.requel.project.command.AddStoryToStoryContainer
Command;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("addStoryToStoryContainerCommand")
@Scope("prototype")
public class AddStoryToStoryContainerCommandImpl extends
AbstractEditProjectCommand implements
AddStoryToStoryContainerCommand {

/**
```

```

* @param assistantManager
* @param userRepository
* @param projectRepository
*/
@.Autowired
public AddStoryToStoryContainerCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory,
    CommandHandler commandHandler) {
    super(assistantManager, userRepository, projectRepository,
    projectCommandFactory, annotationCommandFactory, commandHandler);
}

private Story story;
private StoryContainer storyContainer;

@Override
public Story getStory() {
    return story;
}

@Override
public void setStory(Story story) {
    this.story = story;
}

@Override
public StoryContainer getStoryContainer() {
    return storyContainer;
}

@Override
public void setStoryContainer(StoryContainer storyContainer) {
    this.storyContainer = storyContainer;
}

@Override
public void execute() {
    Story addedStory = getRepository().get(getStory());
    StoryContainer addingContainer =
getRepository().get(getStoryContainer());
    addedStory.getReferers().add(addingContainer);
    addingContainer.getStories().add(addedStory);
}

```

```

        // replaced the supplied objects with the updated objects for
        // retrieval.
        addedStory = getRepository().merge(addedStory);
        addingContainer = getRepository().merge(addingContainer);
        setStory(addedStory);
        setStoryContainer(addingContainer);
    }
}

```

addstorytostorycontainercontroller.java

```

/*
 * $Id: AddStoryToStoryContainerController.java,v 1.3 2008/12/13
00:41:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import
edu.harvard.fas.rregan.requel.project.command.AddStoryToStoryContainer
Command;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * This controller is to be used in a story container entity editor
where the
 * editor contains a StorysTable and allows adding existing storys to
the
 * entity.
 *
 * @author ron
 */

```

```

public class AddStoryToStoryContainerController extends
AbstractRequestCommandController {
    static final long serialVersionUID = 0;

    private final StoryContainer storyContainer;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     * @param storyContainer
     */
    public AddStoryToStoryContainerController(EventDispatcher
eventDispatcher,
        ProjectCommandFactory commandFactory, CommandHandler
commandHandler,
        StoryContainer storyContainer) {
        super(eventDispatcher, commandFactory, commandHandler);
        this.storyContainer = storyContainer;
    }

    /**
     * FIXME: there is a flaw in that the select entity event doesn't
have an
     * expected destination so any select entity event with a story will
trigger
     * this controller and may produce unintended results.
     */
    @Override
    public void actionPerformed(ActionEvent event) {
        if (event instanceof SelectEntityEvent) {
            SelectEntityEvent selectEntityEvent = (SelectEntityEvent) event;
            if (selectEntityEvent.getObject() instanceof Story) {
                Story story = (Story) selectEntityEvent.getObject();
                try {
                    ProjectCommandFactory factory = getCommandFactory();
                    AddStoryToStoryContainerCommand command = factory
                        .newAddStoryToStoryContainerCommand();
                    command.setEditedBy(null); // TODO: need user?
                    command.setStory(story);
                    command.setStoryContainer(storyContainer);
                    command = getCommandHandler().execute(command);
                    fireEvent(new UpdateEntityEvent(this, null, command.getStory()));
                    // TODO: this may not be needed because the update listeners for
things that
                    // are story containers will probably pickup the previous event
and get the
                }
            }
        }
    }
}

```

```
// updated story container from the story.
fireEvent(new UpdateEntityEvent(this, null,
command.getStoryContainer())));
} catch (Exception e) {
// TODO: what can fail?
log.error(e, e);
}
}
}
}
```

addwordtodictionaryposition.java

```
/*
 * $Id: AddWordToDictionaryPosition.java,v 1.6 2009/01/08 10:11:23
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * This is a special position such that when it is chosen as the
resolution of
 * an issue via resolveIssue() the word is added to the dictionary.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.impl.AddWordToDictionaryPosi
tion")
@XmlRootElement(name = "addWordToDictionaryPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "addWordToDictionaryPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class AddWordToDictionaryPosition extends PositionImpl {
    static final long serialVersionUID = 0L;
```

```

/**
 * @param text
 * @param createdBy
 */
public AddWordToDictionaryPosition(String text, User createdBy) {
    super(AddWordToDictionaryPosition.class.getName(), text, createdBy);
}

protected AddWordToDictionaryPosition() {
    super();
    // for hibernate
}
}

```

adjectivematchingrule.java

```

/*
 * $Id: AdjectiveMatchingRule.java,v 1.3 2009/02/11 09:02:54 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ListIterator;
import org.apache.log4j.Logger;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * @author ron
 */
public class AdjectiveMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
Logger.getLogger(AdjectiveMatchingRule.class);

    /**
     *
     */
@Override
public void match(DictionaryRepository dictionaryRepository, NLPText
verb,

```

```

ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException {
NLPText word = textIterator.next();
// TODO: what about an adjective phrase?
if (!word.is(PartOfSpeech.ADJECTIVE)) {
    throw SemanticRoleLabelerException.matchFailed(this, word);
}
log.debug("matched: " + word.getText());
}

@Override
public String toString() {
    return getClass().getSimpleName();
}
}

```

adminuserinitializer.java

```

/*
 * $Id: AdminUserInitializer.java,v 1.9 2009/03/29 11:59:30 rregan Exp
$ $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**
 * Create the admin user if it doesn't exist.
 *
 * @author ron
 */
@Component("adminUserInitializer")
@Scope("prototype")
public class AdminUserInitializer extends AbstractSystemInitializer {

```

```
private final UserRepository userRepository;
private final EditUserCommand command;
private final CommandHandler commandHandler;

/**
 * @param userRepository
 * @param commandHandler
 * @param command
 */
@Autowired
public AdminUserInitializer(UserRepository userRepository,
CommandHandler commandHandler,
EditUserCommand command) {
super(100);
this.userRepository = userRepository;
this.commandHandler = commandHandler;
this.command = command;
}

@Override
public void initialize() {
try {
userRepository.findUserByUsername("admin");
} catch (NoSuchUserException e) {
try {
command.setUsername("admin");
command.setPassword("admin");
command.setRepassword("admin");
command.setName("System Administrator");
command.setEmailAddress("rreganjr@acm.org");
command.setOrganizationName("Requel");
command.addUserRoleName(SystemAdminUserRole.getRoleName(SystemAdminUserRole.class));
commandHandler.execute(command);
} catch (Exception e2) {
log.error("failed to initialize the admin user: " + e2, e2);
}
}
}
}
```

adverbmatchingrule.java

```
/*
 * $Id: AdverbMatchingRule.java,v 1.3 2009/02/11 09:02:54 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
```

```
/*
package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * @author ron
 */
public class AdverbMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
Logger.getLogger(AdverbMatchingRule.class);

    /**
     *
     */
@Override
    public void match(DictionaryRepository dictionaryRepository, NLPText
verb,
        ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException {
    NLPText word = textIterator.next();
    if (!word.is(PartOfSpeech.ADVERB)) {
        throw SemanticRoleLabelerException.matchFailed(this, word);
    }
    log.debug("matched: " + word.getText());
}

@Override
    public String toString() {
        return getClass().getSimpleName();
    }
}
```

analysisinvokingcommandhandler.java

```
/*
 * $Id: AnalysisInvokingCommandHandler.java,v 1.1 2009/02/13 12:08:04
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
```

```

/*
package edu.harvard.fas.rregan.requel.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.command.CommandHandler;

/**
 * A CommandHandler decorator that executes a command, and then if it
implements
 * the EditAnalyzableCommand interface and didn't throw an exception
it calls
 * the invokeAnalysis() method on the command.
 *
 * @author ron
 */
public class AnalysisInvokingCommandHandler implements CommandHandler
{

    private final CommandHandler commandHandler;

    /**
     * @param exceptionMapper
     * @param commandHandler
     */
    public AnalysisInvokingCommandHandler(CommandHandler commandHandler)
    {
        this.commandHandler = commandHandler;
    }

    public <T extends Command> T execute(T command) throws Exception {
        T executedCommand = commandHandler.execute(command);
        if (executedCommand instanceof AnalyzableEditCommand) {
            ((AnalyzableEditCommand) executedCommand).invokeAnalysis();
        }
        return executedCommand;
    }
}

```

analyzableeditcommand.java

```

/*
 * $Id: AnalyzableEditCommand.java,v 1.1 2009/02/13 12:08:03 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.requel.command;

/**
 * An EditCommand with a secondary process for analyzing the results
of the edit
 * command after the execute() method completes successfully.
 *
 * @author ron
 */
public interface AnalyzableEditCommand extends EditCommand {

    /**
     * Turn on or off analysis. By default it is on.
     *
     * @param analysisEnabled
     */
    public void setAnalysisEnabled(boolean analysisEnabled);

    /**
     * After a command executes successfully, the
     * AssistantInvokingCommandHandler calls this method. It is up to the
     * command to check if analysis should happen and to apply the
appropriate
     * analysis.
     *
     * @see {@link AnalysisInvokingCommandHandler}
     */
    public void invokeAnalysis();
}

```

annotatable.java

```

/*
 * $Id: Annotatable.java,v 1.4 2008/08/08 10:02:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

import java.util.Set;

/**
 * Represents something that can have an Annotation such as a note, or
issue.
 *
 * @author ron
 */

```

```

public interface Annotatable {
    /**
     * @return The annotations attached to this object.
     */
    public Set<Annotation> getAnnotations();
}

```

annotation.java

```

/*
 * $Id: Annotation.java,v 1.10 2009/02/16 10:10:09 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * An abstraction of something that can be added to an Annotatable
 * object such
 * as an Issue or Note. An annotation may itself be annotated.
 *
 * @author ron
 */
public interface Annotation extends Comparable<Annotation>,
CreatedEntity, Describable {

    /**
     * @return an object used as a context for a group of annotations.
     */
    public Object getGroupingObject();

    /**
     * @return the simple name of the type of annotation
     */
    public String getTypeName();

    /**
     * @return The annotatable objects that have this annotation
     * attached.
     */
    public Set<Annotatable> getAnnotatables();
}

```

```

    /**
     * @return The text of the annotation.
     */
    public String getText();

    /**
     * @return return true if this issue must be resolved.
     */
    public boolean isMustBeResolved();

    /**
     * @return return true if this issue has been resolved.
     */
    public boolean isResolved();

    /**
     * @return a message appropriate for the state of the annotation.
     */
    public String getStatusMessage();
}

```

annotationcommandfactory.java

```

/*
 * $Id: AnnotationCommandFactory.java,v 1.15 2009/02/23 08:49:49
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.command.CommandFactory;
import edu.harvard.fas.rregan.requel.annotation.Position;

/**
 * @author ron
 */
public interface AnnotationCommandFactory extends CommandFactory {

    /**
     * @return a new EditNoteCommand for creating or editing a note
     * annotation.
     */
    public EditNoteCommand newEditNoteCommand();

    /**
     * @return a new EditIssueCommand for creating or editing an issue
     */

```

```

        *      annotation.
    */
public EditIssueCommand newEditIssueCommand();

/**
 * @return a new EditLexicalIssueCommand for creating or editing an
issue
 *      annotation related to word issues, like spelling, glossary
 *      entries etc.
 */
public EditLexicalIssueCommand newEditLexicalIssueCommand();

/**
 * @param position -
 *      the position used to determine the type of
ResolveIssueCommand
 *      to return.
 * @return a new ResolveIssueCommand for resolving an issue
annotation.
 */
public ResolveIssueCommand newResolveIssueCommand(Position position);

/**
 * @return a new EditPositionCommand for creating or editing a
position of
 *      an issue.
 */
public EditPositionCommand newEditPositionCommand();

/**
 * @return a new EditChangeSpellingPositionCommand for creating or
editing a
 *      position that indicates a possible spelling correction.
 */
public EditChangeSpellingPositionCommand
newEditChangeSpellingPositionCommand();

/**
 * @return a new EditAddWordToDictionaryPositionCommand for adding
 */
public EditAddWordToDictionaryPositionCommand
newEditAddWordToDictionaryPositionCommand();

/**
 * @return a new EditArgumentCommand for creating or editing an
argument for
 *      or against a position of an issue.

```

```

        */
public EditArgumentCommand newEditArgumentCommand();

/**
 * @return a new RemoveAnnotationFromAnnotatableCommand for removing
an
 *      annotation from an annotatable object and cleaning up all
 *      references.
 */
public RemoveAnnotationFromAnnotatableCommand
newRemoveAnnotationFromAnnotatableCommand();

/**
 * @return a new DeleteNoteCommand for deleting a note from the
system and
 *      cleaning up all references from annotatable objects.
 */
public DeleteNoteCommand newDeleteNoteCommand();

/**
 * @return a new DeleteIssueCommand for deleting an issue and its
components
 *      from the system and cleaning up all references from
annotatable
 *      objects.
 */
public DeleteIssueCommand newDeleteIssueCommand();

/**
 * @return a new DeletePositionCommand for deleting a position from
the
 *      system and cleaning up all references from issues.
 */
public DeletePositionCommand newDeletePositionCommand();

/**
 * @return a new DeleteArgumentCommand for deleting an argument from
the
 *      system and cleaning up all references from positions.
 */
public DeleteArgumentCommand newDeleteArgumentCommand();
}

```

annotationcommandfactoryimpl.java

```
/*
 * $Id: AnnotationCommandFactoryImpl.java,v 1.17 2009/02/13 12:07:56
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.AbstractCommandFactory;
import edu.harvard.fas.rregan.command.CommandFactoryStrategy;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.DeleteArgumentCommand;
import edu.harvard.fas.rregan.requel.annotation.command.DeleteIssueCommand;
import edu.harvard.fas.rregan.requel.annotation.command.DeleteNoteCommand;
import edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditAddWordToDictionaryPositionCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditChangeSpellingPositionCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditIssueCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditLexicalIssueCommand;
```

```
import
edu.harvard.fas.rregan.requel.annotation.command.EditNoteCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.ResolveIssueCommand;
import
edu.harvard.fas.rregan.requel.annotation.impl.AddWordToDictionaryPosit
ion;
import
edu.harvard.fas.rregan.requel.annotation.impl.ChangeSpellingPosition;
import edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;

/**
 * @author ron
 */
@Controller("annotationCommandFactory")
@Scope("singleton")
public class AnnotationCommandFactoryImpl extends
AbstractCommandFactory implements
AnnotationCommandFactory {

    // TODO: move the configuration to spring
    // Map of position types to resolver command types for those
    // positions.
    private static final Map<Class<? extends Position>, Class<? extends
ResolveIssueCommand>> positionToResolverCommand = new HashMap<Class<?
extends Position>, Class<? extends ResolveIssueCommand>>();
    static {
        positionToResolverCommand.put(PositionImpl.class,
ResolveIssueCommandImpl.class);
        positionToResolverCommand.put(ChangeSpellingPosition.class,
ResolveIssueWithChangeSpellingPositionCommandImpl.class);
        positionToResolverCommand.put(AddWordToDictionaryPosition.class,
ResolveIssueWithAddWordToDictionaryPositionCommandImpl.class);
    }

    /**
     * Register a command to use to resolve a specific type of position.
     *
     * @param positionType
     * @param commandType
     */
}
```

```

public static void addPositionResolverCommand(Class<? extends Position> positionType,
    Class<? extends ResolveIssueCommand> commandType) {
    positionToResolverCommand.put(positionType, commandType);
}

/**
 * @param creationStrategy -
 *          the strategy to use for creating new Command instances
 */
@.Autowired
public AnnotationCommandFactoryImpl(CommandFactoryStrategy
creationStrategy) {
    super(creationStrategy);
}

@Override
public EditNoteCommand newEditNoteCommand() {
    return (EditNoteCommand)
getCreationStrategy().newInstance(EditNoteCommandImpl.class);
}

@Override
public EditIssueCommand newEditIssueCommand() {
    return (EditIssueCommand)
getCreationStrategy().newInstance(EditIssueCommandImpl.class);
}

@Override
public EditLexicalIssueCommand newEditLexicalIssueCommand() {
    return (EditLexicalIssueCommand) getCreationStrategy().newInstance(
        EditLexicalIssueCommandImpl.class);
}

@Override
public ResolveIssueCommand newResolveIssueCommand(Position position)
{
    Class<?> positionType = position.getClass();
    while (positionType != null) {
        Class<? extends ResolveIssueCommand> resolverClass =
positionToResolverCommand
            .get(positionType);
        if (resolverClass == null) {
            for (Class<?> intf : positionType.getInterfaces()) {
                resolverClass = positionToResolverCommand.get(intf);
                if (resolverClass != null) {
                    break;
                }
            }
        }
        if (resolverClass != null) {
            positionType = positionType.getSuperclass();
        }
    }
    if (resolverClass != null) {
        return (ResolveIssueCommand)
getCreationStrategy().newInstance(resolverClass);
    }
    throw new RuntimeException("unexpected position type: " +
position.getClass().getName()
        + " no resolver command");
}

@Override
public EditPositionCommand newEditPositionCommand() {
    return (EditPositionCommand) getCreationStrategy().newInstance(
        EditPositionCommandImpl.class);
}

@Override
public EditChangeSpellingPositionCommand
newEditChangeSpellingPositionCommand() {
    return (EditChangeSpellingPositionCommand)
getCreationStrategy().newInstance(
        EditChangeSpellingPositionCommandImpl.class);
}

@Override
public EditAddWordToDictionaryPositionCommand
newEditAddWordToDictionaryPositionCommand() {
    return (EditAddWordToDictionaryPositionCommand)
getCreationStrategy().newInstance(
        EditAddWordToDictionaryPositionCommandImpl.class);
}

@Override
public EditArgumentCommand newEditArgumentCommand() {
    return (EditArgumentCommand) getCreationStrategy().newInstance(
        EditArgumentCommandImpl.class);
}

@Override
public RemoveAnnotationFromAnnotatableCommand
newRemoveAnnotationFromAnnotatableCommand() {
    return (RemoveAnnotationFromAnnotatableCommand)
getCreationStrategy().newInstance(

```

```

        RemoveAnnotationFromAnnotatableCommandImpl.class);
    }

@Override
public DeleteArgumentCommand newDeleteArgumentCommand() {
    return (DeleteArgumentCommand) getCreationStrategy().newInstance(
        DeleteArgumentCommandImpl.class);
}

@Override
public DeleteIssueCommand newDeleteIssueCommand() {
    return (DeleteIssueCommand)
getCreationStrategy().newInstance(DeleteIssueCommandImpl.class);
}

@Override
public DeleteNoteCommand newDeleteNoteCommand() {
    return (DeleteNoteCommand)
getCreationStrategy().newInstance(DeleteNoteCommandImpl.class);
}

@Override
public DeletePositionCommand newDeletePositionCommand() {
    return (DeletePositionCommand) getCreationStrategy().newInstance(
        DeletePositionCommandImpl.class);
}
}

```

annotationexistsexception.java

```

/*
 * $Id: AnnotationExistsException.java,v 1.2 2008/12/13 00:41:47
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;

/**
 * @author ron
 */
public class AnnotationExistsException extends EntityException {
    static final long serialVersionUID = 0;

```

```

protected static String MSG_FOR_MESSAGE = "An annotation already
exists with message '%s'";
protected static String MSG_FOR_LEXICAL_ISSUE = "A lexical issue
already exists for word '%s'";

/**
 * @param message -
 *          the message used to search for an issue that already
exists
 * @return
 */
public static AnnotationExistsException forMessage(String message) {
    return new AnnotationExistsException(Annotation.class, null,
"message", message,
    EntityExceptionActionType.Reading, MSG_FOR_MESSAGE, message);
}

/**
 * @param word -
 *          the word used to search for an issue that already
exists
 * @param annotatableEntityPropertyName -
 *          the property of the entity in question.
 * @return
 */
public static AnnotationExistsException forWord(String word,
String annotatableEntityPropertyName) {
    return new AnnotationExistsException(Issue.class, null, "word",
word,
    EntityExceptionActionType.Reading, MSG_FOR_LEXICAL_ISSUE, word,
    annotatableEntityPropertyName);
}

/**
 * @param format
 * @param args
 */
protected AnnotationExistsException(Class<?> entityType, Object
entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

```

```

/**
 * @param cause
 * @param format
 * @param args
 */
protected AnnotationExistsException(Throwable cause, Class<?>
entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

annotationreferertable.java

```

/*
 * $Id: AnnotationRefererTable.java,v 1.1 2009/02/15 09:31:34 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A table of domain entities that refer to an annotation (note or
annotation.)
*
* @author ron
*/
public class AnnotationRefererTable extends
AbstractRequelNavigatorTable {
static final long serialVersionUID = 0L;

static {
ComponentManipulators.setManipulator(AnnotationRefererTable.class,
new AnnotatablesTableManipulator());
}

/**
 * The name to use in the properties file of the panel that includes
the
* AnnotationContainersTable to define the label of the annotation
* containers field. If the property is undefined the panel should
use a
* sensible default such as "Referers".
*/

```

```

public static final String PROP_ANNOTATABLES_LABEL =
"AnnotationReferers.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the view button in the glossary term containers edit table
column. If
 * the property is undefined "View" is used.
 */
public static final String PROP_VIEW_CONTAINER_BUTTON_LABEL =
"ViewAnnotationContainer.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the edit button in the annotation container edit table column.
If the
 * property is undefined "Edit" is used.
 */
public static final String PROP_EDIT_CONTAINER_BUTTON_LABEL =
>EditAnnotationContainer.Label";

private Annotation annotation;
private final NavigatorTable table;

/**
 * @param editMode
 * @param resourceBundleHelper
 */
public AnnotationRefererTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
    super(editMode, resourceBundleHelper);
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    table = new NavigatorTable(getTableConfig());
    table.setLayoutData(layoutData);
    add(table);
}

protected Annotation getAnnotation() {
    return annotation;
}

protected void setAnnotation(Annotation annotation) {
    this.annotation = annotation;
    if (annotation != null) {

```

```

        table.setModel(new NavigatorTableModel((Collection)
annotation.getAnnotatables()));
    } else {
        table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    }
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Object annotationReferer = model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_VIEW_CONTAINER_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_EDIT_CONTAINER_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                    PanelActionType.Editor, annotationReferer, annotationReferer
                    .getClass(), null, WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
                NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                openEditorButton.setLayoutData(rld);
                return openEditorButton;
            }
        }));
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Description",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Object annotationReferer = model.getBackingObject(row);
            if (annotationReferer instanceof Describable) {
                return ((Describable) annotationReferer).getDescription();
            }
        }
    }));

```

```

        }
        return annotationReferer.toString();
    }
}));


tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Object annotationReferer = model.getBackingObject(row);
            if (annotationReferer instanceof CreatedEntity) {
                return ((CreatedEntity)
annotationReferer).getCreatedBy().getUsername();
            }
            return "";
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Object object = model.getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            if (object instanceof CreatedEntity) {
                return formatter.format(((CreatedEntity)
object).getDateCreated());
            }
            return "";
        }
    }));
}

return tableConfig;
}

private static class AnnotatablesTableManipulator extends
AbstractComponentManipulator {

protected AnnotatablesTableManipulator() {
    super();
}

@Override

```

```
public Object getModel(Component component) {
    return getValue(component, Annotation.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getAnnotation());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setAnnotation((Annotation) value);
}

private AnnotationRefererTable getComponent(Component component) {
    return (AnnotationRefererTable) component;
}
}
```

annotationrepository.java

```
/*
 * $Id: AnnotationRepository.java,v 1.11 2009/01/09 09:56:16 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requiel.annotation;

import edu.harvard.fas.rregan.repository.Repository;
import
edu.harvard.fas.rregan.requiel.annotation.impl.AddWordToDictionaryPosit
ion;
```

```

import
edu.harvard.fas.rregan.requel.annotation.impl.ChangeSpellingPosition;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;

/**
 * @author ron
 */
public interface AnnotationRepository extends Repository {

    /**
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     *          annotations.
     * @param text -
     *          the text of the position to match.
     * @return
     * @throws NoSuchPositionException
     */
    public Position findPosition(Object groupingObject, String text)
        throws NoSuchPositionException;

    /**
     * Find an existing position adding a word to the dictionary.
     *
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     *          annotations.
     * @param word -
     *          the word to be added to the dictionary
     * @return the position
     * @throws NoSuchPositionException -
     *          if an add word to dictionary position doesn't exist
     *          for the
     *          supplied issue.
     */
    public AddWordToDictionaryPosition
        findAddWordToDictionaryPosition(Object groupingObject,
            String word) throws NoSuchPositionException;

    /**
     * Find an existing position on a specific issue for changing the
     * spelling
     *          * of a word.
     *
     * @param issue
     * @param proposedWord
     * @return
     */
}

```

```

        * @throws NoSuchPositionException
        */
    public ChangeSpellingPosition findChangeSpellingPosition(LexicalIssue
        issue, String proposedWord)
        throws NoSuchPositionException;

    /**
     * Find a lexical issue where the word (text) matches the supplied
     * word.
     *
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     *          annotations.
     * @param annotatable -
     *          the annotated entity that the issue is attached to.
     * @param word -
     *          the word in question.
     * @return
     * @throws NoSuchAnnotationException
     */
    public LexicalIssue findLexicalIssue(Object groupingObject,
        Annotatable annotatable, String word)
        throws NoSuchAnnotationException;

    /**
     * Find a lexical issue where the word (text) matches the supplied
     * word and
     *          * the property name of the issue matches the
     *          annotatableEntityPropertyName.
     *
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     *          annotations.
     * @param annotatable -
     *          the annotated entity that the issue is attached to.
     * @param word -
     *          the word in question.
     * @param annotatableEntityPropertyName -
     *          the property of the annotatable entity the issue is
     *          concerning.
     * @return
     * @throws NoSuchAnnotationException
     */
    public LexicalIssue findLexicalIssue(Object groupingObject,
        Annotatable annotatable,
        String word, String annotatableEntityPropertyName) throws
        NoSuchAnnotationException;
}

```

```

/**
 * Find an issue with the supplied message.
 *
 * @param groupingObject -
 *      An object used as the "owner" of a group of
annotations.
 * @param annotatable -
 *      the annotated entity that the issue is attached to.
 * @param message
 * @return
 */
public Issue findIssue(Object groupingObject, Annotatable
annotatable, String message);

/**
 * Find an note with the supplied annotatable.
 *
 * @param groupingObject -
 *      An object used as the "owner" of a group of
annotations.
 * @param annotatable -
 *      the annotated entity that the note is attached to.
 * @param message
 * @return
 */
public Note findNote(Object groupingObject, Annotatable annotatable,
String message);
}

```

annotationstable.java

```

/*
 * $Id: AnnotationsTable.java,v 1.12 2009/02/20 07:27:40 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Row;

```

```

import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowStyleData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;
*/

```

```

 * A component to add to Panels of annotatable entity editors to
enable editing
 * of the annotations of the entity.
 *
 * @author ron
 */
public class AnnotationsTable extends AbstractRequestNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(AnnotationsTable.class,
            new AnnotationsTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * AnnotationsTable to define the label of the annotations field. If the
     * property is undefined the panel should use a sensible default such
as
     * "Annotations".
     */
    public static final String PROP_LABEL_ANNOTATIONS =
"Annotations.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the view button in the annotation edit table column. If the
property
     * is undefined "View" is used.
     */
    public static final String PROP_VIEW_ANNOTATION_BUTTON_LABEL =
"ViewAnnotation.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the edit button in the annotation edit table column. If the
property
     * is undefined "Edit" is used.
     */
    public static final String PROP_EDIT_ANNOTATION_BUTTON_LABEL =
>EditAnnotation.Label";

}

    * The name to use in the containing panels properties file to set
the label
     * of the add issue button under the annotations table. If the
property is
     * undefined "Add Issue" is used.
     */
    public static final String PROP_ADD_ISSUE_BUTTON_LABEL =
"AddIssue.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the add note button under the annotations table. If the
property is
     * undefined "Add Note" is used.
     */
    public static final String PROP_ADD_NOTE_BUTTON_LABEL =
"AddNote.Label";

    private Annotatable annotatable;
    private final NavigatorTable table;
    private final NavigatorButton openIssueEditorButton;
    private final NavigatorButton openNoteEditorButton;

    /**
     * @param editMode
     * @param resourceBundleHelper
     */
    public AnnotationsTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
        ColumnLayoutData layoutData = new ColumnLayoutData();
        layoutData.setAlignment(Alignment.ALIGN_CENTER);
        table = new NavigatorTablegetTableConfig());
        table.setLayoutData(layoutData);
        add(table);

        Row buttons = new Row();
        buttons.setLayoutData(layoutData);
        String buttonLabel = getResourceBundleHelpergetLocale()).getString(
            PROP_ADD_ISSUE_BUTTON_LABEL, "Add Issue");
        openIssueEditorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
        openIssueEditorButton.setStyleName(Panels.STYLE_NAME_DEFAULT);
        buttons.add(openIssueEditorButton);
    }
}

```

```

buttonLabel =
getResourceBundleHelper(getLocale()).getString(PROP_ADD_NOTE_BUTTON_LA
BEL,
        "Add Note");
openNoteEditorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openNoteEditorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
buttons.add(openNoteEditorButton);
add(buttons);
}

protected Annotatable getAnnotatable() {
    return annotatable;
}

protected void setAnnotatable(Annotatable annotatable) {
this.annotatable = annotatable;
if (annotatable != null) {
    table.setModel(new NavigatorTableModel((Collection)
annotatable.getAnnotations()));
    if (!isReadOnlyMode()) {
        NavigationEvent openIssueEditorEvent = new OpenPanelEvent(this,
            PanelActionType.Editor, getAnnotatable(), Issue.class, null,
            WorkflowDisposition.NewFlow);
        openIssueEditorButton.setEventToFire(openIssueEditorEvent);
        openIssueEditorButton.setEnabled(true);
        openIssueEditorButton.setVisible(true);

        NavigationEvent openNoteEditorEvent = new OpenPanelEvent(this,
            PanelActionType.Editor, getAnnotatable(), Note.class, null,
            WorkflowDisposition.NewFlow);
        openNoteEditorButton.setEventToFire(openNoteEditorEvent);
        openNoteEditorButton.setEnabled(true);
        openNoteEditorButton.setVisible(true);
    } else {
        openIssueEditorButton.setVisible(false);
        openNoteEditorButton.setVisible(false);
        openIssueEditorButton.setEnabled(false);
        openNoteEditorButton.setEnabled(false);
    }
} else {
    table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    openIssueEditorButton.setVisible(false);
    openNoteEditorButton.setVisible(false);
    openIssueEditorButton.setEnabled(false);
    openNoteEditorButton.setEnabled(false);
}
}

@Override
public boolean isReadOnlyMode() {
// TODO: this causes a project package dependency
// project entities are subject to stakeholder permissions, so if
this
// note is concerned with a project or project entity check the
// stakeholder permissions for editing annotations.
boolean projectEntity = false;
User user = (User) getApp().getUser();
Annotatable annotatable = getAnnotatable();
if (annotatable != null) {
    Stakeholder stakeholder = null;
    if (annotatable instanceof Project) {
        projectEntity = true;
        Project project = (Project) annotatable;
        stakeholder = project.getUserStakeholder(user);
    } else if (annotatable instanceof ProjectOrDomainEntity) {
        ProjectOrDomainEntity podEntity = (ProjectOrDomainEntity)
annotatable;
        if (podEntity.getProjectOrDomain() instanceof Project) {
            projectEntity = true;
            Project project = (Project) podEntity.getProjectOrDomain();
            stakeholder = project.getUserStakeholder(user);
        }
    }
    if (stakeholder != null) {
        return !stakeholder.hasPermission(Annotation.class,
StakeholderPermissionType.Edit);
    }
}
return projectEntity;
}

private NavigatorTableConfig getTableConfig() {
NavigatorTableConfig tableConfig = new NavigatorTableConfig();

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        String buttonLabel = null;
        if (isReadOnlyMode()) {

```

```

buttonLabel = getResourceBundleHelper(getLocale()).getString(
    PROP_VIEW_ANNOTATION_BUTTON_LABEL, "View");
} else {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_EDIT_ANNOTATION_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor, annotation, annotation.getClass(),
null,
    WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
}
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Type",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        return annotation.getTypeName();
    }
});
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Status",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        // TODO: replace with resource strings
        return annotation.getStatusMessage();
    }
});
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Must Be
Resolved?",
new NavigatorTableCellValueFactory() {

```

```

    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        // TODO: replace with resource strings
        return (annotation.isMustBeResolved() ? "Yes" : "No");
    }
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Text",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        return annotation.getText();
    }
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        return annotation.getCreatedBy().getUsername();
    }
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Annotation annotation = (Annotation)
model.getBackingObject(row);
        DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
        return formatter.format(annotation.getDateCreated());
    }
});

return tableConfig;

```

```

}

private static class AnnotationsTableManipulator extends
AbstractComponentManipulator {

protected AnnotationsTableManipulator() {
super();
}

@Override
public Object getModel(Component component) {
return getValue(component, Annotatable.class);
}

@Override
public void setModel(Component component, Object valueModel) {
setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
// nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
return type.cast(getComponent(component).getAnnotatable());
}

@Override
public void setValue(Component component, Object value) {
getComponent(component).setAnnotatable((Annotatable) value);
}

private AnnotationsTable getComponent(Component component) {
return (AnnotationsTable) component;
}
}
}

```

appawarecontroller.java

```

/*
 * $Id: AppAwareController.java,v 1.1 2008/02/20 11:36:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.uiframework.controller;

import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;

/**
 * @author ron
 */
public interface AppAwareController extends Controller {

/**
 * Set the app this command is aware of
 *
 * @param app
 */
public void setApp(UIFrameworkApp app);
}
```

applicationcontextcommandfactorystrategy.java

```

/*
 * $Id: ApplicationContextCommandFactoryStrategy.java,v 1.1 2008/12/13
00:40:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.command;

import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

/**
 * @author ron
 */
@Component("commandFactoryStrategy")
@Scope("prototype")
public class ApplicationContextCommandFactoryStrategy implements
CommandFactoryStrategy,
ApplicationContextAware {

private ApplicationContext applicationContext;
```

```

protected ApplicationContextCommandFactoryStrategy() {
    super();
}

@Override
public void setApplicationContext(ApplicationContext
    applicationContext) throws BeansException {
    this.applicationContext = applicationContext;
}

/***
 * @return
 */
protected ApplicationContext getApplicationContext() {
    return applicationContext;
}

/***
 * @see
edu.harvard.fas.rregan.command.CommandFactoryStrategy#newInstance(java
.lang.Class)
 */
@Override
public Command newInstance(Class<? extends Command> commandType) {
    return (Command)
getApplicationContext().getAutowireCapableBeanFactory().createBean(
    commandType);
}

}

```

applicationexception.java

```

/*
 * $Id: ApplicationException.java,v 1.7 2009/01/03 10:24:38 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan;

import java.util.Formatter;
import org.apache.log4j.Logger;
/***

```

```

    * @author ron
    */
public class ApplicationException extends RuntimeException {
    private static final Logger log =
Logger.getLogger(ApplicationException.class);
    static final long serialVersionUID = 0;

    protected static String MSG_NOT_IMPLEMENTED = "Not implemented.";
    protected static String MSG_FAILED_TO_INITIALIZE_COMPONENT = "Failed
to initialize the %s component: %s";
    protected static String MSG_UNSUPPORTED_DATE = "The supplied date
'%s' is not in a recognized format.";
    protected static String MSG_PRE_GENERATED = "%s";
    protected static String MSG_MISSING_RESOURCE_BUNDLE = "The resource
bundle '%s' could not be loaded: %s.";
    protected static String MSG_NO_RESOURCE_BUNDLE = "No resource bundle
was supplied.";

    /**
     * @param name -
     *          the name used to find a project that doesn't exist
     * @return
     */
    public static ApplicationException notImplemented() {
        return new ApplicationException(MSG_NOT_IMPLEMENTED);
    }

    /**
     * @param type -
     *          the class that failed to initialize
     * @param cause -
     *          the reason the initialization failed
     * @return
     */
    public static ApplicationException
failedToInitializeComponent(Class<?> type, Throwable cause) {
        return new ApplicationException(MSG_FAILED_TO_INITIALIZE_COMPONENT,
type.getName(), cause
            .toString());
    }

    /**
     * @param type -
     *          the class that failed to initialize
     * @param details -
     *          a detailed message if an exception wasn't the cause of
the

```

```

        failure.

    * @return
    */
public static ApplicationException
failedToInitializeComponent(Class<?> type, String details) {
    return new ApplicationException(MSG_FAILED_TO_INITIALIZE_COMPONENT,
type.getName(), details);
}

/***
 * @param dateString
 * @return
 */
public static ApplicationException unsupportedDateString(String
dateString) {
    return new ApplicationException(MSG_UNSUPPORTED_DATE, dateString);
}

/***
 * @param bundleName
 * @param cause
 * @return
 */
public static ApplicationException missingResourceBundle(String
bundleName, Throwable cause) {
    return new ApplicationException(cause, MSG_MISSING_RESOURCE_BUNDLE,
bundleName, cause);
}

/***
 * @return
 */
public static ApplicationException missingResourceBundle() {
    return new ApplicationException(MSG_NO_RESOURCE_BUNDLE);
}

/***
 * @param format -
 *          a format string appropriate for java.util.Formatter
 * @param args -
 *          variable args list that map to the variables in the
format
 *          string
 */
protected ApplicationException(String format, Object... args) {
    super(new Formatter().format(format, args).toString());
    if (log.isDebugEnabled()) {

```

```

        log.debug(getMessage());
    }
}

/**
 * @param cause -
 *          a caught exception that resulted in this exception
 * @param format -
 *          a format string appropriate for java.util.Formatter
 * @param args -
 *          variable args list that map to the variables in the
format
 *          string
 */
protected ApplicationException(Throwable cause, String format,
Object... args) {
    super(new Formatter().format(format, args).toString(), cause);
    if (log.isDebugEnabled()) {
        log.debug(new Formatter().format(format, args).toString(), cause);
    }
}
}
```

argument.java

```

/*
 * $Id: Argument.java,v 1.5 2008/04/14 08:57:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

import edu.harvard.fas.rregan.requel.CreatedEntity;

/**
 * An argument for or against a Position of an Annotation.
 *
 * @author ron
 */
public interface Argument extends Comparable<Argument>, CreatedEntity
{

    /**
     * The position that the argument is for or against.
     *
     * @return
     */

```

```

public Position getPosition();

/**
 * @return the text of the argument describing why a position should
be or
 *         should not be chosen as the resolution to an issue.
 */
public String getText();

/**
 * @return the level at which this argument is for or against the
position.
 */
public ArgumentPositionSupportLevel getSupportLevel();
}

```

argumenteditorpanel.java

```

/*
 * $Id: ArgumentEditorPanel.java,v 1.22 2009/02/22 09:07:23 rregan Exp
$Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.MessageFormat;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import
edu.harvard.fas.rregan.requel.annotation.ArgumentPositionSupportLevel;

```

```

import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteArgumentCommand
;
import
edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand;
import
edu.harvard.fas.rregan.requel.project.Project;
import
edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelEditorPanel;
import
edu.harvard.fas.rregan.requel.user.User;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;
/***
 * @author ron
 */
public class ArgumentEditorPanel extends AbstractRequelEditorPanel {
    private static final Logger log =
Logger.getLogger(ArgumentEditorPanel.class);

    static final long serialVersionUID = 0L;

    /**
     * The name to use in the ArgumentEditorPanel.properties file to set
the
     * label of the argument text field. If the property is undefined
"Argument"
     * is used.
     */
    public static final String PROP_LABEL_ARGUMENT = "Argument.Label";

    /**
     * The name to use in the ArgumentEditorPanel.properties file to set
the

```

```

 * label of the position support level field. If the property is
undefined
 * "Position Support Level" is used.
 */
public static final String PROP_LABEL_POSITION_SUPPORT_LEVEL =
"PositionSupportLevel.Label";

private final AnnotationCommandFactory annotationCommandFactory;
private UpdateListener updateListener;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted = false;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 */
public ArgumentEditorPanel(CommandHandler commandHandler,
    AnnotationCommandFactory annotationCommandFactory) {
    this(ArgumentEditorPanel.class.getName(), commandHandler,
annotationCommandFactory);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param annotationCommandFactory
 */
public ArgumentEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    AnnotationCommandFactory annotationCommandFactory) {
    super(resourceBundleName, Argument.class, commandHandler);
    this.annotationCommandFactory = annotationCommandFactory;
}

/**
 * If the editor is editing an existing argument the title specified
in the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Edit Argument"<br>

```

```

 * For a new argument it first tries PROP_NEW_OBJECT_PANEL_TITLE,
then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Argument"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    if (getArgument() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Edit Argument"));
        return MessageFormat.format(msgPattern, getPosition().toString());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale())
                .getString(PROP_PANEL_TITLE, "New Argument"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    Argument argument = getArgument();
    if (argument != null) {
        addInput("text", PROP_LABEL_ARGUMENT, "Argument", new TextArea(),
new StringDocumentEx(
            argument.getText()));

        addInput("supportLevel", PROP_LABEL_POSITION_SUPPORT_LEVEL,
"Position Support Level",
            new SelectField(), new CombinedListModel(
                getArgumentPositionSupportLevelNames(),
argument.getSupportLevel()
                .toString(), true));
    } else {
        addInput("text", PROP_LABEL_ARGUMENT, "Position", new TextArea(),
new StringDocumentEx());
    }
}

```

```

    addInput("supportLevel", PROP_LABEL_POSITION_SUPPORT_LEVEL,
"Position Support Level",
    new SelectField(), new CombinedListModel(
        getArgumentPositionSupportLevelNames(), "", true));
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);

}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

private Set<String> getArgumentPositionSupportLevelNames() {
    Set<String> supportLevelNames = new TreeSet<String>();
    for (ArgumentPositionSupportLevel supportLevel :
ArgumentPositionSupportLevel.values()) {
        supportLevelNames.add(supportLevel.toString());
    }
    return supportLevelNames;
}

@Override
public boolean isReadOnlyMode() {
    // project entities are subject to stakeholder permissions, so if
this
    // position is on an issue concerned with a project or project
entity
    // check the stakeholder permissions for editing annotations.
    User user = (User) getApp().getUser();
    Project project = getProject();
}

```

```

    if (project != null) {
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Annotation.class,
StakeholderPermissionType.Edit);
        }
    }
    return false;
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditArgumentCommand command =
getAnnotationCommandFactory().newEditArgumentCommand();
        command.setArgument(getArgument());
        command.setPosition(getPosition());
        command.setEditedBy(getCurrentUser());
        command.setText(getInputValue("text", String.class));
        command.setSupportLevelName(getInputValue("supportLevel",
String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
                updateListener);
        }
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getArgument()));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        }
    }
}

```

```

} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the argument: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
try {
    DeleteArgumentCommand deleteArgumentCommand =
getAnnotationCommandFactory()
    .newDeleteArgumentCommand();
    deleteArgumentCommand.setArgument(getArgument());
    deleteArgumentCommand.setEditedBy(getCurrentUser());
    deleteArgumentCommand =
getCommandHandler().execute(deleteArgumentCommand);
    deleted = true;
    getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getPosition()));
} catch (Exception e) {
    setGeneralMessage("Could not delete entity: " + e);
}
}

private Project getProject() {
Project project = null;
if (getTargetObject() != null) {
    Position position = null;
    if (getTargetObject() instanceof Position) {
        position = (Position) getTargetObject();
    } else if (getTargetObject() instanceof Argument) {
        position = ((Argument) getTargetObject()).getPosition();
    }
    if ((position != null) && !position.getIssues().isEmpty()) {
        for (Issue issue : position.getIssues()) {
            for (Annotatable annotatable : issue.getAnnotatables()) {
                if (annotatable instanceof Project) {
                    project = (Project) annotatable;
                    break;
                } else if (annotatable instanceof ProjectOrDomainEntity) {
                    ProjectOrDomainEntity podEntity = (ProjectOrDomainEntity)
annotatable;
                    if (podEntity.getProjectOrDomain() instanceof Project) {
                        project = (Project) podEntity.getProjectOrDomain();
                        break;
                    }
                }
            }
        }
    }
    return project;
}

private Position getPosition() {
if (getTargetObject() instanceof Position) {
    return (Position) getTargetObject();
} else if (getTargetObject() instanceof Argument) {
    return ((Argument) getTargetObject()).getPosition();
}
return null;
}

private Argument getArgument() {
if (getTargetObject() instanceof Argument) {
    return (Argument) getTargetObject();
}
return null;
}

private AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ArgumentEditorPanel panel;

    private UpdateListener(ArgumentEditorPanel panel) {
        this.panel = panel;
    }

    @Override

```

```

public void actionPerformed(ActionEvent e) {
    if (panel.deleted) {
        return;
    }
    Argument existingArgument = panel.getArgument();
    if ((e instanceof UpdateEntityEvent) && (existingArgument != null))
    {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        Argument updatedArgument = null;
        if (event.getObject() instanceof Argument) {
            updatedArgument = (Argument) event.getObject();
            if (existingArgument.equals(updatedArgument)) {
                if (event instanceof DeletedEntityEvent) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingArgument));
                } else {
                    panel.setInputValue("text", updatedArgument.getText());
                    panel.setTargetObject(updatedArgument);
                }
            }
        } else if ((event.getObject() instanceof Position)
            && (existingArgument.getPosition() != null)
            && existingArgument.getPosition().equals(event.getObject()))
            && (event instanceof DeletedEntityEvent)) {
            // if the position was deleted then all the arguments were
            // deleted.
            panel.deleted = true;
            panel.getEventDispatcher().dispatchEvent(
                new DeletedEntityEvent(this, panel, existingArgument));
        }
    }
}

```

argumentimpl.java

```

package edu.harvard.fas.rregan.requel.annotation.impl;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;

```

```

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.ArgumentPositionSupportLevel;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "arguments")
@XmlRootElement(name = "argument", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "argument", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ArgumentImpl implements Argument, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private Position position;

```

```

private String text;
private ArgumentPositionSupportLevel supportLevel;
private User createdBy;
private Date dateCreated = new Date();
// start at 1 so hibernate recognizes the new
// instance as the initial value and not stale.
private int version = 1;

/**
 * @param position
 * @param text
 * @param supportLevel
 * @param createdBy
 */
public ArgumentImpl(Position position, String text,
ArgumentPositionSupportLevel supportLevel,
User createdBy) {
setPosition(position);
setText(text);
setSupportLevel(supportLevel);
setCreatedBy(createdBy);
setDateCreated(new Date());
}

protected ArgumentImpl() {
// for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
return id;
}

protected void setId(Long id) {
this.id = id;
}

@Version
protected int getVersion() {
return version;
}

protected void setVersion(int version) {
    this.version = version;
}

@XmlTransient
@ManyToOne(targetEntity = PositionImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
public Position getPosition() {
    return position;
}

protected void setPosition(Position position) {
    this.position = position;
}

@XmlElement(name = "text", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

@XmlAttribute(name = "supportLevel")
public ArgumentPositionSupportLevel getSupportLevel() {
    return supportLevel;
}

public void setSupportLevel(ArgumentPositionSupportLevel
supportLevel) {
    this.supportLevel = supportLevel;
}

@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
public User getCreatedBy() {
    return createdBy;
}

protected void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

```

```

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

@Override
public int compareTo(Argument o) {
    ArgumentImpl other = (ArgumentImpl) o;
    int issueCompare = (getPosition() == null ? -1 :
    getPosition().compareTo(
        other.getPosition()));
    int dateCompare = (getDateCreated() == null ? -1 :
    getDateCreated().compareTo(
        other.getDateCreated()));
    int createdByCompare = (getCreatedBy() == null ? -1 :
    getCreatedBy().compareTo(
        other.getCreatedBy()));
    return (issueCompare != 0 ? issueCompare : (dateCompare != 0 ?
    dateCompare
        : (createdByCompare != 0 ? createdByCompare : (getText() == null ?
    -1 : getText()
        .compareTo(other.getText())))));
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getPosition() == null) ? 0 :
        getPosition().hashCode());
        result = prime * result + ((getText() == null) ? 0 :
        getText().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final ArgumentImpl other = (ArgumentImpl) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getPosition() == null) {
        if (other.getPosition() != null) {
            return false;
        }
    } else if (!getPosition().equals(other.getPosition())) {
        return false;
    }
    if (getText() == null) {
        if (other.getText() != null) {
            return false;
        }
    } else if (!getText().equals(other.getText())) {
        return false;
    }
    return true;
}

/**
 * This is for JAXB to patchup the parent/child relationship.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
 * supplied.
 * @param parent
 * @see UnmarshallerListener
 */

```

```

public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser,
    Object parent) {
    setPosition((Position) parent);
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
}

/**
 * This class is used by JAXB to convert the id of an entity into an
xml id
 * string that will be distinct from other entity xml id strings by
the use
 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "ARG_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

argumentpositionsupportlevel.java

```

package edu.harvard.fas.rregan.requel.annotation;

import javax.xml.bind.annotation.XmlEnum;
import javax.xml.bind.annotation.XmlType;
/**

```

```

 * Arguments can be for or against a position, this enumerates the
level of
 * support.
 *
 * @author ron
 */
@XmlEnum()
@XmlType(namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public enum ArgumentPositionSupportLevel {

    /**
     * This argument is in strong support of this position.
     */
    StronglyFor(2),

    /**
     * This argument is in support of the position.
     */
    For(1),

    /**
     * This argument is neither for or against the position.
     */
    Neutral(0),

    /**
     * This argument is against the position.
     */
    Against(-1),

    /**
     * This argument is strongly against the position.
     */
    StronglyAgainst(-2);

    private int supportLevel;

    private ArgumentPositionSupportLevel(int supportLevel) {
        this.supportLevel = supportLevel;
    }

    public int supportLevel() {
        return supportLevel;
    }
}
```

assistantfacade.java

```
/*
 * $Id: AssistantFacade.java,v 1.1 2009/03/30 11:54:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.core.task.TaskExecutor;
import org.springframework.core.task.TaskRejectedException;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.NLPProcessorFactory;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * A Facade for applying assistants to projects and project entities.
 *
 * @author ron
 */
@Component("assistantFacade")
@Scope("singleton")
// TODO: changed this to a singleton because of the following
exception
// Exception during analysis of goal text:
```

```
// org.springframework.beans.factory.UnsatisfiedDependencyException:
Error
// creating bean with name
//
'edu.harvard.fas.rregan.requel.project.impl.command.EditGlossaryTermCo
mmandImpl':
// Unsatisfied dependency expressed through constructor argument with
// index 0 of type
//
[edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantManager
]:
// Error creating bean with name 'assistantManager': Scope 'session'
is not
// active for the current thread; consider defining a scoped proxy for
this bean
// if you intend to refer to it from a singleton; nested exception is
// java.lang.IllegalStateException: No thread-bound request found: Are
you
// referring to request attributes outside of an actual web request?
If you are
// actually operating within a web request and still receive this
message, your
// code is probably running outside of
DispatcherServlet/DispatcherPortlet: In
// this case, use RequestContextListener or RequestContextFilter to
expose the
// current request.
public class AssistantFacade {
    private static final Logger log =
Logger.getLogger(AssistantFacade.class);

    private final TaskExecutor taskExecutor;
    private final CommandHandler commandHandler;
    private final AnnotationCommandFactory annotationCommandFactory;
    private final ProjectCommandFactory projectCommandFactory;
    private final UserRepository userRepository;
    private final ProjectRepository projectRepository;
    private final AnnotationRepository annotationRepository;
    private final DictionaryRepository dictionaryRepository;
    private final NLPProcessorFactory nlpProcessorFactory;
    private final UpdatedEntityNotifier updatedEntityNotifier;

    /**
     * @param taskExecutor
     * @param commandHandler
     * @param projectCommandFactory
     * @param annotationCommandFactory
```

```

* @param projectRepository
* @param userRepository
* @param dictionaryRepository
* @param annotationRepository
* @param nlpProcessorFactory
* @param updatedEntityNotifier -
*      after an entity is analyzed it is passed to the
notifier to
*      tell the UI components that reference the entity to
refresh
*/
@.Autowired
public AssistantFacade(TaskExecutor taskExecutor, CommandHandler
commandHandler,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory,
ProjectRepository projectRepository,
    UserRepository userRepository, DictionaryRepository
dictionaryRepository,
    AnnotationRepository annotationRepository, NLPProcessorFactory
nlpProcessorFactory,
    UpdatedEntityNotifier updatedEntityNotifier) {
this.taskExecutor = taskExecutor;
this.commandHandler = commandHandler;
this.projectCommandFactory = projectCommandFactory;
this.annotationCommandFactory = annotationCommandFactory;
this.projectRepository = projectRepository;
this.userRepository = userRepository;
this.annotationRepository = annotationRepository;
this.dictionaryRepository = dictionaryRepository;
this.nlpProcessorFactory = nlpProcessorFactory;
this.updatedEntityNotifier = updatedEntityNotifier;
}

protected CommandHandler getCommandHandler() {
    return commandHandler;
}

protected ProjectCommandFactory getProjectCommandFactory() {
    return projectCommandFactory;
}

protected AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

protected UserRepository getUserRepository() {

```

```

        return userRepository;
    }

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

protected AnnotationRepository getAnnotationRepository() {
    return annotationRepository;
}

protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}

protected TaskExecutor getTaskExecutor() {
    return taskExecutor;
}

protected NLPProcessorFactory getNlpProcessorFactory() {
    return nlpProcessorFactory;
}

protected UpdatedEntityNotifier getUpdatedEntityNotifier() {
    return updatedEntityNotifier;
}

/**
 * Analyze all the entities in a project.
 *
 * @param project
 */
public void analyzeProject(final Project project) {
try {
    getTaskExecutor().execute(new Runnable() {
        @Override
        public void run() {
            try {
                User assistantUser =
getUserRepository().findUserByUsername("assistant");
                LexicalAssistant lexicalAssistant = new LexicalAssistant(
                    getCommandHandler(), getProjectCommandFactory(),
                    getAnnotationCommandFactory(), getAnnotationRepository(),
                    getProjectRepository(), getDictionaryRepository(),
                    getNlpProcessorFactory());
                ProjectAssistant projectAssistant = new
ProjectAssistant(lexicalAssistant,

```

```

assistantUser);

// TODO: the DomainObjectWrappingAdvice that wraps
// persistence objects with a DomainObjectWrapper isn't
// getting invoked in the command's invokeAnalysis()
// method because the get() method for the domain object
// is called locally in the object. Reloading the object
// through a repository solves the problem, but using
// aspectj may be better.
projectAssistant.analyzegetRepository().get(project));
getUpdatedEntityNotifier().entityUpdated(project);
} catch (Exception e) {
    log.error("exception in project assistant: " + e, e);
}
});

log.info("started analysis of " + project);
} catch (TaskRejectedException e) {
    log.error("failed to execute analysis on '" + project + "' ", e);
}
}

/**
 * TODO: this executes the analysis in a seperate thread using
resources
 * originating in this thread. That may cause problems in a multi-
user
 * system.
 *
 * @param updatedGoal -
 *          the goal that has been edited
 * @param originalGoal -
 *          a copy of the same goal before it was edited, this is
used to
 *          limit analysis to only things that have changed. If it
is not
 *          supplied everything in updatedGoal will be analyzed.
 */
public void analyzeGoal(final Goal updatedGoal) {
try {
    getTaskExecutor().execute(new Runnable() {
        @Override
        public void run() {
            User assistantUser =
getUserRepository().findUserByUsername("assistant");
            LexicalAssistant lexicalAssistant = new
LexicalAssistant(getCommandHandler(),

```

```

getProjectCommandFactory(), getAnnotationCommandFactory(),
getAnnotationRepository(), getProjectRepository(),
getDictionaryRepository(), getNlpProcessorFactory());
GoalAssistant goalAssistant = new GoalAssistant(lexicalAssistant,
assistantUser);

// TODO: the DomainObjectWrappingAdvice that wraps
// persistence objects with a DomainObjectWrapper isn't
// getting invoked in the command's invokeAnalysis()
// method because the get() method for the domain object
// is called locally in the object. Reloading the object
// through a repository solves the problem, but using
// aspectj may be better.
goalAssistant.setEntitygetRepository().get(updatedGoal);
goalAssistant.analyze();
getUpdatedEntityNotifier().entityUpdated(updatedGoal);
})
};

log.info("started analysis of " + updatedGoal);
} catch (TaskRejectedException e) {
    log.error("failed to execute analysis on '" + updatedGoal + "' ",
e);
}
}

/**
 * @param updatedStory
 * @param originalStory
 */
public void analyzeStory(final Story updatedStory) {
try {
    getTaskExecutor().execute(new Runnable() {
        @Override
        public void run() {
            User assistantUser =
getUserRepository().findUserByUsername("assistant");
            LexicalAssistant lexicalAssistant = new
LexicalAssistant(getCommandHandler(),
                getProjectCommandFactory(), getAnnotationCommandFactory(),
                getAnnotationRepository(), getProjectRepository(),
                getDictionaryRepository(), getNlpProcessorFactory());
            StoryAssistant storyAssistant = new
StoryAssistant(lexicalAssistant,
                assistantUser);

// TODO: the DomainObjectWrappingAdvice that wraps
// persistence objects with a DomainObjectWrapper isn't

```

```

// getting invoked in the command's invokeAnalysis()
// method because the get() method for the domain object
// is called locally in the object. Reloading the object
// through a repository solves the problem, but using
// aspectj may be better.
storyAssistant.setEntity(getProjectRepository().get(updatedStory)
);
storyAssistant.analyze();
getUpdatedEntityNotifier().entityUpdated(updatedStory);
});
log.info("started analysis of " + updatedStory);
} catch (TaskRejectedException e) {
    log.error("failed to execute analysis on '" + updatedStory + "' ", e);
}
}

/**
 * @param updatedActor
 * @param originalActor
 */
public void analyzeActor(final Actor updatedActor) {
try {
    getTaskExecutor().execute(new Runnable() {
        @Override
        public void run() {
            User assistantUser =
getUserRepository().findUserByUsername("assistant");
            LexicalAssistant lexicalAssistant = new
LexicalAssistant(getCommandHandler(),
                getProjectCommandFactory(), getAnnotationCommandFactory(),
                getAnnotationRepository(), getProjectRepository(),
                getDictionaryRepository(), getNlpProcessorFactory());
            ActorAssistant actorAssistant = new
ActorAssistant(lexicalAssistant,
                assistantUser);
            // TODO: the DomainObjectWrappingAdvice that wraps
            // persistence objects with a DomainObjectWrapper isn't
            // getting invoked in the command's invokeAnalysis()
            // method because the get() method for the domain object
            // is called locally in the object. Reloading the object
            // through a repository solves the problem, but using
            // aspectj may be better.
            actorAssistant.setEntity(getProjectRepository().get(updatedActor));
        }
    });
}
}


```

```

actorAssistant.analyze();
getUpdatedEntityNotifier().entityUpdated(updatedActor);
})
);
log.info("started analysis of " + updatedActor);
} catch (TaskRejectedException e) {
    log.error("failed to execute analysis on '" + updatedActor + "' ", e);
}
}

/**
 * @param updatedUseCase
 */
public void analyzeUseCase(final UseCase updatedUseCase) {
try {
    getTaskExecutor().execute(new Runnable() {
        @Override
        public void run() {
            User assistantUser =
getUserRepository().findUserByUsername("assistant");
            LexicalAssistant lexicalAssistant = new
LexicalAssistant(getCommandHandler(),
                getProjectCommandFactory(), getAnnotationCommandFactory(),
                getAnnotationRepository(), getProjectRepository(),
                getDictionaryRepository(), getNlpProcessorFactory());
            ScenarioAssistant scenarioAssistant = new
ScenarioAssistant(lexicalAssistant,
                assistantUser);
            ActorAssistant actorAssistant = new
ActorAssistant(lexicalAssistant,
                assistantUser);
            UseCaseAssistant useCaseAssistant = new
UseCaseAssistant(lexicalAssistant,
                scenarioAssistant, actorAssistant, assistantUser);

            // TODO: the DomainObjectWrappingAdvice that wraps
            // persistence objects with a DomainObjectWrapper isn't
            // getting invoked in the command's invokeAnalysis()
            // method because the get() method for the domain object
            // is called locally in the object. Reloading the object
            // through a repository solves the problem, but using
            // aspectj may be better.
            useCaseAssistant.setEntity(getProjectRepository().get(updatedUseCase));
            useCaseAssistant.analyze();
        }
    });
}
}


```

```

        getUpdatedEntityNotifier().entityUpdated(updatedUseCase);
    }
});
log.info("started analysis of " + updatedUseCase);
} catch (TaskRejectedException e) {
    log.error("failed to execute analysis on '" + updatedUseCase + "'",
              e);
}
}

/**
 * @param updatedScenarioStep
 */
public void analyzeScenarioStep(final Step updatedScenarioStep) {
    try {
        getTaskExecutor().execute(new Runnable() {
            @Override
            public void run() {
                User assistantUser =
                    getUserRepository().findUserByUsername("assistant");
                LexicalAssistant lexicalAssistant = new
                    LexicalAssistant(getCommandHandler(),
                                     getProjectCommandFactory(), getAnnotationCommandFactory(),
                                     getAnnotationRepository(), getProjectRepository(),
                                     getDictionaryRepository(), getNlpProcessorFactory());
                ScenarioAssistant scenarioAssistant = new
                    ScenarioAssistant(lexicalAssistant,
                                      assistantUser);
                // TODO: the DomainObjectWrappingAdvice that wraps
                // persistence objects with a DomainObjectWrapper isn't
                // getting invoked in the command's invokeAnalysis()
                // method because the get() method for the domain object
                // is called locally in the object. Reloading the object
                // through a repository solves the problem, but using
                // aspectj may be better.
                scenarioAssistant.setEntity(getProjectRepository().get(updatedScenarioStep));
                scenarioAssistant.analyze();
                getUpdatedEntityNotifier().entityUpdated(updatedScenarioStep);
            }
        });
        log.info("started analysis of " + updatedScenarioStep);
    } catch (TaskRejectedException e) {
        log.error("failed to execute analysis on '" + updatedScenarioStep +
                  "' ", e);
    }
}

```

```

    /**
     * @param updatedScenario
     */
    public void analyzeScenario(final Scenario updatedScenario) {
        analyzeScenarioStep(updatedScenario);
    }
}

```

assistantuserinitializer.java

```

/*
 * $Id: AssistantUserInitializer.java,v 1.9 2009/03/29 11:59:30 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**
 * Create the assistant user if it doesn't exist. The assistant user
 * is a
 * psuedo-user that represents assistants that do analysis of project
 * entities.
 *
 * @author ron
 */
@Component("assistantUserInitializer")
@Scope("prototype")
public class AssistantUserInitializer extends AbstractSystemInitializer {

    private final UserRepository userRepository;
    private final EditUserCommand command;

```

```

private final CommandHandler commandHandler;

/**
 * @param userRepository
 * @param commandHandler
 * @param command
 */
@Autowired
public AssistantUserInitializer(UserRepository userRepository,
CommandHandler commandHandler,
EditUserCommand command) {
super(101);
this.userRepository = userRepository;
this.commandHandler = commandHandler;
this.command = command;
}

@Override
public void initialize() {
try {
userRepository.findUserByUsername("assistant");
} catch (NoSuchUserException e) {
try {
command.setUsername("assistant");
// TODO: this user shouldn't be able to login, but a password is
// required
command.setPassword("assistant");
command.setRepassword("assistant");
command.setName("Analysis Assistant");
command.setEmailAddress("rreganjr@acm.org");
command.setOrganizationName("Requel");
command.addUserRoleName(SystemAdminUserRole.getRoleName(ProjectUser
rRole.class));
command.setEditable(Boolean.FALSE);
commandHandler.execute(command);
} catch (Exception e2) {
log.error("failed to initialize the assistant user: " + e2, e2);
}
}
}
}
}

```

batchcommand.java

```

/*
 * $Id: BatchCommand.java,v 1.1 2008/12/13 00:41:01 rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.command;

import java.util.Collection;

/**
 * Execute a series of commands. If execution fails the results in
getCommands()
 * may be inconsistant.
 *
 * @author ron
 */
public interface BatchCommand extends Command {

    public void addCommand(Command command);

    public void addCommands(Collection<Command> commands);

    public Collection<Command> getCommands();
}
```

batchcommandimpl.java

```

/*
 * $Id: BatchCommandImpl.java,v 1.1 2008/12/13 00:41:04 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.command;

import java.util.Collection;
import java.util.LinkedList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

/**
 * Execute a series of commands. If execution fails the results in
getCommands()
 * may be inconsistant.
 *
```

```

* @author ron
*/
@Controller("batchCommand")
@Scope("prototype")
public class BatchCommandImpl implements BatchCommand {

    private final CommandHandler commandHandler;
    private final List<Command> commands = new LinkedList<Command>();

    /**
     * @param commandHandler
     */
    @Autowired
    public BatchCommandImpl(CommandHandler commandHandler) {
        this.commandHandler = commandHandler;
    }

    public void addCommand(Command command) {
        commands.add(command);
    }

    public void addCommands(Collection<Command> commands) {
        commands.addAll(commands);
    }

    public Collection<Command> getCommands() {
        return commands;
    }

    /**
     * @see edu.harvard.fas.rregan.command.Command#execute()
     */
    @Override
    public void execute() throws Exception {
        for (int i = 0; i < commands.size(); i++) {
            commands.set(i, commandHandler.execute(commands.get(i)));
        }
    }
}

```

buildwordnetdefinitionwordscommand.java

```

/*
 * $Id: BuildWordNetDefinitionWordsCommand.java,v 1.1 2008/12/13
 * 00:40:11 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

*/
package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;

/**
 * Take a Synset and create SynsetDefinitionWords for
 * the word forms defined in the synset wsd field.
 *
 * @author ron
 */
public interface BuildWordNetDefinitionWordsCommand extends Command {

    /**
     *
     * @param synset - the synset to build the words for
     */
    public void setSynset(Synset synset);
}

```

calculatewordfrequencycommand.java

```

/*
 * $Id: CalculateWordFrequencyCommand.java,v 1.1 2008/12/13 00:40:12
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;

/**
 * @author ron
 */
public interface CalculateWordFrequencyCommand extends Command {
}

```

calculatewordfrequencycommandimpl.java

```

/*

```

```

* $Id: CalculateWordFrequencyCommandImpl.java,v 1.1 2008/12/13
00:39:53 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorFile;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentence;
import
edu.harvard.fas.rregan.nlp.dictionary.command.CalculateWordFrequencyCo
mmand;

/**
 * @author ron
 */
@Controller("calculateWordFrequencyCommand")
@Scope("prototype")
public class CalculateWordFrequencyCommandImpl extends
AbstractDictionaryCommand implements
CalculateWordFrequencyCommand {

    /**
     * @param dictionaryRepository
     * @param lemmatizer
     */
    @Autowired
    public CalculateWordFrequencyCommandImpl(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }

    /**
     * @see edu.harvard.fas.rregan.command.Command#execute()
     */
    @Override
    public void execute() throws Exception {
        for (SemcorFile file : getDictionaryRepository().findSemcorFiles())
{
            for (SemcorSentence sentence : file.getSentences()) {
}
}
}
}

```

```

}
}
```

category.java

```

/*
 * $Id: Category.java,v 1.2 2009/01/03 10:24:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

/**
 * Wordnet Synset Category
 */
@Entity
@Table(name = "categorydef")
@XmlRootElement(name = "category")
public class Category implements Comparable<Category>, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private String name;
    private String pos;

    protected Category() {
}

    @Id
    @Column(name = "categoryid", unique = true, nullable = false)
    @XmlID
    @XmlAttribute(name = "id")
    @XmlJavaTypeAdapter(IdAdapter.class)
    public Long getId() {
}
}

```

```

    return this.id;
}

public void setId(Long id) {
    this.id = id;
}

@Column(name = "name", length = 32)
public String getName() {
    return this.name;
}

public void setName(String name) {
    this.name = name;
}

@Column(name = "pos", length = 2)
public String getPos() {
    return this.pos;
}

public void setPos(String pos) {
    this.pos = pos;
}

@Override
public int compareTo(Category o) {
    return getName().compareTo(o.getName()) * 100 +
    getPos().compareTo(o.getPos());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getName() == null) ? 0 :
        getName().hashCode());
        result = prime * result + ((getPos() == null) ? 0 :
        getPos().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final Category other = (Category) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getName() == null) {
        if (other.getName() != null) {
            return false;
        }
    } else if (!getName().equals(other.getName())) {
        return false;
    }
    if (getPos() == null) {
        if (other.getPos() != null) {
            return false;
        }
    } else if (!getPos().equals(other.getPos())) {
        return false;
    }
    return true;
}

/**
 * This class is used by JAXB to convert the id of an entity into an
 * xml id
 * string that will be distinct from other entity xml id strings by
 * the use
 * of a prefix.
 *
 * @author ron
 */
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "CAT_";
}

```

```

@Override
public Long unmarshal(String id) throws Exception {
    return new Long(id.substring(prefix.length()));
}

@Override
public String marshal(Long id) throws Exception {
    if (id != null) {
        return prefix + id.toString();
    }
    return "";
}
}

```

changespellingposition.java

```

/*
 * $Id: ChangeSpellingPosition.java,v 1.14 2009/01/08 10:11:23 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * This is a special position such that when it is chosen as the
 * resolution of
 * an issue via resolveIssue() the annotatable that the issue is
 * assigned to is
 * updated based on the description of what the issue problem was.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.impl.ChangeSpellingPosition"
)

```

```

@XmlRootElement(name = "changeSpellingPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "changeSpellingPosition", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ChangeSpellingPosition extends PositionImpl {
    static final long serialVersionUID = 0L;

    private String proposedWord;

    /**
     * @param text
     * @param createdBy
     * @param proposedWord
     */
    public ChangeSpellingPosition(String text, User createdBy, String
proposedWord) {
        super(ChangeSpellingPosition.class.getName(), text, createdBy);
        setProposedWord(proposedWord);
    }

    protected ChangeSpellingPosition() {
        super();
        // for hibernate
    }

    /**
     * @return the proposed word for changing the spelling of the issue
     * word.
     */
    @XmlAttribute(name = "proposedWord")
    public String getProposedWord() {
        return proposedWord;
    }

    /**
     * @param proposedWord -
     *          the suggested word that should replace the misspelled
     * word in
     *          the annotatable object.
     */
    public void setProposedWord(String proposedWord) {
        this.proposedWord = proposedWord;
    }
}

```

checkboxmanipulator.java

```
/*
 * $Id: CheckBoxManipulator.java,v 1.8 2008/10/11 21:47:44 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.button.ToggleButtonModel;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * A generic interface for manipulating a CheckBox control.
 *
 * @author ron
 */
public class CheckBoxManipulator extends AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getComponent(component).isSelected());
    }

    public void setValue(Component component, Object value) {
        if ((value != null) && (value instanceof Boolean)) {
            getComponent(component).setSelected((Boolean) value);
        } else {
            getComponent(component).setSelected(false);
        }
    }

    public void addListenerToDetectChangesToInput(final EditMode
        editMode, Component component) {
        getComponent(component).addChangeListener(new ChangeListener() {
            static final long serialVersionUID = 0L;

            public void stateChanged(ChangeEvent e) {
                editMode.setStateEdited(true);
            }
        });
    }
}
```

```
@Override
public ToggleButtonModel getModel(Component component) {
    return (ToggleButtonModel) getComponent(component).getModel();
}

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setModel((ToggleButtonModel) valueModel);
}

private CheckBox getComponent(Component component) {
    return (CheckBox) component;
}
}
```

checkboxset.java

```
/*
 * $Id: CheckBoxSet.java,v 1.5 2008/10/11 08:22:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.layout.RowStyleData;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.CheckBoxSetManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.ComponentManipulators;

/**
 * TODO: make the checkboxes layout in a grid such that the options
 * are
 * displayed in as close to a square configuration as possible.

```

```

/*
 * @author ron
 */
public class CheckBoxSet extends Row {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(CheckBoxSet.class, new
        CheckBoxSetManipulator());
    }

    private final Map<String, CheckBox> optionBoxes = new HashMap<String,
    CheckBox>();
    private final RowLayoutData firstCellLayout;
    private final RowLayoutData restCellLayout;

    private CheckBoxSetModel checkBoxModel;

    /**
     *
     */
    public CheckBoxSet() {
        this(new CheckBoxSetModel());
    }

    /**
     * @param checkBoxModel
     */
    public CheckBoxSet(CheckBoxSetModel checkBoxModel) {
        firstCellLayout = new RowLayoutData();
        firstCellLayout.setAlignment(new Alignment(Alignment.CENTER,
        Alignment.CENTER));
        firstCellLayout.setInsets(new Insets(0, 0, 5, 0));

        restCellLayout = new RowLayoutData();
        restCellLayout.setAlignment(new Alignment(Alignment.CENTER,
        Alignment.CENTER));
        restCellLayout.setInsets(new Insets(5, 0, 5, 0));

        setModel(checkBoxModel);
        setStyleName(Panel.STYLE_NAME_DEFAULT);
    }

    /**
     * @param options
     * @param initialSelection
     */
    public CheckBoxSet(Set<String> options, Set<String> initialSelection,
    boolean singleSelection) {
        this(new CheckBoxSetModel(options, initialSelection,
        singleSelection));
    }

    @Override
    public void dispose() {
        super.dispose();
        removeAll();
        optionBoxes.clear();
    }

    /**
     * @return
     */
    public CheckBoxSetModel getModel() {
        return checkBoxModel;
    }

    /**
     * @param checkBoxModel
     */
    public void setModel(CheckBoxSetModel checkBoxModel) {
        this.checkBoxModel = checkBoxModel;
        optionBoxes.clear();

        // the layout for the first cell doesn't have an inset on the left
        RowLayoutData cellLayout = firstCellLayout;
        for (String option : checkBoxModel.getOptions()) {
            CheckBox optionBox = new CheckBox(option);
            optionBox.setStyleName(Panel.STYLE_NAME_DEFAULT);
            optionBox.setModel(checkBoxModel.getToggleButtonModel(option));
            optionBoxes.put(option, optionBox);
            optionBox.setLayoutData(cellLayout);

            // set the layout for the next cell
            cellLayout = restCellLayout;
        }
    }
}

```

checkboxsetmanipulator.java

```
/*
 * $Id: CheckBoxSetManipulator.java,v 1.6 2008/10/13 22:58:58 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import java.util.Collection;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;
import edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxSet;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxSetModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * A generic interface for manipulating a CheckBoxSet control.
 *
 * @author ron
 */
public class CheckBoxSetManipulator extends
AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getModel(component).getSelectedOptions());
    }

    public void setValue(Component component, Object value) {
        getModel(component).clearSelection();
        if (value instanceof Collection<?>) {
            for (Object o : (Collection<?>) value) {
                setSingleValue(component, o);
            }
        } else {
            setSingleValue(component, value);
        }
    }

    private void setSingleValue(Component component, Object value) {
        getModel(component). setSelected((String) value, true);
    }
}
```

```
public void addListenerToDetectChangesToInput(finalEditMode
editMode, Component component) {
    getModel(component).addChangeListener(new ChangeListener() {
        static final long serialVersionUID = 0L;

        public void stateChanged(ChangeEvent e) {
            editMode.setStateEdited(true);
        }
    });
}

@Override
public CheckBoxSetModel getModel(Component component) {
    return getComponent(component).getModel();
}

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setModel((CheckBoxSetModel) valueModel);
}

private CheckBoxSet getComponent(Component component) {
    return (CheckBoxSet) component;
}
}
```

checkboxsetmodel.java

```
/*
 * $Id: CheckBoxSetModel.java,v 1.2 2008/03/17 09:49:27 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import nextapp.echo2.app.button.DefaultToggleButtonModel;
import nextapp.echo2.app.button.ToggleButtonModel;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;

/**

```

```

 * @author ron
 */
public class CheckBoxSetModel implements ChangeListener {
    static final long serialVersionUID = 0L;

    private final Set<ChangeListener> changeListeners = new
    HashSet<ChangeListener>();
    private final Map<String, ToggleButtonModel> checkBoxModels = new
    HashMap<String, ToggleButtonModel>();
    private boolean singleSelection = false;

    /**
     *
     */
    public CheckBoxSetModel() {
    }

    /**
     * @param options
     * @param initialSelection
     * @param singleSelection
     */
    public CheckBoxSetModel(Set<String> options, Set<String>
initialSelection,
        boolean singleSelection) {
        setSingleSelection(singleSelection);
        for (String option : options) {
            ToggleButtonModel optionModel = new DefaultToggleButtonModel();
            optionModel.setSelected(initialSelection.contains(option));
            putToggleButtonModel(option, optionModel);
        }
    }

    /**
     * Add a ToggleButtonModel that holds the state for the specified
     * check
     * option.
     *
     * @param optionName
     * @param model
     */
    public void putToggleButtonModel(String optionName, ToggleButtonModel
model) {
        model.addChangeListener(this);
        checkBoxModels.put(optionName, model);
    }
}

```

```

 /**
 * Return the underlying ToggleButtonModel for a specific option.
 *
 * @param optionName
 * @return
 */
public ToggleButtonModel getToggleButtonModel(String optionName) {
    return checkBoxModels.get(optionName);
}

/**
 * Configure the model to support single or multiple selection.
 *
 * @param singleSelection -
 *           set true if only one checkbox can be checked at one
time
 */
public void setSingleSelection(boolean singleSelection) {
    this.singleSelection = singleSelection;
}

/**
 * @return true if only one checkbox can be checked at one time.
 */
public boolean isSingleSelection() {
    return singleSelection;
}

public boolean isSelected(String option) {
    if (checkBoxModels.containsKey(option)) {
        return checkBoxModels.get(option).isSelected();
    }
    return false;
}

public void setSelected(String option, boolean selected) {
    if (checkBoxModels.containsKey(option)) {
        checkBoxModels.get(option).setSelected(selected);
    }
}

public void clearSelection() {
    for (String optionName : checkBoxModels.keySet()) {
        if (checkBoxModels.get(optionName).isSelected()) {
            checkBoxModels.get(optionName).setSelected(false);
        }
    }
}

```

```

}

/**
 * Return the set of options
 *
 * @return
 */
public Set<String> getOptions() {
    Set<String> options = new HashSet<String>();
    for (String optionName : checkBoxModels.keySet()) {
        options.add(optionName);
    }
    return options;
}

/**
 * Return the set of selected options by name.
 *
 * @return
 */
public Set<String> getSelectedOptions() {
    Set<String> selectedOptions = new HashSet<String>();
    for (String optionName : checkBoxModels.keySet()) {
        if (checkBoxModels.get(optionName).isSelected()) {
            selectedOptions.add(optionName);
        }
    }
    return selectedOptions;
}

@Override
public void stateChanged(ChangeEvent e) {
    if (isSingleSelection()) {
        ToggleButtonModel source = (ToggleButtonModel) e.getSource();
        if (source.isSelected()) {
            for (ToggleButtonModel model : checkBoxModels.values()) {
                if (!model.equals(source)) {
                    model.setSelected(false);
                }
            }
        }
    }
}

public void addChangeListener(ChangeListener l) {
    changeListeners.add(l);
}

```

```

public void removeChangeListener(ChangeListener l) {
    changeListeners.remove(l);
}

/**
 * Notifies all listeners that have registered for this event type.
 */
public void fireStateChanged() {
    if (!changeListeners.isEmpty()) {
        ChangeEvent e = new ChangeEvent(this);
        for (ChangeListener listener : changeListeners) {
            listener.stateChanged(e);
        }
    }
}

```

checkboxtreeiset.java

```

/*
 * $Id: CheckBoxTreeSet.java,v 1.4 2009/02/20 09:32:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.Set;

import nextapp.echo2.app.Component;
import echopointng.Tree;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.DefaultTreeCellRenderer;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.CheckBoxTreeSetManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.ComponentManipulators;

/**
 * @author ron
 */
public class CheckBoxTreeSet extends Tree {
    static final long serialVersionUID = 0L;

```

```

static {
    ComponentManipulators.setManipulator(CheckBoxTreeSet.class,
        new CheckBoxTreeSetManipulator());
}

/**
 *
 */
public CheckBoxTreeSet() {
    this(new CheckBoxTreeSetModel());
}

/***
 * @param optionPaths
 * @param initialSelection
 */
public CheckBoxTreeSet(Set<String> optionPaths, Set<String>
initialSelection) {
    this(new CheckBoxTreeSetModel(optionPaths, initialSelection));
}

/***
 * @param checkBoxModel
 */
public CheckBoxTreeSet(CheckBoxTreeSetModel checkBoxModel) {
    setModel(checkBoxModel);
    setCellRenderer(new PathLevelEnablementTreeCellRenderer());
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}

/***
 * @param checkBoxModel
 */
public void setModel(CheckBoxTreeSetModel checkBoxModel) {
    super.setModel(checkBoxModel);
}

@Override
public void setEnabled(boolean newValue) {
    if ((getModel() == null) || !((CheckBoxTreeSetModel)
getModel()).isPathLevelEnablement()) {

```

```

        super.setEnabled(newValue);
    }
}

private static class PathLevelEnablementTreeCellRenderer extends
DefaultTreeCellRenderer {
    static final long serialVersionUID = 0L;

    @Override
    public Component getTreeCellRendererComponent(Tree tree, Object
node, boolean selected,
        boolean expanded, boolean leaf) {
        if ((tree instanceof CheckBoxTreeSet)
&& (tree.getModel() instanceof CheckBoxTreeSetModel)) {
            CheckBoxTreeSetModel checkBoxTreeSetModel = (CheckBoxTreeSetModel)
tree.getModel();
            if (node instanceof DefaultMutableTreeNode) {
                Object value = ((DefaultMutableTreeNode) node).getUserObject();
                if (value instanceof Component) {
                    Component c = (Component) value;
                    if (!checkBoxTreeSetModel.isPathLevelEnablement()) {
                        c.setEnabled(tree.isRenderEnabled());
                    }
                    return c;
                }
            }
        }
        return null;
    }
}

```

checkboxtreesetmanipulator.java

```

/*
 * $Id: CheckBoxTreeSetManipulator.java,v 1.5 2008/10/13 22:58:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.manipulators;

import java.util.Collection;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ChangeEvent;

```

```

import nextapp.echo2.app.event.ChangeListener;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSet;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSetModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * A generic interface for manipulating a CheckBoxTreeSet control.
 *
 * @author ron
 */
public class CheckBoxTreeSetManipulator extends
AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getModel(component).getSelectedOptions());
    }

    public void setValue(Component component, Object value) {
        getModel(component).clearSelection();
        if (value instanceof Collection<?>) {
            for (Object o : (Collection<?>) value) {
                setSingleValue(component, o);
            }
        } else {
            setSingleValue(component, value);
        }
    }

    private void setSingleValue(Component component, Object value) {
        getModel(component).setSelected((String) value, true);
    }

    public void addListenerToDetectChangesToInput(final EditMode
editMode, Component component) {
        getModel(component).addChangeListener(new ChangeListener() {
            static final long serialVersionUID = 0L;

            public void stateChanged(ChangeEvent e) {
                editMode.setStateEdited(true);
            }
        });
    }

    @Override
    public CheckBoxTreeSetModel getModel(Component component) {

```

```

        return (CheckBoxTreeSetModel) getComponent(component).getModel();
    }

    @Override
    public void setModel(Component component, Object valueModel) {
        getComponent(component).setModel((CheckBoxTreeSetModel) valueModel);
    }

    private CheckBoxTreeSet getComponent(Component component) {
        return (CheckBoxTreeSet) component;
    }
}

```

checkboxtreeisetmodel.java

```

/*
 * $Id: CheckBoxTreeSetModel.java,v 1.6 2009/02/20 10:26:16 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.button.DefaultToggleButtonModel;
import nextapp.echo2.app.button.ToggleButtonModel;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;

import org.apache.log4j.Logger;

import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.DefaultTreeModel;

/**
 * A tree model for a CheckBoxTreeSet where each path is assigned a
 * ToggleButtonModel representing on/off or true/false for each path.
 */

```

```

 * @author ron
 */
public class CheckBoxTreeSetModel extends DefaultTreeModel implements
ChangeListener {
    static final long serialVersionUID = 0L;
    private static final Logger log =
Logger.getLogger(CheckBoxTreeSetModel.class);
    private final Set<ChangeListener> changeListeners = new
HashSet<ChangeListener>();
    private final Map<String, ToggleButtonModel> checkBoxModels = new
HashMap<String, ToggleButtonModel>();
    private Set<String> editablePaths = null;
    private boolean checkboxesOnLeavesOnly = false;

    /**
     * create an empty model
     */
    public CheckBoxTreeSetModel() {
        this(null);
    }

    /**
     * @param editablePaths
     */
    protected CheckBoxTreeSetModel(Set<String> editablePaths) {
        super(new DefaultMutableTreeNode("", true));
        this.editablePaths = editablePaths;
    }

    /**
     * create a model with unix style paths representing the tree nodes
     * along
     * with a set of paths representing the nodes that are initially
     * checked.
     * Each level in the tree will have a checkbox except for the root
     * and all
     * nodes will be enabled or disabled via the setEnabled() method.
     *
     * @param optionPaths
     * @param initialSelection
     */
    public CheckBoxTreeSetModel(Set<String> optionPaths, Set<String>
initialSelection) {
        this(optionPaths, null, initialSelection, false);
    }

    /**

```

```

     * create a model with unix style paths representing the tree nodes
     * along
     * with a set of paths that are editable and a set of paths
     * representing the
     * nodes that are initially checked. if checkboxesOnLeavesOnly is
     * true only
     * the leaves will have checkboxes, otherwise all nodes at all levels
     * will
     * have a checkbox except for the root.<br>
     * NOTE: the setEnabled() method will have no effect on the state of
     * the
     * checkboxes.
     *
     * @param optionPaths
     * @param editablePaths
     * @param initialSelection
     * @param checkboxesOnLeavesOnly
     */
    public CheckBoxTreeSetModel(Set<String> optionPaths, Set<String>
editablePaths,
        Set<String> initialSelection, boolean checkboxesOnLeavesOnly) {
        this(editablePaths);
        this.checkboxesOnLeavesOnly = checkboxesOnLeavesOnly;
        log.debug("optionPaths = " + optionPaths);
        log.debug("initialSelection = " + initialSelection);
        for (String optionPath : optionPaths) {
            buildPath(optionPath, initialSelection);
        }
    }

    private void buildPath(String optionPath, Set<String>
initialSelection) {
        log.debug("build path: " + optionPath);
        DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode)
getRoot();

        String pathRoot = "";
        for (String pathSegment : optionPath.split("/")) {
            log.debug("segment: " + pathSegment);
            Enumeration<DefaultMutableTreeNode> childrenEnum =
parentNode.children();
            Component nodeComponent = null;
            boolean foundMatch = false;
            while (childrenEnum.hasMoreElements()) {
                DefaultMutableTreeNode node = childrenEnum.nextElement();

                nodeComponent = (Component) node.getUserObject();

```

```

if (nodeComponent instanceof CheckBox) {
    if (((CheckBox) nodeComponent).getText().equals(pathSegment)) {
        log.debug("found existing node for " + pathSegment);
        parentNode = node;
        foundMatch = true;
        break;
    }
} else if (nodeComponent instanceof Label) {
    if (((Label) nodeComponent).getText().equals(pathSegment)) {
        log.debug("found existing node for " + pathSegment);
        parentNode = node;
        foundMatch = true;
        break;
    }
}
if (!foundMatch) {
    if (!checkboxesOnLeavesOnly || optionPath.endsWith(pathSegment)) {
        log.debug("creating checkbox node for " + pathSegment);
        ToggleButtonModel optionModel = new DefaultToggleButtonModel();
        optionModel.setSelected(initialSelection.contains(pathRoot +
pathSegment));
        putToggleButtonModel(pathRoot + pathSegment, optionModel);
        nodeComponent = new CheckBox(pathSegment);
        if (editablePaths != null) {
            if (editablePaths.contains(optionPath)) {
                nodeComponent.setEnabled(true);
            } else {
                nodeComponent.setEnabled(false);
            }
        }
        ((CheckBox) nodeComponent).setModel(optionModel);
    } else {
        log.debug("creating label node for " + pathSegment);
        nodeComponent = new Label(pathSegment);
    }
    DefaultMutableTreeNode newNode = new
DefaultMutableTreeNode(nodeComponent, true);
    insertNodeInto(newNode, parentNode, 0);
    parentNode = newNode;
}
pathRoot = pathRoot + pathSegment + "/";
}
}

/**
 * Add a ToggleButtonModel that holds the state for the specified
 * check
 * option.
 *
 * @param optionPath
 * @param model
 */
public void putToggleButtonModel(String optionPath, ToggleButtonModel
model) {
    log.debug("adding model for path " + optionPath);
    model.addChangeListener(this);
    checkBoxModels.put(optionPath, model);
}

/**
 * Return the underlying ToggleButtonModel for a specific option.
 *
 * @param optionPath
 * @return
 */
public ToggleButtonModel getToggleButtonModel(String optionPath) {
    return checkBoxModels.get(optionPath);
}

/**
 * @param optionPath
 * @return
 */
public boolean isSelected(String optionPath) {
    if (checkBoxModels.containsKey(optionPath)) {
        return checkBoxModels.get(optionPath).isSelected();
    }
    return false;
}

/**
 * @return true if checkboxes are enabled/disabled by specific paths
 *         specified in the constructor via editablePaths.
 */
public boolean isPathLevelEnablement() {
    return (editablePaths != null);
}

/**
 * @param optionPath
 * @param selected
 */

```

```

public void setSelected(String optionPath, boolean selected) {
    if (checkBoxModels.containsKey(optionPath)) {
        checkBoxModels.get(optionPath).setSelected(selected);
    }
}

/**
 * Mark all the checkboxes as unchecked.
 */
public void clearSelection() {
    for (String optionPath : checkBoxModels.keySet()) {
        if (checkBoxModels.get(optionPath).isSelected()) {
            checkBoxModels.get(optionPath).setSelected(false);
        }
    }
}

/**
 * Mark all the checkboxes as checked.
 */
public void selectAll() {
    for (String optionPath : checkBoxModels.keySet()) {
        if (!checkBoxModels.get(optionPath).isSelected()) {
            checkBoxModels.get(optionPath).setSelected(true);
        }
    }
}

/**
 * Return the set of optionPaths
 */
@return
public Set<String> getOptions() {
    Set<String> options = new HashSet<String>();
    for (String optionPath : checkBoxModels.keySet()) {
        options.add(optionPath);
    }
    return options;
}

/**
 * Return the set of selected options by path.
 */
@return
public Set<String> getSelectedOptions() {

```

```

    Set<String> selectedOptions = new HashSet<String>();
    log.debug("optionPaths: " + checkBoxModels.keySet());

    for (String optionPath : checkBoxModels.keySet()) {
        log.debug("optionPath: " + optionPath);
        if (checkBoxModels.get(optionPath).isSelected()) {
            log.debug("is checked: " + optionPath);
            selectedOptions.add(optionPath);
        }
    }
    return selectedOptions;
}

@Override
public void stateChanged(ChangeEvent e) {
    // TODO: toggle parent on? disable children when parent is off?
}

/**
 * @param l
 */
public void addChangeListener(ChangeListener l) {
    changeListeners.add(l);
}

/**
 * @param l
 */
public void removeChangeListener(ChangeListener l) {
    changeListeners.remove(l);
}

/**
 * Notifies all listeners that have registered for this event type.
 */
public void fireStateChanged() {
    if (!changeListeners.isEmpty()) {
        ChangeEvent e = new ChangeEvent(this);
        for (ChangeListener listener : changeListeners) {
            listener.stateChanged(e);
        }
    }
}

```

classpathfilemanagerimpl.java

```
/*
 * $Id: ClassPathFileManagerImpl.java,v 1.2 2008/05/21 09:22:01 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl.wordnet;

import java.io.IOException;
import java.net.URL;
import java.net.URLDecoder;
import java.util.Map;

import net.didion.jwnl.JWNLEException;
import net.didion.jwnl.JWNLRuntimeException;
import net.didion.jwnl.dictionary.file.DictionaryFile;
import net.didion.jwnl.dictionary.file_manager.FileManager;
import net.didion.jwnl.dictionary.file_manager.FileManagerImpl;
import net.didion.jwnl.util.factory.Param;

/**
 * Extends the default JWNL file manager to use a classpath relative
path if the
 * supplied path starts with "classpath:"
 */
public class ClassPathFileManagerImpl extends FileManagerImpl
implements FileManager {

    /**
     * Uninitialized FileManagerImpl.
     */
    public ClassPathFileManagerImpl() {
    }

    /**
     * Construct a file manager backed by a set of files contained in the
     * default WN search directory.
     *
     * @param searchDir
     * @param dictionaryFileType
     * @throws IOException
     */
    public ClassPathFileManagerImpl(String searchDir, Class<?>
dictionaryFileType)
        throws IOException {
    }
}
```

```
        super(searchDir, dictionaryFileType);
    }

    /**
     * This is a copy of the FileManagerImpl.create() method with the
addtion of
     * the code implementing support for the PATH starting with
classpath: to
     * search for the path relative to the classpath.
     */
    @Override
    public Object create(Map params) throws JWNLEException {
        Class<?> fileClass = null;
        try {
            fileClass = Class.forName(((Param)
params.get(FILE_TYPE)).getValue());
        } catch (ClassNotFoundException ex) {
            throw new JWNLRuntimeException("DICTIONARY_EXCEPTION_002", ex);
        }
        checkFileType(fileClass);

        String path = ((Param) params.get(PATH)).getValue();
        try {
            if (path.startsWith("classpath:")) {
                URL dictPath =
getClass().getResource(path.substring("classpath:".length()));
                path = URLDecoder.decode(dictPath.getPath(), "utf8");
            }
            return new ClassPathFileManagerImpl(path, fileClass);
        } catch (IOException ex) {
            throw new JWNLEException("DICTIONARY_EXCEPTION_016", fileClass, ex);
        }
    }

    /**
     * Checks the type to ensure it's valid.<br>
     * NOTE: This is an exact copy of the checkFileType() method from
     * FileManagerImpl because it is private but used by the create()
method.
     *
     * @param c
     */
    private void checkFileType(Class<?> c) {
        if (!DictionaryFile.class.isAssignableFrom(c)) {
            throw new JWNLRuntimeException("DICTIONARY_EXCEPTION_003", c);
        }
    }
}
```

```
}
```

closepaneevent.java

```
/*
 * $Id: ClosePanelEvent.java,v 1.7 2008/09/12 01:00:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import edu.harvard.fas.rregan.uiframework.panel.Panel;

/**
 * An event that causes a specified panel to be closed.
 *
 * @author ron
 */
public class ClosePanelEvent extends NavigationEvent {
    static final long serialVersionUID = 0;

    private final Panel panelToClose;

    public ClosePanelEvent(Panel panelToClose) {
        this(panelToClose, panelToClose);
    }

    /**
     * @param source
     * @param panelToClose
     */
    public ClosePanelEvent(Object source, Panel panelToClose) {
        this(source, ClosePanelEvent.class.getName(), panelToClose, null);
    }

    protected ClosePanelEvent(Object source, String command, Panel
panelToClose,
        Object destinationObject) {
        super(source, command, destinationObject);
        this.panelToClose = panelToClose;
    }

    /**
     * @return the panel that should be closed.
     */
    public Panel getPanelToClose() {
        return panelToClose;
    }
}
```

```
}
```

collectionmanipulator.java

```
/*
 * $Id: CollectionManipulator.java,v 1.10 2008/10/15 09:20:06 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class CollectionManipulator extends
AbstractComponentManipulator {

    public void addListenerToDetectChangesToInput(EditMode editMode,
Component container) {
        for (int i = 0; i < container.getComponentCount(); i++) {
            Component component = container.getComponent(i);
            ComponentManipulator man =
ComponentManipulators.getManipulator(component);
            if (man != null) {
                man.addListenerToDetectChangesToInput(editMode, component);
            }
        }
    }

    @Override
    public Object getModel(Component container) {
        for (int i = 0; i < container.getComponentCount(); i++) {
            Component component = container.getComponent(i);
            ComponentManipulator man =
ComponentManipulators.getManipulator(component);
            if (man != null) {
                return man.getModel(component);
            }
        }
        return null;
    }
}
```

```

@Override
public Object getOptionModel(Component container) {
    for (int i = 0; i < container.getComponentCount(); i++) {
        Component component = container.getComponent(i);
        ComponentManipulator man =
            ComponentManipulators.getManipulator(component);
        if (man != null) {
            return man.getModel(component);
        }
    }
    return null;
}

@Override
public void setModel(Component container, Object valueModel) {
    for (int i = 0; i < container.getComponentCount(); i++) {
        Component component = container.getComponent(i);
        ComponentManipulator man =
            ComponentManipulators.getManipulator(component);
        if (man != null) {
            man.setModel(component, valueModel);
        }
    }
}

@Override
public void setOptionModel(Component container, Object optionModel) {
    for (int i = 0; i < container.getComponentCount(); i++) {
        Component component = container.getComponent(i);
        ComponentManipulator man =
            ComponentManipulators.getManipulator(component);
        if (man != null) {
            man.setModel(component, optionModel);
        }
    }
}

public <T> T getValue(Component container, Class<T> type) {
    for (int i = 0; i < container.getComponentCount(); i++) {
        Component component = container.getComponent(i);
        ComponentManipulator man =
            ComponentManipulators.getManipulator(component);
        if (man != null) {
            return man.getValue(component, type);
        }
    }
    return null;
}

}

public void setValue(Component container, Object value) {
    for (int i = 0; i < container.getComponentCount(); i++) {
        Component component = container.getComponent(i);
        ComponentManipulator man =
            ComponentManipulators.getManipulator(component);
        if (man != null) {
            man.setValue(component, value);
        }
    }
}
}

```

colocationsenserelationinfo.java

```

/*
 * $Id: ColocationSenseRelationInfo.java,v 1.1 2008/12/14 11:36:12
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * @author ron
 */
public class ColocationSenseRelationInfo extends
AbstractSenseRelationInfo {

    /**
     * Enum defining the source of the colocation.
     */
    public static enum ColocationSource {
        /**
         * The colocation occurs in a disambiguated wordnet synset
         * definition.
         */
        WordNetDefinition(),

        /**
         * The colocation occurs in a Semcor Corpus sentence.
         */
        SemcorSentence();
    }
}

```

```

    */
SemcorCorpus();

}

private final ColocationSource colocationSource;

protected ColocationSenseRelationInfo(ColocationSource
colocationSource, Sense sense1,
    Sense sense2, String reason) {
super(sense1, sense2, 0.25, reason);
this.colocationSource = colocationSource;
}

@Override
public String toString() {
    return "colocation(" + colocationSource + ", " + getSense1() + ", "
+ getSense2() + ") -> "
    + getRank() + " {" + getReason() + "}";
}
}

```

combinedlistmodel.java

```

/*
 * $Id: CombinedListModel.java,v 1.7 2008/10/13 22:58:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.Collections;
import java.util.Set;

import nextapp.echo2.app.event.ChangeListener;
import nextapp.echo2.app.event.ListDataListener;
import nextapp.echo2.app.list.DefaultListModel;
import nextapp.echo2.app.list.DefaultListSelectionModel;
import nextapp.echo2.app.list.ListModel;
import nextapp.echo2.app.list.ListSelectionModel;

/**
 * @author ron
 */
public class CombinedListModel implements ListModel,
ListSelectionModel {

```

```

static final long serialVersionUID = 0L;

final private DefaultListSelectionModel listSelectionModel;
final private DefaultListModel listModel;

/**
 * @param options
 * @param initialSelection
 * @param singleSelection
 */
public CombinedListModel(Set<?> options, Object initialSelection,
boolean singleSelection) {
    this(options, Collections.singleton(initialSelection),
singleSelection);
}

/**
 * @param options
 * @param initialSelection
 * @param singleSelection
 */
public CombinedListModel(Set<?> options, Set<?> initialSelection,
boolean singleSelection) {
    if (singleSelection && (initialSelection.size() > 1)) {
        throw new IllegalArgumentException(
            "single selection was specified, but more than one initial
selected item was supplied.");
    }
    listModel = new DefaultListModel(options.toArray());
    listSelectionModel = new DefaultListSelectionModel();
    listSelectionModel.setSelectionMode(singleSelection ?
ListSelectionModel.SINGLE_SELECTION
        : ListSelectionModel.MULTIPLE_SELECTION);
    for (Object item : initialSelection) {
        int index = listModel.indexOf(item);
        if (index > -1) {
            listSelectionModel.setSelectedIndex(index, true);
        }
    }
}

/**
 * @param listSelectionModel
 * @param listModel
 */
public CombinedListModel(DefaultListSelectionModel
listSelectionModel,

```

```

    DefaultListModel listModel) {
this.listSelectionModel = listSelectionModel;
this.listModel = listModel;
}

public void addListDataListener(ListDataListener l) {
listModel.addListDataListener(l);
}

public int indexOf(Object item) {
return listModel.indexOf(item);
}

public Object get(int index) {
if ((index > -1) && (index < listModel.size())) {
return listModel.get(index);
}
return null;
}

public void removeListDataListener(ListDataListener l) {
listModel.removeListDataListener(l);
}

public int size() {
return listModel.size();
}

public void addChangeListener(ChangeListener l) {
listSelectionModel.addChangeListener(l);
}

public void clearSelection() {
listSelectionModel.clearSelection();
}

public int getMaxSelectedIndex() {
return listSelectionModel.getMaxSelectedIndex();
}

public int getMinSelectedIndex() {
return listSelectionModel.getMinSelectedIndex();
}

public int getSelectionMode() {
return listSelectionModel.getSelectionMode();
}

public boolean isSelectedIndex(int index) {
return listSelectionModel.isSelectedIndex(index);
}

public boolean isSelectionEmpty() {
return listSelectionModel.isSelectionEmpty();
}

public void removeChangeListener(ChangeListener l) {
listSelectionModel.removeChangeListener(l);
}

public void setSelectedItem(Object item) {
if (getSelectionMode() == ListSelectionModel.SINGLE_SELECTION) {
clearSelection();
}
setSelectedIndex(indexOf(item), true);
}

public void setSelectedIndex(int index, boolean selected) {
listSelectionModel.setSelectedIndex(index, selected);
}

public void setSelectionMode(int selectionMode) {
listSelectionModel.setSelectionMode(selectionMode);
}
}

```

combinedtextlistmodel.java

```

/*
 * $Id: CombinedTextListModel.java,v 1.1 2008/03/18 10:25:44 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.Set;

import nextapp.echo2.app.event.DocumentListener;
import nextapp.echo2.app.event.ListDataListener;
import nextapp.echo2.app.list.DefaultListModel;
import nextapp.echo2.app.list.ListModel;
import nextapp.echo2.app.text.Document;

```

```

import nextapp.echo2.app.text.StringDocument;
import echopointng.text.StringDocumentEx;

/**
 * @author ron
 */
public class CombinedTextListModel implements Document, ListModel {
    static final long serialVersionUID = 0L;

    private final StringDocument document;
    private final DefaultListModel listModel;

    /**
     * @param defaultText
     * @param options
     */
    public CombinedTextListModel(Set<?> options, String defaultText) {
        this(new DefaultListModel(options.toArray()), new
StringDocumentEx(defaultText));
    }

    /**
     * @param document
     * @param listModel
     */
    public CombinedTextListModel(DefaultListModel listModel,
StringDocument document) {
        super();
        this.document = document;
        this.listModel = listModel;
    }

    /**
     * @see
     * @see nextapp.echo2.app.text.Document#addDocumentListener(nextapp.echo2.app.event.DocumentListener)
     */
    @Override
    public void addDocumentListener(DocumentListener l) {
        document.addDocumentListener(l);
    }

    /**
     * @see nextapp.echo2.app.text.Document#getText()
     */
    @Override
    public String getText() {
        return document.getText();
    }

    /**
     * @see
     * @see nextapp.echo2.app.text.Document#removeDocumentListener(nextapp.echo2.app.event.DocumentListener)
     */
    @Override
    public void removeDocumentListener(DocumentListener l) {
        document.removeDocumentListener(l);
    }

    /**
     * @see nextapp.echo2.app.list.ListModel#addListDataListener(nextapp.echo2.app.event.ListDataListener)
     */
    @Override
    public void addListDataListener(ListDataListener l) {
        listModel.addListDataListener(l);
    }

    /**
     * @see nextapp.echo2.app.list.ListModel#get(int)
     */
    @Override
    public Object get(int index) {
        return listModel.get(index);
    }

    /**
     * @see
     * @see nextapp.echo2.app.list.ListModel#removeListDataListener(nextapp.echo2.app.event.ListDataListener)
     */
}

```

```

@Override
public void removeListDataListener(ListDataListener l) {
    listModel.removeListDataListener(l);
}

/**
 * @see nextapp.echo2.app.list.ListModel#size()
 */
@Override
public int size() {
    return listModel.size();
}

}

comboboxmanipulator.java
```

```

/*
 * $Id: ComboBoxManipulator.java,v 1.5 2008/10/13 22:58:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.DocumentEvent;
import nextapp.echo2.app.event.DocumentListener;
import nextapp.echo2.app.list.ListModel;
import nextapp.echo2.app.text.Document;
import echopointng.ComboBox;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class ComboBoxManipulator extends AbstractComponentManipulator
{

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getComponent(component).getText());
    }

    public void setValue(Component component, Object value) {
        getComponent(component).setText((String) value);
    }
}
```

```

public void addListenerToDetectChangesToInput(finalEditMode
editMode, Component component) {
    final ComboBox comboBox = (ComboBox) component;
    comboBox.getTextField().getDocument().addDocumentListener(new
DocumentListener() {
    static final long serialVersionUID = 0L;

    public void documentUpdate(DocumentEvent e) {
        editMode.setStateEdited(true);
    }
});
}

@Override
public Document getModel(Component component) {
    return getComponent(component).getTextField().getDocument();
}

@Override
public void setModel(Component component, Object valueModel) {
    if (valueModel instanceof Document) {
        getComponent(component).getTextField().setDocument((Document)
valueModel);
    }
    if (valueModel instanceof ListModel) {
        getComponent(component).setListModel((ListModel) valueModel);
    }
}

private ComboBox getComponent(Component component) {
    return (ComboBox) component;
}
}
```

command.java

```

/*
 * $Id: Command.java,v 1.1 2008/12/13 00:40:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

/**
```

```

 * A command represent an activity that changes the state of the
application's
 * persistent state, such as adding or editing a user, project, or
element of a
 * project. A command encompasses an atomic unit of work such that all
the
 * changes made by the command succeed or fail in unison.<br>
 * A command may be a composite made up of multiple commands. In such a
case the
 * changes made by all the constituent commands either succeed or fail
as a
 * single unit.<br>
 * Commands may be stateful and an instance of a command will only be
used once
 * and discarded.
 *
 * @author ron
 */
public interface Command {

/**
 * The execute() method entails the unit of work of the command. The
 * execution will be transactional. If an exception is thrown from a
command
 * none of the work done by the command will be persisted. The state
of the
 * entity objects supplied to the command will be in the original
state when
 * supplied to the command.<br>
 * If the command completes successfully the new state of changed
entity
 * objects will be returned through the public getter methods.<br>
 * It is the responsibility of the executor of the command to
propagate the
 * changes to entities made by a command to the rest of the system
(such as
 * any user interface objects.)
 *
 * @throws Exception
*/
public void execute() throws Exception;
}

```

commandfactory.java

```

/*
 * $Id: CommandFactory.java,v 1.1 2008/12/13 00:41:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.command;

/**
 * A CommandFactory is used to get fresh instances of a command. Each
command
 * instance should only be used once and discarded.<br>
 * A CommandFactory has a strategy for creating new commands by
command type.
 *
 * @see CommandFactoryStrategy
 * @author ron
 */
public interface CommandFactory {

    public BatchCommand newBatchCommand();
}


```

commandfactorystrategy.java

```

/*
 * $Id: CommandFactoryStrategy.java,v 1.1 2008/12/13 00:40:58 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

/**
 * Each CommandFactory uses a strategy to create new command
instances. A
 * strategy may get commands from a container (such as the Spring IOC
container
 * or a J2EE container where the commands are stateful session beans)
or create
 * commands directly.
 *
 * @author ron
 */

```

```

public interface CommandFactoryStrategy {
    /**
     * Create a new instance of a command or return a "clean" instance of
     * a command where the state would be the same as if a new command were
     * returned.<br>
     * The returned command must be guaranteed to be uniquely referred to
     * by a single client at a time. A command will not be thread-safe.
     *
     * @param commandType -
     *          the type of command to create
     * @return a "clean" instance of a command
     */
    public Command newInstance(Class<? extends Command> commandType);
}

```

commandhandler.java

```

/*
 * $Id: CommandHandler.java,v 1.1 2008/12/13 00:40:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

/**
 * The context for the execution of a command. A handler demarcates
 * the transactionality of a command. A handler may execute the command
 * locally or
 * remotely transparently from the process that supplies the command
 * for
 * execution.
 *
 * @author ron
 */
public interface CommandHandler {

    /**
     * @param <T>
     *          The type of command
     * @param command -
     *          the command to execute and return on success.
     * @return a command that executed successfully.
     * @throws Exception
     */
}

```

```

    */
    public <T extends Command> T execute(T command) throws Exception;
}

```

componentmanipulator.java

```

/*
 * $Id: ComponentManipulator.java,v 1.7 2008/10/11 21:47:44 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * A standard interface for working with input components to set and
 * get the
 * model or value being edited.
 *
 * @author ron
 */
public interface ComponentManipulator {

    public void addListenerToDetectChangesToInput(finalEditMode
        editMode, Component component);

    public <T> T getValue(Component component, Class<T> type);

    public void setValue(Component component, Object value);

    public Object getModel(Component component);

    public void setModel(Component component, Object model);
}

```

componentmanipulators.java

```

/*
 * $Id: ComponentManipulators.java,v 1.2 2008/10/15 09:20:06 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import java.util.HashMap;
import java.util.Map;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.filetransfer.UploadSelect;
import nextapp.echo2.app.list.AbstractListComponent;
import nextapp.echo2.app.text.TextComponent;
import echopointng.ComboBox;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;

/**
 * @author ron
 */
public class ComponentManipulators {
    private static final Map<Class<? extends Component>, ComponentManipulator> componentManipulators = new HashMap<Class<? extends Component>, ComponentManipulator>();
    static {
        // add component manipulators for echo2 and echopointng components
        CollectionManipulator collectionManipulator = new CollectionManipulator();
        componentManipulators.put(TextComponent.class, new TextComponentManipulator());
        componentManipulators.put(Label.class, new LabelManipulator());
        componentManipulators.put(CheckBox.class, new CheckBoxManipulator());
        componentManipulators.put(AbstractListComponent.class, new AbstractListComponentManipulator());
        componentManipulators.put(ComboBox.class, new ComboBoxManipulator());
        componentManipulators.put(Row.class, collectionManipulator);
        componentManipulators.put(Column.class, collectionManipulator);
        componentManipulators.put(UploadSelect.class, new NullComponentManipulator());
        componentManipulators.put(Button.class, new NullComponentManipulator());
    }
}

/**
 * Each component should use this method in a static initializer
 * block to
 *     * add a manipulator for that component to the editor.
 *     *
 *     * @param componentType
 *     * @param manipulator
 *     * @see NavigatorTable for an example of using this method in a
 * static
 *     *      initializer.
 *     */
public static void setManipulator(Class<? extends Component> componentType,
ComponentManipulator manipulator) {
    componentManipulators.put(componentType, manipulator);
}

/**
 * Get the component manipulator for a specific component. If a
manipulator
 * doesn't exist for the component's type, walk up the inheritance
tree
 * until a manipulator is found.
 *
 * @param inputComponent
 * @return
 */
public static ComponentManipulator getManipulator(Component inputComponent) {
    ComponentManipulator manipulator = null;
    Class<?> componentType = inputComponent.getClass();
    while (manipulator == null) {
        manipulator = componentManipulators.get(componentType);
        if (manipulator != null) {
            return manipulator;
        } else {
            componentType = componentType.getSuperclass();
            if (Object.class.equals(componentType)) {
                break;
            }
        }
    }
    throw new RuntimeException("no manipulator for component type " +
inputComponent.getClass());
}
}

```

constituenttreedepthfinder.java

```
/*
 * $Id: ConstituentTreeDepthFinder.java,v 1.1 2009/03/22 11:08:21
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.NLPText;

/***
 * An NLPTextWalkerFunction that takes an NLPText and returns the
maximum depth
 * of the syntax structure.
 *
 * @author ron
 */
public class ConstituentTreeDepthFinder implements
NLPTextWalkerFunction<Integer> {

private int currentLevel = 0;
private int maxLevel = 0;

/***
 * Create a tree printer with an initial buffer size of 1000
characters and
 * pretty printing on that adds tabs, spaces, and new lines to make
the tree
 * easier to read.
 */
public ConstituentTreeDepthFinder() {
}

@Override
public void init() {
    currentLevel = 0;
    maxLevel = 0;
}

@Override
public void begin(NLPText text) {
    currentLevel++;
    if (currentLevel > maxLevel) {
        maxLevel++;
    }
}
```

```
}

@Override
public Integer end(NLPText t) {
    currentLevel--;
    return maxLevel;
}
}
```

constituenteprinter.java

```
/*
 * $Id: ConstituentTreePrinter.java,v 1.6 2009/01/26 10:19:00 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/***
 * An NLPTextWalkerFunction that takes an NLPText and prints out the
tree of the
 * phrase structure. For example:<br>
 *
 * <pre>
 * (ROOT
 *   (S
 *     (NP (DT The) (NN system#1))
 *     (VP (VBZ notifies#1)
 *       (NP
 *         (NP (DT the) (NNS stakeholders#1))
 *         (PP (IN of)
 *           (NP (NNS changes#4))))
 *         (PP (TO to)
 *           (NP (DT the) (NN project#1)))) (. .)))
 *   </pre>
 *
 * <p>
 * The tags, such as S, NP, VP, etc. are based on the Penn Treebank
tag set. The
 * (#) indicates the guessed word sense from WordNet.
 *
 * @see {@link ParseTag}

```

```

* @author ron
*/
public class ConstituentTreePrinter implements
NLPTextWalkerFunction<StringBuilder> {

private final boolean pretty;
private final int bufferSize;
private StringBuilder sb;
private int indent = 0;

/**
 * Create a tree printer with an initial buffer size of 1000
characters and
 * pretty printing on that adds tabs, spaces, and new lines to make
the tree
 * easier to read.
 */
public ConstituentTreePrinter() {
    this(1000);
}

/**
 * Create a tree printer with the specified initial buffer size and
pretty
 * printing on that adds tabs, spaces, and new lines to make the tree
easier
 * to read.
 *
 * @param bufferSize -
 *          the initial size of the buffer for building the tree.
 */
public ConstituentTreePrinter(int bufferSize) {
    this(bufferSize, true);
}

/**
 * Create a tree printer with the specified initial buffer size and
pretty
 * printing state.
 *
 * @param bufferSize -
 *          the initial size of the buffer for building the tree.
 * @param pretty -
 *          when true tabs, spaces, and new lines are added to make
the
 *          tree easier to read.
 */

```

```

public ConstituentTreePrinter(int bufferSize, boolean pretty) {
    this.bufferSize = bufferSize;
    this.pretty = pretty;
}

@Override
public void init() {
    sb = new StringBuilder(bufferSize);
}

@Override
public void begin(NLPText text) {
    if (pretty) {
        if (!
GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLeve
l())) {
            sb.append("\n");
            for (int i = 0; i < indent; i++) {
                sb.append("\t");
            }
        } else {
            sb.append(" ");
        }
    }
    sb.append("(");
    sb.append(text.getParseTag().getText());
    if
(GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLev
el())) {
        String word = text.getText();
        sb.append(" ");
        if ((word != null) && (word.length() > 0)) {
            sb.append(word);
        }
        if (text.getDictionaryWordSense() != null) {
            sb.append("#");
            sb.append(text.getDictionaryWordSense().getRank());
        }
    }
    if (pretty && !
GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLeve
l())) {
        indent++;
    }
}

@Override

```

```

public StringBuilder end(NLPText t) {
    sb.append(")");
    if (pretty && !GrammaticalStructureLevel.WORD.equals(t.getGrammaticalStructureLevel()))
    {
        indent--;
    }
    return sb;
}

```

constraintviolationexceptionadapter.java

```

/*
 * $Id: ConstraintViolationExceptionAdapter.java,v 1.1 2008/12/13
00:41:15 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository.jpa;

import org.hibernate.exception.ConstraintViolationException;

import com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;

/**
 * @author ron
 */
public class ConstraintViolationExceptionAdapter implements
EntityExceptionAdapter {

    private final String propertyName;

    /**
     * @param propertyName -
     *          name of the property that must be unique.
     */
    public ConstraintViolationExceptionAdapter(String propertyName) {
        this.propertyName = propertyName;
    }
}

```

```

/**
 * @see
edu.harvard.fas.rregan.repository.EntityExceptionAdapter#convert(java.
lang.Throwable,
                    java.lang.Object,
                    edu.harvard.fas.rregan.repository.EntityExceptionActionType)
*/
@Override
public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
                                EntityExceptionActionType actionType) {
    if (entityType == null) {
        if (entity == null ) {
            ConstraintViolationException cve = (ConstraintViolationException)
original;
            if
(MySQLIntegrityConstraintViolationException.class.equals(cve.getCause(
).getClassName())) {
                MySQLIntegrityConstraintViolationException icve =
(MySQLIntegrityConstraintViolationException) cve.getCause();
                return EntityException.uniquenessConflict(icve,
icve.getMessage());
            }
        }
    }
    return EntityException.uniquenessConflict(entityType, entity,
propertyName, actionType);
}
}

```

controller.java

```

/*
 * $Id: Controller.java,v 1.1 2008/02/20 11:36:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.controller;

import java.util.Set;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

/**
 */

```

```

 * A Command encapsulates a function in the system that is invoked
through event
 * passing via the EventDispatcher and optionally responds with an
event.
 *
 * @author ron
 */
public interface Controller extends ActionListener {

    /**
     * @return a Set of the types of events this command handles
    */
    public Set<Class<? extends ActionEvent>> getEventTypesToListenFor();

    /**
     * Add an ActionListener that listens for events generated by this
Command
     * when a service method is invoked. This is primarily for the
     * EventDispatcher to listen for events to forward to other
listeners.<br/>NOTE:
     * it is the responsibility of the actionPerformed to remove itself
when it
     * is no longer interested in the events. Not doing so could result
in the
     * listener remaining in memory.
     *
     * @param actionPerformed
    */
    public void addActionListener(ActionListener actionPerformed);

    /**
     * Remove an ActionListener currently listening for events.
     *
     * @param actionPerformed
    */
    public void removeActionListener(ActionListener actionPerformed);
}

```

convertsteptoscenariocommand.java

```

/*
 * $Id: ConvertStepToScenarioCommand.java,v 1.1 2008/10/16 06:41:49
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.project.Step;

/**
 * Given a scenario step, convert the step into a scenario updating
all the
 * scenarios the reference the step.
 *
 * @author ron
 */
public interface ConvertStepToScenarioCommand extends
EditScenarioCommand {

    /**
     * @param step -
     *          the Scenario Step to copy.
     */
    public void setOriginalScenarioStep(Step step);
}

```

convertsteptoscenariocommandimpl.java

```

/*
 * $Id: ConvertStepToScenarioCommandImpl.java,v 1.6 2009/03/30
11:54:28 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;

```

```

import
edu.harvard.fas.rregan.requel.project.command.ConvertStepToScenarioCom
mand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
public class ConvertStepToScenarioCommandImpl extends
EditScenarioCommandImpl implements
ConvertStepToScenarioCommand {
    Step originalStep;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param commandHandler
     */
    @Autowired
    public ConvertStepToScenarioCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
            annotationCommandFactory, commandHandler);
    }

    @Override
    public void setOriginalScenarioStep(Step originalStep) {
        this.originalStep = originalStep;
    }

    protected Step getOriginalScenarioStep() {
        return originalStep;
    }
}

```

```

@Override
public void execute() throws Exception {
    Map<Scenario, Integer> stepInScenarioMap = new HashMap<Scenario,
Integer>();
    Set<Annotation> stepAnnotations = new HashSet<Annotation>();
    if (getOriginalScenarioStep() != null) {
        // remove the original step from scenarios and annotations
        // and save them so they can be added to the new scenario and
        // vice versa
        for (Scenario scenario :
getOriginalScenarioStep().getUsingScenarios()) {
            scenario = getRepository().get(scenario);
            for (int index = 0; index < scenario.getSteps().size(); index++) {
                if
(getOriginalScenarioStep().equals(scenario.getSteps().get(index))) {
                    stepInScenarioMap.put(scenario, index);
                    scenario.getSteps().remove(index);
                    break;
                }
            }
        }
        for (Annotation annotation :
getOriginalScenarioStep().getAnnotations()) {
            annotation = getRepository().get(annotation);
            annotation.getAnnotatables().remove(getOriginalScenarioStep());
            stepAnnotations.add(annotation);
        }
        getRepository().delete(getOriginalScenarioStep());
        getRepository().flush();
    }
    // use the super-command to create the new scenario
    super.execute();
    Scenario newScenario = getScenario();

    // TODO: this assume the scenarios that refered to the step haven't
    // changed since this command started executing
    for (Scenario scenario : stepInScenarioMap.keySet()) {
        scenario = getRepository().get(scenario);
        Integer index = stepInScenarioMap.get(scenario);
        if (index >= scenario.getSteps().size()) {
            scenario.getSteps().add(newScenario);
        } else {
            scenario.getSteps().add(index, newScenario);
        }
    }
    for (Annotation annotation : stepAnnotations) {

```

```

annotation.getAnnotatables().add(newScenario);
newScenario.getAnnotations().add(annotation);
}
}
}

```

copyactorcommand.java

```

/*
 * $Id: CopyActorCommand.java,v 1.1 2008/09/26 01:35:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Actor;

/**
 * Given a actor, create a new actor copying references to actors and
goals.
 * If a name isn't supplied a unique name is generated.
 *
 * @author ron
 */
public interface CopyActorCommand extends EditCommand {

    /**
     * @param Actor -
     *          the Actor to copy.
     */
    public void setOriginalActor(Actor actor);

    /**
     * @param newName -
     *          the name for the new actor. if this is not set, or is
set to
     *          a name already in use, a unique name will be generated.
     */
    public void setNewActorName(String newName);

    /**
     * @return the new copy of the use case.
     */
    public Actor getNewActor();
}

```

copyactorcommandimpl.java

```

/*
 * $Id: CopyActorCommandImpl.java,v 1.7 2009/03/30 11:54:28 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.CopyActorCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ActorImpl;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyActorCommand")
@Scope("prototype")
public class CopyActorCommandImpl extends AbstractEditProjectCommand
implements CopyActorCommand {

    private Actor originalActor;
    private Actor newActor;
    private String newActorName;

    /**
     * @param assistantManager
     */

```

```

* @param userRepository
* @param projectRepository
* @param projectCommandFactory
* @param annotationCommandFactory
* @param commandHandler
*/
@.Autowired
public CopyActorCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

protected Actor getOriginalActor() {
    return originalActor;
}

public void setOriginalActor(Actor originalActor) {
    this.originalActor = originalActor;
}

protected String getNewActorName() {
    return newActorName;
}

public void setNewActorName(String newActorName) {
    this.newActorName = newActorName;
}

@Override
public Actor getNewActor() {
    return newActor;
}

protected void setNewActor(Actor newActor) {
    this.newActor = newActor;
}

@Override
public void execute() {
    User editedBy = getRepository().get(getEditedBy());
}

```

```

ActorImpl originalActor = (ActorImpl)
getRepository().get(getOriginalActor());

String newName;
if ((getNewActorName() == null) || (getNewActorName().length() ==
0)) {
    newName = generateNewActorName(originalActor.getName());
} else {
    newName = generateNewActorName(getNewActorName());
}
ActorImpl newActor = getProjectRepository().persist(
    new ActorImpl(originalActor.getProjectOrDomain(), editedBy,
newName, originalActor
    .getText()));
for (Goal goal : originalActor.getGoals()) {
    newActor.getGoals().add(goal);
    goal.getReferers().add(newActor);
}
// TODO: what other references should be added?

// TODO: this assumes that all annotations are appropriate for the
new
// actor
for (Annotation annotation : originalActor.getAnnotations()) {
    newActor.getAnnotations().add(annotation);
    annotation.getAnnotatables().add(newActor);
}
for (GlossaryTerm term : originalActor.getGlossaryTerms()) {
    newActor.getGlossaryTerms().add(term);
    term.getReferers().add(newActor);
}
newActor = getProjectRepository().merge(newActor);
setNewActor(newActor);
}

private String generateNewActorName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findActorByProjectOrDomainAndName(
            getOriginalActor().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findActorByProjectOrDomainAndName(
                    getOriginalActor().getProjectOrDomain(), newName);
                counter++;
            }
        }
    }
}

```

```

        newName = originalName + " " + counter;
    } catch (EntityException e) {
        // new name is not in use
        break;
    }
}
} catch (EntityException e) {
    // new name is not in use
}
return newName;
}
}

```

copygoalcommand.java

```

/*
 * $Id: CopyGoalCommand.java,v 1.1 2008/09/26 01:35:04 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Goal;

/**
 * Given a goal, create a new goal copying references to actors and
goals.
 * If a name isn't supplied a unique name is generated.
 *
 * @author ron
 */
public interface CopyGoalCommand extends EditCommand {

    /**
     * @param Goal -
     *          the Goal to copy.
     */
    public void setOriginalGoal(Goal goal);

    /**
     * @param newName -
     *          the name for the new goal. if this is not set, or is
set to
     *          a name already in use, a unique name will be generated.
     */
}

```

```

    public void setNewGoalName(String newName);

    /**
     * @return the new copy of the use case.
     */
    public Goal getNewGoal();
}

```

copygoalcommandimpl.java

```

/*
 * $Id: CopyGoalCommandImpl.java,v 1.6 2009/03/30 11:54:30 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.CopyGoalCommand;
import edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.GoalImpl;
import edu.harvard.fas.rregan.requel.project.impl.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyGoalCommand")

```

```

@Scope("prototype")
public class CopyGoalCommandImpl extends AbstractEditProjectCommand
implements CopyGoalCommand {

    private Goal originalGoal;
    private Goal newGoal;
    private String newGoalName;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public CopyGoalCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository,
        ProjectRepository projectRepository, ProjectCommandFactory
        projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    protected Goal getOriginalGoal() {
        return originalGoal;
    }

    public void setOriginalGoal(Goal originalGoal) {
        this.originalGoal = originalGoal;
    }

    protected String getNewGoalName() {
        return newGoalName;
    }

    public void setNewGoalName(String newGoalName) {
        this.newGoalName = newGoalName;
    }

    @Override
    public Goal getNewGoal() {
        return newGoal;
    }

    }

    protected void setNewGoal(Goal newGoal) {
        this.newGoal = newGoal;
    }

    @Override
    public void execute() throws Exception {
        User editedBy = getRepository().get(getEditedBy());
        GoalImpl originalGoal = (GoalImpl)
        getRepository().get(getOriginalGoal());

        String newName;
        if ((getNewGoalName() == null) || (getNewGoalName().length() == 0))
        {
            newName = generateNewGoalName(originalGoal.getName());
        } else {
            newName = generateNewGoalName(getNewGoalName());
        }
        GoalImpl newGoal = getProjectRepository().persist(
            new GoalImpl(originalGoal.getProjectOrDomain(), editedBy, newName,
            originalGoal
                .getText()));

        for (GoalRelation goalRelation :
        originalGoal.getRelationsFromThisGoal()) {
            EditGoalRelationCommand editGoalRelationCommand =
            getProjectCommandFactory()
                .newEditGoalRelationCommand();
            editGoalRelationCommand.setEditedBy(editedBy);
            editGoalRelationCommand.setFromGoal(newGoal.getName());
            editGoalRelationCommand.setToGoal(goalRelation.getToGoal().getName(
            ));
            editGoalRelationCommand.setProjectOrDomain(newGoal.getProjectOrDoma
            in());
            editGoalRelationCommand.setRelationType(goalRelation.getRelationTyp
            e().name());
            editGoalRelationCommand =
            getCommandHandler().execute(editGoalRelationCommand);
            newGoal.getRelationsFromThisGoal().add(editGoalRelationCommand.getG
            oalRelation());
        }
        // TODO: this assumes that all annotations are appropriate for the
        new
        // goal
        for (Annotation annotation : originalGoal.getAnnotations()) {
            newGoal.getAnnotations().add(annotation);
        }
    }
}

```

```

annotation.getAnnotatables().add(newGoal);
}
for (GlossaryTerm term : originalGoal.getGlossaryTerms()) {
    newGoal.getGlossaryTerms().add(term);
    term.getReferers().add(newGoal);
}
newGoal = getProjectRepository().merge(newGoal);
setNewGoal(newGoal);
}

private String generateNewGoalName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findGoalByProjectOrDomainAndName(
            getOriginalGoal().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findGoalByProjectOrDomainAndName(
                    getOriginalGoal().getProjectOrDomain(), newName);
                counter++;
                newName = originalName + " " + counter;
            } catch (EntityException e) {
                // new name is not in use
                break;
            }
        }
    } catch (EntityException e) {
        // new name is not in use
    }
    return newName;
}
}

```

copyscenariocommand.java

```

/*
 * $Id: CopyScenarioCommand.java,v 1.2 2008/10/16 06:41:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;

```

```

import edu.harvard.fas.rregan.requel.project.Scenario;

/**
 * Given a scenario, create a new scenario copying references to other
 * entities.
 * If a name isn't supplied a unique name is generated.
 *
 * @author ron
 */
public interface CopyScenarioCommand extends EditCommand {

    /**
     * @param scenario -
     *          the Scenario to copy.
     */
    public void setOriginalScenario(Scenario scenario);

    /**
     * @param newName -
     *          the name for the new scenario. if this is not set, or
     * is set
     *          to a name already in use, a unique name will be
     * generated.
     */
    public void setNewScenarioName(String newName);

    /**
     * @return the new copy of the scenario.
     */
    public Scenario getNewScenario();
}

```

copyscenariocommandimpl.java

```

/*
 * $Id: CopyScenarioCommandImpl.java,v 1.5 2009/03/30 11:54:27 rregan
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;

```

```

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Scenario;
import
edu.harvard.fas.rregan.requel.project.command.CopyScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ScenarioImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyScenarioCommand")
@Scope("prototype")
public class CopyScenarioCommandImpl extends
AbstractEditProjectCommand implements
CopyScenarioCommand {

    private Scenario originalScenario;
    private Scenario newScenario;
    private String newScenarioName;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     */
    @Autowired
    public CopyScenarioCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }
}

```

```

protected Scenario getOriginalScenario() {
    return originalScenario;
}

public void setOriginalScenario(Scenario originalScenario) {
    this.originalScenario = originalScenario;
}

protected String getNewScenarioName() {
    return newScenarioName;
}

public void setNewScenarioName(String newScenarioName) {
    this.newScenarioName = newScenarioName;
}

@Override
public Scenario getNewScenario() {
    return newScenario;
}

protected void setNewScenario(Scenario newScenario) {
    this.newScenario = newScenario;
}

@Override
public void execute() {
    User editedBy = getRepository().get(getEditedBy());
    ScenarioImpl originalScenario = (ScenarioImpl)
getRepository().get(getOriginalScenario());

    String newName;
    if ((getNewScenarioName() == null) || (getNewScenarioName().length()
== 0)) {
        newName = generateNewScenarioName(originalScenario.getName());
    } else {
        newName = generateNewScenarioName(getNewScenarioName());
    }
    ScenarioImpl newScenario = getProjectRepository().persist(
        new ScenarioImpl(originalScenario.getProjectOrDomain(), editedBy,
newName,
            originalScenario.getText(), originalScenario.getType()));

    // TODO: this assumes that all annotations are appropriate for the
new
    // scenario
    for (Annotation annotation : originalScenario.getAnnotations()) {

```

```

newScenario.getAnnotations().add(annotation);
annotation.getAnnotatables().add(newScenario);
}
for (GlossaryTerm term : originalScenario.getGlossaryTerms()) {
    newScenario.getGlossaryTerms().add(term);
    term.getReferers().add(newScenario);
}
newScenario = getProjectRepository().merge(newScenario);
setNewScenario(newScenario);
}

private String generateNewScenarioName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findScenarioByProjectOrDomainAndName(
            getOriginalScenario().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findScenarioByProjectOrDomainAndName(
                    getOriginalScenario().getProjectOrDomain(), newName);
                counter++;
                newName = originalName + " " + counter;
            } catch (EntityException e) {
                // new name is not in use
                break;
            }
        }
    } catch (EntityException e) {
        // new name is not in use
    }
    return newName;
}

```

copyscenariostepcommand.java

```

/*
 * $Id: CopyScenarioStepCommand.java,v 1.1 2008/10/16 06:41:49 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

```

```

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Step;

/**
 * Given a scenario step, create a new step copying references to
 * other entites.
 * If a name isn't supplied a unique name is generated.
 */
@Author ron
*/
public interface CopyScenarioStepCommand extends EditCommand {

    /**
     * @param step -
     *          the Scenario Step to copy.
     */
    public void setOriginalScenarioStep(Step step);

    /**
     * @param newName -
     *          the name for the new scenario step. if this is not set,
     * or is
     *          set to a name already in use, a unique name will be
     * generated.
     */
    public void setNewScenarioStepName(String newName);

    /**
     * @return the new copy of the step.
     */
    public Step getNewScenarioStep();
}

```

copyscenariostepcommandimpl.java

```

/*
 * $Id: CopyScenarioStepCommandImpl.java,v 1.5 2009/03/30 11:54:28
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

```

```

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.requel.project.command.CopyScenarioStepCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.StepImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyScenarioStepCommand")
@Scope("prototype")
public class CopyScenarioStepCommandImpl extends
AbstractEditProjectCommand implements
CopyScenarioStepCommand {

    private Step originalScenarioStep;
    private Step newScenarioStep;
    private String newScenarioStepName;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param commandHandler
     */
    @Autowired
    public CopyScenarioStepCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,

```

```

annotationCommandFactory, commandHandler);
    }

    protected Step getOriginalScenarioStep() {
        return originalScenarioStep;
    }

    public void setOriginalScenarioStep(Step originalScenarioStep) {
        this.originalScenarioStep = originalScenarioStep;
    }

    protected String getNewScenarioStepName() {
        return newScenarioStepName;
    }

    public void setNewScenarioStepName(String newScenarioStepName) {
        this.newScenarioStepName = newScenarioStepName;
    }

    @Override
    public Step getNewScenarioStep() {
        return newScenarioStep;
    }

    protected void setNewScenarioStep(Step newScenarioStep) {
        this.newScenarioStep = newScenarioStep;
    }

    @Override
    public void execute() {
        User editedBy = getRepository().get(getEditedBy());
        StepImpl originalScenarioStep = (StepImpl)
getRepository().get(getOriginalScenarioStep());

        String newName;
        if ((getNewScenarioStepName() == null) ||
(getNewScenarioStepName().length() == 0)) {
            newName =
generateNewScenarioStepName(originalScenarioStep.getName());
        } else {
            newName = generateNewScenarioStepName(getNewScenarioStepName());
        }
        StepImpl newScenarioStep = getProjectRepository().persist(
            new StepImpl(originalScenarioStep.getProjectOrDomain(), editedBy,
newName,
            originalScenarioStep.getText(),
            originalScenarioStep.getType()));
    }
}

```

```

// TODO: this assumes that all annotations are appropriate for the
new
// scenario
for (Annotation annotation : originalScenarioStep.getAnnotations())
{
    newScenarioStep.getAnnotations().add(annotation);
    annotation.getAnnotatables().add(newScenarioStep);
}
for (GlossaryTerm term : originalScenarioStep.getGlossaryTerms()) {
    newScenarioStep.getGlossaryTerms().add(term);
    term.getReferers().add(newScenarioStep);
}
newScenarioStep = getProjectRepository().merge(newScenarioStep);
setNewScenarioStep(newScenarioStep);
}

private String generateNewScenarioStepName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findScenarioByProjectOrDomainAndName(
            getOriginalScenarioStep().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findScenarioByProjectOrDomainAndName(
                    getOriginalScenarioStep().getProjectOrDomain(), newName);
                counter++;
                newName = originalName + " " + counter;
            } catch (EntityException e) {
                // new name is not in use
                break;
            }
        }
    } catch (EntityException e) {
        // new name is not in use
    }
    return newName;
}

```

copystorycommand.java

```

/*
 * $Id: CopyStoryCommand.java,v 1.1 2008/09/26 01:35:03 rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Story;

/**
 * Given a story, create a new story copying references to actors and
goals.
 * If a name isn't supplied a unique name is generated.
 *
 * @author ron
 */
public interface CopyStoryCommand extends EditCommand {

    /**
     * @param Story -
     *          the Story to copy.
     */
    public void setOriginalStory(Story story);

    /**
     * @param newName -
     *          the name for the new story. if this is not set, or is
set to
     *          a name already in use, a unique name will be generated.
     */
    public void setNewStoryName(String newName);

    /**
     * @return the new copy of the use case.
     */
    public Story getNewStory();
}

```

copystorycommandimpl.java

```

/*
 * $Id: CopyStoryCommandImpl.java,v 1.6 2009/03/30 11:54:26 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.command.CopyStoryCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.StoryImpl;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyStoryCommand")
@Scope("prototype")
public class CopyStoryCommandImpl extends AbstractEditProjectCommand
implements CopyStoryCommand {

    private Story originalStory;
    private Story newStory;
    private String newStoryName;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     */
    @Autowired
    public CopyStoryCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository,
        ProjectRepository projectRepository, ProjectCommandFactory
        projectCommandFactory,

```

```

        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
    super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
}

protected Story getOriginalStory() {
    return originalStory;
}

public void setOriginalStory(Story originalStory) {
    this.originalStory = originalStory;
}

protected String getNewStoryName() {
    return newStoryName;
}

public void setNewStoryName(String newStoryName) {
    this.newStoryName = newStoryName;
}

@Override
public Story getNewStory() {
    return newStory;
}

protected void setNewStory(Story newStory) {
    this.newStory = newStory;
}

@Override
public void execute() {
    User editedBy = getRepository().get(getEditedBy());
    StoryImpl originalStory = (StoryImpl)
    getRepository().get(getOriginalStory());

    String newName;
    if ((getNewStoryName() == null) || (getNewStoryName().length() ==
0)) {
        newName = generateNewStoryName(originalStory.getName());
    } else {
        newName = generateNewStoryName(getNewStoryName());
    }
    StoryImpl newStory = getProjectRepository().persist(

```

```

        new StoryImpl(originalStory.getProjectOrDomain(), editedBy,
newName, originalStory
        .getText(), originalStory.getStoryType()));
for (Actor actor : originalStory.getActors()) {
    newStory.getActors().add(actor);
    actor.getReferers().add(newStory);
}
for (Goal goal : originalStory.getGoals()) {
    newStory.getGoals().add(goal);
    goal.getReferers().add(newStory);
}
// TODO: this assumes that all annotations are appropriate for the
new
// story
for (Annotation annotation : originalStory.getAnnotations()) {
    newStory.getAnnotations().add(annotation);
    annotation.getAnnotatables().add(newStory);
}
for (GlossaryTerm term : originalStory.getGlossaryTerms()) {
    newStory.getGlossaryTerms().add(term);
    term.getReferers().add(newStory);
}
newStory = getProjectRepository().merge(newStory);
setNewStory(newStory);
}

private String generateNewStoryName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findStoryByProjectOrDomainAndName(
            getOriginalStory().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findStoryByProjectOrDomainAndName(
                    getOriginalStory().getProjectOrDomain(), newName);
                counter++;
                newName = originalName + " " + counter;
            } catch (EntityException e) {
                // new name is not in use
                break;
            }
        }
    } catch (EntityException e) {
        // new name is not in use
    }
}

```

```

        return newName;
    }
}

```

copyusecasecommand.java

```

/*
 * $Id: CopyUseCaseCommand.java,v 1.2 2008/09/26 01:35:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.UseCase;

/**
 * Given a use case, create a new use case copying references to
actors, goals
 * and stories. If a name isn't supplied a unique name is generated.
 *
 * @author ron
 */
public interface CopyUseCaseCommand extends EditCommand {

    /**
     * @param usecase -
     *          the usecase to copy.
     */
    public void setOriginalUseCase(UseCase usecase);

    /**
     * @param newName -
     *          the name for the new usecase. if this is not set, or is
set to
     *          a name already in use, a unique name will be generated.
     */
    public void setNewUseCaseName(String newName);

    /**
     * @return the new copy of the use case.
     */
    public UseCase getNewUseCase();
}

```

copyusecasecommandimpl.java

```
/*
 * $Id: CopyUseCaseCommandImpl.java,v 1.8 2009/03/30 11:54:27 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.command.CopyScenarioCommand;
import edu.harvard.fas.rregan.requel.project.command.CopyUseCaseCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.UseCaseImpl;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("copyUseCaseCommand")
@Scope("prototype")
public class CopyUseCaseCommandImpl extends AbstractEditProjectCommand
implements
CopyUseCaseCommand {
```

```
private UseCase originalUseCase;
private UseCase newUseCase;
private String newUseCaseName;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@.Autowired
public CopyUseCaseCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

protected UseCase getOriginalUseCase() {
return originalUseCase;
}

public void setOriginalUseCase(UseCase originalUseCase) {
this.originalUseCase = originalUseCase;
}

protected String getNewUseCaseName() {
return newUseCaseName;
}

public void setNewUseCaseName(String newUseCaseName) {
this.newUseCaseName = newUseCaseName;
}

@Override
public UseCase getNewUseCase() {
return newUseCase;
}

protected void setNewUseCase(UseCase newUseCase) {
this.newUseCase = newUseCase;
}

@Override
```

```

public void execute() throws Exception {
    User editedBy = getRepository().get(getEditedBy());
    UseCaseImpl originalUseCase = (UseCaseImpl)
        getRepository().get(getOriginalUseCase());

    String newName;
    if ((getNewUseCaseName() == null) || (getNewUseCaseName().length()
        == 0)) {
        newName = generateNewUseCaseName(originalUseCase.getName());
    } else {
        newName = generateNewUseCaseName(getNewUseCaseName());
    }
    CopyScenarioCommand copyScenarioCommand = getProjectCommandFactory()
        .newCopyScenarioCommand();
    copyScenarioCommand.setEditedBy(editedBy);
    copyScenarioCommand.setOriginalScenario(originalUseCase.getScenario());
    getCommandHandler().execute(copyScenarioCommand);

    UseCaseImpl newUseCase = getProjectRepository().persist(
        new UseCaseImpl(originalUseCase.getProjectOrDomain(),
        originalUseCase
            .getPrimaryActor(), editedBy, newName,
        originalUseCase.getText(),
        copyScenarioCommand.getNewScenario()));
    for (Actor actor : originalUseCase.getActors()) {
        newUseCase.getActors().add(actor);
        actor.getReferers().add(newUseCase);
    }
    for (Story story : originalUseCase.getStories()) {
        newUseCase.getStories().add(story);
        story.getReferers().add(newUseCase);
    }
    for (Goal goal : originalUseCase.getGoals()) {
        newUseCase.getGoals().add(goal);
        goal.getReferers().add(newUseCase);
    }
    // TODO: this assumes that all annotations are appropriate for the
    new
    // use case
    for (Annotation annotation : originalUseCase.getAnnotations()) {
        newUseCase.getAnnotations().add(annotation);
        annotation.getAnnotatables().add(newUseCase);
    }
    for (GlossaryTerm term : originalUseCase.getGlossaryTerms()) {
        newUseCase.getGlossaryTerms().add(term);
        term.getReferers().add(newUseCase);
    }
}

```

```

    }
    newUseCase = getProjectRepository().merge(newUseCase);
    setNewUseCase(newUseCase);
}

private String generateNewUseCaseName(String originalName) {
    String newName = originalName;
    try {
        getProjectRepository().findUseCaseByProjectOrDomainAndName(
            getOriginalUseCase().getProjectOrDomain(), newName);
        int counter = 1;
        newName = originalName + " " + counter;
        while (true) {
            try {
                getProjectRepository().findUseCaseByProjectOrDomainAndName(
                    getOriginalUseCase().getProjectOrDomain(), newName);
                counter++;
                newName = originalName + " " + counter;
            } catch (EntityException e) {
                // new name is not in use
                break;
            }
        }
    } catch (EntityException e) {
        // new name is not in use
    }
    return newName;
}
}

```

createdentity.java

```

/*
 * $Id: CreatedEntity.java,v 1.1 2008/04/09 10:33:30 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel;

import java.util.Date;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public interface CreatedEntity {
}

```

```

/**
 * @return the user that created the entity
 */
public User getCreatedBy();

/**
 * @return the date the entity was created
 */
public Date getDateCreated();
}

```

databasecreationlistener.java

```

/*
 * $Id: DatabaseCreationListener.java,v 1.5 2008/06/27 09:15:54 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel;

import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.DriverPropertyInfo;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;

/**
 * This listener detects if the database specified in the
db.properties file
 * exists, and if it doesn't it creates it. <br/> NOTE: This listener
must be
 * registered before all other listeners that use the database.
 * @sequence.diagram
 * tomcat:Tomcat[a]
 * /dcl:DatabaseCreationListener[a]
 * /dbProperties:File

```

```

* dm:DriverManager[a]
* connection:Connection
* createDbStmt:Statement
*
* tomcat:dcl.new
* tomcat:dcl.contextInitialized()
* dcl:dbProperties.new
* dcl:dm.getConnection(dbUrl)
* [c:exception no database]
* dcl:dm.getConnection(dbUrl)
* dcl:connection.createStatement()
* dcl:createDbStmt.execute("create database")
* [/c]
*
* @author ron
*/
public class DatabaseCreationListener implements
ServletContextListener {
private static final Logger log =
Logger.getLogger(DatabaseCreationListener.class);

public DatabaseCreationListener() {
super();
}

public void contextDestroyed(ServletContextEvent event) {
}

public void contextInitialized(ServletContextEvent event) {
Properties dbProperties = new Properties();
try {
File webInfDirectory = new File(event.getServletContext()
.getRealPath("WEB-INF/classes"));
File dbPropertiesFile = new File(webInfDirectory, "db.properties");
FileInputStream fis = new FileInputStream(dbPropertiesFile);
dbProperties.load(fis);
fis.close();

// make sure the driver is loaded
Class.forName(dbProperties.getProperty("db.driver"));

// create the full url from the properties
StringBuilder jdbcUrlBuilder = new StringBuilder();
jdbcUrlBuilder.append(dbProperties.getProperty("db.baseUrl"));
jdbcUrlBuilder.append(dbProperties.getProperty("db.server"));
jdbcUrlBuilder.append(":");
jdbcUrlBuilder.append(dbProperties.getProperty("db.port"));

```

```

jdbcUrlBuilder.append("/");
// connection string without database
String jdbcUrlNoDatabase = jdbcUrlBuilder.toString();

jdbcUrlBuilder.append(dbProperties.getProperty("db.name"));
jdbcUrlBuilder.append(dbProperties.getProperty("db.urlParams"));

String jdbcUrl = jdbcUrlBuilder.toString();
if (log.isDebugEnabled()) {
    log.debug("jdbc url = " + jdbcUrl);
    listDriverOptions(jdbcUrlNoDatabase);
}

try {
    // try to connect to the database to see if it already exists
    DriverManager.getConnection(jdbcUrl,
        dbProperties.getProperty("db.username"),
        dbProperties.getProperty("db.password"));
} catch (SQLException se) {
    // TODO: check the exception to see if it really was thrown
    // because the database doesn't exist
    // create the database
    Connection con = DriverManager.getConnection(jdbcUrlNoDatabase,
        dbProperties
            .getProperty("db.username"),
        dbProperties.getProperty("db.password"));
    Statement createDbStmt = con.createStatement();
    createDbStmt.execute("create database " +
        dbProperties.getProperty("db.name"));
}
} catch (ClassNotFoundException e) {
    // TODO: throw an exception, the app won't be available
    log.warn("cound not create database '" +
        dbProperties.getProperty("db.name")
        + "', the driver class '" + dbProperties.getProperty("db.driver")
        + "' in db.properties could not be loaded.", e);
} catch (Exception e) {
    // TODO: throw an exception, the app won't be available
    if (dbProperties.getProperty("db.name") == null) {
        log.warn(
            "could not create database, the properties in db.properties
        could not be loaded: "
            + e, e);
    } else {
        log.warn("could not create database '" +
            dbProperties.getProperty("db.name"))
    }
}

```

```

        + "'": " + e, e);
    }
}

/***
 * @param driverName
 * @param jdbcUrl
 * @see http://exampledepot.com/egs/java.sql/GetPropInfo.html
 */
private String listDriverOptions(String jdbcUrl) throws Exception {
    // Get the Driver instance
    Driver driver = DriverManager.getDriver(jdbcUrl);

    // Get available properties
    DriverPropertyInfo[] info = driver.getPropertyInfo(jdbcUrl, null);
    StringBuilder description = new StringBuilder();

    for (int i = 0; i < info.length; i++) {
        description.append(info[i].name);
        description.append(" ");
        description.append(info[i].description);
        description.append(" ");
        description.append(info[i].required);
        description.append(" ");
        description.append(info[i].value);
        description.append(" ");
        description.append(info[i].choices);
        description.append(System.getProperty("line.separator"));
    }
    return description.toString();
}

```

databaseinitializationlistener.java

```

/*
 * $Id: DatabaseInitializationListener.java,v 1.9 2008/12/13 00:41:40
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

```

```

import org.springframework.web.context.WebApplicationContext;
import
org.springframework.web.context.support.WebApplicationContextUtils;

import edu.harvard.fas.rregan.repository.DatabaseInitializer;

/**
 * This listener checks if the database has been initialized with the
base data,
 * and if not it initializes it. <br/> NOTE: This listener must be
registered
 * after the Spring context config.
 *
 * @author ron
 */
public class DatabaseInitializationListener implements
ServletContextListener {

    public void contextDestroyed(ServletContextEvent event) {
    }

    public void contextInitialized(ServletContextEvent event) {
        WebApplicationContext ctx =
WebApplicationContextUtils.getWebApplicationContext(event
        .getServletContext());
        DatabaseInitializer initializer = (DatabaseInitializer)
ctx.getAutowireCapableBeanFactory().getBean("databaseInitializer");
        initializer.initialize();
    }
}

```

databaseinitializer.java

```

/*
 * $Id: DatabaseInitializer.java,v 1.2 2009/01/26 10:19:05 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository;

import java.util.Set;
import java.util.TreeSet;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.SystemInitializer;

/**
 * @author ron
 */
@Component("databaseInitializer")
@Scope("prototype")
public class DatabaseInitializer {

    private final Set<SystemInitializer> entityInitializers;

    /**
     * Create a database initializer with a set of entity initializers
that it
     * will call.
     *
     * @param entityInitializers
     */
    @Autowired
    public DatabaseInitializer(Set<SystemInitializer> entityInitializers)
{
    this.entityInitializers = new
TreeSet<SystemInitializer>(entityInitializers);
}

    /**
     * initialize all the entity initializers.
     */
    public void initialize() {
        for (SystemInitializer initializer : entityInitializers) {
            initializer.initialize();
        }
    }
}

```

databasespelldictionary.java

```

/*
 * $Id: DatabaseSpellDictionary.java,v 1.5 2009/01/24 11:08:38 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;
```

```

import java.io.File;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.List;

import com.swabunga.spell.engine.SpellDictionaryASpell;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * An implementation of Jazzy's SpellDictionary using The database
based
 * DictionaryRepository of WordNet words.
 * <p>
 *
 * @author ron
 */
public class DatabaseSpellDictionary extends SpellDictionaryASpell {

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     * @param phonetic
     * @param encoding
     * @throws IOException
     */
    public DatabaseSpellDictionary(DictionaryRepository
dictionaryRepository, File phonetic,
        String encoding) throws IOException {
        super(phonetic, encoding);
        this.dictionaryRepository = dictionaryRepository;
    }

    /**
     * @param dictionaryRepository
     * @param phonetic
     * @throws IOException
     */
    public DatabaseSpellDictionary(DictionaryRepository
dictionaryRepository, File phonetic)
        throws IOException {
        super(phonetic);
        this.dictionaryRepository = dictionaryRepository;
    }
}

```

```

    /**
     * @param dictionaryRepository
     * @param phonetic
     * @throws IOException
     */
    public DatabaseSpellDictionary(DictionaryRepository
dictionaryRepository, Reader phonetic)
        throws IOException {
        super(phonetic);
        this.dictionaryRepository = dictionaryRepository;
    }

    /**
     * Returns a list of words that have the same phonetic code.
     *
     * @param phoneticCode
     *          The phonetic code common to the list of words
     * @return A list of words having the same phonetic code
     */
    @Override
    protected List<String> getWords(String phoneticCode) {
        List<String> results = new ArrayList<String>();
        for (edu.harvard.fas.rregan.nlp.dictionary.Word word :
dictionaryRepository
            .findWordsByPhoneticCode(phoneticCode)) {
            results.add(word.getLemma());
        }
        return results;
    }

    /**
     * @see
     *      com.swabunga.spell.engine.SpellDictionary#addWord(java.lang.String)
     */
    public void addWord(String text) {
        edu.harvard.fas.rregan.nlp.dictionary.Word word = new
edu.harvard.fas.rregan.nlp.dictionary.Word(
            text, getCode(text));
        dictionaryRepository.persist(word);
    }
}

```

dateadapter.java

```

/*
 * $Id: DateAdapter.java,v 1.5 2008/06/20 08:23:07 rregan Exp $

```

```

* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.requel.utils.jaxb;

import java.util.Date;

import javax.xml.bind.annotation.adapters.XmlAdapter;

import edu.harvard.fas.rregan.requel.utils.DateUtils;

/**
 * @author ron
 */
public class DateAdapter extends XmlAdapter<String, Date> {

    @Override
    public Date unmarshal(String dateString) throws Exception {
        return DateUtils.parseDateOrDefault(dateString, new Date());
    }

    @Override
    public String marshal(Date date) throws Exception {
        if (date != null) {
            return DateUtils.standardDateAndTime.format(date);
        }
        return "";
    }
}

```

dateutils.java

```

package edu.harvard.fas.rregan.requel.utils; import
java.text.DateFormat; import java.text.SimpleDateFormat; import
java.util.Calendar; import java.util.Date; import java.util.HashMap;
import java.util.Map; import
edu.harvard.fas.rregan.ApplicationException; /** * helper functions
for working with dates. */ /* @author ron */ public class DateUtils
{ private DateUtils() { } public static final DateFormat
standardDateAndTime = new SimpleDateFormat( "yyyy-MM-dd'T'HH:mm:ss");
public static final DateFormat standardDate = new
SimpleDateFormat("yyyy-MM-dd"); /* * Date parts */ public static
final DateFormat yearAs_yy = new SimpleDateFormat("yy"); public static
final DateFormat yearAs_yyyy = new SimpleDateFormat("yyyy"); public
static final DateFormat monthAs_M = new SimpleDateFormat("M"); public
static final DateFormat monthAs_MM = new SimpleDateFormat("MM");
public static final DateFormat monthAs MMM = new

```

```

SimpleDateFormat("MMM"); public static final DateFormat dayAs_d = new
SimpleDateFormat("d"); public static final DateFormat dayAs_dd = new
SimpleDateFormat("dd"); public static final DateFormat hoursAs_H = new
SimpleDateFormat("H"); public static final DateFormat hoursAs_HH = new
SimpleDateFormat("HH"); public static final DateFormat hoursAs_h = new
SimpleDateFormat("h"); public static final DateFormat hoursAs_hh = new
SimpleDateFormat("hh"); public static final DateFormat minutesAs_m =
new SimpleDateFormat("m"); public static final DateFormat minutesAs_mm =
new SimpleDateFormat("mm"); /* * Returns a Map of the date parts of
the supplied date.
* NOTE: only yy, yyyy, M, MM, d, dd, H, HH, h, hh, m and mm are
implemented
*/

```

Letter	Date or Time Component *	Presentation *	Examples *
G *	Era designator *	Text *	AD *
Y *	Year *	Year *	1996; 96 *
M *	Month in year *	Month *	July; Jul; 07 *
W *	Week in year *	Number *	27 *
W *	Week in month *	Number *	2 *
D *	Day in year *	Number *	189 *
d *	Day in month *	Number *	10 *
F *	Day of week in month *	Number *	2 *
E *	Day in week *	Text *	Tuesday; Tue *
a *	Am/pm marker *	Text *	PM *
H *	Hour in day (0-23) *	Number *	0 *
k *	Hour in day (1-24) *	Number *	24 *
K *	Hour in am/pm (0-11) *	Number *	0 *
h *	Hour in am/pm (1-12) *	Number *	12 *

```

*      Minute in hour * Number *      30 *
m *
*      Second in minute Number *      55 *
S *
*      Millisecond * Number *      978 *
S *
*      Time zone * General time zone * Pacific Standard Time;
Z *      PST; * GMT-08:00 *
*      Time zone * RFC 822 time zone * -0800
* * @param date * @return */ public static Map getDateParts(Date date)
{ Map dateParts = new HashMap(); dateParts.put("yy",
yearAs_yy.format(date)); dateParts.put("yyyy",
yearAs_yyyy.format(date)); dateParts.put("M", monthAs_M.format(date));
dateParts.put("MM", monthAs_MM.format(date)); dateParts.put("d",
dayAs_d.format(date)); dateParts.put("dd", dayAs_dd.format(date));
dateParts.put("H", hoursAs_H.format(date)); dateParts.put("HH",
hoursAs_HH.format(date)); dateParts.put("h", hoursAs_h.format(date));
dateParts.put("hh", hoursAs_hh.format(date)); dateParts.put("m",
minutesAs_m.format(date)); dateParts.put("mm",
minutesAs_mm.format(date)); return dateParts; } /* * Try to parse the
supplied dateString as a date or datetime and if it * can't be parsed
throw an exception. * * @param dateString * @return * @throws
ApplicationException - * if the supplied dateString can't be parsed.
*/ public static Date parseDate(String dateString) throws
ApplicationException { return parseDateOrDefault(dateString,
null); } /* * Try to parse the supplied dateString as a date or
datetime and if it * can't be parsed return the defaultValue date. * *
@param dateString * @param defaultValue * @return */ public static
Date parseDateOrDefault(String dateString, Date defaultValue) { for
(DateFormat dateFormat : new DateFormat[] { standardDate,
standardDateAndTime }) { try { return dateFormat.parse(dateString);
} catch (Exception e) { } if (defaultValue != null) { return
defaultValue; } throw
ApplicationException.unsupportedDateString(dateString); } /* * Format
a date using java.text.SimpleDateFormat * * @param date * date to
format * @param formatString * date and time pattern suitable for
java.text.SimpleDateFormat * @return */ public static String
simpleDateFormat(Date date, String formatString) { DateFormat df = new
SimpleDateFormat(formatString); String result = df.format(date);
return result; } /* * Returns the given date with milliseconds set to
000 * * @param date * @return */ public static Date startOfSecond(Date
date) { if (date != null) { Calendar cal = Calendar.getInstance();
cal.setTime(date); cal.set(Calendar.MILLISECOND, 0); return
cal.getTime(); } else { return null; } } /* * Returns the given date
with the time set to 00:00:00.000 (midnight) * * @param date * @return

```

```

/* public static Date startOfDay(Date date) { if (date != null)
{ Calendar cal = Calendar.getInstance(); cal.setTime(date);
cal.set(Calendar.HOUR_OF_DAY, 0); cal.set(Calendar.MINUTE, 0);
cal.set(Calendar.SECOND, 0); cal.set(Calendar.MILLISECOND, 0); return
cal.getTime(); } else { return null; } } /* * Returns the given date
with the time set to 11:59:59.999 * * @param date * @return */ public
static Date endOfDay(Date date) { if (date != null) { Calendar cal =
Calendar.getInstance(); cal.setTime(date);
cal.set(Calendar.HOUR_OF_DAY, 23); cal.set(Calendar.MINUTE, 59);
cal.set(Calendar.SECOND, 59); cal.set(Calendar.MILLISECOND, 999);
return cal.getTime(); } else { return null; } } public static Date
addDays(Date date, int days) { if (date != null) { Calendar cal =
Calendar.getInstance(); cal.setTime(date);
cal.add(Calendar.DAY_OF_YEAR, days); return cal.getTime(); } else
{ return null; } } public static Date addHours(Date date, int hours)
{ if (date != null) { Calendar cal = Calendar.getInstance();
cal.setTime(date); cal.add(Calendar.HOUR_OF_DAY, hours); return
cal.getTime(); } else { return null; } } public static Date
nextDay(Date date) { return addDays(date, 1); } public static Date
nextWeek(Date date) { return addDays(date, 7); } public static Date
nextMonth(Date date) { if (date != null) { Calendar cal =
Calendar.getInstance(); cal.setTime(date); cal.add(Calendar.MONTH, 1);
return cal.getTime(); } else { return null; } } public static Date
nextYear(Date date) { if (date != null) { Calendar cal =
Calendar.getInstance(); cal.setTime(date); cal.add(Calendar.YEAR, 1);
return cal.getTime(); } else { return null; } } public static Date
endOfTheWeek(Date date) { if (date != null) { Calendar cal =
Calendar.getInstance(); cal.setTime(date);
cal.set(Calendar.DAY_OF_WEEK, Calendar.SATURDAY); return
cal.getTime(); } else { return null; } }

```

defaultcommandhandler.java

```

/*
 * $Id: DefaultCommandHandler.java,v 1.1 2009/04/01 15:45:47 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import
edu.harvard.fas.rregan.requel.user.exception.NoSuchOrganizationExcepti
on;

```

```

import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**
 * The default handler executes the command in a transaction locally
to the
 * process that supplied the command.
*
* @author ron
*/
public class DefaultCommandHandler implements CommandHandler {
    @Transactional(propagation = Propagation.REQUIRED, noRollbackFor = {
        NoSuchOrganizationException.class, NoSuchUserException.class })
    public <T extends Command> T execute(T command) throws Exception {
        command.execute();
        return command;
    }
}

```

defaulteditortreenode.java

```

/*
 * $Id: DefaultEditorTreeNode.java,v 1.2 2008/10/22 22:41:56 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ActionEvent;
import echopointng.tree.DefaultMutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * A mutable tree node that contains an editor. The node may have a
listener
 * that listens for update events for the entity object that the node
is
 * displaying to indicate that the node view should be updated.
*
* @author ron
*/

```

```

public class DefaultEditorTreeNode extends DefaultMutableTreeNode
implements EditorTreeNode {
    static final long serialVersionUID = 0L;

    private final EventDispatcher eventDispatcher;
    private Component editor;
    private EditorTreeNodeUpdateListener updateListener;

    /**
     * @param eventDispatcher
     * @param editor
     */
    public DefaultEditorTreeNode(EventDispatcher eventDispatcher,
Component editor) {
        this(eventDispatcher, editor, editor);
    }

    /**
     * @param eventDispatcher
     * @param editor
     * @param label
     */
    public DefaultEditorTreeNode(EventDispatcher eventDispatcher,
Component editor, Component label) {
        super(label);
        this.eventDispatcher = eventDispatcher;
        setEditor(editor);
    }

    public void actionPerformed(ActionEvent e) {
        if (updateListener != null) {
            updateListener.actionPerformed(e);
        }
    }

    public Component getEditor() {
        return editor;
    }

    public void setEditor(Component editor) {
        this.editor = editor;
    }

    public EventDispatcher getEventDispatcher() {
        return eventDispatcher;
    }
}

```

```

/**
 * Set a listener that gets notified if the object for the node is
modified.
 * The new listener will be registered with the event dispatcher by
this
 * method. If another listener is already set, it will be
unregistered
 * before the new one is registered. If null is supplied, the old
listener
 * will be unregistered and no listeners will be listening for this
node.
 *
 * @param updateListener
 */
public void setUpdateListener(EditorTreeNodeUpdateListener
updateListener) {
    if (this.updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            this.updateListener);
    }
    this.updateListener = updateListener;
    if (this.updateListener != null) {
        getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.c
lass,
            this.updateListener);
    }
}

public void dispose() {
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
    this.updateListener = null;
}
}
}

```

defaulteventdispatcher.java

```

/*
 * $Id: DefaultEventDispatcher.java,v 1.23 2008/12/17 02:00:43 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.uiframework.navigation.event;

import java.util.Collection;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.PanelDescriptor;
import edu.harvard.fas.rregan.uiframework.screen.Screen;

/**
 * The default implementation of the EventDispatcher. It supports a
single
 * instance of a UIFrameworkApp for a single user, no cross user
events.
 *
 * @author ron
 */
@Component("eventDispatcher")
@Scope("session")
public class DefaultEventDispatcher implements EventDispatcher {
    private static final Logger log =
Logger.getLogger(DefaultEventDispatcher.class);
    static final long serialVersionUID = 0L;

    // TODO: should the entries use a Weak or Soft reference so that if a
    // component is deleted from the UI it won't stick around because it
    // is registered for events?
    private final Map<PanelListenerKey, Set<ActionListener>>
eventTypeListenerSets = new HashMap<PanelListenerKey,
Set<ActionListener>>();

    /**
     */
}

```

```

public DefaultEventDispatcher() {
    log.debug("new DefaultEventDispatcher");
}

/**
 * dispatches an event to all appropriate listeners based on the
event's
 * source, type and action command. This is a simple wrapper for
* actionPerformed()
 *
 * @param event -
 *          an ActionEvent to dispatch
 */
public void dispatchEvent(ActionEvent event) {
    actionPerformed(event);
}

/**
 * The EventDispatcher registers itself as a Listener for all objects
 * registered as event sources so that it can forward events to
listeners
 * indirectly.
 *
 * @throws EventDispatcherMultiException -
 *          if any listener actionPerformed() throws an exception.
It
 *          contains a map keyed by listeners to the exception
thrown.
 *          All listeners will be called no matter which listeners
throw
 *          exceptions and this exception is thrown at the end.
 */
public void actionPerformed(ActionEvent event) {
    log.debug("event = " + event);

    if (event != null) {
        Set<ActionListener> distinctListeners =
getDistinctListenersForEvent(event);

        if (distinctListeners.size() > 0) {
            // all listeners are notified even if some throw exceptions, the
            // thrown exceptions are recorded with the listener that threw
            // it and reported at the end.
            Map<ActionListener, Exception> listenerExceptions = new
HashMap<ActionListener, Exception>();

            for (ActionListener listener : distinctListeners) {

```

```

                try {
                    log.debug("sending event " + event + " to listener " +
listener);
                    listener.actionPerformed(event);
                } catch (Exception e) {
                    listenerExceptions.put(listener, e);
                    log.warn("listener " + listener
                            + " threw an exception from actionPerformed() for event " +
event,
                            e);
                }
            }
            if (!listenerExceptions.isEmpty()) {
                // TODO: maybe this shouldn't throw an exception, but fire
                // an event with a message to be displayed in a window.
                throw new EventDispatcherMultiException(listenerExceptions);
            }
        } else {
            log.warn("no listeners match for event: " + event);
        }
    } else {
        log.warn("null event supplied.");
    }
}

/**
 * NOTE: this is protected for testing purposes, it should not be
called
 * directly.<br>
 *
 * @param event
 */
protected Set<ActionListener>
getDistinctListenersForEvent(ActionEvent event) {
    // Use a set to collect all the distinct listeners so that if a
    // listener is listening based on different criteria, it only
    // receives the event once. Screen listeners are notified first
    // followed by Panels and then all other listeners
    Set<ActionListener> distinctListeners = new TreeSet<ActionListener>(
        new ActionListenerComparator());

    if (event != null) {
        // a listener registered for a super type of the event type should
        // be notified of the event.

        if (event instanceof OpenPanelEvent) {
            // OpenPanelEvent events can be listened to for specialized

```

```

// criteria see PanelListenerKey()
OpenPanelEvent openEvent = (OpenPanelEvent) event;
if (openEvent.getPanel() != null) {
    // if a panel is specified, collect listeners for that panel
    // by event type and super types.
    Class<?> eventType = event.getClass();
    do {
        distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(
    Class.class.cast(eventType), (OpenPanelEvent) event)));
        eventType = eventType.getSuperclass();
    } while (ActionEvent.class.isAssignableFrom(eventType));
}
} else {
    // look up the panel based on the event, action and content
    // types, including event super types and content super
    // types and interfaces
    addListenersByActionAndTargetType(distinctListeners, openEvent);
}
} else if (event instanceof ClosePanelEvent) {
    // ClosePanelEvent events can be listened to for specific panels
    // see PanelListenerKey()
    Class<?> eventType = event.getClass();
    do {
        distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(
    Class.class.cast(eventType), (ClosePanelEvent) event)));
        eventType = eventType.getSuperclass();
    } while (ActionEvent.class.isAssignableFrom(eventType));
}

Class<?> eventType = event.getClass();
do {
    distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(Class.class
    .cast(eventType))));
    if (NavigationEvent.class.isAssignableFrom(eventType)) {
        distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(
    Class.class.cast(eventType), (Object)
((NavigationEvent)event).getDestinationObject())));
    }
    eventType = eventType.getSuperclass();
} while (ActionEvent.class.isAssignableFrom(eventType));
}
return distinctListeners;
}

```

```

protected void addListenersByActionAndTargetType(Set<ActionListener>
distinctListeners,
    OpenPanelEvent openEvent) {
    Class<?> targetType = openEvent.getTargetType();
    Class<?> eventType = openEvent.getClass();
    if (targetType != null) {
        do {
            while (targetType != null) {
                distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(Class.class
    .cast(eventType), openEvent.getPanelActionType(), targetType,
openEvent
    .getPanelName(), openEvent.getPanel(), null)));
            if (targetType.getInterfaces() != null) {
                for (Class<?> face : targetType.getInterfaces()) {
                    distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(
    Class.class.cast(eventType), openEvent.getPanelActionType(),
face,
    openEvent.getPanelName(), openEvent.getPanel(), null)));
                }
            }
            targetType = targetType.getSuperclass();
        }
        eventType = eventType.getSuperclass();
    } while (OpenPanelEvent.class.isAssignableFrom(eventType));
} else {
    // find a panel without a target type
    distinctListeners.addAll(getEventListenersByKey(new
PanelListenerKey(Class.class
    .cast(eventType), openEvent.getPanelActionType(), null, openEvent
    .getPanelName(), openEvent.getPanel(), null)));
}
}

/**
 * Get all the ActionListeners registered to receive specific
ActionEvent
 * types or based on PanelListenerKey (for listeners registered by
 * PanelDescriptors or specific Panels) <br>
 * NOTE: this is protected for testing purposes, it should not be
called
 * directly.
 *

```

```

 * @param eventType -
 *          an ActionEvent class or PanelListenerKey
 * @return
 */
protected Collection<ActionListener>
getEventListenersByKey(PanelListenerKey key) {
    Set<ActionListener> actionListeners =
eventTypeActionListenerSets.get(key);
    if (actionListeners == null) {
        actionListeners = new HashSet<ActionListener>();
        eventTypeActionListenerSets.put(key, actionListeners);
    }
    return actionListeners;
}

public ActionListener addEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener) {
    log.debug("eventType = " + eventType + " listener = " + listener);
    getEventListenersByKey(new
PanelListenerKey(Class.class.cast(eventType)))
        .add(listener);
    return listener;
}

public ActionListener addEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener, Object destinationObject) {
    log.debug("eventType = " + eventType + " listener = " + listener + " "
destinationObject = " + destinationObject);
    getEventListenersByKey(new
PanelListenerKey(Class.class.cast(eventType), destinationObject))
        .add(listener);
    return listener;
}

public void removeEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener) {
    log.debug("eventType = " + eventType + " listener = " + listener);
    getEventListenersByKey(new
PanelListenerKey(Class.class.cast(eventType)))
        .remove(listener);
}

public void removeEventTypeActionListener(Class<? extends
ActionEvent> eventType,

```

```

    ActionListener listener, Object destinationObject) {
    log.debug("eventType = " + eventType + " listener = " + listener);
    getEventListenersByKey(new
PanelListenerKey(Class.class.cast(eventType), destinationObject))
        .remove(listener);
}

/**
 * add a listener for open panel events that match the supplied
 * PanelDescriptor
 */
public ActionListener addOpenPanelEventActionListener(PanelDescriptor
panelDescriptor,
    ActionListener listener) {
    log.debug("panelDescriptor = " + panelDescriptor + " listener = " +
listener);
    getEventListenersByKey(new
PanelListenerKey(panelDescriptor)).add(listener);
    return listener;
}

/**
 * remove a listener for open panel events that match the supplied
 * PanelDescriptor
 */
public void removeOpenPanelEventActionListener(PanelDescriptor
panelDescriptor,
    ActionListener listener) {
    log.debug("panelDescriptor = " + panelDescriptor + " listener = " +
listener);
    getEventListenersByKey(new
PanelListenerKey(panelDescriptor)).remove(listener);
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher#ad
dPanelInstanceEventActionListener(edu.harvard.fas.rregan.uiframework.p
anel.Panel,
    * nextapp.echo2.app.event.ActionListener)
*/
public ActionListener addPanelInstanceEventActionListener(Panel
panel, ActionListener listener) {
    log.debug("panel = " + panel + " listener = " + listener);
    getEventListenersByKey(new PanelListenerKey(ActionEvent.class,
panel)).add(listener);
    return listener;

```

```

}

/**
 * @see
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher#re
movePanelInstanceEventActionListener(edu.harvard.fas.rregan.uiframewor
k.panel.Panel,
    *      nextapp.echo2.app.event.ActionListener)
 */
public void removePanelInstanceEventActionListener(Panel panel,
ActionListener listener) {
    log.debug("panel = " + panel + " listener = " + listener);
    getEventListenersByKey(new PanelListenerKey(ActionEvent.class,
panel)).remove(listener);
}

@Override
public String toString() {
    return getClass().getSimpleName();
}

/**
 * NOTE: this class is protected for testing purposes, it should not
be used
 * directly.
 *
 * @author ron
 */
protected static class PanelListenerKey {
    private final Class<? extends ActionEvent> eventType;
    private final PanelActionType actionType;
    private final Class<?> contentType;
    private final String panelName;
    private final Panel panel;
    private final Object destination;

    /**
     * This is for creating a key to register a listener rather than
looking
     * up a listener.<br>
     * Create a key for the type of event only. this is primarily for
     * controllers that listen for specific types of events.
     *
     * @param eventType
     */
    protected PanelListenerKey(Class<? extends ActionEvent> eventType) {
        this(eventType, null, null, null, null, null);
    }

    /**
     * This is for creating a key to register a listener rather than
register
     * a listener.<br>
     * Create a key that will match OpenPanelEvents to a specific type
of
     * panel. This is used to register a panel manager to get events for
the
     * types of windows it manages.
     *
     * @param panelDescriptor
     */
    protected PanelListenerKey(PanelDescriptor panelDescriptor) {
        this(OpenPanelEvent.class,
            panelDescriptor.getSupportedActionType(), panelDescriptor
                .getSupportedContentTypes(), panelDescriptor.getPanelName(), null,
            null);
    }

    /**
     * This is for creating a key to look up a listener rather than
register
     * a listener.<br>
     * Create a key for a given event, substituting the supplied class
for
     * the event type.
     *
     * @param eventType
     * @param event
     */
    protected PanelListenerKey(Class<? extends ActionEvent> eventType,
OpenPanelEvent event) {
        this(eventType, event.getPanelActionType(), event.getTargetType(),
            event.getPanelName(), event.getPanel(), null);
    }

    /**
     * This is for creating a key to look up a listener rather than
register
     * a listener.<br>
     * Create a key to match a ClosePanelEvent.
     *
     * @param eventType -
     *          the type is specified separately so that the search
can
    }
}

```

```

        * start with the narrowest type that matches, and expand
to
        * search for super types.
        * @param event
        */
protected PanelListenerKey(Class<? extends ActionEvent> eventType,
ClosePanelEvent event) {
    this(eventType, event.getPanelToClose());
}

/**
 * This is for creating a key to register a panel specific listener
and
 * indirectly used to looking up a panel specific listener.<br>
 * Create a key for a specific panel instance with the specified
action
 * type.
 *
 * @param destination -
 *          the destination object, most likely the original panel
or
 *          component (source) of an event that caused a window to
open and is tied to its resulting event, like a select
object event.
 */
protected PanelListenerKey(Class<? extends ActionEvent> eventType,
Object destination) {
    this(eventType, null, null, null, null, destination);
}

/**
 * This is for creating a key to register a panel specific listener
and
 * indirectly used to looking up a panel specific listener.<br>
 * Create a key for a specific panel instance with the specified
action
 * type.
 *
 * @param panel
 */
protected PanelListenerKey(Class<? extends ActionEvent> eventType,
Panel panel) {
    this(eventType, null, null, null, panel, null);
}

/**
 * this is the root constructor used by all the other constructors.

```

```

        * Every part of the key can be specified directly.
        *
        * @param eventType
        * @param actionType
        * @param contentType
        * @param panelName
        * @param panel
        */
protected PanelListenerKey(Class<? extends ActionEvent> eventType,
    PanelActionType actionType, Class<?> contentType, String
panelName, Panel panel,
    Object destination) {
    this.eventType = eventType;
    this.actionType = actionType;
    this.contentType = contentType;
    this.panelName = panelName;
    this.panel = panel;
    this.destination = destination;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((eventType == null) ? 0 :
eventType.hashCode());
    result = prime * result + ((actionType == null) ? 0 :
actionType.hashCode());
    result = prime * result + ((contentType == null) ? 0 :
contentType.hashCode());
    result = prime * result + ((panelName == null) ? 0 :
panelName.hashCode());
    result = prime * result + ((panel == null) ? 0 : panel.hashCode());
    result = prime * result + ((destination == null) ? 0 :
destination.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {

```

```

    return false;
}

final PanelListenerKey other = (PanelListenerKey) obj;
if (eventType == null) {
    if (other.eventType != null) {
        return false;
    }
} else if (!eventType.equals(other.eventType)) {
    return false;
}
if (actionType == null) {
    if (other.actionType != null) {
        return false;
    }
} else if (!actionType.equals(other.actionType)) {
    return false;
}
if (contentType == null) {
    if (other.contentType != null) {
        return false;
    }
} else if (!contentType.equals(other.contentType)) {
    return false;
}
if (panelName == null) {
    if (other.panelName != null) {
        return false;
    }
} else if (!panelName.equals(other.panelName)) {
    return false;
}
if (panel == null) {
    if (other.panel != null) {
        return false;
    }
} else if (!panel.equals(other.panel)) {
    return false;
}
if (destination == null) {
    if (other.destination != null) {
        return false;
    }
} else if (!destination.equals(other.destination)) {
    return false;
}
return true;
}

    }

@Override
public String toString() {
    return getClass().getSimpleName() + "[eventType = " + eventType +
", actionType = "
+ actionType + ", contentType = " + contentType + ", panelName =
" + panelName
+ ", panel = " + panel + ", destination = " + destination + "]";
}
}

// TODO: could this cause two different action listeners to be
// considered
// equal, or
// the same listener to not be equal to itself?
// Order the listeners so that Screens and Panels get messages
// before other components to solve the problem with a panel loading
// its
// state and erasing a change that a control on the panel received.
protected static class ActionListenerComparator implements
Comparator<ActionListener> {
    public int compare(ActionListener o1, ActionListener o2) {
        if (o1 == o2) {
            return 0;
        } else if ((o1 instanceof Screen) && !(o2 instanceof Screen)) {
            return -1;
        } else if ((o1 instanceof Panel) && !(o2 instanceof Panel)) {
            return -1;
        } else {
            // added this because EventDispatcherTest was failing as a
            // listener was not considered equal to itself when using the
            // contains() method
            if (o1.equals(o2)) {
                return 0;
            }
            return o1.hashCode() - o2.hashCode();
        }
    }
};

}

defaultpanelfactory.java
/*

```

```

* $Id: DefaultPanelFactory.java,v 1.6 2008/12/17 08:38:05 rregan Exp
$ Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.uiframework.panel;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import java.util.List;

/**
 * A factory for creating instances of a specific type of panel.
 *
 * @author ron
 */
public class DefaultPanelFactory implements PanelFactory {

    private final PanelActionType supportedActionType;
    private final Class<?> supportedContentType;
    private final String panelName;
    private final Class<? extends Panel> panelType;
    private final List<Object> panelConstructorArgs;

    /**
     * @param panelType
     * @param supportedContentType
     * @param panelName
     */
    public DefaultPanelFactory(Class<? extends Panel> panelType, Class<?>
        supportedContentType,
        String panelName) {
        this(panelType, supportedContentType, PanelActionType.Unspecified,
            panelName, null);
    }

    /**
     * For a panel type that doesn't take an object for editing.
     *
     * @param panelType
     * @param supportedActionType
     * @param panelName
     * @param panelConstructorArgs
     */
    public DefaultPanelFactory(Class<? extends Panel> panelType,
        PanelActionType supportedActionType, String panelName, List<Object>
        panelConstructorArgs) {
        this(panelType, null, supportedActionType, panelName,
            panelConstructorArgs);
    }

    /**
     * @param panelType
     * @param supportedContentType
     * @param supportedActionType
     */
    public DefaultPanelFactory(Class<? extends Panel> panelType, Class<?>
        supportedContentType,
        PanelActionType supportedActionType) {
        this(panelType, supportedContentType, supportedActionType, null,
            null);
    }

    /**
     * @param panelType
     * @param supportedContentType
     * @param supportedActionType
     * @param panelConstructorArgs -
     *           the arguments to desired constructor of the target
     *           class. The
     *           arguments must be objects that are shareable by all
     *           instances
     *           of the panels that get created, for example singleton
     *           type
     *           objects like entity managers. the arguments must be in
     *           the
     *           order specified by the constructor of the panel.
     */
    public DefaultPanelFactory(Class<? extends Panel> panelType, Class<?>
        supportedContentType,
        PanelActionType supportedActionType, List<Object>
        panelConstructorArgs) {
        this(panelType, supportedContentType, supportedActionType, null,
            panelConstructorArgs);
    }

    /**
     * @param panelType
     * @param supportedContentType
     * @param supportedActionType
     * @param panelName
     * @param panelConstructorArgs -
     *           the arguments to desired constructor of the target
     *           class. The

```

```

/*
 * instances
 *      arguments must be objects that are shareable by all
 *      of the panels that get created, for example singleton
 *      objects like entity managers. the arguments must be in
 *      order specified by the constructor of the panel.
 */
public DefaultPanelFactory(Class<? extends Panel> panelType, Class<?>
supportedContentType,
    PanelActionType supportedActionType, String panelName, List<Object>
panelConstructorArgs) {
    this.supportedActionType = supportedActionType;
    this.supportedContentType = supportedContentType;
    this.panelName = panelName;
    this.panelType = panelType;
    this.panelConstructorArgs = panelConstructorArgs;
}

public PanelActionType getSupportedActionType() {
    return supportedActionType;
}

public Class<?> getSupportedContentTypes() {
    return supportedContentType;
}

public String getPanelName() {
    return panelName;
}

public Class<? extends Panel> getPanelType() {
    return panelType;
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.PanelFactory#dispose()
 */
public void dispose() {
    // TODO: cleanup any resources. the default PanelFactory doesn't
    // pool panels so there is nothing to do now.
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.PanelFactory#newInstance()
 */

arguments must be objects that are shareable by all
of the panels that get created, for example singleton
objects like entity managers. the arguments must be in
order specified by the constructor of the panel.

*/
public Panel newInstance() throws NoSuchMethodException,
InvocationTargetException,
InstantiationException, IllegalAccessException {
    if (panelConstructorArgs == null) {
        return getPanelType().newInstance();
    } else {
        return newInstance(panelConstructorArgs);
    }
}

/**
 * @param args
 * @return
 * @throws NoSuchMethodException
 * @throws InvocationTargetException
 * @throws InstantiationException
 * @throws IllegalAccessException
 */
protected Panel newInstance(List<Object> args) throws
NoSuchMethodException,
InvocationTargetException, InstantiationException,
IllegalAccessException {
    // TODO: if the panel doesn't have a constructor that matches the
    // supplied parameters, but does
    // have a constructor that matches the supplied parameters plus one
or
    // more of the supportedActionType,
    // supportedContentType, or panelName then this method should be
able to
    // supply those values directly.

    Constructor<? extends Panel> constructor =
getConstructor(getPanelType(), args);
    return constructor.newInstance(args.toArray());
}

private Constructor<? extends Panel> getConstructor(Class<? extends
Panel> panelType,
List<Object> args) throws NoSuchMethodException {
    Class<?> parameterTypes[] = new Class<?>[args.size()];
    for (int i = 0; i < args.size(); i++) {
        parameterTypes[i] = args.get(i).getClass();
    }
    for (Constructor<?> constructor : panelType.getConstructors()) {
        if (constructor.getParameterTypes().length == args.size()) {
            boolean match = true;

```

```

for (Class<?> paramType : constructor.getParameterTypes()) {
    if (paramType.isAssignableFrom(args.getClass())) {
        match = false;
        break;
    }
}
if (match) {
    return (Constructor) constructor;
}
}

StringBuilder errMsg = new StringBuilder();
errMsg.append(panelType.getName());
errMsg.append("<init>()");
for (Object arg : args) {
    errMsg.append(arg.getClass().getName());
}
errMsg.append(")");
throw new NoSuchMethodException(errMsg.toString());
}

@Override
public String toString() {
    return "supportedActionType = " + getSupportedActionType() + "
supportedContentType = "
    + getSupportedContentTypes() + " panelName = " + getPanelName() + "
panelType = "
    + getPanelType();
}
}

```

defaultpanelmanager.java

```

/*
 * $Id: DefaultPanelManager.java,v 1.14 2008/12/17 02:00:42 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframe.Panel;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import nextapp.echo2.app.event.ActionEvent;

```

```

import org.apache.log4j.Logger;
import edu.harvard.fas.rregan.uiframe.PanelContainer;
import edu.harvard.fas.rregan.uiframe.navigation.event.ClosePanelEvent;
import edu.harvard.fas.rregan.uiframe.navigation.event.EventDispatcher;
import edu.harvard.fas.rregan.uiframe.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframe.navigation.event.OpenPanelEvent;

/**
 * The DefaultPanelManager
 *
 * @author ron
 */
public class DefaultPanelManager implements PanelManager {
    static final long serialVersionUID = 0L;
    private static final Logger log =
        Logger.getLogger(DefaultPanelManager.class);

    private final Map<PanelRegistryKey, PanelDescriptor> panelDescriptors =
        new HashMap<PanelRegistryKey, PanelDescriptor>();
    private final EventDispatcher eventDispatcher;

    private PanelContainer panelContainer;

    /**
     * Create a PanelManager without the managed PanelContainer. The
     * PanelContainer must be set before the PanelManager can handle
     * events
     * through the actionPerformed method.<br/> The manager accepts a Set
     * of
     * PanelDescriptors that may be actual Panel instances or
     * PanelFactory
     * instances. Supplied Panels will be reused each time an Event
     * causes the
     * Panel to be displayed. A PanelFactory may create a new Panel for
     * each
     * request or use a pool of panels.<br/>
     *
     * @param eventDispatcher -
     *          the dispatcher that will send open panel events to the
     *          PanelContainer.
     * @param panelDescriptors -
     */
}
```

```

        *      a set of Panels and/or PanelFactorys that this manager
will      use to populate its assigned panel container.
*/
public DefaultPanelManager(EventDispatcher eventDispatcher,
    Set<PanelDescriptor> panelDescriptors) {
    this.eventDispatcher = eventDispatcher;
    register(panelDescriptors, eventDispatcher);
}

protected EventDispatcher getEventDispatcher() {
    return eventDispatcher;
}

/**
 * Assign the Panel container that this manager will apply events to.
 *
 * @param panelContainer
 */
public void setPanelContainer(PanelContainer panelContainer) {
    if ((panelContainer == null) && (this.panelContainer != null)) {
        this.panelContainer.dispose();
    }
    this.panelContainer = panelContainer;
}

protected PanelContainer getPanelContainer() {
    return panelContainer;
}

/**
 * This is the method that gets called by the EventDispatcher when
the
 * dispatcher gets an OpenPanelEvent or ClosePanelEvent for one of
the
 * PanelDescriptors registered with this manager.
 *
 * @see
nextapp.echo2.app.event.ActionListener#actionPerformed(nextapp.echo2.a
pp.event.ActionEvent)
 */
public void actionPerformed(ActionEvent e) {
    if (e instanceof NavigationEvent) {
        if (e instanceof OpenPanelEvent) {
            OpenPanelEvent event = (OpenPanelEvent) e;
            Panel panel = getPanelForEvent(event);

                getEventDispatcher().addPanelInstanceEventActionListener(panel,
this);
                getPanelContainer().displayPanel(panel,
event.getWorkflowDisposition());
            } else if (e instanceof ClosePanelEvent) {
                ClosePanelEvent event = (ClosePanelEvent) e;
                Panel panelToClose = event.getPanelToClose();
                if (panelToClose != null) {
                    getPanelContainer().undisplayPanel(panelToClose);
                    unregister(panelToClose, getEventDispatcher(),
panelToClose.getTargetObject());
                }
            }
        }
    }
}

public void register(PanelDescriptor panelDescriptor, EventDispatcher
eventDispatcher) {
    register(panelDescriptor, eventDispatcher, null);
}

private void register(PanelDescriptor panelDescriptor,
EventDispatcher eventDispatcher,
Object targetObject) {
    PanelRegistryKey panelRegistryKey = new
PanelRegistryKey(panelDescriptor, targetObject);
    PanelDescriptor existingPanelDescriptor =
panelDescriptors.get(panelRegistryKey);
    if (existingPanelDescriptor != null) {
        eventDispatcher.removeOpenPanelEventActionListener(panelDescriptor,
this);
        log.warn("The Panel or PanelFactory " + panelDescriptor + " is
replacing "
+ existingPanelDescriptor);
    }
    eventDispatcher.addOpenPanelEventActionListener(panelDescriptor,
this);
    panelDescriptors.put(panelRegistryKey, panelDescriptor);
}

private void unregister(PanelDescriptor panelDescriptor,
EventDispatcher eventDispatcher,
Object targetObject) {
    // only unregister panels for specific targets, otherwise the
specific
    // type of panel may never be opened again.
}

```

```

if ((targetObject != null) && (panelDescriptor instanceof Panel)) {
    PanelRegistryKey panelRegistryKey = new
    PanelRegistryKey(panelDescriptor, targetObject);
    if (panelDescriptors.containsKey(panelRegistryKey)) {
        panelDescriptors.remove(panelRegistryKey);
        eventDispatcher.removePanelInstanceEventActionListener((Panel)
panelDescriptor,
            this);
    }
}

public void register(Set<PanelDescriptor> panelDescriptors,
EventDispatcher eventDispatcher) {
    for (PanelDescriptor panelDescriptor : panelDescriptors) {
        register(panelDescriptor, eventDispatcher);
    }
}

/**
 * @return all the PanelDescriptors registered with this manager
 */
protected Collection<PanelDescriptor> getRegisteredPanelDescriptors()
{
    return panelDescriptors.values();
}

protected Panel getPanelForEvent(NavigationEvent e) {
    Panel panel = null;
    if (e instanceof OpenPanelEvent) {
        OpenPanelEvent openEvent = (OpenPanelEvent) e;
        if (openEvent.getPanel() != null) {
            panel = openEvent.getPanel();
        } else {
            Class<?> targetType = openEvent.getTargetType();
            if (targetType != null) {
                while (targetType != null) {
                    panel = getPanel(openEvent.getPanelActionType(), targetType,
openEvent
                        .getPanelName(), openEvent.getTargetObject(),
openEvent.getSource());
                    if (panel != null) {
                        break;
                    }
                }
                if (targetType.getInterfaces() != null) {
                    for (Class<?> face : targetType.getInterfaces()) {
                        panel = getPanel(openEvent.getPanelActionType(), face,
openEvent
                            .getPanelName(), openEvent.getTargetObject(),
openEvent.getSource());
                        if (panel != null) {
                            if (panel != null) {
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
}

private Panel getPanel(String panelActionType, String face,
String panelName, Object targetObject,
String source) {
    panel = getPanel(openEvent.getPanelActionType(), face,
openEvent
    .getPanelName(), openEvent.getTargetObject(),
openEvent.getSource());
    if (panel != null) {
        break;
    }
}
if (panel != null) {
    break;
}
targetType = targetType.getSuperclass();
}
} else {
// panel with no content type
panel = getPanel(openEvent.getPanelActionType(), null, openEvent
    .getPanelName(), openEvent.getTargetObject(),
openEvent.getSource());
}
} else if (e instanceof ClosePanelEvent) {
ClosePanelEvent event = (ClosePanelEvent) e;
panel = event.getPanelToClose();
}
return panel;
}

/**
 * Get a panel responsible for handling events based on the supplied
 * criteria.<br>
 * The returned panel will be initialized via its setup() method, and
have
 * its target object set.<br>
 * If there is an exact match based on the supplied parameters then
that
 * panel will be returned. If there isn't an exact match the
following order
 * for matching will be used unless one of the parameters is
null:<br>
 * <ul>
 * <li>actionType, contentType, panelName, targetObject</li>
 * <li>actionType, contentType, panelName</li>
 * <li>contentType, panelName</li>
 * <li>panelName</li>
 * <li>actionType, contentType</li>
 * </ul>

```

```

* <br>
* In each case where the contentType is not null, the supertype and
* interfaces of the type will be substituted until a match is found
or the
* Object class is reached.
*
* @param actionType -
*           The type of action from the PanelActionType enum. if
this is
*           null PanelActionType.Unspecified will be substituted.
* @param contentType -
*           The type of content that will be supplied to the panel.
If
*           there isn't a panel for the exact type, a panel for the
*           closest supertype or interfaces will be substituted.
* @param panelName -
*           the name of the panel
* @param targetObject -
*           the object that the supplied panel will use to
initialize its
*           state
* @param destinationObject -
*           The destination to set on events fired from the panel
* @return
*/
protected Panel getPanel(PanelActionType actionType, Class<?>
contentType, String panelName,
Object targetObject, Object destinationObject) {
if (actionType == null) {
actionType = PanelActionType.Unspecified;
}
if (panelName != null) {
panelName = panelName.trim();
if ("".equals(panelName)) {
panelName = null;
}
}

// first try to find an existing panel with the targetObject
PanelDescriptor panelDescriptor = getPanelDescriptor(actionType,
contentType, panelName,
targetObject);
// then try for a panel or panel factory without the targetObject
if (panelDescriptor == null) {
panelDescriptor = getPanelDescriptor(actionType, contentType,
panelName, null);
}

```

```

Panel panel = null;
if (panelDescriptor != null) {
if (panelDescriptor instanceof Panel) {
panel = (Panel) panelDescriptor;
} else {
PanelFactory factory = (PanelFactory) panelDescriptor;
try {
panel = factory.newInstance();
// register the new panel for handling events for the
// target object instead of creating a new panel.
if (targetObject != null) {
register(panel, eventDispatcher, targetObject);
}
} catch (Exception e) {
log.error("failed to create new panel with noargs constructor and
factory ''"
+ factory + "': " + e, e);
}
}
}
if (panel != null) {
if (destinationObject != null) {
if (panel.getDestinationObject() != null && !
destinationObject.equals(panel.getDestinationObject())) {
// the panel is already configured
log.warn("the panel " + panel + " is already configured to send
events to destination " + panel.getDestinationObject() + ", not
changing destination to " + destinationObject);
} else {
panel.setDestinationObject(destinationObject);
}
}
panel.setTargetObject(targetObject);
if (!panel.isInitialized()) {
panel.setup();
}
}
return panel;
}

/**
 * NOTE: this is protected for testing purposes only, it should not
be
* called directly.<br>
* get the PanelDescriptor based on the precedence rules
*/

```

```

protected PanelDescriptor getPanelDescriptor(PanelActionType
actionType, Class<?> contentType,
String panelName, Object targetObject) {
PanelDescriptor panelDescriptor = null;

if (panelName != null) {
// actionType, contentType, panelName
panelDescriptor = getPanelDescriptorForContentType(actionType,
contentType, panelName,
targetObject);
// contentType, panelName
if (panelDescriptor == null) {
panelDescriptor =
getPanelDescriptorForContentType(PanelActionType.Unspecified,
contentType, panelName, targetObject);
}
// actionType, panelName
if (panelDescriptor == null) {
panelDescriptor = panelDescriptors.get(new PanelRegistryKey(
actionType, null, panelName, targetObject));
}

// panelName
if (panelDescriptor == null) {
panelDescriptor = panelDescriptors.get(new PanelRegistryKey(
PanelActionType.Unspecified, null, panelName, targetObject));
}
}
if ((panelDescriptor == null) &&
PanelActionType.Unspecified.equals(actionType)) {
// actionType, contentType
panelDescriptor = getPanelDescriptorForContentType(actionType,
contentType, panelName,
targetObject);
}
return panelDescriptor;
}

/**
 * NOTE: this is protected for testing purposes only, it should not
be
 * called directly.<br>
 * if contentType isn't null search for a PanelDescriptor by the
type,
 * supertypes and interfaces.
*/

```

```

protected PanelDescriptor
getPanelDescriptorForContentType(PanelActionType actionType,
Class<?> contentType, String panelName, Object targetObject) {
PanelDescriptor panelDescriptor = null;
if (contentType != null) {
while (contentType != null) {
log.debug(panelDescriptors);
panelDescriptor = panelDescriptors.get(new
PanelRegistryKey(actionType,
contentType, panelName, targetObject));
if (panelDescriptor != null) {
break;
}
contentType = contentType.getSuperclass();
}
}
return panelDescriptor;
}

public void dispose() {
for (PanelDescriptor panelDescriptor :
getRegisteredPanelDescriptors()) {
panelDescriptor.dispose();
}
}

@Override
public String toString() {
return getClass().getSimpleName();
}

private static class PanelRegistryKey {
private final PanelActionType actionType;
private final Class<?> contentType;
private final String panelName;
private final Object targetObject;

/**
 * Create a key for a given panel
 *
 * @param panelDescriptor
 */
protected PanelRegistryKey(PanelDescriptor panelDescriptor, Object
targetObject) {
this(panelDescriptor.getSupportedActionType(), panelDescriptor
.getSupportedContent(), panelDescriptor.getPanelName(),
targetObject);
}

```

```

}

/**
 * Create a key for the given constraints
 *
 * @param actionType
 * @param contentType
 * @param name
 */
protected PanelRegistryKey(PanelActionType actionType, Class<?>
contentType,
    String panelName, Object targetObject) {
    this.actionType = actionType;
    this.contentType = contentType;
    this.panelName = panelName;
    this.targetObject = targetObject;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((actionType == null) ? 0 :
actionType.hashCode());
    result = prime * result + ((contentType == null) ? 0 :
contentType.hashCode());
    result = prime * result + ((panelName == null) ? 0 :
panelName.hashCode());
    result = prime * result + ((targetObject == null) ? 0 :
targetObject.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final PanelRegistryKey other = (PanelRegistryKey) obj;
    if (actionType == null) {
        if (other.actionType != null) {
            return false;
        }
    } else if (!actionType.equals(other.actionType)) {
        return false;
    }
    if (contentType == null) {
        if (other.contentType != null) {
            return false;
        }
    } else if (!contentType.equals(other.contentType)) {
        return false;
    }
    if (panelName == null) {
        if (other.panelName != null) {
            return false;
        }
    } else if (!panelName.equals(other.panelName)) {
        return false;
    }
    if (targetObject == null) {
        if (other.targetObject != null) {
            return false;
        }
    } else if (!targetObject.equals(other.targetObject)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return getClass().getSimpleName() + "[actionType = " + actionType +
", contentType = "
        + contentType + ", panelName = " + panelName + ", targetObject =
"
        + targetObject + "]";
}
}
}

deleteactorcommand.java

/*
 * $Id: DeleteActorCommand.java,v 1.1 2008/11/20 09:55:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Actor;

/**
 * @author ron
 */
public interface DeleteActorCommand extends EditCommand {

    /**
     * Set the actor to delete.
     *
     * @param actor
     */
    public void setActor(Actor actor);
}

```

deleteactorcommandimpl.java

```

/*
 * $Id: DeleteActorCommandImpl.java,v 1.8 2009/03/30 11:54:27 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand;
;
```

```

import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.UseCase;
import
edu.harvard.fas.rregan.requel.project.command.DeleteActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveActorFromActorCont
ainerCommand;
import edu.harvard.fas.rregan.requel.project.impl.AddActorPosition;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

    /**
     * Delete a actor from a project, cleaning up references from other
     * project
     * entities, actor relations and annotations.
     *
     * @author ron
     */
    @Controller("deleteActorCommand")
    @Scope("prototype")
    public class DeleteActorCommandImpl extends AbstractEditProjectCommand
    implements
        DeleteActorCommand {

        private Actor actor;

        /**
         * @param assistantManager
         * @param userRepository
         * @param projectRepository
         * @param projectCommandFactory
         * @param annotationCommandFactory
         * @param commandHandler
         */
        @Autowired
        public DeleteActorCommandImpl(AssistantFacade assistantManager,
                                      UserRepository userRepository,

```

```

    ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
        annotationCommandFactory, commandHandler);
}

@Override
public void setActor(Actor actor) {
    this.actor = actor;
}

protected Actor getActor() {
    return actor;
}

@Override
public void execute() throws Exception {
    Actor actor = getRepository().get(getActor());
    User editedBy = getRepository().get(getEditedBy());
    Set<Annotation> annotations = new
HashSet<Annotation>(actor.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
            .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
        removeAnnotationFromAnnotatableCommand.setAnnotatable(actor);
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
    }
    // remove this entity as a referer to any terms
    for (GlossaryTerm term :
actor.getProjectOrDomain().getGlossaryTerms()) {
        if (term.getReferers().contains(actor)) {
            term.getReferers().remove(actor);
        }
    }
    Set<ActorContainer> actorReferers = new
HashSet<ActorContainer>(actor.getReferers());
    for (ActorContainer actorContainer : actorReferers) {
        RemoveActorFromActorContainerCommand
removeActorFromActorContainerCommand = getProjectCommandFactory()
            .newRemoveActorFromActorContainerCommand();
}

```

```

removeActorFromActorContainerCommand.setEditedBy(editedBy);
removeActorFromActorContainerCommand.setActor(actor);
removeActorFromActorContainerCommand.setActorContainer(actorContain
er);
getCommandHandler().execute(removeActorFromActorContainerCommand);
if (actorContainer instanceof UseCase) {
    UseCase useCase = (UseCase) actorContainer;
    if (useCase.getPrimaryActor().equals(actor)) {
        throw new RuntimeException("The actor '\"' +
actor.getDescription()
            + '\"' is the primary actor for '\"' + useCase.getDescription()
            + '\"' and cannot be deleted unless the use case is deleted
first.");
    }
}
try {
    AddActorPosition actorPosition =
getRepository().findAddActorPosition(
        actor.getProjectOrDomain(), actor.getName());
    DeletePositionCommand deletePositionCommand =
getAnnotationCommandFactory()
        .newDeletePositionCommand();
    deletePositionCommand.setEditedBy(getEditedBy());
    deletePositionCommand.setPosition(actorPosition);
    getCommandHandler().execute(deletePositionCommand);
} catch (NoSuchPositionException e) {
}
actor.getProjectOrDomain().getActors().remove(actor);
getRepository().delete(actor);
}
}

```

deleteargumentcommand.java

```

/*
 * $Id: DeleteArgumentCommand.java,v 1.1 2009/02/13 12:08:01 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.command.EditCommand;

```

```

/**
 * Delete an argument of a position from the system.
 *
 * @author ron
 */
public interface DeleteArgumentCommand extends EditCommand {

    /**
     * Set the argument to delete.
     *
     * @param argument
     */
    public void setArgument(Argument argument);
}

```

deleteargumentcommandimpl.java

```

/*
 * $Id: DeleteArgumentCommandImpl.java,v 1.1 2009/02/13 12:08:00
rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteArgumentCommand
;

/**
 * @author ron
 */
@Controller("deleteArgumentCommand")
@Scope("prototype")

```

```

public class DeleteArgumentCommandImpl extends AbstractEditCommand
implements DeleteArgumentCommand {

    private Argument argument;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public DeleteArgumentCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    /**
     * @see
edu.harvard.fas.rregan.requel.annotation.command.DeleteArgumentCommand
#setArgument(edu.harvard.fas.rregan.requel.annotation.Argument)
     */
    @Override
    public void setArgument(Argument argument) {
        this.argument = argument;
    }

    protected Argument getArgument() {
        return argument;
    }

    /**
     * @see edu.harvard.fas.rregan.command.Command#execute()
     */
    @Override
    public void execute() throws Exception {
        Argument argument = getRepository().get(getArgument());
        Position position = getRepository().get(argument.getPosition());
        position.getArguments().remove(argument);
        getRepository().delete(argument);
    }
}

```

deletedentityevent.java

```
/*
 * $Id: DeletedEntityEvent.java,v 1.1 2009/02/15 09:31:37 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.navigation.event;

import edu.harvard.fas.rregan.uiframework.panel.Panel;

/**
 * @author ron
 */
public class DeletedEntityEvent extends UpdateEntityEvent {
    static final long serialVersionUID = 0;

    /**
     * @param source
     * @param panelToClose
     * @param updatedObject
     */
    public DeletedEntityEvent(Object source, Panel panelToClose, Object
updatedObject) {
        super(source, panelToClose, updatedObject);
    }

    /**
     * @param source
     * @param command
     * @param panelToClose
     * @param updatedObject
     */
    public DeletedEntityEvent(Object source, String command, Panel
panelToClose,
        Object updatedObject) {
        super(source, command, panelToClose, updatedObject);
    }

    /**
     * @param source
     * @param updatedObject
     */
    public DeletedEntityEvent(Panel source, Object updatedObject) {
        super(source, updatedObject);
    }
}
```

```
}
```

deleteglossarytermcommand.java

```
/*
 * $Id: DeleteGlossaryTermCommand.java,v 1.1 2008/11/20 09:55:13
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;

/**
 * @author ron
 */
public interface DeleteGlossaryTermCommand extends EditCommand {

    /**
     * Set the glossaryTerm to delete.
     *
     * @param glossaryTerm
     */
    public void setGlossaryTerm(GlossaryTerm glossaryTerm);
}
```

deleteglossarytermcommandimpl.java

```
/*
 * $Id: DeleteGlossaryTermCommandImpl.java,v 1.5 2009/03/30 11:54:31
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
```

```

import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand;
import edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromAnnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.DeleteGlossaryTermCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import edu.harvard.fas.rregan.requel.project.impl.GlossaryTermImpl;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Delete a glossaryTerm from a project, cleaning up references from
 * other
 * project entities, glossaryTerm relations and annotations.
 *
 * @author ron
 */
@Controller("deleteGlossaryTermCommand")
@Scope("prototype")
public class DeleteGlossaryTermCommandImpl extends
AbstractEditProjectCommand implements
DeleteGlossaryTermCommand {

    private GlossaryTerm glossaryTerm;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
}

```

```

    @Autowired
    public DeleteGlossaryTermCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
        annotationCommandFactory, commandHandler);
}

@Override
public void setGlossaryTerm(GlossaryTerm glossaryTerm) {
    this.glossaryTerm = glossaryTerm;
}

protected GlossaryTerm getGlossaryTerm() {
    return glossaryTerm;
}

@Override
public void execute() throws Exception {
    GlossaryTerm glossaryTerm = getRepository().get(getGlossaryTerm());
    Set<Annotation> annotations = new
HashSet<Annotation>(glossaryTerm.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
            .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(getEditedBy());
        removeAnnotationFromAnnotatableCommand.setAnnotatable(glossaryTerm)
        ;
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
        ;
    }
    glossaryTerm.getReferers().clear();
    // if this is the canonical term for other glossary terms, the
    // other terms must be updated to set the canonical term to null.
    for (GlossaryTerm alternateTerm : glossaryTerm.getAlternateTerms())
    {
        ((GlossaryTermImpl) alternateTerm).setCanonicalTerm(null);
    }
    try {
        AddGlossaryTermPosition addGlossaryTermPosition =
getRepository()

```

```

    .findAddGlossaryTermPosition(glossaryTerm.getProjectOrDomain(),
        glossaryTerm.getName());
    DeletePositionCommand deletePositionCommand =
getAnnotationCommandFactory()
    .newDeletePositionCommand();
    deletePositionCommand.setEditedBy(getEditedBy());
    deletePositionCommand.setPosition(addGlossaryTermPosition);
    getCommandHandler().execute(deletePositionCommand);
} catch (NoSuchEntityException e) {

}
glossaryTerm.getProjectOrDomain().getGlossaryTerms().remove(glossary
Term);
getRepository().delete(glossaryTerm);
}

}


```

deletegoalcommand.java

```

/*
 * $Id: DeleteGoalCommand.java,v 1.1 2008/11/05 18:25:52 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Goal;

/**
 *
 * @author ron
 */
public interface DeleteGoalCommand extends EditCommand {

    /**
     * Set the goal to delete.
     * @param goal
     */
    public void setGoal(Goal goal);
}


```

deletegoalcommandimpl.java

```

/*
```

```

 * $Id: DeleteGoalCommandImpl.java,v 1.6 2009/03/30 11:54:26 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.DeleteGoalCommand;
import edu.harvard.fas.rregan.requel.project.command.DeleteGoalRelationComman
d;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.command.RemoveGoalFromGoalContai
nerCommand;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

    /**
     * Delete a goal from a project, cleaning up references from other
project
     * entities, goal relations and annotations.
     *
     * @author ron

```

```

/*
@Controller("deleteGoalCommand")
@Scope("prototype")
public class DeleteGoalCommandImpl extends AbstractEditProjectCommand
implements DeleteGoalCommand {

    private Goal goal;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public DeleteGoalCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository,
        ProjectRepository projectRepository, ProjectCommandFactory
        projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public void setGoal(Goal goal) {
        this.goal = goal;
    }

    protected Goal getGoal() {
        return goal;
    }

    @Override
    public void execute() throws Exception {
        Goal goal = getRepository().get(getGoal());
        User editedBy = getRepository().get(getEditedBy());
        Set<Annotation> annotations = new
        HashSet<Annotation>(goal.getAnnotations());
        for (Annotation annotation : annotations) {
            RemoveAnnotationFromAnnotatableCommand
            removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
                .newRemoveAnnotationFromAnnotatableCommand();

```

```

            removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
            removeAnnotationFromAnnotatableCommand.setAnnotatable(goal);
            removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
            getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
        ;
    }
    // remove this entity as a referer to any terms
    for (GlossaryTerm term :
        goal.getProjectOrDomain().getGlossaryTerms()) {
        if (term.getReferers().contains(goal)) {
            term.getReferers().remove(goal);
        }
    }
    Set<GoalContainer> goalReferers = new
    HashSet<GoalContainer>(goal.getReferers());
    for (GoalContainer goalContainer : goalReferers) {
        RemoveGoalFromGoalContainerCommand
        removeGoalFromGoalContainerCommand = getProjectCommandFactory()
            .newRemoveGoalFromGoalContainerCommand();
        removeGoalFromGoalContainerCommand.setEditedBy(editedBy);
        removeGoalFromGoalContainerCommand.setGoal(goal);
        removeGoalFromGoalContainerCommand.setGoalContainer(goalContainer);
        getCommandHandler().execute(removeGoalFromGoalContainerCommand);
    }
    goal.getProjectOrDomain().getGoals().remove(goal);
    Set<GoalRelation> goalRelations = new
    HashSet<GoalRelation>(goal.getRelationsFromThisGoal());
    goalRelations.addAll(goal.getRelationsToThisGoal());
    for (GoalRelation goalRelation : goalRelations) {
        DeleteGoalRelationCommand deleteGoalRelationCommand =
        getProjectCommandFactory()
            .newDeleteGoalRelationCommand();
        deleteGoalRelationCommand.setEditedBy(editedBy);
        deleteGoalRelationCommand.setGoalRelation(goalRelation);
        getCommandHandler().execute(deleteGoalRelationCommand);
    }
    getRepository().delete(goal);
}
}

```

deletegoalrelationcommand.java

```

/*
 * $Id: DeleteGoalRelationCommand.java,v 1.1 2008/11/05 18:25:52
 * rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.GoalRelation;

/**
 * Delete a goal relationship between two goals.
 * @author ron
 */
public interface DeleteGoalRelationCommand extends EditCommand {

    /**
     * set the goal relation to delete.
     * @param goalRelation
     */
    public void setGoalRelation(GoalRelation goalRelation);
}

```

deletegoalrelationcommandimpl.java

```

/*
 * $Id: DeleteGoalRelationCommandImpl.java,v 1.4 2009/03/30 11:54:27
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;

```

```

import
edu.harvard.fas.rregan.requel.project.command.DeleteGoalRelationCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 *
 * @author ron
 */
@Controller("deleteGoalRelationCommand")
@Scope("prototype")
public class DeleteGoalRelationCommandImpl extends
AbstractEditProjectCommand implements DeleteGoalRelationCommand {

    private GoalRelation goalRelation;

    @Autowired
    public DeleteGoalRelationCommandImpl(AssistantFacade
assistantManager, UserRepository userRepository, ProjectRepository
projectRepository, ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
projectCommandFactory, annotationCommandFactory, commandHandler);
    }

    @Override
    public void setGoalRelation(GoalRelation goalRelation) {
        this.goalRelation = goalRelation;
    }

    protected GoalRelation getGoalRelation() {
        return goalRelation;
    }

    @Override
    public void execute() throws Exception {
        GoalRelation goalRelation = getRepository().get(getGoalRelation());
        User editedBy = getRepository().get(getEditedBy());
        Set<Annotation> annotations = new
HashSet<Annotation>(goalRelation.getAnnotations());
        for (Annotation annotation : annotations) {

```

```

    RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand =
getAnnotationCommandFactory().newRemoveAnnotationFromAnnotatableCommand();
    removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
    removeAnnotationFromAnnotatableCommand.setAnnotatable(goalRelation);
;
    removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
;
}
goalRelation.getFromGoal().getRelationsFromThisGoal().remove(goalRelation);
goalRelation.getToGoal().getRelationsToThisGoal().remove(goalRelation);
getRepository().delete(goalRelation);
}

}

```

deleteissuecommand.java

```

/*
 * $Id: DeleteIssueCommand.java,v 1.1 2009/02/13 12:08:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Delete an issue, its positions and all the arguments of the
positions. Update
 * all the annotatables that refer to the issue and remove the
reference.
 *
 * @author ron
 */
public interface DeleteIssueCommand extends EditCommand {

/**
 * Set the issue to delete.
 *
 * @param issue
 */

```

```

    public void setIssue(Issue issue);
}

```

deleteissuecommandimpl.java

```

/*
 * $Id: DeleteIssueCommandImpl.java,v 1.2 2009/02/15 09:31:35 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteIssueCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand
;

/**
 * @author ron
 */
@Controller("deleteIssueCommand")
@Scope("prototype")
public class DeleteIssueCommandImpl extends AbstractEditCommand
implements DeleteIssueCommand {

    private Issue issue;

    /**
     * @param commandHandler
     */

```

```

* @param annotationCommandFactory
* @param repository
*/
@.Autowired
public DeleteIssueCommandImpl(CommandHandler commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository repository) {
    super(commandHandler, annotationCommandFactory, repository);
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.DeleteIssueCommand#se
tIssue(edu.harvard.fas.rregan.requel.annotation.Issue)
 */
@Override
public void setIssue(Issue issue) {
    this.issue = issue;
}

protected Issue getIssue() {
    return issue;
}

/***
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Issue issue = getRepository().get(getIssue());
    for (Annotatable annotatable : issue.getAnnotatables()) {
        annotatable.getAnnotations().remove(issue);
    }
    Set<Position> positions = new
    HashSet<Position>(issue.getPositions());
    issue.getPositions().clear();
    for (Position position : positions) {
        position.getIssues().remove(issue);
        if (position.getIssues().isEmpty()) {
            DeletePositionCommand deletePositionCommand =
            getAnnotationCommandFactory()
                .newDeletePositionCommand();
            deletePositionCommand.setPosition(position);
            deletePositionCommand.setEditedBy(getEditedBy());
            getCommandHandler().execute(deletePositionCommand);
        }
    }
}

```

```

        getRepository().delete(issue);
    }
}

```

deletenotecommand.java

```

/*
 * $Id: DeleteNoteCommand.java,v 1.1 2009/02/13 12:08:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Delete a note from the system. Update all the annotatables that
refer to the
 * note and remove the reference.
 *
 * @author ron
 */
public interface DeleteNoteCommand extends EditCommand {

    /**
     * Set the note to delete.
     *
     * @param note
     */
    public void setNote(Note note);
}

```

deletenotecommandimpl.java

```

/*
 * $Id: DeleteNoteCommandImpl.java,v 1.1 2009/02/13 12:07:59 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.DeleteNoteCommand;

/**
 * @author ron
 */
@Controller("deleteNoteCommand")
@Scope("prototype")
public class DeleteNoteCommandImpl extends AbstractEditCommand
implements DeleteNoteCommand {

    private Note note;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public DeleteNoteCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    /**
     * @see
     * @see edu.harvard.fas.rregan.requel.annotation.command.DeleteNoteCommand#set
     * Note(edu.harvard.fas.rregan.requel.annotation.Note)
     */
    @Override
    public void setNote(Note note) {
        this.note = note;
    }

    protected Note getNote() {
        return note;
    }
}

```

```

    /**
     * @see edu.harvard.fas.rregan.command.Command#execute()
     */
    @Override
    public void execute() throws Exception {
        Note note = getRepository().get(getNote());
        for (Annotatable annotatable : note.getAnnotatables()) {
            annotatable.getAnnotations().remove(note);
        }
        getRepository().delete(note);
    }
}

```

deletepositioncommand.java

```

/*
 * $Id: DeletePositionCommand.java,v 1.1 2009/02/13 12:08:02 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Delete a position from an issue and all its arguments.
 *
 * @author ron
 */
public interface DeletePositionCommand extends EditCommand {

    /**
     * Set the position to delete.
     *
     * @param position
     */
    public void setPosition(Position position);
}

```

deletepositioncommandimpl.java

```

/*

```

```

* $Id: DeletePositionCommandImpl.java,v 1.3 2009/02/23 09:39:55
rregan Exp $
* Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
*/

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteArgumentCommand ;
import
edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand ;

/**
 * @author ron
 */
@Controller("deletePositionCommand")
@Scope("prototype")
public class DeletePositionCommandImpl extends AbstractEditCommand
implements DeletePositionCommand {

    private Position position;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public DeletePositionCommandImpl(CommandHandler commandHandler,

```

```

        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    /**
     * @see
     edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand
#setPosition(edu.harvard.fas.rregan.requel.annotation.Position)
 */
@Override
public void setPosition(Position position) {
    this.position = position;
}

protected Position getPosition() {
    return position;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Position position = getRepository().get(getPosition());
    for (Issue issue : position.getIssues()) {
        issue.getPositions().remove(position);
        if (position.equals(issue.getResolvedByPosition())) {
            issue.unresolve();
        }
    }
    Set<Argument> arguments = new
HashSet<Argument>(position getArguments());
    position getArguments().clear();
    for (Argument argument : arguments) {
        DeleteArgumentCommand deleteArgumentCommand =
getAnnotationCommandFactory()
            .newDeleteArgumentCommand();
        deleteArgumentCommand.setArgument(argument);
        deleteArgumentCommand.setEditedBy(getEditedBy());
        getCommandHandler().execute(deleteArgumentCommand);
    }
    getRepository().delete(position);
}
}

```

deletereportgeneratorcommand.java

```
/*
 * $Id: DeleteReportGeneratorCommand.java,v 1.1 2008/11/20 09:55:15
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.project.ReportGenerator;

/**
 * @author ron
 */
public interface DeleteReportGeneratorCommand extends EditCommand {

    /**
     * Set the reportGenerator to delete.
     *
     * @param reportGenerator
     */
    public void setReportGenerator(ReportGenerator reportGenerator);
}
```

deletereportgeneratorcommandimpl.java

```
/*
 * $Id: DeleteReportGeneratorCommandImpl.java,v 1.3 2009/03/30
11:54:26 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
```

```
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import
edu.harvard.fas.rregan.requel.project.command.DeleteReportGeneratorCom
mand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

    /**
     * Delete a reportGenerator from a project, cleaning up references
     * from other
     * project entities, reportGenerator relations and annotations.
     *
     * @author ron
     */
    @Controller("deleteReportGeneratorCommand")
    @Scope("prototype")
    public class DeleteReportGeneratorCommandImpl extends
AbstractEditProjectCommand implements
DeleteReportGeneratorCommand {

    private ReportGenerator reportGenerator;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public DeleteReportGeneratorCommandImpl(AssistantFacade
assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
```

```

    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

@Override
public void setReportGenerator(ReportGenerator reportGenerator) {
    this.reportGenerator = reportGenerator;
}

protected ReportGenerator getReportGenerator() {
    return reportGenerator;
}

@Override
public void execute() throws Exception {
    ReportGenerator reportGenerator =
getRepository().get(getReportGenerator());
    User editedBy = getRepository().get(getEditedBy());
    Set<Annotation> annotations = new
HashSet<Annotation>(reportGenerator.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
        .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
        removeAnnotationFromAnnotatableCommand.setAnnotatable(reportGenerat
or);
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
    }
    reportGenerator.getProjectOrDomain().getReportGenerators().remove(re
portGenerator);
    getRepository().delete(reportGenerator);
}
}

```

deletescenariocommand.java

```

/*
 * $Id: DeleteScenarioCommand.java,v 1.1 2008/11/20 09:55:14 rregan
Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Scenario;

/**
 * @author ron
 */
public interface DeleteScenarioCommand extends EditCommand {

    /**
     * Set the scenario to delete.
     *
     * @param scenario
     */
    public void setScenario(Scenario scenario);
}

```

deletescenariocommandimpl.java

```

/*
 * $Id: DeleteScenarioCommandImpl.java,v 1.4 2009/03/30 11:54:29
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;

```

```

import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.UseCase;
import
edu.harvard.fas.rregan.requel.project.command.DeleteScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Delete a scenario from a project, cleaning up references from other
project
 * entities, scenario relations and annotations.
 *
 * @author ron
 */
@Controller("deleteScenarioCommand")
@Scope("prototype")
public class DeleteScenarioCommandImpl extends
AbstractEditProjectCommand implements
DeleteScenarioCommand {

    private Scenario scenario;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public DeleteScenarioCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public void setScenario(Scenario scenario) {
        this.scenario = scenario;
    }

    protected Scenario getScenario() {
        return scenario;
    }

    @Override
    public void execute() throws Exception {
        Scenario scenario = getRepository().get(getScenario());
        User editedBy = getRepository().get(getEditedBy());
        Set<Annotation> annotations = new
HashSet<Annotation>(scenario.getAnnotations());
        for (Annotation annotation : annotations) {
            RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
                .newRemoveAnnotationFromAnnotatableCommand();
            removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
            removeAnnotationFromAnnotatableCommand.setAnnotatable(scenario);
            removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
            getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
;
        }
        // remove this entity as a referer to any terms
        for (GlossaryTerm term :
scenario.getProjectOrDomain().getGlossaryTerms()) {
            if (term.getReferers().contains(scenario)) {
                term.getReferers().remove(scenario);
            }
            Set<Scenario> scenarioReferers = new
HashSet<Scenario>(scenario.getUsingScenarios());
            for (Scenario scenarioReferer : scenarioReferers) {
                scenarioReferer.getSteps().remove(scenario);
            }
            Set<UseCase> scenarioUseCaseReferers = new
HashSet<UseCase>(scenario.getUsingUseCases());
            for (UseCase usecase : scenarioUseCaseReferers) {
                // TODO: delete the usecase or set a new empty scenario on the
                // usecase.
            }
            scenario.getProjectOrDomain().getScenarios().remove(scenario);
            getRepository().delete(scenario);
        }
    }
}

```

deletescenariostepcommand.java

```
/*
 * $Id: DeleteScenarioStepCommand.java,v 1.1 2008/11/20 09:55:15
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Step;

/**
 * @author ron
 */
public interface DeleteScenarioStepCommand extends EditCommand {

    /**
     * Set the scenarioStep to delete.
     *
     * @param scenarioStep
     */
    public void setScenarioStep(Step scenarioStep);
}
```

deletescenariostepcommandimpl.java

```
/*
 * $Id: DeleteScenarioStepCommandImpl.java,v 1.4 2009/03/30 11:54:26
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
```

```
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.requel.project.command.DeleteScenarioStepComman
d;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Delete a scenarioStep from a project, cleaning up references from
other
 * project entities, scenarioStep relations and annotations.
 *
 * @author ron
 */
@Controller("deleteScenarioStepCommand")
@Scope("prototype")
public class DeleteScenarioStepCommandImpl extends
AbstractEditProjectCommand implements
DeleteScenarioStepCommand {

    private Step scenarioStep;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public DeleteScenarioStepCommandImpl(AssistantFacade
assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
```

```

ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

@Override
public void setScenarioStep(Step scenarioStep) {
    this.scenarioStep = scenarioStep;
}

protected Step getScenarioStep() {
    return scenarioStep;
}

@Override
public void execute() throws Exception {
    Step scenarioStep = getRepository().get(getScenarioStep());
    User editedBy = getRepository().get(getEditedBy());
    Set<Annotation> annotations = new
HashSet<Annotation>(scenarioStep.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
        .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
        removeAnnotationFromAnnotatableCommand.setAnnotatable(scenarioStep)
    ;
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
    ;
    }
    // remove this entity as a referer to any terms
    for (GlossaryTerm term :
scenarioStep.getProjectOrDomain().getGlossaryTerms()) {
        if (term.getReferers().contains(scenarioStep)) {
            term.getReferers().remove(scenarioStep);
        }
    }
    Set<Scenario> scenarioReferers = new
HashSet<Scenario>(scenarioStep.getUsingScenarios());
    for (Scenario scenarioReferer : scenarioReferers) {
        scenarioReferer.getSteps().remove(scenarioStep);
    }
    getRepository().delete(scenarioStep);
}

```

```

    }
}

```

deletestakeholdercommand.java

```

/*
 * $Id: DeleteStakeholderCommand.java,v 1.1 2008/11/20 09:55:15 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Stakeholder;

/**
 * @author ron
 */
public interface DeleteStakeholderCommand extends EditCommand {

/**
 * Set the stakeholder to delete.
 *
 * @param stakeholder
 */
public void setStakeholder(Stakeholder stakeholder);
}

```

deletestakeholdercommandimpl.java

```

/*
 * $Id: DeleteStakeholderCommandImpl.java,v 1.5 2009/03/30 11:54:26
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

```

```

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.command.DeleteStakeholderCommand
;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Delete a stakeholder from a project, cleaning up references from
other
 * project entities, stakeholder relations and annotations.
 */
@Author(ron)
*/
@Controller("deleteStakeholderCommand")
@Scope("prototype")
public class DeleteStakeholderCommandImpl extends
AbstractEditProjectCommand implements
DeleteStakeholderCommand {

    private Stakeholder stakeholder;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public DeleteStakeholderCommandImpl(AssistantFacade assistantManager,

```

```

        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public void setStakeholder(Stakeholder stakeholder) {
        this.stakeholder = stakeholder;
    }

    protected Stakeholder getStakeholder() {
        return stakeholder;
    }

    @Override
    public void execute() throws Exception {
        Stakeholder stakeholder = getRepository().get(getStakeholder());
        Set<Annotation> annotations = new
        HashSet<Annotation>(stakeholder.getAnnotations());
        for (Annotation annotation : annotations) {
            RemoveAnnotationFromAnnotatableCommand
            removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
                .newRemoveAnnotationFromAnnotatableCommand();
            removeAnnotationFromAnnotatableCommand.setEditedBy(getEditedBy());
            removeAnnotationFromAnnotatableCommand.setAnnotatable(stakeholder);
            removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
            getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
        }
        // remove this entity as a referer to any terms
        for (GlossaryTerm term :
            stakeholder.getProjectOrDomain().getGlossaryTerms()) {
            if (term.getReferers().contains(stakeholder)) {
                term.getReferers().remove(stakeholder);
            }
            stakeholder.getProjectOrDomain().getStakeholders().remove(stakeholde
r);
            if (stakeholder.getUser() != null) {
                stakeholder.getUser().getRoleForType(ProjectUserRole.class).getActi
veProjects().remove(
                    stakeholder.getProjectOrDomain());
            }
        }
    }
}

```

```

if (stakeholder.getTeam() != null) {
    stakeholder.getTeam().getMembers().remove(stakeholder);
}
for (GlossaryTerm term :
stakeholder.getProjectOrDomain().getGlossaryTerms()) {
    term.getReferers().remove(stakeholder);
}
for (Goal goal : stakeholder.getGoals()) {
    goal.getReferers().remove(stakeholder);
}
getRepository().delete(stakeholder);
}
}

```

deletestorycommand.java

```

/*
 * $Id: DeleteStoryCommand.java,v 1.1 2008/11/20 09:55:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Story;

/**
 * @author ron
 */
public interface DeleteStoryCommand extends EditCommand {

    /**
     * Set the story to delete.
     *
     * @param story
     */
    public void setStory(Story story);
}

```

deletestorycommandimpl.java

```

/*
 * $Id: DeleteStoryCommandImpl.java,v 1.4 2009/03/30 11:54:31 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromAnnotatableCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.command.DeleteStoryCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.command.RemoveStoryFromStoryContainerCommand;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Delete a story from a project, cleaning up references from other
 * project
 * entities, story relations and annotations.
 *
 * @author ron
 */
@Controller("deleteStoryCommand")
@Scope("prototype")
public class DeleteStoryCommandImpl extends AbstractEditProjectCommand
implements
    DeleteStoryCommand {

    private Story story;
}

```

```

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@.Autowired
public DeleteStoryCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

@Override
public void setStory(Story story) {
    this.story = story;
}

protected Story getStory() {
    return story;
}

@Override
public void execute() throws Exception {
    Story story = getRepository().get(getStory());
    User editedBy = getRepository().get(getEditedBy());
    Set<Annotation> annotations = new
HashSet<Annotation>(story.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
        .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
        removeAnnotationFromAnnotatableCommand.setAnnotatable(story);
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
    }
    // remove this entity as a referer to any terms
}

```

```

        for (GlossaryTerm term :
story.getProjectOrDomain().getGlossaryTerms()) {
            if (term.getReferers().contains(story)) {
                term.getReferers().remove(story);
            }
        }
        Set<StoryContainer> storyReferers = new
HashSet<StoryContainer>(story.getReferers());
        for (StoryContainer storyContainer : storyReferers) {
            RemoveStoryFromStoryContainerCommand
removeStoryFromStoryContainerCommand = getProjectCommandFactory()
            .newRemoveStoryFromStoryContainerCommand();
            removeStoryFromStoryContainerCommand.setEditedBy(editedBy);
            removeStoryFromStoryContainerCommand.setStory(story);
            removeStoryFromStoryContainerCommand.setStoryContainer(storyContain
er);
            getCommandHandler().execute(removeStoryFromStoryContainerCommand);
        }
        story.getProjectOrDomain().getStories().remove(story);
        getRepository().delete(story);
    }
}

```

deleteusecasecommand.java

```

/*
 * $Id: DeleteUseCaseCommand.java,v 1.1 2008/11/20 09:55:13 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.UseCase;

/**
 * @author ron
 */
public interface DeleteUseCaseCommand extends EditCommand {

    /**
     * Set the usecase to delete.
     *
     * @param usecase
     */

```

```
    public void setUseCase(UseCase usecase);  
}
```

deleteusecasecommandimpl.java

```
/*  
 * $Id: DeleteUseCaseCommandImpl.java,v 1.5 2009/03/30 11:54:27 rregan  
Exp $  
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.  
 */  
package edu.harvard.fas.rregan.requel.project.impl.command;  
  
import java.util.HashSet;  
import java.util.Set;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Scope;  
import org.springframework.stereotype.Controller;  
  
import edu.harvard.fas.rregan.command.CommandHandler;  
import edu.harvard.fas.rregan.requel.annotation.Annotation;  
import  
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact  
ory;  
import  
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA  
nnotatableCommand;  
import edu.harvard.fas.rregan.requel.project.Actor;  
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;  
import edu.harvard.fas.rregan.requel.project.Goal;  
import edu.harvard.fas.rregan.requel.project.ProjectRepository;  
import edu.harvard.fas.rregan.requel.project.Scenario;  
import edu.harvard.fas.rregan.requel.project.Story;  
import edu.harvard.fas.rregan.requel.project.UseCase;  
import  
edu.harvard.fas.rregan.requel.project.command.DeleteUseCaseCommand;  
import  
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;  
import  
edu.harvard.fas.rregan.requel.project.command.RemoveActorFromActorCont  
ainerCommand;  
import  
edu.harvard.fas.rregan.requel.project.command.RemoveGoalFromGoalContai  
nerCommand;
```

```
import  
edu.harvard.fas.rregan.requel.project.command.RemoveStoryFromStoryCont  
ainerCommand;  
import  
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;  
import edu.harvard.fas.rregan.requel.user.User;  
import edu.harvard.fas.rregan.requel.user.UserRepository;  
  
/**  
 * Delete a usecase from a project, cleaning up references from other  
project  
 * entities, usecase relations and annotations.  
 *  
 * @author ron  
 */  
@Controller("deleteUseCaseCommand")  
@Scope("prototype")  
public class DeleteUseCaseCommandImpl extends  
AbstractEditProjectCommand implements  
DeleteUseCaseCommand {  
  
    private UseCase usecase;  
  
    /**  
     * @param assistantManager  
     * @param userRepository  
     * @param projectRepository  
     * @param projectCommandFactory  
     * @param annotationCommandFactory  
     * @param commandHandler  
     */  
    @Autowired  
    public DeleteUseCaseCommandImpl(AssistantFacade assistantManager,  
        UserRepository userRepository, ProjectRepository projectRepository,  
        ProjectCommandFactory projectCommandFactory,  
        AnnotationCommandFactory annotationCommandFactory, CommandHandler  
commandHandler) {  
        super(assistantManager, userRepository, projectRepository,  
        projectCommandFactory,  
        annotationCommandFactory, commandHandler);  
    }  
  
    @Override  
    public void setUseCase(UseCase usecase) {  
        this.usecase = usecase;  
    }  
}
```

```

protected UseCase getUseCase() {
    return usecase;
}

@Override
public void execute() throws Exception {
    UseCase usecase = getRepository().get(getUseCase());
    User editedBy = getRepository().get(getEditedBy());
    Set<Annotation> annotations = new
    HashSet<Annotation>(usecase.getAnnotations());
    for (Annotation annotation : annotations) {
        RemoveAnnotationFromAnnotatableCommand
            removeAnnotationFromAnnotatableCommand = getAnnotationCommandFactory()
                .newRemoveAnnotationFromAnnotatableCommand();
        removeAnnotationFromAnnotatableCommand.setEditedBy(editedBy);
        removeAnnotationFromAnnotatableCommand.setAnnotatable(usecase);
        removeAnnotationFromAnnotatableCommand.setAnnotation(annotation);
        getCommandHandler().execute(removeAnnotationFromAnnotatableCommand)
    }
    // remove this entity as a referer to any terms
    for (GlossaryTerm term :
        usecase.getProjectOrDomain().getGlossaryTerms()) {
        if (term.getReferers().contains(usecase)) {
            term.getReferers().remove(usecase);
        }
    }
    Set<Actor> actors = new HashSet<Actor>(usecase.getActors());
    actors.add(usecase.getPrimaryActor());
    for (Actor actor : actors) {
        RemoveActorFromActorContainerCommand
            removeActorFromActorContainerCommand = getProjectCommandFactory()
                .newRemoveActorFromActorContainerCommand();
        removeActorFromActorContainerCommand.setActor(actor);
        removeActorFromActorContainerCommand.setActorContainer(usecase);
        removeActorFromActorContainerCommand.setEditedBy(getEditedBy());
        getCommandHandler().execute(removeActorFromActorContainerCommand);
    }
    Set<Goal> goals = new HashSet<Goal>(usecase.getGoals());
    for (Goal goal : goals) {
        RemoveGoalFromGoalContainerCommand
            removeGoalFromGoalContainerCommand = getProjectCommandFactory()
                .newRemoveGoalFromGoalContainerCommand();
        removeGoalFromGoalContainerCommand.setGoal(goal);
        removeGoalFromGoalContainerCommand.setGoalContainer(usecase);
        removeGoalFromGoalContainerCommand.setEditedBy(getEditedBy());
        getCommandHandler().execute(removeGoalFromGoalContainerCommand);
    }
}

```

```

    }
    Set<Story> stories = new HashSet<Story>(usecase.getStories());
    for (Story story : stories) {
        RemoveStoryFromStoryContainerCommand
            removeStoryFromStoryContainerCommand = getProjectCommandFactory()
                .newRemoveStoryFromStoryContainerCommand();
        removeStoryFromStoryContainerCommand.setStory(story);
        removeStoryFromStoryContainerCommand.setStoryContainer(usecase);
        removeStoryFromStoryContainerCommand.setEditedBy(getEditedBy());
        getCommandHandler().execute(removeStoryFromStoryContainerCommand);
    }
    for (Scenario scenario :
        getProjectRepository().findScenariosUsedByUseCase(usecase)) {
        // TODO: add command RemoveUsecaseFromScenario
        scenario.getUsingUseCases().remove(usecase);
    }
    // TODO: delete the main scenario?
    usecase.getProjectOrDomain().getUseCases().remove(usecase);
    getRepository().delete(usecase);
}
}

```

dependencyprimaryverbfinder.java

```

/*
 * $Id: DependencyPrimaryVerbFinder.java,v 1.3 2009/03/05 08:50:47
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;

/**
 */

```

```

 * This uses the dependency information attached to the NLPText to get
the
 * primary verb of a SENTENCE, CLAUSE, or PHRASE.
 *
 * @author ron
 */
@Component("dependencyPrimaryVerbFinder")
public class DependencyPrimaryVerbFinder implements
NLPPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(DependencyPrimaryVerbFinder.class);

/**
 * Create a new instance. There is no configuration.
 */
public DependencyPrimaryVerbFinder() {
}

/**
 * Set the primary verb in a phrase, clause or sentence on the
supplied
 * NLPText and return the updated NLPText. The primary verb is set on
all
 * the ancestors common to the primary verb and subject up to the
sentence
 * level. The primary verb is determined from the dependency parse as
the
 * governor of the nominal subject.
 *
 * @param text -
 *          the NLPText to find the primary verb of.
 * @see {@link NLPText#getPrimaryVerb()}
 * @see {@link GrammaticalRelation}
 * @see {@link DependencyPrinter}
 */
@Override
public NLPText process(NLPText text) {
    if (text.hasText()) {
        if
(GrammaticalStructureLevel.PARAGRAPH.equals(text.getGrammaticalStructu
reLevel())) {
            for (NLPText sentence : text.getChildren()) {
                process(sentence);
            }
        } else {
            try {
                GrammaticalRelation relation = findPrimarySubjectRelation(text);

```

```

NLPText verb = relation.getGovernor();
text.setPrimaryVerb(verb);
// TODO: The parser may have mis-marked the element?
if (!verb.in(ParseTag.VB, ParseTag.VBD, ParseTag.VBG,
ParseTag.VBN,
ParseTag.VBP, ParseTag.VBZ)) {
    log.debug("fixing mistagged text: " + verb + " " +
verb.getParseTag() + " "
+ verb.getPartOfSpeech());
    ((NLPTextImpl) verb).setParseTag(ParseTag.VB);
    ((NLPTextImpl) verb).setPartOfSpeech(PartOfSpeech.VERB);
    if ((verb.getParent() != null) && !
verb.getParent().is(ParseTag.VP)) {
        ((NLPTextImpl) verb.getParent()).setParseTag(ParseTag.VP);
    }
} catch (NLPPProcessorException e) {
    log.warn(e.toString());
}
}
return text;
}

protected GrammaticalRelation findPrimarySubjectRelation(NLPText
text) {
    for (GrammaticalRelation relation : text.getGrammaticalRelations())
{
        GrammaticalRelationType relType = relation.getType();
        if (relType.isA(GrammaticalRelationType.NOMINAL SUBJECT)
        || relType.isA(GrammaticalRelationType.AGENT)
        || relType.isA(GrammaticalRelationType.NOMINAL_PASSIVE SUBJECT))
{
            return relation;
        }
}
throw NLPPProcessorException.noPrimaryVerb(text);
}
}

```

dependencyprinter.java

```

/*
 * $Id: DependencyPrinter.java,v 1.5 2009/01/28 04:58:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.nlp.impl;

import java.util.HashSet;
import java.util.Set;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * An NLPTextWalkerFunction that takes an NLPText and prints out the
list of
 * dependency relations. For example:<br>
*
* <pre>
* </pre>
*
* @author ron
*/
public class DependencyPrinter implements
NLPTextWalkerFunction<StringBuilder> {

private final Set<GrammaticalRelation> relations = new
HashSet<GrammaticalRelation>();
private final int bufferSize;
private StringBuilder sb;

/**
 *
 */
public DependencyPrinter() {
    this(1000);
}

/**
 * @param bufferSize
 */
public DependencyPrinter(int bufferSize) {
    this.bufferSize = bufferSize;
}

@Override
public void init() {
    sb = new StringBuilder(bufferSize);
    relations.clear();
}

```

```

@Override
public void begin(NLPText text) {
    if
(GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLev
el())) {
    if (text.getDependentOf().size() > 0) {
        for (GrammaticalRelation rel : text.getDependentOf()) {
            if (!relations.contains(rel)) {
                relations.add(rel);
                sb.append("[dep] ");
                sb.append(rel.getDependent().getText());
                sb.append("-");
                sb.append(rel.getDependent().getWordIndex());
                sb.append(" is the ");
                sb.append(rel.getType().getLongName());
                sb.append(" of ");
                sb.append(rel.getGovernor().getText());
                sb.append("-");
                sb.append(rel.getGovernor().getWordIndex());
                sb.append("\n");
            }
        }
    }
}

@Override
public StringBuilder end(NLPText t) {
    return sb;
}
}

```

dependencysubjectfinder.java

```

/*
 * $Id: DependencySubjectFinder.java,v 1.8 2009/02/08 13:25:15 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.Set;

import org.apache.log4j.Logger;

```

```

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.SemanticRole;

/**
 * This uses the dependency information attached to the NLPText to get
the
 * subject of a SENTENCE, CLAUSE, or PHRASE.
 *
 * @author ron
 */
public class DependencySubjectFinder implements NLPProcessor<NLPText>
{
    private static final Logger log =
Logger.getLogger(DependencySubjectFinder.class);

    /**
     * Set the subject of a phrase, clause or sentence.
     *
     * @param text
     */
    @Override
    public NLPText process(NLPText text) {
        for (NLPText word : text.getLeaves()) {
            Set<GrammaticalRelation> relations = word
                .getDependentOfType(GrammaticalRelationType.NOMINAL SUBJECT);
            if (!relations.isEmpty()) {
                for (GrammaticalRelation relation : relations) {
                    log.info(relation);
                    if ((word.getParent() != null)
                        && word.getParent().is(GrammaticalStructureLevel.PHRASE)
                        && word.getParent().is(ParseTag.NP)) {
                        word = word.getParent();
                    }
                    word.setSemanticRole(relation.getGovernor(), SemanticRole.ACTOR);
                    break;
                }
            }
        }
        return text;
    }
}

```

```

    * @param text1
    * @param text2
    * @return the first ancestor in the syntax structure (phrases,
clauses,
    *         sentences) that is common to both NLPText nodes or null if
there
    *         isn't a common ancestor.
    */
private NLPText getMostCommonAncestor(NLPText text1, NLPText text2) {
    if ((text1 == null) || (text2 == null)) {
        return null;
    } else if (text1.equals(text2)) {
        return text1;
    }
    for (NLPText text1Ancestor : text1.getAncestors()) {
        for (NLPText text2Ancestor : text2.getAncestors()) {
            if (text1Ancestor.equals(text2Ancestor)) {
                return text1Ancestor;
            }
        }
    }
    return null;
}

```

dependencyverbrelationfinder.java

```

/*
 * $Id: DependencyVerbRelationFinder.java,v 1.2 2009/02/06 11:49:16
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * This uses the dependency information attached to the NLPText to get
the
 * phrase containing the word of the supplied relation type of the
primary verb
 * of a SENTENCE, CLAUSE, or PHRASE.
 */

```

```

* @author ron
*/
public class DependencyVerbRelationFinder extends
DependencyPrimaryVerbFinder {

    private final GrammaticalRelationType relationType;

    /**
     * @param relationType
     */
    public DependencyVerbRelationFinder(GrammaticalRelationType
relationType) {
        this.relationType = relationType;
    }

    /**
     * If the text is a SENTENCE, CLAUSE or PHRASE, return the direct
object.
     *
     * @param text -
     *          a SENTENCE or CLAUSE to find the relation phrase in
     * @return An NLPText
     */
    @Override
    public NLPText process(NLPText text) {
        NLPText verb = super.process(text);
        if (verb != null) {
            for (GrammaticalRelation relation :
verb.getGovernorOfType(relationType)) {
                return relation.getDependent();
            }
        }
        return null;
    }
}

```

describable.java

```

package edu.harvard.fas.rregan.requel;

import java.util.Comparator;

/**
 * @author ron
 */
public interface Describable {

```

```

    /**
     * Get the des
     *
     * @return The description of the describable object.
     */
    public String getDescription();

    /**
     * Compare the objects that contain goals by the description.
     */
    public static final Comparator<Describable> COMPARATOR = new
DescribableComparator();

    /**
     * A Comparator for collections of goal containers.
     */
    public static class DescribableComparator implements
Comparator<Describable> {
        @Override
        public int compare(Descrivable o1, Describable o2) {
            return o1.getDescription().compareTo(o2.getDescription());
        }
    }
}

```

dictionary.java

```

/*
 * $Id: Dictionary.java,v 1.1 2008/12/13 00:40:37 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import java.util.Collection;
import java.util.NavigableSet;
import java.util.TreeSet;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

/**

```

```

 * The Dictionary class is a transient wrapper for importing and
 exporting the
 * wordnet synsets, senses, words, and categories to and from the
 dictionary.xml
 * file.<br>
 * NOTE: Importing and Exporting this is very slow because there is so
much
 * data. Using sql is at least 100 times faster.
 *
 * @author ron
 */
@XmlRootElement(name = "dictionary")
public class Dictionary {

    private final NavigableSet<Word> words = new TreeSet<Word>();
    private final NavigableSet<Sense> senses = new TreeSet<Sense>();
    private final NavigableSet<Category> categories = new
TreeSet<Category>();
    private final NavigableSet<Synset> synsets = new TreeSet<Synset>();
    private final NavigableSet<SemlinkrefId> semlinkrefs = new
TreeSet<SemlinkrefId>();

    public Dictionary() {
    }

    @XmlAttribute(name = "firstWord")
    public String getFirstWord() {
        return words.first().getLemma();
    }

    @XmlAttribute(name = "lastWord")
    public String getLastWord() {
        return words.last().getLemma();
    }

    @XmlElementWrapper(name = "words")
    @XmlElementRef(name = "word")
    public NavigableSet<Word> getWords() {
        return words;
    }

    public void setWords(Collection<Word> words) {
        this.words.addAll(words);
    }

    public NavigableSet<Sense> getSenses() {
        return senses;
    }
}

```

```

    }

    public void setSenses(Collection<Sense> senses) {
        this.senses.addAll(senses);
    }

    @XmlElementWrapper(name = "categories")
    @XmlElementRef(name = "category")
    public NavigableSet<Category> getCategories() {
        return categories;
    }

    public void setCategories(Collection<Category> categories) {
        this.categories.addAll(categories);
    }

    @XmlElementWrapper(name = "synsets")
    @XmlElementRef(name = "synset")
    public NavigableSet<Synset> getSynsets() {
        return synsets;
    }

    public void setSynsets(Collection<Synset> synsets) {
        this.synsets.addAll(synsets);
    }

    public void setSemlinkrefs(Collection<SemlinkrefId> semlinkrefs) {
        this.semlinkrefs.addAll(semlinkrefs);
    }

    public NavigableSet<SemlinkrefId> getSemlinkrefs() {
        return semlinkrefs;
    }
}

```

dictionarycommandfactory.java

```

/*
 * $Id: DictionaryCommandFactory.java,v 1.3 2009/02/09 10:12:30 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.CommandFactory;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;

```

```

import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;
import
edu.harvard.fas.rregan.nlp.dictionary.VerBNetSelectionRestriction;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.command.LoadWordNetTaggedGl
ossesCommandImpl;

/**
 * @author ron
 */
public interface DictionaryCommandFactory extends CommandFactory {

    /**
     * @return a new ImportDictionaryCommand for loading the dictionary
     from an
     *         xml file.
     */
    public ImportDictionaryCommand newImportDictionaryCommand();

    /**
     * @return a new ExportDictionaryCommand for saving the dictionary to
     an xml
     *         file.
     */
    public ExportDictionaryCommand newExportDictionaryCommand();

    /**
     * @return a new EditDictionaryWordCommand for add/editing a word in
     the
     *         dictionary
     */
    public EditDictionaryWordCommand newEditDictionaryWordCommand();

    /**
     * @return a new EditSynsetCommand for add/editing a synset
     *         (meaning/concept) in the dictionary
     */
    public EditSynsetCommand newEditSynsetCommand();

    /**
     * @return a new EditSenseCommand for add/editing a sense in the
     dictionary
     */
    public EditSenseCommand newEditSenseCommand();
}

```

```

        * @return a new ImportSemcorCommand for importing the SemCor
        semantically
        *         annotated corpus into the database.
        */
    public ImportSemcorCommand newImportSemcorCommand();

    /**
     * @return a new ImportSemcorFileCommand for importing a single file
     of the
     *         SemCor semantically annotated corpus into the database.
     */
    public ImportSemcorFileCommand newImportSemcorFileCommand();

    /**
     * @return a new CalculateWordFrequencyCommand for calculating the
     word
     *         frequency of Semcor words tagged with a sense and setting
     the
     *         frequency on the Sense.
     */
    public CalculateWordFrequencyCommand
newCalculateWordFrequencyCommand();

    /**
     * @return a new SynsetHypernymWalkCommand for calculating the count
     of
     *         hyponym subsumers.
     */
    public SynsetHypernymWalkCommand newSynsetHypernymWalkCommand();

    /**
     * @return a new EditSemlinkRefCommand for editing a SemlinkRef.
     */
    public EditSemlinkRefCommand newEditSemlinkRefCommand();

    /**
     * @return a new WordNetTaggedGlossaryDigester for processing the
     WordNet
     *         synset merged tagged gloss word sense files.<br>
     *         @see {http://wordnet.princeton.edu/glosstag-files/}
     */
    public LoadWordNetTaggedGlossesCommandImpl
newWordNetTaggedGlossaryDigester();

    /**
     * @return a new EditSynsetDefinitionWordCommand for adding a
     *         {@link SynsetDefinitionWord} to a {@link Synset}
     */

```

```

/*
public EditSynsetDefinitionWordCommand
newEditSynsetDefinitionWordCommand();

/**
 * @return a new EditVerbNetSelectionRestriction for adding/editing a
 *         {@link VerbNetSelectionRestriction} of a {@link
VerbNetRoleRef}
 */
public EditVerbNetSelectionRestrictionCommand
newEditVerbNetSelectionRestrictionCommand();
}

```

dictionarycommandfactoryimpl.java

```

/*
 * $Id: DictionaryCommandFactoryImpl.java,v 1.3 2009/02/09 10:12:31
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.AbstractCommandFactory;
import edu.harvard.fas.rregan.command.CommandFactoryStrategy;
import
edu.harvard.fas.rregan.nlp.dictionary.command.CalculateWordFrequenceCo
mmand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditDictionaryWordComman
d;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand;
import edu.harvard.fas.rregan.nlp.dictionary.command.EditSenseCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetDefinitionWord
Command;

```

```

import
edu.harvard.fas.rregan.nlp.dictionary.command.EditVerbNetSelectionRest
rictionCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ExportDictionaryCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportDictionaryCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportSemcorCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportSemcorFileCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.SynsetHypernymWalkComman
d;

/**
 * An implementation of DictionaryCommandFactory
 *
 * @author ron
 */
@Controller("dictionaryCommandFactory")
@Scope("singleton")
public class DictionaryCommandFactoryImpl extends
AbstractCommandFactory implements
DictionaryCommandFactory {

    /**
     * @param creationStrategy -
     *         the strategy to use for creating new Command instances
     */
    @Autowired
    public DictionaryCommandFactoryImpl(CommandFactoryStrategy
creationStrategy) {
        super(creationStrategy);
    }

    @Override
    public ExportDictionaryCommand newExportDictionaryCommand() {
        return (ExportDictionaryCommand) getCreationStrategy().newInstance(
            ExportDictionaryCommandImpl.class);
    }

    @Override
    public ImportDictionaryCommand newImportDictionaryCommand() {
        return (ImportDictionaryCommand) getCreationStrategy().newInstance(
            ImportDictionaryCommandImpl.class);
    }
}

```

```

@Override
public EditDictionaryWordCommand newEditDictionaryWordCommand() {
    return (EditDictionaryWordCommand)
        getCreationStrategy().newInstance(
            EditDictionaryWordCommandImpl.class);
}

@Override
public EditSynsetCommand newEditSynsetCommand() {
    return (EditSynsetCommand)
        getCreationStrategy().newInstance(EditSynsetCommandImpl.class);
}

@Override
public EditSenseCommand newEditSenseCommand() {
    return (EditSenseCommand)
        getCreationStrategy().newInstance(EditSenseCommandImpl.class);
}

@Override
public ImportSemcorCommand newImportSemcorCommand() {
    return (ImportSemcorCommand) getCreationStrategy().newInstance(
        ImportSemcorCommandImpl.class);
}

@Override
public ImportSemcorFileCommand newImportSemcorFileCommand() {
    return (ImportSemcorFileCommand) getCreationStrategy().newInstance(
        ImportSemcorFileCommandImpl.class);
}

@Override
public CalculateWordFrequencyCommand
newCalculateWordFrequencyCommand() {
    return (CalculateWordFrequencyCommand)
        getCreationStrategy().newInstance(
            CalculateWordFrequencyCommandImpl.class);
}

@Override
public SynsetHypernymWalkCommand newSynsetHypernymWalkCommand() {
    return (SynsetHypernymWalkCommand)
        getCreationStrategy().newInstance(
            SynsetHypernymWalkCommandImpl.class);
}

```

```

@Override
public EditSemlinkRefCommand newEditSemlinkRefCommand() {
    return (EditSemlinkRefCommand) getCreationStrategy().newInstance(
        EditSemlinkRefCommandImpl.class);
}

@Override
public EditSynsetDefinitionWordCommand
newEditSynsetDefinitionWordCommand() {
    return (EditSynsetDefinitionWordCommand)
        getCreationStrategy().newInstance(
            EditSynsetDefinitionWordCommandImpl.class);
}

@Override
public LoadWordNetTaggedGlossesCommandImpl
newWordNetTaggedGlossaryDigester() {
    return (LoadWordNetTaggedGlossesCommandImpl)
        getCreationStrategy().newInstance(
            LoadWordNetTaggedGlossesCommandImpl.class);
}

@Override
public EditVerbNetSelectionRestrictionCommand
newEditVerbNetSelectionRestrictionCommand() {
    return (EditVerbNetSelectionRestrictionCommand)
        getCreationStrategy().newInstance(
            EditVerbNetSelectionRestrictionCommandImpl.class);
}

```

dictionaryinitializer.java

```

/*
 * $Id: DictionaryInitializer.java,v 1.3 2009/01/26 10:19:01 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import java.io.IOException;
import java.io.InputStream;
import java.util.zip.GZIPInputStream;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportDictionaryCommand;

/**
 * Load the dictionary from sql if no words exist.
 *
 * @author ron
 */
@Component("dictionaryInitializer")
@Scope("prototype")
public class DictionaryInitializer extends AbstractSystemInitializer {

    /**
     * The name of the property in the DictionaryInitializer.properties
     * file
     * that contains the path to the dictionary xml data file, if not
     * supplied
     * the default file is "resources/nlp/dictionary/dictionary.xml.gz"
     */
    public static final String PROP_DICTIONARY_XML_FILE =
"DictionaryXMLFile";
    public static final String PROP_DICTIONARY_XML_FILE_DEFAULT =
"resources/nlp/dictionary/dictionary.xml.gz";

    private final DictionaryRepository dictionaryRepository;
    private final ImportDictionaryCommand command;
    private final CommandHandler commandHandler;

    /**
     * @param dictionaryRepository
     * @param commandHandler
     * @param command
     */
    @Autowired
    public DictionaryInitializer(DictionaryRepository
dictionaryRepository,
        CommandHandler commandHandler, ImportDictionaryCommand command) {
        super(10);
        this.dictionaryRepository = dictionaryRepository;
        this.commandHandler = commandHandler;
        this.command = command;
    }

    }

    @Override
    public void initialize() {
        try {
            // TODO: this disables the initializer and shouldn't be hard coded
            if (true) {
                return;
            }

            // if no categories are defined assume the dictionary is
            // empty and load it from the xml file.
            if (dictionaryRepository.findCategories().isEmpty()) {
                ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                    DictionaryInitializer.class.getName());
                String dictionaryPath =
resourceBundleHelper.getString(PROP_DICTIONARY_XML_FILE,
                    PROP_DICTIONARY_XML_FILE_DEFAULT);
                log.info("loading dictionary: "
                    +
getClass().getClassLoader().getResource(dictionaryPath).toExternalForm
());
                command.setInputStream(getDataFileInputStream(dictionaryPath));
                commandHandler.execute(command);
            }
            } catch (Exception e) {
                log.error("failed to initialize dictionary from xml: " + e, e);
            }
        }

        private InputStream getDataFileInputStream(String dataFilePath)
throws IOException {
        log.debug("loading data file " + dataFilePath);
        InputStream dataInputStream =
getClass().getClassLoader().getResourceAsStream(dataFilePath);
        if (dataFilePath.endsWith(".gz")) {
            dataInputStream = new GZIPInputStream(dataInputStream);
        }
        return dataInputStream;
    }
}

```

dictionaryphoneticcodeinitializer.java

```
/*
 * $Id: DictionaryPhoneticCodeInitializer.java,v 1.3 2009/01/26
10:19:01 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Word;

/**
 * Load the dictionary from SQL if no words exist.
 *
 * @author ron
 */
@Component("dictionaryPhoneticCodeInitializer")
@Scope("prototype")
public class DictionaryPhoneticCodeInitializer extends
AbstractSystemInitializer {

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     * @param jdbcTemplate
     */
    @Autowired
    public DictionaryPhoneticCodeInitializer(DictionaryRepository
dictionaryRepository,
        JdbcTemplate jdbcTemplate) {
        super(2);
        this.dictionaryRepository = dictionaryRepository;
    }

    @Override
    @Transactional(propagation = Propagation.REQUIRED)
```

```
public void initialize() {
    // TODO: this disables the initializer and shouldn't be hard coded
    if (true) {
        return;
    }

    String phoneticCode =
dictionaryRepository.findWord("a").getPhoneticCode();
    if ((phoneticCode == null) || (phoneticCode.length() == 0)) {
        try {
            log.info("initializing phonetic codes for word net words.");
            for (Word word : dictionaryRepository.findWords()) {
                word
                    .setPhoneticCode(dictionaryRepository.generatePhoneticCode(word
                        .getLemma()));
            }
        } catch (Exception e) {
            log.error("failed to initialize phonetic codes: " + e, e);
        }
    }
}
```

dictionaryrepository.java

```
/*
 * $Id: DictionaryRepository.java,v 1.6 2009/03/27 07:16:08 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import java.util.Collection;
import java.util.List;
import java.util.Set;

import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordExcept
ion;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init.WordNetDefi
nitionWordsInitializer;
```

```

import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init.WordNetHypo
nymCountInitializer;
import edu.harvard.fas.rregan.repository.Repository;

/**
 * @author ron
 */
public interface DictionaryRepository extends Repository {

    /**
     * @return The full dictionary of wordnet synsets, categories, and
words.
     */
    public Dictionary getDictionary();

    /**
     * @param firstWordStartsWith
     * @param lastWordStartsWith
     * @return The words in the dictionary starting with the first word
that
     *         matches firstWordStartsWith and ending with the last word
that
     *         matches lastWordStartsWith
     */
    public Dictionary getDictionary(String firstWordStartsWith, String
lastWordStartsWith);

    /**
     * The category name is looked up by prepending the partOfSpeech to
the
     * category name to get a category name of the form pos.catname, such
as
     * noun.location.
     *
     * @param partOfSpeech
     * @param name -
     *         the simple category name without the leading part of
speech,
     *         for example 'location' and not 'noun.location'
     * @return the category by part of speech and name
     * @throws RuntimeException
     */
    public Category findCategory(PartOfSpeech partOfSpeech, String name);

    /**
     * @param id
     * @return
     */
    public Word findWord(Long id);

    /**
     * @param lemma -
     *         base form of the word.
     * @return a word with associated senses/synsets for the given lemma
     * @throws NoSuchWordException
     */
    public Word findWord(String lemma) throws NoSuchWordException;

    /**
     * @param lemma
     * @return
     * @throws NoSuchWordException
     */
    public Word findWordExact(String lemma) throws NoSuchWordException;

    /**
     * @param lemma -
     *         base form of the word.
     * @param pos -
     *         part of speech
     * @return a word with associated senses/synsets for the given lemma
and
     *         part of speech
     * @throws NoSuchWordException
     */
    public Word findWord(String lemma, PartOfSpeech pos) throws
NoSuchWordException;

    /**
     * @param lemma -
     *         base form of the word.
     * @param pos -
     *         part of speech
     * @return a word with associated senses/synsets for the given lemma
and
     *         part of speech
     * @throws NoSuchWordException
     */
    public Word findWordExact(String lemma, PartOfSpeech pos) throws
NoSuchWordException;

    /**
     * Generate a phonetic code using the Jazzy

```

```

* com.swabunga.spell.engine.DoubleMeta Transformer.
*
* @param word
* @return
*/
public String generatePhoneticCode(String word);

/**
* @param phoneticCode -
*      a code generated by the
* @return a word with associated senses/synsets for the given lemma
* @throws NoSuchWordException
*/
public List<Word> findWordsByPhoneticCode(String phoneticCode) throws
NoSuchWordException;

/**
* @param word
* @return true if the supplied word is in the dictionary
*/
public Boolean isKnownWord(String word);

/**
* @param word
* @param threshold
* @return a list of words as Strings that are similar to the
supplied word
*      with the given threshold.
*/
public List<String> findSpellingSuggestions(String word, int
threshold);

/**
* @param word
* @param maxSuggestions
* @return a list of word Senses that are a more specific (hyponyms)
of the
*      supplied word with the highest information content.
*/
public Set<Sense> findMoreSpecificWords(Sense word, int
maxSuggestions);

/**
* Calculate the information content of a synset using the measure of
Seco,
* Veale and Hayes, which uses the number of sub-terms (hyponyms) to
* determine how specific a term is. All synsets with no hyponyms are
* considered most specific and given a IC rank of 1.0. An artificial
root
* node for all synsets of the same part of speech is assumed to
better
* scale the information content of the lowest common subsumers,
especially
* in the verb hierarchies which tend to be shallow and disjoint.
*
* @param synset
* @param linkType
* @return
*/
public double infoContent(Synset synset, Linkdef linkType);

/**
* TODO: the repository should be for getting only, this should be
part of a
* command.
*
* @param word
*/
public void addToDictionary(String word);

/**
* @return all the words in the dictionary
* @throws RuntimeException
*/
public Collection<Word> findWords();

/**
* @return all the Synset categories in the dictionary
* @throws RuntimeException
*/
public Collection<Category> findCategories();

/**
* @return all the Synsets in the dictionary.
* @throws RuntimeException
*/
public Collection<Synset> findSynsets();

/**
* @return all the Senses in the dictionary.
*/
public Collection<Sense> findSenses();

*/

```

```

* @param lemma
* @param pos
* @param senseIndex
* @return A synset
*/
public Sense findSense(String lemma, PartOfSpeech pos, int
senseIndex);

/**
 * a WordNet sense key is a composite made up of:<br>
 * lemma % ss_type:lex_filenum:lex_id:head_word:head_id The ss_type
(synset
 * type) is a digit:<br>
 * 1 - NOUN<br>
 * 2 - VERB<br>
 * 3 - ADJECTIVE<br>
 * 4 - ADVERB<br>
 * 5 - ADJECTIVE SATELLITE<br>
 * The lex_filenum is a two digit number identifying the
lexicographer file
 * containing the synset.<br>
 * The lex_id is a two digit number that uniquely identifies a
sense.<br>
 * The head_word is the lemma of the first word of an adjective
satellite
 * synset.<br>
 * The head_id is the lex_id of the head_word.<br>
 *
* @param senseKey -
*          the sense key (is this really for a synset?)
* @return a sense by the wornet sense key and word
*/
public Sense findSensesByWordnetSenseKey(String senseKey);

/**
 * @param word
 * @param synset
 * @return
*/
public Sense findSensesByWordAndSynset(Word word, Synset synset);

/**
 * @param lemma
 * @param synsetId
 * @return
*/

```

```

public Sense findSensesByLemmaAndSynsetId(String lemma, Long
synsetId);

/**
 * @param id
 * @return Get a specific synset by id
*/
public Synset findSynset(Long id);

/**
 * @param id
 * @return Get a specific semantic link definition by id
*/
public Linkdef findLinkDef(Long id);

/**
 * @param id
 * @return Get a semantic link by id.
*/
public Semlinkref findSemlinkref(SemlinkrefId id);

/**
 * @param fromSynset
 * @param toSynset
 * @param linkDef
 * @param distance
 * @return Get a Semantic path between two synsets by type and
distance.
*/
public Semlinkref findSemlinkref(Synset fromSynset, Synset toSynset,
Linkdef linkDef,
Integer distance);

/**
 * @param id
 * @return Get a lexo-semantic link by id.
*/
public Lexlinkref findLexlinkref(LexlinkrefId id);

/**
 * @param fromSense
 * @param toSense
 * @param linkDef
 * @return Get a lexo-semantic link between two words
*/
public Lexlinkref findLexlinkref(Sense fromSense, Sense toSense,
Linkdef linkDef);

```

```

/**
 * @param fromSense
 * @param toSense
 * @return a collection of all the lexo-semantic relations between
two
 *         senses.
 */
public Collection<Lexlinkref> findLexlinkref(Sense fromSense, Sense
toSense);

/**
 * @return true if the synset wsdata needs to be expanded during
 *         initialization
 * @see WordNetDefinitionWordsInitializer
 */
public boolean buildSynsetDefinitionWords();

/**
 * @return true if the synset wsdata needs to be expanded during
 *         initialization
 * @see WordNetHyponymCountInitializer
 */
public boolean buildSynsetLinkPathsAndCounts();

/**
 * @return true if the wordnet senses don't have SenseKey's defined.
 */
public boolean buildSenseKeys();

/**
 * This is used by the WordNetHyponymCountInitializer to get all the
 * semantic links from the semlinkref table. Note these are returned
as
 * 4-tuples of the properties and not Semlinkref objects for faster
 * processing.
 *
 * @return a List of (from synset, to synset, link type, distance)
tuples
 */
public List<Object[]> findSemanticLinks();

/**
 * @return All the semantic link types
 */
public Collection<Linkdef> findLinkdefs();

```

```

/**
 * @param synset1
 * @param synset2
 * @return The set of synsets that are hypernyms or the supplied
synsets
 *         having the shortest path between the two synsets. This
will
 *         typically return a set of one synset, although synsets may
have
 *         multiple ancestors and it is possible for multiple
ancestors to
 *         have the same path distance between the synsets, in which
case
 *         all those synsets will be returned. If the two synsets do
not
 *         have a common ancestor (for example a noun and a verb),
then an
 *         empty set is returned.
 */
public Set<Synset> getLowestCommonHypernyms(Synset synset1, Synset
synset2);

/**
 * Wordnet is made up of multiple disjoint hierarchies of synsets.
This
 * returns the root node of those hierarchies.
 *
 * @param synset
 * @return The synset that is a hypernym of the supplied synset and
has no
 *         hypernyms.
 */
public Synset getRootHypernym(Synset synset);

/**
 * @param pos
 * @return The number of concepts (synsets) of the given part of
speech.
 */
public Integer getConceptCount(String pos);

/**
 * @param word1
 * @param word2
 * @return a list of the synsets that contain both supplied words of
any
 *         sense in their definition.

```

```

*/
public List<Synset> findSynsetsWithColocatedDefinitionWords(Word
word1, Word word2);

/**
 * @param sense
 * @param word
 * @return a list of the synsets that contain both the supplied word
sense
 *         and un-sensed word.
 */
public List<Synset>
findSynsetsWithColocatedDefinitionSenseAndWord(Sense sense, Word
word);

/**
 * @return all the Semcor files in the dictionary.
 */
public Collection<SemcorFile> findSemcorFiles();

/**
 * @param sectionName
 * @param fileName
 * @return all the Semcor files in the dictionary.
 */
public SemcorFile findSemcorFile(String sectionName, String
fileName);

/**
 * @return all the Semcor sentences in the dictionary.
 */
public Collection<SemcorSentence> findSemcorSentences();

/**
 * @param sectionName
 * @param fileName
 * @return all the Semcor sentences in the given section and file.
 */
public Collection<SemcorSentence> findSemcorSentences(String
sectionName, String fileName);

/**
 * Find semcor sentences that contain the supplied word
 *
 * @param word -
 *           a dictionary word of any sense
 * @return

```

```

*/
public Collection<SemcorSentence> findSemcorSentences(Word word);

/**
 * Find semcor sentences that contain the supplied word sense
 *
 * @param sense -
 *           a specific sense of the word
 * @return
 */
public Collection<SemcorSentence> findSemcorSentences(Sense sense);

/**
 * @param word1
 * @param word2
 * @return a list of the semcor sentences that contain both supplied
words
 *         of any sense in their definition.
 */
public List<SemcorSentenceWord>
findSemcorSentencesWithColocatedDefinitionWords(Word word1,
Word word2);

/**
 * @param sense
 * @param word
 * @return a list of the semcor sentences that contain both the
supplied
 *         word sense and un-sensed word.
 */
public List<SemcorSentenceWord>
findSemcorSentencesWithColocatedDefinitionSenseAndWord(
Sense sense, Word word);

/**
 * @param name -
 *           the name of the selectional restriction based on
verbnet-2.1
 *           vn_schema-3.xsd selrestrType type.
 * @return a selectional restriction for the given name
 */
public VerbNetSelectionRestrictionType
findVerbNetSelectionRestrictionType(String name);

/**
 * This is a convenience method for comparing the synsets of two
senses.

```

```

/*
 * @param hypernym -
 *           a sense represening the hypernym
 * @param sense
 * @return true if the supplied sense is a hyponym of the supplied
hypernym
 *           sense.
 * @see {@link #isHyponym(Synset, Synset)}
 */
public boolean isHyponym(Sense hypernym, Sense sense);

/**
 * @param hypernym -
 *           a synset that may be the hypernym of the supplied
synset
 * @param synset -
 *           the synset to test
 * @return true if the supplied synset is a hyponym of the supplied
hypernym
 *           synset.
 */
public boolean isHyponym(Synset hypernym, Synset synset);

/**
 * @param hypernym -
 *           The synset that is the hypernym of the returned
hyponyms
 * @return The hypernyms of the supplied hypernym where the distince
between
 * @param maxDistance -
 *           The maximum distance from the specified hypernym to the
 *           hyponyms. Where the direct hyponyms are at a distance
of 1 and
 *           the hyponyms of of the direct hyponyms are at a
distance of 2
 *           etc. the synsets is less than or equal to the maxDepth.
 */
public Collection<Synset> findHyponyms(Synset hypernym, int
maxDistance);
}


```

dictionarysqlinitializer.java

```

/*
 * $Id: DictionarySQLInitializer.java,v 1.3 2009/01/26 20:29:47 rregan
Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.zip.GZIPInputStream;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * Load the dictionary from SQL if no words exist.
 *
 * @author ron
 */
@Component("dictionarySQLInitializer")
@Scope("prototype")
public class DictionarySQLInitializer extends
AbstractSystemInitializer {

/**
 * The name of the property in the
DictionarySQLInitializer.properties file
 * that contains the path to the directory containing all the sql
files, if
 * not supplied the default file is "resources/nlp/dictionary/"
 */
public static final String PROP_DICTIONARY_SQL_FILES_DIRECTORY =
"DictionarySQLFilesDirectory";
public static final String
PROP_DICTIONARY_SQL_FILES_DIRECTORY_DEFAULT =
"resources/nlp/dictionary/";


```

```

/**
 * The name of the property in the
DictionarySQLInitializer.properties file
 * that contains a comma delimited list of sql file names in the
order they
 * should be loaded. The files are expected to be in the directory
defined
 * by PROP_DICTIONARY_SQL_FILES_DIRECTORY.<br>
 * If not supplied the default list is "categorydef.sql.gz,
word.sql.gz,
 * morphdef.sql.gz, morphref.sql.gz, synset.sql.gz, sense.sql.gz,
 * synset_definition_word.sql.gz, synset_subsumer_counts.sql.gz,
 * linkdef.sql.gz, lexlinkref.sql.gz, semlinkref.sql.gz,
vnclass.sql.gz,
 * vnframedef.sql.gz, vnframeref.sql.gz, semcor_file.sql.gz,
 * semcor_sentence.sql.gz, semcor_sentence_word.sql.gz"<br>
 * NOTE: the files may be plain sql files or zipped sql files.
 */
public static final String PROP_DICTIONARY_SQL_FILES =
"DictionarySQLFiles";
public static final String PROP_DICTIONARY_SQL_FILES_DEFAULT =
"categorydef.sql.gz, word.sql.gz, morphdef.sql.gz, morphref.sql.gz,
synset.sql.gz, sense.sql.gz, synset_definition_word.sql.gz,
synset_subsumer_counts.sql.gz, linkdef.sql.gz, lexlinkref.sql.gz,
semlinkref.sql.gz, vnclass.sql.gz, vnframedef.sql.gz,
vnframeref.sql.gz, semcor_file.sql.gz, semcor_sentence.sql.gz,
semcor_sentence_word.sql.gz";

private final DictionaryRepository dictionaryRepository;
private final JdbcTemplate jdbcTemplate;

/**
 * @param dictionaryRepository
 * @param jdbcTemplate
 */
@.Autowired
public DictionarySQLInitializer(DictionaryRepository
dictionaryRepository,
    JdbcTemplate jdbcTemplate) {
super(1);
this.dictionaryRepository = dictionaryRepository;
this.jdbcTemplate = jdbcTemplate;
}

@Override
@Transactional(propagation = Propagation.REQUIRED)

```

```

public void initialize() {
    if (dictionaryRepository.findCategories().isEmpty()) {
        Connection conn = null;
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                DictionarySQLInitializer.class.getName());
            String dictionaryDirPath = resourceBundleHelper.getString(
                PROP_DICTIONARY_SQL_FILES_DIRECTORY,
                PROP_DICTIONARY_SQL_FILES_DIRECTORY_DEFAULT);
            conn = jdbcTemplate.getDataSource().getConnection();
            conn.setAutoCommit(false);
            Statement statement = conn.createStatement();

            String dictionaryFiles =
resourceBundleHelper.getString(PROP_DICTIONARY_SQL_FILES,
                PROP_DICTIONARY_SQL_FILES_DEFAULT);
            for (String sqlFile : dictionaryFiles.split(",")) {
                loadSQLFile(dictionaryDirPath + sqlFile.trim(), statement);
            }
            conn.commit();
        } catch (Exception e) {
            if (conn != null) {
                try {
                    conn.rollback();
                } catch (SQLException se) {
                    log.error("could not rollback: " + se, se);
                }
            }
            log.error("could not load dictionary via SQL: " + e, e);
        }
    }
}

private void loadSQLFile(String path, Statement statement) throws
IOException, SQLException {
    log.info("loading sql file: " + path);
    InputStream inputStream =
getClass().getClassLoader().getResourceAsStream(path);
    if (path.endsWith(".gz")) {
        inputStream = new GZIPInputStream(inputStream);
    }

    BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
    StringBuilder sqlBuffer = new StringBuilder(500);

```

```

while (true) {
    String sql = readSQL(reader, sqlBuffer);
    if (sql.length() == 0) {
        break;
    }
    // log.debug("sql = " + sql);
    statement.executeUpdate(sql);
    sqlBuffer.setLength(0);
}

private static final int STATE_NORMAL = 0;
private static final int STATE_START_COMMENT = 1;
private static final int STATE_IN_COMMENT = 2;
private static final int STATE_END_COMMENT = 3;
private static final int STATE_IN_STRING = 4;
private static final int STATE_ESCAPE = 5;

protected String readSQL(Reader reader, StringBuilder queryBuffer)
throws IOException {
    int state = 0;
    int ch;
    while ((ch = reader.read()) != -1) {
        if ((STATE_NORMAL == state) && (ch == '/')) {
            state = STATE_START_COMMENT;
        } else if ((STATE_ESCAPE == state)) {
            state = STATE_IN_STRING;
            queryBuffer.append((char) ch);
        } else if ((STATE_IN_STRING == state) && (ch == '\\')) {
            state = STATE_ESCAPE;
            queryBuffer.append('\\');
        } else if ((STATE_NORMAL == state) && (ch == '\\')) {
            state = STATE_IN_STRING;
            queryBuffer.append('\\');
        } else if ((STATE_IN_STRING == state) && (ch == '\\')) {
            state = STATE_NORMAL;
            queryBuffer.append('\\');
        } else if (STATE_START_COMMENT == state) {
            if (ch == '/') {
                throw new IOException("// unsupported");
            } else if (ch == '*') {
                state = STATE_IN_COMMENT;
            } else {
                state = STATE_NORMAL;
                queryBuffer.append('/');
                queryBuffer.append((char) ch);
            }
        }
    }
}

```

```

    } else if ((STATE_END_COMMENT == state) && (ch == '/')) {
        state = STATE_NORMAL;
    } else if ((STATE_IN_COMMENT == state) && (ch == '*')) {
        state = STATE_END_COMMENT;
    } else if ((STATE_NORMAL == state) || (STATE_IN_STRING == state)) {
        queryBuffer.append((char) ch);
        if ((STATE_IN_STRING != state) && (ch == ';')) {
            break;
        }
    }
}
return queryBuffer.toString().trim();
}

```

dictionarysuffixexchanginglemmatizerrule.java

```

/*
 * $Id: DictionarySuffixExchangingLemmatizerRule.java,v 1.3 2008/12/15
 * 06:35:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.lemmatizer;

import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordException;

/**
 * Convert a word to its lemma by systematically applying rules that
 * replace a
 * suffix with a new suffix and checking if the manufactured string is
 * in the
 * dictionary.<br>
 * For Example:<br>
 * if a noun ends in 's' replace the 's' with '' and see if the word
 * exists.<br>
 */

```

```

 * if a noun ends in 'ses' replace the 'ses' with 's' and see if the
word
 * exists.<br>
 * <br>
 * The rules are applied by trying to replace the longest suffixes
first.
 *
 * @author ron
 */
public class DictionarySuffixExchangingLemmatizerRule extends
AbstractDictionaryLemmatizerRule {

    private final List<String> suffixExchangesFrom;
    private final List<String> suffixExchangesTo;
    private final Set<PartOfSpeech> appliesToPartsOfSpeech;

    /**
     * @param dictionaryRepository -
     *          the dictionary repository to look up a possible lemma
after
     *          applying the rule to see if it matches a known word.
     * @param appliesToPartsOfSpeech -
     *          a list of the parts of speech this rule applies to.
     * @param suffixExchangesFrom -
     *          a list of the suffix to remove from the word and
replaced by
     *          the corresponding string in suffixExchangesTo
     * @param suffixExchangesTo -
     *          a list of the suffix to append to the word to create
the lemma
     *          after removing the suffix corresponding to this string
in
     *          suffixExchangesFrom
     */
    public DictionarySuffixExchangingLemmatizerRule(DictionaryRepository
dictionaryRepository,
        Set<PartOfSpeech> appliesToPartsOfSpeech, List<String>
suffixExchangesFrom,
        List<String> suffixExchangesTo) {
        super(dictionaryRepository);
        this.appliesToPartsOfSpeech = appliesToPartsOfSpeech;
        this.suffixExchangesFrom = suffixExchangesFrom;
        this.suffixExchangesTo = suffixExchangesTo;
        sortSuffixesByLength();
    }

    /**

```

```

     * sort the suffixes by length, the longest first. suffixes of the
same
     * length are in arbitrary order.
     */
    private void sortSuffixesByLength() {
        int maxSuffixLength = 0;
        Map<Integer, Set<Integer>> suffixesByLength = new HashMap<Integer,
Set<Integer>>();
        for (int index = 0; index < suffixExchangesFrom.size(); index++) {
            String suffix = suffixExchangesFrom.get(index);
            if (!suffixesByLength.containsKey(suffix.length())) {
                if (maxSuffixLength < suffix.length()) {
                    maxSuffixLength = suffix.length();
                }
                suffixesByLength.put(suffix.length(), new HashSet<Integer>());
            }
            suffixesByLength.get(suffix.length()).add(index);
        }
        int swapIndex = 0;
        for (int length = maxSuffixLength; length > 0; length--) {
            Set<Integer> suffixesOfLength = suffixesByLength.get(length);
            if (suffixesOfLength != null) {
                for (int index : suffixesOfLength) {
                    String tmpFrom = suffixExchangesFrom.get(swapIndex);
                    String tmpTo = suffixExchangesTo.get(swapIndex);
                    suffixExchangesFrom.set(swapIndex,
suffixExchangesFrom.get(index));
                    suffixExchangesTo.set(swapIndex, suffixExchangesTo.get(index));
                    suffixExchangesFrom.set(index, tmpFrom);
                    suffixExchangesTo.set(index, tmpTo);
                    swapIndex++;
                }
            }
        }
    }

    @Override
    public String lemmatize(String word, PartOfSpeech partOfSpeech) {
        if (appliesToPartsOfSpeech.contains(partOfSpeech)) {
            for (int index = 0; index < suffixExchangesFrom.size(); index++) {
                String suffix = suffixExchangesFrom.get(index);
                if (word.endsWith(suffix)) {
                    String lemma = word.substring(0, word.length() - suffix.length())
+ suffixExchangesTo.get(index);
                    try {
                        return getDictionaryRepository().findWord(lemma,
partOfSpeech).getLemma();
                    }

```

```

        } catch (NoSuchWordException e) {
            // keep trying
        }
    }
}
return null;
}
}

```

dictionizer.java

```

/*
 * $Id: Dictionizer.java,v 1.7 2009/03/27 07:16:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordExcept
ion;

/**
 * Assign the WordNet "dictionary" word to each NLPText word.
 *
 * @author ron
 */
@Component("dictionizer")
public class Dictionizer implements NLPPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(Dictionizer.class);

    private final DictionaryRepository dictionaryRepository;

```

```

    /**
     * @param dictionaryRepository
     * @param processorFactory
     */
    @Autowired
    public Dictionizer(DictionaryRepository dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }

    @Override
    public NLPText process(NLPText text) {
        if (text.is(GrammaticalStructureLevel.WORD)) {
            if (!text.in(PartOfSpeech.PUNCTUATION, PartOfSpeech.SYMBOL)) {
                try {
                    Word dictionaryWord = null;
                    Sense dictionaryWordSense = null;
                    if (text.in(ParseTag.CD)) {
                        // TODO: look at the siblings and possibly combine
                        // elements into more advanced numbers, for example:
                        // (NP (CD 1) (. .) (CD 25)) -> (NP (NN 1.25))
                        dictionaryWord = dictionaryRepository
                            .findWord("integer", PartOfSpeech.NOUN);
                        dictionaryWordSense = dictionaryWord.getSense(PartOfSpeech.NOUN,
1);
                        // TODO: change the part of speech to NUMBER
                    }
                    if (text.in(ParseTag.NNP, ParseTag.NNPS)) {
                        // TODO: use named entity recognizer to get the most
                        // specific regular noun replacement
                        dictionaryWord = dictionaryRepository.findWord("entity", text
                            .getPartOfSpeech());
                        dictionaryWordSense = dictionaryWord.getSense(PartOfSpeech.NOUN,
1);
                    } else {
                        dictionaryWord = dictionaryRepository.findWord(text.getLemma(),
text
                            .getPartOfSpeech());
                    }
                    text.setDictionaryWord(dictionaryWord);
                    text.setDictionaryWordSense(dictionaryWordSense);
                } catch (NoSuchWordException e) {
                    // not a dictionary word
                }
            }
        } else {

```

```

        for (NLPText word : text.getLeaves()) {
            process(word);
        }
        return text;
    }
}

```

digesterruleloggingdecorator.java

```

/*
 * $Id: DigesterRuleLoggingDecorator.java,v 1.1 2008/12/13 00:39:57
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.apache.commons.digester.Digester;
import org.apache.commons.digester.Rule;
import org.apache.log4j.Logger;
import org.xml.sax.Attributes;

/**
 * Decorates a WordNetTaggedGlossaryDigesterRule to log begin(),
body() and
 * end() method calls before (after for end) calling the same method
on the
 * decorated rule
 *
 * @author ron
 */
public class DigesterRuleLoggingDecorator extends Rule {
    private static final Logger log =
Logger.getLogger(DigesterRuleLoggingDecorator.class);

    private Rule rule = null;

    /**
     * Create an instance that doesn't decorate another digester rule and
justs
     * logs begin(), body(), and end() events.
     */
    public DigesterRuleLoggingDecorator() {
        super();
    }
}

```

```

    /**
     * Create an instance that decorates the supplied digester rule and
logs
     * begin(), body(), and end() events and calls the same method on the
     * decorated rule.
     *
     * @param rule -
     *          the rule being decorated.
     */
    public DigesterRuleLoggingDecorator(Rule rule) {
        super();
        setRule(rule);
    }

    @Override
    public void begin(String namespace, String name, Attributes
attributes) throws Exception {

        if (log.isDebugEnabled()) {
            log.debug("starting element: name = \'" + name + "\' namespace
= \'" + namespace
+ "\' attributes = " + attributesToString(attributes));
        }
        if (getRule() != null) {
            getRule().begin(namespace, name, attributes);
        }
    }

    @Override
    public void body(String namespace, String name, String body) throws
Exception {
        if (log.isDebugEnabled()) {
            log.debug("body of element: name = \'" + name + "\' namespace = \'" +
namespace
+ "\' body = " + body);
        }
        if (getRule() != null) {
            getRule().body(namespace, name, body);
        }
    }

    @Override
    public void end(String namespace, String name) throws Exception {
        if (getRule() != null) {
            getRule().end(namespace, name);
        }
        if (log.isDebugEnabled()) {

```

```

    log.debug("ending element: name = \\" + name + "\\" namespace = \\"+
+ namespace + "\\"");
}
}

@Override
public Digester getDigester() {
    return getRule().getDigester();
}

@Override
public void setDigester(Digester digester) {
    getRule().setDigester(digester);
}

private Rule getRule() {
    return rule;
}

private void setRule(Rule rule) {
    this.rule = rule;
}

private String attributesToString(Attributes attributes) {
    StringBuffer strbuf = new StringBuffer(255);
    strbuf.append("[");
    if ((attributes != null) && (attributes.getLength() > 0)) {
        strbuf.append(attributes.getQName(0));
        strbuf.append(" = \\"");
        strbuf.append(attributes.getValue(0));
        strbuf.append("\\"");
        for (int i = 1; i < attributes.getLength(); i++) {
            strbuf.append(", ");
            strbuf.append(attributes.getQName(i));
            strbuf.append(" = \\"");
            strbuf.append(attributes.getValue(i));
            strbuf.append("\\"");
        }
        strbuf.append("]");
    }
    return strbuf.toString();
}
}

```

documentationinitializationlistener.java

```

/*
 * $Id: DocumentationInitializationListener.java,v 1.2 2008/10/20
02:07:51 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requiel;

import java.io.File;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.utils.Untar;

/**
 * This listener detects if the documentation archives exist and if
they have
 * been extracted or not. If not they are extracted.
 *
 * @author ron
 */
public class DocumentationInitializationListener implements
ServletContextListener {
    private static final Logger log =
Logger.getLogger(DatabaseCreationListener.class);

    /**
     *
     */
    public DocumentationInitializationListener() {
        super();
    }

    public void contextDestroyed(ServletContextEvent event) {
    }

    public void contextInitialized(ServletContextEvent event) {
        File docDirectory = null;
        File docArchiveFile = null;
        File docHtmlDirectory = null;
        try {

```

domainadminuserrole.java

```
/*
 * $Id: DomainAdminUserRole.java,v 1.11 2009/03/29 02:08:32 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.user.AbstractUserRole;

/**
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.DomainAdminUserRole")
```

```
@XmlRootElement(name = "domainAdminUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "domainAdminUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class DomainAdminUserRole extends AbstractUserRole {
    static final long serialVersionUID = 0L;

    static {
        // TODO: domain administration is disabled, to add it back in
        uncomment
        // the following
        /*
         * AbstractUserRole.userRoleTypes.add(DomainAdminUserRole.class);
         */
        AbstractUserRole.userRoleTypePermissions.put(DomainAdminUserRole.class
        ,
            new HashSet<UserRolePermission>());
        /*
     }
}

/**
 *
 */
public DomainAdminUserRole() {
    super();
}
}
```

domainobjectwrapper.java

```
/*
 * $Id: DomainObjectWrapper.java,v 1.1 2008/12/13 00:41:16 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository.jpa;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.SortedSet;
```

```

import java.util.TreeSet;
import javax.persistence.Entity;
import net.sf.cglib.proxy.Enhancer;
import net.sf.cglib.proxy.Factory;
import org.apache.log4j.Logger;
import org.hibernate.proxy.HibernateProxy;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.requel.user.UserSet;
import edu.harvard.fas.rregan.requel.user.impl.UserSetImpl;

/**
 * A component that takes an object and if it is a persistent entity
 * or
 * collection of persistent entities, it wraps all the entities in a
 * proxy using
 * the EntityProxyInterceptor to dispatch method calls.
 *
 * @author ron
 */
@Component("domainObjectWrapper")
@Scope("singleton")
public class DomainObjectWrapper {
    protected static final Logger log =
Logger.getLogger(DomainObjectWrapper.class);

    private final Map<Class<?>, Factory> factoryMap = new HashMap<Class<?>, Factory>();
    private final PersistenceContextHelper persistenceContextHelper;

    /**
     * @param persistenceContextHelper
     */
    @Autowired
    public DomainObjectWrapper(PersistenceContextHelper
persistenceContextHelper) {
        this.persistenceContextHelper = persistenceContextHelper;
    }

    /**
     * Given an object, if it is a persistent entity or a Collection of
     * persistent entities, wrap it/them in a proxy.
    */
}

```

```

*
* @param object
* @return
*/
public Object wrapPersistentEntities(Object object) {
    if (object instanceof Collection<?>) {
        Collection<?> collection = (Collection<?>) object;
        if (collection instanceof UserSet) {
            object = wrapUserSetEntries((UserSet) collection);
        } else if (collection instanceof SortedSet<?>) {
            object = wrapCollectionEntries(new TreeSet<Object>(((SortedSet)
collection)
                .comparator()), collection);
        } else if (collection instanceof Set<?>) {
            object = wrapCollectionEntries(new
HashSet<Object>(collection.size()), collection);
        } else if (collection instanceof List<?>) {
            object = wrapCollectionEntries(new
ArrayList<Object>(collection.size()), collection);
        } else {
            throw new RuntimeException("unexpected collection type: " +
object);
        }
    } else {
        object = wrapEntity(object);
    }
    return object;
}

// TODO: the UserSet may not be needed
protected UserSet wrapUserSetEntries(UserSet collection) {
    Set<Object> set = new HashSet<Object>(collection.size());
    for (Object entity : collection) {
        set.add(wrapEntity(entity));
    }
    return new UserSetImpl(set);
}

protected Object wrapCollectionEntries(Collection<Object>
newCollection,
    Collection<?> origCollection) {
    for (Object entity : origCollection) {
        newCollection.add(wrapEntity(entity));
    }
    return newCollection;
}

```

```

protected Object wrapEntity(Object entity) {
    if (entity != null) {
        // if an entity is already wrapped, don't add another wrapper
        if (EntityProxyInterceptor.isEntityProxy(entity)) {
            return entity;
        }
        // maybe a hibernate proxy
        if (entity instanceof HibernateProxy) {
            entity = ((HibernateProxy) entity).getHibernateLazyInitializer()
                .getImplementation();
        }
        // only wrap entity objects
        if (entity.getClass().getAnnotation(Entity.class) == null) {
            return entity;
        }
        log.debug("wrapping entity: " + entity);
        return getEntityFactory(entity).newInstance(
            new EntityProxyInterceptor(persistenceContextHelper, this,
            entity));
    }
    return null;
}

private Factory getEntityFactory(Object entity) {
    Class<?> entityType = entity.getClass();
    Factory f = factoryMap.get(entityType);
    if (f == null) {
        Enhancer e = new Enhancer();
        e.setSuperclass(entityType);
        e.setCallback(new EntityProxyInterceptor(null, null, null));
        f = (Factory) e.create();
        factoryMap.put(entityType, f);
    }
    return f;
}

```

domainobjectwrappingadvice.java

```

/*
 * $Id: DomainObjectWrappingAdvice.java,v 1.2 2009/01/26 10:19:06
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

```

```

import java.util.Collection;
import java.util.Map;

import org.apache.log4j.Logger;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.aop.framework.Advised;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.SystemInitializer;
import edu.harvard.fas.rregan.command.Command;

/**
 * AOP Advice that wraps domain objects in proxies for transactional
activities
 * after being detached from the JPA session/transactions.
 *
 * @author ron
 */
@Aspect
@Component("domainObjectWrappingAdvice")
@Scope("singleton")
public class DomainObjectWrappingAdvice {
    protected static final Logger log =
        Logger.getLogger(DomainObjectWrappingAdvice.class);

    private final DomainObjectWrapper domainObjectWrapper;

    /**
     * @param domainObjectWrapper
     */
    @Autowired
    public DomainObjectWrappingAdvice(DomainObjectWrapper
        domainObjectWrapper) {
        this.domainObjectWrapper = domainObjectWrapper;
    }

    /**
     * After the execution any method on a repository or a getter on a
command,
     * wrap returned entities in a proxy so that references to properties
always

```

```

 * return the latest persisted state of the object from the database
and
 * load lazy objects as needed.<br>
 * TODO: remove the logging and make this an after advice
 *
 * @param pjp
 * @return the return type of the repository method with entity
objects
 *         wrapped in a proxy.
 * @throws Throwable
 */
@Around(value = "execution(*
edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository.*(..)) ||
execution(* edu.harvard.fas.rregan.command.Command+.get*(..))")
public Object wrapReturnedObject(ProceedingJoinPoint pjp) throws
Throwable {
    // unwrap EntityProxies before passing into repository methods
    Object[] args = pjp.getArgs();
    for (int i = 0; i < args.length; i++) {
        args[i] = EntityProxyInterceptor.unwrap(args[i]);
    }

    StringBuffer sb = null;
    if (log.isDebugEnabled()) {
        sb = new StringBuffer();
        sb.append(pjp.getTarget().getClass().getSimpleName());
        sb.append(".");
        sb.append(pjp.getSignature().getName());
        sb.append("(");
        for (int i = 0; i < args.length; i++) {
            if (i != 0) {
                sb.append(", ");
            }
            if (args[i] instanceof Map) {
                if (((Map) args[i]).size() > 10) {
                    sb.append("<big map>");
                } else {
                    sb.append(args[i]);
                }
            } else if (args[i] instanceof Collection) {
                if (((Collection) args[i]).size() > 10) {
                    sb.append("<big collection>");
                } else {
                    sb.append(args[i]);
                }
            } else {
                sb.append(args[i]);
            }
        }
        sb.append(")");
    }
}

    }
}
sb.append(")");
}

Object retVal = pjp.proceed(args);
if (log.isDebugEnabled() && (sb != null)) {
    sb.append(" -> ");
    if (retVal instanceof Map) {
        if (((Map) retVal).size() > 10) {
            sb.append("<big map>");
        } else {
            sb.append(retVal);
        }
    } else if (retVal instanceof Collection) {
        if (((Collection) retVal).size() > 10) {
            sb.append("<big collection>");
        } else {
            sb.append(retVal);
        }
    } else {
        sb.append(retVal);
    }
    log.debug(sb.toString());
}

// TODO: shouldn't use sun.reflect.Reflection
// don't wrap an entity if the repository is called from inside an
// initializer or command (something that has an active
transaction),
// this is a hack because there is no way to describe this pointcut
in
// Spring, should use Aspectj cflow or cflowbelow pointcut
boolean wrapReturnVal = true;
try {
    for (int index = 1; index < Integer.MAX_VALUE; index++) {
        Class<?> callerClass =
sun.reflect.Reflection.getCallerClass(index);
        if (callerClass == null) {
            // reached the top of the stack
            break;
        }
        if ((Command.class.isAssignableFrom(callerClass) ||
SystemInitializer.class
            .isAssignableFrom(callerClass))
            && (!Advised.class.isAssignableFrom(callerClass))) {
            wrapReturnVal = false;
        }
    }
}

```

```

        break;
    }
} catch (Throwable t) {
    log.warn("terminated search of call stack by a throw: " + t, t);
}

if (wrapRetVal) {
    retVal = domainObjectWrapper.wrapPersistentEntities(retval);
}
return retVal;
}
}

```

domainuserinitializer.java

```

/*
 * $Id: DomainUserInitializer.java,v 1.8 2009/03/29 11:59:30 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.DomainAdminUserRole;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

@Component("domainUserInitializer")
@Scope("prototype")
public class DomainUserInitializer extends AbstractSystemInitializer {

    private final UserRepository userRepository;
    private final EditUserCommand command;
    private final CommandHandler commandHandler;

    @Autowired
    public DomainUserInitializer(UserRepository userRepository,
        CommandHandler commandHandler,

```

```

        EditUserCommand command) {
    super(100);
    this.userRepository = userRepository;
    this.commandHandler = commandHandler;
    this.command = command;
}

@Override
public void initialize() {
    try {
        // TODO: the domain admin user role has been disabled so the domain
        // user is disabled as well
        // userRepository.findUserByUsername("domain");
    } catch (NoSuchUserException e) {
        try {
            command.setUsername("domain");
            command.setPassword("domain");
            command.setRepassword("domain");
            command.setEmailAddress("rreganjr@acm.org");
            command.setOrganizationName("Requel");
            command.addUserRoleName(DomainAdminUserRole.getRoleName(DomainAdmi
nUserRole.class));
            commandHandler.execute(command);
        } catch (Exception e2) {
            log.error("failed to initialize the domain user: " + e2, e2);
        }
    }
}
}

```

downloadbutton.java

```

package edu.harvard.fas.rregan.uiframework.navigation;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.filetransfer.Download;
import nextapp.echo2.app.filetransfer.DownloadProvider;

import org.apache.log4j.Logger;

/**
 * A generic button for sending content to a user as a downloadable
 * file.
 *

```

```

* @author ron
*/
public class DownloadButton extends Button implements ActionListener {
    private static final Logger log =
Logger.getLogger(DownloadButton.class);
    static final long serialVersionUID = 0;

    private DownloadProvider downloadProvider;

    /**
     * @param label -
     *          the label for the button
     * @param downloadProvider -
     *          the provider of the content to download
     */
    public DownloadButton(String label, DownloadProvider
downloadProvider) {
        super(label);
        setDownloadProvider(downloadProvider);
        addActionListener(this);
    }

    /**
     * @param label
     */
    public DownloadButton(String label) {
        this(label, null);
    }

    /**
     * @return
     */
    public DownloadProvider getDownloadProvider() {
        return downloadProvider;
    }

    /**
     * @param downloadProvider
     */
    public void setDownloadProvider(DownloadProvider downloadProvider) {
        this.downloadProvider = downloadProvider;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        Download download = new Download();
        download.setProvider(downloadProvider);
    }
}

```

```

        download.setActive(true);
        getApplicationInstance().enqueueCommand(download);
    }
}

```

editactorcommand.java

```

/*
 * $Id: EditActorCommand.java,v 1.5 2009/01/27 09:30:16 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import java.util.Set;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;

/**
 * @author ron
 */
public interface EditActorCommand extends EditTextEntityCommand {

    /**
     * Set the actor to edit.
     *
     * @param actor
     */
    public void setActor(Actor actor);

    /**
     * Get the new or updated actor.
     *
     * @return
     */
    public Actor getActor();

    /**
     * set a single container. helper that creates a set and adds the
     * single
     * container to it and calls setActorContainers().
     *
     * @param actorContainer
     */
}

```

```

public void setActorContainer(ActorContainer actorContainer);

/**
 * Set the actor containers that refer to the actor. This is the
absolute
 * set of referers.<br>
 * Either this or setAddReferers should be used, this takes
precedence.
 *
 * @param referers
 */
public void setActorContainers(Set<ActorContainer> referers);

/**
 * Set the actor containers to add as referers to the actor.<br>
 * Either this or setReferers should be used, setReferers takes
precedence.
 *
 * @param actorContainers
 */
public void setAddActorContainers(Set<ActorContainer>
actorContainers);
}

```

editactorcommandimpl.java

```

/*
 * $Id: EditActorCommandImpl.java,v 1.13 2009/03/30 11:54:30 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;

```

```

import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.EditActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ActorImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editActorCommand")
@Scope("prototype")
public class EditActorCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditActorCommand {

private Set<ActorContainer> actorContainers;
private Set<ActorContainer> addActorContainers;
private Actor actor;
private String text;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@Autowired
public EditActorCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {

```

```

super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

public Actor getActor() {
    return actor;
}

public void setActor(Actor actor) {
    this.actor = actor;
}

public void setActorContainer(ActorContainer actorContainer) {
    if (actorContainer != null) {
        Set<ActorContainer> actorContainers = new
HashSet<ActorContainer>();
        actorContainers.add(actorContainer);
        setActorContainers(actorContainers);
    }
}

@Override
public void setActorContainers(Set<ActorContainer> actorContainers) {
    this.actorContainers = actorContainers;
}

protected Set<ActorContainer> getActorContainers() {
    return actorContainers;
}

protected Set<ActorContainer> getAddActorContainers() {
    return addActorContainers;
}

@Override
public void setAddActorContainers(Set<ActorContainer>
addActorContainers) {
    this.addActorContainers = addActorContainers;
}

@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

@Override
public void execute() {
    User editedBy = getProjectRepository().get(getEditedBy());
    Set<ActorContainer> containers = null;
    ActorImpl actorImpl = (ActorImpl) getActor();
    ProjectOrDomain projectOrDomain = null;
    if (getProjectOrDomain() != null) {
        projectOrDomain = getProjectOrDomain();
    } else if ((getActorContainers() != null) &&
(getActorContainers().size() > 0)) {
        containers = getActorContainers();
        Object container = getActorContainers().iterator().next();
        if (container instanceof ProjectOrDomain) {
            projectOrDomain = (ProjectOrDomain) container;
        } else if (container instanceof ProjectOrDomainEntity) {
            projectOrDomain = ((ProjectOrDomainEntity)
container).getProjectOrDomain();
        } else {
            // TODO: throw an exception?
        }
    } else if ((getAddActorContainers() != null) &&
(getAddActorContainers().size() > 0)) {
        containers = getAddActorContainers();
        Object container = getAddActorContainers().iterator().next();
        if (container instanceof ProjectOrDomain) {
            projectOrDomain = (ProjectOrDomain) container;
        } else if (container instanceof ProjectOrDomainEntity) {
            projectOrDomain = ((ProjectOrDomainEntity)
container).getProjectOrDomain();
        } else {
            // TODO: throw an exception?
        }
    } else if (getActor() != null) {
        projectOrDomain = getActor().getProjectOrDomain();
    }
    projectOrDomain = getRepository().get(projectOrDomain);

    // check for uniqueness
    try {
        Actor existing =
getProjectRepository().findActorByProjectOrDomainAndName(
            projectOrDomain, getName());
        if (actorImpl == null) {

```

```

        throw EntityException.uniquenessConflict(Actor.class, existing,
FIELD_NAME,
    EntityExceptionActionType.Creating);
} else if (existing.equals(actorImpl)) {
    throw EntityException.uniquenessConflict(Actor.class, existing,
FIELD_NAME,
    EntityExceptionActionType.Updating);
}
} catch (NoSuchEntityException e) {

if (actorImpl == null) {
    actorImpl = getProjectRepository().persist(
        new ActorImpl(projectOrDomain, editedBy, getName(), ""));
    projectOrDomain.getActors().add(actorImpl);
} else if (getName() != null) {
    actorImpl.setName(getName());
}

if (actorImpl == null) {
    actorImpl = getProjectRepository().persist(
        new ActorImpl(projectOrDomain, editedBy, getName(), getText()));
} else {
    if (getName() != null) {
        actorImpl.setName(getName());
    }
    if (getText() != null) {
        actorImpl.setText(getText());
    }
}

actorImpl.getReferers().add(projectOrDomain);
projectOrDomain.getActors().add(actorImpl);

if (containers != null) {
// TODO: equals on a set seems weird
    if (containers.equals(getActorContainers())) {
        actorImpl.getReferers().clear();
    }
    for (ActorContainer container : containers) {
        container = getRepository().get(container);
        actorImpl.getReferers().add(container);
    }
}
setActor(getProjectRepository().merge(actorImpl));

if (containers != null) {

```

```

// TODO: equals on a set seems weird
if (containers.equals(getActorContainers())) {
    actorImpl.getReferers().clear();
}
for (ActorContainer container : containers) {
    container = getRepository().get(container);
    container.getActors().add(actorImpl);
}
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        // TODO: because getActor() is being called locally in the
        // command, the DomainObjectWrappingAdvice that wraps persistence
        // objects with a DomainObjectWrapper isn't getting invoked.
        // Reloading the object through a repository solves the problem,
        but
        // using aspectj may be better.
        getAssistantManager().analyzeActorgetRepository().get(getActor())
    }
}
}
```

editaddactortoprojectpositioncommand.java

```

/*
 * $Id: EditAddActorToProjectPositionCommand.java,v 1.1 2008/09/19
21:56:41 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * Create or edit a position on a LexicalIssue for adding an actor to
the
 * project.
 *
 * @author ron
 */

```

```

public interface EditAddActorToProjectPositionCommand extends
EditPositionCommand {
/**
 * @param projectOrDomain -
 *          the project or domain to add the actor to.
 */
public void setProjectOrDomain(ProjectOrDomain projectOrDomain);
}

```

editaddactortoprojectpositioncommandimpl.java

```

/*
 * $Id: EditAddActorToProjectPositionCommandImpl.java,v 1.5 2009/02/13
12:07:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.annotation.impl.command.EditPositionCommandImpl;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.EditAddActorToProjectPositionCommand;
import edu.harvard.fas.rregan.requel.project.impl.AddActorPosition;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editAddActorToProjectPositionCommand")
@Scope("prototype")

```

```

public class EditAddActorToProjectPositionCommandImpl extends
EditPositionCommandImpl implements
EditAddActorToProjectPositionCommand {

private final ProjectRepository projectRepository;
private ProjectOrDomain projectOrDomain;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
@.Autowired
public EditAddActorToProjectPositionCommandImpl(CommandHandler
commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository annotationRepository, ProjectRepository
projectRepository) {
super(commandHandler, annotationCommandFactory,
annotationRepository);
this.projectRepository = projectRepository;
}

@Override
public void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
this.projectOrDomain = projectOrDomain;
}

protected ProjectOrDomain getProjectOrDomain() {
return projectOrDomain;
}

protected ProjectRepository getProjectRepository() {
return projectRepository;
}

@Override
public void execute() throws Exception {
ProjectOrDomain projectOrDomain =
getRepository().get(getProjectOrDomain());
User editedBy = getRepository().get(getEditedBy());
LexicalIssue issue = (LexicalIssue) getRepository().get(getIssue());
AddActorPosition position = getPosition();
if (position == null) {
try {
position =
getRepository().findAddActorPosition(projectOrDomain,

```

```

        issue.getWord());
    } catch (NoSuchPositionException e) {
        position = getRepository().persist(new AddActorPosition(getText(),
            editedBy));
    }
} else {
    position.setText(getText());
}
if (issue != null) {
    position.getIssues().add(issue);
}
position = getRepository().merge(position);
setPosition(position);

// add the position to the issue after it has been merged so that if
it
// is a proxy it will be unwrapped by the framework.
if (issue != null) {
    issue.getPositions().add(position);
}
setIssue(issue);
}

@Override
public AddActorPosition getPosition() {
    return (AddActorPosition) super.getPosition();
}

protected void setPosition(AddActorPosition position) {
    super.setPosition(position);
}
}

```

editaddwordtodictionarypositioncommand.java

```

/*
 * $Id: EditAddWordToDictionaryPositionCommand.java,v 1.3 2008/08/13
10:16:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

/**
 * Create or edit a position on a LexicalIssue for adding a word to
the
 * dictionary.

```

```

        *
        * @author ron
        */
public interface EditAddWordToDictionaryPositionCommand extends
EditPositionCommand {
}

```

editaddwordtodictionarypositioncommandimpl.java

```

/*
 * $Id: EditAddWordToDictionaryPositionCommandImpl.java,v 1.12
2009/02/16 10:10:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditAddWordToDictiona
ryPositionCommand;
import
edu.harvard.fas.rregan.requel.annotation.impl.AddWordToDictionaryPosi
tion;
import
edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import
edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editAddWordToDictionaryPositionCommand")
@Scope("prototype")
public class EditAddWordToDictionaryPositionCommandImpl extends
EditPositionCommandImpl implements
EditAddWordToDictionaryPositionCommand {

}

```

```

 * @param commandHandler
 * @param annotationCommandFactory
 * @param repository
 */
@.Autowired
public EditAddWordToDictionaryPositionCommandImpl(CommandHandler
commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository repository) {
    super(commandHandler, annotationCommandFactory, repository);
}

@Override
public void execute() throws Exception {
    User editedBy = getRepository().get(getEditedBy());
    LexicalIssue issue = (LexicalIssue) getRepository().get(getIssue());
    AddWordToDictionaryPosition position = getPosition();
    if (position == null) {
        try {
            // search for an existing issue by word
            position =
                getAnnotationRepository().findAddWordToDictionaryPosition(
                    issue.getGroupingObject(), issue.getWord());
        } catch (NoSuchPositionException e) {
            position = getRepository().persist(
                new AddWordToDictionaryPosition(getText(), editedBy));
        }
    } else {
        position.setText(getText());
        position = getRepository().merge(position);
    }
    if (issue != null) {
        position.getIssues().add(issue);
    }
    setPosition(position);
    // add the position to the issue after it has been merged so that if
    // is a
    // proxy it will be unwrapped by the framework.
    if (issue != null) {
        issue.getPositions().add(position);
        setIssue(issue);
    }
}

@Override
public AddWordToDictionaryPosition getPosition() {

```

```

    return (AddWordToDictionaryPosition) super.getPosition();
}

public void setPosition(AddWordToDictionaryPosition position) {
    super.setPosition(position);
}

```

editaddwordtогlossarypositioncommand.java

```

/*
 * $Id: EditAddWordToGlossaryPositionCommand.java,v 1.1 2008/08/13
 * 10:16:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * Create or edit a position on a LexicalIssue for adding a word to
the project
 * glossary.
 *
 * @author ron
 */
public interface EditAddWordToGlossaryPositionCommand extends
EditPositionCommand {
    /**
     * @param projectOrDomain -
     *          the project or domain to add the term to.
     */
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain);
}

```

editaddwordtогlossarypositioncommandimpl.java

```

/*
 * $Id: EditAddWordToGlossaryPositionCommandImpl.java,v 1.8 2009/02/13
 * 12:07:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import
edu.harvard.fas.rregan.requel.annotation.impl.command.EditPositionComm
andImpl;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.EditAddWordToGlossaryPos
itionCommand;
import
edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editAddWordToGlossaryPositionCommand")
@Scope("prototype")
public class EditAddWordToGlossaryPositionCommandImpl extends
EditPositionCommandImpl implements
EditAddWordToGlossaryPositionCommand {

    private final ProjectRepository projectRepository;
    private ProjectOrDomain projectOrDomain;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param annotationRepository
     * @param projectRepository
     */
    @Autowired
    public EditAddWordToGlossaryPositionCommandImpl(CommandHandler
commandHandler,
        AnnotationCommandFactory annotationCommandFactory,

```

```

        AnnotationRepository annotationRepository, ProjectRepository
projectRepository) {
    super(commandHandler, annotationCommandFactory,
annotationRepository);
    this.projectRepository = projectRepository;
}

@Override
public void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
    this.projectOrDomain = projectOrDomain;
}

protected ProjectOrDomain getProjectOrDomain() {
    return projectOrDomain;
}

@Override
public AddGlossaryTermPosition getPosition() {
    return (AddGlossaryTermPosition) super.getPosition();
}

protected void setPosition(AddGlossaryTermPosition position) {
    super.setPosition(position);
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public void execute() throws Exception {
    ProjectOrDomain projectOrDomain =
getRepository().get(getProjectOrDomain());
    User editedBy = getRepository().get(getEditedBy());
    LexicalIssue issue = (LexicalIssue) getRepository().get(getIssue());
    AddGlossaryTermPosition position = getPosition();
    if (position == null) {
        try {
            position =
getProjectRepository().findAddGlossaryTermPosition(projectOrDomain,
                issue.getWord());
        } catch (NoSuchPositionException e) {
            position = getRepository()
                .persist(new AddGlossaryTermPosition(getText(), editedBy));
        }
    } else {
        position.setText(getText());
    }
}

```

```

        }
        if (issue != null) {
            position.getIssues().add(issue);
        }
        position = getRepository().merge(position);
        setPosition(position);
        // add the position to the issue after it has been merged so that if
        // it
        // is a proxy it will be unwrapped by the framework.
        if (issue != null) {
            issue.getPositions().add(position);
        }
        setIssue(issue);
    }
}

```

editannotationcommand.java

```

/*
 * $Id: EditAnnotationCommand.java,v 1.1 2008/09/04 09:47:19 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Base class for annotation editing commands.
 *
 * @author ron
 */
public interface EditAnnotationCommand extends EditCommand {

    /**
     * Set the object used for grouping the annotation.
     *
     * @param groupingObject -
     *          a persistent object
     */
    public void setGroupingObject(Object groupingObject);

    /**
     * Set the text for the issue.
     */

```

```

        *
        * @param text
        */
    public void setText(String text);

    /**
     * Set the thing being annotated with the issue.
     *
     * @param annotatable -
     *          set the entity being annotated.
     */
    public void setAnnotatable(Annotatable annotatable);
}

```

editargumentcommand.java

```

/*
 * $Id: EditArgumentCommand.java,v 1.3 2009/02/17 11:50:48 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Create or edit an argument for a position of an issue.
 *
 * @author ron
 */
public interface EditArgumentCommand extends EditCommand {

    /**
     * Set the argument to edit.
     *
     * @param argument
     */
    public void setArgument(Argument argument);

    /**
     * @return the new or updated argument.
     */
    public Argument getArgument();
}

```

```

/**
 * The position that the argument is for or against.
 *
 * @param position
 */
public void setPosition(Position position);

/**
 * Set the text for the argument.
 *
 * @param text
 */
public void setText(String text);

/**
 * Set the support level of the argument for or against the position.
 *
 * @param supportLevelName
 */
public void setSupportLevelName(String supportLevelName);
}

```

editargumentcommandimpl.java

```

/*
 * $Id: EditArgumentCommandImpl.java,v 1.7 2009/02/17 11:50:47 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import
edu.harvard.fas.rregan.requel.annotation.ArgumentPositionSupportLevel;
import edu.harvard.fas.rregan.requel.annotation.Position;

```

```

import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand;
import edu.harvard.fas.rregan.requel.annotation.impl.ArgumentImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editArgumentCommand")
@Scope("prototype")
public class EditArgumentCommandImpl extends AbstractEditCommand
implements EditArgumentCommand {

    private Position position;
    private Argument argument;
    private String text;
    private String supportLevelName;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditArgumentCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    /**
     * @see
     */
    @Override
    public Argument getArgument() {
        return argument;
    }

    /**
     * @see
     */
    @Override
    public Argument getArgument(edu.harvard.fas.rregan.requel.annotation.Argu
ment)

```

```

*/
@Override
public void setArgument(Argument argument) {
    this.argument = argument;
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand#setPosition(edu.harvard.fas.rregan.requel.annotation.Position)
 */
@Override
public void setPosition(Position position) {
    this.position = position;
}

protected Position getPosition() {
    return position;
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand#setSupportLevelName(java.lang.String)
 */
@Override
public void setSupportLevelName(String supportLevelName) {
    this.supportLevelName = supportLevelName;
}

protected String getSupportLevelName() {
    return supportLevelName;
}

protected ArgumentPositionSupportLevel getSupportLevel() {
    return ArgumentPositionSupportLevel.valueOf(getSupportLevelName());
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.EditArgumentCommand#setText(java.lang.String)
 */
@Override
public void setText(String text) {
    this.text = text;
}

```

```

protected String getText() {
    return text;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    validate();
    User editedBy = getRepository().get(getEditedBy());
    Position position = getRepository().get(getPosition());
    ArgumentImpl argumentImpl = (ArgumentImpl) getArguments();
    if (argumentImpl == null) {
        argumentImpl = getRepository().persist(
            new ArgumentImpl(position, getText(), getSupportLevel(),
                editedBy));
    } else {
        argumentImpl.setText(getText());
        argumentImpl.setSupportLevel(getSupportLevel());
        argumentImpl = getRepository().merge(argumentImpl);
    }
    setArgument(argumentImpl);
    if (position != null) {
        position.getArguments().add(argumentImpl);
        setPosition(position);
    }
}

protected void validate() {
    if ((getText() == null) || "".equals(getText().trim())) {
        throw EntityValidationException.emptyRequiredProperty(Argument.class,
            getArguments(),
            "text", EntityExceptionActionType.Updating);
    }
    if ((getSupportLevelName() == null) ||
        "".equals(getSupportLevelName().trim())) {
        throw EntityValidationException.emptyRequiredProperty(Argument.class,
            getArguments(),
            "supportLevel", EntityExceptionActionType.Updating);
    }
    try {
        if (getSupportLevel() == null) {

```

```
        throw EntityValidationException.validationFailed(Argument.class,
"supportLevel",
        "The specified support level \\" + getSupportLevelName()
        + "\\" is not valid.");
    }
} catch (Exception e) {
    throw EntityValidationException.validationFailed(Argument.class,
"supportLevel",
    "The specified support level \\" + getSupportLevelName() + "\\" is
not valid.");
}
}
```

editchangespellingpositioncommand.java

```
/*
 * $Id: EditChangeSpellingPositionCommand.java,v 1.4 2008/07/30
08:06:48 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

/**
 * @author ron
 */
public interface EditChangeSpellingPositionCommand extends
EditPositionCommand {

    /**
     * Set the proposed word to replace the misspelled word in the
annotatable
     * entity.
     *
     * @param proposedWord -
     *          the proposed word to correct the word in the
annotatable
     *          entity.
     */
    public void setProposedWord(String proposedWord);
}
```

edithangespellingpositioncommandimpl.java

/ *

```
* $Id: EditChangeSpellingPositionCommandImpl.java,v 1.13 2009/02/16
10:10:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditChangeSpellingPos
itionCommand;
import
edu.harvard.fas.rregan.requel.annotation.impl.ChangeSpellingPosition;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Create or edit a position for changing the spelling of a word in an
entity.
 *
 * @author ron
 */
@Controller("editChangeSpellingPositionCommand")
@Scope("prototype")
public class EditChangeSpellingPositionCommandImpl extends
EditPositionCommandImpl implements
EditChangeSpellingPositionCommand {

    private String proposedWord;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditChangeSpellingPositionCommandImpl(CommandHandler
commandHandler,
```

```

AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository) {
super(commandHandler, annotationCommandFactory, repository);
}

protected String getProposedWord() {
return proposedWord;
}

public void setProposedWord(String proposedWord) {
this.proposedWord = proposedWord;
}

@Override
public void execute() throws Exception {
User editedBy = getRepository().get(getEditedBy());
LexicalIssue issue = (LexicalIssue) getRepository().get(getIssue());
ChangeSpellingPosition position = getPosition();
if (position == null) {
try {
// search for an existing position
position =
getRepository().findChangeSpellingPosition(issue,
getProposedWord());
} catch (NoSuchPositionException e) {
position = getRepository().persist(
new ChangeSpellingPosition(getText(), editedBy,
getProposedWord()));
}
} else {
position.setText(getText());
position.setProposedWord(getProposedWord());
position = getRepository().merge(position);
}
if (issue != null) {
position.getIssues().add(issue);
}
setPosition(position);

// add the position to the issue after it has been merged so that if
it
// is a proxy it will be unwrapped by the framework.
if (issue != null) {
issue.getPositions().add(position);
setIssue(issue);
}
}
}

```

```

@Override
public ChangeSpellingPosition getPosition() {
return (ChangeSpellingPosition) super.getPosition();
}

/**
 * @param position
 */
public void setPosition(ChangeSpellingPosition position) {
super.setPosition(position);
}
}

```

editcommand.java

```

/*
 * $Id: EditCommand.java,v 1.4 2008/12/13 00:40:51 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * An abstraction of a command where a user is creating or editing
something.
 *
 * @author ron
 */
public interface EditCommand extends Command {

/**
 * @param editedBy -
 *          the user making the change to the object being edited
 */
public void setEditedBy(User editedBy);
}

```

editdictionarywordcommand.java

```

/*
 * $Id $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;

/**
 * Create or Edit a word in the dictionary.
 *
 * @author ron
 */
public interface EditDictionaryWordCommand extends Command {

    /**
     * The text of the word.
     *
     * @param lemma
     */
    public void setLemma(String lemma);
}

```

editdictionarywordcommandimpl.java

```

/*
 * $Id: EditDictionaryWordCommandImpl.java,v 1.1 2008/12/13 00:39:53
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditDictionaryWordCommand;

/**
 * @author ron
 */
@Controller("editDictionaryWordCommand")
@Scope("prototype")
public class EditDictionaryWordCommandImpl extends
AbstractDictionaryCommand implements

```

```

EditDictionaryWordCommand {

private String lemma;

/**
 * @param dictionaryRepository
 */
@.Autowired
public EditDictionaryWordCommandImpl(DictionaryRepository
dictionaryRepository) {
super(dictionaryRepository);
}

protected String getLemma() {
return lemma;
}

public void setLemma(String lemma) {
this.lemma = lemma;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
try {
getDictionaryRepository().addToDictionary(getLemma());
} catch (Exception e) {
log.error(e, e);
}
}
}

```

editglossarytermcommand.java

```

/*
 * $Id: EditGlossaryTermCommand.java,v 1.5 2009/01/27 09:30:16 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import java.util.Set;

```

```

import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;

/**
 * @author ron
 */
public interface EditGlossaryTermCommand extends EditTextEntityCommand
{

    /**
     * Set the project or domain entities that refer to the glossary
     * term. This
     * is the absolute set of referers.<br>
     * Either this or setAddReferers should be used, this takes
     precedence.
     *
     * @param referers
     */
    public void setReferers(Set<ProjectOrDomainEntity> referers);

    /**
     * Set the project or domain entities to add as referers of the
     * glossary
     * term. This is for adding new referers to the existing set.<br>
     * Either this or setReferers should be used, setReferers takes
     precedence.
     *
     * @param referers
     */
    public void setAddReferers(Set<ProjectOrDomainEntity> referers);

    /**
     * Set the glossary term to edit.
     *
     * @param glossaryTerm
     */
    public void setGlossaryTerm(GlossaryTerm glossaryTerm);

    /**
     * Get the new or updated glossary term.
     *
     * @return
     */
    public GlossaryTerm getGlossaryTerm();
}

```

```

    * Set the glossary term this term is an alternate case of.
    *
    * @param canonicalTerm
    */
    public void setCanonicalTerm(GlossaryTerm canonicalTerm);

    /**
     * Set the project or domain this term is being added to.
     *
     * @param projectOrDomain
     */
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain);
}

```

editglossarytermcommandimpl.java

```

/*
 * $Id: EditGlossaryTermCommandImpl.java,v 1.22 2009/03/30 11:54:27
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.EditGlossaryTermCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.GlossaryTermImpl;

```

```

import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editGlossaryTermCommand")
@Scope("prototype")
public class EditGlossaryTermCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditGlossaryTermCommand {

    private Set<ProjectOrDomainEntity> referers;
    private Set<ProjectOrDomainEntity> addReferers;
    private GlossaryTerm glossaryTerm;
    private GlossaryTerm canonicalTerm;
    private String definition;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public EditGlossaryTermCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public void setReferers(Set<ProjectOrDomainEntity> referers) {
        this.referers = referers;
    }

    protected Set<ProjectOrDomainEntity> getReferers() {
        return referers;
    }
}

@Override
public void setAddReferers(Set<ProjectOrDomainEntity> addReferers) {
    this.addReferers = addReferers;
}

protected Set<ProjectOrDomainEntity> getAddReferers() {
    return addReferers;
}

@Override
public GlossaryTerm getGlossaryTerm() {
    return glossaryTerm;
}

@Override
public void setGlossaryTerm(GlossaryTerm glossaryTerm) {
    this.glossaryTerm = glossaryTerm;
}

/**
 * @return the primary/preferred term the term being edited is an
alternative
 *          to.
 */
public GlossaryTerm getCanonicalTerm() {
    return canonicalTerm;
}

@Override
public void setCanonicalTerm(GlossaryTerm canonicalTerm) {
    this.canonicalTerm = canonicalTerm;
}

/**
 * @return the definition of the term.
 */
public String getDefinition() {
    return definition;
}

@Override
public void setText(String definition) {
    this.definition = definition;
}

@Override

```

```

public void execute() {
    ProjectOrDomain projectOrDomain =
getProjectRepository().get(getProjectOrDomain());
    GlossaryTerm canonicalTerm =
getProjectRepository().get(getCanonicalTerm());
    User editedBy = getProjectRepository().get(getEditedBy());
    GlossaryTermImpl glossaryTermImpl = (GlossaryTermImpl)
getGlossaryTerm();
    // check for uniqueness
    try {
        String name = getName();
        if (name == null) {
            if (glossaryTermImpl == null) {
                throw
EntityValidationException.emptyRequiredProperty(GlossaryTerm.class,
null,
                    "name", EntityExceptionActionType.Updating);
            }
            name = glossaryTermImpl.getName();
        }
        GlossaryTerm existing =
getProjectRepository().findGlossaryTermForProjectOrDomain(
            projectOrDomain, name);
        if (glossaryTermImpl == null) {
            throw EntityException.uniquenessConflict(GlossaryTerm.class,
existing, FIELD_NAME,
                EntityExceptionActionType.Creating);
        } else if (!existing.equals(glossaryTermImpl)) {
            throw EntityException.uniquenessConflict(GlossaryTerm.class,
existing, FIELD_NAME,
                EntityExceptionActionType.Updating);
        }
    } catch (NoSuchEntityException e) {

        if (glossaryTermImpl == null) {
            glossaryTermImpl = getProjectRepository().persist(
                new GlossaryTermImpl(projectOrDomain, getName(), editedBy));
            projectOrDomain.getGlossaryTerms().add(glossaryTermImpl);
        } else if (getName() != null) {
            glossaryTermImpl.setName(getName());
        }
        if (getDefinition() != null) {
            glossaryTermImpl.setText(getDefinition());
        }
        if (canonicalTerm != null) {
            glossaryTermImpl.setCanonicalTerm(canonicalTerm);
        }
    }
}
}

glossaryTermImpl = getProjectRepository().merge(glossaryTermImpl);

if (getReferers() != null) {
    // remove the term for all its referers
    for (ProjectOrDomainEntity entity : glossaryTermImpl.getReferers())
{
    if (entity != null) {
        entity.getGlossaryTerms().remove(glossaryTermImpl);
    }
}
// add the referers to the term and the term to the referers
glossaryTermImpl.getReferers().clear();
for (ProjectOrDomainEntity entity : getReferers()) {
    entity = getProjectRepository().get(entity);
    glossaryTermImpl.getReferers().add(entity);
    entity.getGlossaryTerms().add(glossaryTermImpl);
}
} else if (getAddReferers() != null) {
    for (ProjectOrDomainEntity entity : getAddReferers()) {
        entity = getProjectRepository().get(entity);
        glossaryTermImpl.getReferers().add(entity);
        entity.getGlossaryTerms().add(glossaryTermImpl);
    }
}
if (canonicalTerm != null) {
    canonicalTerm.getAlternateTerms().add(glossaryTermImpl);
}
setCanonicalTerm(canonicalTerm);
setGlossaryTerm(glossaryTermImpl);
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        // TODO: analyze the glossary term?
    }
}
}

editgoalcommand.java
/*
 * $Id: EditGoalCommand.java,v 1.5 2009/01/27 09:30:17 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

*/
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;

/**
 * @author ron
 */
public interface EditGoalCommand extends EditTextEntityCommand {

    /**
     * Set the goal to edit.
     *
     * @param goal
     */
    public void setGoal(Goal goal);

    /**
     * Get the new or updated goal.
     *
     * @return
     */
    public Goal getGoal();

    /**
     * Set the container this goal is being added to.
     *
     * @param goalContainer
     */
    public void setGoalContainer(GoalContainer goalContainer);
}

```

editgoalcommandimpl.java

```

/*
 * $Id: EditGoalCommandImpl.java,v 1.23 2009/03/30 11:54:29 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

```

```

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.EditGoalCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.GoalImpl;
import
edu.harvard.fas.rregan.requel.project.impl.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

    /**
     * @author ron
     */
    @Controller("editGoalCommand")
    @Scope("prototype")
    public class EditGoalCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditGoalCommand {

    private GoalContainer goalContainer;
    private Goal goal;
    private String text;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public EditGoalCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,

```

```

    ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

public Goal getGoal() {
    return goal;
}

public void setGoal(Goal goal) {
    this.goal = goal;
}

@Override
public void setGoalContainer(GoalContainer goalContainer) {
    this.goalContainer = goalContainer;
}

protected GoalContainer getGoalContainer() {
    return goalContainer;
}

@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

@Override
public void execute() {
    GoalContainer goalContainer =
getProjectRepository().get(getGoalContainer());
    User editedBy = getProjectRepository().get(getEditedBy());
    GoalImpl goalImpl = (GoalImpl) getGoal();
    ProjectOrDomain projectOrDomain = null;
    if (goalImpl != null) {
        projectOrDomain = goalImpl.getProjectOrDomain();
    } else if (goalContainer != null) {
        if (goalContainer instanceof ProjectOrDomain) {
            projectOrDomain = (ProjectOrDomain) goalContainer;
        }
    } else if (goalContainer instanceof ProjectOrDomainEntity) {
        projectOrDomain = ((ProjectOrDomainEntity)
goalContainer).getProjectOrDomain();
    } else {
        // TODO: error?
    }
    projectOrDomain = getRepository().get(projectOrDomain);

    // check for uniqueness
    try {
        Goal existing =
getProjectRepository().findGoalByProjectOrDomainAndName(
            projectOrDomain, getName());
        if (goalImpl == null) {
            throw EntityException.uniquenessConflict(Goal.class, existing,
                EditGoalCommand.FIELD_NAME, EntityExceptionActionType.Creating);
        } else if (!existing.equals(goalImpl)) {
            throw EntityException.uniquenessConflict(Goal.class, existing,
                EditGoalCommand.FIELD_NAME, EntityExceptionActionType.Updating);
        }
    } catch (NoSuchEntityException e) {

        if (goalImpl == null) {
            goalImpl = getProjectRepository().persist(
                new GoalImpl(projectOrDomain, editedBy, getName(), getText()));
        } else {
            goalImpl.setName(getName());
            goalImpl.setText(getText());
        }
        if (goalContainer != null) {
            goalImpl.getReferers().add(goalContainer);
        }
        setGoal(getProjectRepository().merge(goalImpl));
        if (goalContainer != null) {
            goalContainer.getGoals().add(goalImpl);
        }
        setGoalContainer(goalContainer);
    }

    @Override
    public void invokeAnalysis() {
        if (isAnalysisEnabled()) {
            getAssistantManager().analyzeGoal(getGoal());
        }
    }
}

```

```
}
```

editgoalrelationcommand.java

```
/*
 * $Id: EditGoalRelationCommand.java,v 1.4 2009/02/13 12:08:05 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.AnalyzableEditCommand;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * Create or Edit the relationship between two goals.
 *
 * @author ron
 */
public interface EditGoalRelationCommand extends AnalyzableEditCommand
{

    /**
     * @param projectOrDomain -
     *          the project or domain that the goals in the relation
     *          belong
     *          to. this is required if goalRelation is null.
     */
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain);

    /**
     * @param goalRelation -
     *          the goal relation to edit, or null for a new relation.
     */
    public void setGoalRelation(GoalRelation goalRelation);

    /**
     * @return after execute this returns the created or edited relation.
     */
    public GoalRelation getGoalRelation();

    /**
     * @param goalName -
     */

}
```

```
        *          the name of the goal that is the origin of the
        *          relationship.
        */
    public void setFromGoal(String goalName);

    /**
     * @param goalName -
     *          the name of the goal that is the target of the
     *          relationship.
     */
    public void setToGoal(String goalName);

    /**
     * @param relationTypeName -
     *          the name of the type of relationship between the goals
     */
    public void setRelationType(String relationTypeName);
}
```

editgoalrelationcommandimpl.java

```
/*
 * $Id: EditGoalRelationCommandImpl.java,v 1.12 2009/03/30 11:54:31
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.GoalRelationType;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
```

```

import
edu.harvard.fas.rregan.requel.project.exception.GoalSelfRelationException;
import edu.harvard.fas.rregan.requel.project.impl.GoalRelationImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editGoalRelationCommand")
@Scope("prototype")
public class EditGoalRelationCommandImpl extends
AbstractProjectCommand implements
EditGoalRelationCommand {

private ProjectOrDomain projectOrDomain;
private GoalRelation goalRelation;
private String fromGoalName;
private String toGoalName;
private String relationTypeName;
private User editedBy;
private boolean analysisEnabled = true;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@Autowired
public EditGoalRelationCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand#
getGoalRelation()
*/
@Override
public GoalRelation getGoalRelation() {
return goalRelation;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand#
setGoalRelation(edu.harvard.fas.rregan.requel.project.GoalRelation)
*/
@Override
public void setGoalRelation(GoalRelation goalRelation) {
this.goalRelation = goalRelation;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand#
setFromGoal(java.lang.String)
*/
@Override
public void setFromGoal(String goalName) {
this.fromGoalName = goalName;
}

protected String getFromGoal() {
return fromGoalName;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand#
setRelationType(java.lang.String)
*/
@Override
public void setRelationType(String relationTypeName) {
this.relationTypeName = relationTypeName;
}

protected String getRelationType() {
return relationTypeName;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand#
setToGoal(java.lang.String)
*/

```

```

/*
@Override
public void setToGoal(String goalName) {
    this.toGoalName = goalName;
}

protected String getToGoal() {
    return toGoalName;
}

@Override
public void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
    this.projectOrDomain = projectOrDomain;
}

protected ProjectOrDomain getProjectOrDomain() {
    return projectOrDomain;
}

/**
 * @see
edu.harvard.fas.rregan.requel.command.EditCommand#setEditedBy(edu.harv
ard.fas.rregan.requel.user.User)
 */
@Override
public void setEditedBy(User user) {
    this.editedBy = user;
}

protected User getEditedBy() {
    return editedBy;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    User editedBy = getRepository().get(getEditedBy());
    ProjectOrDomain projectOrDomain =
getRepository().get(getProjectOrDomain());
    GoalRelationImpl goalRelationImpl = (GoalRelationImpl)
getGoalRelation();
    Goal toGoal =
getRepository().findGoalByProjectOrDomainAndName(getProjectOrDo
main(),
    getToGoal());
}

    Goal fromGoal =
getRepository().findGoalByProjectOrDomainAndName(
    getProjectOrDomain(), getFromGoal());
    GoalRelationType goalRelationType;
    try {
        goalRelationType = GoalRelationType.valueOf(getRelationType());
    } catch (Exception e) {
        throw
EntityValidationException.validationFailed(GoalRelation.class,
"relationType",
        "The goal relation type cannot be " + getRelationType());
    }

    if (toGoal.equals(fromGoal)) {
        throw GoalSelfRelationException.forGoal(toGoal);
    }
    // TODO: check for uniqueness?
    if (goalRelationImpl == null) {
        goalRelationImpl = getProjectRepository().persist(
            new GoalRelationImpl(fromGoal, toGoal, goalRelationType,
editedBy));
    } else {
        goalRelationImpl.setFromGoal(fromGoal);
        goalRelationImpl.setToGoal(toGoal);
        goalRelationImpl.setRelationType(goalRelationType);
        goalRelationImpl = getProjectRepository().merge(goalRelationImpl);
    }
    setGoalRelation(goalRelationImpl);
}

@Override
public void setAnalysisEnabled(boolean analysisEnabled) {
    this.analysisEnabled = analysisEnabled;
}

protected boolean isAnalysisEnabled() {
    return analysisEnabled;
}

@Override
public void invokeAnalysis() {
    // TODO: do goal relations need to be analyzed?
}
}

```

editissuecommand.java

```
/*
 * $Id: EditIssueCommand.java,v 1.4 2008/09/04 09:47:19 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Issue;

/**
 * @author ron
 */
public interface EditIssueCommand extends EditAnnotationCommand {

    /**
     * Set the issue to edit.
     *
     * @param issue
     */
    public void setIssue(Issue issue);

    /**
     * Get the new or updated issue.
     *
     * @return
     */
    public Issue getIssue();

    /**
     * @param mustBeResolved -
     *          set to true if this issue must be resolved.
     */
    public void setMustBeResolved(boolean mustBeResolved);
}
```

editissuecommandimpl.java

```
/*
 * $Id: EditIssueCommandImpl.java,v 1.17 2009/02/17 11:50:47 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.EditIssueCommand;
import edu.harvard.fas.rregan.requel.annotation.impl.IssueImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Create or edit an issue annotation on an annotatable entity.
 *
 * @author ron
 */
@Controller("editIssueCommand")
@Scope("prototype")
public class EditIssueCommandImpl extends AbstractAnnotationCommand
implements EditIssueCommand {

    private Issue issue;
    private boolean mustBeResolved;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditIssueCommandImpl(CommandHandler commandHandler,
                               AnnotationCommandFactory annotationCommandFactory,
                               AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    public Issue getIssue() {
        return issue;
    }
}
```

```

}

public void setIssue(Issue issue) {
    this.issue = issue;
}

protected boolean getMustBeResolved() {
    return mustBeResolved;
}

public void setMustBeResolved(boolean mustBeResolved) {
    this.mustBeResolved = mustBeResolved;
}

@Override
public void execute() {
    validate();
    User editedBy = getRepository().get(getEditedBy());
    Object groupingObject = getRepository().get(getGroupingObject());
    Annotatable annotatable = getRepository().get(getAnnotatable());
    IssueImpl issueImpl = (IssueImpl) getIssue();
    if (issueImpl == null) {
        // TODO: look for an issue with the existing text
        try {
            issueImpl = (IssueImpl)
                getAnnotationRepository().findIssue(groupingObject,
                    annotatable, getText());
        } catch (NoSuchAnnotationException e) {
            issueImpl = getRepository().persist(
                new IssueImpl(groupingObject, getText(), getMustBeResolved(),
                    editedBy));
        }
    } else {
        issueImpl.setText(getText());
        issueImpl.setMustBeResolved(getMustBeResolved());
        issueImpl = getRepository().merge(issueImpl);
    }
    if (annotatable != null) {
        issueImpl.getAnnotatables().add(annotatable);
    }
    setIssue(issueImpl);
    // add the issue to the annotatable after it has been merged so that
    if
        // it is a proxy it will be unwrapped by the framework.
        if (annotatable != null) {
            annotatable.getAnnotations().add(issueImpl);
            setAnnotatable(annotatable);
}

```

```

    }

    protected void validate() {
        if ((getText() == null) || "".equals(getText().trim())) {
            throw EntityValidationException.emptyRequiredProperty(Issue.class,
                getIssue(), "text",
                EntityExceptionActionType.Updating);
        }
    }
}

```

editlexicalissuecommand.java

```

/*
 * $Id: EditLexicalIssueCommand.java,v 1.2 2008/07/31 00:58:15 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.command;

/**
 * @author ron
 */
public interface EditLexicalIssueCommand extends EditIssueCommand {

    /**
     * @param word
     */
    public void setWord(String word);

    /**
     * @param annotatablePropertyName
     */
    public void setAnnotatablePropertyName(String
        annotatablePropertyName);
}

```

editlexicalissuecommandimpl.java

```

/*

```

```

* $Id: EditLexicalIssueCommandImpl.java,v 1.12 2009/02/16 10:10:08
rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditLexicalIssueComma
nd;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editLexicalIssueCommand")
@Scope("prototype")
public class EditLexicalIssueCommandImpl extends EditIssueCommandImpl
implements
EditLexicalIssueCommand {

    private String annotatableEntityPropertyName;
    private String word;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditLexicalIssueCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    }

    protected String getAnnotatableEntityPropertyName() {
        return annotatableEntityPropertyName;
    }

    public void setAnnotatableEntityPropertyName(String
annotatableEntityPropertyName) {
        this.annotatableEntityPropertyName = annotatableEntityPropertyName;
    }

    protected String getWord() {
        return word;
    }

    public void setWord(String word) {
        this.word = word;
    }

    @Override
    public void execute() {
        User editedBy = getRepository().get(getEditedBy());
        Object groupingObject = getRepository().get(getGroupingObject());
        LexicalIssue issue = (LexicalIssue) getIssue();
        Annotatable annotatable = getRepository().get(getAnnotatable());
        if (issue == null) {
            try {
                // search for an existing issue
                if (getAnnotatableEntityPropertyName() == null) {
                    issue =
getAnnotationRepository().findLexicalIssue(getGroupingObject(),
                    annotatable, getWord());
                } else {
                    issue =
getAnnotationRepository().findLexicalIssue(getGroupingObject(),
                    annotatable, getWord(), getAnnotatableEntityPropertyName());
                }
            } catch (NoSuchAnnotationException e) {
                issue = getRepository().persist(
                    new LexicalIssue(groupingObject, getText(), getMustBeResolved(),
                    editedBy,
                    getAnnotatableEntityPropertyName(), getWord()));
            }
        } else {
            if (getText() != null) {
                issue.setText(getText());
            }
        }
    }
}

```

```

if (getWord() != null) {
    issue.setWord(getWord());
}
issue = getRepository().merge(issue);
}
if (annotatable != null) {
    issue.getAnnotatables().add(annotatable);
}
setIssue(issue);
// add the issue to the annotatable after it has been merged so that
if
// it is a proxy it will be unwrapped by the framework.
if (annotatable != null) {
    annotatable.getAnnotations().add(issue);
    setAnnotatable(annotatable);
}
}
}

```

editmode.java

```

/*
 * $Id: EditMode.java,v 1.3 2009/01/21 09:23:21 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframe.panel.editor;

/**
 * Interface to describe something that can have a read-only mode and
edit
 * state. This is typically implemented by a component or panel and
supplied to
 * sub-components to determine if they should support editing or read-
only mode
 * and to indicate to the parent if a data value has been changed.
 *
 * @author ron
 */
public interface EditMode {

/**
 * A component will use this during setup to configure an editor for
editing
 * or for view only. It is expected that this value won't change
after a
 * panel or component is already configured.

```

```

*
* @return true if the mode only allows reading.
*/
public boolean isReadOnlyMode();

/**
 * This is used by a parent component to determine if a sub-component
data
 * value has changed and a save is need.
*
* @return true if the state has changed.
*/
public boolean isStateEdited();

/**
 * An editor component that is a sub-component of a panel or compound
editor
 * calls this method to indicate to the parent that a data value has
changed
 * and needs to be validated and saved.
*
* @param stateEdited
*/
public void setStateEdited(boolean stateEdited);
}
```

editnotecommand.java

```

/*
 * $Id: EditNoteCommand.java,v 1.2 2008/09/04 09:47:19 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Note;

/**
 * Create or edit a note on an annotatable entity.
*
* @author ron
*/
public interface EditNoteCommand extends EditAnnotationCommand {

/**
 * Set the note to edit.
*

```

```

 * @param note
 */
public void setNote(Note note);

/**
 * Get the new or updated note.
 *
 * @return
 */
public Note getNote();
}

```

editnotecommandimpl.java

```

/*
 * $Id: EditNoteCommandImpl.java,v 1.14 2009/02/17 11:50:46 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import edu.harvard.fas.rregan.requel.annotation.Note;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditNoteCommand;
import edu.harvard.fas.rregan.requel.annotation.impl.NoteImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Create or edit a note annotation on an annotatable entity.
 *
 * @author ron
 */

```

```

@Controller("editNoteCommand")
@Scope("prototype")
public class EditNoteCommandImpl extends AbstractAnnotationCommand
implements EditNoteCommand {

    private Note note;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditNoteCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    public Note getNote() {
        return note;
    }

    public void setNote(Note note) {
        this.note = note;
    }

    @Override
    public void execute() {
        validate();
        User editedBy = getRepository().get(getEditedBy());
        Annotatable annotatable = getRepository().get(getAnnotatable());
        Object groupingObject = getRepository().get(getGroupingObject());

        NoteImpl noteImpl = (NoteImpl) getNote();
        if (noteImpl == null) {
            try {
                // see if an existing note exists for the given text
                noteImpl = (NoteImpl)
getAnnotationRepository().findNote(groupingObject,
                    annotatable, getText());
            } catch (NoSuchAnnotationException e) {
                noteImpl = getRepository().persist(
                    new NoteImpl(groupingObject, getText(), editedBy));
            }
        } else {
            noteImpl.setText(getText());
        }
    }
}

```

```

        noteImpl = getRepository().merge(noteImpl);
    }
    if (annotatable != null) {
        noteImpl.getAnnotatables().add(annotatable);
    }
    setNote(noteImpl);
    // add the note to the annotatable after it has been merged so that
if
    // it is a proxy it will be unwrapped by the framework.
    if (annotatable != null) {
        annotatable.getAnnotations().add(noteImpl);
        setAnnotatable(annotatable);
    }
}

protected void validate() {
    if ((getText() == null) || "".equals(getText().trim())) {
        throw EntityValidationException.emptyRequiredProperty(Note.class,
getNote(), "text",
            EntityExceptionActionType.Updating);
    }
}
}

```

editorcomponents.java

```

/*
 * $Id: EditorComponents.java,v 1.3 2009/02/21 10:32:13 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor;

import java.util.HashMap;
import java.util.Map;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * UI component management for an editor based on named fields.

```

```

*
* @author ron
*/
public class EditorComponents implementsEditMode {
    static final long serialVersionUID = 0L;

    private final Map<String, FieldComponents> fieldNameToComponents =
new HashMap<String, FieldComponents>();
    private final Map<Component, String> inputComponentToFieldName = new
HashMap<Component, String>();
    private boolean valid;
    private String generalErrorMessage = "";
    private final EditMode editMode;

    /**
     * @param editMode
     */
    public EditorComponents(EditMode editMode) {
        this.editMode = editMode;
    }

    /**
     * This method should be overloaded to check permissions of the user
making
     * the edits and return true if the user only has read
permissions.<br>
     * use getApp().getUser() to get the user.
     *
     * @return
     */
    public boolean isReadOnlyMode() {
        return editMode.isReadOnlyMode();
    }

    @Override
    public boolean isStateEdited() {
        return editMode.isStateEdited();
    }

    @Override
    public void setStateEdited(boolean stateEdited) {
        editMode.setStateEdited(stateEdited);
    }

    /**
     * @param generalErrorMessage
     */

```

```

public void setGeneralErrorMessage(String generalErrorMessage) {
    this.generalErrorMessage = generalErrorMessage;
}

/**
 * @return
 */
public String getGeneralErrorMessage() {
    return generalErrorMessage;
}

/**
 * @param <T>
 * @param fieldName
 * @param label
 * @param inputComponent
 * @param model
 * @return
 */
public <T extends Component> T addInput(String fieldName, Label
label, T inputComponent,
Object model) {
    if (fieldComponentsExists(fieldName)) {
        throw new IllegalArgumentException("the specified fieldName '" +
fieldName
            + "' already exists.");
    }
    FieldComponents fieldComponents = new FieldComponents();
    fieldNameToComponents.put(fieldName, fieldComponents);
    inputComponentToFieldName.put(inputComponent, fieldName);
    if (isReadOnlyMode()) {
        inputComponent.setEnabled(false);
    }

    Component help = new Label();

    Label message = new Label();

    fieldComponents.label = label;
    fieldComponents.inputComponent = inputComponent;
    fieldComponents.help = help;
    fieldComponents.message = message;
    fieldComponents.model = model;

    ComponentManipulator componentManipulator = ComponentManipulators
        .getManipulator(inputComponent);
    componentManipulator.setModel(inputComponent, model);

    // do this after setting the model
    addListenerToDetectChangesToInput(inputComponent);
    return inputComponent;
}

/**
 * @param fieldName
 * @return The input component for the given input field name.
 */
public Component getInput(String fieldName) {
    return getFieldComponents(fieldName).inputComponent;
}

/**
 * @param fieldName
 * @return
 */
public Label getLabel(String fieldName) {
    return getFieldComponents(fieldName).label;
}

/**
 * @param fieldName
 * @return
 */
public Component getHelp(String fieldName) {
    return getFieldComponents(fieldName).help;
}

/**
 * @param fieldName
 * @return
 */
public Label getMessage(String fieldName) {
    return getFieldComponents(fieldName).message;
}

/**
 * Replace the model of the named input with the new model.
 *
 * @param fieldName -
 *          the name of the field
 * @param model -
 *          the new model.
 */
public void setInputModel(String fieldName, Object model) {
    Component inputComponent = getInput(fieldName);

```

```

ComponentManipulator componentManipulator = ComponentManipulators
    .getManipulator(inputComponent);
componentManipulator.setModel(inputComponent, model);
}

/**
 * @param <T>
 * @param fieldName
 * @param type
 * @return
 */
public <T> T getInputModel(String fieldName, Class<T> type) {
    Component inputComponent = getInput(fieldName);
    ComponentManipulator componentManipulator = ComponentManipulators
        .getManipulator(inputComponent);
    return type.cast(componentManipulator.getModel(inputComponent));
}

/**
 * Set the value of the named input field.
 *
 * @param fieldName -
 *          the name of the input field.
 * @param value -
 *          the new value to set in the field.
 */
public void setInputValue(String fieldName, Object value) {
    Component inputComponent = getInput(fieldName);
    ComponentManipulator componentManipulator = ComponentManipulators
        .getManipulator(inputComponent);
    componentManipulator.setValue(inputComponent, value);
}

/**
 * @param <T>
 * @param fieldName
 * @param type
 * @return
 */
public <T> T getInputValue(String fieldName, Class<T> type) {
    Component inputComponent = getInput(fieldName);
    ComponentManipulator componentManipulator = ComponentManipulators
        .getManipulator(inputComponent);
    return componentManipulator.getValue(inputComponent, type);
}

/**
 * Clear the message text for all input fields.
 */
public void clearValidationMessages() {
    for (FieldComponents fieldComponents :
        fieldNameToComponents.values()) {
        fieldComponents.message.setText("");
    }
    setGeneralErrorMessage("");
}

/**
 * Set the message text of the specified input field.
 *
 * @param fieldName
 * @param message
 */
public void setValidationMessage(String fieldName, String message) {
    getMessage(fieldName).setText(message);
}

/**
 */
public void validateContent() {
    clearValidationMessages();
    // subclasses should call super.validateContent() at the start
    // of all sub-classes.
}

/**
 * Implement saving of the target object here. If the user has
 * invalid input
 * throw a ValidationException. Overriding implementations should
 * call
 * super.save() so that in the future when controls are self
 * validating that
 * code can be put here.
 */
public void save() {
    validateContent();
}

/**
 */
public void cancel() {
    clearValidationMessages();
}

```

```

}

/**
 *
 */
public void delete() {
}

/** 
 * @return
 */
public boolean isValid() {
    return valid;
}

/** 
 * @param valid
 */
public void setValid(boolean valid) {
    this.valid = valid;
}

protected void addListenerToDetectChangesToInput(Component
inputComponent) {
    ComponentManipulators.getManipulator(inputComponent).addListenerToDe
tectChangesToInput(
        this, inputComponent);
}

protected FieldComponents getFieldComponents(String fieldName) {
    if (fieldComponentsExists(fieldName)) {
        return fieldNameToComponents.get(fieldName);
    }
    throw new IllegalArgumentException("the specified fieldName '" +
fieldName
        + "' does not match a field in the editor.");
}

protected boolean fieldComponentsExists(String fieldName) {
    return fieldNameToComponents.containsKey(fieldName);
}

private static class FieldComponents {
    private Label label;
    private Component inputComponent;
    private Object model;
}

```

```

    private Component help;
    private Label message;
}
}
```

editortree.java

```

/*
 * $Id: EditorTree.java,v 1.15 2009/01/23 09:54:25 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import java.io.Serializable;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.Set;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.extras.app.DragSource;
import nextapp.echo2.extras.app.event.DropEvent;
import nextapp.echo2.extras.app.event.DropListener;

import org.apache.log4j.Logger;

import echopointng.Tree;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.DefaultTreeModel;
import echopointng.tree.MutableTreeNode;
import echopointng.tree.TreeModel;
import echopointng.treeTreeNode;
import echopointng.tree.TreePath;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component for building an editor tree based on entity objects.
The tree

```

```

 * nodes are created by NavigatorTreeNodeFactories assigned for
different types
 * of entities.
 *
 * @author ron
 */
public class EditorTree extends Tree implements EditMode {
    private static final Logger log = Logger.getLogger(EditorTree.class);
    static final long serialVersionUID = 0L;

    /**
     * The style name for the drag source label. The label should be
styled with
     * a background image
     */
    public static final String STYLE_NAME_DRAG_SOURCE =
"EditorTree.DragSource";

    /**
     * The style name for the drag source label. The label should be
styled with
     * a background image
     */
    public static final String STYLE_NAME_DROP_TARGET =
"EditorTree.DropTarget";

    static {
        ComponentManipulators.setManipulator(EditorTree.class, new
EditorTreeManipulator());
    }

    private final Map<Class<?>, EditorTreeNodeFactory> treeNodeFactoryMap
= new HashMap<Class<?>, EditorTreeNodeFactory>();
    private final EventDispatcher eventDispatcher;
    private final EditMode editMode;
    private final Map<MutableTreeNode, Component> nodeToDropTargetMap =
new HashMap<MutableTreeNode, Component>();
    private final Map<MutableTreeNode, DragSource> nodeToDragSourceMap =
new HashMap<MutableTreeNode, DragSource>();
    private final Map<Component, MutableTreeNode> dropTargetToNodeMap =
new HashMap<Component, MutableTreeNode>();
    private final Map<DragSource, MutableTreeNode> dragSourceToNodeMap =
new HashMap<DragSource, MutableTreeNode>();
    private final EditorTreeDragAndDropHandler dropHandler;
    private Object rootObject;

    /**

```

```

     * Build a tree starting with the rootObject as the root of the tree,
adding
     * nodes based on the supplied tree node factories, starting with the
     * factory for the rootObject.
     *
     * @param editMode -
     *          an object controlling the edit mode of this tree.
     * @param eventDispatcher
     * @param treeNodeFactories
     * @param rootObject -
     *          the root entity object being edited by the tree.
     * @param disableSelection -
     *          set to true to turn off tree node selection so that
editor
     *          fields at the tree node level will work properly.
     * @param showRootNode -
     *          set to false to not show a node for the root object
being
     *          edited.
     * @param enableDragAndDrop -
     *          a EditorTreeDragAndDropHandler if the tree should
support
     *          dragging and dropping of tree nodes, or null if drag
and drop
     *          should not be enabled.
     */
    public EditorTree(EditMode editMode, EventDispatcher eventDispatcher,
Set<EditorTreeNodeFactory> treeNodeFactories, boolean
disableSelection,
        boolean showRootNode, boolean enableDragAndDrop) {
        super();
        this.eventDispatcher = eventDispatcher;
        this.editMode = editMode;
        if (disableSelection) {
            setSelectionModel(null);
        }
        if (enableDragAndDrop) {
            this.dropHandler = new EditorTreeDragAndDropHandler(this);
        } else {
            this.dropHandler = null;
        }
        setRootVisible(showRootNode);
        setCellRenderer(new EditorTreeCellRenderer());
        for (EditorTreeNodeFactory factory : treeNodeFactories) {
            // if the dropHandler is not null decorate each factory with a
            // factory that decorates each node with drag and drop enabled
            // nodes.
        }
    }
}

```

```

if (dropHandler != null) {
    treeNodeFactoryMap.put(factory.getTargetType(),
        new EditorTreeDragAndDropNodeFactoryDecorator(factory));
} else {
    treeNodeFactoryMap.put(factory.getTargetType(), factory);
}
}

/**
 * @return the object that defines the edit mode and state.
 */
public EditMode getEditMode() {
    return editMode;
}

@Override
public boolean isReadOnlyMode() {
    return editMode.isReadOnlyMode();
}

@Override
public boolean isStateEdited() {
    return editMode.isStateEdited();
}

@Override
public void setStateEdited(boolean stateEdited) {
    editMode.setStateEdited(stateEdited);
}

/**
 * @param node
 * @return return the drop target component for dropping another tree
node
 *          to be inserted where the supplied tree node is located.
 */
public Component getDropTarget(MutableTreeNode node) {
    if (nodeToDropTargetMap.containsKey(node)) {
        return nodeToDropTargetMap.get(node);
    } else {
        Component dropTarget = new Label();
        dropTarget.setStyleName(STYLE_NAME_DROP_TARGET);
        nodeToDropTargetMap.put(node, dropTarget);
        dropTargetToNodeMap.put(dropTarget, node);
        for (DragSource source : nodeToDragSourceMap.values()) {
            source.addDropTarget(dropTarget);
        }
    }
}

```

```
        }
        return dropTarget;
    }
}

/***
 * @param dropTarget
 * @return The tree node that has the supplied dropTarget attached
for drag
 *         and drop.
 */
public MutableTreeNode getNodeForDropTarget(Component dropTarget) {
    return dropTargetToNodeMap.get(dropTarget);
}

/***
 * @param node
 * @return the drag source for grabbing and dragging the supplied
node to a
 *         new location in the tree.
 */
public DragSource getDragSource(MutableTreeNode node) {
    if (nodeToDragSourceMap.containsKey(node)) {
        return nodeToDragSourceMap.get(node);
    } else {
        Component dragSourceLabel = new Label();
        dragSourceLabel.setStyleName(STYLE_NAME_DRAG_SOURCE);
        DragSource dragSource = new DragSource(dragSourceLabel);
        dragSource.addDropTargetListener(dropHandler);
        nodeToDragSourceMap.put(node, dragSource);
        dragSourceToNodeMap.put(dragSource, node);
        for (Component target : nodeToDropTargetMap.values()) {
            dragSource.addDropTarget(target);
        }
        return dragSource;
    }
}

public MutableTreeNode getNodeForDragSource(DragSource dragSource) {
    return dragSourceToNodeMap.get(dragSource);
}

public void moveTreeNodeToBeforeNode(MutableTreeNode nodeToMove,
MutableTreeNode targetNode) {
    TreeNode parent = targetNode.getParent();
    if ((parent != null) && !isDescendantOf(nodeToMove, targetNode)) {
        getModel().insertNodeInto(nodeToMove, (MutableTreeNode) parent,
```

```

    parent.getIndex(targetNode));
getModel().nodeStructureChanged(parent);
if ((getTreeNode() != null) && (getTreeNode().getChildCount() > 0))
{
    // if the parent is the root keep the tree expanded
    expandPath(new TreePath(getModel().getPathToRoot(getTreeNode())));
}
}

private boolean isDescendantOf(TreeNode node1, TreeNode node2) {
boolean result = false;
if (node1.equals(node2)) {
    result = true;
} else if (node2.getParent() != null) {
    result = isDescendantOf(node1, node2.getParent());
}
return result;
}

/**
 */
private void buildTree() {
nodeToDropTargetMap.clear();
nodeToDragSourceMap.clear();
dropTargetToNodeMap.clear();
dragSourceToNodeMap.clear();
TreeNode rootNode;
if (getRootObject() != null) {
    EditorTreeNodeFactory factory =
getEditorTreeNodeFactory(getRootObject());
    if (factory != null) {
        rootNode = factory.createTreeNode(eventDispatcher, this,
getRootObject());
    } else {
        rootNode = new DefaultMutableTreeNode(getRootObject());
    }
} else {
    rootNode = new DefaultMutableTreeNode();
}
setModel(new DefaultTreeModel(rootNode));
expandAll();
}

@Override
public void dispose() {
    setModel(null);
    setRootObject(null);
    super.dispose();
}

private void removeListeners(DefaultTreeModel treeModel) {
    if (treeModel != null) {
        MutableTreeNode rootNode = (MutableTreeNode) treeModel.getRoot();
        if (rootNode != null) {
            Enumeration<MutableTreeNode> enm = new
PostorderEnumeration(rootNode);
            while (enm.hasMoreElements()) {
                MutableTreeNode node = enm.nextElement();
                if (node instanceof EditorTreeNode) {
                    ((EditorTreeNode) node).dispose();
                }
            }
        }
    }
}

/**
 * @return
 */
public Object getRootObject() {
    return rootObject;
}

/**
 * @return the root node in the tree.
 */
public MutableTreeNode getRootNode() {
    return (MutableTreeNode) getModel().getRoot();
}

/**
 * @param rootObject
 */
public void setRootObject(Object rootObject) {
    // this causes invalidate() to be called.
    setModel(new DefaultTreeModel(new
DefaultMutableTreeNode(rootObject)));
    this.rootObject = rootObject;
    if (this.rootObject != null) {
        buildTree();
    }
}
}

```

```

@Override
public DefaultTreeModel getModel() {
    return (DefaultTreeModel) super.getModel();
}

@Override
public void setModel(TreeModel newTreeModel) {
    DefaultTreeModel oldModel = getModel();
    if (oldModel != null) {
        removeListeners(oldModel);
    }
    super.setModel(newTreeModel);
}

/**
 * @param target
 * @return An EditorTreeNodeFactory for creating a node to edit the
supplied
 *         target object.
 */
public EditorTreeNodeFactory getEditorTreeNodeFactory(Object target)
{
    Class<?> targetType = ((target instanceof Class<?>) ? (Class<?>)
target : target.getClass());
    EditorTreeNodeFactory factory = null;
    factory = treeNodeFactoryMap.get(target);
    if (factory != null) {
        return factory;
    }
    if (targetType.getInterfaces() != null) {
        for (Class<?> targetInterface : targetType.getInterfaces()) {
            factory = getEditorTreeNodeFactory(targetInterface);
            if (factory != null) {
                return factory;
            }
        }
    }
    if ((targetType.getSuperclass() != null)
        && !Object.class.equals(targetType.getSuperclass())) {
        factory = getEditorTreeNodeFactory(targetType.getSuperclass());
    }
    return factory;
}

private static class EditorTreeDragAndDropHandler implements
DropListener {

```

```

static final long serialVersionUID = 0L;
private final EditorTree editorTree;

/**
 * @param editorTree
 */
public EditorTreeDragAndDropHandler(EditorTree editorTree) {
    super();
    this.editorTree = editorTree;
}

/**
 * @see
nextapp.echo2.extras.app.event.DropListener#dropPerformed(nextapp.echo
2.extras.app.event.DropEvent)
 */
@Override
public void dropPerformed(DropEvent event) {
    DragSource dragSource = (DragSource) event.getSource();
    Component dropTarget = (Component) event.getTarget();
    MutableTreeNode nodeToMove =
editorTree.getNodeForDragSource(dragSource);
    MutableTreeNode targetNode =
editorTree.getNodeForDropTarget(dropTarget);
    editorTree.moveTreeNodeToBeforeNode(nodeToMove, targetNode);
}

// the following are copied from DefaultMutableTreeNode

public final class PostorderEnumeration implements Enumeration,
Serializable {
    protected TreeNode root;
    protected Enumeration children;
    protected Enumeration subtree;

    public PostorderEnumeration(TreeNode rootNode) {
        super();
        root = rootNode;
        children = root.children();
        subtree = EMPTY_ENUMERATION;
    }

    public boolean hasMoreElements() {
        return root != null;
    }
}

```

```

public Object nextElement() {
    Object retval;

    if (subtree.hasMoreElements()) {
        retval = subtree.nextElement();
    } else if (children.hasMoreElements()) {
        subtree = new PostorderEnumeration((TreeNode)
children.nextElement());
        retval = subtree.nextElement();
    } else {
        retval = root;
        root = null;
    }

    return retval;
}

public static final Enumeration EMPTY_ENUMERATION = new Enumeration()
{
    public boolean hasMoreElements() {
        return false;
    }

    public Object nextElement() {
        throw new NoSuchElementException("No more elements");
    }
};
}

```

editortreecellrenderer.java

```

/*
 * $Id: EditorTreeCellRenderer.java,v 1.5 2008/10/29 07:42:41 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import java.util.Map;
import java.util.WeakHashMap;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;

```

```

import nextapp.echo2.app.Row;
import echopointng.Tree;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.TreeCellRenderer;
import echopointng.treeTreeNode;
import echopointng.xhtml.XhtmlFragment;

/**
 * @author ron
 */
public class EditorTreeCellRenderer implements TreeCellRenderer {
    static final long serialVersionUID = 0L;

    private final Map<Object, Row> wrapperMap = new WeakHashMap<Object,
Row>();

    /**
     * @param enableDragAndDrop
     */
    public EditorTreeCellRenderer() {
    }

    @Override
    public Component getTreeCellRendererComponent(Tree tree, Object node,
boolean selected,
        boolean expanded, boolean leaf) {
        if (node instanceof EditorTreeNodeAbstractDecorator) {
            EditorTreeNodeAbstractDecorator decorator =
(EditorTreeNodeAbstractDecorator) node;
            return decorator.getRenderComponent();
        }
        return getComponent((TreeNode) node);
    }

    @Override
    public Label getTreeCellRendererText(Tree tree, Object node, boolean
sel, boolean expanded,
        boolean leaf) {
        return null;
    }

    @Override
    public XhtmlFragment getTreeCellRendererXhtml(Tree tree, Object
value, boolean selected,
        boolean expanded, boolean leaf) {
        return null;
    }
}

```

```

private Component getComponent(TreeNode node) {
    Object value;
    if (node instanceof EditorTreeNode) {
        value = ((EditorTreeNode) node).getEditor();
    } else if (node instanceof DefaultMutableTreeNode) {
        value = ((DefaultMutableTreeNode) node).getUserObject();
    } else {
        value = null;
    }
    if (value != null) {
        if (value instanceof Component) {
            return (Component) value;
        } else {
            return new Label(value.toString());
        }
    }
    return new Label(node.toString());
}

```

editortreedraganddropnodefactorydecorator.java

```

/*
 * $Id: EditorTreeDragAndDropNodeFactoryDecorator.java,v 1.2
2008/10/29 07:42:41 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import echopointng.tree.MutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public class EditorTreeDragAndDropNodeFactoryDecorator implements
EditorTreeNodeFactory {

    private final EditorTreeNodeFactory decoratedFactory;

    /**
     * @param decoratedFactory
     */

```

```

    public
EditorTreeDragAndDropNodeFactoryDecorator(EditorTreeNodeFactory
decoratedFactory) {
    super();
    this.decoratedFactory = decoratedFactory;
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFac
tory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.even
t.EventDispatcher,
*
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree,
*
java.lang.Object)
*/
@Override
public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, EditorTree tree,
Object object) {
    return new
EditorTreeNodeDragAndDropDecorator(decoratedFactory.createTreeNode(
    eventDispatcher, tree, object), tree);
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFac
tory#getTargetType()
*/
@Override
public Class<?> getTargetType() {
    return decoratedFactory.getTargetType();
}
}

```

editortreemanipulator.java

```

/*
 * $Id: EditorTreeManipulator.java,v 1.2 2008/10/16 01:00:30 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

```

```

import java.util.ArrayList;
import java.util.Comparator;
import java.util.Enumeration;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import nextapp.echo2.app.Component;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.DefaultTreeModel;
import echopointng.tree.TreeModelEvent;
import echopointng.tree.TreeModelListener;
import echopointng.treeTreeNode;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * @author ron
 */
public class EditorTreeManipulator extends
AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        if (Map.class.equals(type)) {
            Map<List<Object>, Object> values = new TreeMap<List<Object>,
Object>(new NodeComparator());
            DefaultMutableTreeNode rootNode = (DefaultMutableTreeNode)
getModel(component)
                .getRoot();
            if (rootNode != null) {
                Enumeration<DefaultMutableTreeNode> enm =
rootNode.depthFirstEnumeration();
                while (enm.hasMoreElements()) {
                    DefaultMutableTreeNode node = enm.nextElement();
                    if (node instanceof EditorTreeNode) {
                        EditorTreeNode editorNode = (EditorTreeNode) node;
                        Component editor = editorNode.getEditor();
                        ComponentManipulator man =
ComponentManipulators.getManipulator(editor);

```

```

                        if (man != null) {
                            values.put(getPathEntites(editorNode), man.getValue(editor,
Object.class));
                        }
                    }
                }
            }
            return type.cast(values);
        }
        return type.cast(getModel(component).getRoot());
    }

    private List<Object> getPathEntites(EditorTreeNode editorNode) {
        List<Object> path = new ArrayList<Object>(editorNode.getDepth() +
1);
        for (TreeNode ancestorNode : editorNode.getPath()) {
            if (ancestorNode instanceof EditorTreeNode) {
                Component editor = ((EditorTreeNode)ancestorNode).getEditor();
                ComponentManipulator man =
ComponentManipulators.getManipulator(editor);
                if (man != null) {
                    path.add(man.getValue(editor, Object.class));
                }
            }
        }
        return path;
    }

    public void setValue(Component component, Object value) {
        getComponent(component).setRootObject(value);
    }

    public void addListenerToDetectChangesToInput(final EditMode
editMode, Component component) {
        getModel(component).addTreeModelListener(new TreeModelListener() {
            static final long serialVersionUID = 0L;

            @Override
            public void treeNodesChanged(TreeModelEvent e) {
                editMode.setStateEdited(true);
            }

            @Override
            public void treeNodesInserted(TreeModelEvent e) {
                editMode.setStateEdited(true);
            }
        });
    }
}

```

```

@Override
public void treeNodesRemoved(TreeModelEvent e) {
    editMode.setStateEdited(true);
}

@Override
public void treeStructureChanged(TreeModelEvent e) {
    editMode.setStateEdited(true);
}

};

@Override
public DefaultTreeModel getModel(Component component) {
    return getComponent(component).getModel();
}

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setModel((DefaultTreeModel) valueModel);
}

private EditorTree getComponent(Component component) {
    return (EditorTree) component;
}

private static class NodeComparator implements
Comparator<List<Object>> {

@Override
public int compare(List<Object> o1, List<Object> o2) {
    if (o1.size() != o2.size()) {
        return (o1.size() - o2.size());
    }

    for (int index = 0; index < o1.size(); index++) {
        int hash = (o1.get(index).hashCode() - o2.get(index).hashCode());
        if (hash != 0) {
            return hash;
        }
    }
    return 0;
}
}

```

editortreenode.java

```

/*
 * $Id: EditorTreeNode.java,v 1.4 2008/10/28 07:13:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ActionListener;
import echopointng.tree.MutableTreeNode;
import echopointng.tree.TreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * A node in an editor tree that contains an editor component for
editing the
 * content of the node.
 *
 * @author ron
 */
public interface EditorTreeNode extends MutableTreeNode,
ActionListener {

/**
 * @return The editor component of this node.
 */
public Component getEditor();

public void setEditor(Component editor);

public EventDispatcher getEventDispatcher();

/**
 * Set a listener that gets notified if the object for the node is
modified.
 * The new listener will be registered with the event dispatcher by
this
 * method. If another listener is already set, it will be
unregistered
 * before the new one is registered. If null is supplied, the old
listener
 * will be unregistered and no listeners will be listening for this
node.
 *

```

```

 * @param updateListener
 */
public void setUpdateListener(EditorTreeNodeUpdateListener
updateListener);

public void dispose();

// copied from DefaultMutableTreeeNode because they aren't in
// MutableTreeNode but should be

public int getDepth();

public TreeNode[] getPath();

public void add(MutableTreeNode newChild);
}

```

editortreenodeabstractdecorator.java

```

/*
 * $Id: EditorTreeNodeAbstractDecorator.java,v 1.3 2009/01/15 01:33:44
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import java.util.ArrayList;
import java.util.Enumeration;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ActionEvent;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.MutableTreeNode;
import echopointng.tree.TreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public abstract class EditorTreeNodeAbstractDecorator implements
EditorTreeNode {
    static final long serialVersionUID = 0L;
    private final MutableTreeNode decoratedNode;

```

```

 /**
 * @param treeNode
 * @param tree
 */
public EditorTreeNodeAbstractDecorator(MutableTreeNode treeNode,
EditorTree tree) {
    this.decoratedNode = treeNode;
}

@Override
public void insert(MutableTreeNode child, int index) {
    this.children();
    decoratedNode.insert(child, index);
}

@Override
public void remove(int index) {
    decoratedNode.remove(index);
}

@Override
public void remove(MutableTreeNode node) {
    decoratedNode.remove(node);
}

@Override
public void removeFromParent() {
    MutableTreeNode parent = (MutableTreeNode) getParent();
    if (parent != null) {
        parent.remove(this);
    }
}

@Override
public void setParent(MutableTreeNode newParent) {
    decoratedNode.setParent(newParent);
}

// this should be on the MutableTreeNode interface, but its only on
the
// implementation class DefaultMutableTreeNode
public Object getUserObject() {
    if (decoratedNode instanceof DefaultMutableTreeNode) {
        return ((DefaultMutableTreeNode) decoratedNode).getUserObject();
    }
    throw new RuntimeException("stupid tree implementation.");
}

```

```

@Override
public void setUserObject(Object object) {
    decoratedNode.setUserObject(object);
}

@Override
public Enumeration<Object> children() {
    return decoratedNode.children();
}

@Override
public String getActionCommand() {
    return decoratedNode.getActionCommand();
}

@Override
public boolean getAllowsChildren() {
    return decoratedNode.getAllowsChildren();
}

@Override
public TreeNode getChildAt(int childIndex) {
    return decoratedNode.getChildAt(childIndex);
}

@Override
public int getChildCount() {
    return decoratedNode.getChildCount();
}

@Override
public int getIndex(TreeNode node) {
    return decoratedNode.getIndex(node);
}

@Override
public TreeNode getParent() {
    return decoratedNode.getParent();
}

@Override
public boolean isLeaf() {
    return decoratedNode.isLeaf();
}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (decoratedNode instanceof EditorTreeNode) {
        ((EditorTreeNode) decoratedNode).actionPerformed(e);
    }
}

@Override
public void dispose() {
    if (decoratedNode instanceof EditorTreeNode) {
        ((EditorTreeNode) decoratedNode).dispose();
    }
}

@Override
public int getDepth() {
    if (decoratedNode instanceof EditorTreeNode) {
        return ((EditorTreeNode) decoratedNode).getDepth();
    } else if (decoratedNode instanceof DefaultMutableTreeNode) {
        return ((DefaultMutableTreeNode) decoratedNode).getDepth();
    } else if (decoratedNode.getChildCount() == 0) {
        return 0;
    } else {
        throw new RuntimeException("getDepth isn't implemented for general
TreeNode.");
    }
}

@Override
public Component getEditor() {
    if (decoratedNode instanceof EditorTreeNode) {
        return ((EditorTreeNode) decoratedNode).getEditor();
    }
    return null;
}

@Override
public EventDispatcher getEventDispatcher() {
    if (decoratedNode instanceof EditorTreeNode) {
        return ((EditorTreeNode) decoratedNode).getEventDispatcher();
    }
    return null;
}

@Override
public TreeNode[] getPath() {

```

```

if (decoratedNode instanceof EditorTreeNode) {
    return ((EditorTreeNode) decoratedNode).getPath();
} else if (decoratedNode instanceof DefaultMutableTreeNode) {
    return ((DefaultMutableTreeNode) decoratedNode).getPath();
} else {
    ArrayList<TreeNode> nodeList = new ArrayList<TreeNode>(10);
    TreeNode parent = decoratedNode;
    while (parent != null) {
        nodeList.add(parent);
        parent = parent.getParent();
    }
    ArrayList<TreeNode> reverseNodes = new
    ArrayList<TreeNode>(nodeList.size());
    for (int i = nodeList.size() - 1; i >= 0; i--) {
        reverseNodes.add(nodeList.get(i));
    }
    return reverseNodes.toArray(new TreeNode[nodeList.size()]);
}
}

@Override
public void add(MutableTreeNode newChild) {
    if (decoratedNode instanceof EditorTreeNode) {
        ((EditorTreeNode) decoratedNode).add(newChild);
    } else if (decoratedNode instanceof DefaultMutableTreeNode) {
        ((DefaultMutableTreeNode) decoratedNode).add(newChild);
    } else {
        throw new RuntimeException("The decorated node " + decoratedNode
            + " doesn't support add()");
    }
}

@Override
public void setEditor(Component editor) {
    if (decoratedNode instanceof EditorTreeNode) {
        ((EditorTreeNode) decoratedNode).setEditor(editor);
    }
}

@Override
public void setUpdateListener(EditorTreeNodeUpdateListener
updateListener) {
    if (decoratedNode instanceof EditorTreeNode) {
        ((EditorTreeNode) decoratedNode).setUpdateListener(updateListener);
    }
}

```

```

// the tree object model depends on the implementation and not the
// interfaces so the decorated node must be accessed and passed to
// tree methods.
public MutableTreeNode getDecoratedNode() {
    return decoratedNode;
}

/**
 * @return The node with decorations
 */
public Component getRenderComponent() {
    if (decoratedNode instanceof EditorTreeNodeAbstractDecorator) {
        return ((EditorTreeNodeAbstractDecorator)
decoratedNode).getRenderComponent();
    }
    return getEditor();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    // get the root decorated node of this and the obj supplied and
    compare them:
    while (obj instanceof EditorTreeNodeAbstractDecorator) {
        obj = ((EditorTreeNodeAbstractDecorator)obj).getDecoratedNode();
    }
    Object thisNode = getDecoratedNode();
    while (thisNode instanceof EditorTreeNodeAbstractDecorator) {
        thisNode =
((EditorTreeNodeAbstractDecorator)thisNode).getDecoratedNode();
    }
    return thisNode.equals(obj);
}

@Override
public int hashCode() {
    return decoratedNode.hashCode();
}

```

editortreenodeactionbuttondecorator.java

```
/*
```

```

* $Id: EditorTreeNodeActionButtonDecorator.java,v 1.4 2009/01/22
10:36:30 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.layout.RowLayoutData;
import echopointng.tree.MutableTreeNode;

/**
 * A decorator for adding a button to a tree node that invokes some
action.
 *
 * @author ron
 */
public class EditorTreeNodeActionButtonDecorator extends
EditorTreeNodeAbstractDecorator implements
EditorTreeNode {
static final long serialVersionUID = 0L;

private final Button button;
private final RowLayoutData rowLayoutTop = new RowLayoutData();

/**
 * @param treeNode
 * @param tree
 * @param button
 */
public EditorTreeNodeActionButtonDecorator(MutableTreeNode treeNode,
EditorTree tree,
Button button) {
super(treeNode, tree);
this.button = button;
rowLayoutTop.setAlignment(Alignment.ALIGN_TOP);
}

@Override
public void dispose() {
super.dispose();
button.dispose();
}

```

```

@Override
public Component getRenderComponent() {
Component editor = super.getRenderComponent();
Row wrapper = new Row();
// button.setLayoutData(rowLayoutTop);
// editor.setLayoutData(rowLayoutTop);
// TODO: add a side/alignment property to indicate which side the
button
// goes on.
wrapper.add(button);
wrapper.add(editor);
return wrapper;
}
}

```

editortreenodedraganddropdecorator.java

```

/*
* $Id: EditorTreeNodeDragAndDropDecorator.java,v 1.8 2009/01/23
09:54:25 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Row;
import nextapp.echo2.extras.app.DragSource;
import echopointng.tree.MutableTreeNode;

/**
 * A decorator for adding dragging and dropping between tree nodes in
the
 * editor.
 *
 * @author ron
 */
public class EditorTreeNodeDragAndDropDecorator extends
EditorTreeNodeAbstractDecorator implements
EditorTreeNode {
static final long serialVersionUID = 0L;

private final Component dropTarget;
private final DragSource dragSource;

```

```

/**
 * @param treeNode
 * @param tree
 */
public EditorTreeNodeDragAndDropDecorator(MutableTreeNode treeNode,
EditorTree tree) {
    super(treeNode, tree);
    this.dragSource = tree.getDragSource(this);
    this.dropTarget = tree.getDropTarget(this);
}

/**
 * @return The target component for dropping a drag source for this
tree
 *         node.
 */
public Component getDropTarget() {
    return dropTarget;
}

/**
 * @return The draggable component for dropping on a target
representing this
 *         tree node.
 */
public DragSource getDragSource() {
    return dragSource;
}

@Override
public void dispose() {
    super.dispose();
    dragSource.dispose();
    dropTarget.dispose();
}

@Override
public Component getRenderComponent() {
    Component editor = super.getRenderComponent();
    Row wrapper = new Row();
    // arrange the drag and drop components vertically
    // to the left of the editor.
    Column vertWrapper = new Column();
    vertWrapper.setCellSpacing(new Extent(0));
    vertWrapper.add(getDragSource());
    vertWrapper.add(getDropTarget());
}

```

```

        wrapper.add(vertWrapper);
        wrapper.add(editor);
        return wrapper;
    }
}

```

editortreenodefactory.java

```

/*
 * $Id: EditorTreeNodeFactory.java,v 1.2 2008/10/29 07:42:41 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import echopointng.tree.MutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public interface EditorTreeNodeFactory {

    /**
     * Return the type of entity object this factory builds TreeNodes
for.
     *
     * @return
     */
    public Class<?> getTargetType();

    /**
     * Given some object create an appropriate tree representation and
return
     * the root of that tree.
     *
     * @param eventDispatcher
     * @param tree
     * @param object
     * @return
     */
    public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, EditorTree tree,
        Object object);
}

```

editortreenodeupdatelistener.java

```
/*
 * $Id: EditorTreeNodeUpdateListener.java,v 1.1 2008/10/15 09:20:05
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.tree;

import nextapp.echo2.app.event.ActionListener;

/**
 * @author ron
 */
public interface EditorTreeNodeUpdateListener extends ActionListener {
```

editpositioncommand.java

```
/*
 * $Id: EditPositionCommand.java,v 1.2 2008/05/22 09:26:57 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * Create or edit a position for an issue.
 *
 * @author ron
 */
public interface EditPositionCommand extends EditCommand {

    /**
     * Set the issue for a new position to be attached to.
     *
     * @param issue
     */
    public void setIssue(Issue issue);

    /**

```

```
     * Set the position to edit.
     *
     * @param position
     */
    public void setPosition(Position position);

    /**
     * Get the new or updated position.
     *
     * @return
     */
    public Position getPosition();

    /**
     * Set the text for the position.
     *
     * @param text
     */
    public void setText(String text);

}
```

editpositioncommandimpl.java

```
/*
 * $Id: EditPositionCommandImpl.java,v 1.14 2009/02/17 11:50:47 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
```

```
import edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("editPositionCommand")
@Scope("prototype")
public class EditPositionCommandImpl extends AbstractEditCommand
implements EditPositionCommand {

    private Position position;
    private Issue issue;
    private String text;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public EditPositionCommandImpl(CommandHandler commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        AnnotationRepository repository) {
        super(commandHandler, annotationCommandFactory, repository);
    }

    /**
     * @see edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand#setIssue(edu.harvard.fas.rregan.requel.annotation.Issue)
     */
    @Override
    public void setIssue(Issue issue) {
        this.issue = issue;
    }

    protected Issue getIssue() {
        return issue;
    }

    /**
     * @see edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand#getPosition()
     */
```

```
 */
@Override
public Position getPosition() {
    return position;
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand#s
etPosition(edu.harvard.fas.rregan.requel.annotation.Position)
 */
@Override
public void setPosition(Position position) {
    this.position = position;
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand#s
etText(java.lang.String)
 */
@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    validate();
    User editedBy = getRepository().get(getEditedBy());
    Issue issue = getRepository().get(getIssue());
    PositionImpl position = (PositionImpl) getPosition();
    if (position == null) {
        // look for existing position that matches the text and
        // reference it with the issue.
        if (issue != null) {
            try {
                getAnnotationRepository().findPosition(issue.getGroupingObject(),
getText());
            } catch (NoSuchEntityException e) {

```

```

        position = getRepository().persist(new PositionImpl(getText(),
editedBy));
    }
} else {
    position = getRepository().persist(new PositionImpl(getText(),
editedBy));
}
else {
    position.setText(getText());
    position = getRepository().merge(position);
}
if (issue != null) {
    position.getIssues().add(issue);
}
setPosition(position);

// add the position to the issue after it has been merged so that if
it
// is a proxy it will be unwrapped by the framework.
if (issue != null) {
    issue.getPositions().add(position);
    setIssue(issue);
}
}

protected void validate() {
    if ((getText() == null) || "".equals(getText().trim())) {
        throw
EntityValidationException.emptyRequiredProperty(Position.class,
getPosition(),
        "text", EntityExceptionActionType.Updating);
    }
}
}

```

editprojectcommand.java

```

/*
 * $Id: EditProjectCommand.java,v 1.7 2009/02/13 12:08:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.AnalyzableEditCommand;
import edu.harvard.fas.rregan.requel.project.Project;

```

```

/**
 * @author ron
 */
public interface EditProjectCommand extends AnalyzableEditCommand {

/**
 * The name of the "name" field used to correlate to the field in an
editor
 * and through exceptions.
 */
public static final String FIELD_NAME = "name";

/**
 * @param name
 */
public void setName(String name);

/**
 * @param description
 */
public void setText(String description);

/**
 * @param organizationName
 */
public void setOrganizationName(String organizationName);

/**
 * @param project
 */
public void setProject(Project project);

/**
 * @return
 */
public Project getProject();
}

```

editprojectcommandimpl.java

```

/*
 * $Id: EditProjectCommandImpl.java,v 1.23 2009/03/30 11:54:29 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;
```

```

import java.io.InputStream;

import org.apache.commons.io.IOUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.command.EditProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditReportGeneratorComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ProjectImpl;
import edu.harvard.fas.rregan.requel.project.impl.StakeholderImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchOrganizationExcepti
on;
import edu.harvard.fas.rregan.requel.user.impl.OrganizationImpl;

/**
 * @author ron
 */
@Controller("editProjectCommand")
@Scope("prototype")
public class EditProjectCommandImpl extends AbstractEditProjectCommand
implements
EditProjectCommand {

```

```

    public static final String BUILTIN_REPORT_GENERATOR_PATH =
"resources/xslt/project2html.xslt";

    private String name;
    private String description;
    private String organizationName;
    private Project project;
    private boolean analysisEnabled = true;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public EditProjectCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
    ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
        annotationCommandFactory, commandHandler);
}

    @Override
    public Project getProject() {
        return project;
    }

    public void setProject(Project project) {
        this.project = project;
    }

    protected String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    protected String getText() {

```

```

    return description;
}

public void setText(String description) {
    this.description = description;
}

protected String getOrganizationName() {
    return organizationName;
}

public void setOrganizationName(String organizationName) {
    this.organizationName = organizationName;
}

@Override
public void execute() {
    Organization organization = null;
    try {
        organization =
getUserRepository().findOrganizationByName(getOrganizationName());
    } catch (NoSuchOrganizationException e) {
        organization = getUserRepository().persist(new
OrganizationImpl(getOrganizationName()));
    }
    User user = getUserRepository().get(getEditedBy());
    ProjectImpl projectImpl = (ProjectImpl) getProject();

    // check for uniqueness
    try {
        Project existing =
getProjectRepository().findProjectByName(getName());
        if (projectImpl == null) {
            throw EntityException.uniquenessConflict(Project.class, existing,
FIELD_NAME,
                EntityExceptionActionType.Creating);
        } else if (!existing.equals(projectImpl)) {
            throw EntityException.uniquenessConflict(Project.class, existing,
FIELD_NAME,
                EntityExceptionActionType.Updating);
        }
    } catch (NoSuchEntityException e) {

        if (projectImpl == null) {
            projectImpl = createProject(organization, user);
        } else {

```

```

            projectImpl.setName(getName());
            projectImpl.setOrganization(organization);
        }
        projectImpl.setText(getText());
        projectImpl = getRepository().merge(projectImpl);
        setProject(projectImpl);
    }

    @Override
    public void setAnalysisEnabled(boolean analysisEnabled) {
        this.analysisEnabled = analysisEnabled;
    }

    protected boolean isAnalysisEnabled() {
        return analysisEnabled;
    }

    @Override
    public void invokeAnalysis() {
        // TODO: does the project need to be analyzed?
    }

    private ProjectImpl createProject(Organization organization, User
user) {
        ProjectImpl projectImpl = getProjectRepository().persist(
            new ProjectImpl(getName(), user, organization));

        StakeholderImpl creatorStakeholder = getProjectRepository().persist(
            new StakeholderImpl(projectImpl, user, user));
        for (StakeholderPermission permission : getProjectRepository()
            .findAvailableStakeholderPermissions()) {
            creatorStakeholder.grantStakeholderPermission(permission);
        }

        User assistantUser =
getUserRepository().findUserByUsername("assistant");
        getProjectRepository().persist(new StakeholderImpl(projectImpl,
user, assistantUser));

        ProjectUserRole role = user.getRoleForType(ProjectUserRole.class);
        role.getActiveProjects().add(projectImpl);

        addBuiltInReportGenerator(projectImpl, user);

        return projectImpl;
    }
}

```

```

private void addBuiltinReportGenerator(Project project, User user) {
    try {
        InputStream inputStream =
getClass().getClassLoader().getResourceAsStream(
        BUILTIN_REPORT_GENERATOR_PATH);
        EditReportGeneratorCommand command = getProjectCommandFactory()
            .newEditReportGeneratorCommand();
        command.setEditedBy(user);
        command.setProjectOrDomain(project);
        command.setName("HTML Specification");
        command.setText(IOUtils.toString(inputStream));
        getCommandHandler().execute(command);
    } catch (Exception e) {
        log.error("The builtin report generator could not be added to " +
project, e);
    }
}

```

editprojectordomainentitycommand.java

```

/*
 * $Id: EditProjectOrDomainEntityCommand.java,v 1.8 2009/02/13
12:08:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.AnalyzableEditCommand;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * @author ron
 */
public interface EditProjectOrDomainEntityCommand extends
AnalyzableEditCommand {

/**
 * The name of the "name" field used to correlate to the field in an
editor
 * and through exceptions.
 */
public static final String FIELD_NAME = "name";

/**

```

```

 * Set the project or domain this entity is a part of.
 *
 * @param projectOrDomain
 */
public void setProjectOrDomain(ProjectOrDomain projectOrDomain);

/**
 * @param name
 */
public void setName(String name);
}

```

editreportgeneratorcommand.java

```

/*
 * $Id: EditReportGeneratorCommand.java,v 1.3 2009/01/27 09:30:17
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.project.ReportGenerator;

/**
 * @author ron
 */
public interface EditReportGeneratorCommand extends
EditTextEntityCommand {

/**
 * Set the ReportGenerator to edit.
 *
 * @param ReportGenerator
 */
public void setReportGenerator(ReportGenerator ReportGenerator);

/**
 * Get the new or updated ReportGenerator.
 *
 * @return
 */
public ReportGenerator getReportGenerator();
}

```

editreportgeneratorcommandimpl.java

```
/*
 * $Id: EditReportGeneratorCommandImpl.java,v 1.7 2009/03/30 11:54:30
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import
edu.harvard.fas.rregan.requel.project.command.EditReportGeneratorComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ReportGeneratorImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editReportGeneratorCommand")
@Scope("prototype")
public class EditReportGeneratorCommandImpl extends
AbstractEditProjectOrDomainEntityCommand
implements EditReportGeneratorCommand {

    private ReportGenerator reportGenerator;
    private String text;
```

```
 /**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@.Autowired
public EditReportGeneratorCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
    projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

public ReportGenerator getReportGenerator() {
    return reportGenerator;
}

public void setReportGenerator(ReportGenerator reportGenerator) {
    this.reportGenerator = reportGenerator;
}

@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

@Override
public void execute() {
    User editedBy = getProjectRepository().get(getEditedBy());
    ReportGeneratorImpl reportGeneratorImpl = (ReportGeneratorImpl)
getReportGenerator();
    ProjectOrDomain projectOrDomain =
getProjectRepository().get(getProjectOrDomain());

    // check for uniqueness
    try {
```

```

ReportGenerator existing = getProjectRepository()
    .findReportGeneratorByProjectOrDomainAndName(projectOrDomain,
getName());
if (reportGeneratorImpl == null) {
    throw EntityException.uniquenessConflict(ReportGenerator.class,
existing,
    FIELD_NAME, EntityExceptionActionType.Creating);
} else if (!existing.equals(reportGeneratorImpl)) {
    throw EntityException.uniquenessConflict(ReportGenerator.class,
existing,
    FIELD_NAME, EntityExceptionActionType.Updating);
}
} catch (NoSuchEntityException e) {

if (reportGeneratorImpl == null) {
    reportGeneratorImpl = getProjectRepository().persist(
        new ReportGeneratorImpl(projectOrDomain, editedBy, getName(),
getText()));
} else {
    reportGeneratorImpl.setName(getName());
    reportGeneratorImpl.setText(getText());
}
reportGeneratorImpl =
getProjectRepository().merge(reportGeneratorImpl);
setReportGenerator(reportGeneratorImpl);
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        // TODO: analyze report generator?
    }
}
}

```

editscenariocommand.java

```

/*
 * $Id: EditScenarioCommand.java,v 1.4 2008/11/06 06:19:49 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

```

```

import java.util.List;

import edu.harvard.fas.rregan.requel.project.Scenario;

/**
 * @author ron
 */
public interface EditScenarioCommand extends EditScenarioStepCommand {

/**
 * Set the Scenario to edit.
 *
 * @param Scenario
 */
public void setScenario(Scenario Scenario);

/**
 * Get the new or updated Scenario.
 *
 * @return
 */
public Scenario getScenario();

/**
 * @param editStepCommands -
 *          a list of step or scenario edit commands corresponding
to the
 *          steps of this scenario to execute and then add the
steps
 */
public void setStepCommands(List<EditScenarioStepCommand>
editStepCommands);

}


```

editscenariocommandimpl.java

```

/*
 * $Id: EditScenarioCommandImpl.java,v 1.11 2009/03/30 11:54:28 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.ArrayList;
import java.util.List;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioStepCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ScenarioImpl;
import
edu.harvard.fas.rregan.requel.project.impl.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editScenarioCommand")
@Scope("prototype")
public class EditScenarioCommandImpl extends
EditScenarioStepCommandImpl implements
EditScenarioCommand {

    private List<EditScenarioStepCommand> editStepCommands;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
}

```

```

    @Autowired
    public EditScenarioCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    @Override
    public Scenario getScenario() {
        return (Scenario) getStep();
    }

    @Override
    public void setScenario(Scenario scenario) {
        setStep(scenario);
    }

    @Override
    public void setStepCommands(List<EditScenarioStepCommand>
editStepCommands) {
        this.editStepCommands = editStepCommands;
    }

    protected List<EditScenarioStepCommand> getStepCommands() {
        return editStepCommands;
    }

    @Override
    public void execute() throws Exception {
        List<EditScenarioStepCommand> stepEditCommands = new
ArrayList<EditScenarioStepCommand>(
            getStepCommands());
        for (int index = 0; index < stepEditCommands.size(); index++) {
            EditScenarioStepCommand stepCommand = stepEditCommands.get(index);
            // disable analysis of the steps as they will get analyzed
            // when the whole scenario is analyzed.
            stepCommand.setAnalysisEnabled(false);
            stepEditCommands.set(index,
getCommandHandler().execute(stepCommand));
        }
        User editedBy = getProjectRepository().get(getEditedBy());
        ScenarioImpl scenarioImpl = (ScenarioImpl) getScenario();
    }
}

```

```

ProjectOrDomain projectOrDomain =
getRepository().get(getProjectOrDomain());

// check for uniqueness
try {
    Scenario existing =
getProjectRepository().findScenarioByProjectOrDomainAndName(
    projectOrDomain, getName());
    if (scenarioImpl == null) {
        throw EntityException.uniquenessConflict(Scenario.class, existing,
FIELD_NAME,
        EntityExceptionActionType.Creating);
    } else if (!existing.equals(scenarioImpl)) {
        throw EntityException.uniquenessConflict(Scenario.class, existing,
FIELD_NAME,
        EntityExceptionActionType.Updating);
    }
} catch (NoSuchEntityException e) {

if (scenarioImpl == null) {
    scenarioImpl = getProjectRepository().persist(
        new ScenarioImpl(projectOrDomain, editedBy, getName(), getText(),
        getScenarioType()));
} else {
    scenarioImpl.setName(getName());
    scenarioImpl.setText(getText());
    scenarioImpl.setType(getScenarioType());
}
// TODO: merge is failing because the use cases in the
// usedByUseCases property are my proxies and hibernate throws an
// IllegalArgumentException: Unknown entity:
// edu.harvard.fas.rregan.requel.project.impl.UseCaseImpl$  

$EnhancerByCGLIB$$b1ba12b8
for (Scenario usingScenario : scenarioImpl.getUsingScenarios()) {
    usingScenario = getRepository().get(usingScenario);
}

scenarioImpl = getProjectRepository().merge(scenarioImpl);
scenarioImpl.getSteps().clear();
for (EditScenarioStepCommand executedCommand : stepEditCommands) {
    Step step = executedCommand.getStep();
    scenarioImpl.getSteps().add(step);
    step.getUsingScenarios().add(scenarioImpl);
}
setScenario(getProjectRepository().merge(scenarioImpl));
}

```

```

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        getAssistantManager().analyzeScenario(getScenario());
    }
}

```

editscenariostepcommand.java

```

/*
 * $Id: EditScenarioStepCommand.java,v 1.3 2009/01/27 09:30:16 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.project.Step;

/**
 * @author ron
 */
public interface EditScenarioStepCommand extends EditTextEntityCommand
{

    /**
     * Set the Scenario Step to edit.
     *
     * @param step
     */
    public void setStep(Step step);

    /**
     * Get the new or updated Scenario Step.
     *
     * @return
     */
    public Step getStep();

    /**
     * @param scenarioTypeName -
     *          the name of the type of Scenario Step.
     */
    public void setScenarioTypeName(String scenarioTypeName);
}

```

```
}
```

editscenariostepcommandimpl.java

```
/*
 * $Id: EditScenarioStepCommandImpl.java,v 1.5 2009/03/30 11:54:30
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioStepCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.StepImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editScenarioStepCommand")
@Scope("prototype")
public class EditScenarioStepCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditScenarioStepCommand {

    private Step step;
    private String text;
    private String scenarioTypeName;
```

```
 /**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@.Autowired
public EditScenarioStepCommandImpl(AssistantFacade assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
    projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

@Override
public Step getStep() {
    return step;
}

@Override
public void setStep(Step step) {
    this.step = step;
}

@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

protected ScenarioType getScenarioType() {
    if (scenarioTypeName != null) {
        return ScenarioType.valueOf(scenarioTypeName);
    }
    return null;
}

@Override
```

```

public void setScenarioTypeName(String scenarioTypeName) {
    this.scenarioTypeName = scenarioTypeName;
}

@Override
public void execute() throws Exception {
    User editedBy = getProjectRepository().get(getEditedBy());
    StepImpl stepImpl = (StepImpl) getStep();
    ProjectOrDomain projectOrDomain =
        getProjectRepository().get(getProjectOrDomain());

    if (stepImpl == null) {
        stepImpl = getProjectRepository()
            .persist(
                new StepImpl(projectOrDomain, editedBy, getName(), getText(),
                    getScenarioType()));
    } else {
        stepImpl.setName(getName());
        stepImpl.setText(getText());
        stepImpl.setType(getScenarioType());
    }
    setStep(getProjectRepository().merge(stepImpl));
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        getAssistantManager().analyzeScenarioStep(getStep());
    }
}
}

```

editsemilinkrefcommand.java

```

/*
 * $Id: EditSemilinkRefCommand.java,v 1.1 2008/12/13 00:40:06 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Semilinkref;

/**

```

```

 * @author ron
 */
public interface EditSemilinkRefCommand extends Command {

    public void setSemilinkRef(Semilinkref semilinkref);

    public Semilinkref getSemilinkref();

    public void setFromSynset(Long synsetid);

    public void setToSynset(Long synsetid);

    public void setLinkDef(Long linkdefid);

    public void setDistance(Integer distance);
}

```

editsemilinkrefcommandimpl.java

```

/*
 * $Id: EditSemilinkRefCommandImpl.java,v 1.1 2008/12/13 00:39:58
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Semilinkref;
import edu.harvard.fas.rregan.nlp.dictionary.SemilinkrefId;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemilinkRefCommand;
import edu.harvard.fas.rregan.request.NoSuchEntityException;

/**
 * @author ron
 */
@Controller("editSemilinkRefCommand")
@Scope("prototype")
public class EditSemilinkRefCommandImpl extends
AbstractDictionaryCommand implements
EditSemilinkRefCommand {

```

```

private Semlinkref semlinkref;
private Long fromSynsetid;
private Long toSynsetid;
private Long linkdefid;
private Integer distance;

/**
 * @param dictionaryRepository
 */
@.Autowired
public EditSemlinkRefCommandImpl(DictionaryRepository
dictionaryRepository) {
    super(dictionaryRepository);
}

/**
 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#se
tSemlinkRef(edu.harvard.fas.rregan.nlp.dictionary.Semlinkref)
 */
@Override
public void setSemlinkRef(Semlinkref semlinkref) {
    this.semlinkref = semlinkref;
}

/**
 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#ge
tSemlinkref()
 */
@Override
public Semlinkref getSemlinkref() {
    return semlinkref;
}

/**
 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#se
tFromSynset(edu.harvard.fas.rregan.requel.dictionary.Synset)
 */
@Override
public void setFromSynset(Long synsetid) {
    this.fromSynsetid = synsetid;
}

/** @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#se
tToSynset(edu.harvard.fas.rregan.requel.dictionary.Synset)
 */
@Override
public void setToSynset(Long synsetid) {
    this.toSynsetid = synsetid;
}

/** @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#se
tLinkDef(edu.harvard.fas.rregan.requel.dictionary.Linkdef)
 */
@Override
public void setLinkDef(Long linkdefid) {
    this.linkdefid = linkdefid;
}

/** @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand#se
tDistance(java.lang.Integer)
 */
@Override
public void setDistance(Integer distance) {
    this.distance = distance;
}

/** @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Semlinkref semlinkref =
getDictionaryRepository().get(getSemlinkref());
    if (semlinkref == null) {
        SemlinkrefId semlinkrefId = new SemlinkrefId(fromSynsetid,
toSynsetid, linkdefid,
distance);
        try {
            semlinkref =
getDictionaryRepository().findSemlinkref(semlinkrefId);
        } catch (NoSuchEntityException e) {
            semlinkref = getDictionaryRepository().persist(new
Semlinkref(semlinkrefId));
        }
    }
}

```

```

    } else {
        // TODO: allow editing existing or thrown an exception?
    }

}

}

```

editsensecommand.java

```

/*
 * $Id: EditSenseCommand.java,v 1.1 2008/12/13 00:40:09 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.Word;

/**
 * @author ron
 */
public interface EditSenseCommand extends Command {

    /**
     * set the sense to edit or null for a new sense.
     *
     * @param sense
     */
    public void setSense(Sense sense);

    /**
     * @return the new or updated sense.
     */
    public Sense getSense();

    /**
     * set the word for the sense.
     *
     * @param word
     */
    public void setWord(Word word);
}

```

```

    /**
     * set the synset (meaning) for the sense.
     *
     * @param synset
     */
    public void setSynset(Synset synset);

    /**
     * set the rank (typically in order of usage) of the sense.
     *
     * @param rank
     */
    public void setRank(Integer rank);

    /**
     * set the number of times the word was found in the sample
     * sentences.
     *
     * @param sampleFrequency
     */
    public void setSampleFrequency(Integer sampleFrequency);

    /**
     * Set the sense key used in WordNet to identify the sense.
     *
     * @param senseKey
     */
    public void setSenseKey(String senseKey);
}

```

editsensecommandimpl.java

```

/*
 * $Id: EditSenseCommandImpl.java,v 1.1 2008/12/13 00:39:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;

```



```

public void setWord(Word word) {
    this.word = word;
}

protected Word getWord() {
    return word;
}

protected Synset getSynset() {
    return synset;
}

protected Integer getRank() {
    return rank;
}

protected Integer getSampleFrequency() {
    return sampleFrequency;
}

protected String getSenseKey() {
    return senseKey;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Sense sense = getDictionaryRepository().get(getSense());
    Word word = getDictionaryRepository().get(getWord());
    Synset synset = getDictionaryRepository().get(getSynset());
    if (sense == null) {
        sense = getDictionaryRepository().persist(new Sense(word, synset));
        word.getSenses().add(sense);
    }
    if (getSampleFrequency() != null) {
        sense.setSampleFrequency(getSampleFrequency());
    }
    if (getRank() != null) {
        sense.setRank(getRank());
    }
    if (getSenseKey() != null) {
        sense.setSenseKey(getSenseKey());
    }
    setSense(sense);
}

```

```

}
```

editstakeholdercommand.java

```

/*
 * $Id: EditStakeholderCommand.java,v 1.7 2009/01/27 09:30:17 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import java.util.Set;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Stakeholder;

/**
 * @author ron
 */
public interface EditStakeholderCommand extends
EditProjectOrDomainEntityCommand {

    /**
     * The name of the "user" field used to correlate to the field in an
     * editor
     * and through exceptions.
     */
    public static final String FIELD_USER = "user";

    /**
     * Set the project or domain this stakeholder is a part of.
     *
     * @param projectOrDomain
     */
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain);

    /**
     * Set the user that this stakeholder represents in the project.
     *
     * domains should not have user stakeholders, only representational
     * stakeholders such as a standards board.
     *
     * @param username
     */
    public void setUsername(String username);
}

```

```

/**
 * For non-user stakeholders set the name.<br>
 * NOTE: if a user is passed through setStakeholderUser, then the
name set
 * here is ignored.
 *
 * @param name
 */
public void setName(String name);

/**
 * Supply a role (by name) for this stakeholder. If the supplied name
does
 * not match an existing role for the project a new role will be
created.
 *
 * @param roleNames
 */
public void setStakeholderPermissions(Set<String> roleNames);

/**
 * Set the name of the team this stakeholder is on.
 *
 * @param teamName
 */
public void setTeamName(String teamName);

/**
 * Used to pass in an existing stakeholder for editing purposes. If
no
 * stakeholder is supplied a new one will be created.
 *
 * @param stakeholder
 */
public void setStakeholder(Stakeholder stakeholder);

/**
 * Get the stakeholder created or edited via the command. If a
stakeholder
 * was supplied via setStakeholder the returned stakeholder will
represent
 * the same entity although it may not be equal (for example the user
or
 * name has changed.)
 *
 * @return

```

```

*/
public Stakeholder getStakeholder();
}

```

editstakeholdercommandimpl.java

```

/*
 * $Id: EditStakeholderCommandImpl.java,v 1.25 2009/03/30 11:54:31
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectTeam;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import edu.harvard.fas.rregan.requel.project.command.EditStakeholderCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ProjectTeamImpl;
import edu.harvard.fas.rregan.requel.project.impl.StakeholderImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron

```

```

/*
@Controller("editStakeholderCommand")
@Scope("prototype")
public class EditStakeholderCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditStakeholderCommand {

    private Stakeholder stakeholder;
    private String username;
    private Set<String> permissionKeys;
    private String teamName;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler
     */
    @Autowired
    public EditStakeholderCommandImpl(AssistantFacade assistantManager,
        UserRepository userRepository, ProjectRepository projectRepository,
        ProjectCommandFactory projectCommandFactory,
        AnnotationCommandFactory annotationCommandFactory, CommandHandler
        commandHandler) {
        super(assistantManager, userRepository, projectRepository,
        projectCommandFactory,
        annotationCommandFactory, commandHandler);
    }

    public Stakeholder getStakeholder() {
        return stakeholder;
    }

    public void setStakeholder(Stakeholder stakeholder) {
        this.stakeholder = stakeholder;
    }

    protected String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}

```

```

public void setStakeholderPermissions(Set<String> permissionKeys) {
    this.permissionKeys = permissionKeys;
}

protected Set<String> getStakeholderPermissions() {
    return permissionKeys;
}

protected String getTeamName() {
    return teamName;
}

public void setTeamName(String teamName) {
    this.teamName = teamName;
}

@Override
public void execute() {
    ProjectOrDomain projectOrDomain =
getProjectRepository().get(getProjectOrDomain());
    User editedBy = getProjectRepository().get(getEditedBy());
    User user = null;
    String username = getUsername();
    if ((username != null) && (username.length() > 0)) {
        user = getUserRepository().findUserByUsername(username);
    }

    StakeholderImpl stakeholderImpl = (StakeholderImpl)
getStakeholder();

    // check for uniqueness
    try {
        Stakeholder existing;
        if (user != null) {
            existing =
getProjectRepository().findStakeholderByProjectOrDomainAndUser(
                projectOrDomain, user);
        } else {
            existing =
getProjectRepository().findStakeholderByProjectOrDomainAndName(
                projectOrDomain, getName());
        }
        if (stakeholderImpl == null) {
            throw EntityException.uniquenessConflict(Stakeholder.class,
existing,
                (user == null ? FIELD_NAME : FIELD_USER),
                EntityExceptionActionType.Creating);
        }
    }
}

```

```

} else if (!existing.equals(stakeholderImpl)) {
    throw EntityException.uniquenessConflict(Stakeholder.class,
existing,
    (user == null ? FIELD_NAME : FIELD_USER),
    EntityExceptionActionType.Updating);
}
} catch (NoSuchEntityException e) {
}

ProjectTeam oldTeam = null;
ProjectTeam newTeam = null;
if (stakeholderImpl != null) {
    oldTeam = getRepository().get(stakeholderImpl.getTeam());
}

String teamName = getTeamName();
if ((teamName != null) && (teamName.length() > 0)) {
    for (ProjectTeam aTeam : projectOrDomain.getTeams()) {
        if (teamName.equalsIgnoreCase(aTeam.getName())) {
            newTeam = aTeam;
            break;
        }
    }
    if (newTeam == null) {
        // TODO: add/use create team command?
        newTeam = getRepository().persist(
            new ProjectTeamImpl(projectOrDomain, editedBy, teamName));
    }
}

if (stakeholderImpl == null) {
    stakeholderImpl = createStakeholder(projectOrDomain, user,
editedBy, getName());
} else {
    stakeholderImpl.setName(getName());
    stakeholderImpl.setUser(user);
}
stakeholderImpl.setTeam(newTeam);
stakeholderImpl = getProjectRepository().merge(stakeholderImpl);

if (oldTeam != null) {
    oldTeam.getMembers().remove(stakeholderImpl);
}
if (newTeam != null) {
    newTeam.getMembers().add(stakeholderImpl);
}

```

```

for (StakeholderPermission permission : getProjectRepository()
    .findAvailableStakeholderPermissions()) {
    if (stakeholderImpl.hasPermission(permission)
        && !
getStakeholderPermissions().contains(permission.getPermissionKey())) {
        stakeholderImpl.revokeStakeholderPermission(permission);
    } else if (!stakeholderImpl.hasPermission(permission)
        &&
getStakeholderPermissions().contains(permission.getPermissionKey())) {
        stakeholderImpl.grantStakeholderPermission(permission);
    }
}

if (user != null) {
    ProjectUserRole projectRole =
user.getRoleForType(ProjectUserRole.class);
    projectRole.getActiveProjects().add((Project) projectOrDomain);
}
setStakeholder(stakeholderImpl);
}

private StakeholderImpl createStakeholder(ProjectOrDomain
projectOrDomain,
    User stakeholderUser, User createdBy, String name) {
    StakeholderImpl stakeholderImpl = null;
    if (stakeholderUser != null) {
        stakeholderImpl = new StakeholderImpl(projectOrDomain, createdBy,
stakeholderUser);
    } else {
        stakeholderImpl = new StakeholderImpl(projectOrDomain, createdBy,
name);
    }
    return getProjectRepository().persist(stakeholderImpl);
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        // TODO: analyze stakeholder?
    }
}

```

editstorycommand.java

```
/*
```

```

* $Id: EditStoryCommand.java,v 1.4 2009/01/27 09:30:17 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;

/**
 * @author ron
 */
public interface EditStoryCommand extends EditTextEntityCommand {

    /**
     * Set the Story to edit.
     *
     * @param Story
     */
    public void setStory(Story Story);

    /**
     * Get the new or updated Story.
     *
     * @return
     */
    public Story getStory();

    /**
     * Set the container this Story is being added to.
     *
     * @param StoryContainer
     */
    public void setStoryContainer(StoryContainer StoryContainer);

    /**
     * @param storyTypeName -
     *          the name of the type of story
     */
    public void setStoryTypeName(String storyTypeName);
}

```

editstorycommandimpl.java

```
/*
```

```

* $Id: EditStoryCommandImpl.java,v 1.11 2009/03/30 11:54:31 rregan
Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.StoryType;
import edu.harvard.fas.rregan.requel.project.command.EditStoryCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.StoryImpl;
import edu.harvard.fas.rregan.requel.project.impl.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editStoryCommand")
@Scope("prototype")
public class EditStoryCommandImpl extends AbstractEditProjectOrDomainEntityCommand implements EditStoryCommand {

    private StoryContainer storyContainer;
    private Story story;
    private String text;
    private String storyTypeName;

    /**
     */

```

```

* @param assistantManager
* @param userRepository
* @param projectRepository
* @param projectCommandFactory
* @param annotationCommandFactory
* @param commandHandler
*/
@.Autowired
public EditStoryCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

public Story getStory() {
    return story;
}

public void setStory(Story story) {
    this.story = story;
}

@Override
public void setStoryContainer(StoryContainer storyContainer) {
    this.storyContainer = storyContainer;
}

protected StoryContainer getStoryContainer() {
    return storyContainer;
}

@Override
public void setText(String text) {
    this.text = text;
}

protected String getText() {
    return text;
}

protected StoryType getStoryType() {
    if (storyTypeName != null) {
        return StoryType.valueOf(storyTypeName);
    }
    return null;
}

@Override
public void setStoryTypeName(String storyTypeName) {
    this.storyTypeName = storyTypeName;
}

@Override
public void execute() {
    StoryContainer storyContainer =
getProjectRepository().get(getStoryContainer());
    User editedBy = getProjectRepository().get(getEditedBy());
    StoryImpl storyImpl = (StoryImpl) getStory();
    ProjectOrDomain projectOrDomain = null;
    if (storyImpl != null) {
        projectOrDomain = storyImpl.getProjectOrDomain();
    } else if (storyContainer != null) {
        if (storyContainer instanceof ProjectOrDomain) {
            projectOrDomain = (ProjectOrDomain) storyContainer;
        } else if (storyContainer instanceof ProjectOrDomainEntity) {
            projectOrDomain = ((ProjectOrDomainEntity)
storyContainer).getProjectOrDomain();
        } else {
            // TODO: error?
        }
    }
    projectOrDomain = getRepository().get(projectOrDomain);

    // check for uniqueness
    try {
        Story existing =
getProjectRepository().findStoryByProjectOrDomainAndName(
            projectOrDomain, getName());
        if (storyImpl == null) {
            throw EntityException.uniquenessConflict(Story.class, existing,
FIELD_NAME,
            EntityExceptionActionType.Creating);
        } else if (!existing.equals(storyImpl)) {
            throw EntityException.uniquenessConflict(Story.class, existing,
FIELD_NAME,
            EntityExceptionActionType.Updating);
        }
    } catch (NoSuchEntityException e) {
    }
}

```

```

if (storyImpl == null) {
    storyImpl = getProjectRepository().persist(
        new StoryImpl(projectOrDomain, editedBy, getName(), getText(),
        getStoryType()));
} else {
    storyImpl.setName(getName());
    storyImpl.setText(getText());
    storyImpl.setStoryType(getStoryType());
}
if (storyContainer != null) {
    storyImpl.getReferers().add(storyContainer);
    storyContainer.getStories().add(storyImpl);
}
setStory(getProjectRepository().merge(storyImpl));
}

@Override
public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        getAssistantManager().analyzeStory(getStory());
    }
}
}

```

editsynsetcommand.java

```

/*
 * $Id: EditSynsetCommand.java,v 1.1 2008/12/13 00:40:10 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.util.Map;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;

/**
 * @author ron
 */
public interface EditSynsetCommand extends Command {
/** 
 * @param synset -

```

```

        *          the synset to build the words for
        */
    public void setSynset(Synset synset);

    /**
     * Set the number of subsumers of this Synset for each relation/link
     * type.
     *
     * @param subsumerCounts
     */
    public void setSubsumerCounts(Map<Long, Integer> subsumerCounts);
}

```

editsynsetcommandimpl.java

```

/*
 * $Id: EditSynsetCommandImpl.java,v 1.1 2008/12/13 00:39:59 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Linkdef;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetCommand;

/**
 * @author ron
 */
@Controller("editSynsetCommand")
@Scope("prototype")
public class EditSynsetCommandImpl extends AbstractDictionaryCommand
implements EditSynsetCommand {

    private Synset synset;
    private Map<Long, Integer> subsumerCounts;
}

```

```

/**
 * @param dictionaryRepository
 */
@.Autowired
public EditSynsetCommandImpl(DictionaryRepository
dictionaryRepository) {
    super(dictionaryRepository);
}

/**
 * @see
edu.harvard.fas.rregan.requell.dictionary.command.EditSynsetCommand#set
SubsumerCounts(Set<Integer>)
 */
@Override
public void setSubsumerCounts(Map<Long, Integer> subsumerCounts) {
    this.subsumerCounts = subsumerCounts;
}

/**
 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetCommand#setSyn
set(edu.harvard.fas.rregan.nlp.dictionary.Synset)
 */
@Override
public void setSynset(Synset synset) {
    this.synset = synset;
}

protected Integer getSubsumerCounts(Integer linkType) {
    if (subsumerCounts != null && subsumerCounts.containsKey(linkType))
    {
        return subsumerCounts.get(linkType);
    }
    return 0;
}

protected Synset getSynset() {
    return synset;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Synset synset = getDictionaryRepository().get(getSynset());
}

```

```

if (synset != null && subsumerCounts != null) {
    for (Long linkTypeId : subsumerCounts.keySet()) {
        Linkdef linkType =
getDictionaryRepository().findLinkDef(linkTypeId);
        if (subsumerCounts.containsKey(linkTypeId)) {
            synset.setSubsumerCount(linkType,
subsumerCounts.get(linkTypeId));
        } else {
            synset.setSubsumerCount(linkType, 0);
        }
    }
}

```

editsynsetdefinitionwordcommand.java

```

/*
 * $Id: EditSynsetDefinitionWordCommand.java,v 1.1 2008/12/13 00:40:07
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;

/**
 * create/edit a SynsetDefinitionWord and update the Synset.
 *
 * @author ron
 */
public interface EditSynsetDefinitionWordCommand extends Command {

    /**
     * @param word -
     *          set an existing definition word to edit, or leave null
     * to
     *          create a new word.
     */
    public void setSynsetDefinitionWord(SynsetDefinitionWord word);
}

```

```

/**
 * @return the new SynsetDefinitionWord
 */
public SynsetDefinitionWord getSynsetDefinitionWord();

/**
 * @param synset -
 *          the synset to build the words for
 */
public void setSynset(Synset synset);

/**
 * @param sense
 */
public void setSense(Sense sense);

/**
 * @param index
 */
public void setIndex(Integer index);

/**
 * @param text
 */
public void setText(String text);

/**
 * @param parseTag
 */
public void setParseTag(ParseTag parseTag);
}

```

editsynsetdefinitionwordcommandimpl.java

```

/*
 * $Id: EditSynsetDefinitionWordCommandImpl.java,v 1.1 2008/12/13
00:39:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.ParseTag;

```

```

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetDefinitionWord
Command;

/**
 * @author ron
 */
@Controller("editSynsetDefinitionWordCommand")
@Scope("prototype")
public class EditSynsetDefinitionWordCommandImpl extends
AbstractDictionaryCommand implements
EditSynsetDefinitionWordCommand {

private SynsetDefinitionWord synsetDefinitionWord;
private Synset synset;
private Sense sense;
private Integer index;
private String text;
private ParseTag parseTag;

/**
 * @param dictionaryRepository
 */
@.Autowired
public EditSynsetDefinitionWordCommandImpl(DictionaryRepository
dictionaryRepository) {
super(dictionaryRepository);
}

protected Synset getSynset() {
return synset;
}

public void setSynset(Synset synset) {
this.synset = synset;
}

protected Sense getSense() {
return sense;
}

public void setSense(Sense sense) {
this.sense = sense;
}

```

```

}

protected Integer getIndex() {
    return index;
}

public void setIndex(Integer index) {
    this.index = index;
}

protected String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

protected ParseTag getParseTag() {
    return parseTag;
}

public void setParseTag(ParseTag parseTag) {
    this.parseTag = parseTag;
}

public SynsetDefinitionWord getSynsetDefinitionWord() {
    return synsetDefinitionWord;
}

public void setSynsetDefinitionWord(SynsetDefinitionWord
synsetDefinitionWord) {
    this.synsetDefinitionWord = synsetDefinitionWord;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    SynsetDefinitionWord synsetDefinitionWord =
        getDictionaryRepository().get(
            getSynsetDefinitionWord());
    Synset synset = getDictionaryRepository().get(getSynset());
    Sense sense = getDictionaryRepository().get(getSense());

    if (synsetDefinitionWord == null) {
        if (sense != null) {
            synsetDefinitionWord = new SynsetDefinitionWord(synset,
                getIndex(), getText(),
                getParseTag(), sense);
        } else {
            synsetDefinitionWord = new SynsetDefinitionWord(synset,
                getIndex(), getText(),
                getParseTag());
        }
        synsetDefinitionWord =
            getDictionaryRepository().persist(synsetDefinitionWord);
    } else {
        if (!synsetDefinitionWord.getSynset().equals(synset)) {
            synsetDefinitionWord.getSynset().getWords().remove(synsetDefinitionWord);
            synsetDefinitionWord.setSynset(synset);
        }
        synsetDefinitionWord.setIndex(getIndex());
        synsetDefinitionWord.setSense(sense);
        synsetDefinitionWord.setText(getText());
        synsetDefinitionWord.setParseTag(getParseTag());
    }

    if (getIndex() < synset.getWords().size()) {
        synset.getWords().add(getIndex(), synsetDefinitionWord);
    } else {
        synset.getWords().add(synsetDefinitionWord);
    }
    setSynsetDefinitionWord(synsetDefinitionWord);
}
}

edittextentitycommand.java

/*
 * $Id: EditTextEntityCommand.java,v 1.1 2009/01/27 09:30:16 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

/**
 * @author ron
 */

```

```

public interface EditTextEntityCommand extends
EditProjectOrDomainEntityCommand {

/**
 * The name of the "text" field used to correlate to the field in an
editor
 * and through exceptions.
 */
public static final String FIELD_TEXT = "text";

/**
 * Set the text for the entity.
 *
 * @param text
 */
public void setText(String text);
}

```

editusecasecommand.java

```

/*
 * $Id: EditUseCaseCommand.java,v 1.5 2009/01/27 09:30:17 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import java.util.List;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.UseCase;

/**
 * @author ron
 */
public interface EditUseCaseCommand extends EditTextEntityCommand {

/**
 * Set the project or domain this usecase is a part of.
 *
 * @param projectOrDomain
 */
public void setProjectOrDomain(ProjectOrDomain projectOrDomain);

/**
 * Set the primary actor of the usecase by name.
 *

```

```

 * @param actorName
 */
public void setPrimaryActorName(String actorName);

/**
 * Used to pass in an existing usecase for editing purposes. If no
usecase
 * is supplied a new one will be created.
 *
 * @param usecase
 */
public void setUseCase(UseCase usecase);

/**
 * Get the usecase created or edited via the command. If a usecase
was
 * supplied via setUseCase the returned usecase will represent the
same
 * entity although it may not be equal (for example the user or name
has
 * changed.)
 *
 * @return
 */
public UseCase getUseCase();

/**
 * @param editStepCommands -
 *          a list of step or scenario edit commands corresponding
to the
 *          steps of the usecase scenario to execute and then add
to the
 *          usecase.
 */
public void setStepCommands(List<EditScenarioStepCommand>
editStepCommands);
}

```

editusecasecommandimpl.java

```

/*
 * $Id: EditUseCaseCommandImpl.java,v 1.16 2009/03/30 11:54:31 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.command.EditActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioStepCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditUseCaseCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchActorException;
import edu.harvard.fas.rregan.requel.project.impl.UseCaseImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("editUseCaseCommand")
@Scope("prototype")
public class EditUseCaseCommandImpl extends
AbstractEditProjectOrDomainEntityCommand implements
EditUseCaseCommand {
    private UseCase usecase;
}

```

```

private String primaryActorName;
private String text;
private List<EditScenarioStepCommand> editStepCommands;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@Autowired
public EditUseCaseCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository,
ProjectRepository projectRepository, ProjectCommandFactory
projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

public UseCase getUseCase() {
    return usecase;
}

public void setUseCase(UseCase usecase) {
    this.usecase = usecase;
}

public void setPrimaryActorName(String primaryActorName) {
    this.primaryActorName = primaryActorName;
}

protected String getPrimaryActorName() {
    return primaryActorName;
}

protected String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

```

```

@Override
public void setStepCommands(List<EditScenarioStepCommand>
editStepCommands) {
    this.editStepCommands = editStepCommands;
}

protected List<EditScenarioStepCommand> getStepCommands() {
    return editStepCommands;
}

@Override
public void execute() throws Exception {
    ProjectOrDomain projectOrDomain =
getRepository().get(getProjectOrDomain());
    User editedBy = getRepository().get(getEditedBy());
    UseCaseImpl usecaseImpl = (UseCaseImpl) getUseCase();

    // check for uniqueness
    try {
        UseCase existing =
getProjectRepository().findUseCaseByProjectOrDomainAndName(
            projectOrDomain, getName());
        if (usecaseImpl == null) {
            throw EntityException.uniquenessConflict(UseCase.class, existing,
FIELD_NAME,
            EntityExceptionActionType.Creating);
        } else if (existing.equals(usecaseImpl)) {
            throw EntityException.uniquenessConflict(UseCase.class, existing,
FIELD_NAME,
            EntityExceptionActionType.Updating);
        }
    } catch (NoSuchEntityException e) {
    }

    Actor primaryActor;
    try {
        primaryActor =
getProjectRepository().findActorByProjectOrDomainAndName(
            projectOrDomain, getPrimaryActorName());
    } catch (NoSuchActorException e) {
        EditActorCommand editActorCommand =
getProjectCommandFactory().newEditActorCommand();
        editActorCommand.setName(getPrimaryActorName());
        editActorCommand.setActorContainer(projectOrDomain);
        editActorCommand.setEditedBy(editedBy);
        editActorCommand.setProjectOrDomain(projectOrDomain);
    }
}

```

```

    // don't analyze the actor because it only has a name at this
    point.
    editActorCommand.setAnalysisEnabled(false);
    primaryActor =
getCommandHandler().execute(editActorCommand).getActor();
}

if (usecaseImpl == null) {
    EditScenarioCommand editScenarioCommand =
getProjectCommandFactory()
    .newEditScenarioCommand();
    editScenarioCommand.setEditedBy(editedBy);
    editScenarioCommand.setName(getName());
    editScenarioCommand.setProjectOrDomain(projectOrDomain);
    editScenarioCommand.setScenarioTypeName(ScenarioType.Primary.name());
};

    editScenarioCommand.setStepCommands(getStepCommands());
    // the scenario will be analyzed when the use case is analyzed.
    editScenarioCommand.setAnalysisEnabled(false);
    editScenarioCommand =
getCommandHandler().execute(editScenarioCommand);
    usecaseImpl = getProjectRepository().persist(
        new UseCaseImpl(projectOrDomain, primaryActor, editedBy,
getName(), getText(),
        editScenarioCommand.getScenario()));
} else {
    if (getName() != null) {
        usecaseImpl.setName(getName());
    }
    if (getText() != null) {
        usecaseImpl.setText(getText());
    }
    usecaseImpl = getProjectRepository().merge(usecaseImpl);
    Actor existingPrimaryActor =
getRepository().get(usecaseImpl.getPrimaryActor());
    if (primaryActor != null) {
        if (existingPrimaryActor != null) && !
primaryActor.equals(existingPrimaryActor)) {
            existingPrimaryActor.getReferers().remove(usecaseImpl);
        }
        usecaseImpl.setPrimaryActor(primaryActor);
    }
    EditScenarioCommand editScenarioCommand =
getProjectCommandFactory()
    .newEditScenarioCommand();
    editScenarioCommand.setScenario(getRepository().get(usecaseImpl.get
Scenario()));
}

```

```

editScenarioCommand.setEditedBy(editedBy);
editScenarioCommand.setName(getName());
editScenarioCommand.setProjectOrDomain(projectOrDomain);
editScenarioCommand.setScenarioTypeName(ScenarioType.Primary.name())
);
editScenarioCommand.setStepCommands(getStepCommands());
// the scenario will be analyzed when the use case is analyzed.
editScenarioCommand.setAnalysisEnabled(false);
editScenarioCommand =
getCommandHandler().execute(editScenarioCommand);
}
if (projectOrDomain != null) {
projectOrDomain.getUseCases().add(usecaseImpl);
}
if (primaryActor != null) {
primaryActor.getReferers().add(usecaseImpl);
}
setUseCase(usecaseImpl);
}

@Override
public void invokeAnalysis() {
if (isAnalysisEnabled()) {
getAssistantManager().analyzeUseCase(getUseCase());
}
}
}

```

editusercommand.java

```

/*
 * $Id: EditUserCommand.java,v 1.3 2009/02/13 12:08:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.user.command;

import java.util.Map;
import java.util.Set;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */

```

```

public interface EditUserCommand extends Command {

/**
 * @return the new or updated user
 */
public User getUser();

/**
 * set the user to edit, if not set a new user will be created.
 *
 * @param user -
 *          the user to edit.
 */
public void setUser(User user);

/**
 * Set the username of the user.
 *
 * @param username
 */
public void setUsername(String username);

/**
 * Set the password of the user
 *
 * @param password
 */
public void setPassword(String password);

/**
 * Set the repeated password
 *
 * @param repassword
 */
public void setRepassword(String repassword);

/**
 * Set the name of the user.
 *
 * @param name
 */
public void setName(String name);

/**
 * Set the email address of the user.
 *
 * @param emailAddress
 */

```

```

*/
public void setEmailAddress(String emailAddress);

/**
 * Set the phoneNumber of the user.
 *
 * @param phoneNumber
 */
public void setPhoneNumber(String phoneNumber);

/**
 * Set the default organization name of the user. if the organization
 * doesn't exist it will be created.
 *
 * @param organizationName
 */
public void setOrganizationName(String organizationName);

/**
 * Set true if this user account can be edited by the user.
 *
 * @param editable
 */
public void setEditable(Boolean editable);

/**
 * Set the names of the user roles this user has.
 *
 * @param userRoleNames
 */
public void setUserRoleNames(Set<String> userRoleNames);

/**
 * Add a role by name to grant to the user.
 *
 * @param userRoleName
 */
public void addUserRoleName(String userRoleName);

/**
 * Set the names of the permissions for each of the roles that the
 * user has.
 *
 * @param userRolePermissionNames
 */
public void setUserRolePermissionNames(Map<String, Set<String>>
userRolePermissionNames);

```

```

/**
 * Add a permission for a role name to grant to the user.
 *
 * @param userRoleName
 * @param userRolePermissionName
 */
public void addUserRolePermissionName(String userRoleName, String
userRolePermissionName);

/**
 * @param editedBy -
 *          the user invoking the command
 */
public void setEditedBy(User editedBy);
}

```

editusercommandimpl.java

```

/*
 * $Id: EditUserCommandImpl.java,v 1.7 2009/02/17 11:50:51 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.command;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRole;
import edu.harvard.fas.rregan.requel.user UserRolePermission;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;

```

```

import
edu.harvard.fas.rregan.requel.user.exception.NoSuchOrganizationException;
import edu.harvard.fas.rregan.requel.user.impl.OrganizationImpl;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;

/**
 * @author ron
 */
@Component("editUserCommand")
@Scope("prototype")
public class EditUserCommandImpl extends AbstractUserCommand
implements EditUserCommand {

    private User user;
    private String username;
    private String password;
    private String repassword;
    private String name;
    private String emailAddress;
    private String phoneNumber;
    private String organizationName;
    private Boolean editable = Boolean.TRUE;
    private Set<String> userRoleNames = new HashSet<String>();
    private Map<String, Set<String>> userRolePermissionNames = new
HashMap<String, Set<String>>();
    private User editedBy;

    /**
     * @param userRepository
     */
    @Autowired
    public EditUserCommandImpl(UserRepository userRepository) {
        super(userRepository);
    }

    /**
     * @see edu.harvard.fas.rregan.command.Command#execute()
     */
    @Override
    public void execute() {
        UserImpl userImpl = (UserImpl) getUser();
        if (userImpl == null) {
            userImpl = createUser();
        } else {
            userImpl = updateUser(userImpl);
        }
    }
}

setUser(userImpl);

}

protected void validate() {
    if ((getPassword() != null) && !
getPassword().equals(getRepassword())) {
        throw EntityValidationException.validationFailed(User.class,
"password",
"The password fields don't match.");
}
}

private UserImpl createUser() {
    Organization organization =
getOrCreateOrganization(getOrganizationName());
    UserImpl userImpl = new UserImpl(getUsername(), getPassword(),
getRepassword(), getName(),
getEmail(), getPhoneNumber(), organization, getEditable());
    updateRoles(userImpl);
    return getUserRepository().persist(userImpl);
}

private UserImpl updateUser(UserImpl userImpl) {
    Organization organization =
getOrCreateOrganization(getOrganizationName());
    userImpl.setName(getName());
    userImpl.resetPassword(getPassword(), getRepassword());
    userImpl.setEmailAddress(getEmail());
    userImpl.setPhoneNumber(getPhoneNumber());
    userImpl.setOrganization(organization);
    if (getEditedBy().hasRole(SystemAdminUserRole.class)) {
        userImpl.setUsername(getUsername());
        userImpl.setEditable(getEditable());
        updateRoles(userImpl);
    }
    return getUserRepository().merge(userImpl);
}

private Organization getOrCreateOrganization(String organizationName)
{
    try {
        return
getUserRepository().findOrganizationByName(getOrganizationName());
    } catch (NoSuchOrganizationException e) {
        return getUserRepository().persist(new
OrganizationImpl(getOrganizationName()));
    }
}

```

```

}

private void updateRoles(UserImpl userImpl) {
    for (Class<? extends UserRole> userRoleType :
        getUserRepository().findUserRoleTypes()) {
        if (getUserNames().contains(userRoleType.getSimpleName())) {
            userImpl.grantRole(userRoleType);
            AbstractUserRole role = (AbstractUserRole)
                userImpl.getRoleForType(userRoleType);
            for (UserRolePermission permission :
                getUserRepository().findUserRolePermissions(
                    userRoleType)) {
                Set<String> permissionNames =
                    getUserRolePermissionNames().get(
                        role.getRoleName());
                if (permissionNames != null) {
                    if (permissionNames.contains(permission.getName())) {
                        role.grantUserRolePermission(permission);
                    } else {
                        role.revokeUserRolePermission(permission);
                    }
                }
            }
        } else if (userImpl.hasRole(userRoleType)) {
            userImpl.revokeRole(userRoleType);
        }
    }
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

protected String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

protected String getPassword() {
    return password;
}

```

```

public void setPassword(String password) {
    this.password = password;
}

protected String getRepassword() {
    return repassword;
}

public void setRepassword(String repassword) {
    this.repassword = repassword;
}

protected String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

protected String getEmailAddress() {
    return emailAddress;
}

public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
}

protected String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

protected String getOrganizationName() {
    return organizationName;
}

public void setOrganizationName(String organizationName) {
    this.organizationName = organizationName;
}

protected Boolean getEditable() {
    return editable;
}

```

```

}

public void setEditable(Boolean editable) {
    this.setEditable = editable;
}

protected Set<String> getUserRoleNames() {
    return userRoleNames;
}

public void setUserRoleNames(Set<String> userRoleNames) {
    this.userRoleNames = userRoleNames;
}

public void addUserRoleName(String userRoleName) {
    userRoleNames.add(userRoleName);
}

protected Map<String, Set<String>> getUserRolePermissionNames() {
    return userRolePermissionNames;
}

public void setUserRolePermissionNames(Map<String, Set<String>>
userRolePermissionNames) {
    this.userRolePermissionNames = userRolePermissionNames;
}

public void addUserRolePermissionName(String userRoleName, String
userRolePermissionName) {
    if (!userRolePermissionNames.containsKey(userRoleName)) {
        userRolePermissionNames.put(userRoleName, new HashSet<String>());
    }
    userRolePermissionNames.get(userRoleName).add(userRolePermissionName);
}

@Override
public void setEditedBy(User editedBy) {
    this.editedBy = editedBy;
}

protected User getEditedBy() {
    return editedBy;
}

```

editverbnetselectionrestrictioncommand.java

```

/*
 * $Id: EditVerbNetSelectionRestrictionCommand.java,v 1.1 2009/02/09
10:12:31 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;
import
edu.harvard.fas.rregan.nlp.dictionary.VerbNetSelectionRestriction;
import
edu.harvard.fas.rregan.nlp.dictionary.VerbNetSelectionRestrictionType;

/**
 * @author ron
 */
public interface EditVerbNetSelectionRestrictionCommand extends
Command {

    public void setVerbNetRoleRef(VerbNetRoleRef roleRef);

    public void
setVerbNetSelectionRestrictionType(VerbNetSelectionRestrictionType
selResType);

    public void setInclude(String include);

    public void
setVerbNetSelectionRestriction(VerbNetSelectionRestriction selRes);

    public VerbNetSelectionRestriction getVerbNetSelectionRestriction();

    public VerbNetRoleRef getVerbNetRoleRef();
}

```

editverbnetselectionrestrictioncommandimpl.java

```

/*
 * $Id: EditVerbNetSelectionRestrictionCommandImpl.java,v 1.1
2009/02/09 10:12:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

/*
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;
import edu.harvard.fas.rregan.nlp.dictionary.VerBNetSelectionRestriction;
import edu.harvard.fas.rregan.nlp.dictionary.VerBNetSelectionRestrictionType;
import edu.harvard.fas.rregan.nlp.dictionary.command.EditVerBNetSelectionRest
rictionCommand;

/**
 * @author ron
 */
@Controller("editVerBNetSelectionRestrictionCommand")
@Scope("prototype")
public class EditVerBNetSelectionRestrictionCommandImpl extends
AbstractDictionaryCommand implements
EditVerBNetSelectionRestrictionCommand {

    private VerBNetRoleRef roleRef;
    private VerBNetSelectionRestrictionType selResType;
    private String include;
    private VerBNetSelectionRestriction selRes;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public
EditVerBNetSelectionRestrictionCommandImpl(DictionaryRepository
dictionaryRepository) {
    super(dictionaryRepository);
}

@Override
public VerBNetRoleRef getVerBNetRoleRef() {
    return roleRef;
}

@Override

```

```

public void setVerBNetRoleRef(VerBNetRoleRef roleRef) {
    this.roleRef = roleRef;
}

protected String getInclude() {
    return include;
}

@Override
public void setInclude(String include) {
    this.include = include;
}

@Override
public VerBNetSelectionRestriction getVerBNetSelectionRestriction() {
    return selRes;
}

@Override
public void
setVerBNetSelectionRestriction(VerBNetSelectionRestriction selRes) {
    this.selRes = selRes;
}

@Override
public void
setVerBNetSelectionRestrictionType(VerBNetSelectionRestrictionType
selResType) {
    this.selResType = selResType;
}

protected VerBNetSelectionRestrictionType
getVerBNetSelectionRestrictionType() {
    return this.selResType;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {
        VerBNetRoleRef roleRef =
getDictionaryRepository().get(getVerBNetRoleRef());
        VerBNetSelectionRestrictionType type =
getDictionaryRepository().get(
            getVerBNetSelectionRestrictionType());
    }
}

```

```

VerbNetSelectionRestriction restriction =
getVerbNetSelectionRestriction();
if (restriction == null) {
    restriction = new VerbNetSelectionRestriction(roleRef, type,
getInclude());
    getDictionaryRepository().persist(restriction);
} else {
    // TODO: edit an existing?
}
roleRef.getSelectionalRestrictions().add(restriction);
setVerbNetRoleRef(getDictionaryRepository()).merge(roleRef));
setVerbNetSelectionRestriction(restriction);
} catch (Exception e) {
    log.error(e, e);
}
}
}

```

entityexception.java

```

/*
 * $Id: EntityException.java,v 1.2 2009/02/17 11:50:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository;

import edu.harvard.fas.rregan.requel.RequelException;

/**
 * @author ron
 */
public class EntityException extends RequelException {
    static final long serialVersionUID = 0;

    private final Class<?> entityType;
    private final Object entity;
    private final String[] entityPropertyNames;
    private final Object[] entityPropertyValues;
    private final EntityExceptionActionType actionType;
    private final boolean staleEntity;

    protected static String MSG_UNKNOWN_PROBLEM = "An unexpected problem
occured %s the %s concerning %s";
    protected static String MSG_UNIQUENESS_CONFICT = "The %s conflicts
with an existing %s";

```

```

    /**
     * Create an exception for an unexpected problem (it doesn't have a
specific
     * exception to describe it.
     *
     * @param cause
     * @param entityType
     * @param entity
     * @param actionType
     * @return
     */
    public static EntityException forUnknownProblem(Throwable cause,
Class<?> entityType,
Object entity, EntityExceptionActionType actionType) {
        return forUnknownProblem(cause, entityType, entity, new String[] {},,
new Object[] {},,
actionType);
    }

    /**
     * Create an exception for an unexpected problem (it doesn't have a
specific
     * exception to describe it.
     *
     * @param cause
     * @param entityType
     * @param entity
     * @param entityPropertyName
     * @param entityPropertyValue
     * @param actionType
     * @return
     */
    public static EntityException forUnknownProblem(Throwable cause,
Class<?> entityType,
Object entity, String entityPropertyName, Object
entityPropertyValue,
EntityExceptionActionType actionType) {
        return forUnknownProblem(cause, entityType, entity, new String[] []
{ entityPropertyName },
        new Object[] { entityPropertyValue }, actionType);
    }

    /**
     * Create an exception for an unexpected problem (it doesn't have a
specific
     * exception to describe it.
     */

```

```

/*
 * @param cause
 * @param entityType
 * @param entity
 * @param entityPropertyNames
 * @param entityPropertyValues
 * @param actionType
 * @return
 */
public static EntityException forUnknownProblem(Throwable cause,
Class<?> entityType,
Object entity, String[] entityPropertyNames, Object[]
entityPropertyValues,
EntityExceptionActionType actionType) {
StringBuilder propertyInfo = new StringBuilder();
if ((entityPropertyNames != null) && (entityPropertyNames.length >
0)) {
for (int i = 0; i < entityPropertyNames.length; i++) {
propertyInfo.append(entityPropertyNames[i]);
propertyInfo.append(" of '");
propertyInfo.append(entityPropertyNames[i]);
propertyInfo.append("'");
if (i < entityPropertyNames.length - 1) {
propertyInfo.append(" and ");
}
} else {
propertyInfo.append("unknown property values.");
}
return new EntityException(cause, entityType, entity,
entityPropertyNames,
entityPropertyValues, actionType, MSG_UNKNOWN_PROBLEM,
actionType.name(),
(entityType != null ? entityType.getSimpleName() : "<unknown>"),
propertyInfo
.toString());
}

/**
 * @param entityType
 * @param entity
 * @param propertyName
 * @param actionType
 * @return
 */
public static EntityException uniquenessConflict(Class<?> entityType,
Object entity,

```

```

String propertyName, EntityExceptionActionType actionType) {
return new EntityException(entityType, entity, new String[]
{ propertyName }, null,
actionType, MSG_UNIQUENESS_CONFICT, propertyName, (entityType !=
null ? entityType
.getSimpleName() : "<unknown>"));
}

/**
 * @param cause
 * @param msg
 * @return
 */
public static EntityException uniquenessConflict(Throwable cause,
String msg) {
return new EntityException(cause, null, null, new String[] {}, new
Object[] {}),
EntityExceptionActionType.Updating, MSG_PRE_GENERATED, msg);
}

protected EntityException(Class<?> entityType, Object entity,
String[] entityPropertyNames,
Object[] entityPropertyValues, EntityExceptionActionType
actionType, String format,
Object... messageArgs) {
this(entityType, entity, entityPropertyNames, entityPropertyValues,
actionType, false,
format, messageArgs);
}

protected EntityException(Class<?> entityType, Object entity, String
entityPropertyName,
Object entityPropertyValue, EntityExceptionActionType actionType,
String format,
Object... messageArgs) {
this(entityType, entity, new String[] { entityPropertyName },
new Object[] { entityPropertyValue }, actionType, false, format,
messageArgs);
}

/**
 * @param entityType
 * @param entity
 * @param entityPropertyName
 * @param entityPropertyValues
 * @param actionType
 * @param staleEntity

```

```

 * @param format
 * @param messageArgs
 */
protected EntityException(Class<?> entityType, Object entity,
String[] entityPropertyNames,
Object[] entityPropertyValues, EntityExceptionActionType
actionType,
boolean staleEntity, String format, Object... messageArgs) {
super(format, messageArgs);
this.entityType = entityType;
this.entity = entity;
this.entityPropertyNames = entityPropertyNames;
this.entityPropertyValues = entityPropertyValues;
this.actionType = actionType;
this.staleEntity = staleEntity;
}

protected EntityException(Throwable cause, Class<?> entityType,
Object entity,
String entityPropertyName, Object entityPropertyValue,
EntityExceptionActionType actionType, String format, Object...
messageArgs) {
this(cause, entityType, entity, new String[] { entityPropertyName },
new Object[] { entityPropertyValue }, actionType, format,
messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected EntityException(Throwable cause, Class<?> entityType,
Object entity,
String[] entityPropertyNames, Object[] entityPropertyValues,
EntityExceptionActionType actionType, String format, Object...
messageArgs) {
super(cause, format, messageArgs);
this.entityType = entityType;
this.entity = entity;
this.entityPropertyNames = entityPropertyNames;
this.entityPropertyValues = entityPropertyValues;
this.actionType = actionType;
this.staleEntity = false;
}

/**

```

```

 * @return the type (class) of entity in getEntity
 */
public Class<?> getEntityType() {
    return entityType;
}

/**
 * @return the entity (object) that caused the exception
 */
public Object getEntity() {
    return entity;
}

/**
 * @return the name of the property on the entity that caused the
exception.
 */
public String[] getEntityPropertyNames() {
    return entityPropertyNames;
}

/**
 * @return the value of the property that caused the exception.
 */
public Object[] getEntityPropertyValues() {
    return entityPropertyValues;
}

/**
 * @return the type of activity (create, read, update, or delete) on
the
 *         entity when the exception occurred.
 */
public EntityExceptionActionType getActionType() {
    return actionType;
}

/**
 * @return true if this exception was caused by an out of date
entity.
 */
public boolean isStaleEntity() {
    return staleEntity;
}
}
```

entityexceptionactiontype.java

```
/*
 * $Id: EntityExceptionActionType.java,v 1.1 2008/12/13 00:40:41
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository;

/**
 * @author ron
 */
public enum EntityExceptionActionType {

    Creating(), Reading(), Updating(), Deleting(), Unknown();

    private EntityExceptionActionType() {

    }
}
```

entityexceptionadapter.java

```
/*
 * $Id: EntityExceptionAdapter.java,v 1.1 2008/12/13 00:40:43 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository;

/**
 * @author ron
 */
public interface EntityExceptionAdapter {

    /**
     * Convert an API specific exception into an appropriate
     EntityException
     * with an appropriate message to display to the user.
     *
     * @param original -
     *          the original API specific exception/throwable thrown.
```

```
     * @param entityType -
     *          the canonical type of Object (interface) to display.
     * @param entity -
     *          the entity that caused the exception
     * @param actionType -
     *          the type of action (create, read, update, delete) being
done
     *          when the exception happened.
     * @return
     */
    public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
        EntityExceptionActionType actionType);

}
```

entityexistsexceptionadapter.java

```
/*
 * $Id: EntityExistsExceptionAdapter.java,v 1.1 2008/12/13 00:41:16
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository.jpa;

import org.hibernate.exception.ConstraintViolationException;

import com.mysql.jdbc.exceptions.MySQLIntegrityConstraintViolationException;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;

/**
 * @author ron
 */
public class EntityExistsExceptionAdapter implements
EntityExceptionAdapter {

    /**
     * @param propertyName -
     *          name of the property that must be unique.
     */
    public EntityExistsExceptionAdapter() {
```

```

}

/**
 * @see
edu.harvard.fas.rregan.repository.EntityExceptionAdapter#convert(java.
lang.Throwable,
 *      java.lang.Object,
 *      edu.harvard.fas.rregan.repository.EntityExceptionActionType)
 */
@Override
public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
EntityExceptionActionType actionType) {

    if (entityType == null) {
        if (entity == null) {
            if
(ConstraintViolationException.class.equals(original.getCause().getClas
s())) {
                ConstraintViolationException cve = (ConstraintViolationException)
original.getCause();
                if
(MySQLIntegrityConstraintViolationException.class.equals(cve.getCause(
).getClass())) {
                    MySQLIntegrityConstraintViolationException icve =
(MySQLIntegrityConstraintViolationException) cve.getCause();
                    return EntityException.uniquenessConflict(icve,
icve.getMessage());
                }
            }
        }
    }
    return EntityException.uniquenessConflict(entityType, entity, null,
actionType);
}

```

entitylockexception.java

```

/*
 * $Id: EntityLockException.java,v 1.3 2008/12/18 02:01:54 rregan Exp
$Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.request;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;

/**
 * @author ron
 */
public class EntityLockException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_STALE_ENTITY = "The %s is out of date.";
    protected static String MSG_LOCK_ENTITY = "failed to acquire lock.";

    /**
     * @param entityType
     * @param entity
     * @param actionType
     * @return
     */
    public static EntityLockException staleEntity(Class<?> entityType,
Object entity,
EntityExceptionActionType actionType) {
        return new EntityLockException(entityType, entity, null, null,
actionType,
        MSG_STALE_ENTITY, (entityType == null ? "Unknown
Type":entityType.getSimpleName()));
    }

    /**
     * @param entityType
     * @param entity
     * @param actionType
     * @return
     */
    public static EntityLockException deadlock(Class<?> entityType,
Object entity,
EntityExceptionActionType actionType) {
        return new EntityLockException(entityType, entity, null, null,
actionType, MSG_LOCK_ENTITY,
        entityType.getSimpleName());
    }

    /**
     * @param format
     * @param args
     */

```

```

protected EntityLockException(Class<?> entityType, Object entity,
String entityPropertyName,
Object entityValue, EntityExceptionActionType actionType, String
format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected EntityLockException(Throwable cause, Class<?> entityType,
Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

entityproxyinterceptor.java

```

/*
 * $Id: EntityProxyInterceptor.java,v 1.4 2009/04/01 15:45:45 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

import net.sf.cglib.proxy.Callback;
import net.sf.cglib.proxy.Factory;
import net.sf.cglib.proxy.MethodInterceptor;
import net.sf.cglib.proxy.MethodProxy;

import org.apache.log4j.Logger;
import org.hibernate.LazyInitializationException;

/**

```

```

 * A MethodInterceptor that gets assigned to a proxy for entity
objects by the
 * DomainObjectWrappingAdvice. The proxy is an empty dummy object
which acts as
 * a reference to the entity. The "real" entity is stored in this
interceptor
 * and refreshed as needed.<br>
 * TODO: as data is loaded from the db it should be cached in the
local copy for
 * faster access. Ideally only lazy loaded objects should require
accessing the
 * database. Also, how will updates in other transactions get
propogated to this
 * object?
 *
 * @author ron
 */
public class EntityProxyInterceptor implements MethodInterceptor {
protected static final Logger log =
Logger.getLogger(EntityProxyInterceptor.class);
private long lastRefreshTime = 0;
private Object entity;
final private PersistenceContextHelper persistenceContextHelper;
final private DomainObjectWrapper domainObjectWrapper;

/**
 * @param persistenceContextHelper -
 *           a helper that attaches/gets the latest version of the
entity.
 * @param domainObjectWrapper -
 *           a helper that wraps entities in a proxy
 * @param entity -
 *           the real object being proxied, the proxy is really a
dummy
 *           object.
 */
public EntityProxyInterceptor(PersistenceContextHelper
persistenceContextHelper,
DomainObjectWrapper domainObjectWrapper, Object entity) {
this.persistenceContextHelper = persistenceContextHelper;
this.domainObjectWrapper = domainObjectWrapper;
this.entity = entity;
}

public Object intercept(Object proxy, Method method, Object[] args,
MethodProxy methodProxy)
throws Throwable {

```

```

method.setAccessible(true);
if ("finalize".equals(method.getName())) {
    return null;
}
if (isSetterMethod(method)) {
    log.debug("possibly setting a persistent value on a proxy,"
        + " which may be bad outside of a command.");
}
Object retVal;
if ((System.currentTimeMillis() - lastRefreshTime) > 1000) {
    // TODO: what about using the entity version if available instead
    // of
    // reloading the whole object from the database?
    retVal = persistenceContextHelper.invokeInTransaction(this,
domainObjectWrapper,
    method, args);
} else {
    // try to invoke the method on the cached version, if it fails
    // because of lazy loading, then invoke the method
    // transactionally.
    try {
        try {
            retVal = domainObjectWrapper
                .wrapPersistentEntities(method.invoke(entity, args));
        } catch (InvocationTargetException e) {
            throw e.getCause();
        }
    } catch (LazyInitializationException e) {
        retVal = persistenceContextHelper.invokeInTransaction(this,
domainObjectWrapper,
    method, args);
    }
}
return retVal;
}

/**
 * @return the current cached version of the entity
 */
public Object getEntity() {
    return entity;
}

/**
 * set the cached version of the entity.
 *
 * @param entity

```

```

    /*
     * If the supplied object is a proxy with "advice" from the
EntityProxy,
     * return the entity attached to the callback, otherwise return the
supplied
     * object.
     *
     * @param possibleProxy
     * @return
     */
    public static <T> T unwrap(T possibleProxy) {
        if (possibleProxy instanceof Factory) {
            Callback callback = ((Factory) possibleProxy).getCallback(0);
            if (callback instanceof EntityProxyInterceptor) {
                return (T) ((EntityProxyInterceptor) callback).getEntity();
            }
        }
        return possibleProxy;
    }

    /**
     * @param possibleProxy
     * @return true if possibleProxy is an EntityProxy
     */
    public static boolean isEntityProxy(Object possibleProxy) {
        if (possibleProxy instanceof Factory) {
            Callback callback = ((Factory) possibleProxy).getCallback(0);
            if (callback instanceof EntityProxyInterceptor) {
                return true;
            }
        }
        return false;
    }

    /**
     * @param method
     * @return
     */
    public static boolean isSetterMethod(Method method) {
        if (method.getName().startsWith("set")) {
            return true;
        }
    }

```

```

    }
    return false;
}
}

```

entityvalidationexception.java

```

/*
 * $Id: EntityValidationException.java,v 1.5 2009/02/17 11:50:46
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel;

import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;

/**
 * @author ron
 */
public class EntityValidationException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_VALIDATION_FAILED = "validation failed:
%s";
    protected static String MSG_EMPTY_VALUE = "%s cannot be empty.';

    /**
     * Create an EntityException for an empty property that is required.
     *
     * @param entityType
     * @param entity
     * @param propertyName
     * @return a EntityException
     */
    public static EntityValidationException emptyRequiredProperty(Class<?
> entityType,
        Object entity, String propertyName, EntityExceptionActionType
actionType) {
        return new EntityValidationException(entityType, entity,
            propertyName, null, actionType,

```

```

            MSG_EMPTY_VALUE, propertyName);
    }

    /**
     * @param entityType
     * @param propertyName
     * @param message
     * @return
     */
    public static EntityValidationException validationFailed(Class<?>
entityType,
        String propertyName, String message) {
        return new EntityValidationException(entityType, null, propertyName,
            null,
            EntityExceptionActionType.Unknown, MSG_VALIDATION_FAILED,
            message);
    }

    /**
     * @param cause
     * @param entityType
     * @param entity
     * @param actionType
     * @return
     */
    public static EntityValidationException
validationFailed(InvalidStateException cause,
        Class<?> entityType, Object entity, EntityExceptionActionType
actionType) {
        StringBuilder causeMsg = new StringBuilder();
        for (InvalidValue value : cause.getInvalidValues()) {
            causeMsg.append(System.getProperty("line.separator"));
            causeMsg.append(value.getBeanClass().getSimpleName());
            causeMsg.append(" ");
            causeMsg.append(value.getPropertyName());
            causeMsg.append(" ");
            causeMsg.append(value.getValue());
            causeMsg.append(": ");
            causeMsg.append(value.getMessage());
        }
        return new EntityValidationException(cause, entityType, entity,
            null, null, actionType,
            MSG_VALIDATION_FAILED, causeMsg);
    }

    /**
     * @param entityType

```

```

* @param entity
* @param entityPropertyName
* @param entityValue
* @param actionType
* @param format
* @param messageArgs
*/
public EntityValidationException(Class<?> entityType, Object entity,
String entityPropertyName,
    Object entityValue, EntityExceptionActionType actionType, String
format,
    Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
* @param cause
* @param entityType
* @param entity
* @param entityPropertyName
* @param entityValue
* @param actionType
* @param format
* @param messageArgs
*/
public EntityValidationException(Throwable cause, Class<?>
entityType, Object entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
    messageArgs);
}

@Override
public InvalidStateException getCause() {
    return (InvalidStateException) super.getCause();
}
}

eventdispatcher.java

/*
 * $Id: EventDispatcher.java,v 1.6 2008/09/12 00:15:11 rregan Exp $

```

```

* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.uiframework.navigation.event;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelDescriptor;

/**
 * The EventDispatcher is a broker for events. Listeners can be
registered for
 * specific types of events, open panel events for specific types of
panels, and
 * for all events to specific panel instances.<br>
 * PanelManagers use the add/remove methods to
*
* @author ron
*/
public interface EventDispatcher extends ActionListener {

/**
 * @param event
 */
public void dispatchEvent(ActionEvent event);

/**
 * Add a listener for a specific type of event or any sub class of
that
 * event.<br>
 * This method is most often used by Controllers such as the
 * LoginController or simple Listeners such as
 * NavigatorTreeNodeUpdateListener that listens for entity changes
and
 * updates a navigation tree.
*
* @param eventType -
*          the class of an ActionEvent or sub class to listen for.
* @param listener -
*          a Controller or simple listener that will receive the
event
*          when dispatched through the dispatchEvent() method.
* @return the supplied listener
*/
public ActionListener addEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener);

```

eventdispatcher.java

```

/**
 * Add a listener for a specific type of event or any sub class of
that
 * event and a specific destination, such as a select button.
 *
 * @param eventType -
 *      the class of an ActionEvent or sub class to listen for.
 * @param listener -
 *      a Controller or simple listener that will receive the
event
 *      when dispatched through the dispatchEvent() method.
 * @param destinationObject -
 *      the destination object. Events will only get fired to
the listener
 *      if the destination of the event matches the registered
destination.
 * @return the supplied listener
 */
public ActionListener addEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener, Object destinationObject);

/**
 * Remove a listener for a specific type of event.
 *
 * @param eventType -
 *      the class of an ActionEvent or sub class to listen for.
 * @param listener -
 *      the Controller or simple listener that was registered
to
 *      receive the events and will be removed.
 */
public void removeEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener);

/**
 * Remove a listener for a specific type of event and destination.
Listeners registered
 * with the addEventTypeActionListener() that takes a
destinationObject should be
 * removed with this method.
 *
 * @param eventType -
 *      the class of an ActionEvent or sub class to listen for.
 * @param listener -

```

```

        *      the Controller or simple listener that was registered
to
        *      receive the events and will be removed.
        * @param destinationObject -
        *      the destination object the listener is registered with.
 */
public void removeEventTypeActionListener(Class<? extends
ActionEvent> eventType,
    ActionListener listener, Object destinationObject);

/**
 * Add a listener for an OpenPanelEvent for the specified
PanelDescriptor.
 * When an OpenPanelEvent is dispatched that matches the
panelDescriptor,
 * the specified listener will be notified.
 *
 * @param panelDescriptor
 * @param listener
 * @return the supplied listener
 */
public ActionListener addOpenPanelEventActionListener(PanelDescriptor
panelDescriptor,
    ActionListener listener);

/**
 * Remove a listener for an OpenPanelEvent for the specified
 * PanelDescriptor.
 *
 * @param panelDescriptor
 * @param listener
 */
public void removeOpenPanelEventActionListener(PanelDescriptor
panelDescriptor,
    ActionListener listener);

/**
 * Add a listener for all events to a specific panel. The
PanelManager
 * should call this each time a panel is opened so that when it is
hidden
 * and re-displayed the manager is notified.
 *
 * @param panel
 * @param listener
 * @return the supplied listener
 */

```

```

public ActionListener addPanelInstanceEventActionListener(Panel
panel, ActionListener listener);

/**
 * Remove a listener for the panel for close events. The PanelManager
should
 * call this each time a panel is closed.
*
* @param panel
* @param listener
*/
public void removePanelInstanceEventActionListener(Panel panel,
ActionListener listener);
}

```

eventdispatcherexception.java

```

/*
 * $Id: EventDispatcherException.java,v 1.2 2008/02/16 22:42:40 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import edu.harvard.fas.rregan.uiframework.UIFrameworkException;

public class EventDispatcherException extends UIFrameworkException {
    static final long serialVersionUID = 0L;

    public static String MSG_ILLEGAL_EVENT_SOURCE_TYPE = "The event
source %s is a class that isn't a Command.";
    public static String MSG_ILLEGAL_EVENT_SOURCE_INSTANCE = "The event
source %s of type %s has no addActionListener method to register
ActionListeners.";
    public static String MSG_PROBLEM_REGISTERING_LISTENER = "couldn't
register EventDispatcher as listener for event source %s,
addActionListener threw an exception: %s";
    public static String MSG_ILLEGAL_EVENT_TYPE = "The eventType %s is
not an ActionEvent class or String.";

    public static EventDispatcherException illegalEventSourceType(Class<?
> eventSourceType) {
        return new EventDispatcherException(MSG_ILLEGAL_EVENT_SOURCE_TYPE,
eventSourceType);
    }
}

```

```

public static EventDispatcherException
illegalEventSourceObject(Object eventSource) {
    String eventSourceTypeName = (eventSource instanceof Class ?
((Class<?>) eventSource)
        .getName() : eventSource.getClass().getName());
    return new
EventDispatcherException(MSG_ILLEGAL_EVENT_SOURCE_INSTANCE,
eventSource,
    eventSourceTypeName);
}

public static EventDispatcherException
problemRegisteringListenerToSource(Object eventSource,
    Throwable t) {
    return new
EventDispatcherException(MSG_PROBLEM_REGISTERING_LISTENER,
eventSource, t);
}

public static EventDispatcherException illegalEventType(Object
eventType) {
    String eventTypeName = (eventType instanceof Class ? ((Class<?>)
eventType).getName()
        : eventType.getClass().getName());
    return new EventDispatcherException(MSG_ILLEGAL_EVENT_TYPE,
eventTypeName);
}

/**
 * @param format -
*           a format string appropriate for java.util.Formatter
* @param args -
*           variable args list that map to the variables in the
format
*           string
*/
protected EventDispatcherException(String format, Object... args) {
    super(format, args);
}

/**
 * @param cause -
*           a caught exception that resulted in this exception
* @param format -
*           a format string appropriate for java.util.Formatter
* @param args -
*/

```

```

        *      variable args list that map to the variables in the
format
        *      string
*/
protected EventDispatcherException(Throwable cause, String format,
Object... args) {
    super(cause, format, args);
}
}

```

eventdispatchermultiexception.java

```

/*
 * $Id: EventDispatcherMultiException.java,v 1.2 2008/02/22 21:51:41
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import java.util.Map;

import nextapp.echo2.app.event.ActionListener;

public class EventDispatcherMultiException extends
EventDispatcherException {
    static final long serialVersionUID = 0L;

    private final Map<ActionListener, Exception> listenerExceptions;
    private final String message;

    public EventDispatcherMultiException(Map<ActionListener, Exception>
listenerExceptions) {
        super("");
        this.listenerExceptions = listenerExceptions;
        StringBuilder msg = new StringBuilder();
        if (listenerExceptions != null && listenerExceptions.size() > 0) {
            if (listenerExceptions.size() > 1) {
                msg.append("multiple listeners threw exceptions: ");
                msg.append(System.getProperty("line.separator"));
            }
            for (ActionListener listener : listenerExceptions.keySet()) {
                msg.append(listener.getClass().getName());
                msg.append(": ");
                msg.append(listenerExceptions.get(listener).toString());
                msg.append(System.getProperty("line.separator"));
            }
        }
    }
}

```

```

} else {
    msg.append("no exceptions");
}
message = msg.toString();
}

public Map<ActionListener, Exception> getListenerExceptions() {
    return listenerExceptions;
}

@Override
public String getMessage() {
    return message;
}

@Override
public String toString() {
    return getClass().getName() + ":" + getMessage();
}
}

```

exceptionmapper.java

```

/*
 * $Id: ExceptionMapper.java,v 1.4 2009/03/22 11:08:22 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Stack;

import javax.persistence.EntityExistsException;
import javax.persistence.OptimisticLockException;

import org.apache.log4j.Logger;
import org.hibernate.PropertyValueException;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.ConstraintViolationException;
import org.hibernate.exception.LockAcquisitionException;
import org.hibernate.validator.InvalidStateException;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.dao.CannotAcquireLockException;
import
org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureEx
ception;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;

/**
 * Map exceptions related to persistence and validation activities or
entities
 * from other packages to project specific exceptions for easier
handling.
 *
 * @author ron
 */
@Component("exceptionMapper")
@Scope("singleton")
public class ExceptionMapper {
    protected static final Logger log =
Logger.getLogger(ExceptionMapper.class);
    private final Map<ExceptionMapKey, EntityExceptionAdapter>
exceptionMap = new HashMap<ExceptionMapKey, EntityExceptionAdapter>();

    /**
     */
    public ExceptionMapper() {
        // TODO: make this configurable through spring
        addExceptionAdapter(PropertyValueException.class,
            new GenericPropertyValueExceptionAdapter());
        addExceptionAdapter(InvalidStateException.class, new
        InvalidStateExceptionAdapter());
        addExceptionAdapter(ConstraintViolationException.class,
            new ConstraintViolationExceptionAdapter("unknown"));
        addExceptionAdapter(OptimisticLockException.class, new
        OptimisticLockExceptionAdapter());
        addExceptionAdapter(StaleObjectStateException.class, new
        OptimisticLockExceptionAdapter());
        addExceptionAdapter(LockAcquisitionException.class, new
        OptimisticLockExceptionAdapter());
        addExceptionAdapter(CannotAcquireLockException.class, new
        OptimisticLockExceptionAdapter());
        addExceptionAdapter(HibernateOptimisticLockingFailureException.class
        ,
    }

    new OptimisticLockExceptionAdapter()));

    addExceptionAdapter(EntityExistsException.class, new
EntityExistsExceptionAdapter());
}

/**
 * Use to programmatically add exception mapping
 *
 * @param exceptionType
 * @param adapter
 * @param entityClasses
 */
public void addExceptionAdapter(Class<? extends Throwable>
exceptionType,
        EntityExceptionAdapter adapter, Class<?>... entityClasses) {
    if (entityClasses.length > 0) {
        for (Class<?> entityClass : entityClasses) {
            exceptionMap.put(new ExceptionMapKey(exceptionType, entityClass),
adapter);
        }
    } else {
        exceptionMap.put(new ExceptionMapKey(exceptionType, null),
adapter);
    }
}

/**
 * Convert the supplied exception to an EntityException, or if the
supplied
 * exception is already an EntityException, return it. This method
should
 * only be used if more information about the entity that the
exception is
 * concerning is not known.
 *
 * @see #convertException(Exception, Class, Object,
 *          EntityExceptionActionType)
 * @param exception
 * @return
 */
public RuntimeException convertException(Exception exception) {
    return convertException(exception, null, null,
EntityExceptionActionType.Unknown);
}

*/

```

```

 * Convert the supplied exception to an EntityException, or if the
supplied
 * exception is already an EntityException, return it.
 *
 * @param exception
 * @param entityType
 * @param entity
 * @param actionType
 * @return
 */
public RuntimeException convertException(Exception exception, Class<?
> entityType,
Object entity, EntityExceptionActionType actionType) {
if (exception instanceof EntityException) {
return (EntityException) exception;
}

log.debug("converting exception: " + exception, exception);

// unwind nested exceptions with most specific on top
Stack<Throwable> causeStack = getCauseStack(exception);
while (!causeStack.empty()) {
Throwable thrown = causeStack.pop();

// look for a matching type up the hierarchy and through the
// interfaces
Class<?> currentEntityType = entityType;
if (currentEntityType != null) {
do {
List<Class<?>> entityTypes = new ArrayList<Class<?>>();
entityTypes.add(currentEntityType);
Collections.addAll(entityTypes,
currentEntityType.getInterfaces());
for (Class<?> thisEntityType : entityTypes) {
EntityExceptionAdapter adapter = exceptionMap.get(new
ExceptionMapKey(
thrown.getClass(), thisEntityType));
if (adapter != null) {
return adapter.convert(thrown, thisEntityType, entity,
actionType);
}
}
currentEntityType = currentEntityType.getSuperclass();
} while (currentEntityType != null);
} else {
EntityExceptionAdapter adapter = exceptionMap.get(new
ExceptionMapKey(thrown

```

```

        .getClass(), null));
if (adapter != null) {
return adapter.convert(thrown, null, entity, actionType);
}
}

// if the exception can't be converted, but is a runtime exception,
// return it
if (exception instanceof RuntimeException) {
return (RuntimeException) exception;
}
return EntityException.forUnknownProblem(exception, entityType,
entity, actionType);
}

private Stack<Throwable> getCauseStack(Throwable thrown) {
Stack<Throwable> causeStack = new Stack<Throwable>();
Throwable cause = thrown;
do {
causeStack.push(cause);
cause = cause.getCause();
} while (cause != null);
return causeStack;
}

private static class ExceptionMapKey {
private final Class<?> thrownType;
private final Class<?> entityType;

private ExceptionMapKey(Class<?> thrownType, Class<?> entityType) {
this.thrownType = thrownType;
this.entityType = entityType;
}

@Override
public int hashCode() {
final int prime = 31;
int result = 1;
result = prime * result + ((entityType == null) ? 0 :
entityType.hashCode());
result = prime * result + ((thrownType == null) ? 0 :
thrownType.hashCode());
return result;
}

@Override

```

```

public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final ExceptionMapKey other = (ExceptionMapKey) obj;
    if (entityType == null) {
        if (other.entityType != null) {
            return false;
        }
    } else if (!entityType.equals(other.entityType)) {
        return false;
    }
    if (thrownType == null) {
        if (other.thrownType != null) {
            return false;
        }
    } else if (!thrownType.equals(other.thrownType)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return getClass().getSimpleName() + ": thrown type = " + thrownType
+ " entity type = "
+ entityType;
}
}
}

```

exceptionmappingcommandhandler.java

```

/*
 * $Id: ExceptionMappingCommandHandler.java,v 1.2 2008/12/18 12:05:38
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

```

```

import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;

/**
 * A command handler that wraps another command handler catching
exceptions
 * thrown and converting them to application specific exceptions by a
 * customizable exception mapper.
 *
 * @author ron
 */
public class ExceptionMappingCommandHandler implements CommandHandler {
    private final ExceptionMapper exceptionMapper;
    private final CommandHandler commandHandler;

    /**
     * @param exceptionMapper
     * @param commandHandler
     */
    public ExceptionMappingCommandHandler(ExceptionMapper exceptionMapper,
        CommandHandler commandHandler) {
        this.exceptionMapper = exceptionMapper;
        this.commandHandler = commandHandler;
    }

    public <T extends Command> T execute(T command) throws Exception {
        try {
            return commandHandler.execute(command);
        } catch (Exception e) {
            throw exceptionMapper.convertException(e);
        }
    }
}

```

exportdictionarycommand.java

```

/*
 * $Id: 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.io.OutputStream;

```

```

import edu.harvard.fas.rregan.command.Command;

/**
 * Export a dictionary of categories, synsets, words, and senses
 * to an XML file.
 *
 * @author ron
 */
public interface ExportDictionaryCommand extends Command {

    public void setStartingFrom(String startingFrom);

    public void setEndingAt(String endingAt);

    public void setOutputStream(OutputStream outputStream);

}

```

exportdictionarycommandimpl.java

```

/*
 * $Id: ExportDictionaryCommandImpl.java,v 1.1 2008/12/13 00:39:56
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.Category;
import edu.harvard.fas.rregan.nlp.dictionary.Dictionary;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.Word;

```

```

import
edu.harvard.fas.rregan.nlp.dictionary.command.ExportDictionaryCommand;

/**
 * @author ron
 */
@Controller("exportDictionaryCommand")
@Scope("prototype")
public class ExportDictionaryCommandImpl extends
AbstractDictionaryCommand implements
ExportDictionaryCommand {

    /**
     * An array of classes to pass to JAXBContext.newInstance() that
     * includes
     * all the classes that are used in XmlElementRef annotations
     */
    public static final Class<?>[] CLASSES_FOR_JAXB;
    static {
        List<Class<?>> classes = new ArrayList<Class<?>>();
        classes.add(Dictionary.class);
        classes.add(Word.class);
        classes.add(Synset.class);
        classes.add(Category.class);
        classes.add(Sense.class);
        CLASSES_FOR_JAXB = classes.toArray(new Class<?>[classes.size()]);
    }

    private OutputStream outputStream;
    private String startingFrom = null;
    private String endingAt = null;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public ExportDictionaryCommandImpl(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }

    /*
     * (non-Javadoc)
     *
     * @see
     edu.harvard.fas.rregan.request.dictionary.ExportDictionaryCommand#setSt
artingFrom(java.lang.String)

```

```

/*
public void setStartingFrom(String startingFrom) {
    this.startingFrom = startingFrom;
}

/*
 * (non-Javadoc)
 *
 * @see
edu.harvard.fas.rregan.requel.dictionary.ExportDictionaryCommand#setEn
dingAt(java.lang.String)
 */
public void setEndingAt(String endingAt) {
    this.endingAt = endingAt;
}

/*
 * (non-Javadoc)
 *
 * @see
edu.harvard.fas.rregan.requel.dictionary.ExportDictionaryCommand#setOu
tputStream(java.io.OutputStream)
 */
public void setOutputStream(OutputStream outputStream) {
    this.outputStream = outputStream;
}

protected OutputStream getOutputStream() {
    return outputStream;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {
        JAXBContext context = JAXBContext.newInstance(CLASSES_FOR_JAXB);
        Marshaller marshaller = context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
        marshaller.marshal(getDictionaryRepository().getDictionary(starting
From, endingAt),
            getOutputStream());
    } catch (Exception e) {
        log.error(e, e);
    }
}

```

```

    }
}
```

exportprojectcommand.java

```

/*
 * $Id: ExportProjectCommand.java,v 1.2 2008/12/13 00:41:18 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import java.io.OutputStream;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.project.Project;

/**
 * Export the supplied project to the supplied stream.
 *
 * @author ron
 */
public interface ExportProjectCommand extends Command {

    /**
     * @param project -
     *          the project to export
     */
    public void setProject(Project project);

    /**
     * @param outputStream -
     *          the stream to write the output to.
     */
    public void setOutputStream(OutputStream outputStream);
}
```

exportprojectcommandimpl.java

```

/*
 * $Id: ExportProjectCommandImpl.java,v 1.12 2009/03/30 11:54:28
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
```

```

package edu.harvard.fas.rregan.requel.project.impl.command;

import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.impl.AbstractAnnotation;
import
edu.harvard.fas.rregan.requel.annotation.impl.AddWordToDictionaryPosit
ion;
import
edu.harvard.fas.rregan.requel.annotation.impl.ChangeSpellingPosition;
import
edu.harvard.fas.rregan.requel.annotation.impl.IssueImpl;
import
edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import
edu.harvard.fas.rregan.requel.annotation.impl.NoteImpl;
import
edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;
import
edu.harvard.fas.rregan.requel.project.DomainAdminUserRole;
import
edu.harvard.fas.rregan.requel.project.Project;
import
edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import
edu.harvard.fas.rregan.requel.project.command.ExportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.AddActorPosition;
import
edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import
edu.harvard.fas.rregan.requel.project.impl.ProjectImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import
edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import
edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.impl.OrganizationImpl;
import
edu.harvard.fas.rregan.requel.user.impl.UserImpl;

```

```

/**
 * @author ron
 */
@Controller("exportProjectCommand")
@Scope("prototype")
public class ExportProjectCommandImpl extends AbstractProjectCommand
implements
ExportProjectCommand {

/**
 * An array of classes to pass to JAXBContext.newInstance() that
includes
 * all the classes that are used in XmlElementRef annotations
 */
public static final Class<?>[] CLASSES_FOR_JAXB;
static {
List<Class<?>> classes = new ArrayList<Class<?>>();
classes.add(ProjectImpl.class); // project entites are found by
// reachability
classes.add(AbstractAnnotation.class);
classes.add(NoteImpl.class);
classes.add(IssueImpl.class);
classes.add(LexicalIssue.class);
classes.add(UserImpl.class);
classes.add(OrganizationImpl.class);
classes.add(SystemAdminUserRole.class);
classes.add(ProjectUserRole.class);
classes.add(DomainAdminUserRole.class);
classes.add(PositionImpl.class);
classes.add(ChangeSpellingPosition.class);
classes.add(AddWordToDictionaryPosition.class);
classes.add(AddGlossaryTermPosition.class);
classes.add(AddActorPosition.class);
CLASSES_FOR_JAXB = classes.toArray(new Class<?>[classes.size()]);
}

private Project project;
private OutputStream outputStream;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler

```

```

/*
@.Autowired
public ExportProjectCommandImpl(AssistantFacade assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
    projectCommandFactory,
    annotationCommandFactory, commandHandler);
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.ExportProjectCommand#set
Project(edu.harvard.fas.rregan.requel.project.Project)
 */
@Override
public void setProject(Project project) {
    this.project = project;
}

@Override
public void setOutputStream(OutputStream outputStream) {
    this.outputStream = outputStream;
}

protected OutputStream getOutputStream() {
    return outputStream;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {
        // NOTE: the classes referenced by Xml
        JAXBContext context = JAXBContext.newInstance(CLASSES_FOR_JAXB);
        Marshaller marshaller = context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
        marshaller.marshal(project, getOutputStream());
    } catch (Exception e) {
        log.error(e, e);
    }
}
}

```

```

}
```

generatereportcommand.java

```

/*
 * $Id: GenerateReportCommand.java,v 1.2 2008/12/13 00:41:18 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import java.io.OutputStream;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;

/**
 * Export the supplied project to the supplied stream.
 *
 * @author ron
 */
public interface GenerateReportCommand extends Command {

    /**
     * @param reportGenerator -
     *
     */
    public void setReportGenerator(ReportGenerator reportGenerator);

    /**
     * @param outputStream -
     *          the stream to write the generated report to.
     */
    public void setOutputStream(OutputStream outputStream);
}
```

generatereportcommandimpl.java

```

/*
 * $Id: GenerateReportCommandImpl.java,v 1.6 2009/03/30 11:54:30
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl.command;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;

import javax.xml.transform.ErrorListener;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import org.apache.xalan.processor.TransformerFactoryImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import org.xml.sax.InputSource;
import org.xml.sax.SAXNotSupportedException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.command.ExportProjectCommand;
import edu.harvard.fas.rregan.requel.project.command.GenerateReportCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * Generate a report for a project given the report generator and
 * output stream.
 */

```

```

/*
 * @author ron
 */
@Controller("exportProjectCommand")
@Scope("prototype")
public class GenerateReportCommandImpl extends AbstractProjectCommand
implements
GenerateReportCommand {

private ReportGenerator reportGenerator;
private OutputStream outputStream;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@Autowired
public GenerateReportCommandImpl(AssistantFacade assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.GenerateReportCommand#se
tReportGenerator(edu.harvard.fas.rregan.requel.project.ReportGenerator)
 */
@Override
public void setReportGenerator(ReportGenerator reportGenerator) {
this.reportGenerator = reportGenerator;
}

protected ReportGenerator getReportGenerator() {
return reportGenerator;
}

@Override

```

```

public void setOutputStream(OutputStream outputStream) {
    this.outputStream = outputStream;
}

protected OutputStream getOutputStream() {
    return outputStream;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {
        ReportGenerator reportGenerator =
getRepository().get(getReportGenerator());

        // export the project to a tmp file
        ExportProjectCommand exportCommand = getProjectCommandFactory()
            .newExportProjectCommand();
        exportCommand.setProject((Project)
reportGenerator.getProjectOrDomain());
        File tmpUpload = File.createTempFile("projectExport", ".xml");
        OutputStream projectOutputStream = new FileOutputStream(tmpUpload);
        exportCommand.setOutputStream(projectOutputStream);
        exportCommand = getCommandHandler().execute(exportCommand);
        projectOutputStream.close();

        InputStream projectInputStream = new FileInputStream(tmpUpload);
        InputStream xsltInputStream = new
ByteArrayInputStream(reportGenerator.getText()
    .getBytes());

        // create an XMLReader so we can set parsing features on and off
        XMLReader reader = XMLReaderFactory.createXMLReader();

        // enable validation if supported
        try {
            reader.setFeature("http://xml.org/sax/features/validation",
false);
            reader.setFeature("http://apache.org/xml/features/validation/schema",
false);
        } catch (SAXNotSupportedException e) {
            log.warn("The parser does not support XML validation.");
        }
        SAXSource saxSource = new SAXSource(reader, new
InputSource(projectInputStream));
    }
}

```

```

    Transformer transformer = new
TransformerFactoryImpl().newTransformer(new StreamSource(
    xsltInputStream));
    transformer.setErrorListener(new AnErrorListener());
    transformer.transform(saxSource, new StreamResult(new
BufferedWriter(
    new OutputStreamWriter(getOutputStream(), "UTF8"))));
} catch (Exception e) {
    log.error(e, e);
}
}

/**
 * Handle processing errors: throw an exception on error and
fatalError,
 * only log warnings.
 */
private class AnErrorListener implements ErrorListener {
    public void warning(TransformerException e) throws
TransformerException {
        // only log warnings
        log.warn("Parser warning: " + e.getMessage());
    }

    public void error(TransformerException e) throws
TransformerException {
        log.error("Parsing error: " + e.getMessage());
        throw e;
    }

    public void fatalError(TransformerException e) throws
TransformerException {
        log.error("Fatal parsing error: " + e.getMessage());
        throw e;
    }
}
}

```

genericpropertyvalueexceptionadapter.java

```

/*
 * $Id: GenericPropertyValueExceptionAdapter.java,v 1.2 2009/02/17
11:50:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.repository.jpa;

import org.hibernate.PropertyValueException;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.requel.EntityValidationException;

/**
 * @author ron
 */
public class GenericPropertyValueExceptionAdapter implements
EntityExceptionAdapter {

/**
 * @see
edu.harvard.fas.rregan.repository.EntityExceptionAdapter#convert(java.
lang.Throwable,
 *      java.lang.Object,
 *      edu.harvard.fas.rregan.repository.EntityExceptionActionType)
 */
@Override
public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
EntityExceptionActionType actionType) {
PropertyValueException pve = (PropertyValueException) original;
String propertyName = pve.getPropertyName();
if (original.getMessage().startsWith(
"not-null property references a null or transient value")) {
return EntityValidationException.emptyRequiredProperty(entityType,
entity,
propertyName, actionType);
} else {
return EntityException.forUnknownProblem(original, entityType,
entity, propertyName,
null, actionType);
}
}
}

```

glossaryterm.java

```

/*
 * $Id: GlossaryTerm.java,v 1.7 2008/11/20 09:55:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

/**
 * A GlossaryTerm is an entry in a glossary that defines a specific
term. The
 * "name" of the entry must be unique in the glossary.<br>
 * Multiple terms may refer to the same concept and are linked to a
canonical
 * term, which represents the primary term.
 *
 * @author ron
 */
public interface GlossaryTerm extends TextEntity {

/**
 * @return The term that is considered the primary/preferred way of
referencing this concept.
 */
public GlossaryTerm getCanonicalTerm();

/**
 * @return The set of terms that refer to this term as the canonical
term.
 */
public Set<GlossaryTerm> getAlternateTerms();

/**
 * @return project or domain entities that refer to this term, such
as goals
 */
public Set<ProjectOrDomainEntity> getReferers();
}
```

glossaryterm2glossarytermimpladapter.java

```

/*
 * $Id: GlossaryTerm2GlossaryTermImplAdapter.java,v 1.1 2008/08/13
01:29:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import javax.xml.bind.annotation.XmlTransient;
```

```

import javax.xml.bind.annotation.adapters.XmlAdapter;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
/**
 * Adapter for JAXB to convert interface GlossaryTerm to class
GlossaryTermImpl and back.
 *
 * @author ron
 */
@XmlTransient
public class GlossaryTerm2GlossaryTermImplAdapter extends
XmlAdapter<GlossaryTermImpl, GlossaryTerm> {

    @Override
    public GlossaryTermImpl marshal(GlossaryTerm glossaryTerm) throws
Exception {
        return (GlossaryTermImpl) glossaryTerm;
    }

    @Override
    public GlossaryTerm unmarshal(GlossaryTermImpl glossaryTerm) throws
Exception {
        return glossaryTerm;
    }
}

```

glossarytermeditorpanel.java

```

/*
 * $Id: GlossaryTermEditorPanel.java,v 1.15 2009/03/23 11:02:56 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;

```

```

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGlossaryTermComman
d;
import
edu.harvard.fas.rregan.requel.project.command.EditGlossaryTermCommand;
import
edu.harvard.fas.rregan.requel.project.command.ReplaceGlossaryTermComma
nd;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import edu.harvard.fas.rregan.uiframework.navigation.SelectorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.SelectorButton.Selection
IndicatorLabelAdapter;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * Editor for Glossary Terms. Allow editing the definition, select a
canonical
 * term and change all uses of this term in project entities to the
canonical
 * term.
 *
 * @author ron
 */
public class GlossaryTermEditorPanel extends
AbstractRequelProjectEditorPanel {

```

```

private static final Logger log =
Logger.getLogger(GlossaryTermEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the GlossaryTermEditorPanel.properties file to
set the
 * label of the name field. If the property is undefined "Term" is
used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the GlossaryTermEditorPanel.properties file to
set the
 * label of the text field. If the property is undefined
"Description" is
 * used.
 */
public static final String PROP_LABEL_TEXT = "Text.Label";

/**
 * The name to use in the GlossaryTermEditorPanel.properties file to
set the
 * label of the canonical term field. If the property is undefined
 * "Canonical Term" is used.
 */
public static final String PROP_LABEL_CANONICAL_TERM =
"CanonicalTerm.Label";

/**
 * The name to use in the GlossaryTermEditorPanel.properties file to
set the
 * label of the replace term function button. If the property is
undefined
 * "Replace Term" is used.
 */
public static final String PROP_LABEL_REPLACE_TERM_BUTTON =
"ReplaceTermButton.Label";

private UpdateListener updateListener;
private Button replaceTermButton;
private SelectorButton canonicalTermSelectorButton;

// this is set by the DeleteListener so that the UpdateListener can
ignore

```

```

// events between when the object was deleted and the panel goes
away.

private boolean deleted;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public GlossaryTermEditorPanel(CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
this(GlossaryTermEditorPanel.class.getName(), commandHandler,
projectCommandFactory,
projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public GlossaryTermEditorPanel(String resourceBundleName,
CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
super(resourceBundleName, GlossaryTerm.class, commandHandler,
projectCommandFactory,
projectRepository);
}

/**
 * If the editor is editing an existing goal the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Goal: {0}"<br>
 * Valid variables are:<br>
 * {0} - goal name<br>
 * {1} - project/domain name<br>
 * For new goal it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Goal"<br>

```

```

/*
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    if (getGlossaryTerm() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Term: {0}"));
        return MessageFormat.format(msgPattern,
            getGlossaryTerm().getName(),
            getProjectOrDomain().getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "New Term"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    GlossaryTerm glossaryTerm = getGlossaryTerm();
    if (glossaryTerm != null) {
        TextInput nameInput = addInput(EditGlossaryTermCommand.FIELD_NAME,
            PROP_LABEL_NAME,
            "Term", new TextField(), new
StringDocumentEx(glossaryTerm.getName()));
        nameInput.setEnabled(false);
        addInput(EditGlossaryTermCommand.FIELD_TEXT, PROP_LABEL_TEXT,
            "Description",
            new TextArea(), new StringDocumentEx(glossaryTerm.getText()));
        canonicalTermSelectorButton = addInput("canonicalTerm",
            PROP_LABEL_CANONICAL_TERM,
            "Canonical Term", new SelectorButton(
                new CanonicalTermSelectionIndicatorLabelAdapter(),
                GlossaryTerm.class,
                glossaryTerm.getCanonicalTerm(), Project.class, glossaryTerm
                    .getProjectOrDomain(),

```

```

ProjectManagementPanelNames.PROJECT_GLOSSARY_TERM_SELECTOR_PANE
L_NAME),
    glossaryTerm.getCanonicalTerm());
    addMultiRowInput("termContainers",
        GlossaryTermRefererTable.PROP_LABELGLOSSARYTERMCONTAINERS,
        "Referring Entities", new GlossaryTermRefererTable(this,
            getResourceBundleHelper(getLocale()), glossaryTerm);
    addMultiRowInput("annotations",
        AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelper(getLocale()),
            glossaryTerm);
    } else {
        addInput(EditGlossaryTermCommand.FIELD_NAME, PROP_LABEL_NAME,
            "Name", new TextField(),
            new StringDocumentEx());
        addInput(EditGlossaryTermCommand.FIELD_TEXT, PROP_LABEL_TEXT,
            "Description",
            new TextArea(), new StringDocumentEx());
        canonicalTermSelectorButton = addInput("canonicalTerm",
            PROP_LABEL_CANONICAL_TERM,
            "Canonical Term", new SelectorButton(
                new CanonicalTermSelectionIndicatorLabelAdapter(),
                GlossaryTerm.class,
                null, Project.class, getProjectOrDomain(),
                ProjectManagementPanelNames.PROJECT_GLOSSARY_TERM_SELECTOR_PANE
L_NAME),
            null);
        addMultiRowInput("termContainers",
            GlossaryTermRefererTable.PROP_LABELGLOSSARYTERMCONTAINERS,
            "Referring Entities", new GlossaryTermRefererTable(this,
                getResourceBundleHelper(getLocale()), null));
        addMultiRowInput("annotations",
            AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
            new AnnotationsTable(this, getResourceBundleHelper(getLocale()),
                null));
    }

    replaceTermButton = addActionButton(new
Button(getResourceBundleHelper(getLocale()))
    .getString(PROP_LABEL_REPLACE_TERM_BUTTON, "Replace Term")));
    replaceTermButton.addActionListener(new
ReplaceTermButtonListener(this));
    enableReplaceTermButton();

    canonicalTermSelectorButton.addChangeListener(new
CanonicalTermChangeListener(this));

```

```

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditGlossaryTermCommand command = getProjectCommandFactory()
            .newEditGlossaryTermCommand();
        command.setProjectOrDomain(getProjectOrDomain());
        command.setGlossaryTerm(getGlossaryTerm());
        command.setCanonicalTerm(getInputValue("canonicalTerm",
GlossaryTerm.class));
        command.setEditedBy(getCurrentUser());
        command.setName(getInputValue(EditGlossaryTermCommand.FIELD_NAME,
String.class));
        command.setText(getInputValue(EditGlossaryTermCommand.FIELD_TEXT,
String.class));
    }
}

```

```

command = getCommandHandler().execute(command);
setValid(true);
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
getEventDispatcher().dispatchEvent(
    new UpdateEntityEvent(this, command.getGlossaryTerm()));
enableReplaceTermButton();
} catch (EntityException e) {
    if (e.isStaleEntity()) {
        // TODO: compare the original values before the user edited
        // to the current revisions values and if they are the same
        // then update the new revision with the user's changes and
        // continue, otherwise show the new changed value vs. the users
        // new values.
        String newName = getInputValue(EditGlossaryTermCommand.FIELD_NAME,
String.class);
        String newText = getInputValue(EditGlossaryTermCommand.FIELD_TEXT,
String.class);
        GlossaryTerm newGlossaryTerm =
getProjectRepository().get(getGlossaryTerm());

        setTargetObject(newGlossaryTerm);
        if (!newName.equals(newGlossaryTerm.getName()))
            || !newText.equals(newGlossaryTerm.getText())) {
            setGeneralMessage("The term was changed by another user and the
value conflicts with your input.");
            if (!newName.equals(newGlossaryTerm.getName())) {
                setValidationMessage(EditGlossaryTermCommand.FIELD_NAME, "Your
input '"
                    + newName + "'");
                setInputValue(EditGlossaryTermCommand.FIELD_NAME,
newGlossaryTerm.getName());
            }
            if (!newText.equals(newGlossaryTerm.getText())) {
                setValidationMessage(EditGlossaryTermCommand.FIELD_TEXT, "Your
input '"
                    + newText + "'");
                setInputValue(EditGlossaryTermCommand.FIELD_TEXT,
newGlossaryTerm.getText());
            }
        } else {
            getEventDispatcher()
                .dispatchEvent(new UpdateEntityEvent(this, newGlossaryTerm));
        }
    }
}

```

```

} else if ((e.getEntityPropertyNames() != null)
    && (e.getEntityPropertyNames().length > 0)) {
    for (String propertyName : e.getEntityPropertyNames()) {
        setValidationMessage(propertyName, e.getMessage());
    }
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the goal: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
    try {
        DeleteGlossaryTermCommand deleteGlossaryTermCommand =
getProjectCommandFactory()
        .newDeleteGlossaryTermCommand();
        deleteGlossaryTermCommand.setEditedBy(getCurrentUser());
        deleteGlossaryTermCommand.setGlossaryTerm(getGlossaryTerm());
        deleteGlossaryTermCommand =
getCommandHandler().execute(deleteGlossaryTermCommand);
        deleted = true;
        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getGlossaryTerm()));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private void enableReplaceTermButton() {
    if (((getGlossaryTerm() != null)
        && (((getGlossaryTerm().getCanonicalTerm() != null) ||
(getInputValue(
        "canonicalTerm", GlossaryTerm.class) != null)) &&
(getGlossaryTerm()
        .getReferers().size() > 0))) {
        replaceTermButton.setEnabled(true);
    }
} else {
    replaceTermButton.setEnabled(false);
}
}

private ProjectOrDomain getProjectOrDomain() {
    if (getTargetObject() instanceof ProjectOrDomain) {
        return (ProjectOrDomain) getTargetObject();
    } else if (getTargetObject() instanceof ProjectOrDomainEntity) {
        return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
    }
    return null;
}

private GlossaryTerm getGlossaryTerm() {
    if (getTargetObject() instanceof GlossaryTerm) {
        return (GlossaryTerm) getTargetObject();
    }
    return null;
}

private static class CanonicalTermSelectionIndicatorLabelAdapter
implements
    SelectionIndicatorLabelAdapter {
    @Override
    public String getLabelString(Object selectedObject) {
        if (selectedObject != null) {
            return ((GlossaryTerm) selectedObject).getName();
        }
        return "<no term selected>";
    }
}

private static class CanonicalTermChangeListener implements
ChangeListener {
    static final long serialVersionUID = 0L;

    private final GlossaryTermEditorPanel panel;

    private CanonicalTermChangeListener(GlossaryTermEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void stateChanged(ChangeEvent e) {

```

```

        panel.enableReplaceTermButton();
    }

}

private static class ReplaceTermButtonListener implements
ActionListener {
    static final long serialVersionUID = 0L;

    private final GlossaryTermEditorPanel panel;

    private ReplaceTermButtonListener(GlossaryTermEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            // TODO: should this be done in a separate thread?
            ReplaceGlossaryTermCommand command =
panel.getProjectCommandFactory()
                .newReplaceGlossaryTermCommand();
            command.setEditedBy(panel.getCurrentUser());
            command.setGlossaryTerm(panel.getGlossaryTerm());
            if (panel.getGlossaryTerm().getCanonicalTerm() == null) {
                command.setCanonicalTerm(panel.getInputValue("canonicalTerm",
                    GlossaryTerm.class));
            }
            command = panel.getCommandHandler().execute(command);
            GlossaryTerm glossaryTerm = command.getGlossaryTerm();
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null, glossaryTerm));
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null,
                    glossaryTerm.getCanonicalTerm()));
            for (ProjectOrDomainEntity entity : command.getUpdatedEntities())
{
                panel.getEventDispatcher().dispatchEvent(
                    new UpdateEntityEvent(this, null, entity));
            }
        } catch (Exception ex) {
            panel.setGeneralMessage(ex.toString());
            log.warn(ex, ex);
        }
    }

    private static class UpdateListener implements ActionListener {

```

```

        static final long serialVersionUID = 0L;

        private final GlossaryTermEditorPanel panel;

        private UpdateListener(GlossaryTermEditorPanel panel) {
            this.panel = panel;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            if (panel.deleted) {
                return;
            }
            GlossaryTerm currentTerm = panel.getGlossaryTerm();
            if ((e instanceof UpdateEntityEvent) && (currentTerm != null)) {
                // TODO: the simple thing to do is just update the UI with the
                // existing object. It should be a proxy so the lastest values
                // will automatically get loaded. Although the cases such as
                // the ReplaceTermButtonListener above may send lots of messages
                // and this may refresh unnecessarily many times.

                UpdateEntityEvent event = (UpdateEntityEvent) e;
                GlossaryTerm updatedTerm = null;
                if (event.getObject() instanceof GlossaryTerm) {
                    updatedTerm = (GlossaryTerm) event.getObject();
                    if ((event instanceof DeletedEntityEvent) &&
currentTerm.equals(updatedTerm)) {
                        panel.deleted = true;
                        panel.getEventDispatcher().dispatchEvent(
                            new DeletedEntityEvent(this, panel, currentTerm));
                        return;
                    }
                } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                    // the entity may now refer to the term, so reload the term
                    updatedTerm = currentTerm;
                } else if (event.getObject() instanceof Annotation) {
                    // an issue related to the term may have changed
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    if (currentTerm.getAnnotations().contains(updatedAnnotation)) {
                        currentTerm.getAnnotations().remove(updatedAnnotation);
                    }
                    if (!(event instanceof DeletedEntityEvent)) {
                        currentTerm.getAnnotations().add(updatedAnnotation);
                    }
                    updatedTerm = currentTerm;
                }
            }
        }
    }
}

```

```
if ((updatedTerm != null) && updatedTerm.equals(currentTerm)) {  
    // TODO: check the input fields to see if the user has made  
    // a change before resetting the object and updating the  
    // input fields.  
    panel.setInputValue(EditGlossaryTermCommand.FIELD_NAME,  
updatedTerm.getName());  
    panel.setInputValue(EditGlossaryTermCommand.FIELD_TEXT,  
updatedTerm.getText());  
    panel.setInputValue("canonicalTerm",  
updatedTerm.getCanonicalTerm());  
    panel.setInputValue("termContainers", updatedTerm);  
    panel.setInputValue("annotations", updatedTerm);  
  
    panel.setTargetObject(updatedTerm);  
    panel.enableReplaceTermButton();  
}  
}  
}  
}
```

glossarytermimpl.java

```
/*
 * $Id: GlossaryTermImpl.java,v 1.17 2009/02/23 08:49:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlID;
```

```
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.AnyMetaDef;
import org.hibernate.annotations.ManyToOne;
import org.hibernate.annotations.MetaValue;
import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * Implementation of a Glossary Term.
 *
 * @author ron
 */
@Entity
@Table(name = "terms", uniqueConstraints =
{ @UniqueConstraint(columnNames = {
    "projectordomain_id", "name" }) })
@XmlRootElement(name = "term", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "term", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class GlossaryTermImpl extends AbstractTextEntity implements
GlossaryTerm,
    Comparable<GlossaryTerm> {
    static final long serialVersionUID = 0L;

    private GlossaryTerm canonicalTerm;
    private Set<GlossaryTerm> alternateTerms = new
TreeSet<GlossaryTerm>();
    private Set<ProjectOrDomainEntity> referers = new
TreeSet<ProjectOrDomainEntity>(
        new ProjectOrDomainEntityComparator());

    /**
     * @param projectOrDomain
     * @param name
```

```

 * @param createdBy
 */
public GlossaryTermImpl(ProjectOrDomain projectOrDomain, String name,
User createdBy) {
    setProjectOrDomain(projectOrDomain);
    setName(name);
    setCreatedBy(createdBy);
}

protected GlossaryTermImpl() {
    // for hibernate
}

@Override
@Column(nullable = false, unique = false)
@NotEmpty(message = "a unique name is required.")
@XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return super.getName();
}

// hack for JAXB to set the name, for some reason it won't use the
inherited
// method.
@Override
public void setName(String name) {
    super.setName(name);
}

@Transient
@XmlID
@XmlAttribute(name = "id")
public String getXmlId() {
    return "TRM_" + getId();
}

@Transient
@XmlTransient
@Override
public String getDescription() {
    return "Term: " + getName();
}

@XmlIDREF
@XmlAttribute(name = "canonicalTerm")
@XmlJavaTypeAdapter(GlossaryTerm2GlossaryTermImplAdapter.class)

```

```

    @ManyToOne(targetEntity = GlossaryTermImpl.class, cascade =
    CascadeType.REFRESH ), fetch = FetchType.EAGER, optional = true)
@Override
public GlossaryTerm getCanonicalTerm() {
    return canonicalTerm;
}

/**
 * @param canonicalTerm
 */
public void setCanonicalTerm(GlossaryTerm canonicalTerm) {
    this.canonicalTerm = canonicalTerm;
}

@XmlTransient
@Override
@OneToMany(targetEntity = GlossaryTermImpl.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.MERGE, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "canonicalTerm")
@Sort(type = SortType.NATURAL)
public Set<GlossaryTerm> getAlternateTerms() {
    return alternateTerms;
}

protected void setAlternateTerms(Set<GlossaryTerm> alternateTerms) {
    this.alternateTerms = alternateTerms;
}

@XmlTransient
@ManyToMany(fetch = FetchType.LAZY, metaColumn = @Column(name =
"referer_type", length = 255, nullable = false))
@AnyMetaDef(idType = "long", metaType = "string", metaValues = {
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Actor",
targetEntity = ActorImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Stakeholder", targetEntity =
StakeholderImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Goal",
targetEntity = GoalImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Scenario", targetEntity =
ScenarioImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Step",
targetEntity = StepImpl.class),

```

```

    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCase",
targetEntity = UseCaseImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Story",
targetEntity = StoryImpl.class) })
@JoinTable(name = "terms_referers", joinColumns = { @JoinColumn(name
= "term_id") }, inverseJoinColumns = {
    @JoinColumn(name = "referer_type"), @JoinColumn(name =
"referer_id") })
@Sort(type = SortType.COMPARATOR, comparator =
ProjectOrDomainEntityComparator.class)
public Set<ProjectOrDomainEntity> getReferers() {
    return referers;
}

protected void setReferers(Set<ProjectOrDomainEntity>
referersToThisTerm) {
    this.referers = referersToThisTerm;
}

/**
 * This is for JAXB to fixup the parent child relationship with the
 * canonical term.
 *
 * @see UnmarshallerListener
 */
@Override
public void afterUnmarshal() {
    if (getCanonicalTerm() != null) {
        getCanonicalTerm().getAlternateTerms().add(this);
    }
}

@Override
public int compareTo(GlossaryTerm o) {
    int projectCompare =
(getProjectOrDomain().getName().compareTo(o.getProjectOrDomain()
    .getName()));
    int nameCompare = (getName().compareTo(o.getName()));
    return (projectCompare != 0 ? projectCompare : nameCompare);
}
}

```

glossarytermnavigatorpanel.java

```
/*
```

```

 * $Id: GlossaryTermNavigatorPanel.java,v 1.2 2009/02/22 09:07:22
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * Panel for viewing and editing glossary terms for a project or
domain.
 *
 * @author ron
 */
public class GlossaryTermNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(GlossaryTermNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private GlossaryTermUpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the GlossaryTermNavigatorPanel.properties
to set
     * the label on the new GlossaryTerm button.
     */
    public static final String PROP_NEW_GLOSSARY_TERM_BUTTON_LABEL =
"NewGlossaryTermButton.Label";

    /**
     * Property name to use in the GlossaryTermNavigatorPanel.properties
to set
     * the label on the edit GlossaryTerm button in each row of the
table.
     */
    public static final String PROP_EDIT_GLOSSARY_TERM_BUTTON_LABEL =
>EditGlossaryTermButton.Label";

    /**
     * Property name to use in the GlossaryTermNavigatorPanel.properties
to set
     * the label on the view goal button in each row of the table when
the user
     * doesn't have edit permission.
     */
    public static final String PROP_VIEW_GLOSSARY_TERM_BUTTON_LABEL =
"ViewGlossaryTermButton.Label";

    /**
     */
    public GlossaryTermNavigatorPanel() {
        super(GlossaryTermNavigatorPanel.class.getName(), Project.class,
ProjectManagementPanelNames.PROJECT_GLOSSARY_TERMS_NAVIGATOR_PANEL
NAME);
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory()) {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
                String editGlossaryTermButtonLabel = null;
                if (isReadOnlyMode()) {
                    editGlossaryTermButtonLabel =
getResourceBundleHelper(getLocale())
.getString(PROP_VIEW_GLOSSARY_TERM_BUTTON_LABEL, "View");
                } else {
                    editGlossaryTermButtonLabel =
getResourceBundleHelper(getLocale())
.getString(PROP_EDIT_GLOSSARY_TERM_BUTTON_LABEL, "Edit");
                }

                NavigationEvent openGlossaryTermEditor = new
OpenPanelEvent(this,
                    PanelActionType.Editor, glossaryTerm, GlossaryTerm.class,
null,
                    WorkflowDisposition.NewFlow);
                NavigatorButton editGlossaryTermButton = new NavigatorButton(
editGlossaryTermButtonLabel, getEventDispatcher(),
openGlossaryTermEditor);
                editGlossaryTermButton.setStyleName(STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                editGlossaryTermButton.setLayoutData(rld);
            }
        });
    }
}

```

```

        return editGlossaryTermButton;
    }
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getName();
        }
    }));
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Definition",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getText();
        }
    }));
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Canonical Term",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getCanonicalTerm() == null ? "" :
glossaryTerm
                .getCanonicalTerm().getName();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            return entity.getCreatedBy() == null ? "" :
entity.getCreatedBy()
                .getUsername();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    }));
setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "GlossaryTerms: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "GlossaryTerms: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

```

```

}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
        Alignment.DEFAULT));

    String closeButtonLabel =
    getResourceBundleHelpergetLocale().getString(
        PROP_NEW_GLOSSARY_TERM_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
    getEventDispatcher(),
        closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);

    if (!isReadOnlyMode()) {
        String newGlossaryTermButtonLabel =
        getResourceBundleHelpergetLocale().getString(
            PROP_NEW_GLOSSARY_TERM_BUTTON_LABEL, "Add");
        NavigationEvent openGlossaryTermEditor = new OpenPanelEvent(this,
            PanelActionType.Editor, getProjectOrDomain(), GlossaryTerm.class,
        null,
            WorkflowDisposition.NewFlow);
        NavigatorButton newGlossaryTermButton = new
        NavigatorButton(newGlossaryTermButtonLabel,
            getEventDispatcher(), openGlossaryTermEditor);
        newGlossaryTermButton.setStyleName(STYLE_NAME_DEFAULT);
        buttonsWrapper.add(newGlossaryTermButton);
    }
}

```

```

add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
        updateListener);
    updateListener = new GlossaryTermUpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.class,
        updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(GlossaryTerm.class,
                StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(pod.getGlossaryTerms());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

private static class GlossaryTermUpdateListener implements
ActionListener {
    static final long serialVersionUID = 0L;

    private final GlossaryTermNavigatorPanel panel;
}
```

```
private GlossaryTermUpdateListener(GlossaryTermNavigatorPanel panel)
{
    this.panel = panel;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
        if (event.getObject() instanceof ProjectOrDomain) {
            updatedPod = (ProjectOrDomain) event.getObject();
        } else if (event.getObject() instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity) event.getObject();
            updatedPod = updatedEntity.getProjectOrDomain();
        } else if (event.getObject() instanceof Annotation) {
            if (!(event instanceof DeletedEntityEvent)) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                for (Annotatable annotatable :
                    updatedAnnotation.getAnnotatables()) {
                    if ((annotatable instanceof ProjectOrDomain)
                        && annotatable.equals(panel.getProjectOrDomain())) {
                        updatedPod = (ProjectOrDomain) annotatable;
                        break;
                    } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                        ProjectOrDomainEntity entity = (ProjectOrDomainEntity) annotatable;
                        if (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                            updatedPod = entity.getProjectOrDomain();
                            break;
                        }
                    }
                }
            }
        }
    }
    if (panel.getProjectOrDomain().equals(updatedPod)) {
        panel.setTargetObject(updatedPod);
    }
}
}
```

glossarytermreferertable.java

```
/*
 * $Id: GlossaryTermRefererTable.java,v 1.4 2009/01/08 06:48:45 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
```

```

import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A table of domain entities that refer to glossary terms.
 *
 * @author ron
 */
public class GlossaryTermRefererTable extends
AbstractRequestNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(GlossaryTermRefererTable.class,
            new GlossaryTermContainersTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
     * the
     * GlossaryTermContainersTable to define the label of the glossary
     * term
     * containers field. If the property is undefined the panel should
     * use a
     * sensible default such as "Glossary Term Referers".
     */
    public static final String PROP_LABELGLOSSARYTERMCONTAINERS =
"GlossaryTermReferers.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the view button in the glossary term containers edit table
     * column. If
     * the property is undefined "View" is used.
     */
    public static final String
PROPVIEWGLOSSARYTERMCONTAINERBUTTONLABEL =
"ViewGlossaryTermContainer.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     */
    * of the edit button in the glossary term container edit table
    column. If
    * the property is undefined "Edit" is used.
    */
    public static final String
PROP_EDITGLOSSARYTERMCONTAINERBUTTONLABEL =
>EditGlossaryTermContainer.Label";

    private GlossaryTerm glossaryTerm;
    private final NavigatorTable table;

    /**
     * @param editMode
     * @param resourceBundleHelper
     */
    public GlossaryTermRefererTable(EditMode editMode,
ResourceBundleHelper resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
        ColumnLayoutData layoutData = new ColumnLayoutData();
        layoutData.setAlignment(Alignment.ALIGN_CENTER);
        table = new NavigatorTable(getTableConfig());
        table.setLayoutData(layoutData);
        add(table);
    }

    protected GlossaryTerm getGlossaryTerm() {
        return glossaryTerm;
    }

    protected void setGlossaryTerm(GlossaryTerm glossaryTerm) {
        this.glossaryTerm = glossaryTerm;
        if (glossaryTerm != null) {
            table.setModel(new NavigatorTableModel((Collection)
glossaryTerm.getReferers()));
        } else {
            table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
        }
    }

    private NavigatorTableConfig getTableConfig() {
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {

```

```

ProjectOrDomainEntity glossaryTermReferer =
(ProjectOrDomainEntity) model
    .getBackingObject(row);
String buttonLabel = null;
if (isReadOnlyMode()) {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_VIEW_GLOSSARY_TERM_CONTAINER_BUTTON_LABEL, "View");
} else {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_EDIT_GLOSSARY_TERM_CONTAINER_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor, glossaryTermReferer,
glossaryTermReferer
    .getClass(), null, WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
});
});

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Description",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            ProjectOrDomainEntity glossaryTermReferer =
(ProjectOrDomainEntity) model
                .getBackingObject(row);
            return glossaryTermReferer.getDescription();
        }
    });
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            ProjectOrDomainEntity glossaryTermReferer =
(ProjectOrDomainEntity) model
                .getBackingObject(row);
            return glossaryTermReferer.getCreatedBy().getUsername();
        }
    }));
}

.tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            ProjectOrDomainEntity glossaryTermReferer =
(ProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            return formatter.format(glossaryTermReferer.getDateCreated());
        }
    }));
}

private static class GlossaryTermContainersTableManipulator extends
AbstractComponentManipulator {
protected GlossaryTermContainersTableManipulator() {
    super();
}

@Override
public Object getModel(Component component) {
    return getValue(component, GlossaryTerm.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
}
}

```

```

    return type.cast(getComponent(component).getGlossaryTerm());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setGlossaryTerm((GlossaryTerm) value);
}

private GlossaryTermRefererTable getComponent(Component component) {
    return (GlossaryTermRefererTable) component;
}
}
}

```

glossarytermselectorpanel.java

```

/*
 * $Id: GlossaryTermSelectorPanel.java,v 1.4 2009/03/27 13:43:14
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;

```

```

import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
;
import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class GlossaryTermSelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(GlossaryTermSelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the GlossaryTermSelectorPanel.properties
     * to set
     * the label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

```

```

    /**
     * @param projectRepository
     */
    public GlossaryTermSelectorPanel(ProjectRepository projectRepository)
    {
        super(GlossaryTermSelectorPanel.class.getName(), Project.class,
              ProjectManagementPanelNames.PROJECT_GLOSSARY_TERM_SELECTOR_PANEL_NAME);
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.setRowLevelSelection(true);

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Term",
                new NavigatorTableCellValueFactory() {
                    @Override
                    public Object getValueAt(NavigatorTableModel model, int column,
                                             int row) {
                        GlossaryTerm glossaryTerm = (GlossaryTerm)
                                model.getBackingObject(row);
                        return glossaryTerm.getName();
                    }
                }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Definition",
                new NavigatorTableCellValueFactory() {
                    @Override
                    public Object getValueAt(NavigatorTableModel model, int column,
                                             int row) {
                        GlossaryTerm glossaryTerm = (GlossaryTerm)
                                model.getBackingObject(row);
                        return glossaryTerm.getText();
                    }
                }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
                new NavigatorTableCellValueFactory() {
                    @Override
                    public Object getValueAt(NavigatorTableModel model, int column,
                                             int row) {
                        GlossaryTerm glossaryTerm = (GlossaryTerm)
                                model.getBackingObject(row);
                        return glossaryTerm.getCreatedBy().getUsername();
                    }
                }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
                new NavigatorTableCellValueFactory() {
                    @Override
                    public Object getValueAt(NavigatorTableModel model, int column,
                                             int row) {
                        GlossaryTerm glossaryTerm = (GlossaryTerm)
                                model.getBackingObject(row);
                        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
                        return format.format(glossaryTerm.getDateCreated());
                    }
                }));
    }

    setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
 * or
 * domain, by default the title is "Select Term"<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Select Term");
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

@Override

```

```

public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
                                             Alignment.DEFAULT));

    String closeButtonLabel =
        getResourceBundleHelper(getLocale()).getString(
            PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
        getEventDispatcher(),
        closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);
    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent
            .class,
            updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
        ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(GlossaryTerm.class,
                StakeholderPermissionType.Edit);
        }
    }
    return true;
}

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 */

```

```

     * @return an adapter to get the collection of items to select from
from the
     *      target object.
     */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
    return new NavigatorTableModelAdapter() {
        private ProjectOrDomain targetObject;

        @Override
        public Collection<Object> getCollection() {
            return (Collection) targetObject.getGlossaryTerms();
        }

        @Override
        public void setTargetObject(Object targetObject) {
            this.targetObject = (ProjectOrDomain) targetObject;
        }
    };
}

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the goal
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
        getTable().getSelectedObject(),
        getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GlossaryTermSelectorPanel panel;

    private UpdateListener(GlossaryTermSelectorPanel panel) {
        this.panel = panel;
    }
}

```

```

@Override
public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
        if (event.getObject() instanceof ProjectOrDomain) {
            updatedPod = (ProjectOrDomain) event.getObject();
        } else if (event.getObject() instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
                event.getObject();
            updatedPod = updatedEntity.getProjectOrDomain();
        } else if (event.getObject() instanceof Annotation) {
            if (!(event instanceof DeletedEntityEvent)) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                for (Annotatable annotatable :
                    updatedAnnotation.getAnnotatables()) {
                    if ((annotatable instanceof ProjectOrDomain)
                        && annotatable.equals(panel.getProjectOrDomain())))
                        {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        }
                    else if ((annotatable instanceof ProjectOrDomainEntity))
                        {
                            ProjectOrDomainEntity entity =
                                (ProjectOrDomainEntity)
                            annotatable;
                            if
                            (entity.getProjectOrDomain().equals(panel.getProjectOrDomain()))
                                {
                                    updatedPod = entity.getProjectOrDomain();
                                    break;
                                }
                            }
                        }
                    }
                if (panel.getProjectOrDomain().equals(updatedPod)) {
                    panel.setTargetObject(updatedPod);
                }
            }
        }
    }
}

```

glossarytermstable.java

```

/*
 * $Id: GlossaryTermsTable.java,v 1.1 2009/03/05 08:50:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

/*
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

```

```

/**
 * A table of domain entities that refer to glossary terms.
 *
 * @author ron
 */
public class GlossaryTermsTable extends AbstractRequestNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(GlossaryTermsTable.class,
            new GlossaryTermsTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
     * the
     * GlossaryTermsTable to define the label of the glossary term field.
     * If the
     * property is undefined the panel should use a sensible default such
     * as
     * "Glossary Terms".
     */
    public static final String PROP_LABELGLOSSARYTERM =
"GlossaryTerms.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the view button in the glossary term edit table column. If the
     * property is undefined "View" is used.
     */
    public static final String PROPVIEWGLOSSARYTERM_BUTTONLABEL =
"ViewGlossaryTerm.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the edit button in the glossary term edit table column. If the
     * property is undefined "Edit" is used.
     */
    public static final String PROPEditGLOSSARYTERM_BUTTONLABEL =
>EditGlossaryTerm.Label";

    private ProjectOrDomainEntity entity;
    private final NavigatorTable table;

    /**

```

```

        * @param editMode
        * @param resourceBundleHelper
        */
    public GlossaryTermsTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
        ColumnLayoutData layoutData = new ColumnLayoutData();
        layoutData.setAlignment(Alignment.ALIGN_CENTER);
        table = new NavigatorTable(getTableConfig());
        table.setLayoutData(layoutData);
        add(table);
    }

    protected ProjectOrDomainEntity getProjectOrDomainEntity() {
        return entity;
    }

    protected void setProjectOrDomainEntity(ProjectOrDomainEntity entity)
{
        this.entity = entity;
        if (entity != null) {
            table.setModel(new NavigatorTableModel((Collection)
entity.getGlossaryTerms()));
        } else {
            table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
        }
    }

    private NavigatorTableConfig getTableConfig() {
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
PROPVIEWGLOSSARYTERM_BUTTONLABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
PROPEditGLOSSARYTERM_BUTTONLABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,

```

```

        PanelActionType.Editor, glossaryTerm, glossaryTerm.getClass()
        null, WorkflowDisposition.NewFlow);
    NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
    openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
    RowLayoutData rld = new RowLayoutData();
    rld.setAlignment(Alignment.ALIGN_CENTER);
    openEditorButton.setLayoutData(rld);
    return openEditorButton;
}
}));}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Term",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getName();
        }
    }));
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Definition",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getText();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            return glossaryTerm.getCreatedBy().getUsername();
        }
    }));
}

```

```

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GlossaryTerm glossaryTerm = (GlossaryTerm)
model.getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            return formatter.format(glossaryTerm.getDateCreated());
        }
    }));
    return tableConfig;
}

private static class GlossaryTermsTableManipulator extends
AbstractComponentManipulator {

    protected GlossaryTermsTableManipulator() {
        super();
    }

    @Override
    public Object getModel(Component component) {
        return getValue(component, ProjectOrDomainEntity.class);
    }

    @Override
    public void setModel(Component component, Object valueModel) {
        setValue(component, valueModel);
    }

    @Override
    public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
        // nothing to do.
    }

    @Override
    public <T> T getValue(Component component, Class<T> type) {
        return
type.cast(getComponent(component).getProjectOrDomainEntity());
    }

    @Override
    public void setValue(Component component, Object value) {

```

```

    getComponent(component).setProjectOrDomainEntity((ProjectOrDomainEn
tity) value);
}

private GlossaryTermsTable getComponent(Component component) {
    return (GlossaryTermsTable) component;
}
}
}

```

goal.java

```

/*
 * $Id: Goal.java,v 1.11 2008/09/07 03:46:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

/**
 * A Project Entity representing desired properties, constraints,
features and
 * functions of the system being defined. Goals are primarily textual
with
 * relationships between them indicating a goal supports or conflicts
with
 * another goal.
 *
 * @author ron
 */
public interface Goal extends TextEntity, Comparable<Goal> {

    /**
     * The relationships that this goal has with other goals.
     *
     * @return The GoalRelations that originate from this goal
     */
    public Set<GoalRelation> getRelationsFromThisGoal();

    /**
     * The relationships that other goals have with this goal.
     *
     * @return The GoalRElations that terminate at this goal
     */
    public Set<GoalRelation> getRelationsToThisGoal();
}

```

```

    /**
     * Return the project entities that have direct associations to this
goal.
     *
     * @return
     */
    public Set<GoalContainer> getReferers();
}

```

goal2goalimpladapter.java

```

/*
 * $Id: Goal2GoalImplAdapter.java,v 1.2 2008/06/25 09:58:49 rregan Exp
 */
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import edu.harvard.fas.rregan.requel.project.Goal;

/**
 * Adapter for JAXB to convert interface Goal to class GoalImpl and
back.
 *
 * @author ron
 */
@XmlTransient
public class Goal2GoalImplAdapter extends XmlAdapter<GoalImpl, Goal> {

    @Override
    public GoalImpl marshal(Goal goal) throws Exception {
        return (GoalImpl) goal;
    }

    @Override
    public Goal unmarshal(GoalImpl goal) throws Exception {
        return goal;
    }
}

```

goalassistant.java

```
/*
 * $Id: GoalAssistant.java,v 1.31 2009/01/23 09:54:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses goals and adds annotations with suggestions.
 *
 * @author ron
 */
public class GoalAssistant extends TextEntityAssistant {

    /**
     * @param lexicalAssistant -
     *          assistant for analyzing text for spelling, terms and
     *          other
     *          word oriented analysis.
     * @param assistantUser -
     *          the user to use as the creator of the annotation
     *          entities.
     */
    public GoalAssistant(LexicalAssistant lexicalAssistant, User
assistantUser) {
        super(GoalAssistant.class.getName(), lexicalAssistant,
assistantUser);
    }

    /**
     * @param goal
     */
    @Override
    public void analyze() {
        super.analyze();
        if (getEntity() instanceof Goal) {
            analyzeGoalRelations((Goal) getEntity());
        }
    }
}
```

```
protected void analyzeGoalRelations(Goal goal) {
    for (GoalRelation goalRelation : goal.getRelationsFromThisGoal()) {
        ...
    }
}
```

goalcontainer.java

```
/*
 * $Id: GoalContainer.java,v 1.4 2008/09/01 08:40:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Comparator;
import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * A thing that can contain/refer to goals.
 *
 * @author ron
 */
public interface GoalContainer extends Describable, CreatedEntity {
    /**
     * The goals referenced.
     *
     * @return
     */
    public Set<Goal> getGoals();

    /**
     * Compare the objects that contain goals by the description.
     */
    public static final Comparator<GoalContainer> COMPARATOR = new
GoalContainerComparator();

    /**
     * A Comparator for collections of goal containers.
     */
    public static class GoalContainerComparator implements
Comparator<GoalContainer> {
        @Override

```

```

    public int compare(GoalContainer o1, GoalContainer o2) {
        return o1.getDescription().compareTo(o2.getDescription());
    }
}
}

```

goalcontainerstable.java

```

/*
 * $Id: GoalContainersTable.java,v 1.4 2009/01/08 06:48:45 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCellValueFactory;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColumnConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConfig;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * Table displaying the goal containers of a goal
 *
 * @author ron
 */
public class GoalContainersTable extends AbstractRequelNavigatorTable
{
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(GoalContainersTable.class,
            new GoalContainersTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
     * the
     * GoalContainersTable to define the label of the goal containers
     * field. If
     * the property is undefined the panel should use a sensible default
     * such as
     * "Goal Referers".
     */
    public static final String PROP_LABEL_GOAL_CONTAINERS =
"GoalContainers.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the view button in the goal containers edit table column. If
     * the
     * property is undefined "View" is used.
     */
}

```

```

public static final String PROP_VIEW_GOAL_CONTAINER_BUTTON_LABEL =
"ViewGoalContainer.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the edit button in the goal container edit table column. If the
 * property is undefined "Edit" is used.
 */
public static final String PROP_EDIT_GOAL_CONTAINER_BUTTON_LABEL =
>EditGoalContainer.Label";

private Goal goal;
private final NavigatorTable table;

/**
 * @param editMode
 * @param resourceBundleHelper
 */
public GoalContainersTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
    super(editMode, resourceBundleHelper);
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    table = new NavigatorTable(getTableConfig());
    table.setLayoutData(layoutData);
    add(table);
}

protected Goal getGoal() {
    return goal;
}

protected void setGoal(Goal goal) {
    this.goal = goal;
    if (goal != null) {
        table.setModel(new NavigatorTableModel((Collection)
goal.getReferers()));
    } else {
        table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    }
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",


```

```

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        GoalContainer goalContainer = (GoalContainer)
model.getBackingObject(row);
        String buttonLabel = null;
        if (isReadOnlyMode()) {
            buttonLabel = getResourceBundleHelpergetLocale().getString(
                PROP_VIEW_GOAL_CONTAINER_BUTTON_LABEL, "View");
        } else {
            buttonLabel = getResourceBundleHelpergetLocale().getString(
                PROP_EDIT_GOAL_CONTAINER_BUTTON_LABEL, "Edit");
        }
        NavigationEvent openEditorEvent = new OpenPanelEvent(this,
            PanelActionType.Editor, goalContainer,
            goalContainer.getClass(),
            null, WorkflowDisposition.NewFlow);
        NavigatorButton openEditorButton = new
        NavigatorButton(buttonLabel,
            getEventDispatcher(), openEditorEvent);
        openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
        RowLayoutData rld = new RowLayoutData();
        rld.setAlignment(Alignment.ALIGN_CENTER);
        openEditorButton.setLayoutData(rld);
        return openEditorButton;
    }
});

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Description",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GoalContainer goalContainer = (GoalContainer)
model.getBackingObject(row);
            return goalContainer.getDescription();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {


```

```

        GoalContainer goalContainer = (GoalContainer)
model.getBackingObject(row);
        return goalContainer.getCreatedBy().getUsername();
    }
}));}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            GoalContainer goalContainer = (GoalContainer)
model.getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            return formatter.format(goalContainer.getDateCreated());
        }
    });
}

return tableConfig;
}

private static class GoalContainersTableManipulator extends
AbstractComponentManipulator {

protected GoalContainersTableManipulator() {
    super();
}

@Override
public Object getModel(Component component) {
    return getValue(component, Goal.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {

```

```

        return type.cast(getComponent(component).getGoal());
    }

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setGoal((Goal) value);
}

private GoalContainersTable getComponent(Component component) {
    return (GoalContainersTable) component;
}
}

goaleditorpanel.java

/*
 * $Id: GoalEditorPanel.java,v 1.20 2009/03/23 11:02:56 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.TreeSet;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
```

```

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.CopyGoalCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGoalCommand;
import edu.harvard.fas.rregan.requel.project.command.EditGoalCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */

```

```

public class GoalEditorPanel extends AbstractRequelProjectEditorPanel
{
    private static final Logger log =
Logger.getLogger(GoalEditorPanel.class);

    static final long serialVersionUID = 0L;

    /**
     * The name to use in the GoalEditorPanel.properties file to set the
     * label
     * of the name field. If the property is undefined "Name" is used.
     */
    public static final String PROP_LABEL_NAME = "Name.Label";

    /**
     * The name to use in the GoalEditorPanel.properties file to set the
     * label
     * of the text field. If the property is undefined "Text" is used.
     */
    public static final String PROP_LABEL_TEXT = "Text.Label";

    /**
     * The name to use in the GoalEditorPanel.properties file to set the
     * label
     * of the goal relations field. If the property is undefined
     * "Relations" is
     * used.
     */
    public static final String PROP_LABEL_RELATIONS = "Relations.Label";

    /**
     * The name to use in the GoalEditorPanel.properties file to set the
     * label
     * of the view button in the goal relation edit table column. If the
     * property is undefined "View" is used.
     */
    public static final String PROP_VIEW_GOAL_RELATION_BUTTON_LABEL =
"ViewGoalRelation.Label";

    /**
     * The name to use in the GoalEditorPanel.properties file to set the
     * label
     * of the edit button in the goal relation edit table column. If the
     * property is undefined "Edit" is used.
     */
    public static final String PROP_EDIT_GOAL_RELATION_BUTTON_LABEL =
>EditGoalRelation.Label";

```

```

/**
 * The name to use in the GoalEditorPanel.properties file to set the
label
 * of the add button in the goal relation editor. If the property is
 * undefined "Add" is used.
 */
public static final String PROP_ADD_GOAL_RELATION_BUTTON_LABEL =
"AddGoalRelation.Label";

private UpdateListener updateListener;
private Button copyButton;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public GoalEditorPanel(CommandHandler commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    this(GoalEditorPanel.class.getName(), commandHandler,
    projectCommandFactory,
    projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public GoalEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    super(resourceBundleName, Goal.class, commandHandler,
    projectCommandFactory,
    projectRepository);
}

```

```

/**
 * If the editor is editing an existing goal the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Goal: {0}"<br>
 * Valid variables are:<br>
 * {0} - goal name<br>
 * {1} - project/domain name<br>
 * For new goal it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Goal"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    if (getGoal() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Goal: {0}"));
        return MessageFormat.format(msgPattern, getGoal().getName(),
getProjectOrDomain()
        .getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"New Goal"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    Goal goal = getGoal();
    if (goal != null) {

```

```

        addInput(EditGoalCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
TextField(),
        new StringDocumentEx(goal.getName()));
        addInput(EditGoalCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text", new
TextArea(),
        new StringDocumentEx(goal.getText()));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABEL_GLOSSARY_TERM,
        "Glossary Terms", new GlossaryTermsTable(this,
        getResourceBundleHelper(getLocale()), goal);
        addMultiRowInput("goalContainers",
GoalContainersTable.PROP_LABEL_GOAL_CONTAINERS,
        "Referring Entities", new GoalContainersTable(this,
        getResourceBundleHelper(getLocale()), goal);
        addMultiRowInput("relations", PROP_LABEL_RELATIONS, "Relations To
Other Goals",
        getRelationsTable(), new NavigatorTableModel(Collection) goal
        .getRelationsFromThisGoal());
        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelper(getLocale())),
goal);

        copyButton = addActionButton(new
Button(getResourceBundleHelper(getLocale()).getString(
        PROP_LABEL_COPY_BUTTON, "Copy"));
        copyButton.addActionListener(new CopyListener(this));
        copyButton.setEnabled(!isReadOnlyMode());
    } else {
        addInput(EditGoalCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
TextField(),
        new StringDocumentEx());
        addInput(EditGoalCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text", new
TextArea(),
        new StringDocumentEx());
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABEL_GLOSSARY_TERM,
        "Glossary Terms", new GlossaryTermsTable(this,
        getResourceBundleHelper(getLocale()), goal);
        addMultiRowInput("goalContainers",
GoalContainersTable.PROP_LABEL_GOAL_CONTAINERS,
        "Referring Entities", new GoalContainersTable(this,
        getResourceBundleHelper(getLocale()), null);
        addMultiRowInput("relations", PROP_LABEL_RELATIONS, "Relations",
getRelationsTable(),
        new NavigatorTableModel(new TreeSet<Object>()));

```

```

        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelper(getLocale())),
null);
    }

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
    updateListener = null;
}

private Component getRelationsTable() {
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    Column col = new Column();
    NavigatorTable relationTable = new
NavigatorTable(getGoalRelationTableConfig());
    relationTable.setLayoutData(layoutData);
    col.add(relationTable);
    if ((getGoal() != null) && !isReadOnlyMode()) {
        String buttonLabel = null;
        buttonLabel = getResourceBundleHelper(getLocale()).getString(
            PROP_ADD_GOAL_RELATION_BUTTON_LABEL, "Add");
        NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor,
            getGoal(), GoalRelation.class, null,
WorkflowDisposition.NewFlow);
        NavigatorButton openEditorButton = new NavigatorButton(buttonLabel,
            getEventDispatcher(), openEditorEvent);
        openEditorButton.setStyleName(STYLE_NAME_DEFAULT);
    }
}

```

```

openEditorButton.setLayoutData(layoutData);
col.add(openEditorButton);
}
return col;
}

private NavigatorTableConfig getGoalRelationTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GoalRelation goalRelation = (GoalRelation)
model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_VIEW_GOAL_RELATION_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_EDIT_GOAL_RELATION_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                    PanelActionType.Editor, goalRelation, GoalRelation.class,
null,
                    WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                openEditorButton.setLayoutData(rld);
                return openEditorButton;
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Relation
Type",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GoalRelation goalRelation = (GoalRelation)
model.getBackingObject(row);
                return goalRelation.getRelationType().toString();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("To
Goal",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GoalRelation goalRelation = (GoalRelation)
model.getBackingObject(row);
                return goalRelation.getToGoal().getName();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GoalRelation goalRelation = (GoalRelation)
model.getBackingObject(row);
                return goalRelation.getCreatedBy().getUsername();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                GoalRelation goalRelation = (GoalRelation)
model.getBackingObject(row);
                DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                return formatter.format(goalRelation.getDateCreated());
            }
        }));
}

return tableConfig;
}

@Override
public void cancel() {
    super.cancel();
}

```

```

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}

@Override
public void save() {
    try {
        super.save();
        EditGoalCommand command =
getProjectCommandFactory().newEditGoalCommand();
        command.setGoal(getGoal());
        command.setGoalContainer(getGoalContainer());
        command.setEditedBy(getCurrentUser());
        command.setName(getInputValue(EditGoalCommand.FIELD_NAME,
String.class));
        command.setText(getInputValue(EditGoalCommand.FIELD_TEXT,
String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
            updateListener);
        }
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getGoal()));
    } catch (EntityException e) {
        if (e.isStaleEntity()) {
            // TODO: compare the original values before the user edited
            // to the current revisions values and if they are the same
            // then update the new revision with the user's changes and
            // continue, otherwise show the new changed value vs. the users
            // new values.
            String newName = getInputValue(EditGoalCommand.FIELD_NAME,
String.class);
            String newText = getInputValue(EditGoalCommand.FIELD_TEXT,
String.class);
            Goal newGoal = getProjectRepository().get(getGoal());

            setTargetObject(newGoal);
            if (!newName.equals(newGoal.getName()) || !
newText.equals(newGoal.getText())) {
                setGeneralMessage("The goal was changed by another user and the
value conflicts with your input.");
        }
    }
}

```

```

if (!newName.equals(newGoal.getName())) {
    setValidationMessage(EditGoalCommand.FIELD_NAME, "Your input '" +
newName
        + "'");
    setInputValue(EditGoalCommand.FIELD_NAME, newGoal.getName());
}
if (!newText.equals(newGoal.getText())) {
    setValidationMessage(EditGoalCommand.FIELD_TEXT, "Your input '" +
newText
        + "'");
    setInputValue(EditGoalCommand.FIELD_TEXT, newGoal.getText());
}
} else {
    getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
newGoal));
}

} else if ((e.getEntityPropertyNames() != null)
    && (e.getEntityPropertyNames().length > 0)) {
    for (String propertyName : e.getEntityPropertyNames()) {
        setValidationMessage(propertyName, e.getMessage());
    }
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the goal: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
    try {
        DeleteGoalCommand deleteGoalCommand =
getProjectCommandFactory().newDeleteGoalCommand();
        deleteGoalCommand.setEditedBy(getCurrentUser());
        deleteGoalCommand.setGoal(getGoal());
        deleteGoalCommand = getCommandHandler().execute(deleteGoalCommand);
        deleted = true;
    }
}

```

```

        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getGoal()));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private ProjectOrDomain getProjectOrDomain() {
    if (getTargetObject() instanceof ProjectOrDomain) {
        return (ProjectOrDomain) getTargetObject();
    } else if (getTargetObject() instanceof ProjectOrDomainEntity) {
        return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
    }
    return null;
}

private GoalContainer getGoalContainer() {
    if (getTargetObject() instanceof GoalContainer) {
        return (GoalContainer) getTargetObject();
    }
    return null;
}

private Goal getGoal() {
    if (getTargetObject() instanceof Goal) {
        return (Goal) getTargetObject();
    }
    return null;
}

private static class CopyListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GoalEditorPanel panel;

    private CopyListener(GoalEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            CopyGoalCommand copyGoalCommand = panel.getProjectCommandFactory()
                .newCopyGoalCommand();
            copyGoalCommand.setEditedBy(panel.getCurrentUser());
            copyGoalCommand.setOriginalGoal(panel.getGoal());
            copyGoalCommand =
panel.getCommandHandler().execute(copyGoalCommand);
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null,
copyGoalCommand.getNewGoal()));
            panel.getEventDispatcher().dispatchEvent(
                new OpenPanelEvent(this, PanelActionType.Editor, copyGoalCommand
                    .getNewGoal(), Goal.class, null));
        } catch (Exception e) {
            panel.setGeneralMessage("Could not copy entity: " + e);
        }
    }
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GoalEditorPanel panel;

    private UpdateListener(GoalEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        Goal existingGoal = panel.getGoal();
        if ((e instanceof UpdateEntityEvent) && (existingGoal != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            Goal updatedGoal = null;
            if (event.getObject() instanceof Goal) {
                updatedGoal = (Goal) event.getObject();
                if ((event instanceof DeletedEntityEvent) &&
existingGoal.equals(updatedGoal)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingGoal));
                    return;
                }
            } else if (event.getObject() instanceof GoalRelation) {
                GoalRelation updatedGoalRelation = (GoalRelation)
event.getObject();
                updatedGoal = updatedGoalRelation.getFromGoal();
            } else if (event.getObject() instanceof Annotation) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
            }
        }
    }
}

```

```

if (event instanceof DeletedEntityEvent) {
    if (existingGoal.getAnnotations().contains(updatedAnnotation)) {
        existingGoal.getAnnotations().remove(updatedAnnotation);
        updatedGoal = existingGoal;
    }
} else if
(updatedAnnotation.getAnnotatables().contains(existingGoal)) {
    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
        if (annotatable.equals(existingGoal)) {
            updatedGoal = (Goal) annotatable;
            break;
        }
    }
}
if ((updatedGoal != null) && updatedGoal.equals(existingGoal)) {
    // TODO: check the input fields to see if the user has made
    // a change before resetting the object and updating the
    // input fields.
    panel.setInputValue(EditGoalCommand.FIELD_NAME,
updatedGoal.getName());
    panel.setInputValue(EditGoalCommand.FIELD_TEXT,
updatedGoal.getText());
    panel.setInputValue("glossaryTerms", updatedGoal);
    panel.setInputValue("goalContainers", updatedGoal);
    panel.setInputValue("relations",
updatedGoal.getRelationsFromThisGoal());
    panel.setInputValue("annotations", updatedGoal);
    panel.setTargetObject(updatedGoal);
}
}
}

goalimpl.java

/*
 * $Id: GoalImpl.java,v 1.28 2009/02/12 11:01:35 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.annotations.AnyMetaDef;
import org.hibernate.annotations.ManyToOne;
import org.hibernate.annotations.MetaValue;
import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Entity
@Table(name = "goals", uniqueConstraints =
{ @UniqueConstraint(columnNames =
{ "projectordomain_id", "name" }) })
@XmlRootElement(name = "goal", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "goal", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class GoalImpl extends AbstractTextEntity implements Goal {
    static final long serialVersionUID = 0L;
}

```

```

private Set<GoalRelation> relationsFromThisGoal = new
TreeSet<GoalRelation>();
private Set<GoalRelation> relationsToThisGoal = new
TreeSet<GoalRelation>();
private Set<GoalContainer> referersToThisGoal = new
TreeSet<GoalContainer>(
    GoalContainer.COMPARATOR);

/**
 * @param projectOrDomain
 * @param name
 * @param createdBy
 * @param text
 */
public GoalImpl(ProjectOrDomain projectOrDomain, User createdBy,
String name, String text) {
    super(projectOrDomain, createdBy, name, text);
    // add to collection last so that sorting in the collection by
entity
    // properties has access to all the properties.
    projectOrDomain.getGoals().add(this);
}

protected GoalImpl() {
    // for hibernate
}

@Override
@Column(nullable = false, unique = false)
@NotEmpty(message = "a unique name is required.")
@XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return super.getName();
}

// hack for JAXB to set the name, for some reason it won't use the
inherited
// method.
@Override
public void setName(String name) {
    super.setName(name);
}

@Transient
@XmlID
@XmlAttribute(name = "id")

```

```

public String getXmlId() {
    return "GOL_" + getId();
}

@Transient
public String getDescription() {
    return "Goal: " + getName();
}

@XmlElementWrapper(name = "goalRelations", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = GoalRelationImpl.class)
@OneToMany(targetEntity = GoalRelationImpl.class, cascade =
{ CascadeType.REFRESH }, fetch = FetchType.LAZY, mappedBy =
"fromGoalInternal")
@JoinColumn(name = "fromGoalInternal_id")
public Set<GoalRelation> getRelationsFromThisGoal() {
    return relationsFromThisGoal;
}

protected void setRelationsFromThisGoal(Set<GoalRelation>
relationsFromThisGoal) {
    this.relationsFromThisGoal = relationsFromThisGoal;
}

@XmlTransient
@OneToMany(targetEntity = GoalRelationImpl.class, cascade =
{ CascadeType.ALL }, fetch = FetchType.LAZY, mappedBy =
"toGoalInternal")
@JoinColumn(name = "toGoalInternal_id")
public Set<GoalRelation> getRelationsToThisGoal() {
    return relationsToThisGoal;
}

protected void setRelationsToThisGoal(Set<GoalRelation>
relationsToThisGoal) {
    this.relationsToThisGoal = relationsToThisGoal;
}

@XmlTransient
@ManyToMany(fetch = FetchType.LAZY, metaColumn = @Column(name =
"goalcontainer_type", length = 255, nullable = false))
@AnyMetaDef(idType = "long", metaType = "string", metaValues = {
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCase",
targetEntity = UseCaseImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Requirement",
targetEntity = RequirementImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Stakeholder",
targetEntity = StakeholderImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCaseRequirement",
targetEntity = UseCaseRequirementImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.StakeholderRequirement",
targetEntity = StakeholderRequirementImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.StakeholderUseCase",
targetEntity = StakeholderUseCaseImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.StakeholderRequirementUseCase",
targetEntity = StakeholderRequirementUseCaseImpl.class)
})

```

```

    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Scenario", targetEntity =
ScenarioImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Story",
targetEntity = StoryImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Actor",
targetEntity = ActorImpl.class),
    @MetaValue(value =
"edu.harvard.fas.rregan.requel.project.Stakeholder", targetEntity =
StakeholderImpl.class))
    @JoinTable(name = "goals_goalcontainers", joinColumns =
{ @JoinColumn(name = "goal_id") }, inverseJoinColumns = {
    @JoinColumn(name = "goalcontainer_type"), @JoinColumn(name =
"goalcontainer_id") })
    @Sort(type = SortType.COMPARATOR, comparator =
GoalContainer.GeoalContainerComparator.class)
    public Set<GoalContainer> getReferers() {
        return referersToThisGoal;
    }

    protected void setReferers(Set<GoalContainer> referersToThisGoal) {
        this.referersToThisGoal = referersToThisGoal;
    }

    @Override
    public int compareTo(Goal o) {
        return getName().compareToIgnoreCase(o.getName());
    }
}

```

goalnameinuseexception.java

```

/*
 * $Id: GoalNameInUseException.java,v 1.3 2008/12/13 00:40:01 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.exception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;

/**
 * @author ron

```

```

    /*
    public class GoalNameInUseException extends EntityException {
        static final long serialVersionUID = 0;

        protected static String MSG_FOR_NAME = "A goal already exists with
name '%s' for project '%s';

        /**
         * @param project -
         *          the project the goal name already exists in
         * @param name -
         *          the name used to find a project that doesn't exist
         * @return
         */
        public static GoalNameInUseException forName(Project project, String
name) {
            return new GoalNameInUseException(Goal.class, null, "name", name,
                EntityExceptionActionType.Creating, MSG_FOR_NAME, name,
                project.getName());
        }

        protected GoalNameInUseException(Class<?> entityType, Object entity,
String entityPropertyName,
            Object entityValue, EntityExceptionActionType actionType, String
format,
            Object... messageArgs) {
            super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
        }

        protected GoalNameInUseException(Throwable cause, Class<?>
entityType, Object entity,
            String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
            String format, Object... messageArgs) {
            super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
                messageArgs);
        }
    }
}

```

goalnavigatorpanel.java

```

/*
 * $Id: GoalNavigatorPanel.java,v 1.2 2009/02/23 07:37:22 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
e;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * TODO: integrate the GoalsTable with this.
 *
 * @author ron
 */
public class GoalNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(GoalNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private UpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the GoalNavigatorPanel.properties to set
     * the
     * * label on the new goal button.
     */
    public static final String PROP_NEW_GOAL_BUTTON_LABEL =
"NewGoalButton.Label";

    /**
     * Property name to use in the GoalNavigatorPanel.properties to set
     * the
     * * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * Property name to use in the GoalNavigatorPanel.properties to set
     * the
     * * label on the edit goal button in each row of the table.
     */

```

```

public static final String PROP_EDIT_GOAL_BUTTON_LABEL =
>EditGoalButton.Label;

/**
 * Property name to use in the GoalNavigatorPanel.properties to set
the
 * label on the view goal button in each row of the table when the
user
 * doesn't have edit permission.
 */
public static final String PROP_VIEW_GOAL_BUTTON_LABEL =
"ViewGoalButton.Label";

/**
 */
public GoalNavigatorPanel() {
super(GoalNavigatorPanel.class.getName(), Project.class,
ProjectManagementPanelNames.PROJECT_GOALS_NAVIGATOR_PANEL_NAME);
NavigatorTableConfig tableConfig = new NavigatorTableConfig();

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column, int row) {
Goal goal = (Goal) model.getBackingObject(row);
String buttonLabel = null;
if (isReadOnlyMode()) {
buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_VIEW_GOAL_BUTTON_LABEL, "View");
} else {
buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_EDIT_GOAL_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor, goal, Goal.class, null,
WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
}
});
}

```

```

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column, int row) {
Goal goal = (Goal) model.getBackingObject(row);
return goal.getName();
}
}));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By", new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column, int row) {
AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
.getBackingObject(row);
return entity.getCreatedBy().getUsername();
}
}));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created", new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column, int row) {
AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
.getBackingObject(row);
DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
return format.format(entity.getDateCreated());
}
}));

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "Goals: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>

```

```

/*
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "Goals: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
    Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
}

```

```

buttonsWrapper.add(closeButton);

if (!isReadOnlyMode()) {
    String newGoalButtonLabel =
getResourceBundleHelper(getLocale()).getString(
        PROP_NEW_GOAL_BUTTON_LABEL, "Add");
    NavigationEvent openGoalEditor = new OpenPanelEvent(this,
    PanelActionType.Editor,
        getProjectOrDomain(), Goal.class, null,
    WorkflowDisposition.NewFlow);
    NavigatorButton newGoalButton = new
NavigatorButton(newGoalButtonLabel,
        getEventDispatcher(), openGoalEditor);
    newGoalButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(newGoalButton);
}

add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Goal.class,
    StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(((ProjectOrDomain) targetObject).getGoals());
    }
}

```

```

    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GoalNavigatorPanel panel;

    private UpdateListener(GoalNavigatorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
&& annotatable.equals(panel.getProjectOrDomain())) {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                            if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain()))
                                updatedPod = entity.getProjectOrDomain();
                            break;
                        }
                    }
                }
            }
        }
    }
}

```

```
        }
    }
}
if (panel.getProjectOrDomain().equals(updatedPod)) {
    panel.setTargetObject(updatedPod);
}
}
```

goalrelation.java

```
/*
 * $Id: GoalRelation.java,v 1.3 2008/11/06 00:52:48 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;

/**
 * A GoalRelation is a uni-directional relationship from one goal to
another. A
 * goal may only have one relationship to another goal. Supported
relationships
 * are defined by the GoalRelationshipType.
 *
 * @author ron
 */
public interface GoalRelation extends Annotatable,
Comparable<GoalRelation>, CreatedEntity, Describable {

 /**
 * @return The goal that is the origin of the relationship.
 */
 public Goal getFromGoal();

 /**
 * @return The goal that is the target of the relationship.
 */
 public Goal getToGoal();

 /**
```

```

 * @return The type of relationship that the from goal has with the
to goal.
 */
public GoalRelationType getRelationType();
}

```

goalrelationeditorpanel.java

```

/*
 * $Id: GoalRelationEditorPanel.java,v 1.17 2009/02/22 09:07:21 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.GoalRelationType;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGoalRelationCommand;

```

```

import
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import
edu.harvard.fas.rregan.requel.user.User;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * @author ron
 */
public class GoalRelationEditorPanel extends
AbstractRequelProjectEditorPanel {
private static final Logger log =
Logger.getLogger(GoalRelationEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the GoalRelationEditorPanel.properties file to
set the
 * label of the from goal field. If the property is undefined "From
Goal" is
 * used.
 */
public static final String PROP_LABEL_FROM_GOAL = "FromGoal.Label";

/**
 * The name to use in the GoalRelationEditorPanel.properties file to
set the
 * label of the to goal field. If the property is undefined "To Goal"
is
 * used.
 */
public static final String PROP_LABEL_TO_GOAL = "ToGoal.Label";

```

```

    }

    /**
     * @return the value of Panel.PROP_PANEL_TITLE or "Goal Relation" if
     * the
     * title isn't defined
     * @see Panel.PROP_PANEL_TITLE
     * @see
     edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
     */
    @Override
    public String getTitle() {
        // return
        // getResourceBundleHelpergetLocale().getString(PROP_PANEL_TITLE,
        // "Goal Relation");
        if ((getGoalRelation() != null) && (getGoalRelation().getToGoal() != null)) {
            String msgPattern = getResourceBundleHelpergetLocale().getString(
                PROP_EXISTING_OBJECT_PANEL_TITLE,
                getResourceBundleHelpergetLocale().getString(PROP_PANEL_TITLE,
                    "Goal Relation From: {0}"));
            return MessageFormat.format(msgPattern,
                getGoalRelation().getToGoal().getName(),
                getProjectOrDomain().getName());
        } else {
            String msg = getResourceBundleHelpergetLocale().getString(
                PROP_NEW_OBJECT_PANEL_TITLE,
                getResourceBundleHelpergetLocale().getString(PROP_PANEL_TITLE,
                    "New Goal Relation"));
            return msg;
        }
    }

    @Override
    public void setup() {
        super.setup();
        GoalRelation goalRelation = getGoalRelation();
        if (goalRelation != null) {
            addInput("fromGoal", PROP_LABEL_FROM_GOAL, "From Goal", new
SelectField(),
            new CombinedListModel(getGoalNames(),
goalRelation.getFromGoal().getName(),
            true));
            addInput("toGoal", PROP_LABEL_TO_GOAL, "To Goal", new
SelectField(),
            new CombinedListModel(getGoalNames(),
goalRelation.getToGoal().getName(), true));
        }
    }
}

```

```

        addInput("relationType", PROP_LABEL_RELATION_TYPE, "Relation", new
SelectField(),
        new CombinedListModel(getGoalRelationTypeNames(), goalRelation
            .getRelationType().toString(), true));
    addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelpergetLocale()),,
goalRelation);
    // TODO: special permission to delete?
    deleteButton = addActionButton(new
Button getResourceBundleHelpergetLocale()
    .getString(PROP_LABEL_DELETE_BUTTON, "Delete"));
    deleteButton.addActionListener(new DeleteListener(this));
    deleteButton.setEnabled(!isReadOnlyMode());
} else {
    addInput("fromGoal", PROP_LABEL_FROM_GOAL, "From Goal", new
SelectField(),
        new CombinedListModel(getGoalNames(), getFromGoal().getName(),
true));
    addInput("toGoal", PROP_LABEL_TO_GOAL, "To Goal", new
SelectField(),
        new CombinedListModel(getGoalNames(), "", true));
    addInput("relationType", PROP_LABEL_RELATION_TYPE, "Relation", new
SelectField(),
        new CombinedListModel(getGoalRelationTypeNames(), "", true));
    addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
null);
}
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);

```

```

        if (stakeholder != null) {
            return !stakeholder.hasPermission(Goal.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditGoalRelationCommand command = getProjectCommandFactory()
            .newEditGoalRelationCommand();
        command.setGoalRelation(goalRelation);
        command.setProjectOrDomain(getProjectOrDomain());
        command.setEditedBy(currentUser());
        command.setFromGoal(getInputValue("fromGoal", String.class));
        command.setToGoal(getInputValue("toGoal", String.class));
        command.setRelationType(getInputValue("relationType",
String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
                updateListener);
        }
        getEventDispatcher().dispatchEvent(
            new UpdateEntityEvent(this, command.getGoalRelation()));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        }
    }
}

```

```

} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the goal relation: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

private Set<String> getGoalNames() {
    Set<String> goalNames = new TreeSet<String>();
    if (getProjectOrDomain() != null) {
        for (Goal goal : getProjectOrDomain().getGoals()) {
            goalNames.add(goal.getName());
        }
    }
    return goalNames;
}

private Set<String> getGoalRelationTypeNames() {
    Set<String> goalRelationTypeNames = new TreeSet<String>();
    for (GoalRelationType relationType : GoalRelationType.values()) {
        goalRelationTypeNames.add(relationType.toString());
    }
    return goalRelationTypeNames;
}

private ProjectOrDomain getProjectOrDomain() {
    ProjectOrDomain pod = null;
    if (getTargetObject() instanceof Goal) {
        pod = getFromGoal().getProjectOrDomain();
    } else if (getTargetObject() instanceof GoalRelation) {
        pod = getGoalRelation().getFromGoal().getProjectOrDomain();
    }
    return pod;
}

private Goal getFromGoal() {
    Goal goal = null;
    if (getTargetObject() instanceof Goal) {
        goal = (Goal) getTargetObject();
    } else if (getTargetObject() instanceof GoalRelation) {
        goal = getGoalRelation().getFromGoal();
    }
    return goal;
}

private GoalRelation getGoalRelation() {
    if (getTargetObject() instanceof GoalRelation) {
        return (GoalRelation) getTargetObject();
    }
    return null;
}

private static class DeleteListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GoalRelationEditorPanel panel;

    private DeleteListener(GoalRelationEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            Goal toGoalForUpdateEvent = panel.getGoalRelation().getToGoal();
            Goal fromGoalForUpdateEvent =
panel.getGoalRelation().getFromGoal();
            DeleteGoalRelationCommand deleteGoalRelationCommand = panel
                .getProjectCommandFactory().newDeleteGoalRelationCommand();
            deleteGoalRelationCommand.setEditedBy(panel.getCurrentUser());
            deleteGoalRelationCommand.setGoalRelation(panel.getGoalRelation());
;
            deleteGoalRelationCommand = panel.getCommandHandler().execute(
                deleteGoalRelationCommand);
            panel.deleted = true;
            panel.getEventDispatcher().dispatchEvent(
                new DeletedEntityEvent(this, null, toGoalForUpdateEvent));
            panel.getEventDispatcher().dispatchEvent(
                new DeletedEntityEvent(this, panel, fromGoalForUpdateEvent));
        } catch (Exception e) {
            panel.setGeneralMessage("Could not delete goal relation: " + e);
        }
    }
}

```

```

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final GoalRelationEditorPanel panel;

    private UpdateListener(GoalRelationEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        GoalRelation existingGoalRelation = (GoalRelation)
        panel.getTargetObject();
        if ((e instanceof UpdateEntityEvent) && (existingGoalRelation != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            GoalRelation updatedGoalRelation = null;
            if (event.getObject() instanceof GoalRelation) {
                updatedGoalRelation = (GoalRelation) event.getObject();
            }
            if ((event instanceof DeletedEntityEvent)
                && existingGoalRelation.equals(updatedGoalRelation)) {
                panel.deleted = true;
                panel.getEventDispatcher().dispatchEvent(
                    new DeletedEntityEvent(this, panel, existingGoalRelation));
                return;
            }
        } else if (event.getObject() instanceof Goal) {
            if (event instanceof DeletedEntityEvent) {
                // TODO: if the to/from goal is deleted then this goal
                // relation was probably deleted too.

            } else {
                // update the selection lists
                Component fromGoalComponent = panel.getInput("fromGoal");
                Component toGoalComponent = panel.getInput("toGoal");
                ComponentManipulator cm = ComponentManipulators
                    .getManipulator(fromGoalComponent);
                cm.setModel(fromGoalComponent, new
                CombinedListModel(panel.getGoalNames(),
                    existingGoalRelation.getFromGoal().getName(), true));
                cm = ComponentManipulators.getManipulator(toGoalComponent);
                cm.setModel(toGoalComponent, new
                CombinedListModel(panel.getGoalNames(),
                    existingGoalRelation.getToGoal().getName(), true));
            }
        }
        if (event.getObject() instanceof Annotation) {
            Annotation updatedAnnotation = (Annotation) event.getObject();
            if (event instanceof DeletedEntityEvent) {
                if
                (existingGoalRelation.getAnnotations().contains(updatedAnnotation)) {
                    existingGoalRelation.getAnnotations().remove(updatedAnnotation)
                ;
                updatedGoalRelation = existingGoalRelation;
            }
            } else if
            (updatedAnnotation.getAnnotatables().contains(existingGoalRelation)) {
                for (Annotatable annotatable :
                updatedAnnotation.getAnnotatables()) {
                    if (annotatable.equals(existingGoalRelation)) {
                        updatedGoalRelation = (GoalRelation) annotatable;
                        break;
                    }
                }
            }
            if ((updatedGoalRelation != null)
                && updatedGoalRelation.equals(existingGoalRelation)) {
                // TODO: check the input fields to see if the user has made
                // a change before resetting the object and updating the
                // input fields.
                panel.setTargetObject(updatedGoalRelation);
                panel.setInputValue("annotations", updatedGoalRelation);
            }
        }
        // if another goal was added or removed, update the to/from
        // selectors
        if (event.getObject() instanceof Goal) {
            // update the selection lists
            Component fromGoalComponent = panel.getInput("fromGoal");
            Component toGoalComponent = panel.getInput("toGoal");
            ComponentManipulator cm = ComponentManipulators
                .getManipulator(fromGoalComponent);
            cm.setModel(fromGoalComponent, new
            CombinedListModel(panel.getGoalNames(),
                existingGoalRelation.getFromGoal().getName(), true));
            cm = ComponentManipulators.getManipulator(toGoalComponent);
            cm.setModel(toGoalComponent, new
            CombinedListModel(panel.getGoalNames(),
                existingGoalRelation.getToGoal().getName(), true));
        }
    }
}

```

```
}
```

goalrelationimpl.java

```
/*
 * $Id: GoalRelationImpl.java,v 1.26 2009/02/12 11:01:35 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.io.Serializable;
import java.util.Date;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Sort;
```

```
import org.hibernate.annotations.SortType;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.impl.AbstractAnnotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalRelation;
import edu.harvard.fas.rregan.requel.project.GoalRelationType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBAnnotatablePatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "goal_relations", uniqueConstraints =
{ @UniqueConstraint(columnNames =
{ "toGoalInternal_id", "fromGoalInternal_id" }) })
@XmlRootElement(name = "goalRelation", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "goalRelation", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class GoalRelationImpl implements GoalRelation, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private Goal fromGoal;
    private Goal toGoal;
    private GoalRelationType relationType;
    private Set<Annotation> annotations = new TreeSet<Annotation>();
    private User createdBy;
    private Date dateCreated = new Date();
    // start at 1 so hibernate recognizes the new instance as the initial
    value
    // and not stale.
    private int version = 1;
```

```

/**
 * @param fromGoal -
 *          the origin of the relationship, this goal has the
relationship
 *          with the toGoal
 * @param toGoal -
 *          the target of the relationship
 * @param relationType -
 *          the type of relationship, see GoalRelationType
 * @param createdBy -
 *          the user that created the relationship
 */
public GoalRelationImpl(Goal fromGoal, Goal toGoal, GoalRelationType
relationType,
    User createdBy) {
    setFromGoal(fromGoal);
    setToGoal(toGoal);
    setRelationType(relationType);
    setCreatedBy(createdBy);
    setDateCreated(new Date());
}

protected GoalRelationImpl() {
    // for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return id;
}

protected void setId(Long id) {
    this.id = id;
}

@Version
protected int getVersion() {
    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

```

```

@Override
@XmlTransient
@Transient
public String getDescription() {
    return "Goal Relation: " + getFromGoal().getName() + " " +
getRelationType().name() + " "
        + getToGoal().getName();
}

@Override
@Enumerated(EnumType.STRING)
@XmlAttribute(name = "relationType")
@XmlJavaTypeAdapter(GoalRelationTypeAdapter.class)
public GoalRelationType getRelationType() {
    return relationType;
}

public void setRelationType(GoalRelationType relationType) {
    this.relationType = relationType;
}

@Transient
@XmlTransient
public Goal getFromGoal() {
    return getFromGoalInternal();
}

public void setFromGoal(Goal fromGoal) {
    if (getFromGoalInternal() != null) {
        getFromGoalInternal().getRelationsFromThisGoal().remove(this);
    }
    setFromGoalInternal(fromGoal);
    fromGoal.getRelationsFromThisGoal().add(this);
}

@ManyToOne(targetEntity = GoalImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
protected Goal getFromGoalInternal() {
    return fromGoal;
}

protected void setFromGoalInternal(Goal fromGoal) {
    this.fromGoal = fromGoal;
}

@Transient

```

```

@XmlIDREF
@XmlAttribute(name = "toGoal")
@XmlJavaTypeAdapter(Goal2GoalImplAdapter.class)
public Goal getToGoal() {
    return getToGoalInternal();
}

public void setToGoal(Goal toGoal) {
    if (getToGoalInternal() != null) {
        getToGoalInternal().getRelationsToThisGoal().remove(this);
    }
    setToGoalInternal(toGoal);
    toGoal.getRelationsToThisGoal().add(this);
}

@XmlTransient
@ManyToOne(targetEntity = GoalImpl.class, cascade =
{ CascadeType.REFRESH }, optional = false)
public Goal getToGoalInternal() {
    return toGoal;
}

public void setToGoalInternal(Goal toGoal) {
    this.toGoal = toGoal;
}

@XmlElementWrapper(name = "annotations", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
// changed xml mapping to output references to annotations instead of
the
// annotations directly because
// an annotation may be shared by multiple entities causing
duplicates on
// import. this makes report
// generating via xslt more complicated because of the indirection.
@XmlIDREF
@XmlElement(name = "annotationRef", type = AbstractAnnotation.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
// @XmlElementRef(type = AbstractAnnotation.class)
@ManyToMany(targetEntity = AbstractAnnotation.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY)
@Sort(type = SortType.NATURAL)
public Set<Annotation> getAnnotations() {
    return annotations;
}

```

```

protected void setAnnotations(Set<Annotation> annotations) {
    this.annotations = annotations;
}

@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH }, optional = false)
@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
public User getCreatedBy() {
    return createdBy;
}

protected void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

@Override
public int compareTo(GoalRelation o) {
    int compareFromGoal = (getFromGoal() == null ? -1 : getFromGoal()
        .compareTo(o.getFromGoal()));
    int compareToGoal = (getToGoal() == null ? -1 :
        getToGoal().compareTo(o.getToGoal()));
    int compareRelationType = (getRelationType() == null ? -1 :
        getRelationType().compareTo(
            o.getRelationType()));
    return (compareFromGoal != 0 ? compareFromGoal : (compareToGoal != 0
        ? compareToGoal
        : compareRelationType));
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {

```

```

if (tmpHashCode == null) {
    if (getId() != null) {
        tmpHashCode = new Integer(getId().hashCode());
    }
    final int prime = 31;
    int result = 1;
    result = prime * result
        + ((getFromGoalInternal() == null) ? 0 :
    getFromGoalInternal().hashCode());
    result = prime * result
        + ((getToGoalInternal() == null) ? 0 :
    getToGoalInternal().hashCode());
    result = prime * result
        + ((getRelationType() == null) ? 0 :
    getRelationType().hashCode());
    tmpHashCode = new Integer(result);
}
return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final GoalRelationImpl other = (GoalRelationImpl) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getFromGoalInternal() == null) {
        if (other.getFromGoalInternal() != null) {
            return false;
        }
    } else if (!
    getFromGoalInternal().equals(other.getFromGoalInternal())) {
        return false;
    }
    if (getToGoalInternal() == null) {
        if (other.getToGoalInternal() != null) {
            return false;
        }
    } else if (!
    getToGoalInternal().equals(other.getToGoalInternal())))
        return false;
    }
    if (getRelationType() == null) {
        if (other.getRelationType() != null) {
            return false;
        }
    } else if (!getRelationType().equals(other.getRelationType()))
        return false;
    }
    return true;
}

@Override
public String toString() {
    return GoalRelation.class.getName() + "[" + getId() + "]: " +
    getFromGoal().toString()
        + " " + getRelationType().toString() + " "
    getToGoal().toString();
}

/**
 * This class is used by JAXB to convert the id of an entity into an
 * xml id
 * string that will be distinct from other entity xml id strings by
 * the use
 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "GLR_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

```

    /**
     * This is for JAXB to patchup the parent/child relationship and to
patchup
     * existing persistent objects for the objects that are attached
directly to
     * this object.
    *
    * @param userRepository
    * @param defaultCreatedByUser -
    *           the user to be set as the created by if no user is
supplied.
    * @param parent
    * @see UnmarshallerListener
    */
public void afterUnmarshal(final UserRepository userRepository, User
defaultCreatedByUser,
    Object parent) {
    setFromGoal((Goal) parent);
    UnmarshallingContext.getInstance().addPatcher(new
JAXBAnnotatablePatcher(this));
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
}

/**
 * This class is used by JAXB to convert the GoalRelationType of a
 * GoalRelation into a string for an attribute in the xml file and
the
 * reverse when unmarshalling.
 *
 * @author ron
 */
@XmlTransient
public static class GoalRelationTypeAdapter extends
XmlAdapter<String, GoalRelationType> {

    @Override
    public GoalRelationType unmarshal(String typeString) throws
Exception {
        return GoalRelationType.valueOf(typeString);
    }

    @Override
    public String marshal(GoalRelationType type) throws Exception {
        return type.toString();
    }
}

```

```

    }
}
}
```

goalrelationtype.java

```

/*
 * $Id: GoalRelationType.java,v 1.2 2008/04/09 10:33:27 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

/**
 * The type of relationship for a GoalRelation.
 *
 * @author ron
 */
public enum GoalRelationType {

    /**
     * Declares that the from goal has a positive influence on the
success of
     * the to goal.
     */
    Supports,

    /**
     * Declares that the from goal is incompatible with the to goal such
that
     * both can not be satisfied.
     */
    Conflicts;

    private GoalRelationType() {
    }
}

```

goalselectorpanel.java

```

/*
 * $Id: GoalSelectorPanel.java,v 1.3 2009/02/23 07:37:22 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;
```

```

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;

```

```

import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class GoalSelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(GoalSelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the GoalNavigatorPanel.properties to set
     * the
     *   * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * @param projectRepository
     */
    public GoalSelectorPanel(ProjectRepository projectRepository) {
        super(GoalSelectorPanel.class.getName(), Project.class,
              ProjectManagementPanelNames.PROJECT_GOALS_SELECTOR_PANEL_NAME);
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.setRowLevelSelection(true);

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Goal goal = (Goal) model.getBackingObject(row);
                    return goal.getName();
                }
            }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",

```

```

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
    int row) {
        Goal goal = (Goal) model.getBackingObject(row);
        return goal.getCreatedBy().getUsername();
    }
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
    int row) {
        Goal goal = (Goal) model.getBackingObject(row);
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        return format.format(goal.getDateCreated());
    }
}));
}

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the title is "Select Goal"<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Select Goal");
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);
    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Goal.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}
}

```

```

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 *
 * @return an adapter to get the collection of items to select from
from the
 * target object.
 */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
    return new NavigatorTableModelAdapter() {
        private ProjectOrDomain targetObject;

        @Override
        public Collection<Object> getCollection() {
            return (Collection) targetObject.getGoals();
        }

        @Override
        public void setTargetObject(Object targetObject) {
            this.targetObject = (ProjectOrDomain) targetObject;
        }
    };
}

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the goal
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;
}

```

```

private final GoalSelectorPanel panel;

private UpdateListener(GoalSelectorPanel panel) {
    this.panel = panel;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
        if (event.getObject() instanceof ProjectOrDomain) {
            updatedPod = (ProjectOrDomain) event.getObject();
        } else if (event.getObject() instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
            updatedPod = updatedEntity.getProjectOrDomain();
        } else if (event.getObject() instanceof Annotation) {
            if (!(event instanceof DeletedEntityEvent)) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                    if ((annotatable instanceof ProjectOrDomain)
&& annotatable.equals(panel.getProjectOrDomain())) {
                        updatedPod = (ProjectOrDomain) annotatable;
                        break;
                    } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                        ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                        if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                            updatedPod = entity.getProjectOrDomain();
                            break;
                        }
                    }
                }
            }
        }
        if (panel.getProjectOrDomain().equals(updatedPod)) {
            panel.setTargetObject(updatedPod);
        }
    }
}

```

goalselrelationexception.java

```
/*
 * $Id: GoalSelfRelationException.java,v 1.1 2009/03/26 00:43:03
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requell.project.exception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requell.project.Goal;
import edu.harvard.fas.rregan.requell.project.GoalRelation;

/**
 * @author ron
 */
public class GoalSelfRelationException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG = "A goal cannot have a relation to
itself.";

    /**
     * @param goal -
     *          the goal
     * @return
     */
    public static GoalSelfRelationException forGoal(Goal goal) {
        return new GoalSelfRelationException(GoalRelation.class, null,
"toGoal", goal,
        EntityExceptionActionType.Creating, MSG);
    }

    protected GoalSelfRelationException(Class<?> entityType, Object
entity,
        String entityPropertyName, Object entityValue,
EntityExceptionActionType actionPerformed,
        String format, Object... messageArgs) {
        super(entityType, entity, entityPropertyName, entityValue,
actionPerformed, format, messageArgs);
    }

    protected GoalSelfRelationException(Throwable cause, Class<?>
entityType, Object entity,
```

```
        String entityPropertyName, Object entityValue,
EntityExceptionActionType actionPerformed,
        String format, Object... messageArgs) {
        super(cause, entityType, entity, entityPropertyName, entityValue,
actionPerformed, format,
        messageArgs);
    }
}
```

goalstable.java

```
/*
 * $Id: GoalsTable.java,v 1.5 2009/01/08 06:48:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requell.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requell.project.Goal;
import edu.harvard.fas.rregan.requell.project.GoalContainer;
import edu.harvard.fas.rregan.requell.project.Project;
import edu.harvard.fas.rregan.requell.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requell.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.requell.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requell.ui.AbstractRequellNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component to add to Panels of goal container entity editors to
enable
 * editing of the goals of the entity.
 *
 * @author ron
 */
public class GoalsTable extends AbstractRequelNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(GoalsTable.class, new
GoalsTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * GoalsTable to define the label of the goals field. If the property
is
     * undefined the panel should use a sensible default such as "Goals".
     */
    public static final String PROP_LABEL_GOALS = "Goals.Label";

    /**
     * The name to use in the containing panels properties file to set
the
     * label
     * of the view button in the goal edit table column. If the property
is
     * undefined "View" is used.
     */
    public static final String PROP_VIEW_GOAL_BUTTON_LABEL =
"ViewGoal.Label";

    /**
     * The name to use in the containing panels properties file to set
the
     * label
     * of the remove button in the goal edit table column. If the
property
is
     * undefined "Remove" is used.
     */
    public static final String PROP_REMOVE_GOAL_BUTTON_LABEL =
"RemoveGoal.Label";

    /**
     * The name to use in the containing panels properties file to set
the
     * label
     * of the edit button in the goal edit table column. If the property
is
     * undefined "Edit" is used.
     */
    public static final String PROP_EDIT_GOAL_BUTTON_LABEL =
>EditGoal.Label";

    /**
     * The name to use in the containing panels properties file to set
the
     * label
     * of the new goal button under the goals table. If the property is
     * undefined "New" is used.
     */
    public static final String PROP_NEW_GOAL_BUTTON_LABEL =
>NewGoal.Label";

    /**
     * The name to use in the containing panels properties file to set
the
     * label
     * of the find goal button under the goals table. If the property is
     * undefined "Find" is used.
     */
}

```

```

/*
public static final String PROP_FIND_GOAL_BUTTON_LABEL =
"FindGoal.Label";

private GoalContainer goalContainer;
private final NavigatorTable table;
private final NavigatorButton openGoalEditorButton;
private final NavigatorButton openGoalSelectorButton;
private final ProjectCommandFactory projectCommandFactory;
private final CommandHandler commandHandler;
private AddGoalToGoalContainerController addGoalController;
private RemoveGoalFromGoalContainerController removeGoalController;

/**
 * @param editMode
 * @param resourceBundleHelper
 * @param projectCommandFactory -
 *          passed to the add/remove goal to goal container
controller
 * @param commandHandler -
 *          passed to the add/remove goal to goal container
controller
 */
public GoalsTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper,
    ProjectCommandFactory projectCommandFactory, CommandHandler
commandHandler) {
    super(editMode, resourceBundleHelper);
    this.projectCommandFactory = projectCommandFactory;
    this.commandHandler = commandHandler;
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    table = new NavigatorTable(getTableConfig());
    table.setLayoutData(layoutData);
    add(table);

    Row buttons = new Row();
    buttons.setLayoutData(layoutData);

    String buttonLabel = getResourceBundleHelpergetLocale().getString(
        PROP_NEW_GOAL_BUTTON_LABEL, "New");
    openGoalEditorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
    openGoalEditorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
    openGoalEditorButton.setVisible(false);
    buttons.add(openGoalEditorButton);
}

```

```

buttonLabel =
getResourceBundleHelpergetLocale().getString(PROP_FIND_GOAL_BUTTON_L
ABEL,
"Find");
openGoalSelectorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openGoalSelectorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
openGoalSelectorButton.setVisible(false);
buttons.add(openGoalSelectorButton);

add(buttons);

}

protected GoalContainer getGoalContainer() {
    return goalContainer;
}

protected void setGoalContainer(GoalContainer goalContainer) {
    this.goalContainer = goalContainer;

    if (goalContainer != null) {
        table.setModel(new NavigatorTableModel((Collection)
goalContainer.getGoals()));
        if (!isReadOnlyMode()) {
            NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor,
            getGoalContainer(), Goal.class, null,
WorkflowDisposition.NewFlow);
            openGoalEditorButton.setEventToFire(openEditorEvent);
            openGoalEditorButton.setVisible(true);

            ProjectOrDomain pod = null;
            if (getGoalContainer() instanceof ProjectOrDomain) {
                pod = (ProjectOrDomain) getGoalContainer();
            } else if (getGoalContainer() instanceof ProjectOrDomainEntity) {
                pod = ((ProjectOrDomainEntity)
getGoalContainer()).getProjectOrDomain();
            }
            if (pod != null) {
                NavigationEvent openGoalSelectorEvent = new OpenPanelEvent(this,
                    PanelActionType.Selector, pod, Project.class,
                    ProjectManagementPanelNames.PROJECT_GOALS_SELECTOR_PANEL_NAME,
                    WorkflowDisposition.ContinueFlow);

                openGoalSelectorButton.setEventToFire(openGoalSelectorEvent);
                openGoalSelectorButton.setVisible(true);
            }
        }
    }
}

```

```

} else {
    openGoalSelectorButton.setVisible(false);
}

// use the the goal table (this) as the destination because it
// is used as the source to the open panel events created above
if (addGoalController != null) {
    getEventDispatcher().removeEventTypeActionListener(SelectEntityEvent.class,
        addGoalController, this);
}
addGoalController = new AddGoalToGoalContainerController(getEventDispatcher(),
    projectCommandFactory, commandHandler, goalContainer);
getEventDispatcher().addEventTypeActionListener(SelectEntityEvent.class,
    addGoalController, this);

// use the the goal table (this) as the destination because it
// is used as the source to the open panel events created above
if (removeGoalController != null) {
    getEventDispatcher().removeEventTypeActionListener(
        RemoveGoalFromGoalContainerEvent.class, removeGoalController,
        this);
}
removeGoalController = new RemoveGoalFromGoalContainerController(
    getEventDispatcher(), projectCommandFactory, commandHandler,
    goalContainer);
getEventDispatcher().addEventTypeActionListener(
    RemoveGoalFromGoalContainerEvent.class, removeGoalController,
    this);

}
} else {
    table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    openGoalEditorButton.setVisible(false);
    openGoalSelectorButton.setVisible(false);
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override

```

```

        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    Row buttonsContainer = new Row();
    RowLayoutData buttonLayout = new RowLayoutData();
    buttonLayout.setAlignment(Alignment.ALIGN_CENTER);
    buttonLayout.setInsets(new Insets(5, 0));

    Goal goal = (Goal) model.getBackingObject(row);
    String buttonLabel = null;
    if (isReadOnlyMode()) {
        buttonLabel = getResourceBundleHelpergetLocale().getString(
            PROP_VIEW_GOAL_BUTTON_LABEL, "View");
    } else {
        buttonLabel = getResourceBundleHelpergetLocale().getString(
            PROP_EDIT_GOAL_BUTTON_LABEL, "Edit");
    }
    NavigationEvent openEditorEvent = new
OpenPanelEvent(GoalsTable.this,
        PanelEventType.Editor, goal, goal.getClass(), null,
        WorkflowDisposition.NewFlow);
    NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
        getEventDispatcher(), openEditorEvent);
    openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
    openEditorButton.setLayoutData(buttonLayout);
    buttonsContainer.add(openEditorButton);

    if (!isReadOnlyMode()) {
        buttonLabel = getResourceBundleHelpergetLocale().getString(
            PROP_REMOVE_GOAL_BUTTON_LABEL, "Remove");
        NavigationEvent removeGoalEvent = new
RemoveGoalFromGoalContainerEvent(
            GoalsTable.this, goal, goalContainer, GoalsTable.this);
        NavigatorButton removeGoalButton = new
NavigatorButton(buttonLabel,
            getEventDispatcher(), removeGoalEvent);
        removeGoalButton.setStyleName(Panel.STYLE_NAME_PLAIN);
        removeGoalButton.setLayoutData(buttonLayout);
        buttonsContainer.add(removeGoalButton);
    }
    return buttonsContainer;
}
)));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override

```

```

    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    Goal goal = (Goal) model.getBackingObject(row);
    return goal.getName();
}
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    Goal goal = (Goal) model.getBackingObject(row);
    return goal.getCreatedBy().getUsername();
}
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    Goal goal = (Goal) model.getBackingObject(row);
    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
    return formatter.format(goal.getDateCreated());
}
});
}

return tableConfig;
}

private static class GoalsTableManipulator extends
AbstractComponentManipulator {

protected GoalsTableManipulator() {
super();
}

@Override
public Object getModel(Component component) {
return getValue(component, GoalContainer.class);
}

@Override
public void setModel(Component component, Object valueModel) {

```

```

        setValue(component, valueModel);
    }

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
// nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
return type.cast(getComponent(component).getGoalContainer());
}

@Override
public void setValue(Component component, Object value) {
getComponent(component).setGoalContainer((GoalContainer) value);
}

private GoalsTable getComponent(Component component) {
return (GoalsTable) component;
}
}
}
```

grammaticalrelation.java

```

/*
 * $Id: GrammaticalRelation.java,v 1.3 2008/07/23 10:05:21 rregan Exp
 */
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp;

/**
 * @author ron
 */
public interface GrammaticalRelation extends
Comparable<GrammaticalRelation> {

/**
 * @return The type of grammatical relationship, such as subject,
modifier,
* etc.
*/

```

```

public GrammaticalRelationType getType();

/**
 * @return The governor or "from" element of the relationship
 */
public NLPText getGovernor();

/**
 * @return The dependent or "to" element of the relationship
 */
public NLPText getDependent();
}

```

grammaticalrelationimpl.java

```

/*
 * $Id: GrammaticalRelationImpl.java,v 1.3 2009/02/10 03:30:46 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * @author ron
 */
public class GrammaticalRelationImpl implements GrammaticalRelation {

    private final GrammaticalRelationType type;
    private final NLPText governor;
    private final NLPText dependent;

    /**
     * @param type
     * @param governor
     * @param dependent
     */
    public GrammaticalRelationImpl(GrammaticalRelationType type, NLPText
governor, NLPText dependent) {
        this.type = type;
        this.governor = governor;
        this.dependent = dependent;
        (NLPTextImpl) governor).addGovernorOf(this);
    }
}

```

```

        ((NLPTextImpl) dependent).addDependentOf(this);
    }

    /**
     * @see edu.harvard.fas.rregan.nlp.GrammaticalRelation#getType()
     */
    public GrammaticalRelationType getType() {
        return type;
    }

    /**
     * @see edu.harvard.fas.rregan.nlp.GrammaticalRelation#getGovernor()
     */
    public NLPText getGovernor() {
        return governor;
    }

    /**
     * @see edu.harvard.fas.rregan.nlp.GrammaticalRelation#getDependent()
     */
    public NLPText getDependent() {
        return dependent;
    }

    /**
     * @see
     * @see edu.harvard.fas.rregan.nlp.GrammaticalRelation#compareTo(edu.harvard.f
as.rregan.nlp.impl.GrammaticalRelation)
     */
    public int compareTo(GrammaticalRelation o) {
        if
        (getDependent().getWordIndex().equals(o.getDependent().getWordIndex()))
        {
            if
            (getGovernor().getWordIndex().equals(o.getGovernor().getWordIndex()))
            {
                return
                getType().getShortName().compareTo(o.getType().getShortName());
            }
            return
            (getGovernor().getWordIndex().compareTo(o.getGovernor().getWordIndex()));
        }
        return
        (getDependent().getWordIndex().compareTo(o.getDependent().getWordIndex
())));
    }
}

```

```

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((getDependent() == null) ? 0 :
getDependent().hashCode());
    result = prime * result + ((getGovernor() == null) ? 0 :
getGovernor().hashCode());
    result = prime * result + ((getType() == null) ? 0 :
getType().hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    final GrammaticalRelationImpl other = (GrammaticalRelationImpl) obj;
    if (dependent == null) {
        if (other.dependent != null)
            return false;
    } else if (!dependent.equals(other.dependent))
        return false;
    if (governor == null) {
        if (other.governor != null)
            return false;
    } else if (!governor.equals(other.governor))
        return false;
    if (type == null) {
        if (other.type != null)
            return false;
    } else if (!type.equals(other.type))
        return false;
    return true;
}

@Override
public String toString() {
    return getType().getShortName() + "(" + getGovernor().getText() +
"_" +
        + getGovernor().getWordIndex() + ", " + getDependent().getText() +
"_" +

```

```

        + getDependent().getWordIndex() + ")";
    }
}

```

grammaticalrelationtype.java

```

/*
 * $Id: GrammaticalRelationType.java,v 1.10 2009/02/06 11:49:18 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

import java.util.HashMap;
import java.util.Map;

/**
 * Grammatical relationships between words in a sentence. Based on the
 * dependency parse from the Stanford Parser.
 *
 * @see {@link
 * "http://nlp.stanford.edu/software/dependencies_manual.pdf"}
 * @author ron
 */
public enum GrammaticalRelationType {

    /**
     * The root relationship indicating that the governor "governs" the
     * dependent.
     */
    GOVERNOR(null, "gov", "governor", ""),
    /**
     *
     */
    DEPENDENT(null, "dep", "dependent", ""),
    /**
     *
     */
    PREDICATE(DEPENDENT, "pred", "predicate", ""),
    /**
     *
     */
    AUXILIARY(DEPENDENT, "aux", "auxiliary", ""),
    /**
     *
     */
    ADJECTIVE(DEPENDENT, "adv", "adjective", ""),
    /**
     *
     */
    ADVERB(DEPENDENT, "adv", "adverb", ""),
    /**
     *
     */
    ADPOSITION(DEPENDENT, "adv", "adposition", ""),
    /**
     *
     */
    CASE(DEPENDENT, "adv", "case", ""),
    /**
     *
     */
    COORDINATOR(DEPENDENT, "adv", "coordinator", ""),
    /**
     *
     */
    CONJUNCTION(DEPENDENT, "adv", "conjunction", ""),
    /**
     *
     */
    DETERMINER(DEPENDENT, "adv", "determiner", ""),
    /**
     *
     */
    NUMBER(DEPENDENT, "adv", "number", ""),
    /**
     *
     */
    PRONOUN(DEPENDENT, "adv", "pronoun", ""),
    /**
     *
     */
    PREPOSITION(DEPENDENT, "adv", "preposition", ""),
    /**
     *
     */
    RELATIVE(DEPENDENT, "adv", "relative", ""),
    /**
     *
     */
    VERB(DEPENDENT, "adv", "verb", "")
}

```

```

/***
 *
 */
PASSIVE_AUXILIARY(AUXILIARY, "auxpass", "passive auxiliary", ""),
/***
 *
 */
COPULA(
    AUXILIARY,
    "cop",
    "copula",
    "a copula is a word used to link the subject of a sentence with a
predicate. It is sometimes referred to as a linking verb."),
/***
 *
 */
CONJUNCT(DEPENDENT, "conj", "conjunction", ""),
/***
 *
 */
COORDINATION(DEPENDENT, "cc", "coordination", ""),
/***
 * a relation between the main verb of a clause and other sentential
 * elements, such as a sentential parenthetical, a clause after a ":" or a
 * ";".<br>
 * For example:<br>
 * "The guy, John said, left early in the morning."<br>
 * parataxis(left, said)
 */
PARATAxis(DEPENDENT, "parataxis", "parataxis", ""),
/***
 *
 */
PUNCTUATION(DEPENDENT, "punct", "punctuation", ""),
/***
 *
 */
ARGUMENT(DEPENDENT, "arg", "argument", ""),

```

```

/***
 *
 */
SUBJECT(ARGUMENT, "subj", "subject", ""),
/***
 *
 */
NOMINAL SUBJECT(SUBJECT, "nsubj", "nominal subject",
    "a subject consisting of a noun, and possibly modifiers."),
/***
 *
 */
NOMINAL_PASSIVE SUBJECT(SUBJECT, "nsubjpass", "nominal passive
subject", ""),
/***
 * csubj : clausal subject<br>
 * A clausal subject is a clausal syntactic subject of a clause, i.e.
the
 * subject is itself a clause. The governor of this relation might
not
 * always be a verb: when the verb is a copular verb, the root of the
clause
 * is the complement of the copular verb. In the two following
examples,
 * what she said is the subject.<br>
 * What she said makes sense. - csubj (makes, said)<br>
 * What she said is not true. - csubj (true, said)<br>
 */
CLAUStAL SUBJECT(
    SUBJECT,
    "csubj",
    "clausal subject",
    "A clausal subject is a clausal syntactic subject of a clause, i.e.
the subject is itself a clause."),
/***
 * csubjpass: clausal passive subject<br>
 * A clausal passive subject is a clausal syntactic subject of a
passive
 * clause. In the example below, that she lied is the subject.<br>
 * That she lied was suspected by everyone. - csubjpass(suspected,
lied)<br>
 */

```

```

CLAUSAL_PASSIVE SUBJECT(CLAUSAL SUBJECT, "csubjpass", "clausal
passive subject",
    "A clausal passive subject is a clausal syntactic subject of a
passive clause."),
    /**
     * A complement ties an object to a verb in a different phrase or
clause
    */
COMPLEMENT(
    ARGUMENT,
    "comp",
    "complement",
    "A complement of a VP is any object (direct or indirect) of that
VP, or a "
    + "clause or adjectival phrase which functions like an object; a
complement of "
    + "a clause is an complement of the VP which is the predicate of
that clause."),
    /**
     *
    */
ATTRIBUTIVE(COMPLEMENT, "attr", "attributive", ""),
    /**
     *
    */
OBJECT(COMPLEMENT, "obj", "object", ""),
    /**
     *
    */
DIRECT_OBJECT(OBJECT, "dobj", "direct object", ""),
    /**
     *
    */
INDIRECT_OBJECT(OBJECT, "iobj", "indirect object", ""),
    /**
     *
    */
PREPOSITIONAL_OBJECT(OBJECT, "pobj", "prepositional object", ""),
    /**
     *
    */
    */
PREPOSITIONAL_COMPLEMENT(OBJECT, "pcomp", "prepositional complement",
"),
    /**
     *
    */
CLAUSAL_COMPLEMENT(COMPLEMENT, "ccomp", "clausal complement", ""),
    /**
     *
     * xcomp: open clausal complement<br>
     * An open clausal complement (xcomp) of a VP or an ADJP is a clausal
complement without its own subject, whose reference is determined
by an
     * external subject. These complements are always non-finite. The
name xcomp
     * is borrowed from Lexical-Functional Grammar.<br>
     * He says that you like to swim. - xcomp(like, swim)<br>
     * I am ready to leave. - xcomp(ready, leave)<br>
     * John hates eating fish. - xcomp(hates, eating)<br>
    */
XCLAUSAL_COMPLEMENT(COMPLEMENT, "xcomp", "xclausal complement",
    "An open clausal complement without its own subject."),
    /**
     *
     * example: I bet Bob a dollar [that] Pi is a number.
    */
COMPLEMENTIZER(COMPLEMENT, "complm", "complementizer",
    "Like a subordinating conjunction used to show the relationship
between the two clauses"),
    /**
     *
    */
MARKER(COMPLEMENT, "mark", "marker", ""),
    /**
     *
    */
RELATIVE(COMPLEMENT, "rel", "relative", ""),
    /**
     *
    */
REFERENT(DEPENDENT, "ref", "referent", ""),
    /**

```

```

/*
 */
EXPLETIVE(DEPENDENT, "expl", "explicative", ""),
/***
 * accom: adjectival complement<br>
 * An adjectival complement of a VP is an adjectival phrase which
functions
 * as the complement (like an object of the verb); an adjectival
complement
 * of a clause is the adjectival complement of the VP which is the
predicate
 * of that clause. <br>
 * She looks very beautiful. - accom(looks, beautiful)
 */
ADJECTIVAL_COMPLEMENT(COMPLEMENT, "acomp", "adjectival complement",
 "An adjectival complement of a VP is an adjectival phrase which
functions as the complement"),

/***
 *
 */
MODIFIER(DEPENDENT, "mod", "modifier", ""),
/***
 *
 */
ADVERBIAL_CLAUSE_MODIFIER(MODIFIER, "advcl", "adverbial clause
modifier", ""),
/***
 *
 */
TEMPORAL_MODIFIER(MODIFIER, "tmod", "temporal modifier", ""),
/***
 *
 */
RELATIVE_CLAUSE_MODIFIER(MODIFIER, "rcmod", "relative clause
modifier", ""),
/***
 *
 */
NUMERIC_MODIFIER(MODIFIER, "num", "numeric modifier", ""),
/***

```

```

/*
 */
ADJECTIVAL_MODIFIER(MODIFIER, "amod", "adjectival modifier", ""),
/***
 *
 */
NN_MODIFIER(MODIFIER, "nn", "nn modifier", ""),
/***
 *
 */
APPOSITIONAL_MODIFIER(MODIFIER, "appos", "appositional modifier",
 ""),
/***
 * abbrev: abbreviation modifer<br>
 * An abbreviation modifer of an NP is a parenthesized NP that serves
to
 * abbreviate the NP (or to define an abbreviation).<br>
 * The Australian Broadcasting Corporation (ABC). - abbrev(Corporation, ABC)
 */
ABBREVIATION_MODIFIER(
    APPOSITIONAL_MODIFIER,
    "abbrev",
    "abbreviation modifier",
    "An abbreviation modifer of an NP is a parenthesized NP that serves
to abbreviate the NP (or to define an abbreviation)."),
/***
 *
 */
PARTICIPIAL_MODIFIER(MODIFIER, "partmod", "participial modifier",
 ""),
/***
 *
 */
INFINITIVAL_MODIFIER(MODIFIER, "infmod", "infinitival modifier", ""),
/***
 *
 */
ADVERBIAL_MODIFIER(MODIFIER, "advmod", "adverbial modifier", ""),
/***

```

```

/*
NEGATION_MODIFIER(ADVERBIAL_MODIFIER, "neg", "negation modifier",
"),

/***
*
*/
DETERMINER(MODIFIER, "det", "determiner", ""),
/***
*
*/
PREDETERMINER(MODIFIER, "predet", "predeterminer", ""),
/***
*
*/
PRECONJUNCT(MODIFIER, "preconj", "preconjunct", ""),
/***
*
*/
POSSESSION_MODIFIER(MODIFIER, "poss", "possession modifier", ""),
/***
*
*/
POSSESSIVE_MODIFIER(MODIFIER, "possessive", "possessive modifier",
"),
/***
*
*/
PREPOSITIONAL_MODIFIER(MODIFIER, "prep", "prepositional modifier",
"),
/***
*
*/
PHRASAL_VERB_PARTICLE(MODIFIER, "prt", "phrasal verb particle", ""),
/***
*
*/
SEMANTIC_DEPENDENT(DEPENDENT, "sdep", "semantic dependent", ""),
*/

```

```

/**
 * xsubj : controlling subject<br>
 * A controlling subject is the relation between the head of a open
clausal
 * complement (xcomp) and the external subject of that clause.<br>
 * Tom likes to eat fish. - xsubj(eat, Tom)
*/
CONTROLLING SUBJECT(
SEMANTIC_DEPENDENT,
"xsubj",
"controlling subject",
"A controlling subject is the relation between the head of a open
clausal complement (xcomp) and the external subject of that clause.",

/**
 * agent: agent<br>
 * An agent is the complement of a passive verb which is introduced
by the
 * preposition "by" and does the action.<br>
 * The man has been killed by the police. - agent(killed, police)<br>
 * Effects caused by the protein are important. - agent(caused,
protein)<br>
 * NOTE: may be returned as the prepositional object:<br>
 * pobj(by-5, police-7)<br>
 * det(man-1, The-0)<br>
 * det(police-7, the-6)<br>
 * aux(killed-4, has-2)<br>
 * nsubjpass(killed-4, man-1)<br>
 * auxpass(killed-4, been-3)<br>
 * prep(killed-4, by-5)<br>
*/
AGENT(DEPENDENT, "agent", "agent", ""),
/**
*
*/
COMPOUND_NUMBER_MODIFIER(MODIFIER, "number", "compound number
modifier", ""),
/**
*
*/
PURPOSE_CLAUSE_MODIFIER(MODIFIER, "purpcl", "purpose clause
modifier", ""),
*/

```

```

*/
QUANTIFIER_MODIFIER(MODIFIER, "quantmod", "quantifier modifier", ""),
/**
 *
 */
MEASURE_PHRASE(MODIFIER, "measure", "measure phrase", ""),
private static final Map<String, GrammaticalRelationType>
grammaticalRelationsByShortName = new HashMap<String,
GrammaticalRelationType>();
static {
    for (GrammaticalRelationType grt : values()) {
        grammaticalRelationsByShortName.put(grt.getShortName(), grt);
    }
}
public static GrammaticalRelationType
getGrammaticalRelationByShortName(String shortName) {
    return grammaticalRelationsByShortName.get(shortName);
}

private final GrammaticalRelationType parent;
private final String shortName;
private final String longName;
private final String description;

private GrammaticalRelationType(GrammaticalRelationType parent,
String shortName,
    String longName, String description) {
this.parent = parent;
this.shortName = shortName;
this.longName = longName;
this.description = description;
}

/**
 * @return A more general relation that this relation is a subtype
of.
 */
public GrammaticalRelationType getParent() {
    return parent;
}

/**
 * @return the short name of the relation as returned by the Stanford
 *         Parser.

```

```

*/
public String getShortName() {
    return shortName;
}

/**
 * @return A human understandable name.
 */
public String getLongName() {
    return longName;
}

/**
 * @return A description of what the relationship means
 */
public String getDescription() {
    return description;
}

/**
 * @param relation -
 *          the relation to compare with this one to see if this
one is
 *          equal to or a sub type of the supplied relation.
 * @return true if this relation is equal to or a sub type of the
supplied
 *         relation.
 */
public boolean isA(GrammaticalRelationType relation) {
    GrammaticalRelationType thisOrAncestor = this;
    do {
        if (thisOrAncestor.equals(relation)) {
            return true;
        }
        thisOrAncestor = thisOrAncestor.getParent();
    } while (thisOrAncestor != null);
    return false;
}

@Override
public String toString() {
    return getShortName();
}
}
```

grammaticalstructurelevel.java

```
/*
 * $Id: GrammaticalStructureLevel.java,v 1.2 2008/10/31 06:09:55
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp;

/**
 * TODO: role this into GrammaticalRelationType
 *
 * @author ron
 */
public enum GrammaticalStructureLevel {
    /**
     * Default if the level of the text is unknown.
     */
    UNKNOWN(),

    /**
     * A collection of words without any particular grammatical relation
     * between them. This may be used by NLPProcessors as a utility for doing its
     * work.
     */
    BAGOFWORDS(),

    /**
     * 
     */
    SENTENCE(),

    /**
     * A clause is a collection of grammatically-related words including
     * a predicate and a subject
     */
    CLAUSE(),
}
```

```
    /**
     * A phrase is a group of two or more grammatically linked words
     * without a
     *   * subject and predicate
     */
    PHRASE(),

    /**
     *
     */
    WORD();
}
```

hashutils.java

```
/*
 * $Id: HashUtils.java,v 1.2 2008/02/16 22:42:46 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan;

import java.security.MessageDigest;

/**
 * HashUtils
 *
 * @author ron
 */
public class HashUtils {

    private static final char[] hexadecimal = { '0', '1', '2', '3', '4',
        '5', '6', '7', '8', '9',
        'a', 'b', 'c', 'd', 'e', 'f' };

    /**
     * calculate the MD5 hash on the supplied input bytes and return it
     * as a 16 byte array.
     *
     * @param input
     *   a byte array
     * @return the 128bit digest
     * @throws ApplicationException
     */
    public static byte[] getMD5HashDigest(byte[] input) throws
        ApplicationException {
```

```

try {
    return MessageDigest.getInstance("MD5").digest(input);
} catch (Exception e) {
    throw new ApplicationException(e, "Exception creating MD5 hash
digest");
}
}

/**
 * @param input
 *          a string
 * @return a 32 character string representation of the 128bit digest
 *          comparable with the results of
 *          org.apache.catalina.realm.RealmBase digest method
 * @throws ApplicationException
 */
public static String getMD5HashDigestString(String input) throws
ApplicationException {
    return encode(getMD5HashDigest(input.getBytes()));
}

/**
 * Encodes the 128 bit (16 bytes) MD5 into a 32 character String.
 *
 * @param binaryData
 *          Array containing the digest
 * @return Encoded MD5, or null if encoding failed This was taken
from
 *          org.apache.catalina.util.MD5Encoder so that the hash
created/used
 *          by the Tomcat container managed authentication matches
what is
 *          used by the user management to reset passwords.
 */
private static String encode(byte[] binaryData) {

    if (binaryData.length != 16) {
        return null;
    }

    char[] buffer = new char[32];

    for (int i = 0; i < 16; i++) {
        int low = (binaryData[i] & 0x0f);
        int high = ((binaryData[i] & 0xf0) >> 4);
        buffer[i * 2] = hexadecimal[high];
        buffer[i * 2 + 1] = hexadecimal[low];
    }
}

```

```

    }

    return new String(buffer);
}

}

```

identitylemmatizerrule.java

```

/*
 * $Id: IdentityLemmatizerRule.java,v 1.3 2008/12/15 06:36:00 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.lemmatizer;

import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordExcept
ion;

/**
 * The identity rule just searches for the given word as the lemma for
the given
 * part of speech and returns the word if it exists in the dictionary.
<br>
 * For example:<br>
 * saw + verb -> null<br>
 * saw + noun -> saw<br>
 * saws + noun -> null<br>
 *
 * @author ron
 */
public class IdentityLemmatizerRule extends
AbstractDictionaryLemmatizerRule {

    /**
     * @param dictionaryRepository
     */
    public IdentityLemmatizerRule(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }
}

```

```

/**
 * @see
edu.harvard.fas.rregan.nlp.LemmatizerRule#lemmatize(java.lang.String,
 *      edu.harvard.fas.rregan.nlp.PartOfSpeech)
 */
@Override
public String lemmatize(String word, PartOfSpeech partOfSpeech) {
    try {
        return getDictionaryRepository().findWord(word,
partOfSpeech).getLemma();
    } catch (NoSuchWordException e) {
        // no match
    }
    return null;
}

```

importdictionarycommand.java

```

/*
 * $Id $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.io.InputStream;

import edu.harvard.fas.rregan.command.Command;

/**
 * Import a dictionary of categories, synsets, words, and senses
 * from an XML file.
 *
 * @author ron
 */
public interface ImportDictionaryCommand extends Command {

    /**
     *
     * @param inputStream - the stream to the dictionary xml file.
     */
    public void setInputStream(InputStream inputStream);
}

```

importdictionarycommandimpl.java

```

/*
 * $Id: ImportDictionaryCommandImpl.java,v 1.1 2008/12/13 00:39:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.io.InputStream;

import javax.persistence.NoResultException;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.Dictionary;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.UnmarshallerListener;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportDictionaryCommand;

/**
 * @author ron
 */
@Controller("importDictionaryCommand")
@Scope("prototype")
public class ImportDictionaryCommandImpl extends
AbstractDictionaryCommand implements
ImportDictionaryCommand {

    private InputStream inputStream;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public ImportDictionaryCommandImpl(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }
}

```

```

/*
 * (non-Javadoc)
 *
 * @see
edu.harvard.fas.rregan.requel.dictionary.ImportDictionaryCommand#setIn
putStream(java.io.InputStream)
 */
public void setInputStream(InputStream inputStream) {
    this.inputStream = inputStream;
}

protected InputStream getInputStream() {
    return inputStream;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {
        // NOTE: the annotation classes need to be explicitly supplied to
        // the newInstance or an IllegalAnnotationExceptions will occur for
        // AbstractProjectOrDomainEntity.getAnnotations()
        JAXBContext context = JAXBContext
            .newInstance(ExportDictionaryCommandImpl.CLASSES_FOR_JAXB);
        Unmarshaller unmarshaller = context.createUnmarshaller();
        unmarshaller.setListener(new
UnmarshallerListener(getDictionaryRepository()));
        Dictionary dictionary = (Dictionary)
unmarshaller.unmarshal(getInputStream());
        for (Word word : dictionary.getWords()) {
            try {
                getDictionaryRepository().persist(word);
                // Word existingWord =
                // wordNetRepository.getWord(word.getLemma());
            } catch (NoResultException e) {
                getDictionaryRepository().persist(word);
            } catch (Exception e) {
                log.error("could not save word '" + word.getLemma() + "': " + e,
e);
            }
        }
    } catch (Exception e) {
        log.error(e, e);
    }
}

```

```

    }
}
```

importprojectcommand.java

```

/*
 * $Id: ImportProjectCommand.java,v 1.5 2009/02/13 12:08:06 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import java.io.InputStream;

import edu.harvard.fas.rregan.requel.command.AnalyzableEditCommand;
import edu.harvard.fas.rregan.requel.project.Project;

/**
 * Import the contents of a project to create a new project or to
update an
 * existing project from a supplied input stream.
 *
 * @author ron
 */
public interface ImportProjectCommand extends AnalyzableEditCommand {

    /**
     * @param project -
     *          the project to update from the contents of the input
stream.
     *          Leave null to create a new project.
     */
    public void setProject(Project project);

    /**
     * @return the created or updated project.
     */
    public Project getProject();

    /**
     * @param inputStream -
     *          the stream to read the project contents from.
     */
    public void setInputStream(InputStream inputStream);

    /**

```

```

 * @param name -
 *          over ride the name of the project being imported.
 */
public void setName(String name);
}

```

importprojectcommandimpl.java

```

/*
 * $Id: ImportProjectCommandImpl.java,v 1.23 2009/03/30 11:54:27
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.io.InputStream;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

import org.apache.commons.io.IOUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.command.EditReportGeneratorComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.ProjectImpl;
import edu.harvard.fas.rregan.requel.project.impl.StakeholderImpl;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.User;

```

```

import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Controller("importProjectCommand")
@Scope("prototype")
public class ImportProjectCommandImpl extends
AbstractEditProjectCommand implements
ImportProjectCommand {

private Project project;
private InputStream inputStream;
private String name;
private boolean analysisEnabled = false;

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 * @param projectCommandFactory
 * @param annotationCommandFactory
 * @param commandHandler
 */
@.Autowired
public ImportProjectCommandImpl(AssistantFacade assistantManager,
 UserRepository userRepository, ProjectRepository projectRepository,
 ProjectCommandFactory projectCommandFactory,
 AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
annotationCommandFactory, commandHandler);
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand#get
Project()
 */
@Override
public Project getProject() {
return project;
}
}

```

```

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand#set
Project(edu.harvard.fas.rregan.requel.project.Project)
 */
@Override
public void setProject(Project project) {
    this.project = project;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand#set
InputStream(java.io.InputStream)
 */
@Override
public void setInputStream(InputStream inputStream) {
    this.inputStream = inputStream;
}

protected InputStream getInputStream() {
    return inputStream;
}

protected String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

protected boolean isAnalysisEnabled() {
    return analysisEnabled;
}

public void setAnalysisEnabled(boolean analysisEnabled) {
    this.analysisEnabled = analysisEnabled;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    try {

```

```

        User createdBy = getUserRepository().get(getEditedBy());
        // NOTE: the annotation classes need to be explicitly supplied to
        // the newInstance or an IllegalAnnotationExceptions will occur for
        // AbstractProjectOrDomainEntity.getAnnotations()
        JAXBContext context = JAXBContext
            .newInstance(ExportProjectCommandImpl.CLASSES_FOR_JAXB);
        Unmarshaller unmarshaller = context.createUnmarshaller();
        unmarshaller.setListener(new
        UnmarshallerListener(getProjectRepository(),
            getUserRepository(), createdBy, getName()));
        ProjectImpl project = (ProjectImpl)
        unmarshaller.unmarshal(getInputStream());
        if (project.getCreatedBy() == null) {
            project.setCreatedBy(createdBy);
        }
        // add the user that imported the project as a stakeholder
        addUserAsStakeholder(project, createdBy, createdBy);

        try {
            // add the assistant as a stakeholder
            addUserAsStakeholder(project,
                getUserRepository().findUserByUsername("assistant"),
                createdBy);
        } catch (NoSuchUserException e) {
            log.warn("The assistant user doesn't exist and could not "
                + "be added as a stakeholder to " + project.getName());
        }
        setProject(getProjectRepository().persist(project));

        // TODO: use a command to edit each stakeholder?
        // add the project to all the stakeholder user ProjectUserRoles
        for (Stakeholder stakeholder : project.getStakeholders()) {
            if (stakeholder.isUserStakeholder()) {
                stakeholder.getUser().getRoleForType(ProjectUserRole.class).getAc
                tiveProjects()
                    .add(project);
            }
        }
        if (project.getReportGenerators().isEmpty()) {
            addBuiltinReportGenerator(project, createdBy);
        }
    } catch (Exception e) {
        log.error(e, e);
    }
}

```

@Override

```

public void invokeAnalysis() {
    if (isAnalysisEnabled()) {
        getAssistantManager().analyzeProject(getProject());
    }
}

private void addUserAsStakeholder(Project project, User user, User
editedBy) {
    // TODO: should this use the EditStakeholderCommand?
    if (user.hasRole(ProjectUserRole.class)) {
        boolean alreadyAStakeholder = false;
        for (Stakeholder stakeholder : project.getStakeholders()) {
            if (user.equals(stakeholder.getUser())) {
                alreadyAStakeholder = true;
                break;
            }
        }
        if (!alreadyAStakeholder) {
            Stakeholder creatorStakeholder = new StakeholderImpl(project,
            editedBy, user);
            getProjectRepository().persist(creatorStakeholder);
            for (StakeholderPermission permission : getProjectRepository()
                .findAvailableStakeholderPermissions()) {
                creatorStakeholder.grantStakeholderPermission(permission);
            }
            project.getStakeholders().add(creatorStakeholder);
        }
    }
}

private void addBuiltinReportGenerator(Project project, User user) {
    try {
        InputStream inputStream =
            getClass().getClassLoader().getResourceAsStream(
                EditProjectCommandImpl.BUILTIN_REPORT_GENERATOR_PATH);
        EditReportGeneratorCommand command = getProjectCommandFactory()
            .newEditReportGeneratorCommand();
        command.setEditedBy(user);
        command.setProjectOrDomain(project);
        command.setName("HTML Specification");
        command.setText(IOUtils.toString(inputStream));
        getCommandHandler().execute(command);
    } catch (Exception e) {
        log.error("The builtin report generator could not be added to " +
        project, e);
    }
}

```

```

    }

```

importsemcorcommand.java

```

/*
 * $Id: ImportSemcorCommand.java,v 1.1 2008/12/13 00:40:12 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.net.URL;

import edu.harvard.fas.rregan.command.Command;

/**
 * Import the Semcor data from the source files. Semcor is a
 * semantically
 * annotated corpus based on the Brown corpus.
 *
 * @author ron
 */
public interface ImportSemcorCommand extends Command {

    /**
     * @param baseURL -
     *          the url to the root semcor directory. This directory
     * should
     *          contain the
     */
    public void setBaseURL(URL baseURL);

}

```

importsemcorcommandimpl.java

```

/*
 * $Id: ImportSemcorCommandImpl.java,v 1.2 2008/12/15 06:35:57 rregan
 * Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.net.URL;

```

```

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.Category;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorFile;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentence;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentenceWord;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportSemcorCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordExcept
ion;
import edu.harvard.fas.rregan.nlp.impl.NLPTextImpl;
import edu.mit.jsemcor.element.IContext;
import edu.mit.jsemcor.element.IContextID;
import edu.mit.jsemcor.element.ISentence;
import edu.mit.jsemcor.element.IWordform;
import edu.mit.jsemcor.main.IConcordance;
import edu.mit.jsemcor.main.IConcordanceSet;
import edu.mit.jsemcor.main.Semcor;

/**
 * @author ron
 */
@Controller("importSemcorCommand")
@Scope("prototype")
public class ImportSemcorCommandImpl extends AbstractDictionaryCommand
implements
ImportSemcorCommand {

    private final NLPProcessor<NLPText> lemmatizer;

    private URL baseURL;

    /**
     * @param dictionaryRepository

```

```

        * @param lemmatizer
        */
    @Autowired
    public ImportSemcorCommandImpl(DictionaryRepository
dictionaryRepository,
        @Qualifier("lemmatizer")
        NLPProcessor<NLPText> lemmatizer) {
        super(dictionaryRepository);
        this.lemmatizer = lemmatizer;
    }

    protected NLPProcessor<NLPText> getLemmatizer() {
        return lemmatizer;
    }

    protected URL getBaseURL() {
        return baseURL;
    }

    public void setBaseURL(URL baseURL) {
        this.baseURL = baseURL;
    }

    protected Sense getSense(String lemma, String parseTag, int
senseRank)
        throws NoSuchWordException {
        PartOfSpeech pos = ParseTag.valueOf(parseTag).getPartOfSpeech();
        try {
            // the SemCor lemmas aren't always the lemma
            NLPText nlpLemma = getLemmatizer().process(new NLPTextImpl(lemma,
pos));
            Word dictionaryWord =
getDictionaryRepository().findWordExact(nlpLemma.getLemma(), pos);
            Set<Sense> posSenses = dictionaryWord.getSenses(pos);
            if (senseRank == 0) {
                if (posSenses.size() == 1) {
                    return posSenses.iterator().next();
                }
                // senseRank of 0 may mean no wordnet sense?
                return null;
            }
            for (Sense sense : posSenses) {
                if (sense.getRank() == senseRank) {
                    return sense;
                }
            }
        }

```

```

} catch (NoSuchWordException e) {
try {
if (lemma.contains("_")) {
    return getSense(lemma.replace('_', ' '), parseTag, senseRank);
}
} catch (Exception ee) {
log.error(e);
}
}
return null;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
try {
// construct the semcor object and open it
IConcordanceSet semcor = new Semcor(getBaseURL());
semcor.open();

for (IConcordance section : semcor.values()) {
    for (IContextID id : section.getContextIDs()) {
        IContext context = section.getContext(id);
        loadSemcorFile(section, context);
    }
}
} catch (Exception e) {
log.error(e, e);
}
}

protected void loadSemcorFile(IConcordance section, IContext context)
{
    SemcorFile file = getDictionaryRepository().persist(
        new SemcorFile(section.getName(), context.getFilename()));
    for (ISentence sentence : context.getSentences()) {
        SemcorSentence semSentence = getDictionaryRepository().persist(
            new SemcorSentence(file, new Long(sentence.getNumber())));
        for (IWordform wordform : sentence.getWordList()) {
            if (log.isDebugEnabled()) {
                log.debug("section: " + section.getName());
                log.debug("context: " + context.getFilename());
                if (wordform.getCommand() != null) {
                    log.debug("command: " + wordform.getCommand().getAttribute() +
" " +
+ wordform.getCommand().getData() + " " +
+ wordform.getCommand().getMeaning() + " " +
+ wordform.getCommand().getValue());
                }
                log.debug("data: " + wordform.getData());
                log.debug("distance: " + wordform.getDistance());
                log.debug("number: " + wordform.getNumber());
                log.debug("redefinition: " + wordform.getRedefinition());
                log.debug("separator string: " + wordform.getSeparatorString());
                log.debug("text: " + wordform.getText());
                log.debug("TokenIndex: " + wordform.getTokenIndex());
                log.debug("ConstituentTokens: " +
wordform.getConstituentTokens());
                if (wordform.getOtherTag() != null) {
                    log.debug("OtherTag: " + wordform.getOtherTag().getAttribute() +
" " +
+ wordform.getOtherTag().getData() + " " +
+ wordform.getOtherTag().getValue());
                }
                if (wordform.getPOSTag() != null) {
                    log.debug("POSTag: " + wordform.getPOSTag().getAttribute() + " " +
+ wordform.getPOSTag().getData() + " " +
+ wordform.getPOSTag().getMeaning() + " " +
+ wordform.getPOSTag().getValue());
                }
                if (wordform.getSemanticTag() != null) {
                    log.debug("SemanticTag: " + wordform.getSemanticTag().getLemma() +
" " +
+ wordform.getSemanticTag().getData() + " " +
+ wordform.getSemanticTag().getLexicalSense() + " } " +
+ wordform.getSemanticTag().getProperNounCategory() + " { " +
+ wordform.getSemanticTag().getSenseNumber() + " } ");
                }
            }
        ParseTag parseTag;
        try {
            parseTag = ParseTag.valueOf(wordform.getPOSTag().getValue());
        } catch (Exception e) {
            parseTag = ParseTag.X;
            log.warn("semcor pos tag error for " + section.getName() + " " +
+ context.getFilename() + "sentence " +
semSentence.getSentenceIndex() +
+ " word " + wordform.getNumber() + " " + wordform.getText() +
": " +
+ wordform.getPOSTag().getValue());
        }
        SemcorSentenceWord semWord = null;
    }
}
}

```

```

        if (wordform.getSemanticTag() == null) {
            semWord = new SemcorSentenceWord(semSentence,
                wordform.getNumber(), wordform
                    .getText(), parseTag);
        } else if (wordform.getSemanticTag().getProperNounCategory() != null) {
            String categorySimpleName =
                wordform.getSemanticTag().getProperNounCategory()
                    .getValue();
            Category properNounCategory =
                getDictionaryRepository().findCategory(
                    parseTag.getPartOfSpeech(), categorySimpleName);
            Sense sense = getSense(wordform.getSemanticTag().getLemma(),
                wordform
                    .getPOSTag().getValue(),
                wordform.getSemanticTag().getSenseNumber()
                    .get(0));
            semWord = new SemcorSentenceWord(semSentence,
                wordform.getNumber(), wordform
                    .getText(), ParseTag.valueOf(wordform.getPOSTag().getValue()),
                sense,
                    properNounCategory);
        } else {
            Sense sense = getSense(wordform.getSemanticTag().getLemma(),
                wordform
                    .getPOSTag().getValue(),
                wordform.getSemanticTag().getSenseNumber()
                    .get(0));
            if (wordform.getRedefinition() != null) {
                // there could be a typo in the text, replace it with
                // the redefined value
                semWord = new SemcorSentenceWord(semSentence,
                    wordform.getNumber(),
                        wordform.getRedefinition(),
                        ParseTag.valueOf(wordform.getPOSTag()
                            .getValue()), sense);
            } else {
                semWord = new SemcorSentenceWord(semSentence,
                    wordform.getNumber(),
                        wordform.getText(), ParseTag.valueOf(wordform.getPOSTag()
                            .getValue()), sense);
            }
        }
        semWord = getDictionaryRepository().persist(semWord);
        semSentence.getWords().add(semWord);
    }
}

```

```

        }
    }
}
```

importsemcorfilecommand.java

```

/*
 * $Id: ImportSemcorFileCommand.java,v 1.1 2008/12/13 00:40:07 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.command;

import edu.harvard.fas.rregan.command.Command;
import edu.mit.jsemcor.element.IContext;
import edu.mit.jsemcor.main.IConcordance;

/**
 * Import a specific Semcor data file from the source files. Semcor is
 * a semantically annotated corpus based on the Brown corpus.
 *
 * @author ron
 */
public interface ImportSemcorFileCommand extends Command {

    /**
     * @param section -
     *          the semcor section to load the file from.
     */
    public void setSection(IConcordance section);

    /**
     * @param context -
     *          the context (file) to load.
     */
    public void setContext(IContext context);
}
```

importsemcorfilecommandimpl.java

```

/*
 * $Id: ImportSemcorFileCommandImpl.java,v 1.1 2008/12/13 00:39:54
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.command.ImportSemcorFileCommand;
import edu.mit.jsemcor.element.IContext;
import edu.mit.jsemcor.main.IConcordance;

/**
 * @author ron
 */
@Controller("importSemcorFileCommand")
@Scope("prototype")
public class ImportSemcorFileCommandImpl extends
ImportSemcorCommandImpl implements
    ImportSemcorFileCommand {

    private IConcordance section;
    private IContext context;

    /**
     * @param dictionaryRepository
     * @param lemmatizer
     */
    @Autowired
    public ImportSemcorFileCommandImpl(DictionaryRepository
dictionaryRepository,
        @Qualifier("lemmatizer")
        NLPPProcessor<NLPText> lemmatizer) {
        super(dictionaryRepository, lemmatizer);
    }

    protected IConcordance getSection() {
        return section;
    }

    public void setSection(IConcordance section) {
        this.section = section;
    }
}

```

```

protected IContext getContext() {
    return context;
}

public void setContext(IContext context) {
    this.context = context;
}

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() {
    loadSemcorFile(getSection(), getContext());
}
}

```

initappevent.java

```

/*
 * $Id: InitAppEvent.java,v 1.2 2008/09/12 00:15:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.login;

import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event gets fired after a successful login by the
LoginOkController.
 * Controllers that initialize the UI and state of the application
should listen
 * for this event and configure the app.
 *
 * @author ron
 */
public class InitAppEvent extends NavigationEvent {
    static final long serialVersionUID = 0L;

    private Object user;

    /**
     * @param source
     */

```

```

 * @param user
 */
public InitAppEvent(Object source, Object user) {
    super(source, InitAppEvent.class.getName());
    setUser(user);
}

/**
 * If the app has user specific initialization, controllers that do
the
 * initialization can use the user attached to this event.
 *
 * @return
 */
public Object getUser() {
    return user;
}

protected void setUser(Object user) {
    this.user = user;
}
}

```

integerNLPTextWalker.java

```

/*
 * $Id: IntegerNLPTextWalker.java,v 1.1 2009/03/22 11:08:22 rregan Exp
$Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

/**
 * A driver for NLPTextWalkerFunction that returns an Integer value.
 *
 * @author ron
 */
public class IntegerNLPTextWalker extends
AbstractNLPTextWalker<Integer, Integer> {

/***
 * @param function -
 *          a NLPTextWalkerFunction that returns a StringBuilder
 */

```

```

    public IntegerNLPTextWalker(NLPTextWalkerFunction<Integer> function)
    {
        super(function);
    }

    /**
     * @see
edu.harvard.fas.rregan.nlp.impl.AbstractNLPTextWalker#transformResults
(java.lang.Object)
     */
    @Override
    public Integer transformResults(Integer result) {
        return result;
    }
}

```

invalidstateexceptionadapter.java

```

/*
 * $Id: InvalidStateExceptionAdapter.java,v 1.1 2008/12/13 00:41:17
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository.jpa;

import org.hibernate.validator.InvalidStateException;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.request.EntityValidationException;

/**
 * @author ron
 */
public class InvalidStateExceptionAdapter implements
EntityExceptionAdapter {

    /**
     * @see
edu.harvard.fas.rregan.repository.EntityExceptionAdapter#convert(java.
lang.Throwable,
     *          java.lang.Object,
     *          edu.harvard.fas.rregan.repository.EntityExceptionActionType)
     */
}

```

```

@Override
public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
    EntityExceptionActionType actionType) {
    return
EntityValidationException.validationFailed((InvalidStateException)
original,
    entityType, entity, actionType);
}

}

```

issue.java

```

/*
 * $Id: Issue.java,v 1.7 2009/02/23 09:39:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

import java.util.Date;
import java.util.Set;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public interface Issue extends Annotation {
    /**
     * @return The position options for resolving the issue.
     */
    public Set<Position> getPositions();

    /**
     * @return if the issue is resolved, the position used to resolve it.
     */
    public Position getResolvedByPosition();

    /**
     * @return if the issue is resolved, the user that resolve it.
     */
    public User getResolvedByUser();

    /**
     * @return if the issue is resolved, the date it was resolve.
     */
}

```

```

*/
public Date getResolvedDate();

/**
 * Resolve this issue with the given position and user. The resolved
date is
 * set to the current time.
 *
 * @param resolvedByPosition
 * @param resolvedByUser
 */
public void resolve(Position resolvedByPosition, User
resolvedByUser);

/**
 * Clear the resolution of the issue position, user and date.<br>
 * TODO: undo the action of the resolving position.
 */
public void unresolve();
}

```

issueeditorpanel.java

```

/*
 * $Id: IssueEditorPanel.java,v 1.39 2009/03/23 11:02:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.ColumnLayoutData;

```

```

import nextapp.echo2.app.layout.RowLayoutData;
import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;
import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.DeleteIssueCommand;
import edu.harvard.fas.rregan.requel.annotation.command.EditIssueCommand;
import edu.harvard.fas.rregan.requel.annotation.command.ResolveIssueCommand;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCellValueFactory;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColumnConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConfig;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import
edu.harvard.fas.rregan.uiframework.panel.editor.ToggleButtonModelEx;

/**
 * @author ron
 */
public class IssueEditorPanel extends
AbstractRequelAnnotationEditorPanel {
private static final Logger log =
Logger.getLogger(IssueEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the issue text field. If the property is undefined "Issue" is
used.
 */
public static final String PROP_LABEL_ISSUE = "Issue.Label";

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the issue must be resolved field. If the property is undefined
"Must
 * Be Resolved?" is used.
 */
public static final String PROP_LABEL_MUST_BE_RESOLVED =
"MustBeResolved.Label";

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the resolved by field. If the property is undefined "Resolved
By" is
 * used.
 */
public static final String PROP_LABEL_RESOLVED_BY =
"ResolvedBy.Label";

/**

```

```

 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the positions field. If the property is undefined "Positions"
is used.
 */
public static final String PROP_LABEL_POSITIONS = "Positions.Label";

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the view button in the position edit table column. If the
property is
 * undefined "View" is used.
 */
public static final String PROP_VIEW_POSITION_BUTTON_LABEL =
"ViewPosition.Label";

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the edit button in the position edit table column. If the
property is
 * undefined "Edit" is used.
 */
public static final String PROP_EDIT_POSITION_BUTTON_LABEL =
>EditPosition.Label;

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the button used to resolve the issue with the position. If the
 * property is undefined "Resolve" is used.
 */
public static final String PROP_RESOLVE_WITH_POSITION_BUTTON_LABEL =
"ResolvePosition.Label";

/**
 * The name to use in the IssueEditorPanel.properties file to set the
label
 * of the add button in the positions editor. If the property is
undefined
 * "Add" is used.
 */
public static final String PROP_ADD_POSITION_BUTTON_LABEL =
>AddPosition.Label;

private final AnnotationCommandFactory annotationCommandFactory;

```

```

private UpdateListener updateListener;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted = false;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public IssueEditorPanel(CommandHandler commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository annotationRepository) {
this(IssueEditorPanel.class.getName(), commandHandler,
annotationCommandFactory,
annotationRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public IssueEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository annotationRepository) {
super(resourceBundleName, Issue.class, commandHandler,
annotationRepository);
this.annotationCommandFactory = annotationCommandFactory;
}

/**
 * If the editor is editing an existing issue the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Edit Issue"<br>
 * For a new issue it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>

```

```

* "New Issue"<br>
*
* @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
* @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
* @see Panel.PROP_PANEL_TITLE
* @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    if (getIssue() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Edit Issue"));
        return MessageFormat.format(msgPattern, getIssue().toString());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "New Issue"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    String resolvedByText = "Unresolved";
    Issue issue = getIssue();
    if (issue != null) {
        addInput("text", PROP_LABEL_ISSUE, "Issue", new TextArea(), new
StringDocumentEx(issue
        .getText()));
        addInput("mustBeResolved", PROP_LABEL_MUST_BE_RESOLVED, "Must Be
Resolved?",
            new CheckBox(), new
ToggleButtonModelEx(issue.isMustBeResolved()));
        if (issue.getResolvedByPosition() != null) {
            resolvedByText = issue.getResolvedByPosition().getText();
        }
        addInput("resolvedBy", PROP_LABEL_RESOLVED_BY, "Resolved By", new
Label(),
            resolvedByText);
        addMultiRowInput("positions", PROP_LABEL_POSITIONS, "Positions",
getPositionsTable(),
            new NavigatorTableModel((Collection) issue.getPositions()));
    }
}

```

```

    addMultiRowInput("annotatables",
AnnotationRefererTable.PROP_ANNOTATABLES_LABEL,
    "Referring Entities", new AnnotationRefererTable(this,
        getResourceBundleHelper(getLocale()), issue));
} else {
    addInput("text", PROP_LABEL_ISSUE, "Issue", new TextArea(), new
StringDocumentEx());
    addInput("mustBeResolved", PROP_LABEL_MUST_BE_RESOLVED, "Must Be
Resolved?",
        new CheckBox(), new ToggleButtonModelEx(true));
    addInput("resolvedBy", PROP_LABEL_RESOLVED_BY, "Resolved By", new
Label(),
        resolvedByText);
    addMultiRowInput("positions", PROP_LABEL_POSITIONS, "Positions",
getPositionsTable(),
        new NavigatorTableModel(new TreeSet<Object>()));
    addMultiRowInput("annotatables",
AnnotationRefererTable.PROP_ANNOTATABLES_LABEL,
    "Referring Entities", new AnnotationRefererTable(this,
        getResourceBundleHelper(getLocale()), null));
}
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);

}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

private Component getPositionsTable() {
    ColumnLayoutData layoutData = new ColumnLayoutData();

```

```

layoutData.setAlignment(Alignment.ALIGN_CENTER);
Column col = new Column();
NavigatorTable relationTable = new
NavigatorTable(getPositionsTableConfig());
relationTable.setLayoutData(layoutData);
col.add(relationTable);
if ((getIssue() != null) && !isReadOnlyMode()) {
    String buttonLabel = null;
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_ADD_POSITION_BUTTON_LABEL, "Add");
    NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor,
    getIssue(), Position.class, null, WorkflowDisposition.NewFlow);
    NavigatorButton openEditorButton = new NavigatorButton(buttonLabel,
        getEventDispatcher(), openEditorEvent);
    openEditorButton.setStyleName(STYLE_NAME_DEFAULT);
    openEditorButton.setLayoutData(layoutData);
    col.add(openEditorButton);
}
return col;
}

private NavigatorTableConfig getPositionsTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", new
    NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Position position = (Position) model.getBackingObject(row);
            String buttonLabel = null;
            if (isReadOnlyMode()) {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_VIEW_POSITION_BUTTON_LABEL, "View");
            } else {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_EDIT_POSITION_BUTTON_LABEL, "Edit");
            }
            NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                PanelActionType.Editor, position, Position.class, null,
                WorkflowDisposition.NewFlow);
            NavigatorButton openEditorButton = new
            NavigatorButton(buttonLabel,
                getEventDispatcher(), openEditorEvent);
            openEditorButton.setStyleName(STYLE_NAME_PLAIN);
            RowLayoutData rld = new RowLayoutData();

```

```

rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
})
)));
}

if (!isReadOnlyMode()) {
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", new
    NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            final Position position = (Position)
model.getBackingObject(row);
            String buttonLabel =
getResourceBundleHelper(getLocale()).getString(
                PROP_RESOLVE_WITH_POSITION_BUTTON_LABEL, "Resolve");
            Button resolveButton = new Button(buttonLabel);
            resolveButton.addActionListener(new ResolveListener(
                IssueEditorPanel.this, position));
            resolveButton.setStyleName(STYLE_NAME_PLAIN);
            RowLayoutData rld = new RowLayoutData();
            rld.setAlignment(Alignment.ALIGN_CENTER);
            resolveButton.setLayoutData(rld);
            return resolveButton;
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Text", new
    NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Position position = (Position) model.getBackingObject(row);
            return position.getText();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By", new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Position position = (Position) model.getBackingObject(row);
            return position.getCreatedBy().getUsername();
        }
    }));
}

```

```

        }
    });

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Position position = (Position) model.getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            return formatter.format(position.getDateCreated());
        }
    });
}

return tableConfig;
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditIssueCommand command =
getAnnotationCommandFactory().newEditIssueCommand();
        command.setGroupingObject(getGroupingObject());
        command.setIssue(getIssue());
        command.setAnnotatable(getAnnotatable());
        command.setEditedBy(getCurrentUser());
        command.setText(getInputValue("text", String.class));
        command.setMustBeResolved(getInputValue("mustBeResolved",
Boolean.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
                updateListener);
        }
    }
}

```

```

        }
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getIssue())));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        } else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
            InvalidStateException ise = (InvalidStateException) e.getCause();
            for (InvalidValue invalidValue : ise.getInvalidValues()) {
                String propertyName = invalidValue.getPropertyName();
                setValidationMessage(propertyName, invalidValue.getMessage());
            }
        } else {
            setGeneralMessage(e.toString());
        }
    } catch (Exception e) {
        log.error("could not save the goal: " + e, e);
        setGeneralMessage("Could not save: " + e);
    }
}

@Override
public void delete() {
    super.delete();
    try {
        Issue issue = getIssue();
        Set<Position> positions = getIssue().getPositions();
        DeleteIssueCommand deleteIssueCommand =
getAnnotationCommandFactory()
            .newDeleteIssueCommand();
        deleteIssueCommand.setIssue(issue);
        deleteIssueCommand.setEditedBy(getCurrentUser());
        deleteIssueCommand =
getCommandHandler().execute(deleteIssueCommand);
        deleted = true;
        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
issue));
        for (Position position : positions) {
            getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
position));
        }
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

```

```

}

private Issue getIssue() {
    if (getTargetObject() instanceof Issue) {
        return (Issue) getTargetObject();
    }
    return null;
}

private AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

private static class ResolveListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final IssueEditorPanel panel;
    private final Position position;

    private ResolveListener(IssueEditorPanel panel, Position position) {
        this.panel = panel;
        this.position = position;
    }

    public void actionPerformed(ActionEvent event) {
        try {
            ResolveIssueCommand command = panel.getAnnotationCommandFactory()
                .newResolveIssueCommand(position);
            command.setIssue(panel.getIssue());
            command.setPosition(position);
            command.setEditedBy(panel.getCurrentUser());
            command = panel.getCommandHandler().execute(command);
            Issue issue = command.getIssue();
            panel.getEventDispatcher().dispatchEvent(new
UpdateEntityEvent(panel, issue));
        } catch (Exception e) {
            log.error("Exception resolving issue " + position.getIssues() + "
with position "
                + position, e);
            panel.setGeneralMessage(e.toString());
        }
    }
}

// TODO: it may be better to have different standardized update
listeners

// for different types of updated or deleted objects associated with
the
// input controls like an annotatables table.
private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final IssueEditorPanel panel;

    private UpdateListener(IssueEditorPanel panel) {
        this.panel = panel;
    }

    public void actionPerformed(ActionEvent e) {
        try {
            if (panel.deleted) {
                return;
            }
            Issue existingIssue = panel.getIssue();
            if ((e instanceof UpdateEntityEvent) && (existingIssue != null)) {
                UpdateEntityEvent event = (UpdateEntityEvent) e;
                Issue updatedIssue = null;
                if (event.getObject() instanceof Issue) {
                    updatedIssue = (Issue) event.getObject();
                    if ((event instanceof DeletedEntityEvent)
                        && existingIssue.equals(updatedIssue)) {
                        panel.deleted = true;
                        panel.getEventDispatcher().dispatchEvent(
                            new DeletedEntityEvent(this, panel, existingIssue));
                        return;
                    }
                } else if (event.getObject() instanceof Position) {
                    Position updatedPosition = (Position) event.getObject();
                    if (event instanceof DeletedEntityEvent) {
                        existingIssue.getPositions().remove(updatedPosition);
                        updatedIssue = existingIssue;
                    } else if
(updatedPosition.getIssues().contains(panel.getIssue())) {
                        for (Issue positionIssue : updatedPosition.getIssues()) {
                            if (positionIssue.equals(existingIssue)) {
                                updatedIssue = positionIssue;
                                break;
                            }
                        }
                    }
                } else if ((event instanceof DeletedEntityEvent)
                    && (event.getObject() instanceof Annotatable)) {
                    Annotatable annotatable = (Annotatable) event.getObject();
                }
            }
        }
    }
}

```

```
        if (existingIssue.getAnnotatables().contains(annotatable)) {
            existingIssue.getAnnotatables().remove(annotatable);
        }
        updatedIssue = existingIssue;
    }
    if ((updatedIssue != null) && updatedIssue.equals(existingIssue))
    {
        panel.setTargetObject(updatedIssue);
        panel.setInputValue("text", updatedIssue.getText());
        panel.setInputValue("mustBeResolved",
        updatedIssue.isMustBeResolved());
        if (updatedIssue.getResolvedByPosition() != null) {
            panel.setInputValue("resolvedBy",
            updatedIssue.getResolvedByPosition()
            .getText());
        }
        panel.setInputValue("annotatables", updatedIssue);
        panel.setInputValue("positions", updatedIssue.getPositions());
    }
}
} catch (Exception ex) {
    log.error("Exception processing update.", ex);
}
}
```

issueimpl.java

```
/*
 * $Id: IssueImpl.java,v 1.23 2009/02/23 09:39:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import java.util.Date;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
```

```
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.xml.sax.SAXException;

import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl.Position2Po
sitionImplAdapter;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.Issue")
@XmlRootElement(name = "issue", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "issue", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class IssueImpl extends AbstractAnnotation implements Issue {
    static final long serialVersionUID = 0L;

    private Set<Position> positions = new TreeSet<Position>();
    private boolean mustBeResolved;
```

```

private Position resolvedByPosition;
private User resolvedByUser;
private Date resolvedDate;

/**
 * @param groupingObject -
 *      An object used as the "owner" of a group of
annotations.
 * @param text -
 *      the text message of the annotation
 * @param mustBeResolved -
 *      flag indicating if this issue must be resolved before
the
 *      project can be complete.
 * @param createdBy -
 *      the user that created the issue
 */
public IssueImpl(Object groupingObject, String text, boolean
mustBeResolved, User createdBy) {
    this(Issue.class.getName(), groupingObject, text, mustBeResolved,
createdBy);
}

/**
 * @param type -
 *      the class name of this annotation
 * @param groupingObject -
 *      An object used as the "owner" of a group of
annotations.
 * @param text -
 *      the text message of the annotation
 * @param mustBeResolved -
 *      flag indicating if this issue must be resolved before
the
 *      project can be complete.
 * @param createdBy -
 *      the user that created the issue
 */
protected IssueImpl(String type, Object groupingObject, String text,
boolean mustBeResolved,
User createdBy) {
    super(type, groupingObject, text, createdBy);
    setMustBeResolved(mustBeResolved);
}

protected IssueImpl() {
    // for hibernate
}

}

@Transient
public String getTypeName() {
    return "Issue";
}

// changed xml mapping to output references to positions instead of
the
// positions directly because a position may be shared by multiple
// issues causing duplicates on import. this makes report generating
via
// xslt more complicated because of the indirection.
@XmlElementWrapper(name = "positions", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel", required = false)
@XmlIDREF
@XmlElement(name = "positionRef", type = PositionImpl.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = PositionImpl.class, mappedBy = "issues",
cascade = {
    CascadeType.MERGE, CascadeType.PERSIST, CascadeType.REFRESH },
fetch = FetchType.LAZY)
@JoinTable(name = "position_issue", joinColumns = { @JoinColumn(name =
"issue_id") }, inverseJoinColumns = { @JoinColumn(name =
"position_id") })
public Set<Position> getPositions() {
    return positions;
}

protected void setPositions(Set<Position> positions) {
    this.positions = positions;
}

@XmlAttribute(name = "mustBeResolved")
public boolean isMustBeResolved() {
    return mustBeResolved;
}

/**
 * @param mustBeResolved
 */
public void setMustBeResolved(boolean mustBeResolved) {
    this.mustBeResolved = mustBeResolved;
}

@Transient
public boolean isResolved() {
}

```

```

    return (resolvedByPosition != null);
}

@Transient
public String getStatusMessage() {
    if (resolvedByPosition != null) {
        return "Resolution: " + resolvedByPosition.getText();
    }
    return "Unresolved";
}

@ManyToOne(targetEntity = PositionImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, optional = true)
@XmlIDREF
@XmlAttribute(name = "resolvedByPosition")
@XmlJavaTypeAdapter(Position2PositionImplAdapter.class)
public Position getResolvedByPosition() {
    return resolvedByPosition;
}

protected void setResolvedByPosition(Position resolvedByPosition) {
    this.resolvedByPosition = resolvedByPosition;
}

@Override
public void resolve(Position resolvedByPosition, User resolvedByUser)
{
    setResolvedByPosition(resolvedByPosition);
    setResolvedByUser(resolvedByUser);
    setResolvedDate(new Date());
}

@Override
public void unresolve() {
    setResolvedByPosition(null);
    setResolvedByUser(null);
    setResolvedDate(null);
}

@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, optional = true)
@XmlIDREF
@XmlAttribute(name = "resolvedByUser")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
public User getResolvedByUser() {
    return resolvedByUser;
}

protected void setResolvedByUser(User resolvedByUser) {
    this.resolvedByUser = resolvedByUser;
}

@XmlAttribute(name = "dateResolved")
@XmlJavaTypeAdapter(DateAdapter.class)
@Temporal(TemporalType.TIMESTAMP)
public Date getResolvedDate() {
    return resolvedDate;
}

protected void setResolvedDate(Date resolvedDate) {
    this.resolvedDate = resolvedDate;
}

/**
 * This is for JAXB to patchup the parent/child relationship and swap
the
 * resolved by user with an existing user.
 *
 * @param userRepository
 * @param annotatable
 * @see UnmarshallerListener
 */
public void afterUnmarshal(final UserRepository userRepository) {
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {
                if (getResolvedByUser() != null) {
                    User existingUser =
userRepository.findUserByUsername(getResolvedByUser()
.getUsername());
                    setResolvedByUser(existingUser);
                }
                // fix up the positions reference to the issue
                for (Position position : getPositions()) {
                    position.getIssues().add(IssueImpl.this);
                }
            } catch (NoSuchUserException e) {
                // the new user will be persisted automatically
            }
        }
    });
}

```

```
}
```

jaxbannotatablepatcher.java

```
/*
 * $Id: JAXBAnnotatablePatcher.java,v 1.2 2009/01/07 09:50:37 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;

/**
 * This is used to add an annotatable to all the annotations after
import.
 *
 * @author ron
 */
public class JAXBAnnotatablePatcher implements Patcher {

    private final Annotatable annotatable;

    /**
     * @param annotation
     * @param annotatable
     */
    public JAXBAnnotatablePatcher(Annotatable annotatable) {
        this.annotatable = annotatable;
    }

    @Override
    public void run() throws SAXException {
        try {
            for (Annotation annotation : annotatable.getAnnotations()) {
                annotation.getAnnotatables().add(annotatable);
            }
        } catch (RuntimeException e) {
            throw e;
        }
    }
}
```

```
        } catch (Exception e) {
            throw new SAXException2(e);
        }
    }
}
```

jaxbannotationgroupedbypatcher.java

```
/*
 * $Id: JAXBAnnotationGroupedByPatcher.java,v 1.3 2009/01/07 09:50:38
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.impl.AbstractAnnotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;

/**
 * This is used to set the grouping object
 *
 * @author ron
 */
public class JAXBAnnotationGroupedByPatcher implements Patcher {

    private final Annotation annotation;
    private final Annotatable annotatable;

    /**
     * @param annotation
     * @param annotatable
     */
    public JAXBAnnotationGroupedByPatcher(Annotation annotation,
                                         Annotatable annotatable) {
        this.annotatable = annotatable;
        this.annotation = annotation;
    }
}
```

```

@Override
public void run() throws SAXException {
    try {
        // annotations expect the annotatable object and
        // the group object, which should be the
        // project.
        if (annotatable instanceof ProjectOrDomain) {
            ((AbstractAnnotation) annotation).setGroupingObject(annotatable);
        } else if (annotatable instanceof ProjectOrDomainEntity) {
            ((AbstractAnnotation) annotation)
                .setGroupingObject(((ProjectOrDomainEntity) annotatable)
                    .getProjectOrDomain());
        }
    } catch (RuntimeException e) {
        throw e;
    } catch (Exception e) {
        throw new SAXException2(e);
    }
}

```

jaxbcreatedentitypatcher.java

```

/*
 * $Id: JAXBCreatedEntityPatcher.java,v 1.5 2009/01/07 09:50:37 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
/***

```

```

 * This is used to swap the createdBy User objects on CreatedEntity
objects
 * assigned by the JAXB Unmarshaller with existing users from the
database.
 *
 * @author ron
 */
public class JAXBCreatedEntityPatcher implements Patcher {

    private final CreatedEntity createdEntity;
    private final UserRepository userRepository;
    private final User defaultCreatedByUser;

    /**
     * @param userRepository
     * @param createdEntity
     * @param defaultCreatedByUser
     */
    public JAXBCreatedEntityPatcher(UserRepository userRepository,
CreatedEntity createdEntity,
        User defaultCreatedByUser) {
        this.userRepository = userRepository;
        this.createdEntity = createdEntity;
        this.defaultCreatedByUser = defaultCreatedByUser;
    }

    @Override
    public void run() throws SAXException {
        try {
            if (createdEntity.getCreatedBy() != null) {
                try {
                    User existingUser =
userRepository.findUserByUsername(createdEntity
                        .getCreatedBy().getUsername());
                    setCreatedBy(createdEntity, existingUser);
                } catch (NoSuchUserException e) {
                    // new user
                    // TODO: the following causes an exception because the
                    // JAXBUserRolePermissionPatcher hasn't run yet to map
                    // the persistant permissions to the ones created by
                    // the xml. Solved by making references to User in
                    // implementors of CreatedEntity cascade
                    // userRepository.persist(createdEntity.getCreatedBy());
                }
            } else {
                setCreatedBy(createdEntity, defaultCreatedByUser);
            }
        }
    }
}

```

```
        } catch (RuntimeException e) {
            throw e;
        } catch (Exception e) {
            throw new SAXException2(e);
        }
    }

private void setCreatedBy(CreatedEntity createdEntity, User user)
    throws IllegalAccessException, InvocationTargetException {
    Class<?> entityType = createdEntity.getClass();
    while (entityType != null) {
        try {
            Method createdBySetter =
                entityType.getDeclaredMethod("setCreatedBy", User.class);
            createdBySetter.setAccessible(true);
            createdBySetter.invoke(createdEntity, user);
            return;
        } catch (NoSuchMethodException e) {
            // try super class
            entityType = entityType.getSuperclass();
        }
    }
}
```

jaxborganizedentitypatcher.java

```
/*
 * $Id: JAXBOrganizedEntityPatcher.java,v 1.4 2009/01/07 09:50:38
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;

import edu.harvard.fas.rregan.requel.OrganizedEntity;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchOrganizationExcepti
on;
```

```
/**  
 * This is used to swap the organization objects on CreatedEntity  
objects  
 * assigned by the JAXB Unmarshaller with existing users from the  
database.  
 *  
 * @author ron  
 */  
public class JAXBOrganizedEntityPatcher implements Patcher {  
  
    private final OrganizedEntity organizedEntity;  
    private final UserRepository userRepository;  
  
    /**  
     * @param userRepository  
     * @param organizedEntity  
     */  
    public JAXBOrganizedEntityPatcher(UserRepository userRepository,  
OrganizedEntity organizedEntity) {  
        this.userRepository = userRepository;  
        this.organizedEntity = organizedEntity;  
    }  
  
    @Override  
    public void run() throws SAXException {  
        try {  
            if (organizedEntity.getOrganization() != null) {  
                try {  
                    Organization existingOrganization = userRepository  
                        .findOrganizationByName(organizedEntity.getOrganization().getNa  
me());  
                    organizedEntity.setOrganization(existingOrganization);  
                } catch (NoSuchOrganizationException e) {  
                    // new organization  
                    organizedEntity.setOrganization(userRepository.persist(organizedE  
ntity  
                        .getOrganization()));  
                }  
            } catch (RuntimeException e) {  
                throw e;  
            } catch (Exception e) {  
                throw new SAXException2(e);  
            }  
        }  
    }  
}
```

jaxbuserrolepatcher.java

```
/*
 * $Id: JAXBUserRolePatcher.java,v 1.1 2009/01/07 09:50:37 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;

import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.UserRole;
import edu.harvard.fas.rregan.requel.user.UserRolePermission;

/**
 * This is used to swap the UserRolePermission object on UserRole
 * objects
 * assigned by the JAXB Unmarshaller with existing permissions from
 * the
 * database. It also patches the user
 *
 * @author ron
 */
public class JAXBUserRolePatcher implements Patcher {

    private final UserRole userRole;
    private final UserRepository userRepository;

    /**
     * @param userRepository
     * @param userRole
     */
    public JAXBUserRolePatcher(UserRepository userRepository, UserRole
        userRole) {
        this.userRepository = userRepository;
        this.userRole = userRole;
    }

    @Override
    public void run() throws SAXException {
        try {

```

```
            for (UserRolePermission permission :
userRepository.findUserRolePermissions(userRole
                .getClass())) {
                if (userRole.hasUserRolePermission(permission)) {
                    userRole.revokeUserRolePermission(permission);
                    userRole.grantUserRolePermission(permission);
                }
            }
        } catch (RuntimeException e) {
            throw e;
        } catch (Exception e) {
            throw new SAXException2(e);
        }
    }
}
```

jpaannotationrepository.java

```
/*
 * $Id: JpaAnnotationRepository.java,v 1.17 2009/01/09 09:56:17 rregan
 * Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import javax.persistence.NoResultException;
import javax.persistence.OptimisticLockException;
import javax.persistence.Query;

import org.hibernate.PropertyValueException;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.LockAcquisitionException;
import org.hibernate.validator.InvalidStateException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.dao.CannotAcquireLockException;
import org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureException;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository;
```

```

import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;
import edu.harvard.fas.rregan.repository.jpa.GenericPropertyValueExceptionAdapter;
import edu.harvard.fas.rregan.repository.jpa.InvalidStateExceptionAdapter;
import edu.harvard.fas.rregan.repository.jpa.OptimisticLockExceptionAdapter;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationExistsException;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.annotation.Position;

/**
 * EJB3/JPA based repository
 *
 * @author ron
 */
@Repository("annotationRepository")
@Scope("singleton")
@Transactional(propagation = Propagation.REQUIRED, noRollbackFor =
{ NoSuchPositionException.class,
  NoSuchAnnotationException.class, AnnotationExistsException.class,
  EntityException.class })
public class JpaAnnotationRepository extends AbstractJpaRepository
implements AnnotationRepository {

    /**
     * @param exceptionMapper
     */
    @Autowired
    public JpaAnnotationRepository(ExceptionMapper exceptionMapper) {
        super(exceptionMapper);
        addExceptionAdapter(PropertyValueExceptionAdapter.class, Position.class,
                           Issue.class, Note.class, Argument.class);
    }
}

```

```

        addExceptionAdapter(InvalidStateException.class, new
                           InvalidStateExceptionAdapter(),
                           Position.class, Issue.class, Note.class, Argument.class);

        addExceptionAdapter(OptimisticLockException.class, new
                           OptimisticLockExceptionAdapter(),
                           Position.class, Issue.class, Note.class, Argument.class);

        addExceptionAdapter(StaleObjectStateException.class, new
                           OptimisticLockExceptionAdapter(),
                           Position.class, Issue.class, Note.class, Argument.class);

        addExceptionAdapter(LockAcquisitionException.class, new
                           OptimisticLockExceptionAdapter(),
                           Position.class, Issue.class, Note.class, Argument.class);

        addExceptionAdapter(CannotAcquireLockException.class, new
                           OptimisticLockExceptionAdapter(),
                           Position.class, Issue.class, Note.class, Argument.class);

        addExceptionAdapter(HibernateOptimisticLockingFailureException.class
                           ,
                           new OptimisticLockExceptionAdapter(), Position.class, Issue.class,
                           Note.class,
                           Argument.class);
    }

    @Override
    public Position findPosition(Object groupingObject, String text)
        throws NoSuchPositionException {
        try {
            // TODO: use named query so it can be configured externally
            Query query = getEntityManager().createQuery(
                "select object(position) from PositionImpl as position "
                + "inner join position.issues issue where position.text like
:text "
                + "and issue.groupingObject = :groupingObject");
            query.setParameter("groupingObject", groupingObject);
            query.setParameter("text", text);
            return (Position) query.getSingleResult();
        } catch (NoResultException e) {
            throw NoSuchPositionException.makeText(text);
        } catch (Exception e) {
            throw convertException(e, Position.class, null,
                                  EntityExceptionActionType.Reading);
        }
    }
}

```

```

public AddWordToDictionaryPosition
findAddWordToDictionaryPosition(Object groupingObject,
    String word) throws NoSuchPositionException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(position) from AddWordToDictionaryPosition as
position "
        + "inner join position.issues issue where issue.word like :word
"
        + "and issue.groupingObject = :groupingObject");
    query.setParameter("groupingObject", groupingObject);
    query.setParameter("word", word);
    return (AddWordToDictionaryPosition) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchPositionException.forAddingWordToDictionary(word);
} catch (Exception e) {
    throw convertException(e, Position.class, null,
EntityExceptionActionType.Reading);
}
}

/**
 * @see
edu.harvard.fas.rregan.requel.annotation.AnnotationRepository#findChan
geSpellingPosition(edu.harvard.fas.rregan.requel.annotation.Issue,
    *      java.lang.String)
 */
@Override
public ChangeSpellingPosition findChangeSpellingPosition(LexicalIssue
issue, String proposedWord)
    throws NoSuchPositionException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(position) from ChangeSpellingPosition as position
"
        + "inner join position.issues issue "
        + "where position.proposedWord = :proposedWord "
        + "and issue.groupingObject = :groupingObject");
    query.setParameter("groupingObject", issue.getGroupingObject());
    query.setParameter("proposedWord", proposedWord);
    return (ChangeSpellingPosition) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchPositionException.forChangeSpelling(issue,
proposedWord);
}
}

    } catch (Exception e) {
        throw convertException(e, Position.class, null,
EntityExceptionActionType.Reading);
    }
}

@Override
public Issue findIssue(Object groupingObject, Annotatable
annotatable, String message) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(issue) from IssueImpl as issue "
        + "where issue.text like :message "
        + "and issue.groupingObject = :groupingObject");
    query.setParameter("groupingObject", groupingObject);
    query.setParameter("message", message);
    return (Issue) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchAnnotationException.forMessage(message);
} catch (Exception e) {
    throw convertException(e, Issue.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public LexicalIssue findLexicalIssue(Object groupingObject,
Annotatable annotatable, String word)
throws NoSuchAnnotationException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(issue) from LexicalIssue as issue "
        + "where issue.word like :word "
        + "and issue.groupingObject = :groupingObject "
        + "and issue.annotatablePropertyName is null");
    query.setParameter("word", word);
    query.setParameter("groupingObject", groupingObject);
    return (LexicalIssue) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchAnnotationException.forWord(word, null);
} catch (Exception e) {
    throw convertException(e, Issue.class, null,
EntityExceptionActionType.Reading);
}
}
}

```

```

@Override
public LexicalIssue findLexicalIssue(Object groupingObject,
Annotatable annotatable,
String word, String annotatableEntityPropertyName) throws
NoSuchAnnotationException {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager()
.createQuery(
"select object(issue) from LexicalIssue as issue "
+ "where issue.word like :word "
+ "and issue.groupingObject = :groupingObject "
+ "and issue.annotatableEntityPropertyName like
:annotatableEntityPropertyName");
query.setParameter("word", word);
query.setParameter("groupingObject", groupingObject);
query.setParameter("annotatableEntityPropertyName",
annotatableEntityPropertyName);
return (LexicalIssue) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchAnnotationException.forWord(word,
annotatableEntityPropertyName);
} catch (Exception e) {
throw convertException(e, Issue.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public Note findNote(Object groupingObject, Annotatable annotatable,
String message) {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager()
.createQuery(
"select object(note) from NoteImpl as note "
+ "where note.text like :message and note.groupingObject =
:groupingObject");
query.setParameter("groupingObject", groupingObject);
query.setParameter("message", message);
// query.setParameter("annotatable", annotatable);
return (Note) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchAnnotationException.forMessage(message);
} catch (Exception e) {

```

```

        throw convertException(e, Note.class, null,
EntityExceptionActionType.Reading);
    }
}
}

```

jpadictionaryrepository.java

```

/*
 * $Id: JpaDictionaryRepository.java,v 1.7 2009/03/27 07:16:08 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.jpa;

import java.io.File;
import java.io.IOException;
import java.math.BigInteger;
import java.net.URI;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.persistence.NoResultException;
import javax.persistence.OptimisticLockException;
import javax.persistence.Query;

import org.hibernate.PropertyValueException;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.LockAcquisitionException;
import org.hibernate.validator.InvalidStateException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.dao.CannotAcquireLockException;
import
org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureEx
ception;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

```

```

import com.swabunga.spell.engine.DoubleMeta;
import com.swabunga.spell.engine.SpellDictionary;
import com.swabunga.spell.engine.SpellDictionaryHashMap;
import com.swabunga.spell.engine.Translator;
import com.swabunga.spell.event.SpellChecker;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.Category;
import edu.harvard.fas.rregan.nlp.dictionary.Dictionary;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Lexlinkref;
import edu.harvard.fas.rregan.nlp.dictionary.LexlinkrefId;
import edu.harvard.fas.rregan.nlp.dictionary.Linkdef;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorFile;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentence;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentenceWord;
import edu.harvard.fas.rregan.nlp.dictionary.Semlinkref;
import edu.harvard.fas.rregan.nlp.dictionary.SemlinkrefId;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import
edu.harvard.fas.rregan.nlp.dictionary.VerbNetSelectionRestrictionType;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.NoSuchWordExcept
ion;
import edu.harvard.fas.rregan.nlp.impl.DatabaseSpellDictionary;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository;
import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;
import
edu.harvard.fas.rregan.repository.jpa.GenericPropertyValueExceptionAda
pter;
import
edu.harvard.fas.rregan.repository.jpa.InvalidStateExceptionAdapter;
import
edu.harvard.fas.rregan.repository.jpa.OptimisticLockExceptionAdapter;
import edu.harvard.fas.rregan.request.NoSuchEntityException;

/**
 * EJB3/JPA based repository
 *
 * @author ron
 */
@Repository("dictionaryRepository")

```

```

@Scope("singleton")
@Transactional(propagation = Propagation.REQUIRED, noRollbackFor =
{ NoResultException.class,
  NoSuchEntityException.class, EntityException.class })
public class JpaDictionaryRepository extends AbstractJpaRepository
implements DictionaryRepository {

    /**
     * The name of the property in the DictionaryTool.properties file
     * that
     * contains a pipe "|" delimited list of paths to dictionary files
     * relative
     * to the classpath.<br>
     * the simplest dictionary is "resources/nlp/jazzy/english.0"
     *
     * @see PROP_ENGLISH_DICTIONARY_FILES_DEFAULT for the default paths
     * of the
     * files
     */
    public static final String PROP_ENGLISH_DICTIONARY_FILES =
"EnglishDictionaryFiles";

    /**
     * The default path to the tokenizer model file
     */
    public static final String PROP_ENGLISH_DICTIONARY_FILES_DEFAULT =
"resources/nlp/jazzy/eng_com.dic|"
    + "resources/nlp/jazzy/center.dic|"
    + "resources/nlp/jazzy/color.dic|"
    + "resources/nlp/jazzy/ize.dic|"
    + "resources/nlp/jazzy/labeled.dic|"
    + "resources/nlp/jazzy/yze.dic";

    private static final Collection<SpellDictionary> staticDictionaries;
    static {
        try {
            List<SpellDictionary> dictionaries = new
ArrayList<SpellDictionary>(10);
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                JpaDictionaryRepository.class.getName());

            String dictionaryFilePaths = resourceBundleHelper.getString(
                PROP_ENGLISH_DICTIONARY_FILES,
                PROP_ENGLISH_DICTIONARY_FILES_DEFAULT);

            File dictionaryFile;

```

```

if (dictionaryFilePaths.contains("|")) {
    for (String dictionaryFilePath : dictionaryFilePaths.split("\\|"))
    {
        if (!"".equals(dictionaryFilePath.trim())) {
            log.debug("loading dictionary: " + dictionaryFilePath);
            URI dictionaryFileURI =
JpaDictionaryRepository.class.getClassLoader()
                .getResource(dictionaryFilePath).toURI();
            dictionaryFile = new File(dictionaryFileURI);
            // TODO: try using SpellDictionaryDisk for less
            // memory usage than SpellDictionaryHashMap
            dictionaries.add(new SpellDictionaryHashMap(dictionaryFile));
        }
    }
} else {
    URI dictionaryFileURI =
JpaDictionaryRepository.class.getClassLoader().getResource(
    dictionaryFilePaths).toURI();
    dictionaryFile = new File(dictionaryFileURI);
    // TODO: try using SpellDictionaryDisk for less memory usage
    // than SpellDictionaryHashMap
    dictionaries.add(new SpellDictionaryHashMap(dictionaryFile));
}
staticDictionaries =
Collections.unmodifiableCollection(dictionaries);
} catch (Exception e) {
    throw new ExceptionInInitializerError(e);
}
}

private final Transformator jazzyTransformator = new DoubleMeta();
private SpellChecker spellChecker = new SpellChecker();

/**
 * @param exceptionMapper
 */
@.Autowired
public JpaDictionaryRepository(ExceptionMapper exceptionMapper) {
    super(exceptionMapper);
    spellChecker = new SpellChecker();
    for (SpellDictionary dictionary : staticDictionaries) {
        spellChecker.addDictionary(dictionary);
    }
try {
    // Passing null as the phonetic code file causes the dictionary to
    // use the DoubleMeta Transformator to create phentic codes for
    // spelling corrections.
    spellChecker.setUserDictionary(new DatabaseSpellDictionary(this,
(File) null));
} catch (IOException e) {
    // This will never happen because null is passed as the phonetic
    // code file.
}

addExceptionAdapter(PropertyValueException.class,
    new GenericPropertyValueExceptionAdapter(), Word.class,
Category.class,
    Synset.class);

addExceptionAdapter(InvalidStateException.class, new
InvalidStateExceptionAdapter(),
    Word.class, Category.class, Synset.class);

addExceptionAdapter(OptimisticLockException.class, new
OptimisticLockExceptionAdapter(),
    Word.class, Category.class, Synset.class);

addExceptionAdapter(StaleObjectStateException.class, new
OptimisticLockExceptionAdapter(),
    Word.class, Category.class, Synset.class);

addExceptionAdapter(LockAcquisitionException.class, new
OptimisticLockExceptionAdapter(),
    Word.class, Category.class, Synset.class);

addExceptionAdapter(CannotAcquireLockException.class, new
OptimisticLockExceptionAdapter(),
    Word.class, Category.class, Synset.class);

addExceptionAdapter(HibernateOptimisticLockingFailureException.class
,
    new OptimisticLockExceptionAdapter(), Word.class, Category.class,
Synset.class);
}

protected SpellChecker getSpellChecker() {
    return spellChecker;
}

public Boolean isKnownWord(String word) {
    if (isNumber(word)) {
        return Boolean.TRUE;
    }
    return getSpellChecker().isCorrect(word);
}

```

```

}

public List<String> findSpellingSuggestions(String word, int
threshold) {
    List<String> results = new ArrayList<String>();
    boolean containsDigits = containsDigits(word);

    for (Object obj : getSpellChecker().getSuggestions(word, threshold))
    {
        com.swabunga.spell.engine.Word jazzyWord =
        (com.swabunga.spell.engine.Word) obj;
        // if the word doesn't contain any digits, don't include numbers in
        // the suggestions
        if (!containsDigits && isNumber(jazzyWord.getWord())) {
            continue;
        }
        results.add(jazzyWord.getWord());
    }
    return results;
}

@Override
public Set<Sense> findMoreSpecificWords(Sense sense, int
maxSuggestions) {
    sense = get(sense);
    Set<Sense> suggestions = new HashSet<Sense>();
    // map of senses by the number of senses of its word
    Map<Integer, List<Sense>> senseByWordSenseCount = new
    HashMap<Integer, List<Sense>>();
    for (Synset hyponym : findHyponyms(sense.getSynset(), 1)) {
        // use the senses of the words with the least number of senses
        for (Sense s : hyponym.getSenses()) {
            int senseCount =
                s.getWord().getSenses(sense.getSynset().getPartOfSpeech()).size();
            List<Sense> senses = senseByWordSenseCount.get(senseCount);
            if (senses == null) {
                senses = new ArrayList<Sense>();
                senseByWordSenseCount.put(senseCount, senses);
            }
            senses.add(s);
        }
    }
    for (int senseCount = 1; senseByWordSenseCount.size() > 0;
senseCount++) {
        if (senseByWordSenseCount.containsKey(senseCount)) {
            for (Sense s : senseByWordSenseCount.remove(senseCount)) {
                suggestions.add(s);
            }
        }
    }
}

```

```

        if (suggestions.size() >= maxSuggestions) {
            break;
        }
    }
    if (suggestions.size() >= maxSuggestions) {
        break;
    }
    return suggestions;
}

/**
 * Calculate the information content of a synset using the measure of
Seco,
 * Veale and Hayes, which uses the number of sub-terms (hyponyms) to
 * determine how specific a term is. All synsets with no hyponyms are
 * considered most specific and given a IC rank of 1.0. An artificial
root
 * node for all synsets of the same part of speech is assumed to
better
 * scale the information content of the lowest common subsumers,
especially
 * in the verb hierarchies which tend to be shallow and disjoint.
*
 * @param synset
 * @param linkType
 * @return
 */
public double infoContent(Synset synset, Linkdef linkType) {
    synset = get(synset);
    Integer conceptCount = getConceptCount(synset.getPos());
    double infoContent = (1.0d -
    ((Math.log(synset.getSubsumerCount(linkType) + 1)) / (Math
        .log(conceptCount))));
    if (log.isDebugEnabled()) {
        log.debug("synset: " + synset + " conceptCount: " + conceptCount +
" ic: "
        + infoContent);
    }
    return infoContent;
}

private boolean containsDigits(String word) {
    for (char c : word.toCharArray()) {
        if (Character.isDigit(c)) {
            return true;
        }
    }
}

```

```

        }
    }
    return false;
}

private boolean isNumber(String word) {
    try {
        Integer.parseInt(word);
        return true;
    } catch (NumberFormatException e) {
    }
    try {
        Double.parseDouble(word);
        return true;
    } catch (NumberFormatException e) {
    }
    return false;
}

public void addToDictionary(String word) {
    getSpellChecker().addToDictionary(word);
}

@Override
public Dictionary getDictionary() {
    return getDictionary(null, null);
}

@Override
public Dictionary getDictionary(String firstWordStartsWith, String
lastWordStartsWith) {
    try {
        Dictionary dictionary = new Dictionary();
        if (((firstWordStartsWith == null) ||
"".equals(firstWordStartsWith.trim())))
            && ((lastWordStartsWith == null) ||
"".equals(lastWordStartsWith.trim())))
        {
            dictionary.setCategories(findCategories());
            dictionary.setWords(findWords());
            dictionary.setSynsets(findSynsets());
            dictionary.setSenses(findSenses());
        } else {
            Query query = getEntityManager()
                .createQuery(
                    "select object(word) from Word as word where word.lemma >=
:start and word.lemma < :end");
            query.setParameter("start", firstWordStartsWith);

```

```

            query.setParameter("end", lastWordStartsWith);
            for (Word word : (List<Word>) query.getResultList()) {
                dictionary.getWords().add(word);
                for (Sense sense : word.getSenses()) {
                    Synset synset = sense.getSynset();
                    dictionary.getSynsets().add(synset);
                    dictionary.getCategories().add(synset.getCategory());
                }
            }
            return dictionary;
        } catch (Exception e) {
            throw new RuntimeException("failed to get dictionary between '" +
firstWordStartsWith
                + "'" + " and '" + lastWordStartsWith + "'", e);
        }
    }

    @Override
    public Category findCategory(PartOfSpeech partOfSpeech, String name)
{
    String lookupName = partOfSpeech.name().toLowerCase() + "." +
name.toLowerCase();
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(category) from Category as category where
category.name like :lookupName");
        query.setParameter("lookupName", lookupName);
        return (Category) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchEntityException.byQuery(Category.class, new String[]
{ "partOfSpeech",
    "name" }, new Object[] { partOfSpeech, name });
    } catch (Exception e) {
        throw new RuntimeException("failed to find category for name '" +
lookupName + "'", e);
    }
}

    @Override
    public Word findWord(Long id) {
        return getEntityManager().find(Word.class, id);
    }

    /**

```

```

* @param word -
*          the base form of the word
* @return
* @throws NoSuchWordException
*/
public Word findWord(String word) throws NoSuchWordException {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createQuery(
    "select object(word) from Word as word where word.lemma like
:word");
query.setParameter("word", word);
return (Word) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchWordException.forLemma(word);
} catch (Exception e) {
throw new RuntimeException("failed to find word for text '" + word
+ "'", e);
}
}

/**
* @param lemma -
*          the base form of the word
* @param pos -
*          part of speech
* @return
* @throws NoSuchWordException
*/
public Word findWord(String lemma, PartOfSpeech pos) throws
NoSuchWordException {
try {
Word word = findWord(lemma);
for (Sense sense : word.getSenses(pos)) {
if (sense.getSynset().isPartOfSpeech(pos)) {
return word;
}
}
throw NoSuchWordException.forLemmaAndPOS(lemma, pos.name());
} catch (NoSuchWordException e) {
throw e;
} catch (NoResultException e) {
throw NoSuchWordException.forLemma(lemma);
} catch (Exception e) {
throw new RuntimeException("failed to find word '" + lemma + "' pos
" + pos, e);
}
}

/**
* @param lemma -
*          the base form of the word
* @return
* @throws NoSuchWordException
*/
public Word findWordExact(String lemma) throws NoSuchWordException {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createQuery(
    "select object(word) from Word as word where word.lemma =
:lemma");
query.setParameter("lemma", lemma);
return (Word) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchWordException.forLemma(lemma);
} catch (Exception e) {
throw new RuntimeException("failed to find word for text '" + lemma
+ "'", e);
}
}

/**
* @param lemma -
*          the base form of the word
* @param pos -
*          part of speech
* @return
* @throws NoSuchWordException
*/
public Word findWordExact(String lemma, PartOfSpeech pos) throws
NoSuchWordException {
try {
Word word = findWordExact(lemma);
for (Sense sense : word.getSenses(pos)) {
if (sense.getSynset().isPartOfSpeech(pos)) {
return word;
}
}
throw NoSuchWordException.forLemmaAndPOS(lemma, pos.name());
} catch (NoSuchWordException e) {
throw e;
} catch (NoResultException e) {
throw NoSuchWordException.forLemma(lemma);
} catch (Exception e) {

```

```

        throw new RuntimeException("failed to find word '" + lemma + "' pos
" + pos, e);
    }

}

@Override
public String generatePhoneticCode(String word) {
    return jazzyTransformator.transform(word);
}

public List<Word> findWordsByPhoneticCode(String phoneticCode) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(word) from Word as word where word.phoneticCode
= :phoneticCode");
        query.setParameter("phoneticCode", phoneticCode);
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to find words by phonetic code
'" + phoneticCode
        + "'", e);
    }
}

/***
 * @return
 * @throws RuntimeException
 */
public Collection<Word> findWords() throws RuntimeException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery("select object(word)
from Word as word");
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to find all words.", e);
    }
}

protected Collection<Word> getWords(int minLength, int maxLength)
throws RuntimeException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(word) from Word as word "
+ "where length(word.lemma) >= :minLength "
+ "and length(word.lemma) <= :maxLength");
        query.setParameter("minLength", minLength);
        query.setParameter("maxLength", maxLength);
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to find words of length " +
minLength + " to "
        + maxLength, e);
    }
}

/**
 * @return
 * @throws RuntimeException
 */
public Collection<Category> findCategories() throws RuntimeException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(category) from Category as category");
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to find all categories.", e);
    }
}

public Collection<Sense> findSenses() throws RuntimeException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery("select object(sense) from Sense as sense");
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to get all senses.", e);
    }
}

@Override
public Sense findSense(String lemma, PartOfSpeech partOfSpeech, int
rank) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(sense) from Sense as sense "
            + "where sense.word.lemma = :lemma and "

```

```

        + "sense.synset.pos = :pos and sense.rank = :rank");
query.setParameter("lemma", lemma);
query.setParameter("pos", PartOfSpeech.toWordNetPOS(partOfSpeech));
query.setParameter("rank", rank);
return (Sense) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Sense.class, new String[]
{ "lemma",
    "partOfSpeech", "rank" }, new Object[] { lemma, partOfSpeech,
rank });
} catch (Exception e) {
    throw new RuntimeException("failed to find sense for lemma '" +
lemma + "' pos '"
        + partOfSpeech + ' and rank ' + rank, e);
}
}

@Override
public Sense findSensesByWordAndSynset(Word word, Synset synset) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(sense) from Sense as sense "
        + "where sense.word = :word and sense.synset = :synset");
    query.setParameter("word", word);
    query.setParameter("synset", synset);
    return (Sense) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Sense.class, new String[]
{ "word", "synset" },
    new Object[] { word, synset });
} catch (Exception e) {
    throw new RuntimeException("failed to find sense for word '" + word
+ "' and synset '"
        + synset + "'", e);
}
}

@Override
public Sense findSensesByLemmaAndSynsetId(String lemma, Long
synsetId) {
    return findSensesByWordAndSynset(findWordExact(lemma),
findSynset(synsetId));
}

@Override
public Sense findSensesByWordnetSenseKey(String senseKey) {

```

```

try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(sense) from Sense as sense where sense.senseKey
= :senseKey");
    query.setParameter("senseKey", senseKey);
    return (Sense) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Sense.class, "senseKey",
senseKey);
} catch (Exception e) {
    throw new RuntimeException("failed to find sense for sense key '" +
senseKey + "'", e);
}
}

/**
 * @return
 * @throws RuntimeException
 */
public Collection<Synset> findSynsets() throws RuntimeException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(synset) from Synset as synset");
    return query.getResultList();
} catch (Exception e) {
    throw new RuntimeException("failed to get all synsets.", e);
}
}

@Override
public Lexlinkref findLexlinkref(LexlinkrefId id) {
    return
findLexlinkref(findSensesByWordAndSynset(findWord(id.getWordId())),
findSynset(id
    .getSynsetId())),
findSensesByWordAndSynset(findWord(id.getWord2Id())),
    findSynset(id.getSynset2Id()), findLinkDef(id.getLinkId()));
}

@Override
public Lexlinkref findLexlinkref(Sense fromSense, Sense toSense,
Linkdef linkType) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(

```

```

"select object(lexlink) from Lexlinkref as lexlink "
+ "where lexlink.fromSynset = :fromSynset "
+ "and lexlink.fromWord = :fromWord "
+ "and lexlink.toSynset = :toSynset " + "and lexlink.toWord
= :toWord "
+ "and lexlink.linkType = :linkType ");
query.setParameter("fromSynset", fromSense.getSynset());
query.setParameter("fromWord", fromSense.getWord());
query.setParameter("toSynset", toSense.getSynset());
query.setParameter("toWord", toSense.getWord());
query.setParameter("linkType", linkType);
return (Lexlinkref) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchEntityException.byQuery(Semmlinkref.class, new String[]
{ "fromSense",
"toSense", "linkDef" }, new Object[] { fromSense, toSense,
linkType });
} catch (Exception e) {
log.debug(e, e);
throw new RuntimeException("failed to find " + linkType.getName()
+ " lexo-semantic link from " + fromSense + " to " + toSense, e);
}
}

@Override
public Collection<Lexlinkref> findLexlinkref(Sense fromSense, Sense
toSense) {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createQuery(
"select object(lexlink) from Lexlinkref as lexlink "
+ "where lexlink.fromSynset = :fromSynset "
+ "and lexlink.fromWord = :fromWord "
+ "and lexlink.toSynset = :toSynset " + "and lexlink.toWord
= :toWord "
+ "and lexlink.linkType = :linkType ");
query.setParameter("fromSynset", fromSense.getSynset());
query.setParameter("fromWord", fromSense.getWord());
query.setParameter("toSynset", toSense.getSynset());
query.setParameter("toWord", toSense.getWord());
return query.getResultList();
} catch (Exception e) {
log.debug(e, e);
throw new RuntimeException("failed to find lexo-semantic link from
" + fromSense
+ " to " + toSense, e);
}
}

```

```

        public Collection<Linkdef> findLinkdefs() {
            try {
                // TODO: use named query so it can be configured externally
                Query query = getEntityManager().createQuery(
                    "select object(linkdef) from Linkdef as linkdef");
                return query.getResultList();
            } catch (Exception e) {
                throw new RuntimeException("failed to get all semantic link
definitions.", e);
            }
        }

        @Override
        public boolean buildSynsetDefinitionWords() {
            try {
                // TODO: use named query so it can be configured externally
                Query query = getEntityManager()
                    .createQuery(
                        "select count(synset) from Synset as synset where synset.wsd is
not null and synset.wsd <> '' and not exists elements(synset.words)");
                long count = ((Long) query.getSingleResult()).longValue();
                // TODO: there is one sense that doesn't have a parsed definition,
                // may be a bug in the importer.
                return count > 1L;
            } catch (Exception e) {
                throw new RuntimeException(
                    "failed determining if synset definition words need to be
expanded.", e);
            }
        }

        @Override
        public boolean buildSynsetLinkPathsAndCounts() {
            try {
                Query query = getEntityManager().createNativeQuery(
                    "select count(*) from synset_subsumer_counts");
                long count = ((BigInteger) query.getSingleResult()).longValue();
                return count == 0L;
            } catch (Exception e) {
                throw new RuntimeException(
                    "failed determining if synset semantic link paths and counts need
to be expanded.", e);
            }
        }
    }
}

```

```

public boolean buildSenseKeys() {
    try {
        Query query = getEntityManager().createNativeQuery(
            "select count(*) from sense where sense_key is null");
        long count = ((BigInteger) query.getSingleResult()).longValue();
        return count > 0L;
    } catch (Exception e) {
        throw new RuntimeException("failed determining if senses need sense
key assignment.", e);
    }
}

@Override
public List<Object[]> findSemanticLinks() {
    try {
        Query query = getEntityManager()
            .createNativeQuery(
                "select synset1id, synset2id, linkid, distance from semlinkref
order by synset1id, linkid");
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to get semantic links.", e);
    }
}

@Override
public Linkdef findLinkDef(Long id) {
    return getEntityManager().find(Linkdef.class, id);
}

@Override
public Synset findSynset(Long id) {
    return getEntityManager().find(Synset.class, id);
}

@Override
public List<Synset>
findSynsetsWithColocatedDefinitionSenseAndWord(Sense sense, Word word)
{
    log.debug("sense: " + sense + " word: " + word);
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createNativeQuery(
                "select sy2.* "
                + "from synset syl "

```

```

                + "left join synset_definition_word sdw1 on (sdw1.synset_id =
syl.synsetid) "
                + "left join synset_definition_word sdw2 on (sdw2.synset_id =
syl.synsetid) "
                    + "left join synset sy2 on (sy2.synsetid = sdw2.sense_id) "
                    + "where sdw1.sense_id = :sense "
                    + "and   sdw2.word_id   = :word " +
                    "union " +
                    "select * from synset where synsetid = :sense ",
Synset.class);
        query.setParameter("sense", sense.getId());
        query.setParameter("word", word.getId());
        return query.getResultList();
    } catch (NoResultException e) {
        return new ArrayList<Synset>();
    } catch (Exception e) {
        log.debug(e, e);
        throw new RuntimeException("failed to find colocation in synset
definitions with "
            + sense + " and " + word, e);
    }
}

@Override
public List<Synset> findSynsetsWithColocatedDefinitionWords(Word
word1, Word word2) {
    log.debug("word1: " + word1 + " word2: " + word2);
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createNativeQuery(
                "select sy2.* "
                + "from synset syl "
                + "left join synset_definition_word sdw1 on (sdw1.synset_id =
syl.synsetid) "
                + "left join synset_definition_word sdw2 on (sdw2.synset_id =
syl.synsetid) "
                    + "left join synset sy2 on (sy2.synsetid = sdw2.sense_id) "
                    + "where sdw1.word_id = :word1 "
                    + "and   sdw2.word_id   = :word2 "
                    +
                    " union "
                    +

```

```

"select sy2.* "
+ "from synset syl "
+ "left join synset_definition_word sdw1 on (sdw1.synset_id =
syl.synsetid) "
+ "left join synset_definition_word sdw2 on (sdw2.synset_id =
syl.synsetid) "
+ "left join synset sy2 on (sy2.synsetid = sdw1.sense_id) "
+ "where sdw1.word_id = :word1 "
+ "and sdw2.word_id = :word2", Synset.class);
query.setParameter("word1", word1.getId());
query.setParameter("word2", word2.getId());
return query.getResultList();
} catch (NoResultException e) {
return new ArrayList<Synset>();
} catch (Exception e) {
log.debug(e, e);
throw new RuntimeException("failed to find colocation in synset
defintions with "
+ word1 + " and " + word2, e);
}
}

@Override
public Semlinkref findSemlinkref(SemlinkrefId id) {
return findSemlinkref(findSynset(id.getSynsetId()),
findSynset(id.getSynset2id()),
findLinkDef(id.getLinkId()), id.getDistance());
}

@Override
public Semlinkref findSemlinkref(Synset fromSynset, Synset toSynset,
Linkdef linkType,
Integer distance) {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createQuery(
"select object(semlink) from Semlinkref as semlink "
+ "where semlink.fromSynset = :fromSynset "
+ "and semlink.toSynset = :toSynset "
+ "and semlink.linkType = :linkType "
+ "and semlink.id.distance = :distance");
query.setParameter("fromSynset", fromSynset);
query.setParameter("toSynset", toSynset);
query.setParameter("linkType", linkType);
query.setParameter("distance", distance);
return (Semlinkref) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchEntityException.byQuery(Semlinkref.class, new String[]
{ "fromSynset",
"toSynset", "linkDef", "distance" }, new Object[] { fromSynset,
toSynset,
linkType, distance });
} catch (Exception e) {
log.debug(e, e);
throw new RuntimeException("failed to find " + linkType.getName()
+ " semantic link from " + fromSynset + " to " + toSynset + " at
distance "
+ distance, e);
}
}

private final Map<SynsetPair, Set<Synset>> lowestCommonHyperonymsCache
= new HashMap<SynsetPair, Set<Synset>>();

@Override
public Set<Synset> getLowestCommonHyperonyms(Synset synset1, Synset
synset2) {
try {
Set<Synset> lcsSet = new HashSet<Synset>();
SynsetPair synsetPair = new SynsetPair(synset1, synset2);
if (lowestCommonHyperonymsCache.containsKey(synsetPair)) {
for (Synset synset : lowestCommonHyperonymsCache.get(synsetPair)) {
// get a fresh copy attached to the persistence context
lcsSet.add(get(synset));
}
} else {
if (synset1.getPartOfSpeech().equals(synset2.getPartOfSpeech())) {
Query query = getEntityManager().createNativeQuery(
"select syn.* from synset syn " + "left join semlinkref slr1 on
( "
+ "slr1.synset2id = syn.synsetid ) "
+ "left join semlinkref slr2 on ( "
+ "slr1.synset2id = slr2.synset2id ) "
+ "where slr1.synset1id = :synset1id "
+ "and slr2.synset1id = :synset2id and slr1.linkid = 1 "
+ "and slr2.linkid = 1 "
+ "having min(slr1.distance + slr2.distance)", Synset.class);
query.setParameter("synset1id", synset1.getId());
query.setParameter("synset2id", synset2.getId());
lcsSet.addAll(query.getResultList());
}
lowestCommonHyperonymsCache.put(synsetPair, lcsSet);
}
if (log.isDebugEnabled()) {
}
}
}

```

```

        log.debug("synset1: " + synset1 + " synset2: " + synset2 + " -> "
+ lcsSet);
    }
    return Collections.unmodifiableSet(lcsSet);
} catch (Exception e) {
    log.debug(e, e);
    throw new RuntimeException("failed to get lowest common hypernyms
for " + synset1
        + " and " + synset2, e);
}

private final Map<Synset, Synset> rootHypernymCache = new
HashMap<Synset, Synset>();

@Override
public Synset getRootHypernym(Synset synset) {
    try {
        Synset rootHypernym;
        if (rootHypernymCache.containsKey(synset)) {
            rootHypernym = rootHypernymCache.get(synset);
        } else {
            Query query = getEntityManager().createNativeQuery(
                "select syn.* " + "from synset syn left join semlinkref slr1 on
( "
                + " slr1.synset2id = syn.synsetid) "
                + "where slr1.synsetlid = :synsetid and slr1.linkid = 1 "
                + "order by distance desc ", Synset.class);
            query.setParameter("synsetid", synset.getId());
            query.setMaxResults(1);
            rootHypernym = (Synset) query.getSingleResult();
            rootHypernymCache.put(synset, rootHypernym);
        }
        if (log.isDebugEnabled()) {
            log.debug("synset: " + synset + " rootHypernym: " + rootHypernym);
        }
        return rootHypernym;
    } catch (NoResultException e) {
        // return the original synset
        return synset;
    } catch (Exception e) {
        log.debug(e, e);
        throw new RuntimeException("failed to get root hypernym for " +
synset, e);
    }
}

```

```

private final Map<String, Integer> posConceptCountCache = new
HashMap<String, Integer>();

@Override
public Integer getConceptCount(String pos) {
    try {
        if (!posConceptCountCache.containsKey(pos)) {
            Query query = getEntityManager().createNativeQuery(
                "select count(*) from synset where pos = :pos");
            query.setParameter("pos", pos);
            posConceptCountCache.put(pos, ((BigInteger)
query.getSingleResult()).intValue());
        }
        return posConceptCountCache.get(pos);
    } catch (Exception e) {
        throw new RuntimeException("failed to get concept count for " +
pos, e);
    }
}

private static class SynsetPair {
    private final Synset synset1;
    private final Synset synset2;

    private SynsetPair(Synset synset1, Synset synset2) {
        this.synset1 = synset1;
        this.synset2 = synset2;
    }

    public Synset getSynset1() {
        return synset1;
    }

    public Synset getSynset2() {
        return synset2;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((synset1 == null) ? 0 :
synset1.hashCode());
        result = prime * result + ((synset2 == null) ? 0 :
synset2.hashCode());
        return result;
    }
}

```

```

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final SynsetPair other = (SynsetPair) obj;
    if (synset1 == null) {
        if (other.synset1 != null) {
            return false;
        }
    } else if (!synset1.equals(other.synset1)) {
        return false;
    }
    if (synset2 == null) {
        if (other.synset2 != null) {
            return false;
        }
    } else if (!synset2.equals(other.synset2)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "(" + getSynset1() + ", " + getSynset2() + ")";
}

@Override
public Collection<SemcorSentence> findSemcorSentences() {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(sentence) from SemcorSentence as sentence");
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to get all Semcor Sentences.", e);
    }
}

}

@Override
public Collection<SemcorSentence> findSemcorSentences(String sectionName, String fileName) {
    throw new RuntimeException("not implemented.");
}

@Override
public Collection<SemcorSentence> findSemcorSentences(Sense sense) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(sentence) from SemcorSentence as sentence
where :word in elements(sentence.words)");
        query.setParameter("word", sense.getWord());
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to get all Semcor Sentences for
sense " + sense, e);
    }
}

@Override
public Collection<SemcorSentence> findSemcorSentences(Word word) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(sentence) from SemcorSentence as sentence
where :word in elements(sentence.words)");
        query.setParameter("word", word);
        return query.getResultList();
    } catch (Exception e) {
        throw new RuntimeException("failed to get all Semcor Sentences for
word " + word, e);
    }
}

@Override
public SemcorFile findSemcorFile(String corpusSectionFolder, String corpusFileName) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(

```

```

    "select object(file) from SemcorFile as file where
file.corpusSectionFolder like :corpusSectionFolder and
file.corpusFileName like :corpusFileName");
query.setParameter("corpusSectionFolder", corpusSectionFolder);
query.setParameter("corpusFileName", corpusFileName);
return (SemcorFile) query.getSingleResult();
} catch (NoResultException e) {
throw NoSuchEntityException.byQuery(SemcorFile.class, new String[]
{
    "corpusSectionFolder", "corpusFileName" }, new Object[]
{ corpusSectionFolder,
  corpusFileName });
} catch (Exception e) {
throw new RuntimeException("failed to find semcor file by section "
+ corpusSectionFolder + " and file " + corpusFileName, e);
}
}

@Override
public Collection<SemcorFile> findSemcorFiles() {
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createQuery(
    "select object(file) from SemcorFile as file");
return query.getResultList();
} catch (Exception e) {
throw new RuntimeException("failed to get all Semcor Sentences.", e);
}
}

@Override
public List<SemcorSentenceWord>
findSemcorSentencesWithColocatedDefinitionSenseAndWord(
    Sense sense, Word word) {
log.debug("sense: " + sense + " word: " + word);
try {
// TODO: use named query so it can be configured externally
Query query = getEntityManager().createNativeQuery(
    "select ssw1.* from semcor_sentence_word ssw1 "
    + "left join semcor_sentence ss1 on ( ssw1.sentence_id = ss1.id "
    + ") left join semcor_sentence_word ssw2 on ( "
    + "ssw2.sentence_id = ss1.id ) where ssw1.sense_id = :sense "
    + "and ssw2.word_id = :word " +
"union " +

```

```

    "select ssw2.* from semcor_sentence_word ssw1 "
    + "left join semcor_sentence ss1 on ( ssw1.sentence_id = ss1.id "
    + ") left join semcor_sentence_word ssw2 on ( "
    + "ssw2.sentence_id = ss1.id ) where ssw1.sense_id = :sense "
    + "and ssw2.word_id = :word", SemcorSentenceWord.class);
query.setParameter("sense", sense.getId());
query.setParameter("word", word.getId());
return query.getResultList();
} catch (NoResultException e) {

```

```

        return new ArrayList<SemcorSentenceWord>();
    } catch (Exception e) {
        log.debug(e, e);
        throw new RuntimeException("failed to find colocation in semcor
sentences with "
                + word1 + " and " + word2, e);
    }
}

@Override
public VerbNetSelectionRestrictionType
findVerbNetSelectionRestrictionType(String name) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(selResType) from VerbNetSelectionRestrictionType
as selResType "
            + " where selResType.name like :name");
        query.setParameter("name", name);
        return (VerbNetSelectionRestrictionType) query.getSingleResult();
    } catch (NoResultException e) {
        throw new NoSuchEntityException(
            .byQuery(VerbNetSelectionRestrictionType.class, "name", name));
    } catch (Exception e) {
        throw new RuntimeException(
            "failed to get VerbNetSelectionRestriction by name: " + name, e);
    }
}

@Override
public boolean isHyponym(Sense hypernym, Sense sense) {
    return isHyponym(hypernym.getSynset(), sense.getSynset());
}

@Override
public boolean isHyponym(Synset hypernym, Synset synset) {
    log.debug("checking if " + synset + " is a hyponym of " + hypernym);
    try {
        Query query = getEntityManager().createNativeQuery(
            "select count(*) from semlinkref slr1 left join synset hypernym
on ( "
            + "slr1.synset2id = hypernym.synsetid) "
            + "left join synset hyponym on ( "
            + "slr1.synset1id = hyponym.synsetid) "
            + "where hypernym.synsetid = :hypernym "
            + "and hyponym.synsetid = :hyponym and slr1.linkid = 1");
        query.setParameter("hypernym", hypernym.getId());
    }
}

```

```

        query.setParameter("hyponym", synset.getId());
        long count = ((BigInteger) query.getSingleResult()).longValue();
        return count > 0L;
    } catch (Exception e) {
        log.error(e, e);
        throw new RuntimeException("failed to determine if " + synset + "
is a hyponym of "
                + hypernym, e);
    }
}

@Override
public Collection<Synset> findHyponyms(Synset hypernym, int
maxDistance) {
    log.debug("get the hyponyms of " + hypernym + " with a max distance
of " + maxDistance);
    try {
        Query query = getEntityManager().createNativeQuery(
            "select syl.* from semlinkref slr "
            + "left join synset syl on (slr.synset1id = syl.synsetid) "
            + "left join sense snl on (syl.synsetid = snl.synsetid) "
            + "left join word wd1 on (snl.wordid = wd1.wordid) "
            + "left join synset sy2 on (slr.synset2id = sy2.synsetid) "
            + "left join sense sn2 on (sy2.synsetid = sn2.synsetid) "
            + "left join word wd2 on (sn2.wordid = wd2.wordid) "
            + "where slr.linkid = 1 " + "and sy2.synsetid = :hypernym "
            + "and slr.distance <= :maxDistance", Synset.class);
        query.setParameter("hypernym", hypernym.getId());
        query.setParameter("maxDistance", maxDistance);
        return query.getResultList();
    } catch (Exception e) {
        log.error(e, e);
        throw new RuntimeException("failed to get the hyponyms of " +
hypernym
            + " with a max distance of " + maxDistance, e);
    }
}

```

jparepository.java

```

/*
 * $Id: JpaProjectRepository.java,v 1.32 2009/02/16 10:10:08 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl.repository.jpa;

import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.NoResultException;
import javax.persistence.OptimisticLockException;
import javax.persistence.Query;

import org.hibernate.PropertyValueException;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.ConstraintViolationException;
import org.hibernate.exception.LockAcquisitionException;
import org.hibernate.validator.InvalidStateException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.dao.CannotAcquireLockException;
import
org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureEx
ception;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository;
import
edu.harvard.fas.rregan.repository.jpa.ConstraintViolationExceptionAdap
ter;
import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;
import
edu.harvard.fas.rregan.repository.jpa.GenericPropertyValueExceptionAda
pter;
import
edu.harvard.fas.rregan.repository.jpa.InvalidStateExceptionAdapter;
import
edu.harvard.fas.rregan.repository.jpa.OptimisticLockExceptionAdapter;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchPositionException;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

```

```

import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import
edu.harvard.fas.rregan.requel.project.command.EditProjectCommand;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchActorException;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchGlossaryTermExce
ption;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchProjectException
;
import edu.harvard.fas.rregan.requel.project.impl.AddActorPosition;
import
edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import edu.harvard.fas.rregan.requel.project.impl.GlossaryTermImpl;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * EJB3/JPA based repository
 *
 * @author ron
 */
@Repository("projectRepository")
@Scope("singleton")
@Transactional(propagation = Propagation.REQUIRED, noRollbackFor =
{ NoSuchPositionException.class,
  NoSuchProjectException.class, EntityException.class })
public class JpaProjectRepository extends AbstractJpaRepository
implements ProjectRepository {

    /**
     * @param exceptionMapper
     */
    @Autowired
    public JpaProjectRepository(ExceptionMapper exceptionMapper) {
        super(exceptionMapper);
        addExceptionAdapter(PropertyValueException.class,
            new GenericPropertyValueExceptionAdapter(), Project.class,

```

```

    ProjectOrDomainEntity.class, AddGlossaryTermPosition.class,
AddActorPosition.class);

    addExceptionAdapter(InvalidStateException.class, new
InvalidStateExceptionAdapter(),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(ConstraintViolationException.class,
    new
ConstraintViolationExceptionAdapter(EditProjectCommand.FIELD_NAME),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(OptimisticLockException.class, new
OptimisticLockExceptionAdapter(),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(StaleObjectStateException.class, new
OptimisticLockExceptionAdapter(),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(LockAcquisitionException.class, new
OptimisticLockExceptionAdapter(),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(CannotAcquireLockException.class, new
OptimisticLockExceptionAdapter(),
    Project.class, ProjectOrDomainEntity.class,
AddGlossaryTermPosition.class,
    AddActorPosition.class);

    addExceptionAdapter(HibernateOptimisticLockingFailureException.class
,
    new OptimisticLockExceptionAdapter(), Project.class,
ProjectOrDomainEntity.class,
    AddGlossaryTermPosition.class, AddActorPosition.class);
}

```

```

    public Project findProjectByName(String name) throws
NoSuchProjectException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(project) from ProjectImpl as project where
project.name like :name");
        query.setParameter("name", name.trim());
        return (Project) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchProjectException.forName(name);
    } catch (Exception e) {
        throw convertException(e, Project.class, null,
EntityExceptionActionType.Reading);
    }
}

@Override
public Goal findGoalByProjectOrDomainAndName(ProjectOrDomain pod,
String name)
    throws EntityException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(goal) from GoalImpl as goal where
goal.projectOrDomain = :projectOrDomain and goal.name like :name");
        query.setParameter("projectOrDomain", pod);
        query.setParameter("name", name.trim());
        return (Goal) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchEntityException.byQuery(Goal.class, new String[]
{ "project", "name" },
            new Object[] { pod, name });
    } catch (Exception e) {
        throw convertException(e, Goal.class, null,
EntityExceptionActionType.Reading);
    }
}

@Override
public Scenario findScenarioByProjectOrDomainAndName(ProjectOrDomain
pod, String name)
    throws NoSuchEntityException {
    try {
        // TODO: use named query so it can be configured externally

```

```

Query query = getEntityManager()
    .createQuery(
        "select object(scenario) from ScenarioImpl as scenario where
scenario.projectOrDomain = :projectOrDomain and scenario.name like
:name");
    query.setParameter("projectOrDomain", pod);
    query.setParameter("name", name.trim());
    return (Scenario) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Scenario.class, new String[]
{ "project", "name" },
    new Object[] { pod, name });
} catch (Exception e) {
    throw convertException(e, Scenario.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public Set<Scenario> findScenariosUsedByUseCase(UseCase usecase) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager()
        .createQuery(
            "select object(scenario) from ScenarioImpl as scenario inner
join scenario.usingUseCases as useCases where :usecase in useCases");
    query.setParameter("usecase", usecase);
    return new HashSet<Scenario>(query.getResultList());
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Scenario.class, "usecase",
usecase);
} catch (Exception e) {
    throw convertException(e, Scenario.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public Stakeholder
findStakeholderByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain,
String name) throws NoSuchEntityException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager()
        .createQuery(

```

```

        "select object(stakeholder) from StakeholderImpl as stakeholder
where stakeholder.projectOrDomain = :projectOrDomain and
stakeholder.name like :name and stakeholder.user is null");
    query.setParameter("projectOrDomain", projectOrDomain);
    query.setParameter("name", name.trim());
    return (Stakeholder) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Scenario.class, new String[]
{ "project", "name" },
    new Object[] { projectOrDomain, name });
} catch (Exception e) {
    throw convertException(e, Scenario.class, null,
EntityExceptionActionType.Reading);
}

@Override
public Stakeholder
findStakeholderByProjectOrDomainAndUser(ProjectOrDomain
projectOrDomain,
User user) throws NoSuchEntityException {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager()
        .createQuery(
            "select object(stakeholder) from StakeholderImpl as stakeholder
where stakeholder.projectOrDomain = :projectOrDomain and
stakeholder.user = :user");
    query.setParameter("projectOrDomain", projectOrDomain);
    query.setParameter("user", user);
    return (Stakeholder) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(Scenario.class, new String[]
{ "project", "user" },
    new Object[] { projectOrDomain, user });
} catch (Exception e) {
    throw convertException(e, Scenario.class, null,
EntityExceptionActionType.Reading);
}
}

public StakeholderPermission findStakeholderPermission(Class<?>
entityType,
StakeholderPermissionType permissionType) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager()

```

```

.createQuery(
    "select object(permission) from StakeholderPermissionImpl as
permission where permission.entityType = :entityType and
permission.permissionType = :permissionType");
query.setParameter("entityType", entityType);
query.setParameter("permissionType", permissionType);
return (StakeholderPermission) query.getSingleResult();
} catch (NoResultException e) {
    throw NoSuchEntityException.byQuery(StakeholderPermission.class,
new String[] {
    "entityType", "permissionType" }, new Object[] { entityType,
permissionType });
} catch (Exception e) {
    throw convertException(e, StakeholderPermission.class, null,
EntityExceptionActionType.Reading);
}
}

public Set<StakeholderPermission>
findAvailableStakeholderPermissions() {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(permission) from StakeholderPermissionImpl as
permission");
    return new TreeSet<StakeholderPermission>(query.getResultList());
} catch (Exception e) {
    throw convertException(e, StakeholderPermission.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public GlossaryTerm
findGlossaryTermForProjectOrDomain(ProjectOrDomain projectOrDomain,
String name) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(term) from GlossaryTermImpl as term "
        + "where term.name = :name and "
        + "term.projectOrDomain = :projectOrDomain");
    query.setParameter("name", name.trim());
    query.setParameter("projectOrDomain", projectOrDomain);
    return (GlossaryTerm) query.getSingleResult();
} catch (NoResultException e) {
    throw
        NoSuchGlossaryTermException.forProjectOrDomainWithName(projectOrDomain
        , name);
} catch (Exception e) {
    throw convertException(e, GlossaryTerm.class, null,
EntityExceptionActionType.Reading);
}
}

@Override
public Set<GlossaryTerm> findGlossaryTermsForProjectOrDomainEntity(
    ProjectOrDomainEntity projectOrDomainEntity) {
try {
    String entityType =
projectOrDomainEntity.getProjectOrDomainEntityInterface().getName();
    Object entityId = getId(projectOrDomainEntity);

    // NOTE: there isn't a way to create an HQL query that crosses an
    // ANY relationship
    // boundary, so a native query is the only way
    Query query = getEntityManager()
        .createNativeQuery(
            "select term.* from terms term left outer join terms_referers
referers on term.id=referers.term_id where referers.referer_type =
:entityType and referers.referer_id = :entityId",
            GlossaryTermImpl.class);
    query.setParameter("entityType", entityType);
    query.setParameter("entityId", entityId);
    return new HashSet<GlossaryTerm>(query.getResultList());
} catch (Exception e) {
    throw convertException(e, GlossaryTerm.class, null,
EntityExceptionActionType.Reading);
}
}

public AddGlossaryTermPosition
findAddGlossaryTermPosition(ProjectOrDomain projectOrDomain,
String term) {
try {
    // TODO: use named query so it can be configured externally
    Query query = getEntityManager().createQuery(
        "select object(position) from AddGlossaryTermPosition as position
"
        + "left join position.issues as issue "
        + "where issue.groupingObject = :projectOrDomain and "
        + "issue.word like :term");
    query.setParameter("term", term.trim());
}

```

```

query.setParameter("projectOrDomain", projectOrDomain);
return (AddGlossaryTermPosition) query.getSingleResult();
} catch (NoResultException e) {
    throw
}
NoSuchPositionException.forAddingWordToGlossary(projectOrDomain,
term);
} catch (Exception e) {
    throw convertException(e, AddGlossaryTermPosition.class, null,
        EntityExceptionActionType.Reading);
}
}

@Override
public AddActorPosition findAddActorPosition(ProjectOrDomain
projectOrDomain, String actorName) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(position) from AddActorPosition as position "
            + "left join position.issues as issue "
            + "where issue.groupingObject = :projectOrDomain and "
            + "issue.word like :actorName");
        query.setParameter("actorName", actorName.trim());
        query.setParameter("projectOrDomain", projectOrDomain);
        return (AddActorPosition) query.getSingleResult();
    } catch (NoResultException e) {
        throw
}
NoSuchPositionException.forAddingWordToGlossary(projectOrDomain,
actorName);
} catch (Exception e) {
    throw convertException(e, AddActorPosition.class, null,
        EntityExceptionActionType.Reading);
}
}

@Override
public Actor findActorByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
    throws EntityException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(actor) from ActorImpl as actor where
actor.projectOrDomain = :projectOrDomain and actor.name like :name");
        query.setParameter("projectOrDomain", projectOrDomain);
        query.setParameter("name", name.trim());
    }
}

```

```

        return (Actor) query.getSingleResult();
    } catch (NoResultException e) {
        throw
}
NoSuchActorException.forProjectOrDomainWithName(projectOrDomain,
name);
} catch (Exception e) {
    throw convertException(e, Actor.class, null,
        EntityExceptionActionType.Reading);
}
}

@Override
public Story findStoryByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
    throws EntityException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(story) from StoryImpl as story where
story.projectOrDomain = :projectOrDomain and story.name like :name");
        query.setParameter("projectOrDomain", projectOrDomain);
        query.setParameter("name", name.trim());
        return (Story) query.getSingleResult();
    } catch (NoResultException e) {
        throw
}
NoSuchActorException.forProjectOrDomainWithName(projectOrDomain,
name);
} catch (Exception e) {
    throw convertException(e, Story.class, null,
        EntityExceptionActionType.Reading);
}
}

@Override
public UseCase findUseCaseByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
    throws EntityException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(usecase) from UseCaseImpl as usecase where
usecase.projectOrDomain = :projectOrDomain and usecase.name like
:name");
        query.setParameter("projectOrDomain", projectOrDomain);
        query.setParameter("name", name.trim());
    }
}

```

```

        return (UseCase) query.getSingleResult();
    } catch (NoResultException e) {
        throw
    NoSuchActorException.forProjectOrDomainWithName(projectOrDomain,
name);
    } catch (Exception e) {
        throw convertException(e, UseCase.class, null,
EntityExceptionActionType.Reading);
    }
}

@Override
public ReportGenerator findReportGeneratorByProjectOrDomainAndName(
    ProjectOrDomain projectOrDomain, String name) throws
EntityException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(reportGenerator) from ReportGeneratorImpl as
reportGenerator where reportGenerator.projectOrDomain =
:projectOrDomain and reportGenerator.name = :name");
        query.setParameter("projectOrDomain", projectOrDomain);
        query.setParameter("name", name.trim());
        return (ReportGenerator) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchEntityException.createQuery(Scenario.class, new String[]
{ "project", "name" },
        new Object[] { projectOrDomain, name });
    } catch (Exception e) {
        throw convertException(e, Scenario.class, null,
EntityExceptionActionType.Reading);
    }
}
}

```

jpauserrepository.java

```

package edu.harvard.fas.rregan.requel.user.impl.repository.jpa;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.NoResultException;
import javax.persistence.OptimisticLockException;

```

```

import javax.persistence.Query;

import org.hibernate.PropertyValueException;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.ConstraintViolationException;
import org.hibernate.exception.LockAcquisitionException;
import org.hibernate.validator.InvalidStateException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.dao.CannotAcquireLockException;
import
org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureEx
ception;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository;
import
edu.harvard.fas.rregan.repository.jpa.ConstraintViolationExceptionAdap
ter;
import edu.harvard.fas.rregan.repository.jpa.ExceptionMapper;
import
edu.harvard.fas.rregan.repository.jpa.GenericPropertyValueExceptionAda
pter;
import
edu.harvard.fas.rregan.repository.jpa.InvalidStateExceptionAdapter;
import
edu.harvard.fas.rregan.repository.jpa.OptimisticLockExceptionAdapter;
import
edu.harvard.fas.rregan.repository.jpa.UserPropertyValueExceptionAdapte
r;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRole;
import edu.harvard.fas.rregan.requel.user UserRolePermission;
import edu.harvard.fas.rregan.requel.user.UserSet;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchOrganizationExcepti
on;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

```

```

import edu.harvard.fas.rregan.requel.user.impl.UserSetImpl;

/**
 * @author ron
 */
@Repository("userRepository")
@Scope("singleton")
@Transactional(propagation = Propagation.REQUIRED, noRollbackFor =
{ NoSuchUserException.class,
  NoSuchOrganizationException.class, EntityException.class })
public class JpaUserRepository extends AbstractJpaRepository
implements UserRepository {

    /**
     *
     */
    @Autowired
    public JpaUserRepository(ExceptionMapper exceptionMapper) {
        super(exceptionMapper);
        addExceptionAdapter(PropertyValueException.class, new
UserPropertyValueExceptionAdapter(),
            User.class);
        addExceptionAdapter(PropertyValueException.class,
            new GenericPropertyValueExceptionAdapter(), Organization.class,
UserRole.class,
            UserRolePermission.class);

        addExceptionAdapter(InvalidStateException.class, new
InvalidStateExceptionAdapter(),
            User.class, Organization.class, UserRole.class,
UserRolePermission.class);

        addExceptionAdapter(ConstraintViolationException.class,
            new ConstraintViolationExceptionAdapter("username"), User.class);

        addExceptionAdapter(OptimisticLockException.class, new
OptimisticLockExceptionAdapter(),
            User.class, Organization.class, UserRole.class,
UserRolePermission.class);

        addExceptionAdapter(StaleObjectStateException.class, new
OptimisticLockExceptionAdapter(),
            User.class, Organization.class, UserRole.class,
UserRolePermission.class);

        addExceptionAdapter(LockAcquisitionException.class, new
OptimisticLockExceptionAdapter(),
            User.class, Organization.class, UserRole.class,
UserRolePermission.class);
    }

    User.class, Organization.class, UserRole.class,
UserRolePermission.class);

    addExceptionAdapter(CannotAcquireLockException.class, new
OptimisticLockExceptionAdapter(),
        User.class, Organization.class, UserRole.class,
UserRolePermission.class);

    addExceptionAdapter(HibernateOptimisticLockingFailureException.class
,
        new OptimisticLockExceptionAdapter(), User.class,
Organization.class,
        UserRole.class, UserRolePermission.class);

}

public User findUserByUsername(String username) throws
NoSuchUserException {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(user) from UserImpl as user where user.username
like :username");
        query.setParameter("username", username);
        return (User) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchUserException.forUsername(username);
    } catch (Exception e) {
        throw convertException(e, User.class, null,
EntityExceptionActionType.Reading);
    }
}

public UserSet findUsers() {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(user) from UserImpl as user");
        return new UserSetImpl(query.getResultList());
    } catch (NoResultException e) {
        return (UserSet) new HashSet<User>();
    } catch (Exception e) {
        throw convertException(e, UserSet.class, null,
EntityExceptionActionType.Reading);
    }
}
}

```

```

public UserSet findUsersForRole(Class<? extends UserRole> roleType) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(user) from UserImpl as user inner join
user.userRoles as roles, AbstractUserRole role where role.roleType
like :roleType and role in roles");
        query.setParameter("roleType", roleType.getName());
        return new UserSetImpl(query.getResultList());
    } catch (NoResultException e) {
        return (UserSet) new HashSet<User>();
    } catch (Exception e) {
        throw convertException(e, UserSet.class, null,
EntityExceptionActionType.Reading);
    }
}

public Organization findOrganizationByName(String name) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(organization) from OrganizationImpl as
organization where name like :name");
        query.setParameter("name", name);
        return (Organization) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchOrganizationException.forName(name);
    } catch (Exception e) {
        throw convertException(e, Organization.class, null,
EntityExceptionActionType.Reading);
    }
}

public Set<Organization> findOrganizations() {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager().createQuery(
            "select object(organization) from OrganizationImpl as
organization");
        return new HashSet<Organization>(query.getResultList());
    } catch (Exception e) {
        throw convertException(e, Organization.class, null,
EntityExceptionActionType.Reading);
    }
}

```

```

public Set<String> getOrganizationNames() {
    Set<String> orgNames = new HashSet<String>();
    for (Organization org : findOrganizations()) {
        orgNames.add(org.getName());
    }
    return orgNames;
}

public UserRolePermission findUserRolePermission(Class<? extends
UserRole> userRoleType,
String name) {
    try {
        // TODO: use named query so it can be configured externally
        Query query = getEntityManager()
            .createQuery(
                "select object(permission) from UserRolePermission as
permission where userRoleType like :userRoleType and name like
:name");
        query.setParameter("userRoleType", userRoleType.getName());
        query.setParameter("name", name);
        return (UserRolePermission) query.getSingleResult();
    } catch (NoResultException e) {
        throw NoSuchEntityException.byQuery(UserRolePermission.class,
"name", name);
    } catch (Exception e) {
        throw convertException(e, Organization.class, null,
EntityExceptionActionType.Reading);
    }
}

public Set<UserRolePermission> findUserRolePermissions(Class<? extends
UserRole> userRoleType) {
    Set<UserRolePermission> permissions = new
HashSet<UserRolePermission>();
    for (UserRolePermission permission : AbstractUserRole
        .getAvailableUserRolePermissions(userRoleType)) {
        permissions.add(merge(permission));
    }
    return permissions;
}

public Set<Class<? extends UserRole>> findUserRoleTypes() {
    return AbstractUserRole.getAvailableUserRoles();
}

```

labelmanipulator.java

```
/*
 * $Id: LabelManipulator.java,v 1.8 2008/10/11 21:47:44 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class LabelManipulator extends AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        final Label text = (Label) component;
        return type.cast(text.getText());
    }

    public void setValue(Component component, Object value) {
        final Label text = (Label) component;
        text.setText((String) value);
    }

    public void addListenerToDetectChangesToInput(finalEditMode
        editMode, Component component) {
        // label's don't get changed by users
    }

    @Override
    public String getModel(Component component) {
        return getValue(component, String.class);
    }

    @Override
    public void setModel(Component component, Object valueModel) {
        setValue(component, valueModel);
    }
}
```

lemmatizerrule.java

```
/*
 * $Id: LemmatizerRule.java,v 1.1 2008/10/02 09:38:37 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

/**
 * interface for specifying rules for lemmatizing a word.
 *
 * @author ron
 */
public interface LemmatizerRule {

    /**
     * Given a word and part of speech, apply the rule returning the
     * lemma if
     * this rule is applicable and a valid lemma is found, or returning
     * null.
     *
     * @param word -
     *          the word to lemmatize
     * @param partOfSpeech -
     *          the part of speech such as NOUN, VERB, etc.
     * @return the lemma if the rule is appropriate and a valid lemma is
     *         generated (found in the dictionary), null otherwise.
     */
    public String lemmatize(String word, PartOfSpeech partOfSpeech);
}
```

lexicalassistant.java

```
/*
 * $Id: LexicalAssistant.java,v 1.23 2009/03/27 07:16:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
```

```

import java.util.Set;
import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPPProcessorFactory;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Linkdef;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import
edu.harvard.fas.rregan.requel.annotation.NoSuchAnnotationException;
import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.EditAddWordToDictiona
ryPositionCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditChangeSpellingPos
itionCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditLexicalIssueComma
nd;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.EditAddActorToProjectPos
itionCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditAddWordToGlossaryPos
itionCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditGlossaryTermCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;

```

```

import
edu.harvard.fas.rregan.requel.project.command.RemoveUnneedLexicalIssue
sCommand;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchActorException;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchGlossaryTermExce
ption;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * The lexical assistant checks spelling and identifies noun phrases
that may be
 * glossary terms, actors, or entities in the domain model of a
project.<br>
 * When a word in the text is not in the dictionary a spelling issue
is added to
 * the object being analyzed with positions suggesting an alternate
spelling and
 * a position indicating the word should be added to the
dictionary.<br>
 * Phrases identified as possible glossary terms, actors or domain
objects are
 * filtered by the Information Content and categories of the words in
the
 * phrase.
 *
 * @author ron
 */
public class LexicalAssistant extends AbstractAssistant {
    private static final Logger log =
Logger.getLogger(LexicalAssistant.class);

    // spelling issues
    /**
     * The name of the property in the LexicalAssistant.properties file
for the
     * issue text indicating a word is not known in the dictionary.
     */
    public static final String PROP_UNKNOWN_WORD_MSG =
"UnknownWordMessage";
    public static final String PROP_UNKNOWN_WORD_MSG_DEFAULT = "The
word \"\{0\}\" in the \{1\} is not recognized and may be spelled
incorrectly.';

    /**

```

```

 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating the word should be ignored.
 */
public static final String PROP_IGNORE_WORD_MSG =
"IgnoreWordMessage";
public static final String PROP_IGNORE_WORD_MSG_DEFAULT = "Ignore
this word.";

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating the word should be added to the system
 * dictionary.
 */
public static final String PROP_ADD_TO_DICTIONARY_MSG =
"AddToDictionaryMessage";
public static final String PROP_ADD_TO_DICTIONARY_MSG_DEFAULT =
"Add \"{0}\" to the dictionary.";

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating the word should be replaced by an
alternate
 * spelling. This message takes two parameters. The original word and
the
 * suggested replacement word.
 */
public static final String PROP_SUGGESTED SPELLING_MSG =
"SuggestedSpellingMessage";
public static final String PROP_SUGGESTED SPELLING_MSG_DEFAULT =
"Change the word \"{0}\" to \"{1}\".";

// term, actor, domain phrase issues and position text

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * issue text indicating a phrase may be a glossary term, actor or
domain
 * object/property. The message contains a single parameter for the
phrase.
 */
public static final String PROP_TERM_ACTOR_DOMAIN_MSG =
"TermActorDomainObjectMessage";

```

```

 public static final String PROP_TERM_ACTOR_DOMAIN_MSG_DEFAULT = "The
phrase \"{0}\" is a potential glossary term, actor, or domain
object/property";

 /**
 * The name of the property in the LexicalAssistant.properties file
for the
 * issue text indicating a phrase may be a glossary term or domain
 * object/property. The message contains a single parameter for the
phrase.
 */
public static final String PROP_TERM_DOMAIN_MSG =
"TermDomainObjectMessage";
public static final String PROP_TERM_DOMAIN_MSG_DEFAULT = "The phrase
\"{0}\" is a potential glossary term, or domain object/property.";

 /**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating to ignore the phrase.
 */
public static final String PROP_IGNORE_PHRASE_MSG =
"IgnorePhraseMessage";
public static final String PROP_IGNORE_PHRASE_MSG_DEFAULT = "Ignore
this phrase.";

 /**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating to add the phrase to the project
glossary.
 */
public static final String PROP_ADD_TO_GLOSSARY_MSG =
"AddToGlossary";
public static final String PROP_ADD_TO_GLOSSARY_MSG_DEFAULT =
"Add \"{0}\" to the project glossary.";

 /**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating to add the phrase as an actor to the
project.
 */
public static final String PROP_ADD_AS_ACTOR_MSG = "AddAsActor";
public static final String PROP_ADD_AS_ACTOR_MSG_DEFAULT =
"Add \"{0}\" as an actor to the project.";

```

```

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * note text indicating the use of a word sense by one of the words
in the
 * processed text.
 */
public static final String PROP_SENSE_USE_MSG = "SenseUseMessage";
public static final String PROP_SENSE_USE_MSG_DEFAULT = "The
word \"\{0\}\" was guessed to be the sense \"\{1\}\" meaning \"\{2\}\";

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * note text indicating the use of a word sense by multiple words in
the
 * processed text.
 */
public static final String PROP_MULTI_SENSE_USE_MSG =
"MultiSenseUseMessage";
public static final String PROP_MULTI_SENSE_USE_MSG_DEFAULT = "The
words \"\{0\}\" were guessed to be the sense \"\{1\}\" meaning \"\{2\}\";

// complex sentences
/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * issue text indicating a sentence may be overly complex.
 */
public static final String PROP_COMPLEX_TEXT_MSG =
"ComplexTextMessage";
public static final String PROP_COMPLEX_TEXT_MSG_DEFAULT = "The
text \"\{0\}\" in the \{1\} is complex and may be hard to understand.";

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * syntax tree depth complexity setting. Text with a deeper syntax
level
 * will be annotated with an issue that the text may be overly
complex.
 */
public static final String PROP_COMPLEXITY_DEPTH = "ComplexityDepth";
public static final int PROP_COMPLEXITY_DEPTH_DEFAULT = 12;

// vague word use
/**

```

```

 * The name of the property in the LexicalAssistant.properties file
for the
 * issue text indicating a word may be overly vague.
 */
public static final String PROP_VAGUE_WORD_MSG = "VagueWordMessage";
public static final String PROP_VAGUE_WORD_MSG_DEFAULT = "The
word \"\{0\}\" in the \{1\} is vague and may lead to ambiguity.";

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * information content threshold for detecting vague words.
 */
public static final String PROP_INFO_CONTENT_THRESHOLD =
"InfoContentThreshold";
public static final double PROP_INFO_CONTENT_THRESHOLD_DEFAULT =
0.50;

/**
 * The name of the property in the LexicalAssistant.properties file
for the
 * position text indicating an alternate more specific word.
 */
public static final String PROP_SUGGESTED_MORE_SPECIFIC_WORD_MSG =
"SuggestedMoreSpecificWordMessage";
public static final String
PROP_SUGGESTED_MORE_SPECIFIC_WORD_MSG_DEFAULT = "Change the
word \"\{0\}\" to \"\{1\}\\".';

private final DictionaryRepository dictionaryRepository;
private final ProjectCommandFactory projectCommandFactory;
private final ProjectRepository projectRepository;
private final NLPProcessorFactory nlpProcessorFactory;

/**
 * @param commandHandler -
 *          handler for executing annotation commands.
 * @param projectCommandFactory
 * @param annotationCommandFactory -
 *          factory for creating commands to add annotations to the
goal.
 * @param annotationRepository
 * @param projectRepository
 * @param dictionaryRepository
 * @param nlpProcessorFactory -
 *          natural language processing factory.
 */

```

```

public LexicalAssistant(CommandHandler commandHandler,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository annotationRepository, ProjectRepository
projectRepository,
    DictionaryRepository dictionaryRepository, NLPProcessorFactory
nlpProcessorFactory) {
    super(LexicalAssistant.class.getName(), commandHandler,
annotationCommandFactory,
    annotationRepository);
    this.projectCommandFactory = projectCommandFactory;
    this.projectRepository = projectRepository;
    this.nlpProcessorFactory = nlpProcessorFactory;
    this.dictionaryRepository = dictionaryRepository;
}

protected NLPProcessorFactory getNLPProcessorFactory() {
    return nlpProcessorFactory;
}

protected ProjectCommandFactory getProjectCommandFactory() {
    return projectCommandFactory;
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}

/**
 * Analyze the given text for complexity.
 *
 * @param assistantUser -
 *          the user to mark as the creator of annotations
 * @param projectOrDomain
 * @param thingBeingAnalyzed
 * @param annotatableEntityPropertyName -
 *          The name of the text property in the thing being
analyzed of
 *          the supplied nlpText.
 * @param nlpText -
 *          the text to be analyzed in an NLP Text format, already
processed.
 * @throws Exception

```

```

    */

    public void findPotentialComplexSentences(User assistantUser,
ProjectOrDomain projectOrDomain,
    ProjectOrDomainEntity thingBeingAnalyzed, String
annotatableEntityPropertyName,
    NLPText nlpText) throws Exception {

        if (nlpText.is(GrammaticalStructureLevel.PARAGRAPH)) {
            for (NLPText sentence : nlpText.getChildren()) {
                findPotentialComplexSentences(assistantUser, projectOrDomain,
thingBeingAnalyzed,
                    annotatableEntityPropertyName, sentence);
            }
        } else if (nlpText.is(GrammaticalStructureLevel.SENTENCE)) {
            Integer maxDepth =
getResourceBundleHelper().getInteger(PROP_COMPLEXITY_DEPTH,
PROP_COMPLEXITY_DEPTH_DEFAULT);
            NLPProcessor<Integer> depthFinder = getNLPProcessorFactory()
                .getConstituentTreeDepthFinder();

            Integer depth = depthFinder.process(nlpText);
            if (depth > maxDepth) {
                addComplexityIssue(projectOrDomain, assistantUser,
thingBeingAnalyzed, nlpText,
                    annotatableEntityPropertyName);
            }
        }
    }

    /**
     * Analyze the given text for possible glossary terms. If any are
found add
     * issues to the supplied "thingBeingAnalyzed".
     *
     * @param assistantUser -
     *          the user to mark as the creator of annotations
     * @param projectOrDomain
     * @param thingBeingAnalyzed
     * @param nlpText -
     *          the text to be analyzed in an NLP Text format, already
processed.
     * @throws Exception
     */
    public void findPossibleGlossaryTerms(User assistantUser,
ProjectOrDomain projectOrDomain,
    ProjectOrDomainEntity thingBeingAnalyzed, NLPText nlpText) throws
Exception {

```

```

// TODO: adding a note is for checking the nlp processing and should
be
// removed or made configurable
addNLPNote(projectOrDomain, assistantUser, thingBeingAnalyzed,
"Phrase Structure:\n"
+
getNLPPProcessorFactory().getConstituentTreePrinter().process(nlpText))
;
addNLPNote(projectOrDomain, assistantUser, thingBeingAnalyzed,
"Syntax Dependencies:\n"
+
getNLPPProcessorFactory().getDependencyPrinter().process(nlpText));
addNLPNote(projectOrDomain, assistantUser, thingBeingAnalyzed,
"Semantic Roles:\n"
+
getNLPPProcessorFactory().getSemanticRolePrinter().process(nlpText));

// add a note for each word sense used by which words
Map<Sense, List<NLPText>> senseMap = new HashMap<Sense,
List<NLPText>>(nlpText.getLeaves()
.size());
for (NLPText nlpWord : nlpText.getLeaves()) {
Sense sense = nlpWord.getDictionaryWordSense();
if (sense != null) {
List<NLPText> senseUse = senseMap.get(sense);
if (senseUse == null) {
senseUse = new ArrayList<NLPText>(5);
}
senseUse.add(nlpWord);
}
}
for (Sense sense : senseMap.keySet()) {
List<NLPText> words = senseMap.get(sense);
if (words.size() > 1) {
StringBuilder wordsString = new StringBuilder();
wordsString.append(words.get(0).getText());
for (NLPText word : words.subList(1, words.size())) {
wordsString.append(", ");
wordsString.append(word.getText());
}
addNLPNote(projectOrDomain, assistantUser, thingBeingAnalyzed,
createMessage(
PROP_SENSE_USE_MSG, PROP_SENSE_USE_MSG_DEFAULT, wordsString,
sense
.toString(), sense.getSynset().getDefinition()));
} else {

```

```

addNLPNote(projectOrDomain, assistantUser, thingBeingAnalyzed,
createMessage(
PROP_MULTI_SENSE_USE_MSG, PROP_MULTI_SENSE_USE_MSG_DEFAULT,
words.get(0)
.getText(), sense.toString(),
sense.getSynset().getDefinition()));
}

Set<NLPText> potentialTerms = new
HashSet<NLPText>(nlpText.getLeaves().size());
for (NLPText nounPhrase :
getNLPPProcessorFactory().getNounPhraseFinder().process(nlpText)) {
// TODO: the "rules" for what can be a glossary term
// should be externalized to make it easy to customize

// TODO: use named entity information
// TODO: use word sense information
if (nounPhrase.getLeaves().size() == 1) {
NLPText singleWord = nounPhrase.getLeaves().iterator().next();
if (singleWord.is(ParseTag.NNP) || singleWord.is(ParseTag.NNPS)) {
// add an issue for single proper noun phrases if the phrase
// isn't a sub phrase of a larger phrase in the same text

// TODO: use word sense category?
potentialTerms.add(singleWord);
}
} else {
// walk the parse tree and ignore phrases that contain full
// clauses or other clause types.
boolean addAsPotentialTerm = true;
List<NLPText> todo = new ArrayList<NLPText>();
todo.add(nounPhrase);
while (!todo.isEmpty()) {
NLPText current = todo.remove(0);
if (current.is(GrammaticalStructureLevel.CLAUSE,
|| current.in(ParseTag.PP, ParseTag.VP, ParseTag.INTJ,
ParseTag.LST,
ParseTag.UCP, ParseTag.WHADJP, ParseTag.WHAVP, ParseTag.WHNP,
ParseTag.WHPP)
|| current.in(PartOfSpeech.NUMBER, PartOfSpeech.PUNCTUATION,
PartOfSpeech.SYMBOL)) {
addAsPotentialTerm = false;
break;
} else {
todo.addAll(current.getChildren());
// filter out things that are statements:

```

```

// "his shoes"
//
for (NLPText word : current.getLeaves()) {
    if (word.is(ParseTag.PRP$)) {
        addAsPotentialTerm = false;
    }
}
if (addAsPotentialTerm) {
    potentialTerms.add(nounPhrase);
}
}

// TODO: should this take into account the tags or part of speech of
// each word?
// filter out phrases that are sub-phrases of a longer phrase
Set<NLPText> filteredTerms = new HashSet<NLPText>(potentialTerms);
for (NLPText outerPhrase : potentialTerms) {
    for (NLPText innerPhrase : potentialTerms) {
        if (!outerPhrase.equals(innerPhrase)) {
            if (innerPhrase.getText().toLowerCase().contains(
                outerPhrase.getText().toLowerCase())) {
                filteredTerms.remove(outerPhrase);
                break;
            }
            if (outerPhrase.getText().toLowerCase().contains(
                innerPhrase.getText().toLowerCase())) {
                filteredTerms.remove(innerPhrase);
                break;
            }
        }
    }
}
if (!filteredTerms.isEmpty()) {
    for (NLPText term : filteredTerms) {
        try {
            // if a term exists then add a reference from the thing
            // being analyzed.
            GlossaryTerm glossaryTerm = getProjectRepository()
                .findGlossaryTermForProjectOrDomain(projectOrDomain,
term.getText());
            addProjectOrDomainEntityAsRefererToGlossaryTerm(assistantUser,
glossaryTerm,
                thingBeingAnalyzed);
        } catch (NoSuchGlossaryTermException e) {
            // if the phrase matches an actor name. TODO: ignore or add
            // an issue to actor containers to add a referece to the
            // actor?
            boolean addIssue = true;
            Set<String> namesToMatch = new HashSet<String>();
            namesToMatch.add(term.getText());
            if (!term.isLeaf() &&
term.getLeaves().get(0).in(PartOfSpeech.DETERMINER)) {
                namesToMatch.add(term.getTextRange(1));
            }
            for (String str : namesToMatch) {
                try {
                    getProjectRepository().findActorByProjectOrDomainAndName(
                        projectOrDomain, str);
                    addIssue = false;
                } catch (NoSuchActorException e2) {
                }
            }
            if (addIssue) {
                // add an issue to create a term
                addGlossaryIssue(projectOrDomain, assistantUser,
thingBeingAnalyzed, term);
            }
        }
    }
}

/**
 * Check the spelling of the supplied text adding LexicalIssues to
 * the
 * "thingBeingAnalyzed" for words not found in the dictionary.
 *
 * @param groupingObject -
 *      the object that acts as the owner of the created
 * annotations.
 * @param assistantUser -
 *      the user to assign as the creator/editor of any issues
 * and
 *      positions created.
 * @param projectOrDomain
 * @param thingBeingAnalyzed
 * @param annotatableEntityPropertyName
 * @param nlpText -
 *      the text to be analyzed in an NLP Text format, already
 *      processed.
 * @throws Exception
 */

```

```

public void checkSpelling(User assistantUser, ProjectOrDomain
projectOrDomain,
    ProjectOrDomainEntity thingBeingAnalyzed, String
annotatableEntityPropertyName,
    NLPText nlpText) throws Exception {
    NLPProcessor<Boolean> spellChecker =
getNLPProcessorFactory().getSpellingChecker();
    NLPProcessor<Collection<NLPText>> similarWordFinder =
getNLPProcessorFactory()
        .getSimilarWordFinder();

    for (NLPText word : nlpText.getLeaves()) {
        if (!word.in(PartOfSpeech.PUNCTUATION, PartOfSpeech.NUMBER,
PartOfSpeech.SYMBOL)
            && !word.in(ParseTag.POS, ParseTag.CD)) {
            log.debug("analyzing word : '" + word + "' pos: " +
word.getPartOfSpeech());
            if (!spellChecker.process(word)) {
                addSpellingIssue(similarWordFinder, projectOrDomain,
assistantUser,
                    thingBeingAnalyzed, word, annotatableEntityPropertyName);
            }
        }
    }

    /**
     * Check the senses of the words in the nlpText for vague words (low
     * information content) and suggest some more specific words to use
     * instead.
     *
     * @param assistantUser
     * @param projectOrDomain
     * @param thingBeingAnalyzed
     * @param annotatableEntityPropertyName
     * @param nlpText
     * @throws Exception
     */
    public void checkVagueWordUse(User assistantUser, ProjectOrDomain
projectOrDomain,
        ProjectOrDomainEntity thingBeingAnalyzed, String
annotatableEntityPropertyName,
        NLPText nlpText) throws Exception {
        NLPProcessor<Collection<NLPText>> moreSpecificWordSugester =
getNLPProcessorFactory()
            .getMoreSpecificWordSugester();
        double infoContentThreshold = getResourceBundleHelper().getDouble(

```

```

        PROP_INFO_CONTENT_THRESHOLD, PROP_INFO_CONTENT_THRESHOLD_DEFAULT);
Linkdef linkType = getDictionaryRepository().findLinkDef(1L);

        // Proper nouns may have entity#n#1 assigned as the sense, if that's
the
        // case don't mark it as vague.
        Sense rootNounSense = getDictionaryRepository().findSense("entity",
PartOfSpeech.NOUN, 1);

        for (NLPText word : nlpText.getLeaves()) {
            if ((!word.isNamedEntity())
                && (word.getDictionaryWordSense() != null)
                && (word.getDictionaryWordSense().getSynset() != null)
                && !(word.getDictionaryWordSense().equals(rootNounSense) &&
word.in(
                    ParseTag.NNP, ParseTag.NNPS))) {
                double infoContent = getDictionaryRepository().infoContent(
                    word.getDictionaryWordSense().getSynset(), linkType);
                if (infoContent < infoContentThreshold) {
                    addVagueWordUseIssue(moreSpecificWordSugester, projectOrDomain,
assistantUser,
                        thingBeingAnalyzed, word, annotatableEntityPropertyName);
                }
            }
        }

    /**
     * Analyze updated text of an entity to see if a user has changed the
     * supplied property fixing lexical issues manually and remove
     * annotations
     * that are no longer relevant.
     *
     * @param assistantUser
     * @param projectOrDomain
     * @param thingBeingAnalyzed
     * @param annotatableEntityPropertyName
     * @param nlpText
     * @throws Exception
     */
    public void removeUnneedLexicalIssues(User assistantUser,
ProjectOrDomain projectOrDomain,
    ProjectOrDomainEntity thingBeingAnalyzed, String
annotatableEntityPropertyName,
    NLPText nlpText) throws Exception {
        RemoveUnneedLexicalIssuesCommand command =
getProjectCommandFactory()

```

```

    .newRemoveUnneedLexicalIssuesCommand();
command.setAnnotatableEntityPropertyName(annotatableEntityPropertyName);
me);
command.setNlpText(nlpText);
command.setThingBeingAnalyzed(thingBeingAnalyzed);
command.setProjectOrDomain(projectOrDomain);
command.setEditedBy(assistantUser);
command = getCommandHandler().execute(command);
}

protected void addGlossaryIssue(ProjectOrDomain projectOrDomain, User
assistantUser,
ProjectOrDomainEntity thingBeingAnalyzed, NLPText term) throws
Exception {
EditLexicalIssueCommand editIssueCommand =
getAnnotationCommandFactory()
.newEditLexicalIssueCommand();
// TODO: add issues/positions for partial matches by removing
// determiners from the text.
try {
editIssueCommand.setIssue(getAnnotationRepository().findLexicalIssu
e(projectOrDomain,
thingBeingAnalyzed, term.getText()));
editIssueCommand.setAnnotatable(thingBeingAnalyzed);
editIssueCommand.setEditedBy(assistantUser);
editIssueCommand = getCommandHandler().execute(editIssueCommand);
} catch (NoSuchAnnotationException e) {
editIssueCommand.setGroupingObject(projectOrDomain);
editIssueCommand.setWord(term.getText());
editIssueCommand.setText(createMessage(PROP_TERM_ACTOR_DOMAIN_MSG,
PROP_TERM_ACTOR_DOMAIN_MSG_DEFAULT, term.getText()));
editIssueCommand.setMustBeResolved(true);
editIssueCommand.setAnnotatable(thingBeingAnalyzed);
editIssueCommand.setEditedBy(assistantUser);
editIssueCommand = getCommandHandler().execute(editIssueCommand);
LexicalIssue issue = (LexicalIssue) editIssueCommand.getIssue();
addSimplePositionToIssue(projectOrDomain, assistantUser, issue,
createMessage(
PROP_IGNORE_PHRASE_MSG, PROP_IGNORE_PHRASE_MSG_DEFAULT));
addAddWordToGlossaryPositionToIssue(assistantUser, projectOrDomain,
issue);
addAddActorToProjectPositionToIssue(assistantUser, projectOrDomain,
issue);
}
}

```

```

protected void addSpellingIssue(NLPProcessor<Collection<NLPText>>
similarWordFinder,
ProjectOrDomain projectOrDomain, User assistantUser,
ProjectOrDomainEntity thingBeingAnalyzed, NLPText word,
String annotatableEntityPropertyName) throws Exception {
try {
getAnnotationRepository().findLexicalIssue(projectOrDomain,
thingBeingAnalyzed,
word.getText(), annotatableEntityPropertyName);
} catch (NoSuchAnnotationException e) {
EditLexicalIssueCommand editIssueCommand =
getAnnotationCommandFactory()
.newEditLexicalIssueCommand();
editIssueCommand.setGroupingObject(projectOrDomain);
editIssueCommand.setWord(word.getText());
editIssueCommand.setAnnotatableEntityPropertyName(annotatableEntity
PropertyName);
editIssueCommand.setText(createMessage(PROP_UNKNOWN_WORD_MSG,
PROP_UNKNOWN_WORD_MSG_DEFAULT, word,
annotatableEntityPropertyName));
editIssueCommand.setMustBeResolved(true);
editIssueCommand.setAnnotatable(thingBeingAnalyzed);
editIssueCommand.setEditedBy(assistantUser);
editIssueCommand = getCommandHandler().execute(editIssueCommand);
LexicalIssue issue = (LexicalIssue) editIssueCommand.getIssue();
addAddWordToDictionaryPositionToIssue(assistantUser, issue);
addSimplePositionToIssue(projectOrDomain, assistantUser, issue,
createMessage(
PROP_IGNORE_WORD_MSG, PROP_IGNORE_WORD_MSG_DEFAULT));
addAddWordToGlossaryPositionToIssue(assistantUser, projectOrDomain,
issue);
addAddActorToProjectPositionToIssue(assistantUser, projectOrDomain,
issue);
for (NLPText similarWord : similarWordFinder.process(word)) {
addChangeSpellingPositionToIssue(assistantUser, issue,
createMessage(
PROP_SUGGESTED_SPELLING_MSG,
PROP_SUGGESTED_SPELLING_MSG_DEFAULT, word,
similarWord.getText(), similarWord.getText());
}
}
}

protected void addComplexityIssue(ProjectOrDomain projectOrDomain,
User assistantUser,
ProjectOrDomainEntity thingBeingAnalyzed, NLPText text,

```

```

String annotatableEntityPropertyName) throws Exception {
try {
getAnnotationRepository()
    .findIssue(projectOrDomain, thingBeingAnalyzed, text.getText());
} catch (NoSuchAnnotationException e) {
// TODO: this uses a lexical issue, but isn't word oriented,
// although it is property related so an issue with
// annotatableEntityPropertyName is needed. Should add an in
between
// issue type PropertyRelatedIssue
EditLexicalIssueCommand editIssueCommand =
getAnnotationCommandFactory()
    .newEditLexicalIssueCommand();
editIssueCommand.setGroupingObject(projectOrDomain);
editIssueCommand.setAnnotatableEntityPropertyName(annotatableEntity
PropertyName);
editIssueCommand.setText(createMessage(PROP_COMPLEX_TEXT_MSG,
    PROP_COMPLEX_TEXT_MSG_DEFAULT, text,
    annotatableEntityPropertyName));
editIssueCommand.setMustBeResolved(true);
editIssueCommand.setAnnotatable(thingBeingAnalyzed);
editIssueCommand.setEditedBy(assistantUser);
editIssueCommand = getCommandHandler().execute(editIssueCommand);
LexicalIssue issue = (LexicalIssue) editIssueCommand.getIssue();
addAddWordToDictionaryPositionToIssue(assistantUser, issue);
addSimplePositionToIssue(projectOrDomain, assistantUser, issue,
createMessage(
    PROP_IGNORE_WORD_MSG, PROP_IGNORE_WORD_MSG_DEFAULT));
}
}

protected void addVagueWordUseIssue(NLPProcessor<Collection<NLPText>>
moreSpecificWordFinder,
ProjectOrDomain projectOrDomain, User assistantUser,
ProjectOrDomainEntity thingBeingAnalyzed, NLPText word,
String annotatableEntityPropertyName) throws Exception {
try {
getAnnotationRepository().findLexicalIssue(projectOrDomain,
thingBeingAnalyzed,
word.getText(), annotatableEntityPropertyName);
} catch (NoSuchAnnotationException e) {
EditLexicalIssueCommand editIssueCommand =
getAnnotationCommandFactory()
    .newEditLexicalIssueCommand();
editIssueCommand.setGroupingObject(projectOrDomain);
editIssueCommand.setWord(word.getText());
editIssueCommand.setAnnotatableEntityPropertyName(annotatableEntity
PropertyName);
editIssueCommand.setText(createMessage(PROP_VAGUE_WORD_MSG,
    PROP_VAGUE_WORD_MSG_DEFAULT, word,
    annotatableEntityPropertyName));
editIssueCommand.setMustBeResolved(true);
editIssueCommand.setAnnotatable(thingBeingAnalyzed);
editIssueCommand.setEditedBy(assistantUser);
editIssueCommand = getCommandHandler().execute(editIssueCommand);
LexicalIssue issue = (LexicalIssue) editIssueCommand.getIssue();
addSimplePositionToIssue(projectOrDomain, assistantUser, issue,
createMessage(
    PROP_IGNORE_WORD_MSG, PROP_IGNORE_WORD_MSG_DEFAULT));
for (NLPText moreSpecificWord :
moreSpecificWordFinder.process(word)) {
    addChangeSpellingPositionToIssue(assistantUser, issue,
createMessage(
    PROP_SUGGESTED_MORE_SPECIFIC_WORD_MSG,
    PROP_SUGGESTED_MORE_SPECIFIC_WORD_MSG_DEFAULT, word,
moreSpecificWord
    .getText()), moreSpecificWord.getText());
}
}

// TODO: create a new annotation type for nlp meta data
protected Note addNLPNote(Object groupingObject, User assistantUser,
ProjectOrDomainEntity thingBeingAnalyzed, String noteText) throws
Exception {
return super.addNote(groupingObject, assistantUser,
thingBeingAnalyzed, noteText);
}

protected Position addChangeSpellingPositionToIssue(User
assistantUser, Issue issue,
String positionText, String proposedWord) throws Exception {
EditChangeSpellingPositionCommand editPositionCommand =
getAnnotationCommandFactory()
    .newEditChangeSpellingPositionCommand();
editPositionCommand.setIssue(issue);
editPositionCommand.setText(positionText);
editPositionCommand.setEditedBy(assistantUser);
editPositionCommand.setProposedWord(proposedWord);
editPositionCommand =
getCommandHandler().execute(editPositionCommand);
return editPositionCommand.getPosition();
}
}

```

```

protected Position addAddWordToDictionaryPositionToIssue(User
assistantUser, LexicalIssue issue)
throws Exception {
EditAddWordToDictionaryPositionCommand editPositionCommand =
getAnnotationCommandFactory()
.newEditAddWordToDictionaryPositionCommand();
editPositionCommand.setIssue(issue);
editPositionCommand.setText(createMessage(PROP_ADD_TO_DICTIONARY_MSG
,
PROP_ADD_TO_DICTIONARY_MSG_DEFAULT, issue.getWord()));
editPositionCommand.setEditedBy(assistantUser);
editPositionCommand =
getCommandHandler().execute(editPositionCommand);
return editPositionCommand.getPosition();
}

protected Position addAddWordToGlossaryPositionToIssue(User
assistantUser,
ProjectOrDomain projectOrDomain, LexicalIssue issue) throws
Exception {
EditAddWordToGlossaryPositionCommand editPositionCommand =
getProjectCommandFactory()
.newEditAddWordToGlossaryPositionCommand();
editPositionCommand.setEditedBy(assistantUser);
editPositionCommand.setIssue(issue);
editPositionCommand.setProjectOrDomain(projectOrDomain);
editPositionCommand.setText(createMessage(PROP_ADD_TO_GLOSSARY_MSG,
PROP_ADD_TO_GLOSSARY_MSG_DEFAULT, issue.getWord()));
editPositionCommand =
getCommandHandler().execute(editPositionCommand);
return editPositionCommand.getPosition();
}

protected Position addAddActorToProjectPositionToIssue(User
assistantUser,
ProjectOrDomain projectOrDomain, LexicalIssue issue) throws
Exception {
EditAddActorToProjectPositionCommand editPositionCommand =
getProjectCommandFactory()
.newEditAddActorToProjectPositionCommand();
editPositionCommand.setEditedBy(assistantUser);
editPositionCommand.setIssue(issue);
editPositionCommand.setProjectOrDomain(projectOrDomain);
editPositionCommand.setText(createMessage(PROP_ADD_AS_ACTOR_MSG,
PROP_ADD_AS_ACTOR_MSG_DEFAULT, issue.getWord()));
}

```

```

editPositionCommand =
getCommandHandler().execute(editPositionCommand);
return editPositionCommand.getPosition();
}

protected void addProjectOrDomainEntityAsRefererToGlossaryTerm(User
assistantUser,
GlossaryTerm glossaryTerm, ProjectOrDomainEntity
thingBeingAnalyzed) throws Exception {
EditGlossaryTermCommand command =
getProjectCommandFactory().newEditGlossaryTermCommand();
Set<ProjectOrDomainEntity> entities = new
HashSet<ProjectOrDomainEntity>(1);
entities.add(thingBeingAnalyzed);
command.setAddReferers(entities);
command.setEditedBy(assistantUser);
command.setGlossaryTerm(glossaryTerm);
command = getCommandHandler().execute(command);
}
}

```

lexicalissue.java

```

/*
 * $Id: LexicalIssue.java,v 1.7 2009/03/22 11:08:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * An issue related to a word used in the text of an entity.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue")

```

```

@XmlRootElement(name = "lexicalIssue", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "lexicalIssue", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class LexicalIssue extends IssueImpl {
    static final long serialVersionUID = 0L;

    private String annotatableEntityPropertyName;
    private String word;

    /**
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     * annotations.
     * @param text -
     *          the text message of the annotation
     * @param mustBeResolved -
     *          flag indicating if this issue must be resolved before
     * the
     *          project can be complete.
     * @param createdBy -
     *          the user that created the issue
     * @param annotatableEntityPropertyName -
     *          the name of the property in the annotatable entity that
     *          contains the word in question.
     * @param word -
     *          the word the issue is about.
    */
    public LexicalIssue(Object groupingObject, String text, boolean
mustBeResolved, User createdBy,
    String annotatableEntityPropertyName, String word) {
        super(LexicalIssue.class.getName(), groupingObject, text,
mustBeResolved, createdBy);
        setWord(word);
        setAnnotatableEntityPropertyName(annotatableEntityPropertyName);
    }

    protected LexicalIssue() {
        // for hibernate
    }

    /**
     * @return The word the issue is about.
    */
    @XmlAttribute(name = "word")
    public String getWord() {
        return word;
    }

    }
}

    /**
     * set the word the issue is about.
    */
    public void setWord(String word) {
        this.word = word;
    }

    /**
     * @return The name of the property on the annotatable issue where
     * the word
     *          occurs.
    */
    @XmlAttribute(name = "propertyName")
    public String getAnnotatableEntityPropertyName() {
        return annotatableEntityPropertyName;
    }

    /**
     * @param annotatableEntityPropertyName
    */
    public void setAnnotatableEntityPropertyName(String
annotatableEntityPropertyName) {
        this.annotatableEntityPropertyName = annotatableEntityPropertyName;
    }
}


```

lexicalmatchingrule.java

```

/*
 * $Id: LexicalMatchingRule.java,v 1.3 2009/02/11 09:02:54 rregan Exp
 *
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

```

```

/**
 * Match a specific string or one of a specific string.
 *
 * @author ron
 */
public class LexicalMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
        Logger.getLogger(LexicalMatchingRule.class);

    // list of prepositions that the match must be one of.
    private final List<String> matchingFilter;

    /**
     * Create a rule that matches one of the supplied words
     *
     * @param filter -
     *          a space delimited list of words to match.
     */
    public LexicalMatchingRule(String filter) {
        matchingFilter = Arrays.asList(filter.split(" "));
    }

    /**
     *
     */
    @Override
    public void match(DictionaryRepository dictionaryRepository, NLPText
verb,
                      ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException {
        NLPText word = textIterator.next();
        if (matchingFilter.size() > 0) {
            for (String matchWord : matchingFilter) {
                // TODO: match lemma?
                if (word.getText().equalsIgnoreCase(matchWord)) {
                    log.debug("matched: " + word.getText());
                    return;
                }
            }
        } else {
            log.debug("matched: " + word.getText());
            return;
        }
        throw SemanticRoleLabelerException.matchFailed(this, word);
    }
}

```

```

@Override
public String toString() {
    return getClass().getSimpleName() + ":" + matchingFilter;
}
}

```

lexlinkref.java

```

/*
 * $Id: Lexlinkref.java,v 1.2 2009/01/03 10:24:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.AttributeOverride;
import javax.persistence.AttributeOverrides;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

/**
 * @author ron
 */
@Entity
@Table(name = "lexlinkref")
@XmlRootElement(name = "lexlink")
public class Lexlinkref implements Comparable<Lexlinkref>,
Serializable {
    static final long serialVersionUID = 0;

    private LexlinkrefId id;
    private Synset fromSynset;
    private Word fromWord;
    private Synset toSynset;
    private Word toWord;
    private Linkdef linkType;

    public Lexlinkref() {

```

```
}

public Lexlinkref(LexlinkrefId id) {
    this.id = id;
}

@EmbeddedId
@AttributeOverrides( {
    @AttributeOverride(name = "wordlid", column = @Column(name =
"wordlid", nullable = false, precision = 6, scale = 0)),
    @AttributeOverride(name = "synsetlid", column = @Column(name =
"synsetlid", nullable = false, precision = 9, scale = 0)),
    @AttributeOverride(name = "word2id", column = @Column(name =
"word2id", nullable = false, precision = 6, scale = 0)),
    @AttributeOverride(name = "synset2id", column = @Column(name =
"synset2id", nullable = false, precision = 9, scale = 0)),
    @AttributeOverride(name = "linkid", column = @Column(name =
"linkid", nullable = false, precision = 2, scale = 0)) })
public LexlinkrefId getId() {
    return this.id;
}

public void setId(LexlinkrefId id) {
    this.id = id;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "synsetlid", insertable = false, updatable =
false)
@XmlTransient
public Synset getFromSynset() {
    return fromSynset;
}

public void setFromSynset(Synset fromSynset) {
    this.fromSynset = fromSynset;
}

@ManyToOne(targetEntity = Word.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "wordlid", insertable = false, updatable = false)
@XmlTransient
public Word getFromWord() {
    return fromWord;
}
```

```
public void setFromWord(Word fromWord) {
    this.fromWord = fromWord;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "synset2id", insertable = false, updatable =
false)
@XmlTransient
public Synset getToSynset() {
    return toSynset;
}

public void setToSynset(Synset toSynset) {
    this.toSynset = toSynset;
}

@ManyToOne(targetEntity = Word.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "word2id", insertable = false, updatable = false)
@XmlTransient
public Word getToWord() {
    return toWord;
}

public void setToWord(Word toWord) {
    this.toWord = toWord;
}

@ManyToOne(targetEntity = Linkdef.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "linkid", insertable = false, updatable = false)
@XmlTransient
public Linkdef getLinkType() {
    return linkType;
}

public void setLinkType(Linkdef linkType) {
    this.linkType = linkType;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
```

```

    return false;
}
if (getClass() != obj.getClass()) {
    return false;
}
final Lexlinkref other = (Lexlinkref) obj;

if ((getId() != null) && (other.getId() != null)) {
    return getId().equals(other.getId());
}

if (getFromSynset() == null) {
    if (other.getFromSynset() != null) {
        return false;
    }
} else if (!getFromSynset().equals(other.getFromSynset())) {
    return false;
}

if (getFromWord() == null) {
    if (other.getFromWord() != null) {
        return false;
    }
} else if (!getFromWord().equals(other.getFromWord())) {
    return false;
}

if (getToSynset() == null) {
    if (other.getToSynset() != null) {
        return false;
    }
} else if (!getToSynset().equals(other.getToSynset())) {
    return false;
}

if (getToWord() == null) {
    if (other.getToWord() != null) {
        return false;
    }
} else if (!getToWord().equals(other.getToWord())) {
    return false;
}

if (getLinkType() == null) {
    if (other.getLinkType() != null) {
        return false;
    }
}

```

```

    } else if (!getLinkType().equals(other.getLinkType())) {
        return false;
    }

    return true;
}

@Override
public int compareTo(Lexlinkref o) {
    return getId().compareTo(o.getId());
}
}

```

lexlinkrefid.java

```

/*
 * $Id: LexlinkrefId.java,v 1.2 2009/01/03 10:24:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Embeddable;

/**
 * @author ron
 */
@Embeddable
public class LexlinkrefId implements Comparable<LexlinkrefId>, Serializable {
    static final long serialVersionUID = 0;

    private Long wordlid;
    private Long synsetlid;
    private Long word2id;
    private Long synset2id;
    private Long linkid;

    public LexlinkrefId() {
    }

    public LexlinkrefId(Long wordlid, Long synsetlid, Long word2id, Long
synset2id, Long linkid) {
        this.wordlid = wordlid;
    }
}

```

```

this.synset1id = synset1id;
this.word2id = word2id;
this.synset2id = synset2id;
this.linkid = linkid;
}

@Column(name = "wordlid", nullable = false)
public Long getWordlid() {
    return this.wordlid;
}

public void setWordlid(Long wordlid) {
    this.wordlid = wordlid;
}

@Column(name = "synset1id", nullable = false)
public Long getSynset1id() {
    return this.synset1id;
}

public void setSynset1id(Long synset1id) {
    this.synset1id = synset1id;
}

@Column(name = "word2id", nullable = false)
public Long getWord2id() {
    return this.word2id;
}

public void setWord2id(Long word2id) {
    this.word2id = word2id;
}

@Column(name = "synset2id", nullable = false)
public Long getSynset2id() {
    return this.synset2id;
}

public void setSynset2id(Long synset2id) {
    this.synset2id = synset2id;
}

@Column(name = "linkid", nullable = false)
public Long getLinkid() {
    return this.linkid;
}
}

public void setLinkid(Long linkid) {
    this.linkid = linkid;
}

@Override
public boolean equals(Object other) {
    if ((this == other)) {
        return true;
    }
    if ((other == null)) {
        return false;
    }
    if (!(other instanceof LexlinkrefId)) {
        return false;
    }
    LexlinkrefId castOther = (LexlinkrefId) other;

    return (this.getWordlid() == castOther.getWordlid())
        && (this.getSynset1id() == castOther.getSynset1id())
        && (this.getWord2id() == castOther.getWord2id())
        && (this.getSynset2id() == castOther.getSynset2id())
        && (this.getLinkid() == castOther.getLinkid());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        final int prime = 31;
        int result = 1;
        result = prime * result + getSynset1id().hashCode();
        result = prime * result + getWordlid().hashCode();
        result = prime * result + getSynset2id().hashCode();
        result = prime * result + getWord2id().hashCode();
        result = prime * result + getLinkid().hashCode();
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public int compareTo(LexlinkrefId o) {
    if (this.getSynset1id() != o.getSynset1id()) {
        return (getSynset1id().intValue() - o.getSynset1id().intValue());
    }
    if (this.getSynset2id() != o.getSynset2id()) {

```

```

    return (this.getSynset2id().intValue() -
o.getSynset2id().intValue());
}
if (this.getLinkid() != o.getLinkid()) {
    return (this.getLinkid().intValue() - o.getLinkid().intValue());
}
return 0;
}
}

```

linkdef.java

```

/*
 * $Id: Linkdef.java,v 1.3 2009/02/07 13:36:49 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.annotations.Type;

/**
 * @author ron
 */
@Entity
@Table(name = "linkdef")
public class Linkdef implements Serializable {
    static final long serialVersionUID = 0;

    /**
     * The id of hypernym link definition type.
     */
    public static final int HYPERNYM = 1;

    /**
     * The id of hyponym link definition type.
     */
    public static final int HYPONYM = 2;

    /**

```

```

     * The id of hypernym link definition type.
     */
    public static final int HYPERNYM_INSTANCE = 3;

    /**
     * The id of hyponym link definition type.
     */
    public static final int HYPONYM_INSTANCE = 4;

    private Long linkid;
    private String name;
    private boolean recurses;

    protected Linkdef() {
    }

    protected Linkdef(Long linkid, boolean recurses) {
        this.linkid = linkid;
        this.recurses = recurses;
    }

    protected Linkdef(Long linkid, String name, boolean recurses) {
        this.linkid = linkid;
        this.name = name;
        this.recurses = recurses;
    }

    @Id
    @Column(name = "linkid", unique = true, nullable = false)
    public Long getId() {
        return this.linkid;
    }

    public void setId(Long linkid) {
        this.linkid = linkid;
    }

    @Column(name = "name")
    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Column(name = "recurses", nullable = false)

```

```

@Type(type = "yes_no")
public boolean getRecurses() {
    return this.recurses;
}

public void setRecurses(boolean recurses) {
    this.recurses = recurses;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((name == null) ? 0 : name.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Linkdef other = (Linkdef) obj;
    if (name == null) {
        if (other.name != null) {
            return false;
        }
    } else if (!name.equals(other.name)) {
        return false;
    }
    return true;
}
}

```

loadwordnettaggedglossescommand.java

```

/*
 * $Id: LoadWordNetTaggedGlossesCommand.java,v 1.1 2008/12/13 00:40:08
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

/*
package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.io.InputStream;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;

/**
 * An XML processor for the WordNet synset merged gloss parse and word
sense
 * files.<br>
 * The processor parses the xml and adds {@link SynsetDefinitionWord}s
to the
 * {@link Synset}s<br>
 *
 * @see {http://wordnet.princeton.edu/glosstag-files/}
 * @author ron
 */
public interface LoadWordNetTaggedGlossesCommand extends Command {

    /**
     * @param inputStream -
     *          a stream to a WordNet merged tagged gloss file.
     * @see {http://wordnet.princeton.edu/glosstag-files/}
     */
    public void setInputStream(InputStream inputStream);
}
```

loadwordnettaggedglossescommandimpl.java

```

/*
 * $Id: LoadWordNetTaggedGlossesCommandImpl.java,v 1.1 2008/12/13
00:40:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.io.InputStream;

import org.apache.commons.digester.Digester;
import org.apache.commons.digester.Rule;
import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
```

```

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import org.xml.sax.Attributes;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import edu.harvard.fas.rregan.command.BatchCommand;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetDefinitionWord
Command;
import
edu.harvard.fas.rregan.nlp.dictionary.command.LoadWordNetTaggedGlosses
Command;
import edu.harvard.fas.rregan.request.NoSuchEntityException;

/**
 * An XML processor for the WordNet synset merged gloss parse and word
sense
 * files.<br>
 * The processor parses the xml and adds {@link SynsetDefinitionWord}
to the
 * {@link Synset}s<br>
 *
 * @see {http://wordnet.princeton.edu/glosstag-files/}
 * @author ron
 */
@Controller("wordNetTaggedGlossaryDigester")
@Scope("prototype")
public class LoadWordNetTaggedGlossesCommandImpl extends Digester
implements
LoadWordNetTaggedGlossesCommand {
private static final Logger log =
Logger.getLogger(LoadWordNetTaggedGlossesCommandImpl.class);

private final DictionaryRepository dictionaryRepository;
private final DictionaryCommandFactory dictionaryCommandFactory;
private final CommandHandler commandHandler;

```

```

private InputStream inputStream;

/**
 * @param dictionaryRepository
 * @param dictionaryCommandFactory
 * @param commandHandler
 */
@.Autowired
public LoadWordNetTaggedGlossesCommandImpl(DictionaryRepository
dictionaryRepository,
DictionaryCommandFactory dictionaryCommandFactory, CommandHandler
commandHandler) {
    this.dictionaryRepository = dictionaryRepository;
    this.dictionaryCommandFactory = dictionaryCommandFactory;
    this.commandHandler = commandHandler;

    addRule("*/synset", new DigesterRuleLoggingDecorator(new
SynsetXMLDigesterRule()));
    addRule("*/synset/gloss/def/wf", new DigesterRuleLoggingDecorator(
        new DefinitionWordFormXMLDigesterRule()));
    addRule("*/synset/gloss/def/wf/id", new
DigesterRuleLoggingDecorator(
        new WordFormIdXMLDigesterRule()));
    addRule("*/synset/gloss/def/cf", new DigesterRuleLoggingDecorator(
        new DefinitionColocationFormXMLDigesterRule()));
    addRule("*/synset/gloss/def/cf/id", new
DigesterRuleLoggingDecorator(
        new WordFormIdXMLDigesterRule()));
}

@Override
public void setInputStream(InputStream inputStream) {
    this.inputStream = inputStream;
}

@Override
public void execute() throws Exception {
try {
    // set a parser error handler
    getParser().getXMLReader().setErrorHandler(new TheErrorHandler());
} catch (Exception e) {
    log.warn("Could not set ErrorHandler for logging of xml parse
errors: " + e, e);
}
log.debug("Starting parsing.");
super.parse(inputStream);
log.debug("Ending parsing.");

```

```

}

/**
 * @return
 */
protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}

protected DictionaryCommandFactory getDictionaryCommandFactory() {
    return dictionaryCommandFactory;
}

protected CommandHandler getCommandHandler() {
    return commandHandler;
}

/**
 * @param ofs
 * @param pos
 * @return
 */
protected Synset getSynset(String ofs, String pos) {
    return getDictionaryRepository().findSynset(Long.parseLong(ofs) +
        posToIdBase(pos));
}

/**
 * @param pos
 * @return the base value to add to the synset ofs to get the id of
 * the
 *      synset.
 */
protected long postoIdBase(String pos) {
    if ("n".equals(pos)) {
        return 100000000;
    } else if ("v".equals(pos)) {
        return 200000000;
    } else if ("a".equals(pos)) {
        return 300000000;
    } else if ("s".equals(pos)) {
        return 300000000;
    } else if ("r".equals(pos)) {
        return 400000000;
    }
    throw new RuntimeException("unexpected pos " + pos);
}

```

```

/**
 * The base class for all rules used by this digester.
 */
public static abstract class WordNetTaggedGlossaryDigesterRule
extends Rule {

    /**
     * @return the digester instance that called this rule
     */
    @Override
    public LoadWordNetTaggedGlossesCommandImpl getDigester() {
        return (LoadWordNetTaggedGlossesCommandImpl) super.getDigester();
    }

    private static class SynsetXMLData {
        final private Synset synset;
        private BatchCommand batchCommand;
        String currentColocationId = null;

        protected SynsetXMLData(Synset synset) {
            this.synset = synset;
        }
    }

    private static class SynsetXMLDigesterRule extends
        WordNetTaggedGlossaryDigesterRule {

        @Override
        public void begin(String elementNamespace, String elementName,
            Attributes attributes)
            throws Exception {
            final String ofs = attributes.getValue("ofs");
            final String pos = attributes.getValue("pos");
            SynsetXMLData data = new SynsetXMLData(getDigester().getSynset(ofs,
                pos));
            data.batchCommand =
                getDigester().getDictionaryCommandFactory().newBatchCommand();
            getDigester().push(data);
        }

        @Override
        public void end(String elementNamespace, String elementName) throws
            Exception {
            SynsetXMLData data = (SynsetXMLData) getDigester().pop();
            getDigester().getCommandHandler().execute(data.batchCommand);
        }
    }
}

```

```

}

private static class DefinitionWordFormXMLDigesterRule extends
    WordNetTaggedGlossaryDigesterRule {

    @Override
    public void begin(String elementNamespace, String elementName,
                      Attributes attributes)
        throws Exception {
        final String pos = attributes.getValue("pos");
        final String type = attributes.getValue("type");
        SynsetXMLData data = (SynsetXMLData) getDigester().peek();
        if (data.currentColocationId != null) {
            // a word form ends the last colocation in this synset.
            data.currentColocationId = null;
        }

        EditSynsetDefinitionWordCommand command =
            getDigester().getDictionaryCommandFactory()
                .newEditSynsetDefinitionWordCommand();
        command.setIndex(data.batchCommand.getCommands().size());
        if ("punc".equals(type)) {
            command.setParseTag(ParseTag.PUNC_NON_TERMINATOR);
        } else if ("symb".equals(type)) {
            command.setParseTag(ParseTag.SYMBOL);
        } else {
            command.setParseTag(ParseTag.tagOf(pos));
        }
        command.setSynset(data.synset);
        getDigester().push(command);
    }

    @Override
    public void body(String namespace, String name, String text) throws
        Exception {
        // a colocation element may be ignored if so the top of the stack
        // won't be a command
        if (getDigester().peek() instanceof
            EditSynsetDefinitionWordCommand) {
            EditSynsetDefinitionWordCommand command =
                (EditSynsetDefinitionWordCommand) getDigester()
                    .peek();
            command.setText(text.trim());
        }
    }
}

```

```

@Override
public void end(String elementNamespace, String elementName) throws
Exception {
    // a colocation element may be ignored if so the top of the stack
    // won't be a command
    if (getDigester().peek() instanceof
        EditSynsetDefinitionWordCommand) {
        EditSynsetDefinitionWordCommand command =
            (EditSynsetDefinitionWordCommand) getDigester()
                .pop();
        SynsetXMLData data = (SynsetXMLData) getDigester().peek();
        data.batchCommand.addCommand(command);
    }
}

private static class DefinitionColocationFormXMLDigesterRule extends
    DefinitionWordFormXMLDigesterRule {

    @Override
    public void begin(String elementNamespace, String elementName,
                      Attributes attributes)
        throws Exception {
        final String pos = attributes.getValue("pos");
        final String colocationId = attributes.getValue("coll");
        SynsetXMLData data = (SynsetXMLData) getDigester().peek();
        if (colocationId.equals(data.currentColocationId)) {
            // the primary element of the colocation was already found,
            // ignore the rest of the words as the primary colocation
            // element accounts for all of the words.
        } else {
            // the primary element of the colocation
            data.currentColocationId = colocationId;
            EditSynsetDefinitionWordCommand command = getDigester()
                .getDictionaryCommandFactory().newEditSynsetDefinitionWordCommand();
            command.setIndex(data.batchCommand.getCommands().size());
            command.setParseTag(ParseTag.tagOf(pos));
            command.setSynset(data.synset);
            getDigester().push(command);
        }
    }
}

private static class WordFormIdXMLDigesterRule extends
    WordNetTaggedGlossaryDigesterRule {

```

```

@Override
public void begin(String elementNamespace, String elementName,
Attributes attributes)
throws Exception {
final String senseKey = attributes.getValue("sk");
EditSynsetDefinitionWordCommand command =
(EditSynsetDefinitionWordCommand) getDigester()
.peek();
try {
command.setSense(getDigester().getDictionaryRepository()
.findSensesByWordnetSenseKey(senseKey));
} catch (NoSuchEntityException e) {
log.warn("no sense for key " + senseKey);
}
}

private static class TheErrorHandler implements ErrorHandler {
private static final Logger log = Logger
.getLogger(LoadWordNetTaggedGlossesCommandImpl.class);

protected TheErrorHandler() {
super();
}

public void error(SAXParseException exception) throws SAXException {
log.error(exception, exception);
}

public void fatalError(SAXParseException exception) throws
SAXException {
log.fatal(exception, exception);
}

public void warning(SAXParseException exception) throws SAXException
{
log.warn(exception, exception);
}
}

```

lockacquisitionexceptionadapter.java

```

/*
 * $Id: LockAcquisitionExceptionAdapter.java,v 1.1 2008/12/13 00:41:13
rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.requel.EntityLockException;

/**
 * @author ron
 */
public class LockAcquisitionExceptionAdapter implements
EntityExceptionAdapter {

/*
 * (non-Javadoc)
 *
 * @see
edu.harvard.fas.rregan.requel.EntityExceptionAdapter#convert(java.lang
.Throwable,
 * java.lang.Class, java.lang.Object,
 * edu.harvard.fas.rregan.requel.EntityExceptionActionType)
 */
@Override
public EntityLockException convert(Throwable original, Class<?>
entityType, Object entity,
EntityExceptionActionType actionType) {
return EntityLockException.deadlock(entityType, entity, actionType);
}
}

```

logincommand.java

```

/*
 * $Id: LoginCommand.java,v 1.3 2008/12/13 00:40:44 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.user.User;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**

```

```

 * @author ron
 */
public interface LoginCommand extends Command {

    /**
     * @param username -
     *          the username of the user account to login
     */
    public void setUsername(String username);

    /**
     * @param password -
     *          the password of the user account to login
     */
    public void setPassword(String password);

    /**
     * @return the logged in user
     * @throws NoSuchUserException -
     *          if the username does not refer to a known user or the
     *          password doesn't match the users password.
     */
    public User getUser() throws NoSuchUserException;
}

```

logincommandimpl.java

```

/*
 * $Id: LoginCommandImpl.java,v 1.4 2008/08/25 02:20:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.command.LoginCommand;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**
 * @author ron
 */

```

```

@Component("loginCommand")
@Scope("prototype")
public class LoginCommandImpl extends AbstractUserCommand implements LoginCommand {

    private String username;
    private String password;
    private User user;

    /**
     * @param userRepository
     */
    @Autowired
    public LoginCommandImpl(UserRepository userRepository) {
        super(userRepository);
    }

    public void execute() {
        User user = getUserRepository().findUserByUsername(getUsername());
        if (user.isPassword(password)) {
            setUser(user);
        } else {
            throw NoSuchUserException.wrongPasswordForUser(user);
        }
    }

    public User getUser() throws NoSuchUserException {
        return user;
    }

    protected void setUser(User user) {
        this.user = user;
    }

    @Override
    public void setUsername(String username) {
        this.username = username;
    }

    protected String getUsername() {
        return username;
    }

    protected String getPassword() {
        return password;
    }
}

```

```

@Override
public void setPassword(String password) {
    this.password = password;
}
}

```

logincontroller.java

```

/*
 * $Id: LoginController.java,v 1.7 2008/12/13 00:42:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.login;

import nextapp.echo2.app.event.ActionEvent;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import edu.harvard.fas.rregan.requel.ui.RequelMainScreen;
import edu.harvard.fas.rregan.requel.user.command.LoginCommand;
import edu.harvard.fas.rregan.requel.user.command.UserCommandFactory;
import edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
import edu.harvard.fas.rregan.uiframework.login.LoginEvent;
import edu.harvard.fas.rregan.uiframework.login.LoginFailedEvent;
import edu.harvard.fas.rregan.uiframework.login.LoginOkEvent;

/**
 * Command for processing LoginEvents.
 *
 * @author ron
 */
@Controller("loginController")
@Scope("prototype")
public class LoginController extends AbstractRequelCommandController {
    static final long serialVersionUID = 0;

    /**
     * @param commandFactory
     * @param commandHandler
     */

```

```

    @Autowired
    public LoginController(UserCommandFactory commandFactory,
                          CommandHandler commandHandler) {
        super(commandFactory, commandHandler);
        addEventTypeToListenFor(LoginEvent.class);
    }

    public void actionPerformed(ActionEvent event) {
        if (event instanceof LoginEvent) {
            LoginEvent loginEvent = (LoginEvent) event;
            try {
                UserCommandFactory factory = getCommandFactory();
                LoginCommand command = factory.newLoginCommand();
                command.setUsername(loginEvent.getUsername());
                command.setPassword(loginEvent.getPassword());
                command = getCommandHandler().execute(command);
                fireEvent(new LoginOkEvent(this, command.getUser(),
                    RequelMainScreen.screenName));
            } catch (NoSuchUserException e) {
                fireEvent(new LoginFailedEvent(this, "The username and password
                    are not valid."));
            } catch (Exception e) {
                fireEvent(new LoginFailedEvent(this, "The system encountered an
                    internal error. contact the system administrator."));
            }
        }
    }
}

```

loginevent.java

```

/*
 * $Id: LoginEvent.java,v 1.2 2008/02/27 02:35:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event indicates a user requesting to log into the system with
 * the supplied username and password. This event typically flows from
 * a UI component (the login screen) to an app specific controller for
 * processing login requests.
 */

```

```

 * @author ron
 */
public class LoginEvent extends NavigationEvent {
    static final long serialVersionUID = 0L;

    private String username;
    private String password;

    /**
     * @param source
     * @param username
     * @param password
     */
    public LoginEvent(Object source, String username, String password) {
        super(source, LoginEvent.class.getName());
        setUsername(username);
        setPassword(password);
    }

    /**
     * @return
     */
    public String getUsername() {
        return username;
    }

    protected void setUsername(String username) {
        this.username = username;
    }

    /**
     * @return
     */
    public String getPassword() {
        return password;
    }

    protected void setPassword(String password) {
        this.password = password;
    }
}

loginfailedevent.java
/*

```

```

 * $Id: LoginFailedEvent.java,v 1.2 2008/02/27 02:35:46 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event is a response from the app specific login controller for
 * a LoginEvent when the supplied user credentials are not valid or
 * the
 * controller failed for whatever reason. It contains a message on why
 * the login didn't succeed.
 *
 * @author ron
 */
public class LoginFailedEvent extends NavigationEvent {
    static final long serialVersionUID = 0L;

    private String message;

    public LoginFailedEvent(Object source, String message) {
        super(source, LoginFailedEvent.class.getName());
        setMessage(message);
    }

    public String getMessage() {
        return message;
    }

    protected void setMessage(String message) {
        this.message = message;
    }
}

loginokcontroller.java
/*
 * $Id: LoginOkController.java,v 1.4 2008/08/24 04:13:43 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

```

```

import nextapp.echo2.app.event.ActionEvent;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.uiframework.controller.AbstractAppAwareController;

/**
 * This controller listens for LoginOkEvents containing the user that
logged in
 * and the name of the screen to display. When it receives the event
it sets the
 * user on the UIFrameworkApp and sets the current screen to the one
supplied.
 *
 * @author ron
 */
@Controller("loginOkController")
@Scope("prototype")
public class LoginOkController extends AbstractAppAwareController {
    static final long serialVersionUID = 0;

    /**
     */
    public LoginOkController() {
        super();
        addEventTypeToListenFor(LoginOkEvent.class);
    }

    /**
     * @see
nextapp.echo2.app.event.ActionListener#actionPerformed(nextapp.echo2.a
pp.event.ActionEvent)
     */
    public void actionPerformed(ActionEvent e) {
        if (e instanceof LoginOkEvent) {
            LoginOkEvent event = (LoginOkEvent) e;
            getApp().setUser(event.getUser());
            getApp().setCurrentScreen(event.getScreenToDisplay());

            // tell the app to initialize itself
            fireEvent(new InitAppEvent(this, event.getUser()));
        }
    }
}

```

```

}
```

loginokevent.java

```

/*
 * $Id: LoginOkEvent.java,v 1.3 2008/02/27 02:35:48 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

import edu.harvard.fas.rregan.uiframework.navigation.event.SetScreenEvent;

/**
 * This event is a response from the app specific login controller for
 * a LoginEvent that succeeded. It contains the user and next screen
to
 * display.
 *
 * @author ron
 */
public class LoginOkEvent extends SetScreenEvent {
    static final long serialVersionUID = 0L;

    private Object user;

    /**
     * @param source
     * @param user
     * @param screenNameToDisplay
     */
    public LoginOkEvent(Object source, Object user, String
screenNameToDisplay) {
        super(source, LoginOkEvent.class.getName(), screenNameToDisplay);
        setUser(user);
    }

    /**
     * @return - the user that logged in as an Object
     */
    public Object getUser() {
        return user;
    }

    protected void setUser(Object user) {
        this.user = user;
    }
}

```

```
}
```

loginscreen.java

```
/*
 * $Id: LoginScreen.java,v 1.12 2008/09/12 00:15:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.ContentPane;
import nextapp.echo2.app.Grid;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.PasswordField;
import nextapp.echo2.app.SplitPane;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;

import
edu.harvard.fas.rregan.uiframework.AbstractAppAwareActionListener;
import edu.harvard.fas.rregan.uiframework.screen.AbstractScreen;

/**
 * The login screen collects user credentials and when the user clicks
the
 * submit button it fires a LoginEvent through the EventDispatcher to
the app
 * specific login controller. The login screen has an embedded login
failed
 * controller (loginFailedListener) that listens for login failed
events and
 * displays the supplied message to the user. The look-and-feel of the
screen
 * can be customized through the Echo2 stylesheet and the field and
button
 * labels can be customized through resource bundle property files.
*/
```

```
@org.springframework.stereotype.Component(LoginScreen.screenName)
@Scope("prototype")
public class LoginScreen extends AbstractScreen {
    private static final Logger log =
Logger.getLogger(LoginScreen.class);
    static final long serialVersionUID = 0;

    /**
     * The name of this screen, used to open this screen via a
SetScreenEvent or
     * sub-type.
     */
    public static final String screenName = "loginScreen";

    /**
     * The name of the style to use for configuring the split pane of the
login
     * screen in the Echo2 stylesheet. <br>
     * Example style: <br>
     *
     * <pre>
     * <style name="LoginScreen.SplitPane">
     *   type="nextapp.echo2.app.SplitPane";
     *   <properties>
     *     <property name="layoutData">
     *       <layout-data
type="nextapp.echo2.app.layout.SplitContainerLayoutData">
     *         <properties>
     *           <property name="overflow"/
     *
     * value="nextapp.echo2.app.layout.SplitContainerLayoutData.OVERFLOW_HIDD
EN" />;
     *   </properties>
     *   <layout-data>
     *     <property>
     *       <property name="orientation"/
     *
     * value="nextapp.echo2.app.SplitContainer.ORIENTATION_HORIZONTAL.LEADING
_TRAILING" />;
     *   <property name="separatorPosition" value="
235px" />;
     *   </properties>
     *   </style>
     * </pre>
     */
    public static final String STYLE_NAME_SPLIT_SCREEN =
"LoginScreen.SplitPane";
```

```

/**
 * The name of the style to use for configuring the panel in the left
half
 * of the split pane of the login screen in the Echo2 stylesheet.
<br>
 * Example style: <br>
*
* <pre>
*   &lt;style name="LoginScreen.LeftPane";
*     type="nextapp.echo2.app.ContentPane";&gt;
*     &lt;properties&gt;
*       &lt;property name="backgroundImage";&gt;
*         &lt;fill-image repeat="vertical";&gt;
*           &lt;image
type="nextapp.echo2.app.ResourceImageReference";&gt;
*             &lt;resource-image-reference
*
resource="/resources/images/backgrounds/LoginPaneBackground2.png&
quot; /&gt;
*           &lt;/image&gt;
*           &lt;/fill-image&gt;
*         &lt;/property&gt;
*       &lt;/properties&gt;
*     &lt;/style&gt;
*   </pre>
*/
public static final String STYLE_NAME_LEFT_PANE =
"LoginScreen.LeftPane";

/**
 * The name of the style to use for configuring the column that lays
out the
 * left panel logo of the screen in the Echo2 stylesheet. <br>
 * Example style: <br>
*
* <pre>
*   &lt;style name="LoginScreen.LeftPane.Column";
*     type="nextapp.echo2.app.Column";&gt;
*     &lt;properties&gt;
*       &lt;property name="cellSpacing"; value="
10px";
*         &lt;property name="insets"; value="20px 50px 20px
50px";&gt;
*       &lt;/properties&gt;
*     &lt;/style&gt;
*   </pre>

```

```

/*
public static final String STYLE_NAME_LEFT_PANE =
"LoginScreen.LeftPane.Column";

/**
 * The name of the style to use for configuring the logo in the left
pane of
 * the login screen in the Echo2 stylesheet. <br>
 * Example style: <br>
*
* <pre>
*   &lt;style name="LoginScreen.Logo";
*     type="nextapp.echo2.app.Label";&gt;
*     &lt;properties&gt;
*       &lt;property name="icon";
*         type="nextapp.echo2.app.ResourceImageReference";
*
value="/resources/images/logo_fembot_torso_blue_cutout.png";
/&gt;
*       &lt;/properties&gt;
*     &lt;/style&gt;
*   </pre>
*/
public static final String STYLE_NAME_LEFT_LOGO =
"LoginScreen.Left.Logo";

/**
 * The name of the style to use for configuring the panel in the left
half
 * of the split pane of the login screen in the Echo2 stylesheet.
<br>
 * Example style: <br>
*
* <pre>
*   &lt;style name="LoginScreen.RightPane";
*     type="nextapp.echo2.app.ContentPane";&gt;
*     &lt;properties&gt;
*       &lt;property name="background";
value="#aaaaff"; /&gt;
*       &lt;/properties&gt;
*     &lt;/style&gt;
*   </pre>
*/
public static final String STYLE_NAME_RIGHT_PANE =
"LoginScreen.RightPane";

/**/

```

```

 * The name of the style to use for configuring the column that lays
out the
 * logo, title, subtitle, input control grid, and login button in the
right
 * half of the screen in the Echo2 stylesheet. <br>
 * Example style: <br>
 *
 * <pre>
 *   &lt;style name="LoginScreen.RightPane.Column";
 *     type="nextapp.echo2.app.Column";&gt;
 *     &lt;properties&gt;
 *       &lt;property name="cellSpacing" value="
10px"/&gt;
 *       &lt;property name="insets" value="20px 50px 20px
50px"/&gt;
 *     &lt;/properties&gt;
 *   &lt;/style&gt;
 * </pre>
 */
public static final String STYLE_NAME_RIGHT_PANE_COLUMN =
"LoginScreen.RightPane.Column";

/**
 * The name of the style to use for configuring the logo on the login
screen
 * above the title in the Echo2 stylesheet. <br>
 * Example style: <br>
 *
 * <pre>
 *   &lt;style name="LoginScreen.Logo";
 *     type="nextapp.echo2.app.Label";&gt;
 *     &lt;properties&gt;
 *       &lt;property name="icon";
 *         type="nextapp.echo2.app.ResourceImageReference";
 *         value="/resources/images/Logo209x100.png"/&gt;
 *     &lt;/properties&gt;
 *   &lt;/style&gt;
 * </pre>
 */
public static final String STYLE_NAME_RIGHT_LOGO =
"LoginScreen.Right.Logo";

/**
 * The name of the style to use for configuring the title on the
login
 * screen below the logo in the Echo2 stylesheet. <br>
 * Example style: <br>

```

```

 *
 * <pre>
 *   &lt;style name="LoginScreen.Title";
 *     type="nextapp.echo2.app.Label";&gt;
 *     &lt;properties&gt;
 *       &lt;property name="font";
 *         &lt;font bold="true";
 *           size="22pt"; typeface="Arial" /&gt;
 *       &lt;/property&gt;
 *     &lt;/properties&gt;
 *   &lt;/style&gt;
 * </pre>
 */
public static final String STYLE_NAME_TITLE = "LoginScreen.Title";

/**
 * The name of the style to use for configuring the subtitle on the
login
 * screen below the title in the Echo2 stylesheet. <br>
 * Example style: <br>
 *
 * <pre>
 *   &lt;style name="LoginScreen.Subtitle";
 *     type="nextapp.echo2.app.Label";&gt;
 *     &lt;properties&gt;
 *       &lt;property name="font";
 *         &lt;font bold="true";
 *           size="12pt"; typeface="Arial" /&gt;
 *       &lt;/property&gt;
 *     &lt;/properties&gt;
 *   &lt;/style&gt;
 * </pre>
 */
public static final String STYLE_NAME_SUB_TITLE =
"LoginScreen.Subtitle";

/**
 * The name of the style to use for configuring the layout grid on
the
 * login
 * screen containing the input fields and labels in the Echo2
stylesheet.
 * <br>
 * Example style: <br>
 *
 * <pre>
 *   &lt;style name="LoginScreen.LayoutGrid";
 *     type="nextapp.echo2.app.Grid";&gt;

```

```

*   &lt;properties&gt;
*     &lt;property name="insets" value="10px
10px" /&gt;
*       &lt;property name="border"&gt;
*         &lt;border color="#bbbbff";
*           size="2px" style="groove" /&gt;
*       &lt;/property&gt;
*     &lt;/properties&gt;
*   &lt;/style&gt;
* </pre>
*/
public static final String STYLE_NAME_LAYOUT_GRID =
"LoginScreen.LayoutGrid";

/**
 * The name of the style to use for configuring the username and
password
 * input field labels in the Echo2 stylesheet. <br>
 * Example style: <br>
 *
* <pre>
*   &lt;style name="LoginScreen.Label";
*     base-name="Default";
*     type="nextapp.echo2.app.Label";&gt;
*   &lt;properties&gt;
*     &lt;property name="font"&gt;
*       &lt;font bold="false";
*         size="10pt" typeface="Arial" /&gt;
*     &lt;/property&gt;
*   &lt;/properties&gt;
* &lt;/style&gt;
* </pre>
*/
public static final String STYLE_NAME_LABEL = "LoginScreen.Label";

/**
 * The name of the style to use for configuring the username input
field in
 * the Echo2 stylesheet. <br>
 * Example style: <br>
 *
* <pre>
*   &lt;style name="LoginScreen.Username.InputField";
*     base-name="Default";
*     type="nextapp.echo2.app.text.TextComponent";&gt;
*   &lt;properties&gt;
*
*       &lt;property name="width" value="300px";
*       &lt;property name="font"&gt;
*         &lt;font bold="false";
*           size="10pt" typeface="Arial" /&gt;
*       &lt;/property&gt;
*     &lt;/properties&gt;
*   &lt;/style&gt;
* </pre>
*/
public static final String STYLE_NAME_USERNAME_INPUT =
"LoginScreen.Username.InputField";

/**
 * The name of the style to use for configuring the username input
field in
 * the Echo2 stylesheet. <br>
 * See the username input for an example. <br>
 */
public static final String STYLE_NAME_PASSWORD_INPUT =
"LoginScreen.Password.InputField";

/**
 * The name of the style to use for configuring the login button in
the
 * Echo2 stylesheet. <br>
 * See the Default style for AbstractButton as an example. <br>
 */
public static final String STYLE_NAME_LOGIN_BUTTON =
"LoginScreen.LoginButton";

/**
 * The name of the style to use for configuring the login button in
the
 * Echo2 stylesheet. <br>
 * See the ValidationLabel style for an example. <br>
 */
public static final String STYLE_NAME_VALIDATION = "ValidationLabel";

/**
 * The name of the property to use for the title from the specified
resource
 * (property) file.
 */
public static final String PROP_NAME_TITLE = "Screen.Title";

/**

```

```

 * The name of the property to use for the subtitle from the
specified
 * resource (property) file.
 */
public static final String PROP_NAME_SUB_TITLE = "Screen.Subtitle";

/**
 * The name of the property to use for the username field label from
the
 * specified resource (property) file.
 */
public static final String PROP_NAME_USERNAME_LABEL =
"Username.Label";

/**
 * The name of the property to use for the password field label from
the
 * specified resource (property) file.
 */
public static final String PROP_NAME_PASSWORD_LABEL =
>Password.Label";

/**
 * The name of the property to use for the login button label from
the
 * specified resource (property) file.
 */
public static final String PROP_NAME_LOGIN_BUTTON_LABEL =
>LoginButton.Label";

// the ActionEvent command string for the local event fired from the
// password field when return is hit, or from the login button. This
// event is handle locally by the SubmitLoginListener
private static final String ACTION_SUBMIT_LOGIN =
LoginScreen.class.getName() + ".submitLogin";

private TextField usernameField;
private PasswordField passwordField;
private Label messages;

private final SubmitLoginListener submitLoginListener;
private final LoginFailedListener loginFailedListener;

/**
 * Create a LoginScreen using LoginScreen.properties in the same
classpath
 * directory for the resource bundle.

```

```

 */
public LoginScreen() {
    this(LoginScreen.class.getName());
}

/**
 * Create a LoginScreen using the supplied name for the resource
bundle.
 *
 * @param resourceBundleName -
 *          The name of the resource bundle including the path.
 */
public LoginScreen(String resourceBundleName) {
    super(resourceBundleName);
    submitLoginListener = new SubmitLoginListener(this);
    loginFailedListener = new LoginFailedListener(this);

    log.debug("new login screen created, using resources from: " +
resourceBundleName);
}

@Override
public void setup() {
    super.setup();
    SplitPane splitPanel1 = new SplitPane();
    splitPanel1.setStyleName(STYLE_NAME_SPLIT_SCREEN);
    splitPanel1.add(createLeftPanel());
    splitPanel1.add(createRightPanel());
    add(splitPanel1);
    getEventDispatcher().addEventTypeActionListener(LoginFailedEvent.cla
ss, loginFailedListener);
    resetState();
}

@Override
public void dispose() {
    getEventDispatcher().removeEventTypeActionListener(LoginFailedEvent.
class,
    loginFailedListener);
    super.dispose();
}

private Component createLeftPanel() {
    ContentPane leftPanel = new ContentPane();
    leftPanel.setStyleName(STYLE_NAME_LEFT_PANE);

    Column column = new Column();

```

```

column.setStyleName(STYLE_NAME_LEFT_PANE_COLUMN);
leftPanel.add(column);

Label logo = new Label();
logo.setStyleName(STYLE_NAME_LEFT_LOGO);
column.add(logo);

return leftPanel;
}

private Component createRightPanel() {
ContentPane loginPanel = new ContentPane();
loginPanel.setStyleName(STYLE_NAME_RIGHT_PANE);

Column column = new Column();
column.setStyleName(STYLE_NAME_RIGHT_PANE_COLUMN);
loginPanel.add(column);

Label logo = new Label();
logo.setStyleName(STYLE_NAME_RIGHT_LOGO);
column.add(logo);

Label title = new
Label(getResourceBundleHelpergetLocale()).getString(PROP_NAME_TITLE,
    "Login"));
title.setStyleName(STYLE_NAME_TITLE);
column.add(title);

Label subTitle = new
Label(getResourceBundleHelpergetLocale()).getString(
    PROP_NAME_SUB_TITLE, "Login"));
subTitle.setStyleName(STYLE_NAME_SUB_TITLE);
column.add(subTitle);

Grid grid1 = new Grid(2);
grid1.setStyleName(STYLE_NAME_LAYOUT_GRID);

Label usernameLabel = new
Label(getResourceBundleHelpergetLocale()).getString(
    PROP_NAME_USERNAME_LABEL, "Username:""));
usernameLabel.setStyleName(STYLE_NAME_LABEL);
grid1.add(usernameLabel);

usernameField = new TextField();
usernameField.setStyleName(STYLE_NAME_USERNAME_INPUT);
usernameField.addActionListener(new ActionListener() {
    private static final long serialVersionUID = 0;
}

```

```

        public void actionPerformed(ActionEvent e) {
            getApp().setFocusedComponent(passwordField);
        }
    });
grid1.add(usernameField);

Label passwordLabel = new
Label(getResourceBundleHelpergetLocale()).getString(
    PROP_NAME_PASSWORD_LABEL, "Password:""));
passwordLabel.setStyleName(STYLE_NAME_LABEL);
grid1.add(passwordLabel);

passwordField = new PasswordField();
passwordField.setStyleName(STYLE_NAME_PASSWORD_INPUT);
passwordField.addActionListener(submitLoginListener);
passwordField.setActionCommand(ACTION_SUBMIT_LOGIN);
grid1.add(passwordField);

column.add(grid1);

Button loginButton = new
Button(getResourceBundleHelpergetLocale()).getString(
    PROP_NAME_LOGIN_BUTTON_LABEL, "Login"));
loginButton.setStyleName(STYLE_NAME_LOGIN_BUTTON);
loginButton.setActionCommand(ACTION_SUBMIT_LOGIN);
loginButton.addActionListener(submitLoginListener);
column.add(loginButton);

messages = new Label();
messages.setStyleName(STYLE_NAME_VALIDATION);
column.add(messages);
return loginPanel;
}

/**
 *
 */
public void resetState() {
    usernameField.setText("");
    passwordField.setText("");
    messages.setText("");
    getApp().setFocusedComponent(usernameField);
}

private static class SubmitLoginListener extends
AbstractAppAwareActionListener {

```

```

static final long serialVersionUID = 0;

private final LoginScreen screen;

protected SubmitLoginListener(LoginScreen screen) {
    super(screen.getApp());
    this.screen = screen;
}

public void actionPerformed(ActionEvent event) {
    log.debug("event: " + event);
    if (ACTION_SUBMIT_LOGIN.equals(event.getActionCommand())) {
        String username = screen.usernameField.getText();
        String password = screen.passwordField.getText();
        try {
            screen.getApp().getEventDispatcher().dispatchEvent(
                new LoginEvent(screen, username, password));
            screen.passwordField.setText("");
        } catch (Exception e) {
            log.error("exception executing LoginEvent: " + e, e);
            screen.messages
                .setText("A system problem caused the login to fail. If this
continues please contact the administrator.");
        }
    }
}

private static class LoginFailedListener extends
AbstractAppAwareActionListener {
    static final long serialVersionUID = 0;

    private final LoginScreen screen;

    protected LoginFailedListener(LoginScreen screen) {
        super(screen.getApp());
        this.screen = screen;
    }

    public void actionPerformed(ActionEvent e) {
        log.debug("event: " + e);
        if (e instanceof LoginFailedEvent) {
            screen.messages.setText(((LoginFailedEvent) e).getMessage());
        }
    }
}

```

```

}
```

logoutcontroller.java

```

/*
 * $Id: LogoutController.java,v 1.4 2008/05/01 08:10:18 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.login;

import nextapp.echo2.app.event.ActionEvent;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.requel.ui.AbstractRequelController;
import edu.harvard.fas.rregan.uiframework.login.LogoutEvent;
import edu.harvard.fas.rregan.uiframework.login.LogoutOkEvent;

/**
 * Controller for processing LogoutEvents.
 *
 * @author ron
 */
@Controller
@Scope("prototype")
public class LogoutController extends AbstractRequelController {
    static final long serialVersionUID = 0;

    /**
     */
    public LogoutController() {
        super();
        addEventTypeToListenFor(LogoutEvent.class);
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        if (event instanceof LogoutEvent) {
            // TODO: any work needed when a user logs out
            fireEvent(new LogoutOkEvent(this));
        }
    }
}

```

logoutevent.java

```
/*
 * $Id: LogoutEvent.java,v 1.2 2008/02/27 02:35:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.login;

import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * The logout event gets fired when a user indicates they want to
logout
 * of the application. The event should be fired by a NavigatorButton
 * visible to the user after they login to the system.<br>
 * <code>
 * NavigatorButton logoutButton =
 *   new NavigatorButton("Logout", getEventDispatcher(), new
LogoutEvent(mainScreen));
 * </code>
 * <br>
 * The app should have a LogoutController that does any necessary
cleanup
 * and fires a LogoutOkEvent for the LogoutOkController to update the
UI.
 * @author ron
 */
public class LogoutEvent extends NavigationEvent {
    static final long serialVersionUID = 0L;

    public static final String CMD_LOGOUT = LogoutEvent.class.getName();

    /**
     * @param source
     * @param command
     */
    public LogoutEvent(Object source) {
        super(source, CMD_LOGOUT);
    }
}
```

logoutokcontroller.java

```
/*
 * $Id: LogoutOkController.java,v 1.4 2008/08/20 08:29:01 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.login;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.webcontainer.command.BrowserRedirectCommand;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import
edu.harvard.fas.rregan.uiframework.controller.AbstractAppAwareController;

/**
 * This controller listens for LogoutOkEvents and removes the
 * previously logged in user from the UIFrameworkApp state and
 * sets the current screen to the supplied screen.
 *
 * @author ron
 */
@Controller
@Scope("prototype")
public class LogoutOkController extends AbstractAppAwareController {
    static final long serialVersionUID = 0;

    /**
     */
    public LogoutOkController() {
        this(null);
    }

    /**
     * @param app
     */
    public LogoutOkController(UIFrameworkApp app) {
        super(app);
        addEventTypeToListenFor(LogoutOkEvent.class);
    }
}
```

```

/**
 * @see
nextapp.echo2.app.event.ActionListener#actionPerformed(nextapp.echo2.a
pp.event.ActionEvent)
 */
public void actionPerformed(ActionEvent e) {
    if (e instanceof LogoutOkEvent) {
        getApp().enqueueCommand(new BrowserRedirectCommand("logout"));
    }
}
}

```

logoutokevent.java

```

/*
 * $Id: LogoutOkEvent.java,v 1.3 2008/02/27 02:35:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.login;

import
edu.harvard.fas.rregan.uiframework.navigation.event.SetScreenEvent;

/**
 * The LogoutOkEvent should be fired by the app specific
LogoutController
 * after receiving a LogoutEvent if the state of the app is ok for
logging
 * out the user. The default constructor is hardwired to set the
LoginScreen
 * as the current screen.<br>
 * The LogoutOkEvent is handled by the UIFramework LogoutOkController.
 * @author ron
 */
public class LogoutOkEvent extends SetScreenEvent {
    static final long serialVersionUID = 0L;

    /**
     * Create a LogoutOkEvent that resets the screen to the UIFramework
     * standard LoginScreen.
     * @param source
     */
    public LogoutOkEvent(Object source) {
        this(source, LoginScreen.screenName);
    }
}

```

```

/**
 * Create a LogoutOkEvent that resets the screen to the supplied
 * screen.
 *
 * @param source
 * @param screenName
 */
public LogoutOkEvent(Object source, String screenName) {
    super(source, LogoutOkEvent.class.getName(), screenName);
}
}

```

mainscreentabbednavigation.java

```

/*
 * $Id: MainScreenTabbedNavigation.java,v 1.8 2008/03/02 09:10:24
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.panel.PanelManager;
import edu.harvard.fas.rregan.uiframework.panel.TabbedPaneContainer;

/**
 * @author ron
 */
public class MainScreenTabbedNavigation extends TabbedPaneContainer {
    private static final Logger log =
Logger.getLogger(MainScreenTabbedNavigation.class);
    static final long serialVersionUID = 0;

    /**
     * The style name to use in the stylesheet to configure the tabs.
     */
    public static final String STYLE_NAME_NAV_PANEL_CONTAINER =
"RequelMainScreen.MainScreenTabbedNavigation";

    /**
     * @param navPanels
     * @param panelManager
     */
    public MainScreenTabbedNavigation(PanelManager panelManager) {

```

```

super(MainScreenTabbedNavigation.class.getName(), panelManager);
}

@Override
public void setup() {
    super.setup();
    setStyleName(STYLE_NAME_NAV_PANEL_CONTAINER);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}
}

```

messagehandler.java

```

/*
 * $Id: MessageHandler.java,v 1.1 2008/10/30 05:55:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

/**
 * @author ron
 */
public interface MessageHandler {

    /**
     * Set a message to report to the user.
     *
     * @param message -
     *            the message to report.
     */
    public void setGeneralMessage(String message);
}

```

morespecificwordsuggester.java

```

/*
 * $Id: MoreSpecificWordSuggester.java,v 1.1 2009/03/27 07:16:08
 * rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * Find words that are more specific (less vague) than the supplied
 * word. This
 * uses the WordNet Hyponym hierarchy to find a set of words with a
 * higher
 * information context.
 *
 * @author ron
 */
@Component("moreSpecificWordSuggester")
public class MoreSpecificWordSuggester implements
NLPProcessor<Collection<NLPText>> {
    private static final Logger log =
Logger.getLogger(SpellingSuggester.class);

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public MoreSpecificWordSuggester(DictionaryRepository
dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }
}

```

```

@Override
public Collection<NLPText> process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.WORD) && text.hasText()
        && !text.is(PartOfSpeech.PUNCTUATION)) {
        List<NLPText> results = new ArrayList<NLPText>();
        for (Sense sense : dictionaryRepository.findMoreSpecificWords(text
            .getDictionaryWordSense(), 5)) {
            NLPText word = new NLPTextImpl(sense.getWord().getLemma(),
                GrammaticalStructureLevel.WORD);
            word.setDictionaryWord(sense.getWord());
            word.setDictionaryWordSense(sense);
            results.add(word);
        }
        return results;
    }
    return Collections.EMPTY_LIST;
}

```

morphdef.java

```

/*
 * $Id: Morphdef.java,v 1.2 2009/01/03 10:24:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

/***
 * 
 * 
 * @author ron
 */
@Entity
@Table(name = "morphdef", uniqueConstraints =
@UniqueConstraint(columnNames = "lemma"))
public class Morphdef implements Serializable {
    static final long serialVersionUID = 0;

```

```

private int morphid;
private String lemma;

public Morphdef() {
}

public Morphdef(int morphid, String lemma) {
    this.morphid = morphid;
    this.lemma = lemma;
}

@Id
@Column(name = "morphid", unique = true, nullable = false)
public int getMorphid() {
    return this.morphid;
}

public void setMorphid(int morphid) {
    this.morphid = morphid;
}

@Column(name = "lemma", unique = true)
public String getLemma() {
    return this.lemma;
}

public void setLemma(String lemma) {
    this.lemma = lemma;
}

```

morphref.java

```

/*
 * $Id: Morphref.java,v 1.2 2009/01/03 10:24:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.AttributeOverride;
import javax.persistence.AttributeOverrides;
import javax.persistence.Column;
import javax.persistence.EmbeddedId;
```

```

import javax.persistence.Entity;
import javax.persistence.Table;

/**
 * @author ron
 */
@Entity
@Table(name = "morphref")
public class Morphref implements Serializable {
    static final long serialVersionUID = 0;

    private MorphrefId id;

    public Morphref() {
    }

    public Morphref(MorphrefId id) {
        this.id = id;
    }

    @EmbeddedId
    @AttributeOverrides( {
        @AttributeOverride(name = "morphid", column = @Column(name =
        "morphid", nullable = false, precision = 6, scale = 0)),
        @AttributeOverride(name = "pos", column = @Column(name = "pos",
        nullable = false, length = 2)),
        @AttributeOverride(name = "wordid", column = @Column(name =
        "wordid", nullable = false, precision = 6, scale = 0)) })
    public MorphrefId getId() {
        return this.id;
    }

    public void setId(MorphrefId id) {
        this.id = id;
    }
}

/*
 * $Id: MorphrefId.java,v 1.2 2009/01/03 10:24:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

```

morphrefid.java

```

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Embeddable;

/**
 * @author ron
 */
@Embeddable
public class MorphrefId implements Serializable {
    static final long serialVersionUID = 0;

    private int morphid;
    private String pos;
    private int wordid;

    public MorphrefId() {
    }

    public MorphrefId(int morphid, String pos, int wordid) {
        this.morphid = morphid;
        this.pos = pos;
        this.wordid = wordid;
    }

    @Column(name = "morphid", nullable = false)
    public int getMorphid() {
        return this.morphid;
    }

    public void setMorphid(int morphid) {
        this.morphid = morphid;
    }

    @Column(name = "pos", nullable = false)
    public String getPos() {
        return this.pos;
    }

    public void setPos(String pos) {
        this.pos = pos;
    }

    @Column(name = "wordid", nullable = false)
    public int getWordid() {
        return this.wordid;
    }
}

```

```

public void setWordid(int wordid) {
    this.wordid = wordid;
}

@Override
public boolean equals(Object other) {
    if ((this == other)) {
        return true;
    }
    if ((other == null)) {
        return false;
    }
    if (!(other instanceof MorphrefId)) {
        return false;
    }
    MorphrefId castOther = (MorphrefId) other;

    return (this.getMorphid() == castOther.getMorphid())
        && ((this.getPos() == castOther.getPos()) || ((this.getPos() != null)
        && (castOther.getPos() != null) &&
        this.getPos().equals(castOther.getPos())))
        && (this.getWordid() == castOther.getWordid());
}

@Override
public int hashCode() {
    int result = 17;

    result = 37 * result + this.getMorphid();
    result = 37 * result + (getPos() == null ? 0 :
    this.getPos().hashCode());
    result = 37 * result + this.getWordid();
    return result;
}

```

namedentity.java

```

/*
 * $Id: NamedEntity.java,v 1.2 2008/04/22 23:21:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel;

```

```

/**
 * @author ron
 */
public interface NamedEntity {

    /**
     * @return
     */
    public String getName();
}

```

navigationevent.java

```

/*
 * $Id: NavigationEvent.java,v 1.4 2008/09/12 01:00:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import nextapp.echo2.app.event.ActionEvent;

/**
 * @author ron
 */
public class NavigationEvent extends ActionEvent {
    static final long serialVersionUID = 0;

    private final Object destinationObject;

    /**
     * @param source
     * @param command
     */
    public NavigationEvent(Object source, String command) {
        this(source, command, null);
    }

    /**
     * @param source
     * @param command
     * @param destinationObject - the intended recipient of the event.
     */
    public NavigationEvent(Object source, String command, Object
    destinationObject) {
        super(source, command);
    }
}

```

```

        this.destinationObject = destinationObject;
    }

    /**
     * The intended recipient of the event. this is for events such as
     * an add entity event or remove entity event where a specific
     controller
     * should receive the event to do the adding or removing.
     *
     * @return
     */
    public Object getDestinationObject() {
        return destinationObject;
    }

    @Override
    public String toString() {
        return this.getClass().getSimpleName() + "[source = " + getSource()
+ ", command = " + getActionCommand() + ", destinationObject = " +
destinationObject + "]";
    }
}

```

navigationinitializationcontroller.java

```

/*
 * $Id: NavigationInitializationController.java,v 1.7 2008/12/16
23:03:17 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui;

import nextapp.echo2.app.event.ActionEvent;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.ui.NLPPanelNames;
import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRole;
import edu.harvard.fas.rregan.uiframework.login.InitAppEvent;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
@Controller
@Scope("prototype")
public class NavigationInitializationController extends
AbstractRequelController {
    static final long serialVersionUID = 0;

    private final UserRepository userRepository;

    /**
     * @param userRepository
     */
    @Autowired
    public NavigationInitializationController(UserRepository
userRepository) {
        super();
        this.userRepository = userRepository;
        addEventTypeToListenFor(InitAppEvent.class);
    }

    /**
     * @see
     * nextapp.echo2.app.event.ActionListener#actionPerformed(nextapp.echo2.a
pp.event.ActionEvent)
     */
    public void actionPerformed(ActionEvent e) {
        if (e instanceof InitAppEvent) {
            InitAppEvent event = (InitAppEvent) e;
            User user = (User) event.getUser();
            for (UserRole userRole : user.getUserRoles()) {
                if (userRole instanceof SystemAdminUserRole) {
                    fireEvent(new OpenPanelEvent(this, PanelActionType.Navigator,
getUserRepository()
                    .findUsers(), userRole.getClass(), null,
WorkflowDisposition.NewFlow));
                } else {
                    fireEvent(new OpenPanelEvent(this, PanelActionType.Navigator,
user, userRole
                    .getClass(), null, WorkflowDisposition.NewFlow));
                }
            }
        }
    }
}

```

```

        }
    }
    fireEvent(new OpenPanelEvent(this, PanelActionType.Navigator, null,
null, NLPPanelNames.NLP_NAVIGATOR_PANEL_NAME,
WorkflowDisposition.NewFlow));
}
}

protected UserRepository getUserRepository() {
    return userRepository;
}
}

```

navigatorbutton.java

```

/*
 * $Id: NavigatorButton.java,v 1.2 2008/02/18 10:08:54 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.ImageReference;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * A NavigatorButton is a convenience component for firing a predefined
 * ActionEvent when clicked. This is mainly for firing navigation
events from a
 * navigator to start a workflow or from a panel to open up a helper
panel.
 *
 * @author ron
 */
public class NavigatorButton extends Button implements ActionListener
{
    private static final Logger log =
Logger.getLogger(NavigatorButton.class);
    static final long serialVersionUID = 0;

```

```

private final EventDispatcher eventDispatcher;
private ActionEvent eventToFire;

/**
 * Create a button with a text label, but without setting the event
to fire.
 * If the event to fire is not set, when the button is clicked
nothing will
 * happen.<br/>The event can be set through the setEventToFire()
method.
 *
 * @param label -
 *          A string to use as the label of the button.
 * @param eventDispatcher -
 *          the dispatcher to fire the event through.
 */
public NavigatorButton(String label, EventDispatcher eventDispatcher)
{
    this(label, eventDispatcher, null);
}

/**
 * Create a button with a text label, and the event to fire. The
event can
 * be changed through the setEventToFire() method.
 *
 * @param label -
 *          A string to use as the label of the button.
 * @param eventDispatcher -
 *          the dispatcher to fire the event through.
 * @param eventToFire -
 *          the event to fire when the button is clicked
 */
public NavigatorButton(String label, EventDispatcher eventDispatcher,
(ActionEvent eventToFire) {
    super(label);
    this.eventDispatcher = eventDispatcher;
    this.eventToFire = eventToFire;
    addActionListener(this);
}

/**
 * Create a button with an image label, but without setting the event
to
 * fire. If the event to fire is not set, when the button is clicked
nothing
 * will happen.<br/>The event can be set through the setEventToFire()

```

```

* method.
*
* @param image -
*           An image reference to use as the label of the button.
* @param eventDispatcher -
*           the dispatcher to fire the event through.
*/
public NavigatorButton(ImageReference image, EventDispatcher
eventDispatcher) {
    this(image, eventDispatcher, null);
}

/**
 * Create a button with an image label, and the event to fire. The
event can
 * be changed through the setEventToFire() method.
*
* @param image -
*           An image reference to use as the label of the button.
* @param eventDispatcher -
*           the dispatcher to fire the event through.
* @param eventToFire -
*           the event to fire when the button is clicked
*/
public NavigatorButton(ImageReference image, EventDispatcher
eventDispatcher,
    ActionEvent eventToFire) {
super(image);
this.eventDispatcher = eventDispatcher;
this.eventToFire = eventToFire;
addActionListener(this);
}

/**
 * Set the event to fire when the button is clicked.
*
* @param eventToFire
*/
public void setEventToFire(ActionEvent eventToFire) {
    this.eventToFire = eventToFire;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (eventToFire == null) {
        log.warn(this + " does not have an eventToFire set.");
    } else {

```

```

        eventDispatcher.dispatchEvent(eventToFire);
    }
}
}

```

navigatortable.java

```

/*
 * $Id: NavigatorTable.java,v 1.11 2008/10/11 08:22:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.list.ListSelectionModel;
import nextapp.echo2.app.table.DefaultTableColumnModel;
import nextapp.echo2.app.table.TableCellRenderer;
import nextapp.echo2.app.table.TableColumn;
import nextapp.echo2.app.table.TableColumnModel;

import org.apache.log4j.Logger;

import echopointng.table.DefaultPageableSortableTableModel;
import echopointng.table.PageableSortableTable;
import echopointng.table.SortableTableColumn;
import echopointng.table.SortableTableHeaderRenderer;
import echopointng.table.SortableTableSelectionModel;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Navigator
TableManipulator;

/**
 * @author ron
 */
public class NavigatorTable extends Column {

```

```

private static final Logger log =
Logger.getLogger(NavigatorTable.class);
static final long serialVersionUID = 0L;
static {
    ComponentManipulators.setManipulator(NavigatorTable.class, new
NavigatorTableManipulator());
}

/**
 * The style name in the stylesheet for applying a style to the
paging
 * controls for the table.
 */
public static final String STYLE_NAME_NAVIGATOR_TABLE_PAGING_CONTROLS
= "NavigatorTable.PagingControls";

/**
 * The style name in the stylesheet for applying a style to the first
page
 * button for the table.
 */
public static final String
STYLE_NAME_NAVIGATOR_TABLE_FIRST_PAGE_BUTTON =
"NavigatorTable.firstButton";

/**
 * The style name in the stylesheet for applying a style to the previous
 * page button for the table.
 */
public static final String
STYLE_NAME_NAVIGATOR_TABLE_PREV_PAGE_BUTTON =
"NavigatorTable.prevButton";

/**
 * The style name in the stylesheet for applying a style to the next
page
 * button for the table.
 */
public static final String
STYLE_NAME_NAVIGATOR_TABLE_NEXT_PAGE_BUTTON =
"NavigatorTable.nextButton";

/**
 * The style name in the stylesheet for applying a style to the last
page
 * button for the table.

```

```

*/
public static final String
STYLE_NAME_NAVIGATOR_TABLE_LAST_PAGE_BUTTON =
"NavigatorTable.lastButton";

private NavigatorTableConfig tableConfig;

private final PageableSortableTable table = new
PageableSortableTable();
private final Button firstButton = new Button();
private final Button prevButton = new Button();
private final Button nextButton = new Button();
private final Button lastButton = new Button();
private final Label currentPageLabel = new Label("");
private int currentPage = 0;

/**
 * @param tableConfig -
 *          the configuration of the columns to display in the
table
 */
public NavigatorTable(NavigatorTableConfig tableConfig) {
    super();
    setTableConfig(tableConfig);

    TableCellRenderer defaultHeaderRenderer = new
SortableTableHeaderRenderer();
    TableCellRenderer defaultRenderer = new
OddEvenTableCellRenderer("Table.OddCell",
    "Table.EvenCell");
    TableColumnModel columnModel = new Default TableColumnModel();
    int columnIndex = 0;
    for (NavigatorTableColumnConfig columnConfig :
tableConfig.getColumnConfigs()) {
        log.debug("column [" + columnIndex + "] " +
columnConfig.getTitle());
        TableColumn column = new Sortable TableColumn(columnIndex);
        column.setHeaderValue(columnConfig.getTitle());
        column.setModelIndex(columnIndex);
        column.setHeaderRenderer(defaultHeaderRenderer);
        column.setCellRenderer(defaultRenderer);
        columnModel.addColumn(column);
        columnIndex++;
    }

    if (tableConfig.getRowLevelSelection()) {
        table.setSelectionEnabled(true);
    }
}

```

```

} else {
    table.setSelectionEnabled(false);
}
table.setWidth(new Extent(100, Extent.PERCENT));
table.setAutoCreateColumnsFromModel(false);
table.setColumnModel(columnModel);
table.setStyleName(Panel.STYLE_NAME_DEFAULT);
table.setDefaultHeaderRenderer(defaultHeaderRenderer);
table.setDefaultRenderer(Object.class, defaultRenderer);
add(table);
initPagingButtons();
Row buttons = new Row();
buttons.setStyleName(STYLE_NAME_NAVIGATOR_TABLE_PAGING_CONTROLS);

firstButton.setStyleName(STYLE_NAME_NAVIGATOR_TABLE_FIRST_PAGE_BUTTON);
prevButton.setStyleName(STYLE_NAME_NAVIGATOR_TABLE_PREV_PAGE_BUTTON);
currentLabel.setStyleName(Panel.STYLE_NAME_DEFAULT);
nextButton.setStyleName(STYLE_NAME_NAVIGATOR_TABLE_NEXT_PAGE_BUTTON);
lastButton.setStyleName(STYLE_NAME_NAVIGATOR_TABLE_LAST_PAGE_BUTTON);

buttons.add(firstButton);
buttons.add(prevButton);
buttons.add(currentLabel);
buttons.add(nextButton);
buttons.add(lastButton);
add(buttons);
setInsets(new Insets(10));
}

/**
 * @param tableModel
 */
public void setModel(NavigatorTableModel tableModel) {
    if (tableModel != null) {
        tableModel.setTableConfig(getTableConfig());
        DefaultPageableSortableTableModel wrapperModel = new
DefaultPageableSortableTableModel(
            tableModel);
        wrapperModel.setRowsPerPage(10);
        table.setModel(wrapperModel);
        updatePagingButtons();
        if (getTableConfig().getRowLevelSelection()) {
            SortableTableSelectionModel selectionModel = new
SortableTableSelectionModel(
                wrapperModel);
            selectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            table.setSelectionModel(selectionModel);
        }
        // fix the paging
        if ((DefaultPageableSortableTableModel)
table.getModel()).getTotalPages() <= currentPage) {
            currentPage = ((DefaultPageableSortableTableModel)
table.getModel())
                .getTotalPages() - 1;
        }
        ((DefaultPageableSortableTableModel)
table.getModel()).setCurrentPage(currentPage);
    }
}

public NavigatorTableModel getModel() {
    return (NavigatorTableModel) ((DefaultPageableSortableTableModel)
table.getModel())
    .getUnderlyingTableModel();
}

/**
 * @return
 */
public NavigatorTableConfig getTableConfig() {
    return tableConfig;
}

protected void setTableConfig(NavigatorTableConfig tableConfig) {
    this.tableConfig = tableConfig;
}

public void addSelectionListener(ActionListener listener) {
    if (tableConfig.getRowLevelSelection()) {
        table.addActionListener(listener);
    } else {
        throw new RuntimeException(
            "Can't add a listener to a table that does not have selection
enabled.");
    }
}

public void removeSelectionListener(ActionListener listener) {
    table.removeActionListener(listener);
}

```

```

public Object getSelectedObject() {
    Object selectedObject = null;
    int selectedIndex = table.getSelectionModel().getMinSelectedIndex();
    if (selectedIndex > -1) {
        selectedObject = getModel().getBackingObject(selectedIndex);
    }
    return selectedObject;
}

protected void updatePagingButtons() {
    // update the label
    currentLabel.setText(Integer.toString(currentPage + 1)
        + " of "
        + Integer.toString(((DefaultPageableSortableTableModel)
            table.getModel())
                .getTotalPages())));
    if (currentPage == 0) {
        firstButton.setEnabled(false);
        prevButton.setEnabled(false);
    } else {
        firstButton.setEnabled(true);
        prevButton.setEnabled(true);
    }

    if (currentPage == (((DefaultPageableSortableTableModel)
        table.getModel()).getTotalPages() - 1)) {
        lastButton.setEnabled(false);
        nextButton.setEnabled(false);
    } else {
        lastButton.setEnabled(true);
        nextButton.setEnabled(true);
    }
}

protected void initPagingButtons() {
    nextButton.addActionListener(new ActionListener() {
        static final long serialVersionUID = 0L;

        public void actionPerformed(ActionEvent e) {
            currentPage++;
            ((DefaultPageableSortableTableModel)
                table.getModel()).setCurrentPage(currentPage);
            updatePagingButtons();
        }
    });
    prevButton.addActionListener(new ActionListener() {

```

```

        static final long serialVersionUID = 0L;

        public void actionPerformed(ActionEvent e) {
            currentPage--;
            ((DefaultPageableSortableTableModel)
                table.getModel()).setCurrentPage(currentPage);
            updatePagingButtons();
        }
    });
    firstButton.addActionListener(new ActionListener() {
        static final long serialVersionUID = 0L;

        public void actionPerformed(ActionEvent e) {
            currentPage = 0;
            ((DefaultPageableSortableTableModel)
                table.getModel()).setCurrentPage(0);
            updatePagingButtons();
        }
    });
    lastButton.addActionListener(new ActionListener() {
        static final long serialVersionUID = 0L;

        public void actionPerformed(ActionEvent e) {
            currentPage = (((DefaultPageableSortableTableModel)
                table.getModel())
                    .getTotalPages() - 1);
            ((DefaultPageableSortableTableModel)
                table.getModel()).setCurrentPage(currentPage);
            updatePagingButtons();
        }
    });
}

```

navigatortablecellvaluefactory.java

```

/*
 * $Id: NavigatorTableCellValueFactory.java,v 1.1 2008/03/27 09:25:58
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

/**
 * @author ron
 */

```

```

public interface NavigatorTableCellValueFactory {

    /**
     * @param model
     * @param column
     * @param row
     * @return
     */
    public Object getValueAt(NavigatorTableModel model, final int column,
    final int row);

}

```

navigatortablecolumnconfig.java

```

/*
 * $Id: NavigatorTableColumnConfig.java,v 1.1 2008/03/27 09:25:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

/**
 * A configuration for getting the title and values to display in a
 * column of a
 * table.
 *
 * @author ron
 */
public class NavigatorTableColumnConfig {

    private String title;
    private NavigatorTableCellValueFactory tableCellValueFactory;

    /**
     * @param title
     * @param tableCellValueFactory
     */
    public NavigatorTableColumnConfig(String title,
        NavigatorTableCellValueFactory tableCellValueFactory) {
        setTitle(title);
        setTableCellValueFactory(tableCellValueFactory);
    }

    /**
     * @return The column title
     */

```

```

    */

    public String getTitle() {
        return title;
    }

    /**
     * set the title of a column.
     *
     * @param title
     */
    public void setTitle(String title) {
        this.title = title;
    }

    /**
     * @return the factory for generating the value for a cell in this
     * column
     */
    public NavigatorTableCellValueFactory getTableCellValueFactory() {
        return tableCellValueFactory;
    }

    /**
     * set the factory for generating the value for a cell in this
     * column.
     *
     * @param tableCellValueFactory
     */
    public void setTableCellValueFactory(NavigatorTableCellValueFactory
tableCellValueFactory) {
        this.tableCellValueFactory = tableCellValueFactory;
    }
}

```

navigatortableconfig.java

```

/*
 * $Id: NavigatorTableConfig.java,v 1.2 2008/04/24 02:29:24 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

import java.util.ArrayList;
import java.util.List;

```

```

/**
 * @author ron
 */
public class NavigatorTableConfig {

    private final List<Navigator TableColumnConfig> columnConfigs = new
ArrayList<Navigator TableColumnConfig>();
    private boolean rowLevelSelection = false;

    public NavigatorTableConfig() {
        super();
    }

    public List<Navigator TableColumnConfig> getColumnConfigs() {
        return columnConfigs;
    }

    /**
     * Add a new columnConfig to the end of the column list.
     *
     * @param columnConfig
     */
    public void addColumnConfig(Navigator TableColumnConfig columnConfig)
{
    getColumnConfigs().add(columnConfig);
}

    /**
     * Add a new columnConfig at the specified location moving the
columns from
     * that location and after over by one.
     *
     * @param index
     * @param columnConfig
     */
    public void addColumnConfig(int index, Navigator TableColumnConfig
columnConfig) {
    getColumnConfigs().add(index, columnConfig);
}

    public void setRowLevelSelection(boolean rowLevelSelection) {
        this.rowLevelSelection = rowLevelSelection;
    }

    public boolean getRowLevelSelection() {
        return rowLevelSelection;
    }
}

```

```

}

```

navigatortablemanipulator.java

```

/*
 * $Id: NavigatorTableManipulator.java,v 1.4 2008/10/13 22:58:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import java.util.Collection;

import nextapp.echo2.app.Component;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class NavigatorTableManipulator extends
AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getModel(component).getEntities());
    }

    public void setValue(Component component, Object value) {
        getComponent(component).setModel(new
NavigatorTableModel((Collection) value));
    }

    public void addListenerToDetectChangesToInput(final EditMode
editMode, Component component) {
        // A navigator table doesn't require
    }

    @Override
    public NavigatorTableModel getModel(Component component) {
        return getComponent(component).getModel();
    }
}

```

```

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setModel((NavigatorTableModel) valueModel);
}

private NavigatorTable getComponent(Component component) {
    return (NavigatorTable) component;
}
}

```

navigatortablemodel.java

```

/*
 * $Id: NavigatorTableModel.java,v 1.4 2008/04/30 21:03:15 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;

import nextapp.echo2.app.table.AbstractTableModel;

/**
 * @author ron
 */
public class NavigatorTableModel extends AbstractTableModel {
    static final long serialVersionUID = 0;

    private List<Object> entities;
    private NavigatorTableConfig tableConfig;

    public NavigatorTableModel(NavigatorTableModelAdapter entityAdaptor)
    {
        super();
        setEntities(entityAdaptor.getCollection());
        fireTableDataChanged();
    }

    /**
     * @param entities

```

```

     * @param tableModelConfig
     */
    public NavigatorTableModel(Collection<Object> entities) {
        super();
        setEntities(entities);
        fireTableDataChanged();
    }

    public int getColumnCount() {
        if (getTableConfig() != null) {
            return getTableConfig().getColumnConfigs().size();
        }
        return 0;
    }

    public int getRowCount() {
        return getEntities().size();
    }

    /**
     * Return the object used for supplying data for a specific row if
     * applicable.
     *
     * @param row
     * @return
     */
    public Object getBackingObject(int row) {
        return getEntities().get(row);
    }

    @Override
    public Object getValueAt(int column, int row) {
        return
getTableConfig().getColumnConfigs().get(column).getTableCellValueFacto
ry()
            .getValueAt(this, column, row);
    }

    @Override
    public String getColumnName(int column) {
        return getTableConfig().getColumnConfigs().get(column).getTitle();
    }

    /**
     * @return
     */
    protected NavigatorTableConfig getTableConfig() {

```

```

    return tableConfig;
}

public void setTableConfig(NavigatorTableConfig tableConfig) {
    this.tableConfig = tableConfig;
}

public List<Object> getEntities() {
    return entities;
}

protected void setEntities(Collection<Object> entities) {
    this.entities = new ArrayList<Object>(entities);
}
}

```

navigatortablemodeladapter.java

```

/*
 * $Id: NavigatorTableModelAdapter.java,v 1.1 2008/04/30 21:03:15
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Collection;

/**
 * This interface adapts an entity to return a specific collection
 * of an attribute to display in a navigator table. For example if
 * you want to list the line items of an order entity in a navigator
 * table, the order would be the target of the panel and a
 * NavigatorTableModelAdapter can be used when creating the table to
 * return the line items without having to make a custom table that
 * extracts the line items from the order.
 *
 * @author ron
 */
public interface NavigatorTableModelAdapter {

    /**
     * set the object to get the collection from.
     * @param targetObject
     */
    public void setTargetObject(Object targetObject);

```

```

    /**
     * @return The collection of entities to display in the
NavigatorTable.
     */
    public Collection<Object> getCollection();
}

```

navigatortablepanel.java

```

/*
 * $Id: NavigatorTablePanel.java,v 1.4 2008/10/30 05:55:07 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Collection;

import nextapp.echo2.app.Label;
import edu.harvard.fas.rregan.uiframework.MessageHandler;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;

/**
 * @author ron
 */
public class NavigatorTablePanel extends AbstractPanel implements
MessageHandler {
    static final long serialVersionUID = 0L;

    private NavigatorTable table;
    private NavigatorTableConfig tableConfig;
    private Label generalMessage;

    /**
     * @param tableConfig
     * @param supportedContentType
     * @param panelName
     */

```

```

/*
public NavigatorTablePanel(Class<?> supportedContentType, String
panelName) {
    this(NavigatorTablePanel.class.getName(), supportedContentType,
panelName);
}

/**
 * @param resourceBundleName
 * @param supportedContentType
 * @param panelName
 */
public NavigatorTablePanel(String resourceBundleName, Class<?>
supportedContentType,
    String panelName) {
    super(resourceBundleName, PanelActionType.Navigator,
supportedContentType, panelName);
}

@Override
public void setup() {
    super.setup();
    setTable(new NavigatorTable(getTableConfig()));
    add(getTable());
    if (getTargetObject() != null) {
        getTable().setModel(new NavigatorTableModel((Collection)
getTargetObject()));
    }
    generalMessage = new Label();
    add(generalMessage);
}

@Override
public void setTargetObject(Object targetObject) {
    super.setTargetObject(targetObject);
    if (getTable() != null) {
        getTable().setModel(new NavigatorTableModel((Collection)
targetObject));
    }
}

@Override
public void dispose() {
    super.dispose();
}

@Override

```

```

public void setStyleName(String newValue) {
    super.setStyleName(newValue);
    getTable().setStyleName(newValue);
}

protected NavigatorTableConfig getTableConfig() {
    return tableConfig;
}

protected void setTableConfig(NavigatorTableConfig tableConfig) {
    this.tableConfig = tableConfig;
}

protected NavigatorTable getTable() {
    return table;
}

protected void setTable(NavigatorTable table) {
    this.table = table;
}

@Override
public void setGeneralMessage(String message) {
    generalMessage.setText(message);
}
}


```

navigatortree.java

```

/*
 * $Id: NavigatorTree.java,v 1.9 2008/04/01 06:46:26 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.tree;

import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import nextapp.echo2.app.event.ActionEvent;
import echopointng.Tree;
import echopointng.tree.DefaultMutableTreeNode;
import echopointng.tree.DefaultTreeModel;
import echopointng.tree.TreeModel;
import echopointng.treeTreeNode;
```

```

import echopointng.tree.TreeSelectionEvent;
import echopointng.tree.TreeSelectionListener;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * A component for building a navigation tree based on entity objects.
The tree
 * nodes are created by NavigatorTreeNodeFactories assigned for
different types
 * of entities.
 *
 * @author ron
 */
public class NavigatorTree extends Tree implements
TreeSelectionListener {
    static final long serialVersionUID = 0L;

    private final Map<Class<?>, NavigatorTreeNodeFactory>
treeNodeFactoryMap = new HashMap<Class<?>,
NavigatorTreeNodeFactory>();
    private final EventDispatcher eventDispatcher;
    private Object rootObject;

    /**
     * Build a tree starting with the rootObject as the root of the tree,
adding
     * nodes based on the supplied tree node factories, starting with the
     * factory for the rootObject.
     *
     * @param eventDispatcher
     * @param treeNodeFactories
     * @param rootObject
     */
    public NavigatorTree(EventDispatcher eventDispatcher,
    Set<NavigatorTreeNodeFactory> treeNodeFactories, Object rootObject)
{
    super();
    this.eventDispatcher = eventDispatcher;

    for (NavigatorTreeNodeFactory factory : treeNodeFactories) {
        treeNodeFactoryMap.put(factory.getTargetClass(), factory);
    }
    setRootObject(rootObject);
    addTreeSelectionListener(this);
}
}

    /**
     *
     */
private void buildTree() {
    TreeNode node = new DefaultMutableTreeNode();
    if (getRootElement() != null) {
        NavigatorTreeNodeFactory factory =
getNavigatorTreeNodeFactory(getRootElement());
        if (factory != null) {
            node = factory.createTreeNode(eventDispatcher, this,
getRootElement());
        } else {
            node = new DefaultMutableTreeNode(getRootElement());
        }
    }
    // this causes invalidate() to be called.
    setModel(new DefaultTreeModel(node));
}

@Override
public void dispose() {
    setModel(null);
    setRootObject(null);
    super.dispose();
}

private void removeListeners(DefaultTreeModel treeModel) {
    if (treeModel != null) {
        DefaultMutableTreeNode rootNode = (DefaultMutableTreeNode)
treeModel.getRoot();
        if (rootNode != null) {
            Enumeration<DefaultMutableTreeNode> enm =
rootNode.depthFirstEnumeration();
            while (enm.hasMoreElements()) {
                DefaultMutableTreeNode node = enm.nextElement();
                if (node instanceof NavigatorTreeNode) {
                    ((NavigatorTreeNode) node).dispose();
                }
            }
        }
    }
}

/**
 * @return
 */
public Object getRootElement() {
}

```

```

    return rootObject;
}

/**
 * @param rootObject
 */
public void setRootObject(Object rootObject) {
    setModel(null); // this causes invalidate() to be called.
    this.rootObject = rootObject;
    if (this.rootObject != null) {
        buildTree();
    }
}

@Override
public DefaultTreeModel getModel() {
    return (DefaultTreeModel) super.getModel();
}

@Override
public void setModel(TreeModel newTreeModel) {
    DefaultTreeModel oldModel = getModel();
    if (oldModel != null) {
        removeListeners(oldModel);
    }
    super.setModel(newTreeModel);
}

/**
 * @param target
 * @return
 */
public NavigatorTreeNodeFactory getNavigatorTreeNodeFactory(Object target) {
    Class<?> targetType = ((target instanceof Class<?>) ? (Class<?>) target : target.getClass());
    NavigatorTreeNodeFactory factory = null;
    factory = treeNodeFactoryMap.get(targetType);
    if (factory != null) {
        return factory;
    }
    if (targetType.getInterfaces() != null) {
        for (Class<?> targetInterface : targetType.getInterfaces()) {
            factory = getNavigatorTreeNodeFactory(targetInterface);
            if (factory != null) {
                return factory;
            }
        }
    }
    if ((targetType.getSuperclass() != null)
        && !Object.class.equals(targetType.getSuperclass())))
        factory = getNavigatorTreeNodeFactory(targetType.getSuperclass());
}
return factory;
}

/**
 * Listen for tree selection changes and if they occur on a navigator
node
 * fire the event for that node.
 *
 * @see
echopointng.tree.TreeSelectionListener#valueChanged(echopointng.tree.T
reeSelectionEvent)
*/
public void valueChanged(TreeSelectionEvent e) {
    TreeNode node = (TreeNode) e.getPath().getLastPathComponent();
    if (node instanceof NavigatorTreeNode) {
        ActionEvent eventToFire = ((NavigatorTreeNode) node).getEventToFire();
        if (eventToFire != null) {
            eventDispatcher.dispatchEvent(eventToFire);
        }
    }
}
}

navigatortreenode.java

/*
 * $Id: NavigatorTreeNode.java,v 1.7 2008/09/12 00:15:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.navigation.tree;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import echopointng.tree.DefaultMutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

```

```

/**
 * A mutable tree node that fires a navigation event when clicked. The
node may
 * have a listener that that listens for update events for the entity
object
 * that the node is displaying to indicate that the node view should
be updated.
 *
 * @author ron
 */
public class NavigatorTreeNode extends DefaultMutableTreeNode
implements ActionListener {
    static final long serialVersionUID = 0L;

    private final EventDispatcher eventDispatcher;
    private Object targetObject;
    private ActionEvent eventToFire;
    private NavigatorTreeNodeUpdateListener updateListener;

    public NavigatorTreeNode(EventDispatcher eventDispatcher, Object
targetObject) {
        this(eventDispatcher, targetObject, targetObject);
    }

    public NavigatorTreeNode(EventDispatcher eventDispatcher, Object
targetObject, Object label) {
        this(eventDispatcher, targetObject, label, null);
    }

    public NavigatorTreeNode(EventDispatcher eventDispatcher, Object
targetObject, Object label,
        ActionEvent eventToFire) {
        super(label);
        this.eventDispatcher = eventDispatcher;
        setEventToFire(eventToFire);
        setTargetObject(targetObject);
    }

    public ActionEvent getEventToFire() {
        return eventToFire;
    }

    public void setEventToFire(ActionEvent eventToFire) {
        this.eventToFire = eventToFire;
    }
}

```

```

public void actionPerformed(ActionEvent e) {
    if (updateListener != null) {
        updateListener.actionPerformed(e);
    }
}

public Object getTargetObject() {
    return targetObject;
}

public void setTargetObject(Object targetObject) {
    this.targetObject = targetObject;
}

public EventDispatcher getEventDispatcher() {
    return eventDispatcher;
}

/***
 * Set a listener that gets notified if the object for the node is
modified.
 * The new listener will be registered with the event dispatcher by
this
 * method. If another listener is already set, it will be
unregistered
 * before the new one is registered. If null is supplied, the old
listener
 * will be unregistered and no listeners will be listening for this
node.
 *
 * @param updateListener
 */
public void setUpdateListener(NavigatorTreeNodeUpdateListener
updateListener) {
    if (this.updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            this.updateListener);
    }
    this.updateListener = updateListener;
    if (this.updateListener != null) {
        getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.c
lass,
            this.updateListener);
    }
}

```

```

public void dispose() {
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        this.updateListener = null;
    }
}

```

navigatortreenodefactory.java

```

/*
 * $Id: NavigatorTreeNodeFactory.java,v 1.3 2008/03/07 10:37:27 rregan
rExp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.tree;

import echopointng.tree.MutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public interface NavigatorTreeNodeFactory {

    /**
     * Return the type of entity object this factory builds TreeNodes
for.
     *
     * @return
     */
    public Class<?> getTargetClass();

    /**
     * Given some object create an appropriate tree representation and
return
     * the root of that tree.
     *
     * @param eventDispatcher
     * @param tree
     * @param object
     * @return
     */
}

```

```

    public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
Object object);
}

```

navigatortreenodeupdatelistener.java

```

/*
 * $Id: NavigatorTreeNodeUpdateListener.java,v 1.3 2008/03/07 10:37:27
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.tree;

import nextapp.echo2.app.event.ActionListener;

/**
 * @author ron
 */
public interface NavigatorTreeNodeUpdateListener extends
ActionListener {
}

```

navigatortreepanel.java

```

/*
 * $Id: NavigatorTreePanel.java,v 1.4 2008/12/17 02:00:42 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Set;

import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory;

/**
 * @author ron
 */
public class NavigatorTreePanel extends AbstractPanel {
}

```

```

static final long serialVersionUID = 0L;

private NavigatorTree tree;
private Set<NavigatorTreeNodeFactory> treeNodeFactories;

/**
 * @param treeNodeFactories
 * @param supportedContentType
 */
public NavigatorTreePanel(Set<NavigatorTreeNodeFactory>
treeNodeFactories,
    Class<?> supportedContentType) {
    this(NavigatorTreePanel.class.getName(), treeNodeFactories,
supportedContentType);
}

/**
 * @param resourceBundleName
 * @param treeNodeFactories
 * @param supportedContentType
 */
public NavigatorTreePanel(String resourceBundleName,
    Set<NavigatorTreeNodeFactory> treeNodeFactories, Class<?>
supportedContentType) {
    super(resourceBundleName, PanelActionType.Navigator,
supportedContentType);
    setTreeNodeFactories(treeNodeFactories);
}

/**
 * @param resourceBundleName
 * @param treeNodeFactories
 * @param supportedContentType
 * @param panelName
 */
public NavigatorTreePanel(String resourceBundleName,
    Set<NavigatorTreeNodeFactory> treeNodeFactories, String panelName)
{
    super(resourceBundleName, PanelActionType.Navigator, null,
panelName);
    setTreeNodeFactories(treeNodeFactories);
}

@Override
public void setup() {
    super.setup();
}

    setTree(new NavigatorTree(getEventDispatcher(),
getTreeNodeFactories(), getTargetObject()));
    add(getTree());
}

@Override
public void dispose() {
    super.dispose();
    NavigatorTree tree = getTree();
    if (tree != null) {
        remove(getTree());
        getTree().dispose();
        setTree(null);
    }
}

@Override
public void setStyleName(String newValue) {
    super.setStyleName(newValue);
    getTree().setStyleName(newValue);
}

@Override
public void setTargetObject(Object targetObject) {
    super.setTargetObject(targetObject);
    if (getTree() != null) {
        getTree().setRootObject(targetObject);
    }
}

protected Set<NavigatorTreeNodeFactory> getTreeNodeFactories() {
    return treeNodeFactories;
}

protected void setTreeNodeFactories(Set<NavigatorTreeNodeFactory>
treeNodeFactories) {
    this.treeNodeFactories = treeNodeFactories;
}

protected NavigatorTree getTree() {
    return tree;
}

protected void setTree(NavigatorTree tree) {
    this.tree = tree;
}
}

```

nlpnavigatorpanel.java

```
/*
 * $Id: NLPNavigatorPanel.java,v 1.3 2008/12/19 09:20:37 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.ui;

import java.util.Set;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFactory;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTreePanel;

/**
 * @author ron
 */
public class NLPNavigatorPanel extends NavigatorTreePanel {
    private static final Logger log =
Logger.getLogger(NLPNavigatorPanel.class);
    static final long serialVersionUID = 0;

    /**
     * @param treeNodeFactories
     */
    public NLPNavigatorPanel(Set<NavigatorTreeNodeFactory>
treeNodeFactories) {
        super(NLPNavigatorPanel.class.getName(), treeNodeFactories,
NLPPanelNames.NLP_NAVIGATOR_PANEL_NAME);
    }

    @Override
    public void dispose() {
        super.dispose();
        removeAll();
    }

    @Override
    public void setTargetObject(Object targetObject) {
        // this panel doesn't support a target class
    }

    @Override
```

```
    public Object getTargetObject() {
        return this.getClass();
    }
}
```

nlpnavigatortreenodefactory.java

```
/*
 * $Id: NLPNavigatorTreeNodeFactory.java,v 1.2 2008/12/17 02:00:42
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.ui;

import nextapp.echo2.app.Label;
import nextapp.echo2.app.event.ActionEvent;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import echopointng.tree.MutableTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeUp
dateListener;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
```

```

@Component("nlpNavigatorTreeNodeFactory")
@Scope("singleton")
public class NLPNavigatorTreeNodeFactory extends
AbstractNavigatorTreeNodeFactory {

    /**
     * The property name to use to control the label on the parser node
     * generated by the factory.
     */
    public final static String PROP_PARSER_NODE_LABEL =
"ParserNodeLabel";

    /**
     * @param eventDispatcher
     */
    public NLPNavigatorTreeNodeFactory() {
        super(NLPNavigatorTreeNodeFactory.class.getName(),
NLPNavigatorPanel.class);
    }

    /**
     * @see
     * edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
     ctory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.tre
e.NavigatorTree,
     *          java.lang.Object)
     */
    public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
Object object) {

        NavigationEvent openParserPanelEvent = new OpenPanelEvent(tree,
PanelActionType.Editor,
        null, null, NLPPanelNames.PARSER_PANEL_NAME,
WorkflowDisposition.NewFlow);

        String sparserPanelNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_PARSER_NODE_LABEL, "Parser");

        NavigatorTreeNode parserPanelTreeNode = new
NavigatorTreeNode(eventDispatcher, null,
        new Label(sparserPanelNodeLabel), openParserPanelEvent);

        return parserPanelTreeNode;
    }
}

```

```

private static class UpdateListener implements
NavigatorTreeNodeUpdateListener {
    static final long serialVersionUID = 0L;

    private final NavigatorTreeNode projectTreeNode;

    private UpdateListener(NavigatorTreeNode projectTreeNode) {
        this.projectTreeNode = projectTreeNode;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
        }
    }
}

```

nlppanelnames.java

```

/*
 * $Id: NLPPanelNames.java,v 1.1 2008/12/16 23:03:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.ui;

/**
 * This defines constants for all the panel names used in the nlp
package.
 *
 * @author ron
 */
public interface NLPPanelNames {

    /**
     * The name of the panel for testing the parser.
     */
    public static final String PARSER_PANEL_NAME = "parserPanel";

    public static final String NLP_NAVIGATOR_PANEL_NAME =
"nlpNavigatorPanel";
}

```

nlprocessor.java

```
/*
 * $Id: NLProcessor.java,v 1.5 2009/01/24 11:08:40 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

/**
 * @param <T> -
 *          the type of result from the process method.
 * @author ron
 */
public interface NLProcessor<T> {

    /**
     * Apply this processor to the supplied NLPText.
     *
     * @param text
     * @return a processor specific type
     */
    public T process(NLPText text);
}
```

nlprocessorexception.java

```
/*
 * $Id: NLProcessorException.java,v 1.1 2009/02/10 10:26:04 rregan
 * Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.ApplicationException;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * @author ron
 */
public class NLProcessorException extends ApplicationException {
    static final long serialVersionUID = 0;

    protected static final String MSG_FORMAT_NO_PRIMARY_VERB = "No
primary verb found in text '%s'";
```

```
    /**
     * @param text -
     *          The text being processed
     * @return a NLProcessorException with a message that the primary
verb
     *         wasn't found in the supplied NLPText.
     */
    protected static NLProcessorException noPrimaryVerb(NLPText text) {
        return new NLProcessorException(MSG_FORMAT_NO_PRIMARY_VERB, text);
    }

    /**
     * @param format
     * @param args
     */
    protected NLProcessorException(String format, Object... args) {
        super(format, args);
    }

    /**
     * @param cause
     * @param format
     * @param args
     */
    protected NLProcessorException(Throwable cause, String format,
Object... args) {
        super(cause, format, args);
    }
}
```

nlprocessorfactory.java

```
/*
 * $Id: NLProcessorFactory.java,v 1.17 2009/03/27 07:16:09 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp;

import java.util.Collection;
import java.util.List;

/**
```

```

* @author ron
*/
public interface NLPProcessorFactory {

    /**
     * @param text -
     *      The text that will be processed
     * @return an NLPText used for processing and analysis.
     */
    public NLPText createNLPText(String text);

    /**
     * Create a new NLPText for the given text and apply some processors.
     *
     * @param text -
     *      The text that will be processed
     * @return an NLPText used for processing and analysis.
     */
    public NLPText processText(String text);

    /**
     * @param texts -
     *      two or more NLPText objects to append together.
     * @return a new NLPText consisting of the supplied texts
     */
    public NLPText appendText(NLPText... texts);

    /**
     * @param texts -
     *      two or more NLPText objects to append together.
     * @return a new NLPText consisting of the supplied texts
     */
    public NLPText appendText(List<NLPText> texts);

    /**
     * @return an NLPProcessor that takes an NLPText object and generates
     * a constituent parse tree returned in a String.
     */
    public NLPProcessor<String> getConstituentTreePrinter();

    /**
     * @return an NLPProcessor that takes an NLPText object and generates
     * a description of the word dependencies of the text returned
     * in a String.
     */
    */

    /**
     * @return an NLPProcessor<String> getDependencyPrinter();
     */
    /**
     * @return an NLPProcessor that takes an NLPText object and generates
     * a description of the semantic roles each node plays for the
     * verbs
     *      in the text.
     */
    public NLPProcessor<String> getSemanticRolePrinter();

    /**
     * @return an NLPProcessor that takes an NLPText object and returns
     * the
     *      maximum depth of the syntax structure.
     */
    public NLPProcessor<Integer> getConstituentTreeDepthFinder();

    /**
     * @return an NLPProcessor that takes an NLPText object and updates
     * it with
     *      the lemma (base form) of the text word.
     */
    public NLPProcessor<NLPText> getLemmatizer();

    /**
     * @return an NLPProcessor that takes an NLPText SENTENCE object,
     * identifies
     *      tokens if needed and creates the constituent clauses,
     * phrases and
     *      words that make up the text as well as dependencies
     * between the
     *      words.
     */
    public NLPProcessor<NLPText> getParser();

    /**
     * @return an NLPProcessor that takes an NLPText object and
     * identifies the
     *      sentences in it if appropriate.
     */
    public NLPProcessor<NLPText> getSentencizer();

    /**
     * @return an NLPProcessor that takes an NLPText object and
     * identifies the
     */
}

```

```

        *      words or tokens that make it up.
    */
public NLPProcessor<NLPText> getTokenizer();

/**
 * @return an NLPProcessor that takes a NLPText object and indicates
the
 *      semantic roles: including primary verb, agent, patient and
theme
 *      of the constituents.
 */
public NLPProcessor<NLPText> getSemanticRoleLabeler();

/**
 * @return an NLPProcessor that takes an NLPText object and
identifies all
 *      the noun phrases in it.
 */
public NLPProcessor<Collection<NLPText>> getNounPhraseFinder();

/**
 * @return an NLPProcessor that takes an NLPWord that maybe
misspelled and
 *      returns a collection of words that may be the correct
spelling.
 */
public NLPProcessor<Collection<NLPText>> getSimilarWordFinder();

/**
 * @return an NLPProcessor that takes an NLPWord with sense
information and
 *      suggests more specific senses (hyponyms.)
 */
public NLPProcessor<Collection<NLPText>>
getMoreSpecificWordSugester();

/**
 * @return an NLPProcessor that takes an NLPWord and returns true if
the
 *      word is found in the dictionary, or false if not.
 */
public NLPProcessor<Boolean> getSpellingChecker();

/**
 * @see {@link NLPText#getPrimaryVerb()}
 * @return an NLPProcessor that takes an NLPText object and
identifies the

```

```

        *      primary verb.
    */
public NLPProcessor<NLPText> getPrimaryVerbFinder();

/**
 * @return an NLPProcessor that takes an NLPText object and
disambiguates
 *      all the words.
 */
public NLPProcessor<NLPText> getWordSenseDisambiguator();

/**
 * @return an NLPProcessor that takes an NLPText object and links the
 *      WordNet word reference to each available word in the text.
 */
public NLPProcessor<NLPText> getDictionizer();

/**
 * @return an NLPProcessor that takes an NLPText object and
identifies words
 *      that represent named entities like people, organizations,
and
 *      locations.
 */
public NLPProcessor<NLPText> getNamedEntityResolver();
}


```

nlpprocessorfactoryimpl.java

```

/*
 * $Id: NLPPProcessorFactoryImpl.java,v 1.33 2009/03/30 11:54:32 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;

```

```

import org.springframework.context.ApplicationContextAware;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPPProcessorFactory;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.impl.lemmatizer.SimpleLemmatizer;
import edu.harvard.fas.rregan.nlp.impl.srl.SemanticRoleLabeler;
import edu.harvard.fas.rregan.nlp.impl.srl.SemanticRolePrinter;
import edu.harvard.fas.rregan.nlp.impl.wsd.WordnetWSD;

/**
 * This class serves two duties. It acts as a factory for getting
processors
 * that perform an NLP task on NLPText, and it acts as a simple
controller
 * through the processText() method that orchestrates the primary NLP
Tasks.
 *
 * @author ron
 */
@Component("nlpProcessorFactory")
@Scope("singleton")
public class NLPPProcessorFactoryImpl implements NLPPProcessorFactory,
ApplicationContextAware {
    private static final Logger log =
Logger.getLogger(NLPPProcessorFactoryImpl.class);

    private ApplicationContext applicationContext;

    // cache of processed text
    private final Map<String, NLPText> processedTextCache = new
HashMap<String, NLPText>();

    /**
     * @param dictionaryRepository
     */
    public NLPPProcessorFactoryImpl() {
    }

    @Override
    public void setApplicationContext(ApplicationContext
applicationContext) throws BeansException {
        this.applicationContext = applicationContext;
    }

    /**
     * @return
     */
    protected ApplicationContext getApplicationContext() {
        return applicationContext;
    }

    /**
     * @see
     * edu.harvard.fas.rregan.nlp.NLPPProcessorFactory#createNLPText()
     */
    @Override
    public NLPText createNLPText(String text) {
        return new NLPTextImpl(text);
    }

    @Override
    public NLPText processText(String text) {
        long startTime = System.currentTimeMillis();
        NLPText nlpText = null;
        if (!processedTextCache.containsKey(text)) {
            try {
                nlpText = createNLPText(text);
                getSentencizer().process(nlpText);
                getParser().process(nlpText);
                getPrimaryVerbFinder().process(nlpText);
                getLemmatizer().process(nlpText);
                getDictionary().process(nlpText);
                getNamedEntityResolver().process(nlpText);
                getWordSenseDisambiguator().process(nlpText);
                getSemanticRoleLabeler().process(nlpText);
                processedTextCache.put(text, nlpText);
            } catch (NLPPProcessorException e) {
                log.error("NLP processing threw an exception.", e);
            }
        } else {
            nlpText = processedTextCache.get(text);
        }
        log.info("NLP Processing Time: " + (System.currentTimeMillis() -
startTime) + " ms");
        log.info("processed text: \n" +
getConstituentTreePrinter().process(nlpText));
        return nlpText;
    }

    @Override
    public NLPText appendText(List<NLPText> texts) {

```

```

NLPText newText = new NLPTextImpl(null);
if (texts != null) {
    for (NLPText text : texts) {
        newText.addChild(text.clone());
    }
}
return newText;
}

@Override
public NLPText appendText(NLPText... texts) {
    return appendText(Arrays.asList(texts));
}

@Override
public NLPProcessor<String> getConstituentTreePrinter() {
    return new StringNLPTextWalker(new ConstituentTreePrinter());
}

@Override
public NLPProcessor<String> getDependencyPrinter() {
    return new StringNLPTextWalker(new DependencyPrinter());
}

@Override
public NLPProcessor<String> getSemanticRolePrinter() {
    return new StringNLPTextWalker(new SemanticRolePrinter());
}

@Override
public NLPProcessor<Integer> getConstituentTreeDepthFinder() {
    return new IntegerNLPTextWalker(new ConstituentTreeDepthFinder());
}

/**
 * @see
edu.harvard.fas.rregan.nlp.NLPProcessorFactory#getSentencizer()
 */
@Override
public NLPProcessor<NLPText> getSentencizer() {
    return newInstance(Sentencizer.class);
}

/**
 * @see edu.harvard.fas.rregan.nlp.NLPProcessorFactory#getTokenizer()
 */
@Override
public NLPProcessor<NLPText> getTokenizer() {
    return getParser();
}

/**
 * @see edu.harvard.fas.rregan.nlp.NLPProcessorFactory#getParser()
 */
@Override
public NLPProcessor<NLPText> getParser() {
    return newInstance(StanfordLexicalizedParser.class);
}

@Override
public NLPProcessor<NLPText> getLemmatizer() {
    return newInstance(SimpleLemmatizer.class);
}

public NLPProcessor<NLPText> getWordSenseDisambiguator() {
    return newInstance(WordnetWSD.class);
}

@Override
public NLPProcessor<NLPText> getDictionizer() {
    return newInstance(Dictionizer.class);
}

@Override
public NLPProcessor<Collection<NLPText>> getSimilarWordFinder() {
    return newInstance(SpellingSuggerster.class);
}

@Override
public NLPProcessor<Collection<NLPText>>
getMoreSpecificWordSuggerster() {
    return newInstance(MoreSpecificWordSuggerster.class);
}

@Override
public NLPProcessor<Boolean> getSpellingChecker() {
    return newInstance(SpellingChecker.class);
}

/**
 * @see
edu.harvard.fas.rregan.nlp.NLPProcessorFactory#getSemanticRoleLabeler()
 */
*/

```

```

@Override
public NLPProcessor<NLPText> getSemanticRoleLabeler() {
    return newInstance(SemanticRoleLabeler.class);
}

@Override
public NLPProcessor<Collection<NLPText>> getNounPhraseFinder() {
    return newInstance(NounPhraseFinder.class);
}

@Override
public NLPProcessor<NLPText> getNamedEntityResolver() {
    return newInstance(StanfordNameEntityRecognizer.class);
}

@Override
public NLPProcessor<NLPText> getPrimaryVerbFinder() {
    return newInstance(DependencyPrimaryVerbFinder.class);
}

protected <T> T newInstance(Class<T> processorType) {
    return (T) getApplicationContext().getAutowireCapableBeanFactory()
        .createBean(processorType);
}
}

```

nlptext.java

```

/*
 * $Id: NLPText.java,v 1.29 2009/03/22 11:08:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

import java.util.List;
import java.util.Set;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import edu.harvard.fas.rregan.nlp.impl.wsd.SenseRelationInfo;

/**
 * Represents various levels of text grammatically from word to full
stories.
 *
 * @author ron

```

```

/*
 * public interface NLPText extends Cloneable {
 *
 *     /**
 *      * @return the parent NLPText element that wholly contains this
NLPText
 *      *          element.
 *     */
 *     public NLPText getParent();
 *
 *     /**
 *      * @return the ancestor NLPText nodes of this node in the order
closest
 *      *          first. An empty list is returned for nodes with no
parents.
 *     */
 *     public List<NLPText> getAncestors();
 *
 *     /**
 *      * @param ancestor -
 *      *          a text node.
 *      * @return true if the supplied text node is in the ancestors of this
node.
 *     */
 *     public boolean isDescendentOf(NLPText ancestor);
 *
 *     /**
 *      * @return the text that makes up this element, possibly contained in
sub
 *      *          elements.
 *     */
 *     public String getText();
 *
 *     /**
 *      * Return a substring of the leaves of this node.<br>
 *      * For example the following string of 8 leaves:<br>
 *      * <code>
 *      * these are the leaves of this node .
 *      *
 *      * getTextRange(2) -> "the leaves of this node ."
 *      * getTextRange(0) -> "these are the leaves of this node ."
 *      * </code>
 *      *
 *      * @param startIndex -
 *      *          the zero based index of the first leaf to include.
 *      * @return the string of the leaves starting from the leaf at the
specified

```

```

        index.

*/
public String getTextRange(int startIndex);

/**
 * Return a substring of the leaves of this node.<br>
 * For example the following string of 8 leaves:<br>
 * <code>
 * these are the leaves of this node .
 *
 * getTextRange(2,3) -> "the leaves"
 * getTextRange(0,0) -> "these"
 * </code>
 *
 * @param startIndex -
 *          the zero based index of the first leaf to include.
 * @param endIndex -
 *          the zero based index of the last leaf to include.
 * @return the string of the leaves from the leaf at the specified
index and
 *          ending at the specified index inclusive.
 */
public String getTextRange(int startIndex, int endIndex);

/**
 * @param texts -
 *          zero or more NLPText objects to append to this text.
 * @return a new NLPText consisting of (as children) this text
appended with
 *          the supplied text
 */
public NLPText appendText(NLPText... texts);

/**
 * @param texts -
 *          zero or more NLPText objects to append to this text.
 * @return a new NLPText consisting of (as children) this text
appended with
 *          the supplied text
 */
public NLPText appendText(List<NLPText> texts);

/**
 * @return The Penn Treebank tag for this element.
 */
public ParseTag getParseTag();

```

```

    /**
     * @param tag -
     *          the parse tag
     * @return true if this element has been parsed and the parse tag
matches
     *          the supplied parse tag.
     */
    public boolean is(ParseTag tag);

    /**
     * @param tags -
     *          the parse tags
     * @return true if this element has been parsed and the parse tag
matches
     *          one of the supplied parse tags.
     */
    public boolean in(ParseTag... tags);

    /**
     * @return The GrammaticalStructureLevel of this element, such as
WORD,
     *          PHRASE, or CLAUSE.
     * @see GrammaticalStructureLevel
     */
    public GrammaticalStructureLevel getGrammaticalStructureLevel();

    /**
     * @param grammaticalStructureLevel
     * @return true if this node's GrammaticalStructureLevel equals the
supplied
     *          level.
     */
    public boolean is(GrammaticalStructureLevel
grammaticalStructureLevel);

    /**
     * @param grammaticalStructureLevels
     * @return true if this node's GrammaticalStructureLevel equals one
of the
     *          supplied levels.
     */
    public boolean in(GrammaticalStructureLevel...
grammaticalStructureLevels);

    /**
     * @return A list of the children NLPText elements in the order they
appear
     */

```

```

        *      in this element.
    */
public List<NLPText> getChildren();

/**
 * Add a new child element to the end of the children list and make
its
 * parent this object.
 *
 * @param child
 */
public void addChild(NLPText child);

/**
 * Add a child to this NLPText without removing it from another
NLPText.
 * This is used for adding words from another NLPText to a bag of
words
 * NLPText.
 *
 * @param text
 */
public void addRef(NLPText text);

/**
 * Remove a child from this object.
 *
 * @param child
 */
public void removeChild(NLPText child);

/**
 * @return A list of the leaf elements of all sub nodes. If this
element is
 *         a leaf an empty list is returned. The leaves should all be
Words
 */
public List<NLPText> getLeaves();

/**
 * @return true if this node is a leaf node.
 */
public boolean isLeaf();

/**
 * @return the primary verb of the sentence.
 */

```

```

public NLPText getPrimaryVerb();

/**
 * Set the primary verb of the sentence.
 *
 * @param primaryVerb
 */
public void setPrimaryVerb(NLPText primaryVerb);

/**
 * @param verb -
 *         an NLPText representing the verb to get the semantic
role for.
 * @return The semantic role of this NLPText for the supplied verb or
null.
 */
public SemanticRole getSemanticRole(NLPText verb);

/**
 * @return For a WORD, PHRASE or CLAUSE, the verbs that this NLPText
 *         supports with semantic roles.
 */
public Set<NLPText> getSupportedVerbs();

/**
 * @param verb -
 *         an NLPText representing the verb that this node fills
the
 *         semantic role for.
 * @param semanticRole -
 *         the semantic role of this NLPText node.
 */
public void setSemanticRole(NLPText verb, SemanticRole semanticRole);

/**
 * @return the index of a WORD level element in the original text,
this
 *         returns -1 for non word tokens.
 */
public Integer getWordIndex();

/**
 * @return The part of speech of a WORD level element, this returns
UNKNOWN
 *         for non word elements.
 */
public PartOfSpeech getPartOfSpeech();

```

```

/**
 * @return the lemma (base form) of a WORD level element, returns
null for
 *      non-word elements.
 */
public String getLemma();

/**
 * Set the lemma for a WORD level element.
 *
 * @param lemma -
 *      the base form of the word returned by text.
 */
public void setLemma(String lemma);

/**
 * @return The WordNet word of a WORD level element.
 */
public Word getDictionaryWord();

/**
 * @param dictionaryWord
 */
public void setDictionaryWord(Word dictionaryWord);

/**
 * @return The WordNet word sense of a WORD level element.
 */
public Sense getDictionaryWordSense();

/**
 * @param dictionaryWordSense
 */
public void setDictionaryWordSense(Sense dictionaryWordSense);

/**
 * @return The information collected by the WSD used to determine the
sense
 *      of the word.
 */
public Set<SenseRelationInfo> getDictionaryWordSenseRelationInfo();

/**
 * Set the information collected by the WSD used to determine the
sense of
 * the word.

```

```

*
 * @param relationInfo
 */
public void setDictionaryWordSenseRelationInfo(Set<SenseRelationInfo>
relationInfo);

/**
 * @return true if this word is the name of an entity such as a
person,
 *      organization, or location.
 */
public boolean isNamedEntity();

/**
 * set if this word is the name of an entity such as a person,
organization,
 * or location.
 *
 * @param isNamedEntity
 */
public void setNamedEntity(boolean isNamedEntity);

/**
 * @param partOfSpeech -
 *      the part of speech.
 * @return true if this node is a WORD with the supplied part of
speech,
 *      false otherwise.
 */
public boolean is(PartOfSpeech partOfSpeech);

/**
 * @param partOfSpeechs -
 *      list of parts of speech
 * @return true if the part of speech of this text is in the supplied
list
 *      of parts of speech.
 */
public boolean in(PartOfSpeech... partOfSpeechs);

/**
 * @return
 */
public Set<GrammaticalRelation> getGrammaticalRelations();

/**

```

```

 * @return The grammatical relations this node is the governor of
(relation
 *         from this element) for WORD nodes.
 */
public Set<GrammaticalRelation> getGovernorOf();

/**
 * @param relation -
 *         the type of relation.
 * @return The grammatical relations of the supplied type that this
node is
 *         the governor of for WORD nodes.
 */
public Set<GrammaticalRelation>
getGovernorOfType(GrammaticalRelationType relation);

/**
 * @param relationType -
 *         the type of relation this word governs.
 * @param dependent -
 *         the dependent word of the relation.
 * @return true if this word is the governor in a relation of the
supplied
 *         type to the supplied word.
 */
public boolean isGovernorOf(GrammaticalRelationType relationType,
NLPText dependent);

/**
 * @return The grammatical relations this node is the dependent of
(relation
 *         to this element) for WORD nodes.
 */
public Set<GrammaticalRelation> getDependentOf();

/**
 * @param relation -
 *         the type of relation.
 * @return The grammatical relations of the supplied type that this
node is
 *         the dependent of for WORD nodes
 */
public Set<GrammaticalRelation>
getDependentOfType(GrammaticalRelationType relation);

/**
 * @param relationType -

```

```

 *         the type of relation this word is a dependent of.
 * @param governor -
 *         the governing word of the relation.
 * @return true if this word is the dependent in a relation of the
supplied
 *         type to the supplied word.
 */
public boolean isDependentOf(GrammaticalRelationType relationType,
NLPText governor);

/**
 * @return true if the NLPText is not empty.
 */
public boolean hasText();

/**
 * @return true if a lexical analyzer detected errors in the word
level
 *         properties of a WORD, or if any WORD elements of a large
element
 *         has lexical errors.
 */
public boolean hasLexicalErrors();

/**
 * @return a deep copy of the original object.
 * @throws CloneNotSupportedException
 */
public NLPText clone();
}
```

nlptextimpl.java

```

/*
 * $Id: NLPTextImpl.java,v 1.32 2009/02/12 02:21:17 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
```

```

import java.util.Set;
import java.util.TreeSet;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import edu.harvard.fas.rregan.nlp.impl.wsd.SenseRelationInfo;

/**
 * Represents various levels of text grammatically from word to full
stories.
 *
 * @author ron
 */
public class NLPTextImpl implements NLPText, Comparable<NLPText> {
    private NLPText parent;
    private String text;
    private GrammaticalStructureLevel grammaticalStructureLevel =
GrammaticalStructureLevel.UNKNOWN;
    private ParseTag parseTag = ParseTag.X;
    private final List<NLPText> children = new ArrayList<NLPText>();
    private final Set<GrammaticalRelation> grammaticalRelations = new
HashSet<GrammaticalRelation>();
    private NLPText primaryVerb;
    private final Map<NLPText, SemanticRole> semanticRoleVerbMap = new
HashMap<NLPText, SemanticRole>();

    // word and lexical info
    private PartOfSpeech partOfSpeech = PartOfSpeech.UNKNOWN;
    private int wordIndex = -1;
    private String lemma;
    private boolean lexicalErrors;
    private Word dictionaryWord;
    private Sense dictionaryWordSense;
    private Set<SenseRelationInfo> dictionaryWordSenseRelationInfo;
    private boolean isNamedEntity;

    // dependency relations
    private final Set<GrammaticalRelation> governs = new
TreeSet<GrammaticalRelation>();
}

```

```

    private final Set<GrammaticalRelation> depends = new
TreeSet<GrammaticalRelation>();

    /**
     * @param text
     */
    public NLPTextImpl(String text) {
        setText(text);
    }

    /**
     * For creating a bag of words
     */
    public NLPTextImpl() {
        setGrammaticalStructureLevel(GrammaticalStructureLevel.BAGOFWORDS);
    }

    /**
     * For creating text of sentence and higher level root nodes with no
parent.
     *
     * @param text
     * @param grammaticalStructureLevel
     */
    public NLPTextImpl(String text, GrammaticalStructureLevel
grammaticalStructureLevel) {
        setText(text);
        setGrammaticalStructureLevel(grammaticalStructureLevel);
    }

    /**
     * For creating text of sentence and higher level nodes that have a
parent.
     *
     * @param parent
     * @param text
     * @param grammaticalStructureLevel
     */
    public NLPTextImpl(NLPText parent, String text,
GrammaticalStructureLevel grammaticalStructureLevel) {
        setParent(parent);
        setText(text);
        setGrammaticalStructureLevel(grammaticalStructureLevel);
    }

    /**
     * For creating sentence and lower level nodes based on a parse tag.
     */
}

```

```

*
 * @param parent
 * @param parseTag
 */
protected NLPTextImpl(NLPText parent, ParseTag parseTag) {
    setParent(parent);
    setParseTag(parseTag);
}

/**
 * TODO: this is public for some old tests.
 *
 * @param text
 * @param partOfSpeech
 */
public NLPTextImpl(String text, PartOfSpeech partOfSpeech) {
    setText(text);
    setPartOfSpeech(partOfSpeech);
    setGrammaticalStructureLevel(GrammaticalStructureLevel.WORD);
}

// for word level nodes
protected NLPTextImpl(NLPText parent, int wordIndex, String text,
ParseTag parseTag) {
    setParent(parent);
    setWordIndex(wordIndex);
    setText(text);
    setParseTag(parseTag);
}

public NLPText getParent() {
    return parent;
}

protected void setParent(NLPText parent) {
    this.parent = parent;
}

@Override
public List<NLPText> getAncestors() {
    List<NLPText> ancestors = new ArrayList<NLPText>(10);
    NLPText current = getParent();
    while (current != null) {
        ancestors.add(current);
        current = current.getParent();
    }
    return ancestors;
}

}
@Override
public boolean isDescendentOf(NLPText ancestor) {
    return getAncestors().contains(ancestor);
}

public String getText() {
    if ((text == null) && !getChildren().isEmpty()) {
        StringBuilder sb = new StringBuilder();
        sb.append(getChildren().get(0).getText());
        for (int i = 1; i < getChildren().size(); i++) {
            // don't add a space before a possesive "'s" or punctuation
            NLPText next = getChildren().get(i);
            if (!next.is(ParseTag.POS) && !next.is(PartOfSpeech.PUNCTUATION))
{
                sb.append(" ");
            }
            sb.append(next.getText());
        }
        text = sb.toString();
    }
    return text;
}

@Override
public String getTextRange(int startIndex, int endIndex) {
    StringBuilder sb = new StringBuilder();
    sb.append(getLeaves().get(startIndex).getText());
    for (int i = startIndex + 1; i <= endIndex; i++) {
        NLPText next = getLeaves().get(i);
        // don't add a space before a possesive "'s" or punctuation
        if (!next.is(ParseTag.POS) && !next.is(PartOfSpeech.PUNCTUATION)) {
            sb.append(" ");
        }
        sb.append(next.getText());
    }
    return sb.toString();
}

@Override
public String getTextRange(int startIndex) {
    return getTextRange(startIndex, getLeaves().size() - 1);
}

protected void setText(String text) {
    this.text = text;
}

```

```

}

public ParseTag getParseTag() {
    return parseTag;
}

public boolean is(ParseTag tag) {
    return getParseTag().equals(tag);
}

@Override
public boolean in(ParseTag... tags) {
    for (ParseTag tag : tags) {
        if (is(tag)) {
            return true;
        }
    }
    return false;
}

@Override
public boolean in(GrammaticalStructureLevel...
grammaticalStructureLevels) {
    for (GrammaticalStructureLevel gsl : grammaticalStructureLevels) {
        if (is(gsl)) {
            return true;
        }
    }
    return false;
}

public void setParseTag(ParseTag parseTag) {
    this.parseTag = parseTag;
    if (parseTag.getGrammaticalStructureLevel() != null) {
        setGrammaticalStructureLevel(parseTag.getGrammaticalStructureLevel());
    }
    if (parseTag.getPartOfSpeech() != null) {
        setPartOfSpeech(parseTag.getPartOfSpeech());
    }
}

public GrammaticalStructureLevel getGrammaticalStructureLevel() {
    return grammaticalStructureLevel;
}

    public boolean is(GrammaticalStructureLevel
grammaticalStructureLevel) {
        return
getGrammaticalStructureLevel().equals(grammaticalStructureLevel);
    }

    protected void setGrammaticalStructureLevel(GrammaticalStructureLevel
grammaticalStructureLevel) {
        this.grammaticalStructureLevel = grammaticalStructureLevel;
    }

    public List<NLPText> getChildren() {
        return children;
    }

    @Override
    public NLPText appendText(List<NLPText> texts) {
        NLPText newText = new NLPTextImpl(null);
        newText.addChild(this);
        if (texts != null) {
            for (NLPText t : texts) {
                newText.addChild(t.clone());
            }
        }
        return newText;
    }

    @Override
    public NLPText appendText(NLPText... texts) {
        return appendText(Arrays.asList(texts));
    }

    @Override
    public void addChild(NLPText child) {
        if (child.getParent() != null) {
            child.getParent().removeChild(child);
        }
        ((NLPTextImpl) child).setParent(this);
        getChildren().add(child);
    }

    public void addRef(NLPText text) {
        getChildren().add(text);
    }

    @Override
    public void removeChild(NLPText child) {

```

```

        getChildren().remove(child);
    }

    public List<NLPText> getLeaves() {
        List<NLPText> leaves = new ArrayList<NLPText>();
        addLeaves(leaves);
        return leaves;
    }

    public boolean isLeaf() {
        return getChildren().size() == 0;
    }

    private void addLeaves(final List<NLPText> leaves) {
        for (NLPText child : getChildren()) {
            if (child.isLeaf()) {
                leaves.add(child);
            } else {
                ((NLPTextImpl) child).addLeaves(leaves);
            }
        }
    }

    @Override
    public Set<GrammaticalRelation> getGrammaticalRelations() {
        return grammaticalRelations;
    }

    @Override
    public void setPrimaryVerb(NLPText primaryVerb) {
        this.primaryVerb = primaryVerb;
    }

    @Override
    public NLPText getPrimaryVerb() {
        return primaryVerb;
    }

    @Override
    public Set<NLPText> getSupportedVerbs() {
        return semanticRoleVerbMap.keySet();
    }

    @Override
    public SemanticRole getSemanticRole(NLPText verb) {
        return semanticRoleVerbMap.get(verb);
    }
}

```

```

    @Override
    public void setSemanticRole(NLPText verb, SemanticRole semanticRole) {
        semanticRoleVerbMap.put(verb, semanticRole);
    }

    public Integer getWordIndex() {
        return wordIndex;
    }

    protected void setWordIndex(Integer wordIndex) {
        this.wordIndex = wordIndex;
    }

    public PartOfSpeech getPartOfSpeech() {
        return partOfSpeech;
    }

    public boolean is(PartOfSpeech partOfSpeech) {
        return getPartOfSpeech().equals(partOfSpeech);
    }

    public void setPartOfSpeech(PartOfSpeech partOfSpeech) {
        this.partOfSpeech = partOfSpeech;
    }

    @Override
    public boolean in(PartOfSpeech... partOfSpeeches) {
        for (PartOfSpeech partOfSpeech : partOfSpeeches) {
            if (is(partOfSpeech)) {
                return true;
            }
        }
        return false;
    }

    @Override
    public Word getDictionaryWord() {
        return dictionaryWord;
    }

    public void setDictionaryWord(Word dictionaryWord) {
        this.dictionaryWord = dictionaryWord;
    }

    @Override

```

```

public Sense getDictionaryWordSense() {
    return dictionaryWordSense;
}

public void setDictionaryWordSense(Sense dictionaryWordSense) {
    this.dictionaryWordSense = dictionaryWordSense;
}

@Override
public Set<SenseRelationInfo> getDictionaryWordSenseRelationInfo() {
    return dictionaryWordSenseRelationInfo;
}

@Override
public void setDictionaryWordSenseRelationInfo(Set<SenseRelationInfo>
relationInfo) {
    this.dictionaryWordSenseRelationInfo =
dictionaryWordSenseRelationInfo;
}

@Override
public boolean isNamedEntity() {
    return isNamedEntity;
}

@Override
public void setNamedEntity(boolean isNamedEntity) {
    this.isNamedEntity = isNamedEntity;
}

public boolean hasText() {
    return (getText() != null) && (getText().length() > 0);
}

public Set<GrammaticalRelation> getGovernorOf() {
    return governs;
}

protected void addGovernorOf(GrammaticalRelation relation) {
    governs.add(relation);
}

public Set<GrammaticalRelation>
getGovernorOfType(GrammaticalRelationType relation) {
    Set<GrammaticalRelation> relations = new
TreeSet<GrammaticalRelation>();
    for (GrammaticalRelation rel : getGovernorOf()) {
        if (rel.getType().isA(relation)) {
            relations.add(rel);
        }
    }
    return relations;
}

@Override
public boolean isGovernorOf(GrammaticalRelationType relationType,
NLPText dependent) {
    for (GrammaticalRelation relation : getGovernorOfType(relationType))
{
        if (dependent.equals(relation.getDependent()) &&
this.equals(relation.getGovernor())) {
            return true;
        }
    }
    return false;
}

public Set<GrammaticalRelation> getDependentOf() {
    return depends;
}

protected void addDependentOf(GrammaticalRelation relation) {
    depends.add(relation);
}

public Set<GrammaticalRelation>
getDependentOfType(GrammaticalRelationType relation) {
    Set<GrammaticalRelation> relations = new
TreeSet<GrammaticalRelation>();
    for (GrammaticalRelation rel : getDependentOf()) {
        if ((rel != null) && (rel.getType() != null) &&
rel.getType().isA(relation)) {
            relations.add(rel);
        }
    }
    return relations;
}

@Override
public boolean isDependentOf(GrammaticalRelationType relationType,
NLPText governor) {
    for (GrammaticalRelation relation :
getDependentOfType(relationType)) {

```

```

    if (governor.equals(relation.getGovernor()) &&
this.equals(relation.getDependent())) {
    return true;
}
return false;
}

public String getLemma() {
    return (lemma == null ? getText() : lemma);
}

public void setLemma(String lemma) {
    this.lemma = lemma;
}

public boolean hasLexicalErrors() {
if (isGrammaticalStructureLevel.WORD) {
    return lexicalErrors;
} else {
    for (NLPText leaf : getLeaves()) {
        return leaf.hasLexicalErrors();
    }
}
return false;
}

public void setLexicalErrors(boolean lexicalErrors) {
    this.lexicalErrors = lexicalErrors;
}

@Override
public NLPText clone() {
try {
    return (NLPText) super.clone();
} catch (CloneNotSupportedException e) {
    return null;
}
}

@Override
public int compareTo(NLPText o) {
    int posCompare = getPartOfSpeech().compareTo(o.getPartOfSpeech());
    int textCompare = getText().compareTo(o.getText());
    return (posCompare != 0 ? posCompare : textCompare);
}

```

```

@Override
public int hashCode() {
final int prime = 31;
int result = 1;
result = prime * result
+ ((grammaticalStructureLevel == null) ? 0 :
grammaticalStructureLevel.hashCode());
result = prime * result + ((partOfSpeech == null) ? 0 :
partOfSpeech.hashCode());
result = prime * result + ((text == null) ? 0 : text.hashCode());
result = prime * result + wordIndex;
return result;
}

@Override
public boolean equals(Object obj) {
if (this == obj) {
    return true;
}
if (obj == null) {
    return false;
}
if (!(obj instanceof NLPText)) {
    return false;
}
final NLPText other = (NLPText) obj;
if (grammaticalStructureLevel == null) {
    if (other.getGrammaticalStructureLevel() != null) {
        return false;
    }
} else if (!
grammaticalStructureLevel.equals(other.getGrammaticalStructureLevel()))
{
    return false;
}
if (partOfSpeech == null) {
    if (other.getPartOfSpeech() != null) {
        return false;
    }
} else if (!partOfSpeech.equals(other.getPartOfSpeech())) {
    return false;
}
if (text == null) {
    if (other.getText() != null) {
        return false;
    }
} else if (!text.equals(other.getText())) {

```

```

        return false;
    }
    if (wordIndex != other.getWordIndex()) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return getText();
}
}

```

nlpTextWalkerFunction.java

```

/*
 * $Id: NLPTextWalkerFunction.java,v 1.1 2009/01/26 10:19:01 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * A visitor for NLPText nodes called at the start and end of visiting
 * each node
 * by an TextWalker NLPTextProcessor.
 *
 * @param <R> -
 *          the return type of the results at the end of processing
 * a node.
 * @author ron
 */
public interface NLPTextWalkerFunction<R> {

    /**
     * This method gets called before the first call to begin() when a
     * text is
     * passed in for processing.
     */
    public void init();

    /**
     * This method gets called on a node when it is first visited.
     */

```

```

        *
        * @param node -
        *          the NLPText node.
        */
    public void begin(NLPText node);

    /**
     * This method gets called on a node after all it's children have
     * been
     * visited.
     *
     * @param node -
     *          the NLPText node.
     * @return
     */
    public R end(NLPText node);
}

```

nosuchactorexception.java

```

/*
 * $Id: NoSuchActorException.java,v 1.3 2008/12/13 00:40:01 rregan Exp
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * @author ron
 */
public class NoSuchActorException extends NoSuchEntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_NAME = "No actor for %s with name
    '%s'";

    /**
     * @param projectOrDomain
     * @param name -
     *          the unknown word text.
     */

```

```

 * @return
 */
public static NoSuchActorException
forProjectOrDomainWithName(ProjectOrDomain projectOrDomain,
    String name) {
    return new NoSuchActorException(Actor.class, null, "name", name,
        EntityExceptionActionType.Reading, MSG_FOR_NAME, projectOrDomain,
        name);
}

/**
 * @param format
 * @param args
 */
protected NoSuchActorException(Class<?> entityType, Object entity,
String entityPropertyName,
    Object entityValue, EntityExceptionActionType actionType, String
format,
    Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected NoSuchActorException(Throwable cause, Class<?> entityType,
Object entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
    messageArgs);
}

```

nosuchannotationexception.java

```

/*
 * $Id: NoSuchAnnotationException.java,v 1.3 2008/12/13 00:41:47
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.requel.annotation;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;

/**
 * @author ron
 */
public class NoSuchAnnotationException extends NoSuchEntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_MESSAGE = "No annotation exists with
message '%s'";
    protected static String MSG_FOR_LEXICAL_ISSUE = "No lexical issue
exists for word '%s'";

    /**
     * @param message -
     *          the message used to search for an issue that didn't
exist
     * @return
     */
    public static NoSuchAnnotationException forMessage(String message) {
        return new NoSuchAnnotationException(Annotation.class, null,
"message", message,
            EntityExceptionActionType.Reading, MSG_FOR_MESSAGE, message);
    }

    /**
     * @param word -
     *          the word used to search for an issue that didn't exist
     * @param annotatablePropertyName -
     *          the property of the entity in question.
     * @return
     */
    public static NoSuchAnnotationException forWord(String word, String
annotatablePropertyName) {
        return new NoSuchAnnotationException(Issue.class, null, "word",
word,
            EntityExceptionActionType.Reading, MSG_FOR_LEXICAL_ISSUE, word,
            annotatablePropertyName);
    }

    /**
     * @param format
     * @param args
     */

```

```

/*
protected NoSuchAnnotationException(Class<?> entityType, Object
entity, String entityPropertyName,
Object entityValue, EntityExceptionActionType actionType, String
format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected NoSuchAnnotationException(Throwable cause, Class<?>
entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

nosuchentityexception.java

```

/*
 * $Id: NoSuchEntityException.java,v 1.2 2008/12/13 00:41:37 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;

/**
 * @author ron
 */
public class NoSuchEntityException extends EntityException {
static final long serialVersionUID = 0;

```

```

protected static String MSG_NO SUCH ENTITY SINGLE PROPERTY SEARCH =
"No %s found for %s of %s";
protected static String MSG_NO SUCH ENTITY MULTIPLE PROPERTY SEARCH =
"No %s found for properties %s with values %s";

public static NoSuchEntityException byQuery(Class<?> entityType,
String entityPropertyName, Object entityPropertyValue) {
return new NoSuchEntityException(entityType, null,
entityPropertyName, entityPropertyValue,
EntityExceptionActionType.Reading,
MSG_NO SUCH ENTITY SINGLE PROPERTY SEARCH,
entityType.getSimpleNome(), entityPropertyName, entityPropertyValue);
}

/**
 *
 * @param entityType
 * @param entityPropertyNames
 * @param entityPropertyValues
 * @return
 */
public static NoSuchEntityException byQuery(Class<?> entityType,
String[] entityPropertyNames, Object[] entityPropertyValues) {
return new NoSuchEntityException(entityType, null,
entityPropertyNames, entityPropertyValues,
EntityExceptionActionType.Reading,
MSG_NO SUCH ENTITY MULTIPLE PROPERTY SEARCH,
entityType.getSimpleNome(), entityPropertyNames,
entityPropertyValues);
}

/**
 * @param format
 * @param args
 */
protected NoSuchEntityException(Class<?> entityType, Object entity,
String entityPropertyName,
Object entityPropertyValue, EntityExceptionActionType actionType,
String format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityPropertyValue,
actionType, format, messageArgs);
}

/**

```

```

 * @param format
 * @param args
 */
protected NoSuchEntityException(Class<?> entityType, Object entity,
String[] entityPropertyNames,
Object[] entityPropertyValues, EntityExceptionActionType
actionType, String format,
Object... messageArgs) {
super(entityType, entity, entityPropertyNames, entityPropertyValues,
actionType, format, messageArgs);
}

protected NoSuchEntityException(Throwable cause, Class<?> entityType,
Object entity,
String entityPropertyName, Object entityPropertyValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName,
entityPropertyValue, actionType, format,
messageArgs);
}

protected NoSuchEntityException(Throwable cause, Class<?> entityType,
Object entity,
String[] entityPropertyNames, Object[] entityPropertyValues,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyNames,
entityPropertyValues, actionType, format,
messageArgs);
}
}

```

nosuchglossarytermexception.java

```

/*
 * $Id: NoSuchGlossaryTermException.java,v 1.3 2008/12/13 00:40:02
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;

```

```

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * @author ron
 */
public class NoSuchGlossaryTermException extends NoSuchEntityException
{
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_NAME = "No glossary term for %s with
name '%s'";

    /**
     * @param projectOrDomain
     * @param name -
     *          the unknown word text.
     * @return
     */
    public static NoSuchGlossaryTermException forProjectOrDomainWithName(
        ProjectOrDomain projectOrDomain, String name) {
        return new NoSuchGlossaryTermException(GlossaryTerm.class, null,
        name, name,
        EntityExceptionActionType.Reading, MSG_FOR_NAME, projectOrDomain,
        name);
    }

    /**
     * @param format
     * @param args
     */
    protected NoSuchGlossaryTermException(Class<?> entityType, Object
entity,
        String entityPropertyName, Object entityValue,
        EntityExceptionActionType actionType,
        String format, Object... messageArgs) {
        super(entityType, entity, entityPropertyName, entityValue,
        actionType, format, messageArgs);
    }

    /**
     * @param cause
     * @param format
     * @param args
     */
    protected NoSuchGlossaryTermException(Throwable cause, Class<?>
entityType, Object entity,

```

```

        String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
        String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
        messageArgs);
}
}

```

nosuchorganizationexception.java

```

/*
 * $Id: NoSuchOrganizationException.java,v 1.3 2008/12/13 00:41:36
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.user.Organization;

/**
 * @author ron
 */
public class NoSuchOrganizationException extends NoSuchEntityException
{
    static final long serialVersionUID = 0;

    protected static String MSG_NO_ORGANIZATION_FOR_NAME = "No
organization for name '%s'";

    /**
     * @param username
     * @return
     */
    public static NoSuchOrganizationException forName(String name) {
        return new NoSuchOrganizationException(Organization.class, null,
"name", name,
        EntityExceptionActionType.Reading, MSG_NO_ORGANIZATION_FOR_NAME,
name);
    }

    protected NoSuchOrganizationException(Class<?> entityType, Object
entity, String entityPropertyName,

```

```

        Object entityValue, EntityExceptionActionType actionType, String
format,
        Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

protected NoSuchOrganizationException(Throwable cause, Class<?>
entityType, Object entity,
        String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
        String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
        messageArgs);
}
}

```

nosuchpositionexception.java

```

/*
 * $Id: NoSuchPositionException.java,v 1.11 2009/01/09 09:56:16 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;

/**
 * @author ron
 */
public class NoSuchPositionException extends NoSuchEntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_TEXT = "No position exists with text
'%s'";
    protected static String MSG_FOR_ADDING_WORD_TO_DICTIONARY = "No 'add
word to dictionary' position exists for word '%s'";
    protected static String MSG_FOR_ADDING_WORD_TO_GLOSSARY = "No 'add
word to glossary of %s' position exists for word '%s'";
}

```

```

protected static String MSG_FOR_ADDING_ACTOR_TO_PROJECT = "No 'add
actor to project %s' position exists for name '%s'";
protected static String MSG_FOR_CHANGING SPELLING = "No 'change
spelling' position exists for unknown word '%s' and proposed word
'%s'";

/**
 * @param text -
 *          the text used to search for a position that didn't
exist
 * @return
 */
public static NoSuchPositionException forText(String text) {
    return new NoSuchPositionException(Position.class, null, "text",
text,
    EntityExceptionActionType.Reading, MSG_FOR_TEXT, text);
}

/**
 * @param word -
 *          the word used to search for a position that didn't
exist
 * @return
 */
public static NoSuchPositionException
forAddingWordToDictionary(String word) {
    return new NoSuchPositionException(Position.class, null, "word",
word,
    EntityExceptionActionType.Reading,
MSG_FOR_ADDING_WORD_TO_DICTIONARY, word);
}

/**
 * TODO: this violates project ref in annotation package
 *
 * @param pod -
 *          the project or domain
 * @param word -
 *          the word used to search for a position that didn't
exist
 * @return
 */
public static NoSuchPositionException
forAddingWordToGlossary(ProjectOrDomain pod, String word) {
    return new NoSuchPositionException(Position.class, null, "word",
word,
}

```

```

    EntityExceptionActionType.Reading,
MSG_FOR_ADDING_WORD_TO_GLOSSARY, pod.getName(),
word);
}

/**
 * @param pod
 * @param actorName
 * @return
 */
public static NoSuchPositionException
forAddingActorToProject(ProjectOrDomain pod,
String actorName) {
    return new NoSuchPositionException(Position.class, null,
"actorName", actorName,
    EntityExceptionActionType.Reading,
MSG_FOR_ADDING_ACTOR_TO_PROJECT, pod.getName(),
actorName);
}

/**
 * @param issue
 * @param misspelledWord
 * @param proposedWord
 * @return
 */
public static NoSuchPositionException forChangeSpelling(LexicalIssue
issue, String proposedWord) {
    return new NoSuchPositionException(Position.class, null, null, null,
    EntityExceptionActionType.Reading, MSG_FOR_CHANGING_SPELLING,
issue.getWord(),
proposedWord);
}

/**
 * @param format
 * @param args
 */
protected NoSuchPositionException(Class<?> entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**

```

```

 * @param cause
 * @param format
 * @param args
 */
protected NoSuchPositionException(Throwable cause, Class<?>
entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

nosuchprojectexception.java

```

/*
 * $Id: NoSuchProjectException.java,v 1.4 2008/12/13 00:40:02 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.project.Project;

/**
 * @author ron
 */
public class NoSuchProjectException extends NoSuchEntityException {
static final long serialVersionUID = 0;

protected static String MSG_FOR_NAME = "No project exists for name
'%s'";

/**
 * @param name -
 *          the name used to find a project that doesn't exist
 * @return
 */
public static NoSuchProjectException forName(String name) {
return new NoSuchProjectException(Project.class, null, "name", name,
EntityExceptionActionType.Reading, MSG_FOR_NAME, name);
}
}

```

```

}

/**
 * @param format
 * @param args
 */
protected NoSuchProjectException(Class<?> entityType, Object entity,
String entityPropertyName,
Object entityValue, EntityExceptionActionType actionType, String
format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected NoSuchProjectException(Throwable cause, Class<?>
entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

nosuchroleforuserexception.java

```

/*
 * $Id: NoSuchRoleForUserException.java,v 1.5 2008/12/13 00:41:36
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user UserRole;

/**
 */

```

```

* @author ron
*/
public class NoSuchRoleForUserException extends NoSuchEntityException
{
    static final long serialVersionUID = 0;

    protected static String MSG_NO_ROLE_FOR_USER = "The user '%s' does
not have the role '%s'";

    /**
     * @param user -
     *          the user that doesn't have the role.
     * @param userRoleType -
     *          the type of the role that the user doesn't have.
     * @return
     */
    public static NoSuchRoleForUserException forUserRoleTypeName(User
user,
        Class<? extends UserRole> userRoleType) {
        return new NoSuchRoleForUserException(User.class, user, "userRoles",
userRoleType,
            EntityExceptionActionType.Reading, MSG_NO_ROLE_FOR_USER,
user.getUsername(),
            userRoleType.getSimpleName());
    }

    protected NoSuchRoleForUserException(Class<?> entityType, Object
entity,
        String entityPropertyName, Object entityValue,
        EntityExceptionActionType actionType,
        String format, Object... messageArgs) {
        super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
    }

    protected NoSuchRoleForUserException(Throwable cause, Class<?>
entityType, Object entity,
        String entityPropertyName, Object entityValue,
        EntityExceptionActionType actionType,
        String format, Object... messageArgs) {
        super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
            messageArgs);
    }
}

```

nosuchuserexception.java

```

/*
 * $Id: NoSuchUserException.java,v 1.4 2008/12/13 00:41:36 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public class NoSuchUserException extends NoSuchEntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_NO_USER_FOR_NAME = "No user for username
'%s'";
    protected static String MSG_WRONG_PASSWORD = "The password for
username '%s' was incorrect.";

    /**
     * @param username
     * @return
     */
    public static NoSuchUserException forUsername(String username) {
        return new NoSuchUserException(User.class, null, "username",
username,
            EntityExceptionActionType.Reading, MSG_NO_USER_FOR_NAME,
username);
    }

    /**
     * @param user
     * @return
     */
    public static NoSuchUserException wrongPasswordForUser(User user) {
        return new NoSuchUserException(User.class, user, "password", null,
            EntityExceptionActionType.Reading, MSG_WRONG_PASSWORD,
user.getUsername());
    }
}

```

```

protected NoSuchUserException(Class<?> entityType, Object entity,
String entityPropertyName,
Object entityValue, EntityExceptionActionType actionType, String
format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

protected NoSuchUserException(Throwable cause, Class<?> entityType,
Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

nosuchwordexception.java

```

/*
 * $Id: NoSuchWordException.java,v 1.1 2008/12/15 06:36:02 rregan Exp
$Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.repository;

import edu.harvard.fas.rregan.nlp.dictionary.Word;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.request.NoSuchEntityException;

/**
 * @author ron
 */
public class NoSuchWordException extends NoSuchEntityException {
static final long serialVersionUID = 0;

protected static String MSG_FOR_TEXT = "Unknown word '%s'";
protected static String MSG_FOR_TEXT_AND_POS = "Unknown word '%s'
with part of speech '%s'";

/**
 * @param text -

```

```

*           the unknown word text.
* @return
*/
public static NoSuchWordException forLemma(String text) {
return new NoSuchWordException(Word.class, null, "text", text,
EntityExceptionActionType.Reading, MSG_FOR_TEXT, text);
}

/**
* @param text -
*           the unknown word text.
* @param pos -
*           the part of speech.
* @return
*/
public static NoSuchWordException forLemmaAndPOS(String text, String
pos) {
return new NoSuchWordException(Word.class, null, "text", text,
EntityExceptionActionType.Reading, MSG_FOR_TEXT, text, pos);
}

/**
* @param format
* @param args
*/
protected NoSuchWordException(Class<?> entityType, Object entity,
String entityPropertyName,
Object entityValue, EntityExceptionActionType actionType, String
format,
Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
* @param cause
* @param format
* @param args
*/
protected NoSuchWordException(Throwable cause, Class<?> entityType,
Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}

```

```
}
```

note.java

```
/*
 * $Id: Note.java,v 1.3 2008/04/18 22:41:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

/**
 * A note is an informational annotation without a resolution
 * or positions.
 *
 * @author ron
 */
public interface Note extends Annotation {
```

```
}
```

noteeditorpanel.java

```
/*
 * $Id: NoteEditorPanel.java,v 1.24 2009/03/23 11:02:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.MessageFormat;

import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Note;
```

```
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteNoteCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.EditNoteCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.UpdateEntityEvent;
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent;
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * @author ron
 */
public class NoteEditorPanel extends
AbstractRequelAnnotationEditorPanel {
private static final Logger log =
Logger.getLogger(NoteEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the NoteEditorPanel.properties file to set the
label
 * of the note text field. If the property is undefined "Note" is
used.
 */
public static final String PROP_LABEL_TEXT = "Text.Label";

private final AnnotationCommandFactory annotationCommandFactory;
private UpdateListener updateListener;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted = false;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public NoteEditorPanel(CommandHandler commandHandler,
AnnotationCommandFactory annotationCommandFactory,
```

```

    AnnotationRepository annotationRepository) {
this(NoteEditorPanel.class.getName(), commandHandler,
annotationCommandFactory,
    annotationRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public NoteEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository annotationRepository) {
super(resourceBundleName, Note.class, commandHandler,
annotationRepository);
this.annotationCommandFactory = annotationCommandFactory;
}

/**
 * If the editor is editing an existing note the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Edit Note"<br>
 * For a new note it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Note"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
if (getNote() != null) {
String msgPattern = getResourceBundleHelper(getLocale()).getString(
PROP_EXISTING_OBJECT_PANEL_TITLE,
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Edit Note"));
}

```

```

    return MessageFormat.format(msgPattern, getNote().toString());
} else {
String msg = getResourceBundleHelper(getLocale()).getString(
PROP_NEW_OBJECT_PANEL_TITLE,
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"New Note"));
return msg;
}

@Override
public void setup() {
super.setup();
Note note = getNote();
if (note != null) {
addInput("text", PROP_LABEL_TEXT, "Note", new TextArea(), new
StringDocumentEx(note
.getText()));
addMultiRowInput("annotatables",
AnnotationRefererTable.PROP_ANNOTATABLES_LABEL,
"Referring Entities", new AnnotationRefererTable(this,
getResourceBundleHelper(getLocale()), note));
} else {
addInput("text", PROP_LABEL_TEXT, "Note", new TextArea(), new
StringDocumentEx(""));
addMultiRowInput("annotatables",
AnnotationRefererTable.PROP_ANNOTATABLES_LABEL,
"Referring Entities", new AnnotationRefererTable(this,
getResourceBundleHelper(getLocale()), null));
}

if (updateListener != null) {
getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
super.dispose();
removeAll();
if (updateListener != null) {

```

```

        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditNoteCommand command =
getAnnotationCommandFactory().newEditNoteCommand();
        command.setGroupingObject(getGroupingObject());
        command.setNote(getNote());
        command.setAnnotatable(getAnnotatable());
        command.setEditedBy(getCurrentUser());
        command.setText(getInputValue("text", String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
            updateListener);
        }
        Note note = command.getNote();
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
note));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        } else if ((e.getCause() != null) && (e.getCause() instanceof
InternalServerError)) {
            InvalidStateException ise = (InternalServerError) e.getCause();
            for (InvalidValue invalidValue : ise.getInvalidValues()) {
                String propertyName = invalidValue.getPropertyName();
                setValidationMessage(propertyName, invalidValue.getMessage());
            }
        } else {
            setGeneralMessage(e.toString());
        }
    } catch (Exception e) {
        log.error("could not save the goal: " + e, e);
        setGeneralMessage("Could not save: " + e);
    }
}

@Override
public void delete() {
    try {
        Note note = getNote();
        DeleteNoteCommand deleteNoteCommand = getAnnotationCommandFactory()
            .newDeleteNoteCommand();
        deleteNoteCommand.setNote(note);
        deleteNoteCommand.setEditedBy(getCurrentUser());
        deleteNoteCommand = getCommandHandler().execute(deleteNoteCommand);
        deleted = true;
        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
note));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private Note getNote() {
    if (getTargetObject() instanceof Note) {
        return (Note) getTargetObject();
    }
    return null;
}

private AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

// TODO: it may be better to have different standardized update
listeners
// for different types of updated or deleted objects associated with
the
// input controls like an annotatables table.
private static class UpdateListener implements ActionListener {

```

```

static final long serialVersionUID = 0L;
private final NoteEditorPanel panel;
private UpdateListener(NoteEditorPanel panel) {
    this.panel = panel;
}
public void actionPerformed(ActionEvent e) {
    if (panel.deleted) {
        return;
    }
    Note existingNote = panel.getNote();
    if ((e instanceof UpdateEntityEvent) && (existingNote != null)) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        Note updatedNote = null;
        if (event.getObject() instanceof Note) {
            updatedNote = (Note) event.getObject();
            if ((event instanceof DeletedEntityEvent) &&
existingNote.equals(updatedNote)) {
                panel.deleted = true;
                panel.getEventDispatcher().dispatchEvent(
                    new DeletedEntityEvent(this, panel, existingNote));
            }
        } else if ((event instanceof DeletedEntityEvent)
&& (event.getObject() instanceof Annotatable)) {
            Annotatable annotatable = (Annotatable) event.getObject();
            if (existingNote.getAnnotatables().contains(annotatable)) {
                existingNote.getAnnotatables().remove(annotatable);
            }
            updatedNote = existingNote;
        }
        if ((updatedNote != null) && updatedNote.equals(existingNote)) {
            panel.setInputValue("text", updatedNote.getText());
            panel.setInputValue("annotatables", updatedNote);
            panel.setTargetObject(updatedNote);
        }
    }
}

```

noteimpl.java

```

/*
 * $Id: NoteImpl.java,v 1.10 2009/02/15 09:31:37 rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import edu.harvard.fas.rregan.requel.annotation.Note;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.Note")
@XmlRootElement(name = "note", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "note", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class NoteImpl extends AbstractAnnotation implements Note {
    static final long serialVersionUID = 0L;

    /**
     * @param groupingObject -
     *          An object used as the "owner" of a group of
     * annotations.
     * @param text -
     *          the text of the note.
     * @param createdBy -
     *          the user that added the note to the annotatable object.
     */
    public NoteImpl(Object groupingObject, String text, User createdBy) {
        super(Note.class.getName(), groupingObject, text, createdBy);
    }

    protected NoteImpl() {
        // for hibernate
    }

    @Transient
    public String getStatusMessage() {
        return "Informational";
    }
}
```

```

@Transient
public String getTypeName() {
    return "Note";
}

@Transient
public boolean isMustBeResolved() {
    return false;
}

@Transient
public boolean isResolved() {
    return true;
}

```

nounphrasefinder.java

```

/*
 * $Id: NounPhraseFinder.java,v 1.2 2009/02/09 10:12:28 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.Collection;

import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * Return a collection of NLPText that are all the noun phrases in the
given
 * NLPText.
 *
 * @author ron
 */
@Component("nounPhraseFinder")
public class NounPhraseFinder implements
NLPProcessor<Collection<NLPText>> {

```

```

    public NounPhraseFinder() {
    }

    /**
     * if the text is a SENTENCE, CLAUSE or PHRASE, return all the
subordinate
     * noun phrases.
     */
    @Override
    public Collection<NLPText> process(NLPText text) {
        Collection<NLPText> nounPhrases = new ArrayList<NLPText>();

        if (!text.is(GrammaticalStructureLevel.WORD)) {
            for (NLPText child : text.getChildren()) {
                nounPhrases.addAll(process(child));
            }
        }

        if (text.is(GrammaticalStructureLevel.PHRASE) &&
ParseTag.NP.equals(text.getParseTag())) {
            nounPhrases.add(text);
        }
        return nounPhrases;
    }
}
```

nounphrasematchingrule.java

```

/*
 * $Id: NounPhraseMatchingRule.java,v 1.7 2009/02/11 09:02:55 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;

```

```

import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.dictionaryRepository.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;
import
edu.harvard.fas.rregan.nlp.dictionary.VerbalSelectionRestriction;
import edu.harvard.fas.rregan.nlp.impl.NLPTextImpl;

/**
 * @author ron
 */
public class NounPhraseMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
Logger.getLogger(NounPhraseMatchingRule.class);

    private final VerbNetRoleRef roleRef;
    private final SemanticRole semanticRole;

    /**
     * Create a new rule for matching a noun phrase that fills the
     * specified
     * thematic role.
     *
     * @param roleRef -
     *         a reference to the thematic role including selection
     *         restrictions.
     */
    public NounPhraseMatchingRule(VerbNetRoleRef roleRef) {
        this.roleRef = roleRef;
        semanticRole = roleRef.getVerbNetRole().getSemanticRole();
    }

    /**
     *
     */
    @Override
    public void match(DictionaryRepository dictionaryRepository, NLPText
verb,
        ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException {
        NLPText word = textIterator.next();
        // TODO: syntax restrictions? some roles may be filled by full
clauses
        // that may not be the immediate parent
        if (word.getParent().is(ParseTag.NP) ||
semanticRole.equals(SemanticRole.TOPIC)) {
            // find the noun in this phrase that is the subject/object of the

```

```

// main verb or a modifier of the main verb and see if it meets
// the restriction.
if (roleRef.getSelectionalRestrictions().size() > 0) {
    NLPText mainNoun = getMainNoun(word.getParent(), verb);
    if (mainNoun.getDictionaryWordSense() != null) {
        // TODO: this assumes the restrictions are or-ed together
        for (VerbalSelectionRestriction res :
roleRef.getSelectionalRestrictions()) {
            if (meetsRestriction(dictionaryRepository, res, mainNoun)) {
                word.getParent().setSemanticRole(verb, semanticRole);
                consumeWordsToEndOfNode(textIterator, word.getParent());
                return;
            }
        }
        throw
SemanticRoleLabelerException.matchFailedBySelectionalRestriction(this,
mainNoun);
    }
    throw SemanticRoleLabelerException.matchFailed(this, word);
}
word.getParent().setSemanticRole(verb, semanticRole);
consumeWordsToEndOfNode(textIterator, word.getParent());
return;
}
throw SemanticRoleLabelerException.matchFailed(this, word);
}

private NLPText getMainNoun(NLPText phrase, NLPText verb) {
for (NLPText word : phrase.getLeaves()) {
    if (word.is(PartOfSpeech.NOUN)) {
        for (GrammaticalRelation rel : word.getDependentOf()) {
            if (hasRelationToVerb(rel, verb)) {
                log.debug("main noun: " + word);
                return word;
            }
        }
    }
}
throw SemanticRoleLabelerException.matchFailed(this, phrase);
}

private void consumeWordsToEndOfNode(ListIterator<NLPText>
textIterator, NLPText node) {
    NLPText matchedWords = new NLPTextImpl();
    while (textIterator.hasNext()) {
        NLPText word = textIterator.next();
        if (!word.isDescendentOf(node)) {

```

```

textIterator.previous();
log.debug("matched: " + matchedWords.getText());
return;
}
matchedWords.addRef(word);
}

private boolean meetsRestriction(DictionaryRepository
dictionaryRepository,
VerbNetSelectionRestriction restriction, NLPText word) {
log.debug("word: " + word + " restriction " +
restriction.getInclude()
+ restriction.getType().getName());
Synset restrictionSynset = restriction.getType().getSynset();
log.debug("restriction synset: " + restrictionSynset);
Synset wordSynset = word.getDictionaryWordSense().getSynset();
log.debug("word synset: " + wordSynset);
boolean isHyponym =
dictionaryRepository.isHyponym(restrictionSynset, wordSynset);
if ("+".equals(restriction.getInclude()) && isHyponym)
|| ("-".equals(restriction.getInclude()) && !isHyponym)) {
return true;
}
return false;
}

private boolean hasRelationToVerb(GrammaticalRelation rel, NLPText
verb) {
GrammaticalRelationType relType = rel.getType();
// the object/subject of the verb
if (rel.getGovernor().equals(verb)
&& (relType.isA(GrammaticalRelationType.OBJECT) || relType
.isA(GrammaticalRelationType.SUBJECT))) {
return true;
}
// object/subject of a preposition that modifies the verb
if (rel.getGovernor().is(PartOfSpeech.PREPOSITION)
&& (relType.isA(GrammaticalRelationType.OBJECT) || relType
.isA(GrammaticalRelationType.SUBJECT))) {
for (GrammaticalRelation rel2 : rel.getGovernor().getDependentOf())
{
if
( rel2.getType().isA(GrammaticalRelationType.PREPOSITIONAL_MODIFIER)
&& rel2.getGovernor().equals(verb)) {
return true;
}
}
}

// adjust for bad prepositional phrase attachment
// object/subject of a preposition that is a modifier of a
// subject/object of the verb
if (rel.getGovernor().is(PartOfSpeech.PREPOSITION)) {
for (GrammaticalRelation rel2 : rel.getGovernor().getDependentOf())
{
if
( rel2.getType().isA(GrammaticalRelationType.PREPOSITIONAL_MODIFIER)) {
// modifies an object/subject of the verb
NLPText thing = rel2.getGovernor();
for (GrammaticalRelation rel3 : verb.getGovernorOf()) {
log.debug(rel3 + " -> " + thing);
relType = rel3.getType();
if (rel3.getDependent().equals(thing)
&& (relType.isA(GrammaticalRelationType.OBJECT) || relType
.isA(GrammaticalRelationType.SUBJECT))) {
return true;
}
}
}
}
return false;
}

@Override
public String toString() {
return getClass().getSimpleName() + ":" + semanticRole + " "
+ roleRef.getSelectionalRestrictions();
}
}



## nullcomponentmanipulator.java


/*
 * $Id: NullComponentManipulator.java,v 1.2 2008/10/11 21:47:44 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

```

nullcomponentmanipulator.java

```
/*
 * $Id: NullComponentManipulator.java,v 1.2 2008/10/11 21:47:44 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
```

```

/**
 * @author ron
 */
public class NullComponentManipulator extends
AbstractComponentManipulator {

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
}

@Override
public Object getModel(Component component) {
    return null;
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return null;
}

@Override
public void setModel(Component component, Object model) {
}

@Override
public void setValue(Component component, Object value) {
}

}

```

oddeventablecellrenderer.java

```

/*
 * $Id: OddEvenTableCellRenderer.java,v 1.2 2008/03/28 11:02:34 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.table;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.Table;
import nextapp.echo2.app.table.DefaultTableCellRenderer;

```

```

/**
 * Render table cell components wrapped in Row components with
alternating odd
 * and even styling.
 *
 * @author ron
 */
public class OddEvenTableCellRenderer extends DefaultTableCellRenderer
{
    static final long serialVersionUID = 0;

    private final String evenCellStyleName;
    private final String oddCellStyleName;

    /**
     * @param oddCellStyleName
     * @param evenCellStyleName
     */
    public OddEvenTableCellRenderer(String oddCellStyleName, String
evenCellStyleName) {
        this.oddCellStyleName = oddCellStyleName;
        this.evenCellStyleName = evenCellStyleName;
    }

    @Override
    public Component getTableCellRendererComponent(Table table, Object
value, int column, int row) {
        Component component = getRendererComponent(value);
        Row wrapper = new Row();
        if (row % 2 == 0) {
            wrapper.setStyleName(evenCellStyleName);
        } else {
            wrapper.setStyleName(oddCellStyleName);
        }
        wrapper.add(component);
        return wrapper;
    }

    protected Component getRendererComponent(Object value) {
        Component component;
        if (value instanceof Component) {
            component = (Component) value;
        } else {
            component = new Label((value == null ? "" : value.toString()));
        }
        return component;
    }
}

```

```
}
```

opennlpparser.java

```
/*
 * $Id: OpenNLPParser.java,v 1.3 2009/04/01 09:47:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.net.URLDecoder;
import java.util.List;
import java.util.regex.Pattern;

import opennlp.maxent.MaxentModel;
import opennlp.tools.chunker.ChunkerME;
import opennlp.tools.lang.english.HeadRules;
import opennlp.tools.parser.Parse;
import opennlp.tools.parser.ParserChunker;
import opennlp.tools.parser.ParserME;
import opennlp.tools.util.Sequence;
import opennlp.tools.util.Span;
import edu.harvard.fas.rregan.ApplicationException;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * @author ron
 */
// @Component("openNLPParser")
public class OpenNLPParser extends OpenNLPTagger {

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the open nlp parser head rules file relative to the
     * classpath.
     *
     * @see PROP_PARSER_HEAD_RULES_FILE_DEFAULT for the default location
     * of the
     * file
     */
}
```

```
    public static final String PROP_PARSER_HEAD_RULES_FILE =
"HeadRulesFile";

    /**
     * The default path to the parser head rules file
     */
    public static final String PROP_PARSER_HEAD_RULES_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/head_rules";

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the open nlp parser chunker model file relative to the
     * classpath.
     *
     * @see PROP_PARSER_CHUNKER_MODEL_FILE_DEFAULT for the default
     * location of
     * the file
     */
    public static final String PROP_PARSER_CHUNKER_MODEL_FILE =
"ParserChunkerModelFile";

    /**
     * The default path to the chunker model file
     */
    public static final String PROP_PARSER_CHUNKER_MODEL_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/chunk.bin.gz";

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the open nlp parser build model file relative to the
     * classpath.
     *
     * @see PROP_PARSER_BUILD_MODEL_FILE_DEFAULT for the default location
     * of the
     * file
     */
    public static final String PROP_PARSER_BUILD_MODEL_FILE =
"ParserBuildModelFile";

    /**
     * The default path to the parser build model file
     */
    public static final String PROP_PARSER_BUILD_MODEL_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/build.bin.gz";
```

```

/**
 * The name of the property in the NLPImpl.properties file that
contains the
 * path to the open nlp parser build model file relative to the
classpath.
 *
 * @see PROP_PARSER_CHECK_MODEL_FILE_DEFAULT for the default location
of the
 *      file
 */
public static final String PROP_PARSER_CHECK_MODEL_FILE =
"ParserCheckModelFile";

/**
 * The default path to the parser check model file
 */
public static final String PROP_PARSER_CHECK_MODEL_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/check.bin.gz";

private static Pattern untokenizedParenPattern1 =
Pattern.compile("[^ ]([{{}}])");
private static Pattern untokenizedParenPattern2 =
Pattern.compile("[{{}}]([^{ }])";

private static ParserME parser;

/**
 * create a new parser, initializing if needed.
 *
 * @throws ApplicationException
 *          if the underlying OpenNLP parser fails to initialize
 */
public OpenNLPParser() {
    init();
}

private synchronized void init() {
    if (parser == null) {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                OpenNLPParser.class.getName());

            String headRulesFile =
resourceBundleHelper.getString(PROP_PARSER_HEAD_RULES_FILE,
                PROP_PARSER_HEAD_RULES_FILE_DEFAULT);

```

```

String headRulesPath =
URLDecoder.decode(OpenNLPParser.class.getClassLoader()
    .getResource(headRulesFile).getPath(), "utf8");
HeadRules headRules = new HeadRules(headRulesPath);

String chunkerModelFile = resourceBundleHelper.getString(
    PROP_PARSER_CHUNKER_MODEL_FILE,
PROP_PARSER_CHUNKER_MODEL_FILE_DEFAULT);
ParserChunker chunker = new
Chunker(readGISModel(chunkerModelFile));

String buildModelFile = resourceBundleHelper.getString(
    PROP_PARSER_BUILD_MODEL_FILE,
PROP_PARSER_BUILD_MODEL_FILE_DEFAULT);

String checkModelFile = resourceBundleHelper.getString(
    PROP_PARSER_CHECK_MODEL_FILE,
PROP_PARSER_CHECK_MODEL_FILE_DEFAULT);

parser = new ParserME(readGISModel(buildModelFile),
readGISModel(checkModelFile),
    getTagger(), chunker, headRules);
} catch (Exception e) {
    parser = null;
    throw ApplicationException.failedToInitializeComponent(getClass(),
e);
}
}

/**
 * @see
edu.harvard.fas.rregan.nlp.NLPPProcessor#process(edu.harvard.fas.rregan
.nlp.NLPText)
 */
@Override
public NLPText process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.SENTENCE)) {
        Parse parse = null;
        Span[] spans = null;
        String sentence = text.getText();
        if (text.getLeaves().isEmpty()) {
            sentence =
untokenizedParenPattern1.matcher(sentence).replaceAll("$1 $2");
            sentence =
untokenizedParenPattern2.matcher(sentence).replaceAll("$1 $2");
            spans = getTokenizer(). tokenizePos(sentence);

```

```

} else {
    spans = new Span[text.getLeaves().size()];
    int start = 0;
    int end = 0;
    for (NLPText word : text.getLeaves()) {
        start = sentence.indexOf(word.getText(), start);
        end = start + word.getText().length();
        spans[word.getWordIndex()] = new Span(start, end);
    }
}
Parse topNode = new Parse(sentence, new Span(0, sentence.length()),
"INC", 1, null);
for (int i = 0; i < spans.length; i++) {
    topNode.insert(new Parse(sentence, spans[i], ParserME.TOK_NODE,
0));
}
parse = parser.parse(topNode);
if (parse != null) {
    copyOpenNLPParseToNLPText((NLPTextImpl) text, parse, new
Counter());
}
}
return text;
}

private static void copyOpenNLPParseToNLPText(NLPTextImpl parent,
Parse parse,
Counter wordIndexCounter) {
    ParseTag tag = ParseTag.tagOf(parse.getType());
    NLPTextImpl node;
    if (parse.isPosTag() && (parse.getChildCount() == 1)) {
        // word level tags
        String word = parse.getSpan().toString();
        node = new NLPTextImpl(parent, wordIndexCounter.getCount(), word,
tag);
        parent.getChildren().add(node);
        wordIndexCounter.incr();
    } else {
        if (ParserME.TOP_NODE.equals(parse.getType())) {
            // skip the root
            parent.setParseTag(ParseTag.ROOT);
            node = parent;
        } else {
            // phrase and clause level tags
            node = new NLPTextImpl(parent, tag);
        }
    }
}

```

```

        parent.getChildren().add(node);
    }
    for (Parse child : parse.getChildren()) {
        copyOpenNLPParseToNLPText(node, child, wordIndexCounter);
    }
}

private static class Counter {
    private int count = 0;

    protected void incr() {
        count++;
    }

    protected int getCount() {
        return count;
    }
}

private static class Chunker extends ChunkerME implements
ParserChunker {

    private Chunker(MaxentModel mod) {
        super(mod);
    }

    public Sequence[] topKSequences(List sentence, List tags) {
        return beam.bestSequences(10, sentence.toArray(), new Object[]
{ tags });
    }

    public Sequence[] topKSequences(String[] sentence, String[] tags,
double minSequenceScore) {
        return beam.bestSequences(10, sentence, new Object[] { tags },
minSequenceScore);
    }
}

```

opennlptagger.java

```

/*
 * $Id: OpenNLPTagger.java,v 1.4 2009/04/01 09:47:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.nlp.impl;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import opennlp.maxent.MaxentModel;
import opennlp.tools.parser.ParserTagger;
import opennlp.tools.postag.DefaultPOSContextGenerator;
import opennlp.tools.postag.POSContextGenerator;
import opennlp.tools.postag.POSDictionary;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.postag.TagDictionary;
import opennlp.tools.util.Sequence;
import edu.harvard.fas.rregan.ApplicationException;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * @author ron
 */
// @Component("openNLPTagger")
public class OpenNLPTagger extends OpenNLPTokenizer {

/**
 * The name of the property in the NLPImpl.properties file that
contains the
 * path to the open nlp part-of-speech tagger tag dictionary relative
to the
 * classpath.
 *
 * @see PROP_POSTAGGER_DICTIONARY_FILE_DEFAULT for the default
location of
 *      the file
 */
public static final String PROP_POSTAGGER_DICTIONARY_FILE =
"POSTaggerDictionaryFile";

/**
 * The default path to the POS tagger dictionary file
 */

```

```

    public static final String PROP_POSTAGGER_DICTIONARY_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/tagdict";

    /**
     * The name of the property in the NLPImpl.properties file that
contains the
     * path to the open nlp part-of-speech tagger model file relative to
the
     * classpath.
     *
     * @see PROP_POSTAGGER_MODEL_FILE_DEFAULT for the default location of
the
     *      file.
     */
    public static final String PROP_POSTAGGER_MODEL_FILE =
"POSTaggerModelFile";

    /**
     * The default path to the POS tagger model file.
     */
    public static final String PROP_POSTAGGER_MODEL_FILE_DEFAULT =
"resources/nlp/opennlp-tools/parser/tag.bin.gz";

    private static Tagger postTagger;
    static {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                OpenNLPParser.class.getName());

            String modelFile =
resourceBundleHelper.getString(PROP_POSTAGGER_MODEL_FILE,
                PROP_POSTAGGER_MODEL_FILE_DEFAULT);

            String dictFile =
resourceBundleHelper.getString(PROP_POSTAGGER_DICTIONARY_FILE,
                PROP_POSTAGGER_DICTIONARY_FILE_DEFAULT);
            InputStream dictInputStream =
OpenNLPTagger.class.getClassLoader().getResourceAsStream(
                dictFile);
            TagDictionary dict = new POSDictionary(new BufferedReader(new
InputStreamReader(
                dictInputStream)), true);

            postTagger = new Tagger(readGISModel(modelFile), new
DefaultPOSContextGenerator(null),
                dict);
        }
    }

```

```

} catch (Exception e) {
    posTagger = null;
    throw new ExceptionInInitializerError(e);
}
}

/***
 * create a new POS tagger, initializing if needed.
 *
 * @throws ApplicationException
 *          if the underlying OpenNLP tagger fails to initialize
 */
public OpenNLPTagger() {
    init();
}

private synchronized void init() {
    if (postagger == null) {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                OpenNLPParser.class.getName());

            String modelFile =
resourceBundleHelper.getString(PROP_POSTAGGER_MODEL_FILE,
                PROP_POSTAGGER_MODEL_FILE_DEFAULT);

            String dictFile =
resourceBundleHelper.getString(PROP_POSTAGGER_DICTIONARY_FILE,
                PROP_POSTAGGER_DICTIONARY_FILE_DEFAULT);
            InputStream dictInputStream = OpenNLPTagger.class.getClassLoader()
                .getResourceAsStream(dictFile);
            TagDictionary dict = new POSDictionary(new BufferedReader(new
InputStreamReader(
                dictInputStream)), true);

            postagger = new Tagger(readGISModel(modelFile),
                new DefaultPOSContextGenerator(null), dict);
        } catch (Exception e) {
            postagger = null;
            throw ApplicationException.failedToInitializeComponent(getClass(),
e);
        }
    }
}

/***
 * @see
edu.harvard.fas.rregan.nlp.NLPPProcessor#process(edu.harvard.fas.rregan
.nlp.NLPText)
 */
@Override
public NLPText process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.SENTENCE)) {
        if (text.getLeaves().isEmpty()) {
            super.process(text);
        }
        List<String> words = new ArrayList<String>();
        for (NLPText word : text.getLeaves()) {
            words.add(word.getText());
        }
        List<String> tags = postagger.tag(words);
        for (int i = 0; i < tags.size(); i++) {
            ((NLPTextImpl)
text.getLeaves().get(i)).setParseTag(ParseTag.tagOf(tags.get(i)));
        }
    }
    return text;
}

protected static Tagger getTagger() {
    return postagger;
}

protected static class Tagger extends POSTaggerME implements
ParserTagger {
    private Tagger(MaxentModel mod, POSContextGenerator cg,
TagDictionary dict) {
        super(mod, cg, dict);
    }

    public Sequence[] topKSequences(List sentence) {
        return beam.bestSequences(10, sentence.toArray(), null);
    }

    public Sequence[] topKSequences(String[] sentence) {
        return beam.bestSequences(10, sentence, null);
    }
}
}

```

opennlptokenizer.java

```
/*
 * $Id: OpenNLPTokenizer.java,v 1.4 2009/04/01 09:47:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import opennlp.tools.tokenize.Tokenizer;
import opennlp.tools.tokenize.TokenizerME;
import edu.harvard.fas.rregan.ApplicationException;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * @author ron
 */
// @Component("openNLPTokenizer")
public class OpenNLPTokenizer extends AbstractOpenNLPTool<NLPText> {

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the open nlp tokenizer model file relative to the
     * classpath.
     *
     * @see PROP_ENGLISH_TOKENIZER_MODEL_FILE_DEFAULT for the default
     * location
     *      of the file
     */
    public static final String PROP_ENGLISH_TOKENIZER_MODEL_FILE =
"EnglishTokenizerModelFile";

    /**
     * The default path to the tokenizer model file
     */
    public static final String PROP_ENGLISH_TOKENIZER_MODEL_FILE_DEFAULT
= "resources/nlp/opennlp-tools/EnglishTok.bin.gz";

    private static Tokenizer tokenizer;
    static {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
```

```
        OpenNLPParser.class.getName());

            String modelFile =
resourceBundleHelper.getString(PROP_ENGLISH_TOKENIZER_MODEL_FILE,
PROP_ENGLISH_TOKENIZER_MODEL_FILE_DEFAULT);

            tokenizer = new TokenizerME(readGISModel(modelFile));
        } catch (Exception e) {
            tokenizer = null;
            throw new ExceptionInInitializerError(e);
        }
    }

    /**
     * create a new tokenizer, initializing if needed.
     *
     * @throws ApplicationException
     *          if the underlying OpenNLP tokenizer fails to initialize
     */
    public OpenNLPTokenizer() {
        init();
    }

    private synchronized void init() {
        if (tokenizer == null) {
            try {
                ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                    OpenNLPParser.class.getName());

                String modelFile = resourceBundleHelper.getString(
                    PROP_ENGLISH_TOKENIZER_MODEL_FILE,
                    PROP_ENGLISH_TOKENIZER_MODEL_FILE_DEFAULT);

                tokenizer = new TokenizerME(readGISModel(modelFile));
            } catch (Exception e) {
                tokenizer = null;
                throw ApplicationException.failedToInitializeComponent(getClass(),
e);
            }
        }
    }

    /**
     * @see
     * edu.harvard.fas.rregan.nlp.NLPPProcessor#process(edu.harvard.fas.rregan
     * .nlp.NLPText)
```

```

/*
@Override
public NLPText process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.SENTENCE)
        || text.is(GrammaticalStructureLevel.UNKNOWN)) {
        if (text.getChildren().isEmpty()) {
            for (String word : tokenizer.tokenize(text.getText())) {
                text.getChildren().add(new NLPTextImpl(word,
                    GrammaticalStructureLevel.WORD));
            }
        }
    }
    return text;
}

protected static Tokenizer getTokenizer() {
    return tokenizer;
}
}

```

openpanelevent.java

```

/*
 * $Id: OpenPanelEvent.java,v 1.5 2008/03/01 02:30:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * An event to be handled by a PanelContainer to open an appropriate
Panel.
 *
 * @author ron
 */
public class OpenPanelEvent extends NavigationEvent {
    static final long serialVersionUID = 0;

    private final Panel panel;
    private final PanelActionType panelActionType;
    private final Object targetObject;
    private final Class<?> targetType;

```

```

private final String panelName;
private final WorkflowDisposition disposition;

/**
 * An event for a PanelContainer to open a panel appropriate for the
 * parameters with a WorkflowDisposition of ContinueFlow, meaning the
panel
 * will be opened in a manner appropriate for returning to the
previous
 * window when it is closed, possibly returning some data.
*
 * @param source -
 *          the controller firing the event.
 * @param panelActionType -
 *          the action type (edit, select, navigate) of the window
to
 *          open.
 * @param targetObject -
 *          an object the panel will act on. it may be a different
type
 *          from the supplied targetType.
 * @param targetType -
 *          the type of object(s) the action is appropriate for.
 * @param panelName -
 *          an explicit name (or role) of the panel to open.
 */
public OpenPanelEvent(Object source, PanelActionType panelActionType,
Object targetObject,
Class<?> targetType, String panelName) {
this(source, panelActionType, targetObject, targetType, panelName,
WorkflowDisposition.ContinueFlow);
}

/**
 * An event for a PanelContainer to open a panel appropriate for the
 * parameters.
 *
 * @param source -
 *          the controller firing the event.
 * @param panelActionType -
 *          the action type (edit, select, navigate) of the window
to
 *          open.
 * @param targetObject -
 *          an object the panel will act on. it may be a different
type
 *          from the supplied targetType.

```

```

* @param targetType -
*           the type of object(s) the action is appropriate for.
* @param panelName -
*           an explicit name (or role) of the panel to open.
* @param disposition -
*           a WorkflowDisposition indicating if the panel is
supporting
*           the work of the previous panel (ContinueFlow) or if the
panel
*           is starting an independent task (NewFlow).
*/
public OpenPanelEvent(Object source, PanelActionType panelActionType,
Object targetObject,
Class<?> targetType, String panelName, WorkflowDisposition
disposition) {
super(source, "OpenPanel:" + panelActionType.name());
this.panel = null;
this.panelActionType = (panelActionType==null?
PanelActionType.Unspecified:panelActionType);
this.panelName = panelName;
this.disposition = disposition;
this.targetType = targetType;
this.targetObject = targetObject;
}

/**
 * Tell a panel container to display an existing panel. This event is
 * primarily for a PanelContainer to send to its PanelManager. For
example
 * when a TabbedPaneContainer detects the user clicking on a tab
button.
*
* @param source
* @param panel
*/
public OpenPanelEvent(Object source, Panel panel) {
super(source, null);
this.panel = panel;
this.panelActionType = null;
this.panelName = null;
this.disposition = null;
this.targetType = null;
this.targetObject = null;
}

public PanelActionType getPanelActionType() {
return panelActionType;
}

}
public Panel getPanel() {
return panel;
}

public String getPanelName() {
return panelName;
}

public Object getTargetObject() {
return targetObject;
}

public Class<?> getTargetType() {
return targetType;
}

public WorkflowDisposition getWorkflowDisposition() {
return disposition;
}

public void configurePanelWithData(Panel panelToInit) {
panelToInit.setTargetObject(getTargetObject());
}

@Override
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append(this.getClass().getSimpleName());
sb.append("[source = ");
sb.appendgetSource().getClass().getSimpleName());
if (panelActionType != null) {
sb.append(", panelActionType = ");
sb.append(panelActionType.name());
}
if (panel != null) {
sb.append(", panel = ");
sb.append(panel);
}
if (panelName != null) {
sb.append(", panelName = ");
sb.append(panelName);
}
if (targetObject != null) {
sb.append(", targetObject = ");
sb.append(targetObject);
}
}

```

```

        }
        if (targetType != null) {
            sb.append(", targetType = ");
            sb.append(targetType);
        }
        if (disposition != null) {
            sb.append(", disposition = ");
            sb.append(disposition.name());
        }
        sb.append("]");
    }
    return sb.toString();
}
}

```

optimisticlockexceptionadapter.java

```

/*
 * $Id: OptimisticLockExceptionAdapter.java,v 1.1 2008/12/13 00:41:11
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.requel.EntityLockException;

/**
 * @author ron
 */
public class OptimisticLockExceptionAdapter implements
EntityExceptionAdapter {

/*
 * (non-Javadoc)
 *
 * @see
edu.harvard.fas.rregan.requel.EntityExceptionAdapter#convert(java.lang
(Throwable,
 *      java.lang.Class, java.lang.Object,
 *      edu.harvard.fas.rregan.requel.EntityExceptionActionType)
 */
@Override
public EntityLockException convert(Throwable original, Class<?>
entityType, Object entity,
        EntityExceptionActionType actionType) {

```

```

        return EntityLockException.staleEntity(entityType, entity,
actionType);
    }
}

```

organization.java

```

/*
 * $Id: Organization.java,v 1.2 2009/01/07 09:50:38 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user;

/**
 * An entity for grouping users and projects, such as a department,
company,
 * customer.
 *
 * @author ron
 */
public interface Organization extends Comparable<Organization> {
 /**
 * @return the name of the organization.
 */
 public String getName();
}

```

organizationimpl.java

```

/*
 * $Id: OrganizationImpl.java,v 1.9 2009/01/10 11:08:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

```

```

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.user.Organization;

/**
 * @author ron
 */
@Entity
@Table(name = "organizations")
@XmlRootElement(name = "organization", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "organization", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class OrganizationImpl implements Organization, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private String name;
    private int version = 1; // start at 1 so hibernate recognizes the
new

    // instance as the initial value and not stale.

    /**
     * @param name
     */
    public OrganizationImpl(String name) {
        setName(name);
    }

    protected OrganizationImpl() {
        // for hibernate
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    protected Long getId() {
        return id;
    }

    protected void setId(Long id) {
        this.id = id;
    }
}

@Version
protected int getVersion() {
    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

@Column(unique = true, nullable = false)
@NotEmpty(message = "organization name is required.")
@XmlAttribute(name = "name")
public String getName() {
    return name;
}

protected void setName(String name) {
    this.name = name;
}

@Override
public int compareTo(Organization o) {
    return getName().compareTo(o.getName());
}

@Override
public String toString() {
    return Organization.class.getName() + "[" + getId() + "]: " +
getName();
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
            tmpHashCode = result;
        }
    }
}

```

```

    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
proxy
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    // TODO: will this work for proxies?
    final OrganizationImpl other = (OrganizationImpl) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getName() == null) {
        if (other.getName() != null) {
            return false;
        }
    } else if (!getName().equals(other.getName())) {
        return false;
    }
    return true;
}

```

organizedentity.java

```

package edu.harvard.fas.rregan.requel;

import edu.harvard.fas.rregan.requel.user.Organization;

/**
 * An entity that has an organization assigned.
 * @author ron
 */
public interface OrganizedEntity {
    /**
     * @return the default organization of this user.

```

```

    */
    public Organization getOrganization();

    /**
     * set the default organization of this user.
     *
     * @param organization
     */
    public void setOrganization(Organization organization);
}

```

panel.java

```

/*
 * $Id: Panel.java,v 1.13 2008/10/11 21:47:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import nextapp.echo2.app.LayoutData;

/**
 * A Panel is a component with the responsibility for handling a
specific type
 * of activity on a specific type of content. For example Editing a
User, or
 * selecting a user role type. A panel type may have an explicit name
that is
 * used to refer directly to that window for cases when multiple
windows may be
 * appropriate for a specific action and type. For example
"simpleSelector" vs.
 * "advancedSelector" for selecting users. NOTE: this name is for all
panels of
 * the same type and not an individual panel.
 *
 * @author ron
 */
public interface Panel extends PanelDescriptor {

    /**
     * The property to set in the resource bundle for the panel's title.
     */
    public static final String PROP_PANEL_TITLE = "Panel.Title";

    /**

```

```

 * The style name in the stylesheet for applying a style to the panel
title.
 */
public static final String STYLE_NAME_PANEL_TITLE = "Panel.Title";

/**
 * The default style from the stylesheet.
 */
public static final String STYLE_NAME_DEFAULT = "Default";

/**
 * The style name to use for plain controls.
 */
public static final String STYLE_NAME_PLAIN = "Plain";

/**
 * The style to use for text that indicates a validation problem or
error.
 */
public static final String STYLE_NAME_VALIDATION_LABEL =
"ValidationLabel";

/**
 *
 */
public static final String STYLE_NAME_HELP_LABEL = "HelpLabel";

/**
 * Set the name of the style to use for configuring this panel's
style
 * through the Echo2 stylesheet.
 *
 * @param styleName
 */
public void setStyleName(String styleName);

/**
 * Set the layout data of this panel relative to its container. This
is from
 * the Echo2 Component class.
 *
 * @param layoutData
 */
public void setLayoutData(LayoutData layoutData);

/**
 * @return

```

```

 */
public String getTitle();

/**
 * @return true if the panel's setup() method has been called since
it was
 *         created or last disposed.
 */
public boolean isInitialized();

/**
 * This method gets called once when the panel is first loaded so
that it
 * can create its UI components. The target object will be set before
 * setup() is called.
 */
public void setup();

/**
 * This gets called before a panel is removed so that it can remove
its
 * components and cleanup any state.
 */
public void dispose();

/**
 * Get the content object this panel should act on.
 *
 * @return
 */
public Object getTargetObject();

/**
 * Set the content object this panel should act on.
 *
 * @param targetObject
 */
public void setTargetObject(Object targetObject);

/**
 * @return the expected destination object for events fired by this
panel
 *         for events that support a destination (like selecting an
object.)
 */
public Object getDestinationObject();

```

```

    /**
     * Set the destination of events that have a destination or expected
     * recipient of an event.
     *
     * @param destinationObject
     */
    public void setDestinationObject(Object destinationObject);
}

```

panelactiontype.java

```

/*
 * $Id: PanelActionType.java,v 1.1 2008/02/27 11:34:39 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel;

/**
 * @author ron
 */
public enum PanelActionType {

    /**
     * Indicates the panel is an editor panel for editing some type of
     * content.
     */
    Editor(),

    /**
     * Indicates the panel is a selector panel for choosing some type of
     * content.
     */
    Selector(),

    /**
     * Indicates the panel is a navigator panel for listing and
     * navigating over
     * some type of content.
     */
    Navigator(),

    /**
     * Indicates the panel panel has an unspecified function and is only
     * accessed by name.
     */
    Unspecified();

    private PanelActionType() {

```

```

    }
}
```

panelcontainer.java

```

/*
 * $Id: PanelContainer.java,v 1.5 2008/02/29 19:36:09 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelManager;

/**
 * A PanelContainer manages how a set of related panels managed by the
 * container
 * are displayed. A PanelContainer has a PanelManager that manages the
 * creation
 * and state of panels based on NavigationEvents it receives from the
 * EventDispatcher.
 *
 * @author ron
 */
public interface PanelContainer extends Panel {

    /**
     * Display the supplied panel.
     *
     * @param panel
     * @param disposition
     */
    public void displayPanel(Panel panel, WorkflowDisposition
disposition);

    /**
     * Hide or remove the supplied panel.
     *
     * @param panel
     */
    public void undisplayPanel(Panel panel);

    /**

```

```

 * This is public so that the NavigationEventHandlers can get the
panel to
 * forward the event to.
 *
 * @return
 */
public PanelManager getPanelManager();
}

```

paneldescriptor.java

```

/*
 * $Id: PanelDescriptor.java,v 1.2 2008/02/29 19:36:11 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

/**
 * Describes the action, content type, and/or name of a panel.
 *
 * @author ron
 */
public interface PanelDescriptor {

    /**
     * @return the type of content the panel supports.
     */
    public Class<?> getSupportedContentTypes();

    /**
     * @return the type of action the panel supports.
     */
    public PanelActionType getSupportedActionTypes();

    /**
     * @return for a named panel return the name, otherwise return null.
     */
    public String getPanelName();

    /**
     * @return the class of the panel being described.
     */
    public Class<? extends Panel> getPanelType();

    /**
     * tell the descriptor to cleanup its resources and ui components.
     */

```

```

 */
public void dispose();
}

```

panelfactory.java

```

/*
 * $Id: PanelFactory.java,v 1.3 2008/03/06 02:41:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.lang.reflect.InvocationTargetException;

/**
 * @author ron
 */
public interface PanelFactory extends PanelDescriptor {

    /**
     * @return
     * @throws NoSuchMethodException
     * @throws InvocationTargetException
     * @throws InstantiationException
     * @throws IllegalAccessException
     */
    public Panel newInstance() throws NoSuchMethodException,
        InvocationTargetException,
        InstantiationException, IllegalAccessException;
}

```

panelmanager.java

```

/*
 * $Id: PanelManager.java,v 1.3 2008/03/10 23:57:20 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Set;

import nextapp.echo2.app.event.ActionListener;
import edu.harvard.fas.rregan.uiframework.PanelContainer;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * A PanelManager manages the creation and state of panels displayed
in a
 * PanelContainer. A PanelManager listens for navigation events from
the
 * EventDispatcher concerning panels in the PanelContainer and call
methods on
 * the container to alter the display. A PanelManager is only assigned
to a
 * single PanelContainer. <br>
 * NOTE: In cases where multiple PanelContainers are used, the
developer is
 * responsible for ensuring that two different managers aren't
configured to
 * display the same panels or listen for the same events.
 *
 * @author ron
 */
public interface PanelManager extends ActionListener {

 /**
 * Assign a PanelContainer to the manager for sending events to.
 *
 * @param panelContainer
 */
public void setPanelContainer(PanelContainer panelContainer);

 /**
 * Register a Panel or PanelFactory so that it is available for use
in
 * response to an event.
 *
 * @param panelDescriptor
 * @param eventDispatcher
 */
public void register(PanelDescriptor panelDescriptor, EventDispatcher
eventDispatcher);

 /**
 * Register a set of Panels or PanelFactories.
 *
 * @param panelDescriptor
 * @param eventDispatcher
 */

```

```

    public void register(Set<PanelDescriptor> panelDescriptor,
EventDispatcher eventDispatcher);

    /**
     * Cleanup any cached panels that the manager is holding by calling
dispose() on them.
     */
    public void dispose();
}
```

parserexception.java

```

/*
 * $Id: ParserException.java,v 1.3 2009/01/26 10:19:04 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

import edu.harvard.fas.rregan.ApplicationException;

/**
 * @author ron
 */
public class ParserException extends ApplicationException {
    static final long serialVersionUID = 0;

    public static final String MSG_FORMAT_PARSE_FAILED = "Parse failed.";

    /**
     * @return a ParseException with a message that parsing failed.
     */
    public static ParserException parseFailed() {
        return new ParserException(MSG_FORMAT_PARSE_FAILED);
    }

    protected ParserException(String format, Object... args) {
        super(format, args);
    }

    protected ParserException(Throwable e, String format, Object... args)
{
        super(e, format, args);
    }

    protected ParserException(String msg) {
        super(msg);
    }
}
```

```
}
```

parserpanel.java

```
/*
 * $Id: ParserPanel.java,v 1.3 2008/12/17 08:38:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.ui;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.nlp.NLPProcessorFactory;
import edu.harvard.fas.rregan.nlp.NLPText;
import
edu.harvard.fas.rregan.uiframework.panel.editor.AbstractEditorPanel;

/**
 * @author ron
 */
public class ParserPanel extends AbstractEditorPanel {
    private static final Logger log =
Logger.getLogger(ParserPanel.class);

    static final long serialVersionUID = 0L;

    public static final String PROP_LABEL_TEXT = "Text.Label";
    public static final String PROP_LABEL_OUTPUT = "Output.Label";
    public static final String PROP_LABEL_PARSE_BUTTON =
"ParseButton.Label";

    private final NLPProcessorFactory processorFactory;
    private Button parseButton;

    /**
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */

}
```

```
 */
public ParserPanel(NLPProcessorFactory processorFactory) {
    this(ParserPanel.class.getName(), processorFactory);
}

/**
 * @param resourceBundleName
 * @param processorFactory
 */
public ParserPanel(String resourceBundleName, NLPProcessorFactory
processorFactory) {
    super(resourceBundleName, null, NLPPanelNames.PARSER_PANEL_NAME);
    this.processorFactory = processorFactory;
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String msg =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
        "Parser Panel");
    return msg;
}

@Override
public void setup() {
    super.setup();
    addInput("text", PROP_LABEL_TEXT, "Text", new TextArea(), new
StringDocumentEx());
    addInput("output", PROP_LABEL_OUTPUT, "Output", new TextArea(), new
StringDocumentEx());
    parseButton = addActionButton(new
Button getResourceBundleHelper(getLocale()).getString(
    PROP_LABEL_PARSE_BUTTON, "Parse")));
    parseButton.addActionListener(new ParseListener(this));
    parseButton.setEnabled(true);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}
```

```

@Override
public boolean isReadOnlyMode() {
    return false;
}

@Override
public void cancel() {
    super.cancel();
}

@Override
public void save() {
    super.save();
}

protected NLPPProcessorFactory getProcessorFactory() {
    return processorFactory;
}

private static class ParseListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ParserPanel panel;

    private ParseListener(ParserPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            String text = panel.getInputValue("text", String.class);
            NLPText nlpText = panel.getProcessorFactory().createNLPText(text);
            panel.getProcessorFactory().getSentencizer().process(nlpText);
            panel.getProcessorFactory().getParser().process(nlpText);
            String output =
                panel.getProcessorFactory().getConstituentTreePrinter().process(
                    nlpText);
            panel.setInputValue("output", output);
        } catch (Exception e) {
            panel.setGeneralMessage("Could not parse text: " + e);
        }
    }
}

```

parsetag.java

```

/*
 * $Id: ParseTag.java,v 1.5 2009/02/11 09:02:56 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp;

import java.util.HashMap;
import java.util.Map;

/**
 * Helper for Penn Treebank tags. Adapted from text at
 * http://bulba.sdsu.edu/jeanette/thesis/PennTags.html by JEANETTE
 * PITTIBONE
 * pettibon@stanford.edu
 *
 * @author rreganjr@acm.org
 */
public enum ParseTag {

    /**
     *
     */
    ROOT("ROOT", GrammaticalStructureLevel.SENTENCE, "A Stanford parser
specific tag"),

    // Clause Level Tags

    /**
     * A simple declarative clause.
     */
    S("S", GrammaticalStructureLevel.CLAUSE,
        "simple declarative clause, i.e. one that is not introduced"
        + " by a (possibly empty) subordinating conjunction or a wh-word"
        + " and that does not exhibit subject-verb inversion."),

    /**
     * A clause introduced by a subordinating conjunction.
     */
    SBAR("SBAR", GrammaticalStructureLevel.CLAUSE,
        "Relative and subordinate clauses introduced by a (possibly empty)
subordinating conjunction."),

    /**
     * A direct question introduced by a wh-word or a wh-phrase.
     */

```

```

/*
SBARQ("SBARQ", GrammaticalStructureLevel.CLAUSE,
  "Direct question introduced by a wh-word or a wh-phrase. Indirect"
  + " questions and relative clauses should be bracketed as SBAR,
not SBARQ."),
 */

/***
 * Inverted declarative sentence.
 */
SINV("SINV", GrammaticalStructureLevel.CLAUSE,
  "Inverted declarative sentence, i.e. one in which the subject
follows"
  + " the tensed verb or modal."),

/***
 * Inverted yes/no question.
 */
SQ("SQ", GrammaticalStructureLevel.CLAUSE,
  "Inverted yes/no question, or main clause of a wh-question,
following"
  + " the wh-phrase in SBARQ."),

// Phrase Level Tags

/***
 * Adjective Phrase.
 */
ADJP("ADJP", GrammaticalStructureLevel.PHRASE, "Adjective Phrase."),

/***
 * Adverb Phrase.
 */
ADVP("ADVP", GrammaticalStructureLevel.PHRASE, "Adverb Phrase."),

/***
 * Conjunction Phrase.
 */
CONJP("CONJP", GrammaticalStructureLevel.PHRASE, "Conjunction
Phrase."),

/***
 * Fragment.
 */
FRAG("FRAG", GrammaticalStructureLevel.PHRASE, "Fragment."),

/***
 * Interjection.
 */
INTJ("INTJ", GrammaticalStructureLevel.PHRASE,
  "Interjection. Corresponds approximately to the part-of-speech tag
UH."),
 */

/***
 * @see tag UH
 */
LST("LST", GrammaticalStructureLevel.PHRASE, "List marker. Includes
surrounding punctuation."),
 */

/***
 * Not a Constituent.
 */
NAC("NAC", GrammaticalStructureLevel.PHRASE,
  "Not a Constituent; used to show the scope of certain prenominal"
  + " modifiers within an NP."),
 */

/***
 * Noun Phrase.
 */
NP("NP", GrammaticalStructureLevel.PHRASE, "Noun Phrase."),

/***
 * Used within certain complex NPs to mark the head of the NP.
 */
NX("NX", GrammaticalStructureLevel.PHRASE,
  "Used within certain complex NPs to mark the head of the NP."
  + " Corresponds very roughly to N-bar level but used quite
differently."),
 */

/***
 * Prepositional Phrase.
 */
PP("PP", GrammaticalStructureLevel.PHRASE, "Prepositional Phrase."),

/***
 * Parenthetical.
 */
PRN("PRN", GrammaticalStructureLevel.PHRASE, "Parenthetical."),
 */

/***
 * Particle. like a prepositional phrase ex: John went up the ladder.
parse:
 * (S (NP (NNP John)) (VP (VBD went) (PRT (RP up)) (NP (DT the) (NN

```

```

    * ladder))))(. .))
*/
PRT("PRT", GrammaticalStructureLevel.PHRASE,
    "Particle. Category for words that should be tagged RP."),
/***
 * Quantifier Phrase
 */
QP("QP", GrammaticalStructureLevel.PHRASE,
    "Quantifier Phrase (i.e. complex measure/amount phrase); used
within NP."),
/***
 * Reduced Relative Clause.
 */
RRC("RRC", GrammaticalStructureLevel.PHRASE, "Reduced Relative
Clause."),
/***
 * Unlike Coordinated Phrase.
 */
UCP("UCP", GrammaticalStructureLevel.PHRASE, "Unlike Coordinated
Phrase."),
/***
 * Vereb Phrase.
 */
VP("VP", GrammaticalStructureLevel.PHRASE, "Vereb Phrase."),
/***
 * Wh-adjective Phrase.
 */
WHADJP("WHADJP", GrammaticalStructureLevel.PHRASE,
    "Wh-adjective Phrase. Adjectival phrase containing a wh-adverb, as
in how hot."),
/***
 * Wh-adverb Phrase.
 */
WHAVP("WHAVP", GrammaticalStructureLevel.PHRASE,
    "Wh-adverb Phrase. Introduces a clause with an NP gap. May be null"
    "+ " (containing the 0 complementizer) or lexical, containing a
wh-adverb"
    "+ " such as how or why."),
/***
 * Wh-noun Phrase.
 */
    */
WHNP("WHNP", GrammaticalStructureLevel.PHRASE,
    "Wh-noun Phrase. Introduces a clause with an NP gap. May be null"
    "+ " (containing the 0 complementizer) or lexical, containing some
wh-word,"
    "+ " e.g. who, which book, whose daughter, none of which, or how
many leopards."),
/***
 * Wh-prepositional Phrase.
 */
WHPP(
    "WHPP",
    GrammaticalStructureLevel.PHRASE,
    "Wh-prepositional Phrase. Prepositional phrase containing a wh-noun
phrase"
    "+ " (such as of which or by whose authority) that either
introduces a PP gap or"
    "+ " is contained by a WHNP."),
/***
 * Unknown, uncertain, or unbracketable.
 */
X("X", GrammaticalStructureLevel.PHRASE,
    "Unknown, uncertain, or unbracketable. X is often used for
bracketing typos"
    "+ " and in bracketing the...the-constructions."),
// Word level
/***
 * Coordinating conjunction.
 */
CC("CC", GrammaticalStructureLevel.WORD, PartOfSpeech.CONJUNCTION,
"Coordinating conjunction"),
/***
 * Cardinal number.
 */
CD("CD", GrammaticalStructureLevel.WORD, PartOfSpeech.NUMBER,
"Cardinal number"),
/***
 * Determiner.
 */
DT("DT", GrammaticalStructureLevel.WORD, PartOfSpeech.DETERMINER,
"Determiner"),

```

```

/**
 * Existential there. There is a fly in my soup. (S (NP (EX There))
(VP (VBZ
 * is) (NP (NP (DT a) (NN fly)) (PP (IN in) (NP (PRP$ my) (NN soup))))))
(. .))
*/
EX("EX", GrammaticalStructureLevel.WORD, PartOfSpeech.MODAL,
"Existential there"),

/***
 * Foreign word.
 */
FW("FW", GrammaticalStructureLevel.WORD, PartOfSpeech.UNKNOWN,
"Foreign word"),

/***
 * Preposition or subordinating conjunction.
 */
IN("IN", GrammaticalStructureLevel.WORD, PartOfSpeech.PREPOSITION,
"Preposition or subordinating conjunction"),

/***
 * Adjective.
 */
JJ("JJ", GrammaticalStructureLevel.WORD, PartOfSpeech.ADJECTIVE,
"Adjective"),

/***
 * Adjective, comparative
 */
JJR("JJR", GrammaticalStructureLevel.WORD, PartOfSpeech.ADJECTIVE,
"Adjective, comparative"),

/***
 * Adjective, superlative
 */
JJS("JJS", GrammaticalStructureLevel.WORD, PartOfSpeech.ADJECTIVE,
"Adjective, superlative"),

/***
 * List item marker
 */
LS("LS", GrammaticalStructureLevel.WORD, PartOfSpeech.UNKNOWN, "List
item marker"),

/***
 * Modal
 */

 /**
MD("MD", GrammaticalStructureLevel.WORD, PartOfSpeech.MODAL,
"Modal"),

 /**
 * Noun, singular or mass
 */
NN("NN", GrammaticalStructureLevel.WORD, PartOfSpeech.NOUN, "Noun,
singular or mass"),

 /**
 * Noun, plural
 */
NNS("NNS", GrammaticalStructureLevel.WORD, PartOfSpeech.NOUN, "Noun,
plural"),

 /**
 * Proper noun, singular
 */
NNP("NNP", GrammaticalStructureLevel.WORD, PartOfSpeech.NOUN, "Proper
noun, singular"),

 /**
 * Proper noun, plural
 */
NNPS("NNPS", GrammaticalStructureLevel.WORD, PartOfSpeech.NOUN,
"Proper noun, plural"),

 /**
 * Predeterminer
 */
PDT("PDT", GrammaticalStructureLevel.WORD, PartOfSpeech.DETERMINER,
"Predeterminer"),

 /**
 * Possessive ending for example "John's hair is brown."<br>
 * (S (NP (NP (NNP John) (POS 's)) (NN hair)) (VP (VBZ is) (ADJP (JJ
brown)))) .
 */
POS("POS", GrammaticalStructureLevel.WORD, PartOfSpeech.UNKNOWN,
"Possessive ending"),

 /**
 * Personal pronoun
 */

```

```

PRP("PRP", GrammaticalStructureLevel.WORD, PartOfSpeech.PRONOUN,
"Personal pronoun"),

/***
 * Possessive pronoun
 */
PRPS("PRPS", GrammaticalStructureLevel.WORD, PartOfSpeech.PRONOUN,
"Possessive pronoun"),

/***
 * Adverb
 */
RB("RB", GrammaticalStructureLevel.WORD, PartOfSpeech.ADVERB,
"Adverb"),

/***
 * Adverb, comparative list more or quicker
 */
RBR("RBR", GrammaticalStructureLevel.WORD, PartOfSpeech.ADVERB,
"Adverb, comparative"),

/***
 * Adverb, superlative like most or quickest
 */
RBS("RBS", GrammaticalStructureLevel.WORD, PartOfSpeech.ADVERB,
"Adverb, superlative"),

/***
 * Particle - like an adverb/preposition John went up the ladder.
 "how did
 * John go on the ladder? up" (S (NP (NNP John)) (VP (VBD went) (PRT
(RP
 * up)) (NP (DT the) (NN ladder))))(. .)) vs. John went under the
ladder.
 * "where did John go? under the ladder" (S (NP (NNP John)) (VP (VBD
went) (PP
 * (IN under) (NP (DT the) (NN ladder))))(. .))
 */
RP("RP", GrammaticalStructureLevel.WORD, PartOfSpeech.ADVERB,
"Particle"),

/***
 * Symbol
 */
SYM("SYM", GrammaticalStructureLevel.WORD, PartOfSpeech.SYMBOL,
"Symbol"),

```

```

    /**
     * to - like a particle, may act as an adverb or preposition, can be
     * determined by the phrase. John went to the store. (S (NP (NNP
John)) (VP
     * (VBD went) (PP (TO to) (NP (DT the) (NN store))))(. .)) John went
under the
     * bridge. (S (NP (NNP John)) (VP (VBD went) (PP (IN under) (NP (DT the
(NN
     * bridge))))(. .)) the boat swayed to and fro. (S (NP (DT the) (NN
boat)) (VP
     * (VBN swayed) (ADVP (TO to) (CC and) (RB fro))))(. .)) the boat sailed
to the
     * reef. (S (NP (DT the) (NN boat)) (VP (VBD sailed) (PP (TO to) (NP (DT
the)
     * (NN reef))))(. .)))
     */
    TO("TO", GrammaticalStructureLevel.WORD, PartOfSpeech.PREPOSITION,
"to"),

    /**
     * Interjection
     */
    UH("UH", GrammaticalStructureLevel.WORD, PartOfSpeech.INTERJECTION,
"Interjection"),

    /**
     * Verb, base form
     */
    VB("VB", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB, "Verb,
base form"),

    /**
     * Verb, past tense
     */
    VBD("VBD", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB, "Verb,
past tense"),

    /**
     * Verb, gerund or present participle
     */
    VBG("VBG", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB,
"Verb, gerund or present participle"),

    /**
     * Verb, past participle
     */

```

```

VBN("VBN", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB, "Verb,
past participle"),

/**
 * Verb, non-3rd person singular present
 */
VBP("VBP", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB,
    "Verb, non-3rd person singular present"),

/**
 * Verb, 3rd person singular present
 */
VBZ("VBZ", GrammaticalStructureLevel.WORD, PartOfSpeech.VERB,
    "Verb, 3rd person singular present"),

/**
 * Wh-determiner
 */
WDT("WDT", GrammaticalStructureLevel.WORD, PartOfSpeech.DETERMINER,
    "Wh-determiner"),

/**
 * Wh-pronoun
 */
WP("WP", GrammaticalStructureLevel.WORD, PartOfSpeech.PRONOUN, "Wh-
pronoun"),

/**
 * Possessive wh-pronoun
 */
WP$("WP$", GrammaticalStructureLevel.WORD, PartOfSpeech.PRONOUN,
    "Possessive wh-pronoun"),

/**
 * Wh-adverb
 */
WRB("WRB", GrammaticalStructureLevel.WORD, PartOfSpeech.ADVERB, "Wh-
adverb"),

/**
 *
 */
PUNC_DOLLAR("$", GrammaticalStructureLevel.WORD,
PartOfSpeech.PUNCTUATION, "A dollar sign ($)."),

/**
 * Marks the end of a sentence
 */

        */

PUNC_TERMINATOR(".", GrammaticalStructureLevel.WORD,
PartOfSpeech.PUNCTUATION,
    "Punctuation ending a sentence such as a period, question mark or
exclamation mark."),

/**
 *
 */
PUNC_DQUOTE("\\"", GrammaticalStructureLevel.WORD,
PartOfSpeech.PUNCTUATION,
    "A double quote (\\".\"),",
/**
 *
 */
PUNC_SQUOTE('\'', GrammaticalStructureLevel.WORD,
PartOfSpeech.PUNCTUATION,
    "A single quote ('.'),",
/**
 *
 */
PUNC_NON_TERMINATOR(",", GrammaticalStructureLevel.WORD,
PartOfSpeech.PUNCTUATION,
    "Punctuation joining clauses or phrases such as a comma or
semicolon.");

private static final Map<String, ParseTag> tagsByText = new
HashMap<String, ParseTag>();
static {
    for (ParseTag tag : values()) {
        tagsByText.put(tag.getText(), tag);
    }
}

/**
 * @param text
 * @return The tag enum based on the parser tag text such as "S",
"NP",
 *         "NN", etc.
 */
public static ParseTag tagOf(String text) {
    if (tagsByText.containsKey(text)) {
        return tagsByText.get(text);
    }
    return X;
}

```

```

private final String text;
private final String description;
private final GrammaticalStructureLevel grammaticalStructureLevel;
private final PartOfSpeech partOfSpeech;

private ParseTag(String text, GrammaticalStructureLevel
grammaticalStructureLevel,
    String description) {
    this.text = text;
    this.grammaticalStructureLevel = grammaticalStructureLevel;
    this.partOfSpeech = PartOfSpeech.UNKNOWN;
    this.description = description;
}

private ParseTag(String text, GrammaticalStructureLevel
grammaticalStructureLevel,
    PartOfSpeech partOfSpeech, String description) {
this.text = text;
this.grammaticalStructureLevel = grammaticalStructureLevel;
this.partOfSpeech = partOfSpeech;
this.description = description;
}

/**
 * @return The text of the tag returned by the parser, for example
"S",
 *      "NP", "NNS"
 */
public String getText() {
    return text;
}

/**
 * @return An informative description for the tag
 */
public String getDescription() {
    return description;
}

/**
 * @return The grammatical level of the structure represented by the
tag.
 */
public GrammaticalStructureLevel getGrammaticalStructureLevel() {
    return grammaticalStructureLevel;
}

```

```

/**
 * @return The part of speech for word level tags, unknown for other
levels.
 */
public PartOfSpeech getPartOfSpeech() {
    return partOfSpeech;
}
}

partofspeech.java

/*
 * $Id: PartOfSpeech.java,v 1.10 2009/03/27 07:16:09 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp;

/**
 * The part of speech indicates how a word or token is used in a
sentence.
 *
 * @author ron
 */
public enum PartOfSpeech {
    /**
     * Default if the part of speech is unknown because the word is not
recognized or if the word is ambiguous.
     */
    UNKNOWN(),

    /**
     * modal modifiers such as should, must
     */
    MODAL(),

    /**
     */
    VERB(),

    /**
     */
    NOUN(),
}

```

```

/**
 *
 */
PRONOUN(),
/***
 * A sub class of NOUNs representing a name. Proper nouns are
identified to
 * facilitate spelling and identification of key terms.
 */
PROPERNOUN(),

/***
 * A sub class of NOUNs for numeric values
 */
NUMBER(),

/***
 *
 */
ADJECTIVE(),

/***
 *
 */
ADVERB(),

/***
 *
 */
PREPOSITION(),

/***
 *
 */
CONJUNCTION(),

/***
 *
 */
INTERJECTION(),

/***
 * functional markers that delimit or demarcate parts of a sentence.
 */
PUNCTUATION(),

```

```

/**
 * Non word symbols that aren't punctuation, like math symbols.
Symbols may
 * be marked lexically by the parser, for example in the "Bracketing
 * Guidelines for Treebank II Style" pg 58. an equals may be labelled
as a
 * VP as shown in the following example:<br>
 * (S (NP-SBJ x) (VP = (NP 3)))
 */
SYMBOL(),

/***
 * For example "the", "a", "all", "any", "which"
 *
 * @see http://en.wikipedia.org/wiki/Determiner_%28class%29
 */
DETERMINER();

public boolean in(PartOfSpeech... poss) {
    for (PartOfSpeech pos : poss) {
        if (equals(pos)) {
            return true;
        }
    }
    return false;
}

/**
 * Convert the wordnet part of speech field to a PartOfSpeech enum
value.
 * WordNet uses n - noun, v - verb, a - adjective, r - adverb, s -
adjective
 * satellite
 *
 * @param wordNetPOS
 * @return
 */
public static PartOfSpeech fromWordNetPOS(String wordNetPOS) {
    if ("n".equals(wordNetPOS)) {
        return NOUN;
    } else if ("v".equals(wordNetPOS)) {
        return VERB;
    } else if ("r".equals(wordNetPOS)) {
        return ADVERB;
    } else if ("a".equals(wordNetPOS)) {
        return ADJECTIVE;
    } else if ("s".equals(wordNetPOS)) {

```

```

        return ADJECTIVE;
    }
    return UNKNOWN;
}

/**
 * @param partOfSpeech
 * @return The WordNet part of speech string for the supplied part of
speech
 *         object.
 */
public static String toWordNetPOS(PartOfSpeech partOfSpeech) {
if (partOfSpeech.equals(NOUN)) {
    return "n";
} else if (partOfSpeech.equals(VERB)) {
    return "v";
} else if (partOfSpeech.equals(ADVERB)) {
    return "a";
} else if (partOfSpeech.equals(ADJECTIVE)) {
// TODO: satellite adjectives use "s"
    return "r";
}
return null;
}
}

```

partofspeechandsenseprinter.java

```

/*
 * $Id: PartOfSpeechAndSensePrinter.java,v 1.5 2009/01/26 10:19:00
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;

/**
 * Print out an NLPText annotating each word with its part of speech
and sense.<br>
 * For example:<br>
 * John walked the dog. John#n walk#v#3 the dog#n#1 .
 */

```

```

        * @author ron
    */
public class PartOfSpeechAndSensePrinter implements
NLPTextWalkerFunction<StringBuilder> {

private final int bufferSize;
private StringBuilder sb;

    /**
     */
    public PartOfSpeechAndSensePrinter() {
this(1000);
}

public PartOfSpeechAndSensePrinter(int bufferSize) {
this.bufferSize = bufferSize;
}

@Override
public void init() {
sb = new StringBuilder(bufferSize);
}

@Override
public void begin(NLPText text) {
if
(GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLevel())) {
    sb.append(text.getLemma());
    if ((text.getPartOfSpeech() != null)
&& !text.getPartOfSpeech().in(PartOfSpeech.UNKNOWN,
PartOfSpeech.PUNCTUATION,
PartOfSpeech.DETERMINER, PartOfSpeech.PREPOSITION,
PartOfSpeech.CONJUNCTION)) {
        sb.append("#");
        sb.append(text.getPartOfSpeech().name().substring(0,
1).toLowerCase());
        if (text.getDictionaryWordSense() != null) {
            sb.append("#");
            sb.append(text.getDictionaryWordSense().getRank());
        }
    }
}
}

@Override

```

```

public StringBuilder end(NLPText text) {
    if
(GrammaticalStructureLevel.WORD.equals(text.getGrammaticalStructureLev
el())) {
    sb.append(" ");
}
return sb;
}
}

```

persistencecontexthelper.java

```

/*
 * $Id: PersistenceContextHelper.java,v 1.2 2009/01/22 10:36:30 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import java.lang.reflect.Method;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

/**
 * @author ron
 */
@Component("persistenceContextHelper")
@Scope("singleton")
public class PersistenceContextHelper {
    private static final Logger log =
Logger.getLogger(PersistenceContextHelper.class);

    @PersistenceContext
    private EntityManager entityManager;

    /**
     */
    public PersistenceContextHelper() {

```

```

        }

        /**
         * @param entityHolder
         * @param domainObjectWrapper
         * @param method
         * @param args
         * @return
         * @throws Throwable
         */
        @Transactional(propagation = Propagation.REQUIRED)
        public Object invokeInTransaction(EntityProxyInterceptor
entityHolder,
        DomainObjectWrapper domainObjectWrapper, Method method, Object...
args)
            throws Throwable {
        Object cachedEntity = entityHolder.getEntity();
        Object refreshedEntity = AbstractJpaRepository.attach(entityManager,
cachedEntity);
        entityHolder.setEntity(refreshedEntity);
        StringBuffer sb = null;
        if (log.isDebugEnabled()) {
            sb = new StringBuffer();
            sb.append(method.getName());
            sb.append("(");
            for (int i = 0; (args != null) && (i < args.length); i++) {
                if (i != 0) {
                    sb.append(", ");
                }
                sb.append(args[i]);
            }
            sb.append(")");
        }
        Object rVal = method.invoke(refreshedEntity, args);
        if (log.isDebugEnabled() && (sb != null)) {
            if (!method.getReturnType().equals(void.class)) {
                sb.append(" -> ");
                sb.append(rVal);
            }
            log.debug(sb.toString());
        }
        return domainObjectWrapper.wrapPersistentEntities(rVal);
    }
}

```

position.java

```
/*
 * $Id: Position.java,v 1.9 2008/08/08 08:01:11 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation;

import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * A position on how to resolve an issue.
 *
 * @author ron
 */
public interface Position extends Comparable<Position>, CreatedEntity
{

    /**
     * @return All the issues that have this position as a resolution
     * option.
     */
    public Set<Issue> getIssues();

    /**
     * @return the text describing how to resolve an issue.
     */
    public String getText();

    /**
     * @return The arguments that are for and against this position.
     */
    public Set<Argument> getArguments();

    /**
     * Resolve the supplied issue with this position.
     *
     * @param issue -
     *            the issue to resolve.
     * @param resolvedByUser -
     *            the user that initiated the resolution.
     * @throws Exception -
     *            TODO: what can be thrown?
     */
}
```

```
        /*
         * $Id: Position.java,v 1.9 2008/08/08 08:01:11 rregan Exp $
         * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
         */
        public void resolveIssue(Issue issue, User resolvedByUser) throws
Exception;

        /**
         * Resolve all issues with this position.
         *
         * @param resolvedByUser
         * @throws Exception
         */
        public void resolveAllIssue(User resolvedByUser) throws Exception;
    }
```

positioneditorpanel.java

```
/*
 * $Id: PositionEditorPanel.java,v 1.22 2009/02/22 09:07:23 rregan Exp
 */
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.annotation;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Column;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
```

```

import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeletePositionCommand
;
import
edu.harvard.fas.rregan.requel.annotation.command.EditPositionCommand;
import edu.harvard.fas.rregan.project.Project;
import edu.harvard.fas.rregan.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
l;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * The name to use in the PositionEditorPanel.properties file to set
the
 * label of the position text field. If the property is undefined
"Position"
 * is used.
 */
public static final String PROP_LABEL_POSITION = "Position.Label";

/**
 * The name to use in the PositionEditorPanel.properties file to set
the
 * label of the arguments field. If the property is undefined
"Arguments" is
 * used.
 */
public static final String PROP_LABEL_ARGUMENTS = "Arguments.Label";

/**
 * The name to use in the PositionEditorPanel.properties file to set
the
 * label of the view button in the argument edit table column. If the
 * property is undefined "View" is used.
 */
public static final String PROP_VIEW_ARGUMENT_BUTTON_LABEL =
"ViewArgument.Label";

/**
 * The name to use in the PositionEditorPanel.properties file to set
the
 * label of the edit button in the argument edit table column. If the
 * property is undefined "Edit" is used.
 */
public static final String PROP_EDIT_ARGUMENT_BUTTON_LABEL =
>EditArgument.Label";

/**
 * The name to use in the PositionEditorPanel.properties file to set
the
 */

```

```

 * label of the add button in the arguments editor. If the property
is
 * undefined "Add" is used.
 */
public static final String PROP_ADD_ARGUMENT_BUTTON_LABEL =
"AddArgument.Label";

private final AnnotationCommandFactory annotationCommandFactory;
private UpdateListener updateListener;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted = false;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public PositionEditorPanel(CommandHandler commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository annotationRepository) {
    this(PositionEditorPanel.class.getName(), commandHandler,
annotationCommandFactory,
    annotationRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param annotationCommandFactory
 * @param annotationRepository
 */
public PositionEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    AnnotationCommandFactory annotationCommandFactory,
    AnnotationRepository annotationRepository) {
    super(resourceBundleName, Position.class, commandHandler,
annotationRepository);
    this.annotationCommandFactory = annotationCommandFactory;
}

/**
 * If the editor is editing an existing position the title specified
in the

```

* properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:

 * "Edit Position"

 * For a new position it first tries PROP_NEW_OBJECT_PANEL_TITLE,
then
 * PROP_PANEL_TITLE and finally defaults to:

 * "New Position"

 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
 if (getPosition() != null) {
 String msgPattern = getResourceBundleHelper(getLocale()).getString(
 PROP_EXISTING_OBJECT_PANEL_TITLE,
 getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
 "Edit Position"));
 return MessageFormat.format(msgPattern, getPosition().toString());
 } else {
 String msg = getResourceBundleHelper(getLocale()).getString(
 PROP_NEW_OBJECT_PANEL_TITLE,
 getResourceBundleHelper(getLocale())
 .getString(PROP_PANEL_TITLE, "New Position"));
 return msg;
 }
}

@Override
public void setup() {
 super.setup();
 Position position = getPosition();
 if (position != null) {
 addInput("text", PROP_LABEL_POSITION, "Position", new TextArea(),
new StringDocumentEx(
 position.getText()));
 addMultiRowInput("arguments", PROP_LABEL_ARGUMENTS, "Arguments",
getArgumentsTable(),
 new NavigatorTableModel((Collection) position.getArguments()));
 } else {
 addInput("text", PROP_LABEL_POSITION, "Position", new TextArea(),

```

    new StringDocumentEx());
    addMultiRowInput("arguments", PROP_LABEL_ARGUMENTS, "Arguments",
getArgumentsTable(),
    new NavigatorTableModel(new TreeSet<Object>()));
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);

}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

private Component getArgumentsTable() {
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    Column col = new Column();
    NavigatorTable relationTable = new
NavigatorTable(getArgumentsTableConfig());
    relationTable.setLayoutData(layoutData);
    col.add(relationTable);
    if ((getPosition() != null) && !isReadOnlyMode()) {
        String buttonLabel = null;
        buttonLabel = getResourceBundleHelpergetLocale().getString(
            PROP_ADD_ARGUMENT_BUTTON_LABEL, "Add");
        NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor,
            getPosition(), Argument.class, null,
WorkflowDisposition.NewFlow);
        NavigatorButton openEditorButton = new NavigatorButton(buttonLabel,
            getEventDispatcher(), openEditorEvent);
    }
}

```

```

    openEditorButton.setStyleName(STYLE_NAME_DEFAULT);
    openEditorButton.setLayoutData(layoutData);
    col.add(openEditorButton);
}
return col;
}

private NavigatorTableConfig getArgumentsTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Argument argument = (Argument) model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_VIEW_ARGUMENT_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelpergetLocale().getString(
                        PROP_EDIT_ARGUMENT_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor, argument, Argument.class, null,
WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                openEditorButton.setLayoutData(rld);
                return openEditorButton;
            }
        });
}

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Support
Level",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Argument argument = (Argument) model.getBackingObject(row);
                return argument.getSupportLevel().toString();
            }
        });
}

```

```

        });

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Text",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Argument argument = (Argument) model.getBackingObject(row);
                return argument.getText();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Argument argument = (Argument) model.getBackingObject(row);
                return argument.getCreatedBy().getUsername();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Argument argument = (Argument) model.getBackingObject(row);
                DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                return formatter.format(argument.getDateCreated());
            }
        }));
}

return tableConfig;
}

@Override
public boolean isReadOnlyMode() {
    // project entities are subject to stakeholder permissions, so if
this
    // position is on an issue concerned with a project or project
entity
    // check the stakeholder permissions for editing annotations.
    User user = (User) getApp().getUser();
    Project project = getProject();
}

```

```

        if (project != null) {
            Stakeholder stakeholder = project.getUserStakeholder(user);
            if (stakeholder != null) {
                return !stakeholder.hasPermission(Annotation.class,
                    StakeholderPermissionType.Edit);
            }
        }
        return false;
    }

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditPositionCommand command =
getAnnotationCommandFactory().newEditPositionCommand();
        command.setPosition(getPosition());
        command.setIssue(getIssue());
        command.setEditedBy(getCurrentUser());
        command.setText(getInputValue("text", String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
                updateListener);
        }
        Position position = command.getPosition();
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
            position));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        }
    }
}

```

```

} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}
} catch (Exception e) {
    log.error("could not save the goal: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
    try {
        Set<Issue> issues = getPosition().getIssues();
        DeletePositionCommand deletePositionCommand =
getAnnotationCommandFactory()
            .newDeletePositionCommand();
        deletePositionCommand.setPosition(getPosition());
        deletePositionCommand.setEditedBy(getCurrentUser());
        deletePositionCommand =
getCommandHandler().execute(deletePositionCommand);
        deleted = true;
        for (Issue issue : issues) {
            getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
issue));
        }
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private Project getProject() {
    Project project = null;
    if (getIssue() != null) {
        project = (Project) getIssue().getGroupingObject();
    } else if (getPosition() != null) {
        project = (Project)
getPosition().getIssues().iterator().next().getGroupingObject();
    }
    return project;
}

```

```

private Issue getIssue() {
    if (getTargetObject() instanceof Issue) {
        return (Issue) getTargetObject();
    }
    return null;
}

private Position getPosition() {
    if (getTargetObject() instanceof Position) {
        return (Position) getTargetObject();
    }
    return null;
}

private AnnotationCommandFactory getAnnotationCommandFactory() {
    return annotationCommandFactory;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final PositionEditorPanel panel;

    private UpdateListener(PositionEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        Position existingPosition = panel.getPosition();
        if ((e instanceof UpdateEntityEvent) && (existingPosition != null))
{
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            Position updatedPosition = null;
            if (event.getObject() instanceof Position) {
                updatedPosition = (Position) event.getObject();
                if (existingPosition.equals(updatedPosition)
                    && (event instanceof DeletedEntityEvent)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingPosition));
                    return;
                }
            }
        }
    }
}

```

positionimpl.java

```
package edu.harvard.fas.rregan.requell.annotation.impl;

import java.io.Serializable;
import java.util.Date;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
```

```
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Argument;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.User2UserImplAdapter;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import edu.harvard.fas.rregan.requel.utils.jaxb.DateAdapter;
import edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "positions")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "position_type", discriminatorType =
DiscriminatorType.STRING, length = 255)
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl")
@XmlRootElement(name = "position", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "position", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class PositionImpl implements Position, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private String type;
    private Set<Issue> issues = new TreeSet<Issue>();
```

```

private String text;
private Set<Argument> arguments = new TreeSet<Argument>();
private User createdBy;
private Date dateCreated = new Date();
// start at 1 so hibernate recognizes the new
// instance as the initial value and not stale.
private int version = 1;

/**
 * @param text
 * @param createdBy
 */
public PositionImpl(String text, User createdBy) {
    this(PositionImpl.class.getName(), text, createdBy);
}

/**
 * Create a new position for the issue with the supplied text and the
user.
 *
 * @param type -
 *          the class name of the issue used as the type
discriminator in
 *          the database.
 * @param text
 * @param createdBy
 */
protected PositionImpl(String type, String text, User createdBy) {
    setType(type);
    setText(text);
    setCreatedBy(createdBy);
    setDateCreated(new Date());
}

protected PositionImpl() {
    // for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlElement(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return id;
}

```

```

protected void setId(Long id) {
    this.id = id;
}

@Column(name = "position_type", insertable = false, updatable =
false)
protected String getType() {
    return type;
}

protected void setType(String type) {
    this.type = type;
}

@Version
protected int getVersion() {
    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

@XmlTransient
@ManyToMany(targetEntity = IssueImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "position_issue", joinColumns = { @JoinColumn(name =
"position_id") }, inverseJoinColumns = { @JoinColumn(name =
"issue_id") })
public Set<Issue> getIssues() {
    return issues;
}

protected void setIssues(Set<Issue> issues) {
    this.issues = issues;
}

@XmlElement(name = "text", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

```

```

@XmlElementWrapper(name = "arguments", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = ArgumentImpl.class)
@OneToMany(targetEntity = ArgumentImpl.class, mappedBy = "position",
cascade = CascadeType.ALL, fetch = FetchType.LAZY)
public Set<Argument> getArguments() {
    return arguments;
}

protected void setArguments(Set<Argument> arguments) {
    this.arguments = arguments;
}

@Override
public void resolveIssue(Issue issue, User resolvedByUser) throws
Exception {
    issue.resolve(this, resolvedByUser);
}

@Override
public void resolveAllIssue(User resolvedByUser) throws Exception {
    for (Issue issue : getIssues()) {
        issue.resolve(this, resolvedByUser);
    }
}

@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.REFRESH }, optional = false)
@XmlIDREF()
@XmlAttribute(name = "createdBy")
@XmlJavaTypeAdapter(User2UserImplAdapter.class)
public User getCreatedBy() {
    return createdBy;
}

protected void setCreatedBy(User createdBy) {
    this.createdBy = createdBy;
}

@XmlAttribute(name = "dateCreated")
@XmlJavaTypeAdapter(DateAdapter.class)
@Column(updatable = false)
@Temporal(TemporalType.TIMESTAMP)
public Date getDateCreated() {
    return dateCreated;
}

```

```

protected void setDateCreated(Date dateCreated) {
    this.dateCreated = dateCreated;
}

@Override
public int compareTo(Position o) {
    PositionImpl other = (PositionImpl) o;
    int dateCompare = (getDateCreated() == null ? -1 :
    getDateCreated().compareTo(
        other.getDateCreated()));
    int createdByCompare = (getCreatedBy() == null ? -1 :
    getCreatedBy().compareTo(
        other.getCreatedBy()));
    return (dateCompare != 0 ? dateCompare : (createdByCompare != 0 ?
    createdByCompare
        : (getText() == null ? -1 :
    getText().compareTo(other.getText()))));
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    final int prime = 31;
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        int result = 1;
        result = prime * result
            + ((getDateCreated() == null) ? 0 : getDateCreated().hashCode());
        result = prime * result + ((getCreatedBy() == null) ? 0 :
        getCreatedBy().hashCode());
        result = prime * result + ((getText() == null) ? 0 :
        getText().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {

```

```

    return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
    return false;
}
final PositionImpl other = (PositionImpl) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getText() == null) {
    if (other.getText() != null) {
        return false;
    }
} else if (!getText().equals(other.getText())) {
    return false;
}
return true;
}

/**
 * This is for JAXB to patchup the type
 *
 * @see UnmarshallerListener
 */
public void beforeUnmarshal() {
    setType(getClass().getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
}

/**

```

```

    * This class is used by JAXB to convert the id of an entity into an
xml id
    * string that will be distinct from other entity xml id strings by
the use
    * of a prefix.
    *
    * @author ron
    */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "POS_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

/**
 * Adapter for JAXB to convert interface Position to class
PositionImpl and
 * back.
 */
@XmlTransient
public static class Position2PositionImplAdapter extends
XmlAdapter<PositionImpl, Position> {
    private static final String prefix = "POS_";

    @Override
    public Position unmarshal(PositionImpl entity) throws Exception {
        return entity;
    }

    @Override
    public PositionImpl marshal(Position entity) throws Exception {
        return (PositionImpl) entity;
    }
}

```

predicateverbfinder.java

```
/*
 * $Id: PredicateVerbFinder.java,v 1.3 2009/02/06 11:49:15 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * @author ron
 */
public class PredicateVerbFinder extends PrimaryVerbPhraseFinder {

    /**
     * If the text is a SENTENCE, CLAUSE or PHRASE, return the primary
     * verb.
     *
     * @param text -
     *          a SENTENCE or CLAUSE to find the predicate verb of
     */
    @Override
    public NLPText process(NLPText text) {
        NLPText predicatePhrase = null;
        if (text.is(GrammaticalStructureLevel.CLAUSE)
            || text.is(GrammaticalStructureLevel.SENTENCE)) {
            predicatePhrase = super.process(text);
        } else if (text.is(GrammaticalStructureLevel.PHRASE) &&
        text.is(ParseTag.VP)) {
            predicatePhrase = text;
        }
        if (predicatePhrase != null) {
            for (NLPText child : predicatePhrase.getChildren()) {
                if (child.is(GrammaticalStructureLevel.WORD) &&
                child.is(PartOfSpeech.VERB)) {
                    return child;
                } else if (child.is(GrammaticalStructureLevel.PHRASE)
                    && child.is(ParseTag.VP)) {
                    return process(child);
                }
            }
        }
    }
}
```

```
    }
    return null;
}
}
```

prepositionmatchingrule.java

```
/*
 * $Id: PrepositionMatchingRule.java,v 1.3 2009/02/11 09:02:55 rregan
 * Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * @author ron
 */
public class PrepositionMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
    Logger.getLogger(PrepositionMatchingRule.class);

    // list of prepositions that the match must be one of.
    private final List<String> matchingFilter;

    /**
     * Create a rule that matches any preposition
     */
    public PrepositionMatchingRule() {
        matchingFilter = new ArrayList<String>();
    }

    /**
     * Create a rule that matches one of the supplied preposition

```

```

/*
 * @param prepositions -
 *          a space delimited list of prepositions to match.
 */
public PrepositionMatchingRule(String prepositions) {
    matchingFilter = Arrays.asList(prepositions.split(" "));
}

/**
 *
 */
@Override
public void match(DictionaryRepository dictionaryRepository, NLPText
verb,
    ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException {
    NLPText word = textIterator.next();
    if (word.is(PartOfSpeech.PREPOSITION)) {
        if (matchingFilter.size() > 0) {
            for (String matchWord : matchingFilter) {
                if (word.getText().equalsIgnoreCase(matchWord)) {
                    log.debug("matched: " + word.getText());
                    return;
                }
            }
        } else {
            log.debug("matched: " + word.getText());
            return;
        }
    }
    throw SemanticRoleLabelerException.matchFailed(this, word);
}

@Override
public String toString() {
    return getClass().getSimpleName() + ":" + matchingFilter;
}

```

primaryverbphrasefinder.java

```

/*
 * $Id: PrimaryVerbPhraseFinder.java,v 1.1 2009/02/06 11:49:16 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * @author ron
 */
public class PrimaryVerbPhraseFinder implements NLPPProcessor<NLPText>
{

    /**
     * if the text is a SENTENCE or CLAUSE return the primary verb
     * phrase.
     */
    @Override
    public NLPText process(NLPText text) {
        // TODO: use a processor to mark the primary verb based on verbnet
        // syntax structures if available
        NLPText clause = null;
        if (text.is(GrammaticalStructureLevel.SENTENCE)) {
            // first clause holds the subject?
            for (NLPText child : text.getChildren()) {
                if (child.is(GrammaticalStructureLevel.CLAUSE)) {
                    clause = child;
                    break;
                }
            }
        } else if (text.is(GrammaticalStructureLevel.CLAUSE)) {
            clause = text;
        }
        if (clause != null) {
            for (NLPText child : clause.getChildren()) {
                if (child.is(GrammaticalStructureLevel.PHRASE) &&
child.is(ParseTag.VP)) {
                    return child;
                }
            }
        }
        return null;
    }
}

```

project.java

```
/*
 * $Id: Project.java,v 1.6 2008/06/20 02:36:49 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import edu.harvard.fas.rregan.requel.OrganizedEntity;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public interface Project extends ProjectOrDomain, Annotatable,
Comparable<Project>, OrganizedEntity {

    /**
     * @return The current status of the project.
     */
    public String getStatus();

    /**
     * @param user
     * @return the stakeholder representation of the supplied user.
     */
    public Stakeholder getUserStakeholder(User user);
}
```

projectassistant.java

```
/*
 * $Id: ProjectAssistant.java,v 1.11 2009/01/23 09:54:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import java.util.Collection;
import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Goal;
```

```
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.TextEntity;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * The Project Assistant analyzes the entities of a project via entity
specific
 * assistants.
 *
 * @author ron
 */
public class ProjectAssistant {
    private static final Logger log =
Logger.getLogger(ProjectAssistant.class);

    private final LexicalAssistant lexicalAssistant;
    private final User user;

    /**
     * @param lexicalAssistant -
     *           assistant for analyzing text for spelling, terms and
other
     *           word oriented analysis.
     * @param user -
     *           the user to user as the creator of the annotation
entities.
     */
    public ProjectAssistant(LexicalAssistant lexicalAssistant, User user)
    {
        this.lexicalAssistant = lexicalAssistant;
        this.user = user;
    }

    /**
     * Analyze all the entities in the Project.
     *
     * @param project
     */
    public void analyze(Project project) {
        log.debug("analyzing Project: " + project);
        analyzeGoals(project.getGoals());
        analyzeStories(project.getStories());
        analyzeActors(project.getActors());
        analyzeUseCases(project.getUseCases());
    }
}
```

```

private void analyzeGoals(Collection<Goal> entities) {
    GoalAssistant assistant = new GoalAssistant(lexicalAssistant, user);
    for (TextEntity entity : entities) {
        try {
            assistant.setEntity(entity);
            assistant.analyze();
        } catch (Exception e) {
            log.error("Exception during analysis: " + e, e);
        }
    }
}

private void analyzeStories(Collection<Story> entities) {
    StoryAssistant assistant = new StoryAssistant(lexicalAssistant,
                                                 user);
    for (TextEntity entity : entities) {
        try {
            assistant.setEntity(entity);
            assistant.analyze();
        } catch (Exception e) {
            log.error("Exception during analysis: " + e, e);
        }
    }
}

private void analyzeActors(Collection<Actor> entities) {
    ActorAssistant assistant = new ActorAssistant(lexicalAssistant,
                                                user);
    for (TextEntity entity : entities) {
        try {
            assistant.setEntity(entity);
            assistant.analyze();
        } catch (Exception e) {
            log.error("Exception during analysis: " + e, e);
        }
    }
}

private void analyzeUseCases(Collection<UseCase> entities) {
    ActorAssistant actorAssistant = new ActorAssistant(lexicalAssistant,
                                                       user);
    ScenarioAssistant scenarioAssistant = new ScenarioAssistant(lexicalAssistant,
                                                               user);
    UseCaseAssistant assistant = new UseCaseAssistant(lexicalAssistant,
                                                      scenarioAssistant,
                                                      actorAssistant, user);
}

```

```

for (TextEntity entity : entities) {
    try {
        assistant.setEntity(entity);
        assistant.analyze();
    } catch (Exception e) {
        log.error("Exception during analysis: " + e, e);
    }
}
}

```

projectcommandfactory.java

```

/*
 * $Id: ProjectCommandFactory.java,v 1.24 2009/03/23 11:02:58 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.command.CommandFactory;

/**
 * @author ron
 */
public interface ProjectCommandFactory extends CommandFactory {

    /**
     * @return a new EditProjectCommand for creating or editing a
     * project.
     */
    public EditProjectCommand newEditProjectCommand();

    /**
     * @return a new ExportProjectCommand for exporting a project to xml.
     */
    public ExportProjectCommand newExportProjectCommand();

    /**
     * @return a new ImportProjectCommand for importing a new or updating
     * an
     *      existing project from xml.
     */
    public ImportProjectCommand newImportProjectCommand();

    /**
     *

```

```

 * @return a new EditStakeholderCommand for creating or editing a
 * stakeholder.
 */
public EditStakeholderCommand newEditStakeholderCommand();

/**
 * @return a new EditGoalCommand for creating or editing a goal.
 */
public EditGoalCommand newEditGoalCommand();

/**
 * @return a new EditGoalRelationCommand for creating or editing a
 * relationship between goals.
 */
public EditGoalRelationCommand newEditGoalRelationCommand();

/**
 * @return a new AddGoalToGoalContainerCommand for adding a goal to a
goal
 * container.
 */
public AddGoalToGoalContainerCommand
newAddGoalToGoalContainerCommand();

/**
 * @return a new RemoveGoalFromGoalContainerCommand for removing a
goal from
 * a goal container.
 */
public RemoveGoalFromGoalContainerCommand
newRemoveGoalFromGoalContainerCommand();

/**
 * @return a new EditStoryCommand for creating or editing a Story.
 */
public EditStoryCommand newEditStoryCommand();

/**
 * @return a new AddStoryToStoryContainerCommand for adding a Story
to a
 * Story container.
 */
public AddStoryToStoryContainerCommand
newAddStoryToStoryContainerCommand();

/**
 * @return a new RemoveStoryFromStoryContainerCommand for removing a
Story
 * from a Story container.
 */
public RemoveStoryFromStoryContainerCommand
newRemoveStoryFromStoryContainerCommand();

/**
 * @return a new EditActorCommand for creating or editing an Actor.
 */
public EditActorCommand newEditActorCommand();

/**
 * @return a new AddActorToActorContainerCommand for adding an Actor
to an
 * Actor container.
 */
public AddActorToActorContainerCommand
newAddActorToActorContainerCommand();

/**
 * @return a new RemoveActorFromActorContainerCommand for removing an
Actor
 * from an Actor container.
 */
public RemoveActorFromActorContainerCommand
newRemoveActorFromActorContainerCommand();

/**
 * @return a new EditGlossaryTermCommand for creating or editing a
glossary
 * term.
 */
public EditGlossaryTermCommand newEditGlossaryTermCommand();

/**
 * @return a new EditAddWordToGlossaryPositionCommand for creating or
 * editing a position for adding a term to the project
glossary.
 */
public EditAddWordToGlossaryPositionCommand
newEditAddWordToGlossaryPositionCommand();

/**
 * @return a new EditAddActorToProjectPositionCommand for creating or
 * editing a position for adding an actor to the project.
 */

```

```

public EditAddActorToProjectPositionCommand
newEditAddActorToProjectPositionCommand();

/**
 * @return a new ReplaceGlossaryTermCommand for replacing the term
name in
 *      the referring entities with the name of the canonical
term.
 */
public ReplaceGlossaryTermCommand newReplaceGlossaryTermCommand();

/**
 * @return a new EditUseCaseCommand for creating or editing a
usecase.
 */
public EditUseCaseCommand newEditUseCaseCommand();

/**
 * @return a new EditScenarioCommand for creating or editing a
scenario.
 */
public EditScenarioCommand newEditScenarioCommand();

/**
 * @return a new EditScenarioStepCommand for creating or editing a
scenario
 *      step.
 */
public EditScenarioStepCommand newEditScenarioStepCommand();

/**
 * @return a new CopyUseCaseCommand for creating a copy of an
existing
 *      usecase.
 */
public CopyUseCaseCommand newCopyUseCaseCommand();

/**
 * @return a new CopyStoryCommand for creating a copy of an existing
story.
 */
public CopyStoryCommand newCopyStoryCommand();

/**
 * @return a new CopyActorCommand for creating a copy of an existing
actor.
 */

```

```

public CopyActorCommand newCopyActorCommand();

/**
 * @return a new CopyGoalCommand for creating a copy of an existing
goal.
 */
public CopyGoalCommand newCopyGoalCommand();

/**
 * @return a new CopyScenarioCommand for creating a copy of an
existing
 *      scenario.
 */
public CopyScenarioCommand newCopyScenarioCommand();

/**
 * @return a new CopyScenarioStepCommand for creating a copy of an
existing
 *      scenario.
 */
public CopyScenarioStepCommand newCopyScenarioStepCommand();

/**
 * @return a new ConvertStepToScenarioCommand for converting a step
into a
 *      scenario.
 */
public ConvertStepToScenarioCommand
newConvertStepToScenarioCommand();

/**
 * @return a new EditReportGeneratorCommand for creating or editing a
report
 *      generator.
 */
public EditReportGeneratorCommand newEditReportGeneratorCommand();

/**
 * @return a new GenerateReportCommand for generating a report for a
project.
 */
public GenerateReportCommand newGenerateReportCommand();

public DeleteActorCommand newDeleteActorCommand();

public DeleteGlossaryTermCommand newDeleteGlossaryTermCommand();

```

```

public DeleteReportGeneratorCommand
newDeleteReportGeneratorCommand();

public DeleteScenarioCommand newDeleteScenarioCommand();

public DeleteScenarioStepCommand newDeleteScenarioStepCommand();

public DeleteStakeholderCommand newDeleteStakeholderCommand();

public DeleteStoryCommand newDeleteStoryCommand();

public DeleteUseCaseCommand newDeleteUseCaseCommand();

public DeleteGoalCommand newDeleteGoalCommand();

public DeleteGoalRelationCommand newDeleteGoalRelationCommand();

public RemoveUnneedLexicalIssuesCommand
newRemoveUnneedLexicalIssuesCommand();
}

```

projectcommandfactoryimpl.java

```

/*
 * $Id: ProjectCommandFactoryImpl.java,v 1.28 2009/01/19 09:32:22
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.AbstractCommandFactory;
import edu.harvard.fas.rregan.command.CommandFactoryStrategy;
import
edu.harvard.fas.rregan.requel.project.command.AddActorToActorContainer
Command;
import
edu.harvard.fas.rregan.requel.project.command.AddGoalToGoalContainerCo
mmand;
import
edu.harvard.fas.rregan.requel.project.command.AddStoryToStoryContainer
Command;

```

```

import
edu.harvard.fas.rregan.requel.project.command.ConvertStepToScenarioCom
mand;
import edu.harvard.fas.rregan.requel.project.command.CopyActorCommand;
import edu.harvard.fas.rregan.requel.project.command.CopyGoalCommand;
import
edu.harvard.fas.rregan.requel.project.command.CopyScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.CopyScenarioStepCommand;
import edu.harvard.fas.rregan.requel.project.command.CopyStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.CopyUseCaseCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGlossaryTermComman
d;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGoalCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteGoalRelationComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.DeleteReportGeneratorCom
mand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteScenarioStepComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.DeleteStakeholderCommand
;
import
edu.harvard.fas.rregan.requel.project.command.DeleteStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteUseCaseCommand;
import edu.harvard.fas.rregan.requel.project.command.EditActorCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditAddActorToProjectPos
itionCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditAddWordToGlossaryPos
itionCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditGlossaryTermCommand;
import edu.harvard.fas.rregan.requel.project.command.EditGoalCommand;

```

```

import
edu.harvard.fas.rregan.requel.project.command.EditGoalRelationCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditReportGeneratorComma
nd;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioStepCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditStakeholderCommand;
import edu.harvard.fas.rregan.requel.project.command.EditStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditUseCaseCommand;
import
edu.harvard.fas.rregan.requel.project.command.ExportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.GenerateReportCommand;
import
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveActorFromActorCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.project.command.RemoveGoalFromGoalContai
nerCommand;
import
edu.harvard.fas.rregan.requel.project.command.RemoveStoryFromStoryCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.project.command.RemoveUnneedLexicalIssue
sCommand;
import
edu.harvard.fas.rregan.requel.project.command.ReplaceGlossaryTermComma
nd;

/**
 * An implementation of ProjectCommandFactory.
 *
 * @author ron
 */
@Controller("projectCommandFactory")
@Scope("singleton")

```

```

public class ProjectCommandFactoryImpl extends AbstractCommandFactory
implements
ProjectCommandFactory {

    /**
     * @param creationStrategy -
     *          the strategy to use for creating new Command instances
     */
    @Autowired
    public ProjectCommandFactoryImpl(CommandFactoryStrategy
creationStrategy) {
        super(creationStrategy);
    }

    @Override
    public EditProjectCommand newEditProjectCommand() {
        return (EditProjectCommand)
getCreationStrategy().newInstance(EditProjectCommandImpl.class);
    }

    @Override
    public ExportProjectCommand newExportProjectCommand() {
        return (ExportProjectCommand) getCreationStrategy().newInstance(
            ExportProjectCommandImpl.class);
    }

    @Override
    public ImportProjectCommand newImportProjectCommand() {
        return (ImportProjectCommand) getCreationStrategy().newInstance(
            ImportProjectCommandImpl.class);
    }

    @Override
    public EditStakeholderCommand newEditStakeholderCommand() {
        return (EditStakeholderCommand) getCreationStrategy().newInstance(
            EditStakeholderCommandImpl.class);
    }

    @Override
    public EditGoalCommand newEditGoalCommand() {
        return (EditGoalCommand)
getCreationStrategy().newInstance(EditGoalCommandImpl.class);
    }

    @Override
    public EditGoalRelationCommand newEditGoalRelationCommand() {
        return (EditGoalRelationCommand) getCreationStrategy().newInstance(

```

```

        EditGoalRelationCommandImpl.class);
    }

@Override
public AddGoalToGoalContainerCommand
newAddGoalToGoalContainerCommand() {
    return (AddGoalToGoalContainerCommand)
getCreationStrategy().newInstance(
    AddGoalToGoalContainerCommandImpl.class);
}

@Override
public RemoveGoalFromGoalContainerCommand
newRemoveGoalFromGoalContainerCommand() {
    return (RemoveGoalFromGoalContainerCommand)
getCreationStrategy().newInstance(
    RemoveGoalFromGoalContainerCommandImpl.class);
}

@Override
public EditStoryCommand newEditStoryCommand() {
    return (EditStoryCommand)
getCreationStrategy().newInstance(EditStoryCommandImpl.class);
}

@Override
public AddStoryToStoryContainerCommand
newAddStoryToStoryContainerCommand() {
    return (AddStoryToStoryContainerCommand)
getCreationStrategy().newInstance(
    AddStoryToStoryContainerCommandImpl.class);
}

@Override
public RemoveStoryFromStoryContainerCommand
newRemoveStoryFromStoryContainerCommand() {
    return (RemoveStoryFromStoryContainerCommand)
getCreationStrategy().newInstance(
    RemoveStoryFromStoryContainerCommandImpl.class);
}

@Override
public EditActorCommand newEditActorCommand() {
    return (EditActorCommand)
getCreationStrategy().newInstance(EditActorCommandImpl.class);
}

```

```

@Override
public AddActorToActorContainerCommand
newAddActorToActorContainerCommand() {
    return (AddActorToActorContainerCommand)
getCreationStrategy().newInstance(
    AddActorToActorContainerCommandImpl.class);
}

@Override
public RemoveActorFromActorContainerCommand
newRemoveActorFromActorContainerCommand() {
    return (RemoveActorFromActorContainerCommand)
getCreationStrategy().newInstance(
    RemoveActorFromActorContainerCommandImpl.class);
}

public EditGlossaryTermCommand newEditGlossaryTermCommand() {
    return (EditGlossaryTermCommand) getCreationStrategy().newInstance(
        EditGlossaryTermCommandImpl.class);
}

public EditAddWordToGlossaryPositionCommand
newEditAddWordToGlossaryPositionCommand() {
    return (EditAddWordToGlossaryPositionCommand)
getCreationStrategy().newInstance(
    EditAddWordToGlossaryPositionCommandImpl.class);
}

@Override
public EditAddActorToProjectPositionCommand
newEditAddActorToProjectPositionCommand() {
    return (EditAddActorToProjectPositionCommand)
getCreationStrategy().newInstance(
    EditAddActorToProjectPositionCommandImpl.class);
}

@Override
public ReplaceGlossaryTermCommand newReplaceGlossaryTermCommand() {
    return (ReplaceGlossaryTermCommand)
getCreationStrategy().newInstance(
    ReplaceGlossaryTermCommandImpl.class);
}

@Override
public EditUseCaseCommand newEditUseCaseCommand() {
    return (EditUseCaseCommand)
getCreationStrategy().newInstance(EditUseCaseCommandImpl.class);
}

```

```

}

@Override
public CopyUseCaseCommand newCopyUseCaseCommand() {
    return (CopyUseCaseCommand)
        getCreationStrategy().newInstance(CopyUseCaseCommandImpl.class);
}

@Override
public CopyStoryCommand newCopyStoryCommand() {
    return (CopyStoryCommand)
        getCreationStrategy().newInstance(CopyStoryCommandImpl.class);
}

@Override
public CopyActorCommand newCopyActorCommand() {
    return (CopyActorCommand)
        getCreationStrategy().newInstance(CopyActorCommandImpl.class);
}

@Override
public CopyGoalCommand newCopyGoalCommand() {
    return (CopyGoalCommand)
        getCreationStrategy().newInstance(CopyGoalCommandImpl.class);
}

@Override
public EditScenarioCommand newEditScenarioCommand() {
    return (EditScenarioCommand) getCreationStrategy().newInstance(
        EditScenarioCommandImpl.class);
}

@Override
public CopyScenarioCommand newCopyScenarioCommand() {
    return (CopyScenarioCommand) getCreationStrategy().newInstance(
        CopyScenarioCommandImpl.class);
}

@Override
public CopyScenarioStepCommand newCopyScenarioStepCommand() {
    return (CopyScenarioStepCommand) getCreationStrategy().newInstance(
        CopyScenarioStepCommandImpl.class);
}

@Override
public EditScenarioStepCommand newEditScenarioStepCommand() {
    return (EditScenarioStepCommand) getCreationStrategy().newInstance(
        EditScenarioStepCommandImpl.class);
}

@Override
public ConvertStepToScenarioCommand newConvertStepToScenarioCommand()
{
    return (ConvertStepToScenarioCommand)
        getCreationStrategy().newInstance(
            ConvertStepToScenarioCommandImpl.class);
}

@Override
public EditReportGeneratorCommand newEditReportGeneratorCommand() {
    return (EditReportGeneratorCommand)
        getCreationStrategy().newInstance(
            EditReportGeneratorCommandImpl.class);
}

@Override
public GenerateReportCommand newGenerateReportCommand() {
    return (GenerateReportCommand) getCreationStrategy().newInstance(
        GenerateReportCommandImpl.class);
}

@Override
public DeleteActorCommand newDeleteActorCommand() {
    return (DeleteActorCommand)
        getCreationStrategy().newInstance(DeleteActorCommandImpl.class);
}

@Override
public DeleteGlossaryTermCommand newDeleteGlossaryTermCommand() {
    return (DeleteGlossaryTermCommand)
        getCreationStrategy().newInstance(
            DeleteGlossaryTermCommandImpl.class);
}

@Override
public DeleteReportGeneratorCommand newDeleteReportGeneratorCommand()
{
    return (DeleteReportGeneratorCommand)
        getCreationStrategy().newInstance(
            DeleteReportGeneratorCommandImpl.class);
}

@Override
public DeleteScenarioCommand newDeleteScenarioCommand() {
}

```

```

        return (DeleteScenarioCommand) getCreationStrategy().newInstance(
            DeleteScenarioCommandImpl.class);
    }

@Override
public DeleteScenarioStepCommand newDeleteScenarioStepCommand() {
    return (DeleteScenarioStepCommand)
getCreationStrategy().newInstance(
    DeleteScenarioStepCommandImpl.class);
}

@Override
public DeleteStakeholderCommand newDeleteStakeholderCommand() {
    return (DeleteStakeholderCommand) getCreationStrategy().newInstance(
        DeleteStakeholderCommandImpl.class);
}

@Override
public DeleteStoryCommand newDeleteStoryCommand() {
    return (DeleteStoryCommand)
getCreationStrategy().newInstance(DeleteStoryCommandImpl.class);
}

@Override
public DeleteUseCaseCommand newDeleteUseCaseCommand() {
    return (DeleteUseCaseCommand) getCreationStrategy().newInstance(
        DeleteUseCaseCommandImpl.class);
}

@Override
public DeleteGoalCommand newDeleteGoalCommand() {
    return (DeleteGoalCommand)
getCreationStrategy().newInstance(DeleteGoalCommandImpl.class);
}

@Override
public DeleteGoalRelationCommand newDeleteGoalRelationCommand() {
    return (DeleteGoalRelationCommand)
getCreationStrategy().newInstance(
    DeleteGoalRelationCommandImpl.class);
}

@Override
public RemoveUnneedLexicalIssuesCommand
newRemoveUnneedLexicalIssuesCommand() {
    return (RemoveUnneedLexicalIssuesCommand)
getCreationStrategy().newInstance(

```

```

        RemoveUnneedLexicalIssuesCommandImpl.class);
    }
}

```

projectimpl.java

```

/*
 * $Id: ProjectImpl.java,v 1.31 2009/01/10 11:08:58 rregan Exp $
 * Copyright (c) 2007 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshallingContext;

import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import
edu.harvard.fas.rregan.requel.annotation.impl.AbstractAnnotation;
import edu.harvard.fas.rregan.requel.annotation.impl.PositionImpl;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;

```

```

import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.impl.OrganizationImpl;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBOrganizedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.impl.ProjectImpl")
@XmlRootElement(name = "project", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "project", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ProjectImpl extends AbstractProjectOrDomain implements
Project {
    static final long serialVersionUID = 0L;

    private Organization organization;
    private String status;
    private Set<Annotation> annotations = new TreeSet<Annotation>();

    /**
     * @param name
     * @param creator
     * @param organization
     */
    public ProjectImpl(String name, User creator, Organization
organization) {
        super(ProjectImpl.class.getName(), name, creator);
        setOrganization(organization);
        setStatus("New");
    }

    protected ProjectImpl() {
        super();
        // for JAXB and hibernate
    }

    @Transient
    public String getDescription() {
        return "Project: " + getName();
    }

}
}

@XmlElementRef(type = OrganizationImpl.class)
@ManyToOne(targetEntity = OrganizationImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH })
public Organization getOrganization() {
    return organization;
}

public void setOrganization(Organization organization) {
    this.organization = organization;
}

@XmlElement(name = "status", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getStatus() {
    return status;
}

protected void setStatus(String status) {
    this.status = status;
}

/**
 * This is for JAXB to only output a single definition for each
annotation.
 * In the export file the entities include references to the
annotations
 * instead of the annotations themselves.
 *
 * @return all the annotations of all project entities in a single
set.
 */
@XmlElementWrapper(name = "annotations", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = AbstractAnnotation.class)
@Transient
public Set<Annotation> getAllProjectEntityAnnotations() {
    Set<Annotation> annotations = new HashSet<Annotation>();
    annotations.addAll(getAnnotations());
    for (ProjectOrDomainEntity entity : getProjectEntities()) {
        annotations.addAll(entity.getAnnotations());
    }
    return annotations;
}

```

```

/**
 * This is for JAXB to only output a single definition for each issue
 * position. In the export file the issues include references to the
 * position instead of the position themselves.
 *
 * @return all the annotations of all project entities in a single
set.
 */
@XmlElementWrapper(name = "positions", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = PositionImpl.class)
@Transient
public Set<Position> getAllProjectEntityIssuePositions() {
    Set<Annotation> annotations = getAllProjectEntityAnnotations();
    Set<Position> positions = new HashSet<Position>();
    for (Annotation annotation : annotations) {
        if (annotation instanceof Issue) {
            positions.addAll(((Issue) annotation).getPositions());
        }
    }
    return positions;
}

@XmlElementWrapper(name = "annotations", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel", required = false)
// changed xml mapping to output references to annotations instead of
the
// annotations directly because
// an annotation may be shared by multiple entities causing
duplicates on
// import. this makes report
// generating via xslt more complicated because of the indirection.
@XmlIDREF
@XmlElement(name = "annotationRef", type = AbstractAnnotation.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
// @XmlElementRef(type = AbstractAnnotation.class)
@ManyToMany(targetEntity = AbstractAnnotation.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "project_annotations")
public Set<Annotation> getAnnotations() {
    return annotations;
}

protected void setAnnotations(Set<Annotation> annotations) {
    this.annotations = annotations;
}

```

```

@Transient
public Stakeholder getUserStakeholder(User user) {
    for (Stakeholder stakeholder : getStakeholders()) {
        if (user.equals(stakeholder.getUser())) {
            return stakeholder;
        }
    }
    return null;
}

@Override
public int compareTo(Project o) {
    if (getName() == null) {
        return -1;
    } else if ((o == null) || (o.getName() == null)) {
        return 1;
    }
    return getName().compareToIgnoreCase(o.getName());
}

/**
 * This is for JAXB to patchup existing persistent objects for the
objects
 * that are attached directly to this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */
@Override
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser) {
    super.afterUnmarshal(userRepository, defaultCreatedByUser);
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBOrganizedEntityPatcher(userRepository, this));
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {
                if (ProjectImpl.this.getCreatedBy() != null) {
                    ProjectUserRole projectUserRole =
ProjectImpl.this.getCreatedBy()
                        .getRoleForType(ProjectUserRole.class);

```

```

        if (projectUserRole != null) {
            projectUserRole.getActiveProjects().add(ProjectImpl.this);
        }
    } catch (RuntimeException e) {
        throw e;
    } catch (Exception e) {
        throw new SAXException2(e);
    }
}
});
}

/**
 * @param name -
 *          project name that over rides the name in the xml file.
 */
public void afterUnmarshal(String name) {
    if ((name != null) && !name.trim().equals("")) {
        setName(name);
    }
}
}

```

projectmanagementpanelnames.java

```

/*
 * $Id: ProjectManagementPanelNames.java,v 1.6 2008/12/31 11:49:35
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

/**
 * This defines constants for all the panel names used in the project
management
 * package. These project names will be referenced by OpenPanelEvents,
the
 * panels that implement the named panels, and PanelFactories for
creating the
 * panels (typically in the spring navigationContext.xml configuration
file.)
 *
 * @author ron
 */
public interface ProjectManagementPanelNames {

```

```

    /**
     * The name of the panel to use for editing the project overview. A
panel
     * implementation that is to be used for the project overview should
     * register itself with this name, the panel factory for creating it
and
     * events for opening it.
     */
    public static final String PROJECT_OVERVIEW_PANEL_NAME =
"projectOverview";

    /**
     * The name of the panel for navigating project stakeholders.
     */
    public static final String PROJECT_STAKEHOLDERS_NAVIGATOR_PANEL_NAME =
"projectStakeholdersNavigator";

    /**
     * The name of the panel for navigating project use cases.
     */
    public static final String PROJECT_USE_CASES_NAVIGATOR_PANEL_NAME =
"projectUseCasesNavigator";

    /**
     * The name of the panel for selecting a use case in a project.
     */
    public static final String PROJECT_USE_CASES_SELECTOR_PANEL_NAME =
"projectUseCasesSelector";

    /**
     * The name of the panel for navigating project scenarios.
     */
    public static final String PROJECT_SCENARIOS_NAVIGATOR_PANEL_NAME =
"projectScenariosNavigator";

    /**
     * The name of the panel for selecting a scenario in a project.
     */
    public static final String PROJECT_SCENARIO_SELECTOR_PANEL_NAME =
"projectScenarioSelector";

    /**
     * The name of the panel for navigating project stories.
     */
    public static final String PROJECT_STORIES_NAVIGATOR_PANEL_NAME =
"projectStoriesNavigator";

```

```

/**
 * The name of the panel for selecting a story in a project.
 */
public static final String PROJECT_STORY_SELECTOR_PANEL_NAME =
"projectStorySelector";

/**
 * The name of the panel for navigating project actors.
 */
public static final String PROJECT_ACTORS_NAVIGATOR_PANEL_NAME =
"projectActorsNavigator";

/**
 * The name of the panel for selecting an actor in a project.
 */
public static final String PROJECT_ACTORS_SELECTOR_PANEL_NAME =
"projectActorsSelector";

/**
 * The name of the panel for navigating project goals.
 */
public static final String PROJECT_GOALS_NAVIGATOR_PANEL_NAME =
"projectGoalsNavigator";

/**
 * The name of the panel for selecting a goal in a project.
 */
public static final String PROJECT_GOALS_SELECTOR_PANEL_NAME =
"projectGoalsSelector";

/**
 * The name of the panel for navigating project stakeholders.
 */
public static final String
PROJECT_GLOSSARY_TERMS_NAVIGATOR_PANEL_NAME =
"projectGlossaryTermsNavigator";

/**
 * The name of the panel for navigating project goals.
 */
public static final String PROJECT_GLOSSARY_TERM_SELECTOR_PANEL_NAME
= "projectGlossaryTermsSelector";

/**
 * The name of the panel for navigating project stakeholders.
*/

```

```

    public static final String PROJECT_REPORTS_NAVIGATOR_PANEL_NAME =
"projectReportsNavigator";

    /**
     * The name of the panel for navigating the open issues for all
     entities in
     * a project.
     */
    public static final String PROJECT_OPEN_ISSUES_NAVIGATOR_PANEL_NAME =
"projectOpenIssuesNavigator";
}

```

projectnameinuseexception.java

```

/*
 * $Id: ProjectNameInUseException.java,v 1.3 2008/12/13 00:40:02
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.exception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.project.Project;

/**
 * @author ron
 */
public class ProjectNameInUseException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_NAME = "A project already exists with
name '%s'";

    /**
     * @param name -
     *          the name used to find a project that doesn't exist
     * @return
     */
    public static ProjectNameInUseException forName(String name) {
        return new ProjectNameInUseException(Project.class, null, "name",
name,
            EntityExceptionActionType.Creating, MSG_FOR_NAME, name);
    }
}

```

```

/**
 * @param format
 * @param args
 */
protected ProjectNameInUseException(Class<?> entityType, Object
entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected ProjectNameInUseException(Throwable cause, Class<?>
entityType, Object entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
        messageArgs);
}

```

projectnavigatorpanel.java

```

/*
 * $Id: ProjectNavigatorPanel.java,v 1.2 2008/12/31 11:49:35 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.util.Set;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;

import org.apache.log4j.Logger;

```

```

import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTreePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class ProjectNavigatorPanel extends NavigatorTreePanel {
    private static final Logger log =
Logger.getLogger(ProjectNavigatorPanel.class);
    static final long serialVersionUID = 0;

    /**
     * Property name to use in the ProjectNavigatorPanel.properties to
set the
     * lable on the new project button.
     */
    public static final String PROP_NEW_PROJECT_BUTTON_LABEL =
"NewProjectButton.Label";

    /**
     * Property name to use in the ProjectNavigatorPanel.properties to
set the
     * lable on the find/open projects button.
     */
    public static final String PROP_FIND_PROJECT_BUTTON_LABEL =
"FindProjectButton.Label";

    /**
     * @param treeNodeFactories
     */
    public ProjectNavigatorPanel(Set<NavigatorTreeNodeFactory>
treeNodeFactories) {
        super(ProjectNavigatorPanel.class.getName(), treeNodeFactories,
ProjectUserRole.class);
    }
}

```

```

}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}

@Override
public void setup() {
    ProjectUserRole projectUserRole = ((User) getTargetObject())
        .getRoleForType(ProjectUserRole.class);

    if (projectUserRole.canCreateProjects()) {
        String newProjectButtonLabel =
        getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_PROJECT_BUTTON_LABEL, "New Project");

        NavigationEvent openProjectOverview = new OpenPanelEvent(this,
        PanelActionType.Editor,
        null, Project.class, "projectOverview",
        WorkflowDisposition.NewFlow);

        NavigatorButton newProjectButton = new
        NavigatorButton(newProjectButtonLabel,
            getEventDispatcher(), openProjectOverview);

        newProjectButton.setStyleName(STYLE_NAME_DEFAULT);
        Row newProjectButtonWrapper = new Row();
        newProjectButtonWrapper.setInsets(new Insets(5));
        newProjectButtonWrapper
            .setAlignment(new Alignment(Alignment.CENTER,
            Alignment.DEFAULT));
        newProjectButtonWrapper.add(newProjectButton);
        add(newProjectButtonWrapper);
    }

    String findProjectButtonLabel =
    getResourceBundleHelper(getLocale()).getString(
        PROP_FIND_PROJECT_BUTTON_LABEL, "Find Project");

    NavigationEvent openProjectSearch = new OpenPanelEvent(this,
    PanelActionType.Selector,
    getApp().getUser(), Project.class, "projectSearch",
    WorkflowDisposition.NewFlow);
}

```

```

    NavigatorButton findProjectButton = new
    NavigatorButton(findProjectButtonLabel,
        getEventDispatcher(), openProjectSearch);

    findProjectButton.setStyleName(STYLE_NAME_DEFAULT);
    Row findProjectButtonWrapper = new Row();
    findProjectButtonWrapper.setInsets(new Insets(5));
    findProjectButtonWrapper.setAlignment(new
    Alignment(Alignment.CENTER, Alignment.DEFAULT));
    findProjectButtonWrapper.add(findProjectButton);
    // TODO: project search is not implemented.
    // add(findProjectButtonWrapper);

    super.setup();
}
}

```

projectnavigatortreenodefactory.java

```

/*
 * $Id: ProjectNavigatorTreeNodeFactory.java,v 1.9 2009/03/27 13:43:15
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.Label;
import nextapp.echo2.app.event.ActionEvent;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeUp
dateListener;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
@Component("projectNavigatorTreeNodeFactory")
@Scope("singleton")
public class ProjectNavigatorTreeNodeFactory extends
AbstractNavigatorTreeNodeFactory {

    /**
     * The property name to use to control the label on the stakeholders
     * node
     * generated by the factory.
     */
    public final static String PROP_STAKEHOLDERS_NODE_LABEL =
"StakeholdersNodeLabel";

    /**
     * The property name to use to control the label on the goals node
     * generated
     * by the factory.
     */
    public final static String PROP_GOALS_NODE_LABEL = "GoalsNodeLabel";

    /**
     * The property name to use to control the label on the stories node
     * generated by the factory.
     */
    public final static String PROP_STORIES_NODE_LABEL =
"StoriesNodeLabel";
}

    /**
     * The property name to use to control the label on the stories node
     * generated by the factory.
     */
    public final static String PROP_ACTORS_NODE_LABEL =
"ActorsNodeLabel";

    /**
     * The property name to use to control the label on the stories node
     * generated by the factory.
     */
    public final static String PROP_USE_CASES_NODE_LABEL =
"UseCasesNodeLabel";

    /**
     * The property name to use to control the label on the stories node
     * generated by the factory.
     */
    public final static String PROP_SCENARIOS_NODE_LABEL =
"ScenariosNodeLabel";

    /**
     * The property name to use to control the label on the glossary
     * terms node
     * generated by the factory.
     */
    public final static String PROP_GLOSSARY_TERMS_NODE_LABEL =
"GlossaryTermNodeLabel";

    /**
     * The property name to use to control the label on the glossary
     * terms node
     * generated by the factory.
     */
    public final static String PROP_REPORT_GENERATORS_NODE_LABEL =
"ReportGeneratorsNodeLabel";

    /**
     * The property name to use to control the label on the glossary
     * terms node
     * generated by the factory.
     */
    public final static String PROP_OPEN_ISSUES_NODE_LABEL =
"OpenIssuesNodeLabel";
}

```

```

 * @param eventDispatcher
 */
public ProjectNavigatorTreeNodeFactory() {
    super(ProjectNavigatorTreeNodeFactory.class.getName(),
        Project.class);
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.tre
e.NavigatorTree,
 *          java.lang.Object)
 */
public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
    Object object) {
    Project project = (Project) object;

    NavigationEvent openProjectOverview = new OpenPanelEvent(tree,
PanelActionType.Editor,
    project, Project.class,
ProjectManagementPanelNames.PROJECT_OVERVIEW_PANEL_NAME,
    WorkflowDisposition.NewFlow);

    NavigatorTreeNode projectTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(project.getName()), openProjectOverview);

    // stakeholders
    String stakeholdersNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_STAKEHOLDERS_NODE_LABEL, "Stakeholders");

    NavigationEvent openProjectStakeholders = new OpenPanelEvent(tree,
    PanelActionType.Navigator, project, Project.class,
ProjectManagementPanelNames.PROJECT_STAKEHOLDERS_NAVIGATOR_PANEL_N
AME,
    WorkflowDisposition.NewFlow);

    NavigatorTreeNode stakeholdersTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(stakeholdersNodeLabel), openProjectStakeholders);

    projectTreeNode.add(stakeholdersTreeNode);

    // goals

```

```

        String goalsNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_GOALS_NODE_LABEL, "Goals");

    NavigationEvent openProjectGoals = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
ProjectManagementPanelNames.PROJECT_GOALS_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

    NavigatorTreeNode goalsTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(goalsNodeLabel), openProjectGoals);

    projectTreeNode.add(goalsTreeNode);

    // stories
    String storiesNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_STORIES_NODE_LABEL, "Stories");

    NavigationEvent openProjectStories = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
ProjectManagementPanelNames.PROJECT_STORIES_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

    NavigatorTreeNode storiesTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(storiesNodeLabel), openProjectStories);

    projectTreeNode.add(storiesTreeNode);

    // actors
    String actorsNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_ACTORS_NODE_LABEL, "Actors");

    NavigationEvent openProjectActors = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
ProjectManagementPanelNames.PROJECT_ACTORS_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

    NavigatorTreeNode actorsTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(actorsNodeLabel), openProjectActors);

```

```

projectTreeNode.add(actorsTreeNode);

// useCases
String useCasesNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_USE_CASES_NODE_LABEL, "Use Cases");

NavigationEvent openProjectUseCases = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
    ProjectManagementPanelNames.PROJECT_USE_CASES_NAVIGATOR_PANEL_NAME
,
    WorkflowDisposition.NewFlow);

NavigatorTreeNode useCasesTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(useCasesNodeLabel), openProjectUseCases);

projectTreeNode.add(useCasesTreeNode);

// scenarios
String scenariosNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_SCENARIOS_NODE_LABEL, "Scenarios");

NavigationEvent openProjectScenarios = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
    ProjectManagementPanelNames.PROJECT_SCENARIOS_NAVIGATOR_PANEL_NAME
,
    WorkflowDisposition.NewFlow);

NavigatorTreeNode scenariosTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(scenariosNodeLabel), openProjectScenarios);

projectTreeNode.add(scenariosTreeNode);

// glossary terms
String termsNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_GLOSSARY_TERMS_NODE_LABEL, "Terms");

NavigationEvent openProjectTerms = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
    ProjectManagementPanelNames.PROJECT_GLOSSARY_TERMS_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

NavigatorTreeNode termsTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(termsNodeLabel), openProjectTerms);

projectTreeNode.add(termsTreeNode);

// reports
String reportsNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_REPORT_GENERATORS_NODE_LABEL, "Documents");

NavigationEvent openProjectReports = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
    ProjectManagementPanelNames.PROJECT_REPORTS_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

NavigatorTreeNode reportsTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(reportsNodeLabel), openProjectReports);

projectTreeNode.add(reportsTreeNode);

// open issues
String openIssuesNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_OPEN_ISSUES_NODE_LABEL, "Open Issues");

NavigationEvent openProjectIssues = new OpenPanelEvent(tree,
PanelActionType.Navigator,
    project, Project.class,
    ProjectManagementPanelNames.PROJECT_OPEN_ISSUES_NAVIGATOR_PANEL_NAME,
    WorkflowDisposition.NewFlow);

NavigatorTreeNode openIssuesTreeNode = new
NavigatorTreeNode(eventDispatcher, project,
    new Label(openIssuesNodeLabel), openProjectIssues);

projectTreeNode.add(openIssuesTreeNode);

// TODO: when adding a new child node, update the update listener
below

```

```

projectTreeNode.setUpdateListener(new
UpdateListener(projectTreeNode));
    return projectTreeNode;
}

private static class UpdateListener implements
NavigatorTreeNodeUpdateListener {
    static final long serialVersionUID = 0L;

    private final NavigatorTreeNode projectTreeNode;

    private UpdateListener(NavigatorTreeNode projectTreeNode) {
        this.projectTreeNode = projectTreeNode;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedProject = null;
            ProjectOrDomain existingProject = (ProjectOrDomain)
projectTreeNode
                .getTargetObject();

            if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
event.getObject();
                updatedProject = entity.getProjectOrDomain();
            } else if (event.getObject() instanceof ProjectOrDomain) {
                updatedProject = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof Annotation) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if (existingProject instanceof Project) {
                        updatedProject = existingProject;
                        if (((Project) updatedProject).getAnnotations().contains(
                            updatedAnnotation)) {
                            ((Project) updatedProject).getAnnotations().remove(
                                updatedAnnotation);
                        }
                    }
                } else {
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if (annotatable instanceof ProjectOrDomainEntity) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;

```

```

    } else if
(ProjectManagementPanelNames.PROJECT_STORIES_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectGoals = new OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_STORIES_NAVIGATOR_PANEL_N
AME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectGoals);
} else if
(ProjectManagementPanelNames.PROJECT_GLOSSARY_TERMS_NAVIGATOR_PANEL_NA
ME
>equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectGoals = new OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_GLOSSARY_TERMS_NAVIGATOR_
PANEL_NAME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectGoals);
} else if
(ProjectManagementPanelNames.PROJECT_ACTORS_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectGoals = new OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_ACTORS_NAVIGATOR_PANEL_N
ME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectGoals);
} else if
(ProjectManagementPanelNames.PROJECT_USE_CASES_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectUseCases = new OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_USE_CASES_NAVIGATOR_PANEL_
NAME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectUseCases);
} else if
(ProjectManagementPanelNames.PROJECT_SCENARIOS_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectUseCases = new OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_SCENARIOS_NAVIGATOR_PANEL_
NAME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectUseCases);
} else if
(ProjectManagementPanelNames.PROJECT_REPORTS_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectReportGenerators = new
OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_REPORTS_NAVIGATOR_PANEL_N
AME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectReportGenerators);
} else if
(ProjectManagementPanelNames.PROJECT_OPEN_ISSUES_NAVIGATOR_PANEL_NAME
.equals(eventToFire.getPanelName())) {
    NavigationEvent openProjectReportGenerators = new
OpenPanelEvent(
        projectTreeNode,
        PanelActionType.Navigator,
        updatedProject,
        Project.class,
        ProjectManagementPanelNames.PROJECT_OPEN_ISSUES_NAVIGATOR_PAN
EL_NAME,
        WorkflowDisposition.NewFlow);
    child.setEventToFire(openProjectReportGenerators);
} else {
    throw new RuntimeException(
        "The ProjectNavigatorTreeNodeFactory UpdateListener is
missing configuration for node with event: "
        + eventToFire);
}
}
}

```

```
    }
}
}
}
```

projectopenissuesnavigatorpanel.java

```
/*
 * $Id: ProjectOpenIssuesNavigatorPanel.java,v 1.5 2009/02/23 08:49:50
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.Describable;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
```

```
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * A panel listing the open issues for all the entities in a project.
 *
 * @author ron
 */
public class ProjectOpenIssuesNavigatorPanel extends
NavigatorTablePanel {
private static final Logger log =
Logger.getLogger(ProjectOpenIssuesNavigatorPanel.class);
static final long serialVersionUID = 0;

private UpdateListener updateListener;
private Project project;

/**
 * Property name to use in the GoalNavigatorPanel.properties to set
the
 * label for the text of the cancel/reset button.
 */
public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

/**
 * Property name to use in the GoalNavigatorPanel.properties to set
the
```

```

 * label on the edit goal button in each row of the table.
 */
public static final String PROP_EDIT_BUTTON_LABEL =
"EditButton.Label";

/***
 * Property name to use in the GoalNavigatorPanel.properties to set
the
 * label on the view goal button in each row of the table when the
user
 * doesn't have edit permission.
 */
public static final String PROP_VIEW_BUTTON_LABEL =
"ViewButton.Label";

/***
 */
public ProjectOpenIssuesNavigatorPanel() {
    super(ProjectOpenIssuesNavigatorPanel.class.getName(),
Project.class,
        ProjectManagementPanelNames.PROJECT_OPEN_ISSUES_NAVIGATOR_PANEL_NA
ME);
    setTableConfig(createTableConfig());
}
/***
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "Open Issues: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelpergetLocale().getString(PROP_PANEL_TITLE,
    "Open Issues: {0}");
    ProjectOrDomain pod = getProject();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

```

```
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
    Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
        closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);
    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    return true;
}
```

```

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof Project) {
        project = (Project) targetObject;
        Set<Annotation> openIssues = new HashSet<Annotation>();
        openIssues.addAll((project).getAnnotations());
        for (ProjectOrDomainEntity entity : project.getProjectEntities()) {
            openIssues.addAll(entity.getAnnotations());
        }
        Iterator<Annotation> iter = openIssues.iterator();
        while (iter.hasNext()) {
            Annotation annotation = iter.next();
            if (!(annotation instanceof Issue) || ((Issue)
annotation).isResolved()) {
                iter.remove();
            }
        }
        super.setTargetObject(openIssues);
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected Project getProject() {
    return project;
}

private NavigatorTableConfig createTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Annotation annotation = (Annotation)
model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_VIEW_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_EDIT_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                    Panel ActionType.Editor, annotation, annotation.getClass(),
                    null,
                    WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
                NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                RowLayoutData rld = new RowLayoutData();
                rld.setAlignment(Alignment.ALIGN_CENTER);
                openEditorButton.setLayoutData(rld);
                return openEditorButton;
            }
        }));
}

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Annotatables",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Annotation annotation = (Annotation)
model.getBackingObject(row);
            StringBuilder sb = new StringBuilder();
            Iterator<Annotatable> iter =
annotation.getAnnotatables().iterator();
            while (iter.hasNext()) {
                Annotatable annotatable = iter.next();
                if (annotatable instanceof Describable) {
                    sb.append(((Describable) annotatable).getDescription());
                    if (iter.hasNext()) {
                        sb.append("; ");
                    }
                }
            }
            return sb.toString();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Text",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            Annotation annotation = (Annotation)
model.getBackingObject(row);
            return annotation.getText();
        }
    }));
}

```

```

        });

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Annotation annotation = (Annotation)
model.getBackingObject(row);
                return annotation.getCreatedBy().getUsername();
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Annotation annotation = (Annotation)
model.getBackingObject(row);
                DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                return formatter.format(annotation.getDateCreated());
            }
        }));
}

return tableConfig;
}

private static class UpdateListener implements ActionListener {
static final long serialVersionUID = 0L;

private final ProjectOpenIssuesNavigatorPanel panel;

private UpdateListener(ProjectOpenIssuesNavigatorPanel panel) {
    this.panel = panel;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        if (event.getObject() instanceof Project) {
            Project updatedProject = (Project) event.getObject();
            if (panel.getProject().equals(updatedProject)) {
                panel.setTargetObject(updatedProject);
            }
        }
    }
}
}

```

```

        }

    } else if (event.getObject() instanceof ProjectOrDomainEntity) {
        ProjectOrDomainEntity updatedProjectEntity =
(ProjectOrDomainEntity) event
        .getObject();
        if (updatedProjectEntity.getProjectOrDomain() instanceof Project)
{
            Project updatedProject = (Project) updatedProjectEntity
            .getProjectOrDomain();
            if (panel.getProject().equals(updatedProject)) {
                panel.setTargetObject(updatedProject);
            }
        }
    } else if (event.getObject() instanceof Annotation) {
        Annotation updatedAnnotation = (Annotation) event.getObject();
        if (event instanceof DeletedEntityEvent) {
            ((Set<Annotation>)
panel.getTargetObject()).remove(updatedAnnotation);
        } else {
            for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                if (annotatable instanceof Project) {
                    Project updatedProject = (Project) annotatable;
                    if (panel.getProject().equals(updatedProject)) {
                        panel.setTargetObject(updatedProject);
                    }
                } else if (annotatable instanceof ProjectOrDomainEntity) {
                    ProjectOrDomainEntity entity =
(ProjectOrDomainEntity)
annotatable;
                    if (panel.getProject().equals(entity.getProjectOrDomain())) {
                        panel.setTargetObject(entity.getProjectOrDomain());
                    }
                }
            }
        }
    }
}

```

projectordomain.java

```

/*
 * $Id: ProjectOrDomain.java,v 1.12 2008/12/31 11:49:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

/*
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;
import java.util.SortedSet;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * @author ron
 */
public interface ProjectOrDomain extends CreatedEntity, Describable,
GoalContainer, StoryContainer,
ActorContainer, ScenarioContainer {

/**
 * @return The name used to uniquely identify the domain or project.
 */
public String getName();

/**
 * Change the name of the domain or project.
 *
 * @param name
 */
public void setName(String name);

/**
 * @return get the description of this project or domain.
 */
public String getText();

/**
 * set the description of the project or domain.
 *
 * @param description
 */
public void setText(String description);

/**
 * @return a set of terms specific to this domain or project
 */
public SortedSet<GlossaryTerm> getGlossaryTerms();

/**

```

```

     * @return The set of all use cases defined for this domain or
project.
    */
    public Set<UseCase> getUseCases();

    /**
     * @return The set of all stakeholders defined for this domain or
project.
     *          For a domain this should only be non-user stakeholders
that will
     *          be consistent for future projects.
    */
    public Set<Stakeholder> getStakeholders();

    /**
     * @return
    */
    public Set<ProjectTeam> getTeams();

    /**
     * @return The set of report generators for the project.
    */
    public Set<ReportGenerator> getReportGenerators();

    /**
     * @return all the project entities (goals, actors, stakeholders,
etc.)
    */
    public Set<ProjectOrDomainEntity> getProjectEntities();
}


```

projectordomainentity.java

```

/*
 * $Id: ProjectOrDomainEntity.java,v 1.9 2009/03/05 08:50:47 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Comparator;
import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

```

```

import edu.harvard.fas.rregan.requel.NamedEntity;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;

/**
 * @author ron
 */
public interface ProjectOrDomainEntity extends NamedEntity,
Annotatable, CreatedEntity, Describable {

    /**
     * @return the project or domain that the entity is attached to.
     */
    public ProjectOrDomain getProjectOrDomain();

    /**
     * This is a hack because there is no way to generate a sensibly
     * unique id
     * for entities from the AbstractProjectOrDomainEntity.
     *
     * @return a string appropriate for the xml id
     */
    public String getXmlId();

    /**
     * @return The terms in the glossary that this entity refers to
     */
    public Set<GlossaryTerm> getGlossaryTerms();

    /**
     * This is a hack to get the most specific entity interface of a
     * domain
     * object, even when it is proxied.
     *
     * @return
     */
    public Class<?> getProjectOrDomainEntityInterface();

    /**
     * Compare two project entities by their description.
     */
    public static class ProjectOrDomainEntityComparator implements
        Comparator<ProjectOrDomainEntity> {

        @Override
        public int compare(ProjectOrDomainEntity o1, ProjectOrDomainEntity
o2) {
            if ((o1 == null) || (o2 == null)) {

```

```

                if (o1 == null) {
                    return -1;
                } else {
                    return 1;
                }
            }
            return o1.getDescription().compareTo(o2.getDescription());
        }
    }
}

```

projectordomainentityassistant.java

```

/*
 * $Id: ProjectOrDomainEntityAssistant.java,v 1.11 2009/03/27 07:16:07
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Assistant for analyzing entities based on the ProjectOrDomainEntity
 * interface, such as goals and stories. The assistant applies the
lexical
 * assistant to the name and text properties of each entity.
 *
 * @author ron
 */
public class ProjectOrDomainEntityAssistant extends AbstractAssistant
{
    private static final Logger log =
Logger.getLogger(ProjectOrDomainEntityAssistant.class);

    public static final String PROP_NAME = "Name";
    public static final String PROP_TEXT = "Text";
}

```

```

public static final String PROP_COMBINED = "Combined";

private final User assistantUser;
private final LexicalAssistant lexicalAssistant;

// the entity under analysis
private ProjectOrDomainEntity entity;
private final Map<String, NLPText> propertyNameToNLPText = new
HashMap<String, NLPText>();

/**
 * @param resourceBundleName -
 *          the full class name to use for the resource bundle.
 * @param lexicalAssistant -
 *          assistant for analyzing text for spelling, terms and
other
 *          word oriented analysis.
 * @param assistantUser -
 *          the user to use as the creator of the annotation
entities.
 */
public ProjectOrDomainEntityAssistant(String resourceBundleName,
    LexicalAssistant lexicalAssistant, User assistantUser) {
    super(resourceBundleName, lexicalAssistant.getCommandHandler(),
lexicalAssistant
        .getAnnotationCommandFactory(),
lexicalAssistant.getAnnotationRepository());
    this.lexicalAssistant = lexicalAssistant;
    this.assistantUser = assistantUser;
}

/**
 * @return The entity being analyzed.
 */
public ProjectOrDomainEntity getEntity() {
    return entity;
}

/**
 * Set the entity to analyze. Must be called before analyze.
 *
 * @param entity -
 *          the entity to analyze.
 * @throws IllegalArgumentException -
 *          if the supplied entity is null or the type is not
supported
 *          by the assistant.

```

```

/*
public void setEntity(ProjectOrDomainEntity entity) throws
IllegalArgumentException {
    propertyNameToNLPText.clear();
    this.entity = entity;
    setPropertyText(PROP_NAME, entity.getName());
}

/**
 * Analyze the text in the name property of the entity and notify the
 * UpdatedEntityNotifier that the entity has been updated.<br>
 * NOTE: sub classes should call super.analyze() and not
analyzeName()
 * directly so that the UpdatedEntityNotifier is called.
 *
 * @param entity
 */
public void analyze() {
    // remove any lexical annotations that apply to all the properties
    // collectively (for example glossary term issues) that are no
longer
    // relevant.
    try {
        NLPText nlpText =
getLexicalAssistant().getNLPPProcessorFactory().appendText(
            new ArrayList<NLPText>(propertyNameToNLPText.values()));
        getLexicalAssistant().removeUnneedLexicalIssues(getAssistantUser(),
            entity.getProjectOrDomain(), entity, null, nlpText);
    } catch (Exception e) {
        log.error("failed to remove irrelevant annotations for " + entity,
e);
        try {
            // TODO: adding a note with the exception is probably not
            // helpful to a user.
            addNote(entity.getProjectOrDomain(), getAssistantUser(), entity,
                "removing irrelevant annotations failed and was not recoverable:
" + e);
        } catch (Exception e2) {
            log.error("failed to add note indicating failure to "
                + "remove irrelevant annotations.", e2);
        }
    }
    for (String propertyName : propertyNameToNLPText.keySet()) {
        analyzeProperty(propertyName);
    }
}

```

```

protected NLPText getPropertyNlpText(String propertyName) {
    return propertyNameToNLPText.get(propertyName);
}

protected void setPropertyText(String propertyName, String text) {
    NLPText nlpText =
    getLexicalAssistant().getNLPProcessorFactory().processText(text);
    propertyNameToNLPText.put(propertyName, nlpText);
}

protected void analyzeProperty(String propertyName) {
    ProjectOrDomainEntity entity = getEntity();
    ProjectOrDomain projectOrDomain = entity.getProjectOrDomain();
    NLPText nlpText = getPropertyNlpText(propertyName);
    log.debug("lexically analyzing the " + propertyName + " of " +
entity);

    // first remove any lexical issues related to the name that are no
    // longer relevant
    try {
        getLexicalAssistant().removeUnneedLexicalIssues(getAssistantUser(),
projectOrDomain,
            entity, propertyName, nlpText);
    } catch (Exception e) {
        log.error("failed to remove irrelevant annotations for the " +
propertyName + " of "
            + entity, e);
    try {
        // TODO: adding a note with the exception is probably not
        // helpful to a user.
        addNote(projectOrDomain, getAssistantUser(), entity,
            "removing irrelevant annotations for the " + propertyName + " of "
" + entity
            + " failed and was not recoverable: " + e);
    } catch (Exception e2) {
        log.error("failed to add note indicating failure to "
            + " remove irrelevant annotations.", e2);
    }
}

try {
    getLexicalAssistant().checkSpelling(getAssistantUser(),
projectOrDomain, entity,
    propertyName, nlpText);
} catch (Exception e) {
    log.error("failed to spell check the " + propertyName + " of " +
entity, e);
}

```

```

try {
    // TODO: adding a note with the exception is probably not
    // helpful to a user.
    addNote(projectOrDomain, getAssistantUser(), entity, "spell
checking of the "
        + propertyName + " failed and was not recoverable: " + e);
} catch (Exception e2) {
    log.error("failed to add note indicating failure of spell checking
for the "
        + propertyName + " of " + entity, e2);
}

try {
    getLexicalAssistant().checkVagueWordUse(getAssistantUser(),
projectOrDomain, entity,
    propertyName, nlpText);
} catch (Exception e) {
    log.error("failed to check vague words in the " + propertyName + "
of " + entity, e);
try {
    // TODO: adding a note with the exception is probably not
    // helpful to a user.
    addNote(projectOrDomain, getAssistantUser(), entity, "vague words
checking of the "
        + propertyName + " failed and was not recoverable: " + e);
} catch (Exception e2) {
    log.error("failed to add note indicating failure of vague words
checking for the "
        + propertyName + " of " + entity, e2);
}

try {
    getLexicalAssistant().findPossibleGlossaryTerms(getAssistantUser(),
projectOrDomain,
    entity, nlpText);
} catch (Exception e) {
    log.error("failed to find glossary terms for the " + propertyName + "
of " + entity, e);
try {
    // TODO: adding a note with the exception is probably not
    // helpful to a user.
    addNote(projectOrDomain, getAssistantUser(), entity,
        "glossary term finding failed for the " + propertyName
            + " and was not recoverable: " + e);
} catch (Exception e2) {
}

```

```

        log.error("failed to add note indicating failure of glossary term"
            + " identification for the " + propertyName + " of " + entity,
        e2);
    }
}

try {
    getLexicalAssistant().findPotentialComplexSentences(getAssistantUser(),
        projectOrDomain, entity, propertyName, nlpText);
} catch (Exception e) {
    log.error("failed to check complexity of the " + propertyName + " of " + entity, e);
    try {
        // TODO: adding a note with the exception is probably not
        // helpful to a user.
        addNote(projectOrDomain, getAssistantUser(), entity, "complexity
        checking of the "
            + propertyName + " failed and was not recoverable: " + e);
    } catch (Exception e2) {
        log.error("failed to add note indicating failure of complexity
        checking for the "
            + propertyName + " of " + entity, e2);
    }
}

public LexicalAssistant getLexicalAssistant() {
    return lexicalAssistant;
}

public User getAssistantUser() {
    return
getLexicalAssistant().getAnnotationRepository().get(assistantUser);
}
}

```

projectoverviewpanel.java

```

/*
 * $Id: ProjectOverviewPanel.java,v 1.18 2009/03/22 11:08:24 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

```

```

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.MessageFormat;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.CheckBox;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.filetransfer.DownloadProvider;
import nextapp.echo2.app.filetransfer.UploadEvent;
import nextapp.echo2.app.filetransfer.UploadListener;
import nextapp.echo2.app.filetransfer.UploadSelect;

import org.apache.commons.io.IOUtils;
import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.ComboBox;
import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import
edu.harvard.fas.rregan.requel.project.command.EditProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ExportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ImportProjectCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.uiframework.navigation.DownloadButton;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedTextListModel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.ToggleButtonModelEx;

/**
 * @author ron
 */
public class ProjectOverviewPanel extends
AbstractRequelProjectEditorPanel {
private static final Logger log =
Logger.getLogger(ProjectOverviewPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the name field. If the property is undefined "Name" is
used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the description field. If the property is undefined
 * "Description" is used.
 */
public static final String PROP_LABEL_DESCRIPTION =
"Description.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the organization field. If the property is undefined
 * "Organization" is used.
 */
public static final String PROP_LABEL_ORGANIZATION =
"Organization.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the created by user field. If the property is undefined
"Created
 * By" is used.
 */
public static final String PROP_LABEL_CREATED_BY = "CreatedBy.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the export button. If the property is undefined "Export"
is
 * used.
 */
public static final String PROP_LABEL_EXPORT_BUTTON =
"ExportButton.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the import button. If the property is undefined "Import"
is
 * used.
 */
public static final String PROP_LABEL_IMPORT_BUTTON =
"ImportButton.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the upload button for the project import function. If the
 * property is undefined "Upload" is used.
 */
public static final String PROP_LABEL_UPLOAD_BUTTON =
"UploadButton.Label";

/**
 * The name to use in the ProjectOverviewPanel.properties file to set
the
 * label of the upload button for the project import function. If the
 * property is undefined "Upload" is used.
 */
public static final String PROP_LABEL_ENABLE_ANALYSIS_BUTTON =
"EnableAnalysisButton.Label";

private final UserRepository userRepository;

```

```

private UpdateListener updateListener;
private File tmpUpload;
private Button importButton;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 * @param userRepository
 */
public ProjectOverviewPanel(CommandHandler commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository,
    UserRepository userRepository) {
    this(ProjectOverviewPanel.class.getName(), commandHandler,
projectCommandFactory,
    projectRepository, userRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 * @param userRepository
 */
public ProjectOverviewPanel(String resourceBundleName, CommandHandler
commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository,
    UserRepository userRepository) {
    super(resourceBundleName, Project.class,
        ProjectManagementPanelNames.PROJECT_OVERVIEW_PANEL_NAME,
    commandHandler,
        projectCommandFactory, projectRepository);
    this.userRepository = userRepository;
}

@Override
public boolean isReadOnlyMode() {
    if (getProject() == null) {
        // if this is a new project
        ProjectUserRole projectUserRole = ((User) getApp().getUser())
            .getRoleForType(ProjectUserRole.class);
        return !projectUserRole.canCreateProjects();
    } else {
        return super.isReadOnlyMode();
    }
}

}
}

/**
 * If the editor is editing an existing project the title specified
in the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Project Overview: {0}"<br>
 * Valid variables are:<br>
 * {0} - project name<br>
 * {1} - organization name<br>
 * For new projects it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Project"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    if (getProject() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Project Overview: {0}"));
        return MessageFormat.format(msgPattern, getProject().getName(),
getProject()
            .getOrganization().getName());
    } else {
        String msg = getResourceBundleHelper(getLocale())
            .getString(
                PROP_NEW_OBJECT_PANEL_TITLE,
                getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE
                    ,
                    "New Project"));
        return msg;
    }
}

@Override

```

```

public void setup() {
    super.setup();
    Project project = getProject();
    if (project != null) {
        addInput(EditProjectCommand.FIELD_NAME, PROP_LABEL_NAME, "Name",
new TextField(),
            new StringDocumentEx(project.getName()));
        addInput("description", PROP_LABEL_DESCRIPTION, "Description", new
TextArea(),
            new StringDocumentEx(project.getText()));
        addInput("organizationName", PROP_LABEL_ORGANIZATION,
"Organization", new ComboBox(),
            new
CombinedTextListModel(getUserRepository().getOrganizationNames(),
project
                .getOrganization().getName()));

        User createdBy = project.getCreatedBy();
        StringBuilder sb = new StringBuilder(50);
        sb.append(createdBy.getUsername());
        if ((createdBy.getName() != null) && (createdBy.getName().length() > 0)) {
            sb.append(" [");
            sb.append(createdBy.getName());
            sb.append("]");
        }
        addInput("createdBy", PROP_LABEL_CREATED_BY, "CreatedBy", new
Label(), sb.toString());
        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
project);

        addActionButton(new
DownloadButton(getResourceBundleHelpergetLocale()).getString(
    PROP_LABEL_EXPORT_BUTTON, "Export"), new
ProjectExportDownloadProvider(this));
    } else {
        addInput(EditProjectCommand.FIELD_NAME, PROP_LABEL_NAME, "Name",
new TextField(),
            new StringDocumentEx());
        addInput("description", PROP_LABEL_DESCRIPTION, "Description", new
TextArea(),
            new StringDocumentEx());
        addInput("organizationName", PROP_LABEL_ORGANIZATION,
"Organization", new ComboBox(),

```

```

            new
CombinedTextListModel(getUserRepository().getOrganizationNames(),
""));

        try {
            UploadSelect importXml = addInput("importXml",
PROP_LABEL_IMPORT_BUTTON, "Import",
new UploadSelect(), null);
            importXml.addUploadListerner(new
ProjectImportUploadListerner(this));
            importXml.setEnabledSendButtonText(getResourceBundleHelper(getLocale())
.getString(
    PROP_LABEL_UPLOAD_BUTTON, "Upload"));
        } catch (Exception e) {
            log.error(e, e);
        }
        addInput("enableAnalysis", PROP_LABEL_ENABLE_ANALYSIS_BUTTON,
"Enable Analysis",
new CheckBox(), new ToggleButtonModelEx(true));

        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
null);

        importButton = addActionButton(new
Button(getResourceBundleHelpergetLocale())
.getString(PROP_LABEL_IMPORT_BUTTON, "Import"));
        importButton.addActionListerner(new ImportListerner(this));
        importButton.setEnabled(false);
    }

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListerner(UpdateEntityEven
t.class,
            updateListener);
    }
    updateListener = new UpdateListerner(this);
    getEventDispatcher().addEventTypeActionListerner(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListerner(UpdateEntityEven
t.class,

```

```

        updateListener);
    }

@Override
public void save() {
    try {
        super.save();
        EditProjectCommand command =
getProjectCommandFactory().newEditProjectCommand();
        command.setProject(getProject());
        command.setEditedBygetCurrentUser());
        command.setName(getInputValue(EditProjectCommand.FIELD_NAME,
String.class));
        command.setText(getInputValue("description", String.class));
        command.setOrganizationName(getInputValue("organizationName",
String.class));
        command = getCommandHandler().execute(command);
        setValid(true);

        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
                updateListener);
        }

        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getProject()));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        } else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
            InvalidStateException ise = (InvalidStateException) e.getCause();
            for (InvalidValue invalidValue : ise.getInvalidValues()) {
                String propertyName = invalidValue.getPropertyName();
                setValidationMessage(propertyName, invalidValue.getMessage());
            }
        } else {
            setGeneralMessage(e.toString());
        }
    } catch (Exception e) {
        log.error("could not save the project: " + e, e);
        setGeneralMessage("Could not save: " + e);
    }
}

private void exportProject(OutputStream outputStream) {
    try {
        ExportProjectCommand command =
getProjectCommandFactory().newExportProjectCommand();
        command.setProject(getProject());
        command.setOutputStream(outputStream);
        command = getCommandHandler().execute(command);
    } catch (Exception e) {
        log.error("could not export the project: " + e, e);
        setGeneralMessage("Could not export: " + e);
    }
}

private void importProject(InputStream inputStream) {
    try {
        ImportProjectCommand command =
getProjectCommandFactory().newImportProjectCommand();
        command.setProject(getProject());
        command.setName(getInputValue(EditProjectCommand.FIELD_NAME,
String.class));
        command.setEditedBygetCurrentUser());
        command.setInputStream(inputStream);
        command.setAnalysisEnabled(getInputValue("enableAnalysis",
Boolean.class));
        command = getCommandHandler().execute(command);
        setTargetObject(command.getProject());
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getProject()));
    } catch (Exception e) {
        log.error("could not import the project: " + e, e);
        setGeneralMessage("Could not import: " + e);
    }
}

private Project getProject() {
    return (Project) getTargetObject();
}

private UserRepository getUserRepository() {
    return userRepository;
}

private class ProjectExportDownloadProvider implements
DownloadProvider {
}

```

```

private final ProjectOverviewPanel panel;

private ProjectExportDownloadProvider(ProjectOverviewPanel panel) {
    this.panel = panel;
}

public String getContentType() {
    return "text/xml";
}

public String getFileName() {
    return panel.getProject().getName() + ".xml";
}

public int getSize() {
    return 0;
}

public void writeFile(OutputStream outputStream) throws IOException
{
    panel.exportProject(outputStream);
}

private class ProjectImportUploadListener implements UploadListener {
    static final long serialVersionUID = 0L;

    private final ProjectOverviewPanel panel;

    private ProjectImportUploadListener(ProjectOverviewPanel panel) {
        this.panel = panel;
    }

    @Override
    public void fileUpload(UploadEvent uploadEvent) {
        FileOutputStream tmpOutStream = null;
        try {
            tmpUpload = File.createTempFile("projectImport", ".xml");
            tmpOutStream = new FileOutputStream(tmpUpload);
            IOUtils.copy(uploadEvent.getInputStream(), tmpOutStream);
            panel.setGeneralMessage("Project file " +
uploadEvent.getFileName()
                + " uploaded and ready for import.");
            panel.importButton.setEnabled(true);
        } catch (Exception e) {
            panel.setGeneralMessage("Could not upload file: " + e);
        } finally {
            if (tmpOutStream != null) {
                IOUtils.closeQuietly(tmpOutStream);
            }
        }
    }
}

@Override
public void invalidFileUpload(UploadEvent uploadEvent) {
    panel.setGeneralMessage("Could not upload file.");
}

private static class ImportListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ProjectOverviewPanel panel;

    private ImportListener(ProjectOverviewPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        if (panel.tmpUpload != null) {
            try {
                panel.importProject(new FileInputStream(panel.tmpUpload));
            } catch (FileNotFoundException e) {
                panel.setGeneralMessage("Could not import file: " + e);
            }
        } else {
            panel.setGeneralMessage("A file must be uploaded before
importing.");
        }
    }
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ProjectOverviewPanel panel;

    private UpdateListener(ProjectOverviewPanel panel) {
        this.panel = panel;
    }

    @Override

```

```

public void actionPerformed(ActionEvent e) {
    Project existingProject = (Project) panel.getTargetObject();
    if ((e instanceof UpdateEntityEvent) && (existingProject != null))
    {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        Project updatedProject = null;
        if (event.getObject() instanceof Project) {
            updatedProject = (Project) event.getObject();
        } else if (event.getObject() instanceof Annotation) {
            Annotation updatedAnnotation = (Annotation) event.getObject();
            if (event instanceof DeletedEntityEvent) {
                updatedProject = existingProject;
                if (updatedProject.getAnnotations().contains(updatedAnnotation))
                {
                    updatedProject.getAnnotations().remove(updatedAnnotation);
                }
            } else {
                if
                (updatedAnnotation.getAnnotatables().contains(existingProject)) {
                    for (Annotatable annotatable :
                    updatedAnnotation.getAnnotatables()) {
                        if (annotatable.equals(existingProject)) {
                            updatedProject = (Project) annotatable;
                            break;
                        }
                    }
                }
            }
        }
        if ((updatedProject != null) &&
        updatedProject.equals(existingProject)) {
            // TODO: check the input fields to see if the user has made
            // a change before resetting the object and updating the
            // input fields.
            panel.setInputValue("annotations", updatedProject);
            panel.setTargetObject(updatedProject);
        }
    }
}
}

```

projectrepository.java

```

/*
 * $Id: ProjectRepository.java,v 1.17 2009/02/16 10:10:09 rregan Exp $

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.Repository;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;
import
edu.harvard.fas.rregan.requel.project.exception.NoSuchProjectException
;
import edu.harvard.fas.rregan.requel.project.impl.AddActorPosition;
import
edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * A repository for holding projects and project entities.
 *
 * @author ron
 */
public interface ProjectRepository extends Repository {

    /**
     * @param name -
     *          the name of the project
     * @return the project with the supplied name.
     * @throws NoSuchProjectException -
     *          if a project with the given name does not exist.
     */
    public Project findProjectByName(String name) throws
NoSuchProjectException;

    /**
     * @param projectOrDomain -
     *          the project or domain that contains the goal.
     * @param name -
     *          the name of the goal.
     * @return the goal with the supplied name for the supplied project.
     * @throws NoSuchEntityException -
     *          if a goal does not exist for the project or domain and
name.
     */
    public Goal findGoalByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)

```

```

throws NoSuchEntityException;

/**
 * @param projectOrDomain -
 *          the project or domain that contains the usecase.
 * @param name -
 *          the name of the usecase.
 * @return the usecase with the supplied name for the supplied
project.
 * @throws NoSuchEntityException -
 *          if a usecase does not exist for the project or domain
and
 *          name.
 */
public UseCase findUseCaseByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
    throws NoSuchEntityException;

/**
 * @param projectOrDomain -
 *          the project or domain that contains the story.
 * @param name -
 *          the name of the story.
 * @return the story with the supplied name for the supplied project.
 * @throws NoSuchEntityException -
 *          if a story does not exist for the project or domain
and name.
 */
public Story findStoryByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
    throws NoSuchEntityException;

/**
 * @param projectOrDomain -
 *          the project or domain that contains the scenario.
 * @param name -
 *          the name of the scenario.
 * @return
 * @throws NoSuchEntityException
 */
public Scenario findScenarioByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain,
    String name) throws NoSuchEntityException;

/**
 * @param usecase
 * @return THe scenarios used by the supplied usecase

```

```

*/
public Set<Scenario> findScenariosUsedByUseCase(UseCase usecase);

/**
 * Find a non-user stakeholder by name.
 *
 * @param projectOrDomain -
 *          the project or domain that contains the stakeholder.
 * @param name -
 *          the name of the stakeholder.
 * @return
 * @throws NoSuchEntityException
 */
public Stakeholder
findStakeholderByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain,
    String name) throws NoSuchEntityException;

/**
 * Find a user stakeholder by user.
 *
 * @param projectOrDomain -
 *          the project or domain that contains the stakeholder.
 * @param user -
 *          the user.
 * @return
 * @throws NoSuchEntityException
 */
public Stakeholder
findStakeholderByProjectOrDomainAndUser(ProjectOrDomain
projectOrDomain,
    User user) throws NoSuchEntityException;

/**
 * @param entityType -
 *          the type of project entity (a subclass of
 *          ProjectOrDomainEntity) the permission is for
 * @param permissionType -
 *          the permission type (Edit, Grant, etc.)
 * @return The StakeholderPermission for the supplied entity type and
 *          permission type.
 * @throws EntityException -
 *          if a permission doesn't exist for the given project
entity
 *          type and permission type.
 */

```

```

public StakeholderPermission findStakeholderPermission(Class<?>
entityType,
    StakeholderPermissionType permissionType) throws EntityException;
/**
 * @return All the permissions available to grant on project
entities.
 */
public Set<StakeholderPermission>
findAvailableStakeholderPermissions();

/**
 * TODO: is this needed because the project or domain already holds
its
 * terms. Get a glossary term for the given project or domain.
 *
 * @param projectOrDomain -
 *          the project or domain to search for the term.
 * @param name -
 *          the term name
 * @return
 */
public GlossaryTerm
findGlossaryTermForProjectOrDomain(ProjectOrDomain projectOrDomain,
    String name);

/**
 * @param projectOrDomainEntity -
 *          a project or domain entity to find all the terms for.
 * @return
 */
public Set<GlossaryTerm> findGlossaryTermsForProjectOrDomainEntity(
    ProjectOrDomainEntity projectOrDomainEntity);

/**
 * @param projectOrDomain -
 *          the project or domain to add the term to.
 * @param term -
 *          the term to add.
 * @return
 */
public AddGlossaryTermPosition
findAddGlossaryTermPosition(ProjectOrDomain projectOrDomain,
    String term);

/**
 * @param projectOrDomain

```

```

 * @param name
 * @return
 * @throws EntityException
 *          if an actor with the name doesn't exist for the
project.
 */
public Actor findActorByProjectOrDomainAndName(ProjectOrDomain
projectOrDomain, String name)
throws EntityException;

/**
 * @param projectOrDomain -
 *          the project or domain to add the actor to.
 * @param actorName -
 *          the actor name.
 * @return
 */
public AddActorPosition findAddActorPosition(ProjectOrDomain
projectOrDomain, String actorName);

/**
 * @param projectOrDomain -
 *          the project or domain that contains the report
generator.
 * @param name -
 *          the name of the report generator.
 * @return
 * @throws EntityException
 */
public ReportGenerator findReportGeneratorByProjectOrDomainAndName(
    ProjectOrDomain projectOrDomain, String name) throws
EntityException;
}


```

projectset.java

```

/*
 * $Id: ProjectSet.java,v 1.1 2008/03/27 09:25:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

/**

```

```

 * @author ron
 */
public interface ProjectSet extends Set<Project> {
}

```

projectteam.java

```

package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

public interface ProjectTeam extends ProjectOrDomainEntity,
Comparable<ProjectTeam> {
    public Set<Stakeholder> getMembers();
}

```

projectteamimpl.java

```

/*
 * $Id: ProjectTeamImpl.java,v 1.16 2009/02/12 11:01:35 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

```

```

import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectTeam;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "teams", uniqueConstraints =
{ @UniqueConstraint(columnNames =
{ "projectordomain_id", "name" }) })
@XmlRootElement(name = "team", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "team", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ProjectTeamImpl extends AbstractProjectOrDomainEntity
implements ProjectTeam {
    static final long serialVersionUID = 0L;

    private Set<Stakeholder> members = new TreeSet<Stakeholder>();

    /**
     * @param projectOrDomain
     * @param createdBy
     * @param name
     */
    public ProjectTeamImpl(ProjectOrDomain projectOrDomain, User
createdBy, String name) {
        super(projectOrDomain, createdBy, name);
        projectOrDomain.getTeams().add(this);
    }

    protected ProjectTeamImpl() {
        // for hibernate
    }

    @Override
    @Column(nullable = false, unique = false)

```

```

@NotEmpty(message = "a unique name is required.")
@XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return super.getName();
}

// hack for JAXB to set the name, for some reason it won't use the
inherited
// method.
@Override
public void setName(String name) {
    super.setName(name);
}

@Transient
@XmlID
@XmlAttribute(name = "id")
public String getXmlId() {
    return "TEM_" + getId();
}

@Transient
public String getDescription() {
    return "Team: " + getName();
}

XmlElementWrapper(name = "members", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
XmlElement(name = "stakeholderRef", type = StakeholderImpl.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = StakeholderImpl.class, cascade =
{ CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "team_stakeholders", joinColumns =
{ @JoinColumn(name = "team_id") }, inverseJoinColumns =
{ @JoinColumn(name = "stakeholder_id") })
@Sort(type = SortType.NATURAL)
public Set<Stakeholder> getMembers() {
    return members;
}

protected void setMembers(Set<Stakeholder> members) {
    this.members = members;
}

```

```

/**
 * This is for JAXB to fixup the parent child relationship with the
 * canonical term.
 *
 * @see UnmarshallerListener
 */
public void afterUnmarshal() {
    for (Stakeholder stakeholder : getMembers()) {
        ((StakeholderImpl) stakeholder).setTeam(this);
    }
}

@Override
public int compareTo(ProjectTeam o) {
    return getName().compareTo(o.getName());
}

/**
 * Adapter for JAXB to convert interface ProjectTeam to class
 * ProjectTeamImpl and back.
 *
 * @author ron
 */
@XmlTransient
public static class Team2TeamImplAdapter extends
XmlAdapter<ProjectTeamImpl, ProjectTeam> {

    /**
     * @see
     javax.xml.bind.annotation.adapters.XmlAdapter#marshal(java.lang.Object)
    */
    @Override
    public ProjectTeamImpl marshal(ProjectTeam team) throws Exception {
        return (ProjectTeamImpl) team;
    }

    /**
     * @see
     javax.xml.bind.annotation.adapters.XmlAdapter#unmarshal(java.lang.Object)
    */
    @Override
    public ProjectTeam unmarshal(ProjectTeamImpl team) throws Exception {
        return team;
    }
}

```

```
}
```

projectuserinitializer.java

```
/*
 * $Id: ProjectUserInitializer.java,v 1.9 2009/03/29 11:59:30 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;

/**
 * @author ron
 */
@Component("projectUserInitializer")
@Scope("prototype")
public class ProjectUserInitializer extends AbstractSystemInitializer {

    private final UserRepository userRepository;
    private final EditUserCommand command;
    private final CommandHandler commandHandler;

    /**
     * @param userRepository
     * @param commandHandler
     * @param command
     */
    @Autowired
    public ProjectUserInitializer(UserRepository userRepository,
        CommandHandler commandHandler,
        EditUserCommand command) {
        super(100);
    }
}
```

```
    this.userRepository = userRepository;
    this.commandHandler = commandHandler;
    this.command = command;
}

@Override
public void initialize() {
    try {
        userRepository.findUserByUsername("project");
    } catch (NoSuchUserException e) {
        try {
            command.setUsername("project");
            command.setPassword("project");
            command.setRepassword("project");
            command.setName("Builtin Project User");
            command.setEmailAddress("rreganjr@acm.org");
            command.setOrganizationName("Requel");
            command.addUserRoleName(ProjectUserRole.getRoleName(ProjectUserRole.class));
            command.addUserRolePermissionName(ProjectUserRole
                .getRoleName(ProjectUserRole.class),
                ProjectUserRole.createProjects
                    .getName());
            commandHandler.execute(command);
        } catch (Exception e2) {
            log.error("failed to initialize the project user: " + e2, e2);
        }
    }
}
}
```

projectusernavigatortreenodefactory.java

```
/*
 * $Id: ProjectUserNavigatorTreeNodeFactory.java,v 1.1 2008/09/12
 * 22:44:19 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.util.Enumeration;

import nextapp.echo2.app.Label;
import nextapp.echo2.app.event.ActionEvent;

import org.springframework.context.annotation.Scope;
```

```

import org.springframework.stereotype.Component;

import echopointng.tree.MutableTreeNode;
import echopointng.tree.TreePath;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeUpdateListener;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
@Component("projectUserNavigatorTreeNodeFactory")
@Scope("singleton")
public class ProjectUserNavigatorTreeNodeFactory extends
AbstractNavigatorTreeNodeFactory {

    /**
     * The property name to use to control the label on the project node
     * generated by the factory.
     */
    public final static String PROP_PROJECTS_NODE_LABEL =
"ProjectsNodeLabel";

```

```

    /**
     * @param eventDispatcher
     */
    public ProjectUserNavigatorTreeNodeFactory() {
        super(ProjectUserNavigatorTreeNodeFactory.class.getName(),
User.class);
    }

    /**
     * @see
     edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.tre
e.NavigatorTree,
                     java.lang.Object)
    */
    public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
                                         Object object) {
        User user = (User) object;
        ProjectUserRole projectUserRole =
user.getRoleForType(ProjectUserRole.class);

        String projectsNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
PROP_PROJECTS_NODE_LABEL, "Projects");

        NavigationEvent openProjectListEvent = new OpenPanelEvent(tree,
PanelActionType.Navigator,
                                         user, ProjectUserRole.class, "projectList",
WorkflowDisposition.NewFlow);

        NavigatorTreeNode projectsTreeNode = new
NavigatorTreeNode(eventDispatcher,
                         projectUserRole, new Label(projectsNodeLabel),
openProjectListEvent);

        projectsTreeNode.setUpdateListener(new
NavigatorTreeNodeProjectUpdateListener(
projectsTreeNode, tree));

        for (Project project : projectUserRole.getActiveProjects()) {
            NavigatorTreeNodeFactory factory =
tree.getNavigatorTreeNodeFactory(project);
            projectsTreeNode.add(factory.createTreeNode(eventDispatcher, tree,
project));
        }
    }

```

```

        return projectsTreeNode;
    }

    // TODO: this can be adapted for both Projects and Domains
    private static class NavigatorTreeNodeProjectUpdateListener extends
        AbstractNavigatorTreeNodeUpdateListener {
        static final long serialVersionUID = 0L;

        protected NavigatorTreeNodeProjectUpdateListener(NavigatorTreeNode
            navigatorTreeNode,
            NavigatorTree tree) {
            super(navigatorTreeNode, tree);
        }

        public void actionPerformed(ActionEvent event) {
            if (event instanceof UpdateEntityEvent) {
                UpdateEntityEvent updateEvent = (UpdateEntityEvent) event;
                if ((updateEvent.getObject() != null)) {
                    Project updatedProject = null;
                    if (updateEvent.getObject() instanceof Project) {
                        updatedProject = (Project) updateEvent.getObject();
                    } else if (updateEvent.getObject() instanceof
                        ProjectOrDomainEntity) {
                        ProjectOrDomainEntity updatedStakeholder =
                            (ProjectOrDomainEntity) updateEvent
                                .getObject();
                        updatedProject = (Project)
                            updatedStakeholder.getProjectOrDomain();
                    }
                    if (updatedProject != null) {
                        NavigatorTreeNode projectsNode = getNavigatorTreeNode();
                        // if there are no projects this is needed so the tree
                        // will be redrawn
                        if (projectsNode.isLeaf()) {
                            getTree().getModel().nodeChanged(projectsNode);
                        }
                        MutableTreeNode projectNode = findProjectNode(projectsNode,
                            updatedProject);
                        if (projectNode != null) {
                            getTree().getModel().removeNodeFromParent(projectNode);
                            if (projectNode instanceof NavigatorTreeNode) {
                                ((NavigatorTreeNode) projectNode).dispose();
                            }
                        }
                        addNode(projectsNode, updatedProject);
                        getTree().expandPath(new TreePath(projectsNode.getPath()));
                    }
                }
            }
        }
    }
}

```

```

        }
    }

    private MutableTreeNode findProjectNode(MutableTreeNode
        projectsNode, Project project) {
        Enumeration<MutableTreeNode> enm = projectsNode.children();
        while (enm.hasMoreElements()) {
            MutableTreeNode node = enm.nextElement();
            if (node instanceof NavigatorTreeNode) {
                Object targetObject = ((NavigatorTreeNode)
                    node).getTargetObject();
                if ((targetObject != null) && (targetObject instanceof Project))
                {
                    Project thisProject = (Project) targetObject;
                    if (project.equals(thisProject)) {
                        return node;
                    }
                }
            }
        }
        return null;
    }

    private void addNode(MutableTreeNode projectsNode, Project project)
    {
        // add a new node for this user
        int indexToInsert = 0;
        Enumeration<MutableTreeNode> enm = projectsNode.children();
        while (enm.hasMoreElements()) {
            MutableTreeNode node = enm.nextElement();
            if (node instanceof NavigatorTreeNode) {
                Object targetObject = ((NavigatorTreeNode)
                    node).getTargetObject();
                if ((targetObject != null) && (targetObject instanceof Project))
                {
                    Project thisProject = (Project) targetObject;
                    if (project.compareTo(thisProject) < 1) {
                        break;
                    }
                    indexToInsert++;
                }
            }
        }
        // see http://echo.nextapp.com/site/node/1625
        getTree().getModel().insertNodeInto(
            getTree().getNavigatorTreeNodeFactory(project).createTreeNode(

```

```

        getNavigatorTreeNode().getEventDispatcher(), getTree(),
project),
    projectsNode, indexToInsert);

}
}

}

```

projectuserrole.java

```

/*
 * $Id: ProjectUserRole.java,v 1.21 2009/03/29 02:08:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project;

import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.impl.ProjectImpl;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRolePermission;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;

```

```

import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * A ProjectUserRole represents a user authorized to work with
Projects in the
 * system and maintains a user's specific project data.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.ProjectUserRole")
@XmlRootElement(name = "projectUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "projectUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ProjectUserRole extends AbstractUserRole {
    static final long serialVersionUID = 0L;

    public static final UserRolePermission createProjects = new
UserRolePermission(
    ProjectUserRole.class, "createProjects");
    public static final UserRolePermission inviteUsers = new
UserRolePermission(
    ProjectUserRole.class, "inviteUsers");

    static {
        AbstractUserRole.userRoleTypes.add(ProjectUserRole.class);
        AbstractUserRole.userRoleTypePermissions.put(ProjectUserRole.class,
            new HashSet<UserRolePermission>());
        AbstractUserRole.userRoleTypePermissions.get(ProjectUserRole.class).
add(createProjects);
        //
AbstractUserRole.userRoleTypePermissions.get(ProjectUserRole.class).ad
d(inviteUsers);
    }

    private User user;
    private Set<Project> activeProjects = new TreeSet<Project>();

    /**
     * @param user -
     *          the user this role is assigned to
     */
    public ProjectUserRole(User user) {
        super();
        setUser(user);
    }
}
```

```

}

protected ProjectUserRole() {
    // for hibernate
}

/**
 * @return the user this role is assigned to
 */
@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.PERSIST, CascadeType.REFRESH })
protected User getUser() {
    return user;
}

/***
 * set the user this role is assigned to
 *
 * @param user -
 *          the user this role is assigned to
 */
protected void setUser(User user) {
    this.user = user;
}

/***
 * @return
 */
@ManyToMany(targetEntity = ProjectImpl.class, cascade =
{ CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.EAGER)
@Sort(type = SortType.NATURAL)
public Set<Project> getActiveProjects() {
    return activeProjects;
}

protected void setActiveProjects(Set<Project> activeProjects) {
    this.activeProjects = activeProjects;
}

/***
 * @return true if this user is authorized to create new projects.
 */
public boolean canCreateProjects() {
    return hasUserRolePermission(createProjects);
}

    /**
     * @return true if this user is authorized to invite non-users to
     * become
     *         users of the system.
     */
    public boolean canInviteUsers() {
        return hasUserRolePermission(inviteUsers);
    }

    @Override
    public String toString() {
        return super.toString() + ":" + getUser().getUsername();
    }

    private Integer tmpHashCode = null;

    @Override
    public int hashCode() {
        if (tmpHashCode == null) {
            if (getId() != null) {
                tmpHashCode = new Integer(getId().hashCode());
            } else {
                final int prime = 31;
                int result = 1;
                result = prime * result + ((getRoleName() == null) ? 0 :
getRoleName().hashCode());
                result = prime * result + ((getUser() == null) ? 0 :
getUser().hashCode());
                tmpHashCode = new Integer(result);
            }
        }
        return tmpHashCode.intValue();
    }

    @Override
    public boolean equals(Object obj) {
        if (!super.equals(obj)) {
            return false;
        }
        final ProjectUserRole other = (ProjectUserRole) obj;
        if (getUser() == null) {
            if (other.getUser() != null) {
                return false;
            }
        } else if (!getUser().equals(other.getUser())) {
            return false;
        }
    }
}

```

```

    return true;
}

/**
 * This is for JAXB to patchup the parent/child relationship.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */
public void afterUnmarshal(final UserRepository userRepository,
Object parent) {
    setUser((User) parent);
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            if (getUser() != null) {
                try {
                    User existingUser = userRepository.findUserByUsername(getUser()
                        .getUsername());
                    setUser(existingUser);
                } catch (NoSuchUserException e) {
                }
            } else {
                throw new SAXException2("ProjectUserRole missing User");
            }
        }
    });
}
}

```

propertycontainer.java

```

/*
 * $Id: PropertyContainer.java,v 1.2 2008/02/15 21:59:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils;

import java.util.Map;

public interface PropertyContainer {

```

```

    public Map<String, Object> getProperties();
    public void setProperties(Map<String, Object> properties);
    public void addProperties(Map<String, Object> properties);
    public Object getProperty(String key);
    public Object getProperty(String key, Object defaultValue);
    public void setProperty(String key, Object value);
}

```

propertycontainersupport.java

```

/*
 * $Id: PropertyContainerSupport.java,v 1.2 2008/02/15 21:59:08 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils;

import java.util.HashMap;
import java.util.Map;

public class PropertyContainerSupport implements PropertyContainer {
    private Map<String, Object> properties = new
HashMap<String, Object>();

    public void setProperty(String key, Object value) {
        properties.put(key, value);
    }

    public Object getProperty(String key) {
        return properties.get(key);
    }

    public Object getProperty(String key, Object defaultValue) {
        Object value = properties.get(key);
        if (value != null) {
            return value;
        }
        return defaultValue;
    }
}

```

```

public Map<String, Object> getProperties() {
    Map<String, Object> copy = new
HashMap<String, Object>(properties.size());
    copy.putAll(properties);
    return copy;
}

public void setProperties(Map<String, Object> properties) {
    if (properties == null) {
        this.properties.clear();
    } else {
        this.properties = new
HashMap<String, Object>(properties.size());
        addProperties(properties);
    }
}

public void addProperties(Map<String, Object> properties) {
    this.properties.putAll(properties);
}
}

```

reflectioncommandfactorystrategy.java

```

/*
 * $Id: ReflectionCommandFactoryStrategy.java,v 1.1 2008/12/13
00:41:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.command;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import java.util.List;

/**
 * @author ron
 */
public class ReflectionCommandFactoryStrategy implements
CommandFactoryStrategy {

/**

```

```

 * TODO: add parameters to pass into the command through the
constructor
 */
public ReflectionCommandFactoryStrategy() {
}

/**
 * @see
edu.harvard.fas.rregan.command.CommandFactoryStrategy#newInstance(java
.lang.Class)
 */
public Command newInstance(Class<? extends Command> commandType) {
try {
    return newInstance(commandType, null);
} catch (Exception e) {
    throw new RuntimeException(e);
}
}

/**
 * @param args
 * @return
 * @throws NoSuchMethodException
 * @throws InvocationTargetException
 * @throws InstantiationException
 * @throws IllegalAccessException
 */
protected Command newInstance(Class<? extends Command> commandType,
List<Object> args)
    throws NoSuchMethodException, InvocationTargetException,
InstantiationException,
IllegalAccessException {
Class<?> parameterTypes[] = new Class<?>[args.size()];
for (int i = 0; i < args.size(); i++) {
    parameterTypes[i] = args.get(i).getClass();
}
Constructor<? extends Command> constructor =
commandType.getConstructor(parameterTypes);
return constructor.newInstance(args.toArray());
}
}

```

reflectivetetable.java

```

/*

```

```

* $Id: ReflectiveTable.java,v 1.1 2008/02/15 21:40:33 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.uiframework.reflect.table;

import java.util.Collection;

import org.apache.log4j.Logger;

import nextapp.echo2.app.Border;
import nextapp.echo2.app.Color;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Table;
import nextapp.echo2.app.table.TableColumnModel;

/**
 * A table that reflectively creates columns from a supplied class as
 * a JavaBean or via annotations.
 *
 * see @ReflectiveTableModel and @ReflectiveTableColumnModel for
details
 * of how the columns are created.
 *
 * @author ron
 */
public class ReflectiveTable extends Table {
    private static final Logger log =
Logger.getLogger(ReflectiveTable.class);
    private static final long serialVersionUID = 0;

    public ReflectiveTable(Class<?> clazz, int detailLevel, String
confineToPackagesStartingWith) {
        this(clazz, detailLevel, confineToPackagesStartingWith, null);
    }

    public ReflectiveTable(Class<?> clazz, int detailLevel, String
confineToPackagesStartingWith, Collection<?> options) {
        super(new ReflectiveTableModel(clazz,
confineToPackagesStartingWith, detailLevel, options),
            new ReflectiveTableColumnModel(clazz, detailLevel,
confineToPackagesStartingWith));
        setStyleName("Table");
        setDefaultRenderer(Object.class, new
ReflectiveTableCellRenderer());
        setBorder(new Border(new Extent(1, Extent.PX), Color.BLACK,
Border.STYLE_SOLID));
        TableColumnModel columnModel = getColumnModel();
    }
}

```

```

        for (int i = 0; i < columnModel.getColumnCount() - 1; i++) {
            columnModel.getColumn(i).setWidth(new
Extent(100/columnModel.getColumnCount(), Extent.PERCENT));
        }
        int totalWidth = 0;
        for (int i = 0; i < columnModel.getColumnCount() - 1; i++) {
            totalWidth += columnModel.getColumn(i).getWidth().getValue();
        }
        columnModel.getColumn(columnModel.getColumnCount() -
1).setWidth(new Extent(100 - totalWidth, Extent.PERCENT));
    }

    public void setRowData(Collection<?> options) {
        ((ReflectiveTableModel)getModel()).setRowData(options);
    }
}

```

reflectivetablecellrenderer.java

```

/*
 * $Id: ReflectiveTableCellRenderer.java,v 1.1 2008/02/15 21:40:31
rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.uiframework.reflect.table;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.reflect.ReflectUtils;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Table;
import nextapp.echo2.app.table.TableCellRenderer;

/**
 *
 * @author rreganjr@acm.org
 */
public class ReflectiveTableCellRenderer implements TableCellRenderer {
    private static final Logger log =
Logger.getLogger(ReflectiveTableCellRenderer.class);
    private static final long serialVersionUID = 0;
}

```

```

public Component getTableCellRendererComponent(Table table, Object
value, int column, int row) {
String labelString = "(null)";
if (value != null) {
labelString = ReflectUtils.getLabelForObject(value);
}
Label label = new Label(labelString);

if (row % 2 == 0) {
label.setStyleName("Table.EvenRowLabel");
} else {
label.setStyleName("Table.OddRowLabel");
}
return label;
}
}

```

reflectivetablecolumnmodel.java

```

/*
 * $Id: ReflectiveTableColumnModel.java,v 1.1 2008/02/15 21:40:35
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect.table;

import java.lang.reflect.Method;
import java.util.List;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.reflect.ReflectUtils;

import nextapp.echo2.app.table.DefaultTableModel;
import nextapp.echo2.app.table.TableColumn;

public class ReflectiveTableColumnModel extends
DefaultTableModel {
    private static final Logger log =
Logger.getLogger(ReflectiveTableColumnModel.class);
    private static final long serialVersionUID = 0;
}

```

```

public ReflectiveTableColumnModel(Object object, int selectorLevel,
String confineToPackagesStartingWith) {
this(object.getClass(), selectorLevel,
confineToPackagesStartingWith);
}

public ReflectiveTableColumnModel(Class<?> clazz, int selectorLevel,
String confineToPackagesStartingWith) {
List<Method> valueAccessors =
ReflectUtils.getScalarPropertyMethods(clazz,
confineToPackagesStartingWith, selectorLevel);
int index = 0;
for (Method valueAccessor : valueAccessors) {
TableColumn column = new TableColumn(index);
column.setHeaderValue(ReflectUtils.getLabelForMethod(valueAccessor)
);
addColumn(column);
index++;
}
}
}

```

reflectivetablemodel.java

```

/*
 * $Id: ReflectiveTableModel.java,v 1.1 2008/02/15 21:40:32 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect.table;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.reflect.ReflectUtils;

import nextapp.echo2.app.table.AbstractTableModel;

public class ReflectiveTableModel extends AbstractTableModel {
    private static final Logger log =
Logger.getLogger(ReflectiveTableModel.class);
}

```

```

private static final long serialVersionUID = 0;

private List<Method> valueAccessors;
private List<?> rowData;

/**
 * Create a ReflectiveTableModel
 * @param clazz
 */
public ReflectiveTableModel(Class<?> clazz, String confineToPackagesStartingWith, int displayLevel) {
    super();
    valueAccessors = ReflectUtils.getScalarPropertyMethods(clazz,
        confineToPackagesStartingWith, displayLevel);
}

public ReflectiveTableModel(Class<?> clazz, String confineToPackagesStartingWith, int displayLevel, Collection<?> objects) {
    this(clazz, confineToPackagesStartingWith, displayLevel);
    setRowData(objects);
}

public void setRowData(Collection<?> objects) {
    if (objects != null) {
        rowData = new ArrayList<Object>(objects);
    } else {
        rowData = new ArrayList<Object>();
    }
}

public List<?> getRowData() {
    return rowData;
}

public int getColumnCount() {
    return valueAccessors.size();
}

public int getRowCount() {
    if (rowData != null) {
        return rowData.size();
    }
    return 0;
}

```

```

    /**
     *
     * @param column the column index (0-based)
     * @param row the row index (0-based)
     */
    public Object getValueAt(int column, int row) {
        if (rowData != null) {
            if (column < 0 || column >= getColumnCount()) throw new
                ArrayIndexOutOfBoundsException("column " + column + " is out of the
                range 0.." + getColumnCount());
            if (row < 0 || row >= getRowCount()) throw new
                ArrayIndexOutOfBoundsException("row " + row + " is out of the range
                0.." + getRowCount());

            try {
                Object rowObject = rowData.get(row);
                Method m = valueAccessors.get(column);
                return m.invoke(rowObject);
            } catch (Exception e) {
                log.warn("Could not get value at col " + column + " row " + row +
                    ": " + e, e);
            }
        }
        return null;
    }
}

```

reflectivetableselector.java

```

/*
 * $Id: ReflectiveTableSelector.java,v 1.1 2008/02/15 21:40:34 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect.table;

import java.util.Collection;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

/**
 * A table for selecting a single object from a set of objects. The

```

```

 * of the table are created by reflecting the supplied class to
extract
 * properties. see @ReflectiveTableModel and
@ReflectiveTableColumnModel
 * for details of how the columns are created.
 *
 * @author ron
 */
public class ReflectiveTableSelector extends ReflectiveTable
implements ActionListener {
    private static final Logger log =
Logger.getLogger(ReflectiveTableSelector.class);
    private static final long serialVersionUID = 0;

    private Object selectedOption;

    public ReflectiveTableSelector(Class<?> clazz, int selectorLevel,
String confineToPackagesStartingWith) {
        this(clazz, selectorLevel, confineToPackagesStartingWith, null);
    }

    public ReflectiveTableSelector(Class<?> clazz, int selectorLevel,
String confineToPackagesStartingWith, Collection<?> options) {
        super(clazz, selectorLevel, confineToPackagesStartingWith,
options);
        addActionListener(this);
    }

    public Object getSelectedOption() {
        return this.selectedOption;
    }

    public void actionPerformed(ActionEvent arg0) {
        int selectedRow = getSelectionModel().getMinSelectedIndex();
        if (selectedRow > -1) {
            selectedOption =
((ReflectiveTableModel)getModel()).getRowData().get(selectedRow);
        }
    }
}

```

reflectivetree.java

```

/*
 * $Id: ReflectiveTree.java,v 1.1 2008/02/15 21:41:51 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

 */
package edu.harvard.fas.rregan.uiframework.reflect.tree;

import org.apache.log4j.Logger;

import echopointng.Tree;
import echopointng.tree.TreeNode;
import echopointng.tree.TreeSelectionEvent;
import echopointng.tree.TreeSelectionListener;
import edu.harvard.fas.rregan.uiframework.reflect.ReflectUtils;
import edu.harvard.fas.rregan.uiframework.reflect.UIMethodDisplayHint;

public class ReflectiveTree extends Tree {
    static final long serialVersionUID = 0L;
    private static final Logger log =
Logger.getLogger(ReflectiveTree.class);

    public ReflectiveTree(Object object) throws Exception {
        this(object, null, UIMethodDisplayHint.SHORT_OR_LONG);
    }

    public ReflectiveTree(Object object, String
confineToPackagesStartingWith, int displayLevel)
        throws Exception {
        this(ReflectUtils.getLabelForObject(object), object,
confineToPackagesStartingWith,
            displayLevel);
    }

    public ReflectiveTree(String nodeLabel, Object object, String
confineToPackagesStartingWith,
        int displayLevel) throws Exception {
        super(new ReflectiveTreeModel(nodeLabel, object,
confineToPackagesStartingWith,
            displayLevel));
        this.addTreeSelectionListener(new TreeSelectionListener() {
            public void valueChanged(TreeSelectionEvent evt) {
                if (evt.getNewLeadSelectionPath() != null) {
                    TreeNode node = (TreeNode) evt.getPath().getLastPathComponent();
                    if (node instanceof ReflectiveTreeNode) {
                        if (!((ReflectiveTreeNode) node).isInitialized()) {
                            try {
                                ((ReflectiveTreeNode) node).initializeChildren();
                            } catch (Exception e) {
                                log.error("could not initialize node " + node + ": " + e, e);
                            }
                        }
                    }
                }
            }
        });
    }
}
```

```
        }
    }
}
});
```

reflectivetreemodel.java

```
/*
 * $Id: ReflectiveTreeModel.java,v 1.1 2008/02/15 21:41:51 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect.tree;

import org.apache.log4j.Logger;

import echopointng.tree.DefaultTreeModel;

public class ReflectiveTreeModel extends DefaultTreeModel {
    private static final Logger log =
Logger.getLogger(ReflectiveTreeModel.class);
    private static final long serialVersionUID = 0;

    public ReflectiveTreeModel(String nodeLabel, Object object,
        String confineToPackagesStartingWith, int displayLevel) throws
Exception {
        super(
            new ReflectiveTreeNode(nodeLabel, object,
confineToPackagesStartingWith,
            displayLevel));
    }
}
```

reflectivetreenode.java

```
/*
 * $Id: ReflectiveTreeNode.java,v 1.1 2008/02/15 21:41:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect.tree;

import java.lang.reflect.Method;
import java.util.Arrays;
```

```
import java.util.Collection;
import java.util.Map;

import org.apache.log4j.Logger;

import echopointng.tree.DefaultMutableTreeNode;
import edu.harvard.fas.rregan.uiframework.reflect.ReflectUtils;
import edu.harvard.fas.rregan.uiframework.reflect.UIMethodDisplayHint;

public class ReflectiveTreeNode extends DefaultMutableTreeNode {
    private static final Logger log =
Logger.getLogger(ReflectiveTreeNode.class);
    private static final long serialVersionUID = 0;

    private String nodeLabel;
    private String confineToPackagesStartingWith;
    private int displayLevel;
    private boolean initialized = false;

    public ReflectiveTreeNode(String nodeLabel, Object target,
        String confineToPackagesStartingWith, int displayLevel) throws
Exception {
        this(nodeLabel, target, confineToPackagesStartingWith, displayLevel,
2);
    }

    public ReflectiveTreeNode(String nodeLabel, Object target,
        String confineToPackagesStartingWith, int displayLevel, int
levelsToInitialize)
        throws Exception {
        super(target, true);
        setNodeLabel(nodeLabel);
        setConfineToPackagesStartingWith(confineToPackagesStartingWith);
        setDisplayLevel(displayLevel);
        initializeChildren(levelsToInitialize);
    }

    protected String getNodeLabel() {
        return nodeLabel;
    }

    protected void setNodeLabel(String nodeLabel) {
        this.nodeLabel = nodeLabel;
    }

    protected String getConfineToPackagesStartingWith() {
        return confineToPackagesStartingWith;
```

```

}

protected void setConfineToPackagesStartingWith(String confineToPackagesStartingWith) {
    this.confineToPackagesStartingWith = confineToPackagesStartingWith;
}

protected int getDisplayLevel() {
    return displayLevel;
}

protected void setDisplayLevel(int displayLevel) {
    this.displayLevel = displayLevel;
}

protected boolean isInitialized() {
    return initialized;
}

protected void setInitialized(boolean initialized) {
    this.initialized = initialized;
}

protected void initializeChildren() throws Exception {
    initializeChildren(1);
}

protected void initializeChildren(int levelsToInitialize) throws Exception {
    String nodeLabel = getNodeLabel();
    Object target = getUserObject();
    String confineToPackagesStartingWith =
getConfineToPackagesStartingWith();
    int displayLevel = getDisplayLevel();

    if (levelsToInitialize > 0) {
        log.debug("initializing node for " + target);
        if (target.getClass().isArray()) {
            createArrayBasedNode(nodeLabel, (Object[]) target,
confineToPackagesStartingWith,
            displayLevel, levelsToInitialize);
        } else if (Collection.class.isAssignableFrom(target.getClass())) {
            createCollectionBasedNode(nodeLabel, (Collection<?>) target,
                confineToPackagesStartingWith, displayLevel,
levelsToInitialize);
        } else if (Map.class.isAssignableFrom(target.getClass())) {

```

```

            createMapBasedNode(nodeLabel, (Map<?, ?>) target,
confineToPackagesStartingWith,
                displayLevel, levelsToInitialize);
        } else {
            for (Method method :
ReflectUtils.getPropertyMethods(target.getClass(),
                confineToPackagesStartingWith, displayLevel)) {
                String propertyName = ReflectUtils.getLabelForMethod(method);
                UIMethodDisplayHint annotation = method
                    .getAnnotation(UIMethodDisplayHint.class);
                if ((annotation != null) && (annotation.targetProperty() != null)
                    && (annotation.targetProperty().length() > 0)) {
                    Object targetProperty = method.invoke(target);
                    Method m = ReflectUtils.getPropertyMethod(targetProperty,
annotation
                        .targetProperty());
                    Class<?> returnType = m.getReturnType();
                    if (!ReflectUtils.isScalar(returnType)) {
                        add(new ReflectiveTreeNode(propertyName,
m.invoke(targetProperty),
                confineToPackagesStartingWith, displayLevel,
                (levelsToInitialize - 1)));
                    }
                } else {
                    Class<?> returnType = method.getReturnType();
                    if (!ReflectUtils.isScalar(returnType)) {
                        add(new ReflectiveTreeNode(propertyName, method.invoke(target),
                            confineToPackagesStartingWith, displayLevel,
                            (levelsToInitialize - 1)));
                    }
                }
            }
            setInitialized(true);
        } else if (log.isDebugEnabled()) {
            log.debug("not initializing node for " + target);
        }
    }
}

private void createMapBasedNode(String nodeLabel, Map<?, ?> target,
    String confineToPackagesStartingWith, int displayLevel, int
levelsToInitialize)
    throws Exception {
    DefaultMutableTreeNode rootNode = new
DefaultMutableTreeNode(nodeLabel, true);
    for (Object key : target.keySet()) {
        Object val = target.get(key);

```

```

String label = ReflectUtils.getLabelForObject(val);
rootNode.add(new ReflectiveTreeNode(label, target.get(key),
    confineToPackagesStartingWith, displayLevel, (levelsToInitialize
- 1)));
}
add(rootNode);
}

private void createCollectionBasedNode(String nodeLabel, Collection<?
> target,
    String confineToPackagesStartingWith, int displayLevel, int
levelsToInitialize)
throws Exception {
DefaultMutableTreeNode rootNode = new
DefaultMutableTreeNode(nodeLabel, true);
for (Object element : target) {
    String label = ReflectUtils.getLabelForObject(element);
    rootNode.add(new ReflectiveTreeNode(label, element,
confineToPackagesStartingWith,
        displayLevel, (levelsToInitialize - 1)));
}
add(rootNode);
}

private void createArrayBasedNode(String nodeLabel, Object[] target,
    String confineToPackagesStartingWith, int displayLevel, int
levelsToInitialize)
throws Exception {
createCollectionBasedNode(nodeLabel, Arrays.asList(target),
confineToPackagesStartingWith,
    displayLevel, levelsToInitialize);
}

@Override
public String toString() {
    return getNodeLabel();
}
}

```

reflectutils.java

```

/*
 * $Id: ReflectUtils.java,v 1.1 2008/02/15 21:41:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect;

```

```

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;

/**
 *
 * @author ron
 */
public class ReflectUtils {
    private static final Logger log =
Logger.getLogger(ReflectUtils.class);

    public static List<Method> getScalarPropertyMethods(Class<?> target,
String confineToPackagesStartingWith, int displayLevel) {
        List<Method> propertyMethods = new ArrayList<Method>();
        for (Method method : getPropertyMethods(target,
confineToPackagesStartingWith, displayLevel)) {
            String name = method.getName();
            if (!isScalar(method.getReturnType())) {
                log.debug("Adding method " + name + " of class " +
target.getName() + " with return type of " + method.getReturnType());
                propertyMethods.add(method);
            }
        }
        return propertyMethods;
    }

    public static boolean isScalar(Class<?> target) {
        if (target.isArray() ||
Collection.class.isAssignableFrom(target) ||
Map.class.isAssignableFrom(target)) {
            return false;
        }
        return true;
    }

    /**
     * Given a target class, a package filter
     (confineToPackagesStartingWith), and display

```

```

    * level, return a list of java.lang.reflect.Method objects that
can be used to access
    * properties of objects of the target class type. Methods are
returned for the
    * supplied class and super classes where the full class name
starts with the supplied
    * confineToPackagesStartingWith string, or if
confineToPackagesStartingWith is null
        * the classes full class name doesn't start with java.
    *
    * Methods returned will obey the following rules:
    * - The method is not static.
    * - The method takes no arguments.
    * - The method returns a value (not declared as void).
    * - The method is annotated with a UIMethodDisplayHint
displayLevel that matches the
        * supplied displayLevel.
    * - The displayLevel is LONG and the method follows the JavaBean
naming convention.
        * - The Class name the method is in starts with the string
confineToPackagesStartingWith
            * or if confineToPackagesStartingWith is null does not start
with "java".
    *
    * NOTE: the Methods of any class that starts with
confineToPackagesStartingWith in
        * the super class chain will be returned even if an intervening
class or the original
            * class name doesn't start with confineToPackagesStartingWith.
    *
    * @param target - the target class
    * @param confineToPackagesStartingWith - String like
"edu.harvard.fas.rregan."
    * @param displayLevel UIMethodDisplayHint.SHORT,
UIMethodDisplayHint.LONG or UIMethodDisplayHint.SHORT_OR_LONG
    * @return
    */
public static List<Method> getPropertyMethods(Class<?> target, String
confineToPackagesStartingWith, int displayLevel) {
    List<Method> propertyMethods = new ArrayList<Method>();
    for (Method method : getPropertyMethods(target,
confineToPackagesStartingWith)) {
        UIMethodDisplayHint annotation =
method.getAnnotation(UIMethodDisplayHint.class);
        String name = method.getName();
        if (annotation != null && (annotation.displayLevel() &
displayLevel) != 0) {

```

```

            log.debug("Adding method " + name + " of class " +
target.getName() + " based on annotation displayLevel = " +
annotation.displayLevel());
            propertyMethods.add(method);
        } else if ((displayLevel & UIMethodDisplayHint.LONG) ==
UIMethodDisplayHint.LONG && (name.startsWith("get") ||
name.startsWith("is"))) {
            log.debug("Adding method " + name + " of class " +
target.getName() + " based on JavaBean naming convention.");
            // if there isn't a UI based annotation check if the name matches
the Bean getter pattern
            propertyMethods.add(method);
        }
    }
    return propertyMethods;
}

/**
 * Given a target class return a list of java.lang.reflect.Method
objects that can be
    * used to access properties of objects of the target class type.
Methods are returned
    * for the supplied class and super classes where the full class
name aren't in a java
    * package and the method names follow the JavaBean naming
convention or are annotated
        * with a displayLevel that is not IGNORE.
    *
    * Methods returned will obey the following rules:
    * - The method is not static.
    * - The method takes no arguments.
    * - The method returns a value (not declared as void).
    * - The method is annotated with a UIMethodDisplayHint
displayLevel that is not IGNORE or
        * the method follows the JavaBean naming convention.
    * - The Class name the method is in does not start with "java".
    *
    * @param target - the target class
    * @return
    */
public static List<Method> getPropertyMethods(Class<?> target) {
    return getPropertyMethods(target, null);
}

/**
 * Given a target class return a list of java.lang.reflect.Method
objects that can be

```

```

    * used to access properties of objects of the target class type.
Methods are returned
    * for the supplied class and super classes where the full class
name aren't in a java
    * package and the method names follow the JavaBean naming
convention or are annotated
    * with a displayLevel that is not IGNORE.
*
    * Methods returned will obey the following rules:
    * - The method is not static.
    * - The method takes no arguments.
    * - The method returns a value (not declared as void).
    * - The method is annotated with a UIMethodDisplayHint
displayLevel that is not IGNORE or
    * the method follows the JavaBean naming convention.
    * - The Class name the method is in starts with the string
confineToPackagesStartingWith
    * or if confineToPackagesStartingWith is null does not start
with "java".
*
    * NOTE: the Methods of any class that starts with
confineToPackagesStartingWith in
    * the super class chain will be returned even if an intervening
class or the original
    * class name doesn't start with confineToPackagesStartingWith.
*
    * @param target - the target class
    * @return
    */
public static List<Method> getPropertyMethods(Class<?> target, String
confineToPackagesStartingWith) {
    List<Method> propertyMethods = new ArrayList<Method>();
    if (target.getSuperclass() != null) {
        propertyMethods.addAll(getPropertyMethods(target.getSuperclass(),
confineToPackagesStartingWith));
    }

    if (
        (confineToPackagesStartingWith == null && !
target.getName().startsWith("java")) ||
        (confineToPackagesStartingWith != null &&
target.getName().startsWith(confineToPackagesStartingWith))
    ) {
        for (Method method : target.getDeclaredMethods()) {
            String name = method.getName();

```

```

                // collect non-static public methods with 0 args that return a
value and
                // match JavaBean getter names or are annotated
                if ( !Modifier.isStatic(method.getModifiers()) &&
                    method.getParameterTypes().length == 0 &&
                    !void.class.isAssignableFrom(method.getReturnType()))
                ) {
                    // Annotations may be used to alter the configuration
                    // method.getDeclaredAnnotations();
                    UIMethodDisplayHint annotation =
method.getAnnotation(UIMethodDisplayHint.class);
                    if (annotation != null && annotation.displayLevel() !=
UIMethodDisplayHint.IGNORE) {
                        removeMatchingSuperclassMethod(propertyMethods, name);
                        log.debug("Adding method " + name + " of class " +
target.getName() + " based on annotation displayLevel not IGNORE");
                        propertyMethods.add(method);
                    } else if (annotation == null && (name.startsWith("get") ||

name.startsWith("is"))) {
                        removeMatchingSuperclassMethod(propertyMethods, name);
                        log.debug("Adding method " + name + " of class " +
target.getName() + " based on JavaBean naming convention.");
                        // if there isn't a UI based annotation check if the name
matches the Bean getter pattern
                        propertyMethods.add(method);
                    }
                } else if (log.isDebugEnabled()) {
                    StringBuilder logMsg = new StringBuilder();
                    logMsg.append("Skipping method ");
                    logMsg.append(method.getName());
                    logMsg.append(" of class ");
                    logMsg.append(target.getName());
                    logMsg.append(" because it is");
                    if (Modifier.isStatic(method.getModifiers())) logMsg.append(" static");
                    if (method.getParameterTypes().length > 0) logMsg.append(" takes
parameters");
                    if (void.class.isAssignableFrom(method.getReturnType()))
logMsg.append(" returns null");
                    log.debug(logMsg.toString());
                }
            }
        } else if (log.isDebugEnabled()) {
            if (confineToPackagesStartingWith == null) {
                log.debug("Skipping class " + target.getName() + " because
confineToPackagesStartingWith is null and it is in a java package");
            } else {

```

```

        log.debug("Skipping class " + target.getName() + " because it
doesn't start with " + confineToPackagesStartingWith);
    }
}
return propertyMethods;
}

/**
 * Return a Method of the supplied object that matches the supplied
propertyName either
 * as a UIMethodDisplayHint annotation propertyName() or following
the JavaBean naming
 * convention and doesn't have a UIMethodDisplayHint annotation
displayLevel() of IGNORE.
 *
 * NOTE: the method may be in any class in the objects class
hierarchy, including classes
 * in the java packages.
 *
 * @param target
 * @param propertyName
 * @return
 * @throws NoSuchMethodException if the target object doesn't have a
property named propertyName
 */
public static Method getPropertyMethod(Object target, String
propertyName) throws NoSuchMethodException {

    String getName = "get" +
(Character.isUpperCase(propertyName.charAt(0)) ? propertyName :
Character.toUpperCase(propertyName.charAt(0)) +
propertyName.substring(1));

    String isName = "is" +
(Character.isUpperCase(propertyName.charAt(0)) ? propertyName :
Character.toUpperCase(propertyName.charAt(0)) +
propertyName.substring(1));

    for (Method method : getPropertyMethods(target.getClass(), ""))
        UIMethodDisplayHint annotation =
method.getAnnotation(UIMethodDisplayHint.class);
        if (annotation != null && annotation.propertyName() != null &&
annotation.propertyName().equals(propertyName)) {
            return method;
        } else {
            if (getName.equals(method.getName()) ||
isName.equals(method.getName())) {

```

```

                return method;
            }
        }
    }
    throw new NoSuchMethodException("The class " +
target.getClass().getName() + " does not have a property \\" + +
propertyName + "\\\"");
}

/**
 * Return a pretty name appropriate for labeling the property of a
getter method. The label
 * may be generated from the method name or via a UIMethodDisplayHint
annotation label(). If
 * the name is generated from the method name "get" or "is" is
stripped off the beginning of
 * the name, the first character is made upper case, and a space is
injected before all capital
 * letters after the first.
 *
 * If the method returns a boolean value a "?" is appended to the
end of the name.
 *
 * @param method
 * @return
 */
public static String getLabelForMethod(Method method) {
    String label;
    UIMethodDisplayHint annotation =
method.getAnnotation(UIMethodDisplayHint.class);
    if (annotation != null && annotation.label() != null &&
annotation.label().length() > 0) {
        label = annotation.label();
    } else {
        String methodName = method.getName();
        StringBuilder sb = new StringBuilder(methodName.length() + 100);
        int index = (methodName.startsWith("get")?3:
(methodName.startsWith("is")?2:0));
        for (; index < methodName.length(); index++) {
            char ch = methodName.charAt(index);
            if (Character.isUpperCase(ch)) {
                sb.append(" ");
            }
            sb.append(ch);
        }
        if ( Boolean.class.isAssignableFrom(method.getReturnType()) ||
```

```

        boolean.class.isAssignableFrom(method.getReturnType()) ) {
    sb.append("?");
}
label = sb.toString().trim();
}
return label;
}

public static String getLabelForObject(Object target) {
    String label = null;
    if (target != null) {
        UITypeDisplayHint annotation =
target.getClass().getAnnotation(UITypeDisplayHint.class);
        if (annotation != null && annotation.targetProperty() != null &&
annotation.targetProperty().length() > 0) {
            Method propertyMethod = null;
            try {
                propertyMethod = ReflectUtils.getPropertyMethod(target,
annotation.targetProperty());
                Object result = propertyMethod.invoke(target);
                label = (result!=null?result.toString():"(null)");
            } catch (InvocationTargetException e) {
                log.warn("Invoking the method '" +
propertyMethod.getGenericReturnType() + " " + propertyMethod.getName()
+ "' of the class " + propertyMethod.getDeclaringClass().getName() +
" caused the exception: " + e.getCause(), e);
            } catch (IllegalAccessException e) {
                log.error("Invoking the method '" +
propertyMethod.getGenericReturnType() + " " + propertyMethod.getName()
+ "' of the class " + propertyMethod.getDeclaringClass().getName() +
" is not accessible.", e);
            } catch (NoSuchMethodException e) {
                log.error("A method for the value of the
UITypeDisplayHint.targetProperty() of " + annotation.targetProperty()
+ " for the object " + target + " doesn't match a method.", e);
            }
        } else {
            label = target.toString();
        }
    }
    return label;
}

private static void removeMatchingSuperclassMethod(List<Method>
propertyMethods, String name) {

    for (int index = 0; index < propertyMethods.size(); index++) {

```

```

        if (name.equals(propertyMethods.get(index).getName())) {
            Method removed = propertyMethods.remove(index);
            log.debug("Removing existing method " + removed.getName() + " of
class " + removed.getDeclaringClass().getName());
            break;
        }
    }
}

```

removeactorfromactorcontainercommand.java

```

/*
 * $Id: RemoveActorFromActorContainerCommand.java,v 1.1 2008/09/06
09:31:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;

/**
 * @author ron
 */
public interface RemoveActorFromActorContainerCommand extends
EditCommand {

    /**
     * Set the Actor to remove.
     *
     * @param actor
     */
    public void setActor(Actor actor);

    /**
     * @return the updated Actor.
     */
    public Actor getActor();

    /**
     * Set the container this Actor is being removed from.
     */

```

```

/*
 * @param actorContainer
 */
public void setActorContainer(ActorContainer actorContainer);

/**
 * @return the updated Actor container.
 */
public ActorContainer getActorContainer();
}

```

removeactorfromactorcontainercommandimpl.java

```

/*
 * $Id: RemoveActorFromActorContainerCommandImpl.java,v 1.5 2009/03/30
11:54:29 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveActorFromActorCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("removeActorFromActorContainerCommand")
@Scope("prototype")

```

```

public class RemoveActorFromActorContainerCommandImpl extends
AbstractEditProjectCommand implements
RemoveActorFromActorContainerCommand {

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@.Autowired
public RemoveActorFromActorContainerCommandImpl(AssistantFacade
assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory, annotationCommandFactory, commandHandler);
}

private Actor actor;
private ActorContainer actorContainer;

@Override
public Actor getActor() {
return actor;
}

@Override
public ActorContainer getActorContainer() {
return actorContainer;
}

@Override
public void setActor(Actor actor) {
this.actor = actor;
}

@Override
public void setActorContainer(ActorContainer actorContainer) {
this.actorContainer = actorContainer;
}

@Override
public void execute() {
Actor removedActor = getProjectRepository().get(getActor());
}

```

```

    ActorContainer removingContainer =
getProjectRepository().get(getActorContainer());
    removedActor.getReferers().remove(removingContainer);
    removingContainer.getActors().remove(removedActor);

    // replaced the supplied objects with the updated objects for
retrieval.
    removedActor = getRepository().merge(removedActor);
    removingContainer = getRepository().merge(removingContainer);
    setActor(removedActor);
    setActorContainer(removingContainer);
}
}

```

removeactorfromactorcontainercontroller.java

```
/*
 * $Id: RemoveActorFromActorContainerController.java,v 1.2 2008/12/13
00:41:06 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveActorFromActorCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * This controller is to be used in a actor container entity editor
where the
 * editor contains a ActorsTable and allows removing actors from the
entity.
 *
 * @author ron
 */
```

```
public class RemoveActorFromActorContainerController extends AbstractRequelCommandController {
    static final long serialVersionUID = 0;

    private final ActorContainer actorContainer;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     * @param actorContainer
     */
    public RemoveActorFromActorContainerController(EventDispatcher eventDispatcher,
                                                   ProjectCommandFactory commandFactory, CommandHandler commandHandler,
                                                   ActorContainer actorContainer) {
        super(eventDispatcher, commandFactory, commandHandler);
        this.actorContainer = actorContainer;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        if (event instanceof RemoveActorFromActorContainerEvent) {
            RemoveActorFromActorContainerEvent removeEvent =
(RemoveActorFromActorContainerEvent) event;
            if (actorContainer.equals(removeEvent.getActorContainer())) {
                try {
                    ProjectCommandFactory factory = getCommandFactory();
                    RemoveActorFromActorContainerCommand command = factory
                        .newRemoveActorFromActorContainerCommand();
                    command.setEditedBy(null); // TODO: need user?
                    command.setActor(removeEvent.getActor());
                    command.setActorContainer(actorContainer);
                    command = getCommandHandler().execute(command);
                    fireEvent(new UpdateEntityEvent(this, null, command.getActor()));
                    fireEvent(new UpdateEntityEvent(this, null, actorContainer));
                } catch (Exception e) {
                    // TODO: what can fail?
                    log.error(e, e);
                }
            }
        }
    }
}
```

removeactorfromactorcontainerevent.java

```
/*
 * $Id: RemoveActorFromActorContainerEvent.java,v 1.1 2008/09/12
22:44:18 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui.project;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event is fired from the ActorsTable when a user clicks the
"remove"
 * button to remove the actor from the actor container.
 *
 * @author ron
 */
public class RemoveActorFromActorContainerEvent extends
NavigationEvent {
    static final long serialVersionUID = 0;

    private final Actor actor;
    private final ActorContainer actorContainer;

    /**
     * @param source
     * @param actor
     * @param actorContainer
     * @param destinationObject
     */
    public RemoveActorFromActorContainerEvent(Object source, Actor actor,
        ActorContainer actorContainer, Object destinationObject) {
        super(source, RemoveActorFromActorContainerEvent.class.getName(),
destinationObject);
        this.actor = actor;
        this.actorContainer = actorContainer;
    }

    /**
     * @return the actor to remove from the container.
     */
}
```

```
public Actor getActor() {
    return actor;
}

/**
 * @return the container to remove the actor from.
 */
public ActorContainer getActorContainer() {
    return actorContainer;
}
```

removeannotationfromannotatablecommand.java

```
/*
 * $Id: RemoveAnnotationFromAnnotatableCommand.java,v 1.2 2009/01/19
09:32:22 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.command.EditCommand;

/**
 * @author ron
 */
public interface RemoveAnnotationFromAnnotatableCommand extends
EditCommand {

    /**
     * Set the Annotation to remove.
     *
     * @param annotation
     */
    public void setAnnotation(Annotation annotation);

    /**
     * @return the updated Annotation.
     */
    public Annotation getAnnotation();

    /**
     * Set the annotatable the Annotation is being removed from.
     */
}
```

```

/*
 * @param annotatable
 */
public void setAnnotatable(Annotatable annotatable);

/**
 * @return the updated Annotatable container.
 */
public Annotatable getAnnotatable();
}

```

removeannotationfromannotatablecommandimpl.java

```

/*
 * $Id: RemoveAnnotationFromAnnotatableCommandImpl.java,v 1.5
2009/02/13 12:07:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Note;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteIssueCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.DeleteNoteCommand;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("removeAnnotationFromAnnotatableCommand")

```

```

@Scope("prototype")
public class RemoveAnnotationFromAnnotatableCommandImpl extends
AbstractEditCommand implements
RemoveAnnotationFromAnnotatableCommand {

private Annotatable annotatable;
private Annotation annotation;
private User editedBy;

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param repository
 */
@.Autowired
public RemoveAnnotationFromAnnotatableCommandImpl(CommandHandler
commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository) {
super(commandHandler, annotationCommandFactory, repository);
}

@Override
public Annotatable getAnnotatable() {
return annotatable;
}

@Override
public void setAnnotatable(Annotatable annotatable) {
this.annotatable = annotatable;
}

@Override
public Annotation getAnnotation() {
return annotation;
}

@Override
public void setAnnotation(Annotation annotation) {
this.annotation = annotation;
}

@Override
protected User getEditedBy() {
return editedBy;
}
}

```

```

@Override
public void setEditedBy(User editedBy) {
    this.editedBy = editedBy;
}

@Override
public void execute() throws Exception {
    Annotation annotation = getRepository().get(getAnnotation());
    Annotatable annotatable = getRepository().get(getAnnotatable());

    annotatable.getAnnotations().remove(annotation);
    annotation.getAnnotatables().remove(annotatable);

    // if an annotation has no annotatables it is deleted
    if (annotation.getAnnotatables().isEmpty()) {
        if (annotation instanceof Issue) {
            DeleteIssueCommand deleteIssueCommand =
                getAnnotationCommandFactory()
                    .newDeleteIssueCommand();
            deleteIssueCommand.setIssue((Issue) annotation);
            deleteIssueCommand.setEditedBy(getEditedBy());
            getCommandHandler().execute(deleteIssueCommand);
        } else if (annotation instanceof Note) {
            DeleteNoteCommand deleteNoteCommand =
                getAnnotationCommandFactory()
                    .newDeleteNoteCommand();
            deleteNoteCommand.setNote((Note) annotation);
            deleteNoteCommand.setEditedBy(getEditedBy());
            getCommandHandler().execute(deleteNoteCommand);
        }
    }
}
}

```

removegoalfromgoalcontainercommand.java

```

/*
 * $Id: RemoveGoalFromGoalContainerCommand.java,v 1.1 2008/05/01
 * 08:10:16 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;
import edu.harvard.fas.rregan.requel.command.EditCommand;

```

```

import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;

/**
 * @author ron
 */
public interface RemoveGoalFromGoalContainerCommand extends
EditCommand {

    /**
     * Set the goal to remove.
     *
     * @param goal
     */
    public void setGoal(Goal goal);

    /**
     * @return the updated goal.
     */
    public Goal getGoal();

    /**
     * Set the container this goal is being removed from.
     *
     * @param goalContainer
     */
    public void setGoalContainer(GoalContainer goalContainer);

    /**
     * @return the updated goal container.
     */
    public GoalContainer getGoalContainer();
}

```

removegoalfromgoalcontainercommandimpl.java

```

/*
 * $Id: RemoveGoalFromGoalContainerCommandImpl.java,v 1.9 2009/03/30
 * 11:54:30 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

```

```

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveGoalFromGoalContai
nerCommand;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("removeGoalFromGoalContainerCommand")
@Scope("prototype")
public class RemoveGoalFromGoalContainerCommandImpl extends
AbstractEditProjectCommand implements
RemoveGoalFromGoalContainerCommand {

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@.Autowired
public RemoveGoalFromGoalContainerCommandImpl(AssistantFacade
assistantManager,
UserRepository userRepository, ProjectRepository projectRepository,
ProjectCommandFactory projectCommandFactory,
AnnotationCommandFactory annotationCommandFactory,
CommandHandler commandHandler) {
super(assistantManager, userRepository, projectRepository,
projectCommandFactory, annotationCommandFactory, commandHandler);
}

private Goal goal;
private GoalContainer goalContainer;

@Override
public Goal getGoal() {

```

```

        return goal;
    }

@Override
public GoalContainer getGoalContainer() {
    return goalContainer;
}

@Override
public void setGoal(Goal goal) {
    this.goal = goal;
}

@Override
public void setGoalContainer(GoalContainer goalContainer) {
    this.goalContainer = goalContainer;
}

@Override
public void execute() {
    Goal removedGoal = getProjectRepository().get(getGoal());
    GoalContainer removingContainer =
getProjectRepository().get(getGoalContainer());
    removedGoal.getReferers().remove(removingContainer);
    removingContainer.getGoals().remove(removedGoal);

    // replaced the supplied objects with the updated objects for
retrieval.
    removedGoal = getRepository().merge(removedGoal);
    removingContainer = getRepository().merge(removingContainer);
    setGoal(removedGoal);
    setGoalContainer(removingContainer);
}
}

```

removegoalfromgoalcontainercontroller.java

```

/*
 * $Id: RemoveGoalFromGoalContainerController.java,v 1.2 2008/12/13
00:41:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;

```

```

import edu.harvard.fas.rregan.requel.project.GoalContainer;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveGoalFromGoalContai
nerCommand;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * This controller is to be used in a goal container entity editor
where the
 * editor contains a GoalsTable and allows removing goals from the
entity.
 *
 * @author ron
 */
public class RemoveGoalFromGoalContainerController extends
AbstractRequelCommandController {
    static final long serialVersionUID = 0;

    private final GoalContainer goalContainer;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     * @param goalContainer
     */
    public RemoveGoalFromGoalContainerController(EventDispatcher
eventDispatcher,
        ProjectCommandFactory commandFactory, CommandHandler
commandHandler,
        GoalContainer goalContainer) {
        super(eventDispatcher, commandFactory, commandHandler);
        this.goalContainer = goalContainer;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        if (event instanceof RemoveGoalFromGoalContainerEvent) {
            RemoveGoalFromGoalContainerEvent removeEvent =
(RemoveGoalFromGoalContainerEvent) event;

```

```

        if (goalContainer.equals(removeEvent.getGoalContainer())) {
            try {
                ProjectCommandFactory factory = getCommandFactory();
                RemoveGoalFromGoalContainerCommand command = factory
                    .newRemoveGoalFromGoalContainerCommand();
                command.setEditedBy(null); // TODO: need user?
                command.setGoal(removeEvent.getGoal());
                command.setGoalContainer(goalContainer);
                command = getCommandHandler().execute(command);
                fireEvent(new UpdateEntityEvent(this, null, command.getGoal()));
                fireEvent(new UpdateEntityEvent(this, null, goalContainer));
            } catch (Exception e) {
                // TODO: what can fail?
                log.error(e, e);
            }
        }
    }
}

```

removegoalfromgoalcontainerevent.java

```

/*
 * $Id: RemoveGoalFromGoalContainerEvent.java,v 1.1 2008/09/12
22:44:21 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui.project;

import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event is fired from the GoalsTable when a user clicks the
"remove"
 * button to remove the goal from the goal container.
 *
 * @author ron
 */
public class RemoveGoalFromGoalContainerEvent extends NavigationEvent
{
    static final long serialVersionUID = 0;

```

```

private final Goal goal;
private final GoalContainer goalContainer;

/**
 * @param source
 * @param goal
 * @param goalContainer
 * @param destinationObject
 */
public RemoveGoalFromGoalContainerEvent(Object source, Goal goal,
GoalContainer goalContainer, Object destinationObject) {
    super(source, RemoveGoalFromGoalContainerEvent.class.getName(),
destinationObject);
    this.goal = goal;
    this.goalContainer = goalContainer;
}

/**
 * @return the goal to remove from the container.
 */
public Goal getGoal() {
    return goal;
}

/**
 * @return the container to remove the goal from.
 */
public GoalContainer getGoalContainer() {
    return goalContainer;
}
}

```

removestoryfromstorycontainercommand.java

```

/*
 * $Id: RemoveStoryFromStoryContainerCommand.java,v 1.1 2008/09/03
02:56:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;

```

```

/**
 * @author ron
 */
public interface RemoveStoryFromStoryContainerCommand extends
EditCommand {

/**
 * Set the Story to remove.
 *
 * @param story
 */
public void setStory(Story story);

/**
 * @return the updated Story.
 */
public Story getStory();

/**
 * Set the container this Story is being removed from.
 *
 * @param storyContainer
 */
public void setStoryContainer(StoryContainer storyContainer);

/**
 * @return the updated Story container.
 */
public StoryContainer getStoryContainer();
}

```

removestoryfromstorycontainercommandimpl.java

```

/*
 * $Id: RemoveStoryFromStoryContainerCommandImpl.java,v 1.5 2009/03/30
11:54:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import edu.harvard.fas.rregan.command.CommandHandler;

```

```

import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveStoryFromStoryCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("removeStoryFromStoryContainerCommand")
@Scope("prototype")
public class RemoveStoryFromStoryContainerCommandImpl extends
AbstractEditProjectCommand implements
RemoveStoryFromStoryContainerCommand {

/**
 * @param assistantManager
 * @param userRepository
 * @param projectRepository
 */
@.Autowired
public RemoveStoryFromStoryContainerCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory,
    CommandHandler commandHandler) {
    super(assistantManager, userRepository, projectRepository,
    projectCommandFactory, annotationCommandFactory, commandHandler);
}

private Story story;
private StoryContainer storyContainer;

@Override
public Story getStory() {
    return story;
}

```

```

@Override
public StoryContainer getStoryContainer() {
    return storyContainer;
}

@Override
public void setStory(Story story) {
    this.story = story;
}

@Override
public void setStoryContainer(StoryContainer storyContainer) {
    this.storyContainer = storyContainer;
}

@Override
public void execute() {
    Story removedStory = getProjectRepository().get(getStory());
    StoryContainer removingContainer =
getProjectRepository().get(getStoryContainer());
    removedStory.getReferers().remove(removingContainer);
    removingContainer.getStories().remove(removedStory);

    // replaced the supplied objects with the updated objects for
retrieval.
    removedStory = getRepository().merge(removedStory);
    removingContainer = getRepository().merge(removingContainer);
    setStory(removedStory);
    setStoryContainer(removingContainer);
}
}

```

removestoryfromstorycontainercontroller.java

```

/*
 * $Id: RemoveStoryFromStoryContainerController.java,v 1.2 2008/12/13
00:41:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.StoryContainer;

```

```

import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveStoryFromStoryCont
ainerCommand;
import
edu.harvard.fas.rregan.requel.ui.AbstractRequelCommandController;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
/**
 * This controller is to be used in a Story container entity editor
where the
 * editor contains a StoriesTable and allows removing Storys from the
entity.
 *
 * @author ron
 */
public class RemoveStoryFromStoryContainerController extends
AbstractRequelCommandController {
    static final long serialVersionUID = 0;

    private final StoryContainer storyContainer;

    /**
     * @param eventDispatcher
     * @param commandFactory
     * @param commandHandler
     * @param storyContainer
     */
    public RemoveStoryFromStoryContainerController(EventDispatcher
eventDispatcher,
                                                ProjectCommandFactory commandFactory, CommandHandler
commandHandler,
                                                StoryContainer storyContainer) {
        super(eventDispatcher, commandFactory, commandHandler);
        this.storyContainer = storyContainer;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        if (event instanceof RemoveStoryFromStoryContainerEvent) {
            RemoveStoryFromStoryContainerEvent removeEvent =
(RemoveStoryFromStoryContainerEvent) event;
            if (storyContainer.equals(removeEvent.getStoryContainer())) {

```

```

        try {
            ProjectCommandFactory factory = getCommandFactory();
            RemoveStoryFromStoryContainerCommand command = factory
                .newRemoveStoryFromStoryContainerCommand();
            command.setEditedBy(null); // TODO: need user?
            command.setStory(removeEvent.getStory());
            command.setStoryContainer(storyContainer);
            command = getCommandHandler().execute(command);
            fireEvent(new UpdateEntityEvent(this, null, command.getStory()));
            fireEvent(new UpdateEntityEvent(this, null, storyContainer));
        } catch (Exception e) {
            // TODO: what can fail?
            log.error(e, e);
        }
    }
}
}

```

removestoryfromstorycontainerevent.java

```

/*
 * $Id: RemoveStoryFromStoryContainerEvent.java,v 1.1 2008/09/12
22:44:20 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui.project;

import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;

/**
 * This event is fired from the GoalsTable when a user clicks the
"remove"
* button to remove the goal from the goal container.
 *
 * @author ron
 */
public class RemoveStoryFromStoryContainerEvent extends
NavigationEvent {
    static final long serialVersionUID = 0;

    private final Story story;

```

```

private final StoryContainer storyContainer;

/**
 * @param source
 * @param goal
 * @param goalContainer
 * @param destinationObject
 */
public RemoveStoryFromStoryContainerEvent(Object source, Story story,
    StoryContainer storyContainer, Object destinationObject) {
    super(source, RemoveStoryFromStoryContainerEvent.class.getName(),
        destinationObject);
    this.story = story;
    this.storyContainer = storyContainer;
}

/**
 * @return the goal to remove from the container.
 */
public Story getStory() {
    return story;
}

/**
 * @return the container to remove the goal from.
 */
public StoryContainer getStoryContainer() {
    return storyContainer;
}

```

removeunneedlexicalissuescommand.java

```

/*
 * $Id: RemoveUnneedLexicalIssuesCommand.java,v 1.1 2009/01/19
 * 09:32:23 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;

```

```

/**
 * @author ron
 */
public interface RemoveUnneedLexicalIssuesCommand extends EditCommand
{

    /**
     * @param projectOrDomain
     */
    public void setProjectOrDomain(ProjectOrDomain projectOrDomain);

    /**
     * @return
     */
    public ProjectOrDomainEntity getThingBeingAnalyzed();

    /**
     * @param thingBeingAnalyzed
     */
    public void setThingBeingAnalyzed(ProjectOrDomainEntity
        thingBeingAnalyzed);

    /**
     * @param annotatableEntityPropertyName
     */
    public void setAnnotatableEntityPropertyName(String
        annotatableEntityPropertyName);

    /**
     * Set the NLPText version of the property being analyzed.
     *
     * @param nlpText
     */
    public void setNlpText(NLPText nlpText);
}

```

removeunneedlexicalissuescommandimpl.java

```

/*
 * $Id: RemoveUnneedLexicalIssuesCommandImpl.java,v 1.2 2009/03/30
 * 11:54:30 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.command;

```

```

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.command.RemoveAnnotationFromA
nnotatableCommand;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.command.RemoveUnneedLexicalIssue
sCommand;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
public class RemoveUnneedLexicalIssuesCommandImpl extends
AbstractEditProjectCommand implements
RemoveUnneedLexicalIssuesCommand {

    private ProjectOrDomain projectOrDomain;
    private ProjectOrDomainEntity thingBeingAnalyzed;
    private String annotatableEntityPropertyName;
    private NLPText nlpText;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     * @param commandHandler

```

```

        */

        @Autowired
        public RemoveUnneedLexicalIssuesCommandImpl(AssistantFacade
assistantManager,
            UserRepository userRepository, ProjectRepository projectRepository,
            ProjectCommandFactory projectCommandFactory,
            AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
            super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
                annotationCommandFactory, commandHandler);
        }

        protected ProjectOrDomain getProjectOrDomain() {
            return projectOrDomain;
        }

        public void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
            this.projectOrDomain = projectOrDomain;
        }

        public ProjectOrDomainEntity getThingBeingAnalyzed() {
            return thingBeingAnalyzed;
        }

        public void setThingBeingAnalyzed(ProjectOrDomainEntity
thingBeingAnalyzed) {
            this.thingBeingAnalyzed = thingBeingAnalyzed;
        }

        protected String getAnnotatableEntityPropertyName() {
            return annotatableEntityPropertyName;
        }

        public void setAnnotatableEntityPropertyName(String
annotatableEntityPropertyName) {
            this.annotatableEntityPropertyName = annotatableEntityPropertyName;
        }

        protected NLPText getNlpText() {
            return nlpText;
        }

        public void setNlpText(NLPText nlpText) {
            this.nlpText = nlpText;
        }
    }
}

```

```

/**
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    ProjectOrDomainEntity thingBeingAnalyzed =
getRepository().get(getThingBeingAnalyzed());

    Set<Annotation> annotations = new
HashSet<Annotation>(thingBeingAnalyzed.getAnnotations());
    for (Annotation existingAnnotation : annotations) {
        if (existingAnnotation instanceof LexicalIssue) {
            LexicalIssue lexicalIssue = (LexicalIssue) existingAnnotation;
            if (!lexicalIssue.isResolved()) {
                String textInQuestion = lexicalIssue.getWord();
                if ((textInQuestion != null) && (textInQuestion.length() > 0)
                    && isIssueForProperty(lexicalIssue,
getAnnotatablePropertyName())) {
                    boolean matchFound = false;
                    for (NLPText nlpFragment : getNLPText().getLeaves()) {
                        while ((nlpFragment != null) && !
nlpFragment.equals(getNLPText())) {
                            String nlpFragmentText = nlpFragment.getText();
                            if (nlpFragmentText != null) {
                                log.info("comparing '" + nlpFragmentText + "' to '"
+ textInQuestion + "'");
                                // TODO: split out whitespace and compare
                                // word by word
                                if (nlpFragment.getText().equalsIgnoreCase(textInQuestion)) {
                                    matchFound = true;
                                    break;
                                }
                            }
                            nlpFragment = nlpFragment.getParent();
                        }
                        if (matchFound) {
                            break;
                        }
                    }
                    if (!matchFound) {
                        try {
                            log.info("found lexical issue that is no longer relevant: "
+ lexicalIssue);
                            RemoveAnnotationFromAnnotatableCommand command =
getAnnotationCommandFactory()
                                .newRemoveAnnotationFromAnnotatableCommand();
                            command.setAnnotatable(thingBeingAnalyzed);

```

```

                                command.setAnnotation(lexicalIssue);
                                command = getCommandHandler().execute(command);
                            } catch (Exception e) {
                                log.error("Unable to remove " + lexicalIssue + " from "
+ thingBeingAnalyzed, e);
                            }
                        }
                    }
                }
            }
        }
    }
}

private boolean isIssueForProperty(LexicalIssue issue, String
propertyName) {
    String issuePropertyName = issue.getAnnotatablePropertyName();
    if ((issuePropertyName == null) && (propertyName == null)) {
        return true;
    } else if ((issuePropertyName != null) &&
issuePropertyName.equals(propertyName)) {
        return true;
    }
    return false;
}
}

```

replaceglossarytermcommand.java

```

/*
 * $Id: ReplaceGlossaryTermCommand.java,v 1.3 2009/03/23 11:02:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.command;

import java.util.Set;

import edu.harvard.fas.rregan.requel.command.EditCommand;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;

/**
 * Take a glossary term and replace its text (name) in the referring
entities
 * with the text (name) of the canonical term.

```

```

/*
 * @author ron
 */
public interface ReplaceGlossaryTermCommand extends EditCommand {

    /**
     * When this is set the glossary term is updated to set the canonical
     * term
     * before replacing the text in the entities.
     *
     * @param canonicalTerm
     */
    public void setCanonicalTerm(GlossaryTerm canonicalTerm);

    /**
     * Set the glossary term to replace with the canonical term.
     *
     * @param glossaryTerm
     */
    public void setGlossaryTerm(GlossaryTerm glossaryTerm);

    /**
     * @return The glossary term after all the referring entities have
     * been
     *         updated.
     */
    public GlossaryTerm getGlossaryTerm();

    /**
     * @return The project entities that were changed.
     */
    public Set<ProjectOrDomainEntity> getUpdatedEntities();
}

```

replaceglossarytermcommandimpl.java

```

/*
 * $Id: ReplaceGlossaryTermCommandImpl.java,v 1.10 2009/03/30 11:54:28
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.command.ReplaceGlossaryTermComma
nd;
import edu.harvard.fas.rregan.requel.project.impl.ActorImpl;
import edu.harvard.fas.rregan.requel.project.impl.GlossaryTermImpl;
import edu.harvard.fas.rregan.requel.project.impl.GoalImpl;
import edu.harvard.fas.rregan.requel.project.impl.StoryImpl;
import edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantFacade;
import edu.harvard.fas.rregan.requel.user.UserRepository;

/**
 * @author ron
 */
@Controller("replaceGlossaryTermCommand")
@Scope("prototype")
public class ReplaceGlossaryTermCommandImpl extends
AbstractEditProjectCommand implements
ReplaceGlossaryTermCommand {

    private final Set<ProjectOrDomainEntity> updatedEntities = new
HashSet<ProjectOrDomainEntity>();
    private GlossaryTerm glossaryTerm;
    private GlossaryTerm canonicalTerm;

    /**
     * @param assistantManager
     * @param userRepository
     * @param projectRepository
     * @param projectCommandFactory
     * @param annotationCommandFactory
     */

```

```

 * @param commandHandler
 */
@.Autowired
public ReplaceGlossaryTermCommandImpl(AssistantFacade
assistantManager,
    UserRepository userRepository, ProjectRepository projectRepository,
    ProjectCommandFactory projectCommandFactory,
    AnnotationCommandFactory annotationCommandFactory, CommandHandler
commandHandler) {
    super(assistantManager, userRepository, projectRepository,
projectCommandFactory,
        annotationCommandFactory, commandHandler);
}

protected GlossaryTerm getCanonicalTerm() {
    return canonicalTerm;
}

@Override
public void setCanonicalTerm(GlossaryTerm canonicalTerm) {
    this.canonicalTerm = canonicalTerm;
}

@Override
public GlossaryTerm getGlossaryTerm() {
    return glossaryTerm;
}

@Override
public void setGlossaryTerm(GlossaryTerm glossaryTerm) {
    this.glossaryTerm = glossaryTerm;
}

@Override
public Set<ProjectOrDomainEntity> getUpdatedEntities() {
    return updatedEntities;
}

@Override
public void execute() {
    GlossaryTermImpl glossaryTerm = (GlossaryTermImpl)
getProjectRepository().get(
        getGlossaryTerm());
    GlossaryTerm canonicalTerm =
getProjectRepository().get(getCanonicalTerm());
    if (canonicalTerm == null) {
        canonicalTerm = glossaryTerm.getCanonicalTerm();
    }
}

    } else {
        glossaryTerm.setCanonicalTerm(canonicalTerm);
        canonicalTerm.getAlternateTerms().add(glossaryTerm);
    }

    for (ProjectOrDomainEntity entity : glossaryTerm.getReferers()) {
        if (entity instanceof Goal) {
            GoalImpl goal = (GoalImpl) entity;
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the goal name: " +
goal.getName());
            goal.setName(replaceText(goal.getName(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the goal text: " +
goal.getText());
            goal.setText(replaceText(goal.getText(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
            glossaryTerm.getReferers().remove(goal);
            canonicalTerm.getReferers().add(goal);
        } else if (entity instanceof Story) {
            StoryImpl story = (StoryImpl) entity;
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the story name: " +
story.getName());
            story.setName(replaceText(story.getName(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the story text: " +
story.getText());
            story.setText(replaceText(story.getText(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
            glossaryTerm.getReferers().remove(story);
            canonicalTerm.getReferers().add(story);
        } else if (entity instanceof Actor) {
            ActorImpl actor = (ActorImpl) entity;
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the actor name: " +
actor.getName());
            actor.setName(replaceText(actor.getName(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
            log.debug("replacing '" + glossaryTerm.getName() + "' with '" +
                canonicalTerm.getName() + "' in the actor text: " +
actor.getText());
            actor.setText(replaceText(actor.getText(), glossaryTerm.getName(),
canonicalTerm
                .getName()));
        }
    }
}

```

```

+ canonicalTerm.getName() + "' in the actor text: " +
actor.getText());
actor.setText(replaceText(actor.getText(), glossaryTerm.getName(),
canonicalTerm
    .getName()));
glossaryTerm.getReferers().remove(actor);
canonicalTerm.getReferers().add(actor);
} else {
throw new RuntimeException("Unsupported entity type " +
entity.getClass());
}
updatedEntities.add(entity);
}
}

// TODO: copied from
ResolveIssueWithChangeSpellingPositionCommandImpl,
// should move to a utility class and shared.
protected String replaceText(String textToChange, String fromText,
String toText) {

if (textToChange.contains(fromText)) {
StringBuilder sb = new StringBuilder(textToChange.length() -
fromText.length()
+ toText.length());
// TODO: what about white space or punctuation between the words?
// TODO: what about the case of the words being added, for example
// "The System" being added in the middle of a sentence.
int start =
textToChange.toLowerCase().indexOf(fromText.toLowerCase());
while (start != -1) {
sb.setLength(0);
int end = start + fromText.length();
sb.append(textToChange.substring(0, start));
sb.append(toText);
sb.append(textToChange.substring(end));
textToChange = sb.toString();
start = textToChange.indexOf(fromText);
}
}
return textToChange;
}
}

```

reportdownloadprovider.java

```

/*
 * $Id: ReportDownloadProvider.java,v 1.2 2008/12/13 00:41:05 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui.project;

import java.io.IOException;
import java.io.OutputStream;

import nextapp.echo2.app.filetransfer.DownloadProvider;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import
edu.harvard.fas.rregan.requel.project.command.GenerateReportCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.uiframework.MessageHandler;

/**
 * Execute a report generator and return the report in the supplied
output
 * stream.
 *
 * @author ron
 */
class ReportDownloadProvider implements DownloadProvider {
private final ProjectCommandFactory projectCommandFactory;
private final CommandHandler commandHandler;
private final ReportGenerator reportGenerator;
private final MessageHandler messageHandler;

ReportDownloadProvider(MessageHandler messageHandler,
ProjectCommandFactory projectCommandFactory, CommandHandler
commandHandler,
ReportGenerator reportGenerator) {
this.projectCommandFactory = projectCommandFactory;
this.commandHandler = commandHandler;
this.reportGenerator = reportGenerator;
this.messageHandler = messageHandler;
}

public String getContentType() {

```

```

    return "text/html";
}

public String getFileName() {
    return reportGenerator.getProjectOrDomain().getName() + ".html";
}

public int getSize() {
    return 0;
}

public void writeFile(OutputStream outputStream) throws IOException {
    GenerateReportCommand generateReportCommand = projectCommandFactory
        .newGenerateReportCommand();
    generateReportCommand.setReportGenerator(reportGenerator);
    generateReportCommand.setOutputStream(outputStream);
    try {
        generateReportCommand =
            commandHandler.execute(generateReportCommand);
    } catch (Exception e) {
        messageHandler.setGeneralMessage("Report generation failed: " + e);
    }
}
}

```

reportgenerator.java

```

package edu.harvard.fas.rregan.requel.project;

public interface ReportGenerator extends TextEntity,
Comparable<ReportGenerator> {
}

```

reportgeneratoreditorpanel.java

```

/*
 * $Id: ReportGeneratorEditorPanel.java,v 1.13 2009/03/27 13:43:14
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;

```

```

import nextapp.echo2.app.Button;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.filetransfer.UploadEvent;
import nextapp.echo2.app.filetransfer.UploadListener;
import nextapp.echo2.app.filetransfer.UploadSelect;

import org.apache.commons.io.IOUtils;
import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.command.DeleteReportGeneratorCom
mand;
import edu.harvard.fas.rregan.requel.project.command.EditReportGeneratorCom
mand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.uiframework.navigation.DownloadButton;
import edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

/**
 * @author ron
 */
public class ReportGeneratorEditorPanel extends
AbstractRequelProjectEditorPanel {
    private static final Logger log =
Logger.getLogger(ReportGeneratorEditorPanel.class);

    static final long serialVersionUID = 0L;
}

```

```

/**
 * The name to use in the ReportGeneratorEditorPanel.properties file
to set
 * the label of the name field. If the property is undefined "Name"
is used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the ReportGeneratorEditorPanel.properties file
to set
 * the label of the text field. If the property is undefined "Text"
is used.
 */
public static final String PROP_LABEL_TEXT = "Text.Label";

/**
 * The name to use in the ReportGeneratorEditorPanel.properties file
to set
 * the label of the run report button. If the property is undefined
"Run" is
 * used.
 */
public static final String PROP_LABEL_RUN_BUTTON = "RunButton.Label";

/**
 * The name to use in the ReportGeneratorEditorPanel.properties file
to set
 * the label of the upload button. If the property is undefined
"Upload" is
 * used.
 */
public static final String PROP_LABEL_UPLOAD_BUTTON =
"UploadButton.Label";

private UpdateListener updateListener;
private Button runButton;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public ReportGeneratorEditorPanel(CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
this(ReportGeneratorEditorPanel.class.getName(), commandHandler,
projectCommandFactory,
projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public ReportGeneratorEditorPanel(String resourceBundleName,
CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
super(resourceBundleName, ReportGenerator.class, commandHandler,
projectCommandFactory,
projectRepository);
}

/**
 * If the editor is editing an existing ReportGenerator the title
specified
 * in the properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
 * property is not set it then tries the standard PROP_PANEL_TITLE
and if
 * that does not exist it defaults to:<br>
 * "ReportGenerator: {0}"<br>
 * Valid variables are:<br>
 * {0} - ReportGenerator name<br>
 * {1} - project/domain name<br>
 * For new ReportGenerator it first tries
PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Report"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()

```

```

*/
@Override
public String getTitle() {
    if (getReportGenerator() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Document: {0}"));
        return MessageFormat.format(msgPattern,
            getReportGenerator().getName(),
            getProjectOrDomain().getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale())
                .getString(PROP_PANEL_TITLE, "New Document"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    ReportGenerator reportGenerator = getReportGenerator();
    if (reportGenerator != null) {
        addInput(EditReportGeneratorCommand.FIELD_NAME, PROP_LABEL_NAME,
            "Name",
            new TextField(), new
            StringDocumentEx(reportGenerator.getName()));

        try {
            UploadSelect uploadXslt = addInput("uploadXslt",
                PROP_LABEL_UPLOAD_BUTTON,
                "Upload", new UploadSelect(), null);
            uploadXslt.addUploadListener(new XsltUploadListener(this));
            uploadXslt.setEnabledSendButtonText(getResourceBundleHelper(getLoc
                ale()).getString(
                    PROP_LABEL_UPLOAD_BUTTON, "Upload"));
        } catch (Exception e) {
            log.error(e, e);
            setGeneralMessage("Problem initializing the upload button: " + e);
        }
    }

    addInput(EditReportGeneratorCommand.FIELD_TEXT, PROP_LABEL_TEXT,
        "Text",
        new TextArea(), new StringDocumentEx(reportGenerator.getText()));
}

```

```

        runButton = addActionButton(new
            DownloadButton(getResourceBundleHelper(getLocale())
                .getString(PROP_LABEL_RUN_BUTTON, "Run"), new
            ReportDownloadProvider(this,
                getProjectCommandFactory(), getCommandHandler(),
            getReportGenerator())));
        runButton.setEnabled(true);
    } else {
        addInput("name", PROP_LABEL_NAME, "Name", new TextField(), new
        StringDocumentEx());
    }

    try {
        UploadSelect uploadXslt = addInput("uploadXslt",
            PROP_LABEL_UPLOAD_BUTTON,
            "Upload", new UploadSelect(), null);
        uploadXslt.addUploadListener(new XsltUploadListener(this));
        uploadXslt.setEnabledSendButtonText(getResourceBundleHelper(getLoc
            ale()).getString(
                PROP_LABEL_UPLOAD_BUTTON, "Upload"));
    } catch (Exception e) {
        log.error(e, e);
        setGeneralMessage("Problem initializing the upload button: " + e);
    }
    addInput(EditReportGeneratorCommand.FIELD_TEXT, PROP_LABEL_TEXT,
        "Text",
        new TextArea(), new StringDocumentEx());
    runButton = addActionButton(new
        DownloadButton(getResourceBundleHelper(getLocale())
            .getString(PROP_LABEL_RUN_BUTTON, "Run"), new
        ReportDownloadProvider(this,
            getProjectCommandFactory(), getCommandHandler(),
        getReportGenerator())));
    runButton.setEnabled(false);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
            t.class,
            updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
        ass, updateListener);
}

@Override
public void dispose() {
}

```

```

super.dispose();
removeAll();
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
    updateListener = null;
}
}

@Override
public void cancel() {
super.cancel();
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
}

@Override
public void save() {
try {
    super.save();
    EditReportGeneratorCommand command = getProjectCommandFactory()
        .newEditReportGeneratorCommand();
    command.setReportGenerator(getReportGenerator());
    command.setName(getInputValue(EditReportGeneratorCommand.FIELD_NAME
, String.class));
    command.setText(getInputValue(EditReportGeneratorCommand.FIELD_TEXT
, String.class));
    command.setProjectOrDomain(getProjectOrDomain());
    command.setEditedBy(getCurrentUser());
    command = getCommandHandler().execute(command);
    setValid(true);
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
        updateListener);
        // TODO: remove other listeners?
    }
    getEventDispatcher().dispatchEvent(
        new UpdateEntityEvent(this, command.getReportGenerator()));
} catch (EntityException e) {
    if (e.isStaleEntity()) {
        // TODO: compare the original values before the user edited
        // to the current revisions values and if they are the same
}
}
}

```

```

// then update the new revision with the user's changes and
// continue, otherwise show the new changed value vs. the users
// new values.
String newName =
getInputValue(EditReportGeneratorCommand.FIELD_NAME, String.class);
String newText =
getInputValue(EditReportGeneratorCommand.FIELD_TEXT, String.class);
ReportGenerator newReportGenerator = getProjectRepository().get(
    getReportGenerator());

setTargetObject(newReportGenerator);
if (!newName.equals(newReportGenerator.getName()))
    || !newText.equals(newReportGenerator.getText())) {
    setGeneralMessage("The report was changed by another user and the
value conflicts with your input.");
    if (!newName.equals(newReportGenerator.getName())) {
        setValidationMessage(EditReportGeneratorCommand.FIELD_NAME,
"Your input '"
        + newName + "'");
        setInputValue(EditReportGeneratorCommand.FIELD_NAME,
newReportGenerator
        .getName());
    }
    if (!newText.equals(newReportGenerator.getText())) {
        setValidationMessage(EditReportGeneratorCommand.FIELD_TEXT,
"Your input '"
        + newText + "'";
        setInputValue(EditReportGeneratorCommand.FIELD_TEXT,
newReportGenerator
        .getText());
    }
} else {
    getEventDispatcher().dispatchEvent(
        new UpdateEntityEvent(this, newReportGenerator));
}

} else if ((e.getEntityPropertyNames() != null)
    && (e.getEntityPropertyNames().length > 0)) {
    for (String propertyName : e.getEntityPropertyNames()) {
        setValidationMessage(propertyName, e.getMessage());
    }
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
}
}

```

```

        }
    } else {
        setGeneralMessage(e.toString());
    }

} catch (Exception e) {
    log.error("could not save the reportGenerator: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
    try {
        DeleteReportGeneratorCommand deleteReportGeneratorCommand =
getProjectCommandFactory()
            .newDeleteReportGeneratorCommand();
        deleteReportGeneratorCommand.setEditedBy(getCurrentUser());
        deleteReportGeneratorCommand.setReportGenerator(getReportGenerator());
    };
    deleteReportGeneratorCommand = getCommandHandler()
        .execute(deleteReportGeneratorCommand);
    deleted = true;
    getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getReportGenerator()));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private ProjectOrDomain getProjectOrDomain() {
    if (getTargetObject() instanceof ProjectOrDomain) {
        return (ProjectOrDomain) getTargetObject();
    } else if (getTargetObject() instanceof ProjectOrDomainEntity) {
        return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
    }
    return null;
}

ReportGenerator getReportGenerator() {
    if (getTargetObject() instanceof ReportGenerator) {
        return (ReportGenerator) getTargetObject();
    }
    return null;
}

```

```

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ReportGeneratorEditorPanel panel;

    private UpdateListener(ReportGeneratorEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        ReportGenerator existingReportGenerator =
panel.getReportGenerator();
        if ((e instanceof UpdateEntityEvent) && (existingReportGenerator != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ReportGenerator updatedReportGenerator = null;
            if (event.getObject() instanceof ReportGenerator) {
                updatedReportGenerator = (ReportGenerator) event.getObject();
            if ((event instanceof DeletedEntityEvent)
                && existingReportGenerator.equals(updatedReportGenerator)) {
                panel.deleted = true;
                panel.getEventDispatcher().dispatchEvent(
                    new DeletedEntityEvent(this, panel, existingReportGenerator));
                return;
            }
            } else if (event.getObject() instanceof Annotation) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if
(existingReportGenerator.getAnnotations().contains(updatedAnnotation))
{
                        existingReportGenerator.getAnnotations().remove(updatedAnnotation);
                    }
                updatedReportGenerator = existingReportGenerator;
            } else if (updatedAnnotation.getAnnotatables()
                .contains(existingReportGenerator)) {
                for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                    if (annotatable.equals(existingReportGenerator)) {
                        updatedReportGenerator = (ReportGenerator) annotatable;
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }

    }

    if ((updatedReportGenerator != null)
        && updatedReportGenerator.equals(existingReportGenerator)) {
        // TODO: check the input fields to see if the user has made
        // a change before resetting the object and updating the
        // input fields.
        panel.setInputValue(EditReportGeneratorCommand.FIELD_NAME,
            updatedReportGenerator.getName());
        panel.setInputValue(EditReportGeneratorCommand.FIELD_TEXT,
            updatedReportGenerator.getText());
        panel.setTargetObject(updatedReportGenerator);
    }
}

private class XsltUploadListener implements UploadListener {
    static final long serialVersionUID = 0L;

    private final ReportGeneratorEditorPanel panel;

    private XsltUploadListener(ReportGeneratorEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void fileUpload(UploadEvent uploadEvent) {
        try {
            panel.setInputValue(EditReportGeneratorCommand.FIELD_NAME,
                uploadEvent
                    .getFileName());
            panel.setInputValue(EditReportGeneratorCommand.FIELD_TEXT, IOUtils
                .toString(uploadEvent.getInputStream()));
            panel.runButton.setEnabled(true);
            panel.setGeneralMessage("file " + uploadEvent.getFileName() + " "
                + uploaded);
        } catch (Exception e) {
            panel.setGeneralMessage("Could not upload file: " + e);
        }
    }

    @Override
    public void invalidFileUpload(UploadEvent uploadEvent) {
        panel.setGeneralMessage("Could not upload file.");
    }
}

```

```

        }

    }

}
```

reportgeneratorimpl.java

```

package edu.harvard.fas.rregan.requel.project.impl;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Generate a report for the project by transforming the project xml
 * via an
 * xslt.
 *
 * @author ron
 */
@Entity
@Table(name = "reports", uniqueConstraints =
{ @UniqueConstraint(columnNames =
{ "projectordomain_id", "name" }) })
@XmlRootElement(name = "report", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "report", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ReportGeneratorImpl extends AbstractTextEntity implements
ReportGenerator {
    static final long serialVersionUID = 0L;

    /**
     * @param projectOrDomain
     * @param createdBy
     */

```

```

 * @param name
 * @param text
 */
public ReportGeneratorImpl(ProjectOrDomain projectOrDomain, User
createdBy, String name,
    String text) {
    super(projectOrDomain, createdBy, name, text);
}

protected ReportGeneratorImpl() {
    super();
    // for reflection (hibernate, jaxb, etc.)
}

@Override
@Column(nullable = false, unique = false)
@NotEmpty(message = "a unique name is required.")
@XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return super.getName();
}

// hack for JAXB to set the name, for some reason it won't use the
inherited
// method.
@Override
public void setName(String name) {
    super.setName(name);
}

@Transient
@XmlID
@XmlAttribute(name = "id")
public String getXmlId() {
    return "RPT_" + getId();
}

@Override
@Transient
public String getDescription() {
    return "Report Generator: " + getName();
}

@Override
public int compareTo(ReportGenerator o) {
    return getName().compareToIgnoreCase(o.getName());
}

```

```

    }
}
```

reportgeneratornavigatorpanel.java

```

/*
 * $Id: ReportGeneratorNavigatorPanel.java,v 1.5 2009/03/27 13:43:14
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.DownloadButton;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
l;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class ReportGeneratorNavigatorPanel extends NavigatorTablePanel
{
    private static final Logger log =
Logger.getLogger(ReportGeneratorNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private final CommandHandler commandHandler;
    private final ProjectCommandFactory projectCommandFactory;
    private UpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the
     ReportGeneratorNavigatorPanel.properties to
     * set the label on the new ReportGenerator button.
     */
    public static final String PROP_NEW_REPORT_BUTTON_LABEL =
"NewReportButton.Label";

    /**
     * Property name to use in the
     ReportGeneratorNavigatorPanel.properties to
     * set the label for the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * Property name to use in the
     ReportGeneratorNavigatorPanel.properties to
     * set the label on the edit report button in each row of the table.
     */
    public static final String PROP_EDIT_REPORT_BUTTON_LABEL =
>EditReportButton.Label";

    /**
     * Property name to use in the
     ReportGeneratorNavigatorPanel.properties to
     * set the label on the view report button in each row of the table
     when the
     * user doesn't have edit permission.
     */
    public static final String PROP_VIEW_REPORT_BUTTON_LABEL =
"ViewReportButton.Label";

    /**
     * Property name to use in the
     ReportGeneratorNavigatorPanel.properties to
     * set the label on the run report button in each row of the table
     when the
     * user doesn't have edit permission.
     */
    public static final String PROP_RUN_REPORT_BUTTON_LABEL =
"RunReportButton.Label";

    /**
     * @param commandHandler
     * @param projectCommandFactory
     */
    public ReportGeneratorNavigatorPanel(CommandHandler commandHandler,
        ProjectCommandFactory projectCommandFactory) {
super(ReportGeneratorNavigatorPanel.class.getName(), Project.class,
    ProjectManagementPanelNames.PROJECT_REPORTS_NAVIGATOR_PANEL_NAME);
this.commandHandler = commandHandler;
this.projectCommandFactory = projectCommandFactory;
}

```

```

NavigatorTableConfig tableConfig = new NavigatorTableConfig();

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            ReportGenerator reportGenerator = (ReportGenerator) model
                .getBackingObject(row);
            String buttonLabel = null;
            if (isReadOnlyMode()) {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_VIEW_REPORT_BUTTON_LABEL, "View");
            } else {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_EDIT_REPORT_BUTTON_LABEL, "Edit");
            }
            NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                PanelActionType.Editor, reportGenerator,
                ReportGenerator.class,
                null, WorkflowDisposition.NewFlow);
            NavigatorButton openEditorButton = new
                NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
            openEditorButton.setStyleName(STYLE_NAME_PLAIN);
            RowLayoutData rld = new RowLayoutData();
            rld.setAlignment(Alignment.ALIGN_CENTER);
            openEditorButton.setLayoutData(rld);
            return openEditorButton;
        }
    });
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            ReportGenerator reportGenerator = (ReportGenerator) model
                .getBackingObject(row);
            String buttonLabel = null;
            buttonLabel = getResourceBundleHelper(getLocale()).getString(
                PROP_RUN_REPORT_BUTTON_LABEL, "Run");
            DownloadButton runReportButton = new DownloadButton(buttonLabel,
                new ReportDownloadProvider(ReportGeneratorNavigatorPanel.this,
                    getProjectCommandFactory(), getCommandHandler(),
                    reportGenerator));
            runReportButton.setStyleName(STYLE_NAME_PLAIN);
            RowLayoutData rld = new RowLayoutData();
            rld.setAlignment(Alignment.ALIGN_CENTER);
            runReportButton.setLayoutData(rld);
            return runReportButton;
        }
    }));
};

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            ReportGenerator reportGenerator = (ReportGenerator) model
                .getBackingObject(row);
            return reportGenerator.getName();
        }
    }));
;

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
                (AbstractProjectOrDomainEntity) model
                    .getBackingObject(row);
            return entity.getCreatedBy().getUsername();
        }
    }));
;

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
                (AbstractProjectOrDomainEntity) model
                    .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    }));
;

setTableConfig(tableConfig);
}

```

```

/**
 * Create a title for panel with dynamic information from the project
 * or
 * domain, by default the pattern is "Documents: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelpergetLocale().getString(PROP_PANEL_TITLE,
    "Documents: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
    Alignment.DEFAULT));
}

```

```

String closeButtonLabel =
getResourceBundleHelpergetLocale().getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
NavigationEvent closeEvent = new ClosePanelEvent(this, this);
NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
closeButton.setStyleName(STYLE_NAME_DEFAULT);
buttonsWrapper.add(closeButton);

if (!isReadOnlyMode()) {
    String newReportGeneratorButtonLabel =
getResourceBundleHelpergetLocale().getString(
        PROP_NEW_REPORT_BUTTON_LABEL, "Add");
    NavigationEvent openReportGeneratorEditor = new
OpenPanelEvent(this,
        PanelActionType.Editor, getProjectOrDomain(),
ReportGenerator.class, null,
        WorkflowDisposition.NewFlow);
    NavigatorButton newReportGeneratorButton = new NavigatorButton(
        newReportGeneratorButtonLabel, getEventDispatcher(),
openReportGeneratorEditor);
    newReportGeneratorButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(newReportGeneratorButton);
}

add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
User user = (User) getApp().getUser();
if (getProjectOrDomain() instanceof Project) {
    Project project = (Project) getProjectOrDomain();
    Stakeholder stakeholder = project.getUserStakeholder(user);
    if (stakeholder != null) {
        return !stakeholder.hasPermission(ReportGenerator.class,
            StakeholderPermissionType.Edit);
    }
}

```

```
}

return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(((ProjectOrDomain)
targetObject).getReportGenerators());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

/***
 * @return
 */
public CommandHandler getCommandHandler() {
    return commandHandler;
}

/***
 * @return
 */
public ProjectCommandFactory getProjectCommandFactory() {
    return projectCommandFactory;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ReportGeneratorNavigatorPanel panel;

    private UpdateListener(ReportGeneratorNavigatorPanel panel) {
        this.panel = panel;
    }

@Override
public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
```

```
if (event.getObject() instanceof ProjectOrDomain) {
    updatedPod = (ProjectOrDomain) event.getObject();
} else if (event.getObject() instanceof ProjectOrDomainEntity) {
    ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
    updatedPod = updatedEntity.getProjectOrDomain();
} else if (event.getObject() instanceof Annotation) {
    if (!(event instanceof DeletedEntityEvent)) {
        Annotation updatedAnnotation = (Annotation) event.getObject();
        for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
            if ((annotatable instanceof ProjectOrDomain)
&& annotatable.equals(panel.getProjectOrDomain())) {
                updatedPod = (ProjectOrDomain) annotatable;
                break;
            } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                    updatedPod = entity.getProjectOrDomain();
                    break;
                }
            }
        }
    }
}
if (panel.getProjectOrDomain().equals(updatedPod)) {
    panel.setTargetObject(updatedPod);
}
}
```

repository.java

```
package edu.harvard.fas.rregan.repository;

/**
 * A generic set of methods common to all repositories.
 *
 * @author ron
 */
public interface Repository {
```

```

/**
 * add an object to the repository.
 *
 * @param <T>
 * @param entity -
 *      the object to persist
 * @return an up to date copy of the entity.
 * @throws EntityException
 */
public <T> T persist(T entity) throws EntityException;

/**
 * take a previously persisted object and update its persisted copy
in the
 * repository with the supplied copy and return the latest version.
 *
 * @param <T>
 * @param entity
 * @return
 * @throws EntityException
 */
public <T> T merge(T entity) throws EntityException;

/**
 * takes a previously persisted object and returns the latest
 * version from the database.
 *
 * @param <T>
 * @param entity
 * @return
 * @throws EntityException
 */
public <T> T get(T entity) throws EntityException;

/**
 * takes a persistent object and initializes any lazy loaded
properties.
 * @param <T>
 * @param entity
 * @return
 * @throws EntityException
 */
public <T> T initialize(T entity) throws EntityException;

/**
 * remove the supplied object from the repository.

```

```

*
 * @param entity
 * @throws EntityException
 */
public void delete(Object entity) throws EntityException;

/**
 * force the repository to sync up any pending work, without actually
ending a transaction.
 * @throws EntityException
 */
public void flush() throws EntityException;
}


```

requelexception.java

```

/*
 * $Id: Requelexception.java,v 1.1 2008/03/14 10:26:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel;

import edu.harvard.fas.rregan.ApplicationException;

/**
 * @author ron
 */
public class Requelexception extends ApplicationException {
    static final long serialVersionUID = 0;

    /**
     * @param format
     * @param args
     */
    protected Requelexception(String format, Object... args) {
        super(format, args);
    }

    /**
     * @param cause
     * @param format
     * @param args
     */

```

```

protected RequelException(Throwable cause, String format, Object...
args) {
    super(cause, format, args);
}
}

```

requelmainscreen.java

```

/*
 * $Id: RequelMainScreen.java,v 1.7 2009/02/13 12:08:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.ContentPane;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.SplitPane;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.webcontainer.command.BrowserOpenWindowCommand;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Scope;

import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.PanelContainer;
import edu.harvard.fas.rregan.uiframework.login.LogoutEvent;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.screen.AbstractScreen;

```

```

/**
 * @author ron
 */
@org.springframework.stereotype.Component(RequelMainScreen.screenName)
@Scope("prototype")
public class RequelMainScreen extends AbstractScreen {
    private static final Logger log =
Logger.getLogger(RequelMainScreen.class);
    static final long serialVersionUID = 0;

    public static final String screenName = "mainScreen";

    public static final String STYLE_NAME_OUTER_SPLITTER =
"RequelMainScreen.OuterSplitter";
    public static final String STYLE_NAME_INNER_SPLITTER =
"RequelMainScreen.InnerSplitter";

    public static final String STYLE_NAME_CONTENT_PANEL_CONTAINER =
"RequelMainScreen.ContentPanelContainer";

    public static final String STYLE_NAME_BANNER_SPLIT_PANE =
"RequelMainScreen.Banner.SplitPane";
    public static final String STYLE_NAME_BANNER_LOGO_PANE =
"RequelMainScreen.Banner.LogoPane";
    public static final String STYLE_NAME_BANNER_LOGO_PANE_LAYOUT_ROW =
"RequelMainScreen.Banner.LogoPane.Row";

    /**
     * The name of the style to use for configuring the logo in the
     * banner panel
     * on the main screen in the Echo2 stylesheet. <br/> Example style:
<br/>
     *
     * <pre>
     * &lt;style name="RequelMainScreen.Logo"&gt;
     *   type="nextapp.echo2.app.Label";
     *   &lt;properties&gt;
     *     &lt;property name="icon";
     *       type="nextapp.echo2.app.ResourceImageReference";
     *       value="/resources/images/Logo209x100.png"; /&gt;
     *   &lt;/properties&gt;
     * &lt;/style&gt;
     * </pre>
     */
    public static final String STYLE_NAME_LOGO =
"RequelMainScreen.Banner.Logo";

```

```

public static final String STYLE_NAME_BANNER_LOGOUT_PANE =
"RequelMainScreen.Banner.LogoutPane";
public static final String STYLE_NAME_BANNER_LOGOUT_PANE_LAYOUT_ROW =
"RequelMainScreen.Banner.LogoutPane.Row";

/**
 * The name of the property to use for the logout button label from
the
 * specified resource (property) file.
 */
public static final String PROP_LABEL_LOGOUT_BUTTON =
"LogoutButton.Label";

/**
 * The name of the property to use for the logout button label from
the
 * specified resource (property) file.
 */
public static final String PROP_LABEL_EDIT_ACCOUNT_BUTTON =
>EditAccountButton.Label;

private final PanelContainer mainNavigationPanelContainer;
private final PanelContainer mainContentPanelContainer;

private SplitPane outerSplitter;
private SplitPane horizontalSplitter;

/**
 * TODO: this takes a Set of factories because Spring can only supply
a Map
 * with keyed strings through the AutoWire annotation.
 *
 * @param mainNavigationPanelContainer
 * @param mainContentPanelContainer
 */
@.Autowired
public RequelMainScreen(@Qualifier("mainNavigationPanelContainer")
PanelContainer mainNavigationPanelContainer,
@Qualifier("mainContentPanelContainer")
PanelContainer mainContentPanelContainer) {
    super(RequelMainScreen.class.getName());
    this.mainContentPanelContainer = mainContentPanelContainer;
    this.mainNavigationPanelContainer = mainNavigationPanelContainer;
}

@Override
public void setup() {
    super.setup();
    outerSplitter = new SplitPane();
    outerSplitter.setStyleName(STYLE_NAME_OUTER_SPLITTER);
    outerSplitter.add(getBanner());
    horizontalSplitter = new SplitPane();
    horizontalSplitter.setStyleName(STYLE_NAME_INNER_SPLITTER);
    mainNavigationPanelContainer.setup();
    horizontalSplitter.add((Component) mainNavigationPanelContainer);
    mainContentPanelContainer.setup();
    mainContentPanelContainer.setStyleName(STYLE_NAME_CONTENT_PANEL_CONT
AINER);
    horizontalSplitter.add((Component) mainContentPanelContainer);
    outerSplitter.add(horizontalSplitter);
    add(outerSplitter);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}

private Component getBanner() {
    SplitPane bannerPane = new SplitPane();
    bannerPane.setStyleName(STYLE_NAME_BANNER_SPLIT_PANE);

    // logo setup
    ContentPane logoPane = new ContentPane();
    logoPane.setStyleName(STYLE_NAME_BANNER_LOGO_PANE);
    Row logoPaneRow = new Row();
    logoPaneRow.setStyleName(STYLE_NAME_BANNER_LOGO_PANE_LAYOUT_ROW);
    Label logo = new Label();
    logo.setStyleName(STYLE_NAME_LOGO);
    logoPaneRow.add(logo);
    logoPane.add(logoPaneRow);
    bannerPane.add(logoPane);

    // logout button setup
    ContentPane logoutPane = new ContentPane();
    logoutPane.setStyleName(STYLE_NAME_BANNER_LOGOUT_PANE);
    // use a row to align the logout button to the far right
    Row logoutPaneRow = new Row();
    logoutPaneRow.setStyleName(STYLE_NAME_BANNER_LOGOUT_PANE_LAYOUT_ROW)
;

    NavigationEvent openUserEditor = new OpenPanelEvent(this,
PanelActionType.Editor, getApp())
}

```

```

    .getUser(), User.class, null, WorkflowDisposition.NewFlow);
NavigatorButton editAccountButton = new NavigatorButton(
    getResourceBundleHelpergetLocale()).getString(PROP_LABEL_EDIT_ACCOUNT_BUTTON,
    "Edit Account"), getEventDispatcher(), openUserEditor);
editAccountButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
logoutPaneRow.add(editAccountButton);

    NavigatorButton logoutButton = new
    NavigatorButton(getResourceBundleHelpergetLocale())
        .getString(PROP_LABEL_LOGOUT_BUTTON, "Logout"),
getEventDispatcher(),
    new LogoutEvent(this));
logoutButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
logoutPaneRow.add(logoutButton);
logoutPane.add(logoutPaneRow);
bannerPane.add(logoutPane);

    return bannerPane;
}

private Component getDevDocsButton() {
    Button docsButton = new Button("docs");
    docsButton.setStyleName("Plain");
    docsButton.setWidth(new Extent(80));
    docsButton.setAlignment(Alignment.ALIGN_LEFT);
    docsButton.addActionListener(new ActionListener() {
        static final long serialVersionUID = 0;

    public void actionPerformed(ActionEvent e) {
        try {
            /*
             * open window features width - Defines the width of the new
             * window (in pixels). height - Defines the height of the
             * new window (in pixels). directories - Defines whether the
             * directories (Links) toolbar is shown (yes | no) location -
             * Defines whether the location toolbar is shown (yes | no).
             * menubar - Defines whether the menu toolbar (File, Edit
             * etc.) is shown (yes | no). resizable - Defines whether
             * the user can resize the new window (yes | no). scrollbars -
             * Defines whether the new window has scrollbars (yes | no).
             * status - Defines whether the new window has a status bar
             * (yes | no). toolbar - Defines whether the general toolbar
             * (Back, Forward etc.) is shown (yes | no). left - Defines
             * how many pixels from the left that the new window
             * appears. top - Defines how many pixels from the top that
             * the new window appears. fullscreen - Opens popup in IE

```

```
* fullscreen mode (fullscreen=yes).  
*/  
String features =  
"height=440,width=350,resizable=yes,status=yes,location=yes,scrollbars  
=yes";  
    getApp().enqueueCommand(  
        new BrowserOpenWindowCommand("doc/html/index.html", "Dev Docs",  
            features));  
    } catch (Exception ex) {  
        // TODO: should this be propagated up?  
        log.error("Unexpected exception opening browser window: " + ex,  
ex);  
    }  
});  
return docsButton;  
}  
}
```

requeupdatedentitynotifier.java

```
/*
 * $Id: RequelUpdatedEntityNotifier.java,v 1.2 2009/01/23 09:54:27
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.ui.project;

import org.apache.log4j.Logger;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.requel.project.impl.assistant.UpdatedEntityNoti
fier;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * An updated entity notifier that will send UpdateEvents through the
UI
 * EventDispatcher to notify UI components that a project entity has
been
```

```

* updated.
*
* @author ron
*/
@Component("updatedEntityNotifier")
@Scope("prototype")
public class RequelUpdatedEntityNotifier implements
UpdatedEntityNotifier {
    private static final Logger log =
Logger.getLogger(RequelUpdatedEntityNotifier.class);

    private final EventDispatcher eventDispatcher;

    // TODO: the event dispatcher is a session context object and can't
be
    // assigned to a prototype object.
    // this is a prototype so that the assistant manager can be a
prototype
    // because of an exception commented
    // on at the top of the class:
    //
edu.harvard.fas.rregan.requel.project.impl.assistant.AssistantManager
/**
 * Create a new updated entity notifier with the event dispatcher for
the
 * current user session.
 *
 * @param eventDispatcher
 */
// @Autowired
// public RequelUpdatedEntityNotifier(EventDispatcher
eventDispatcher) {
// this.eventDispatcher = eventDispatcher;
// }
public RequelUpdatedEntityNotifier() {
    eventDispatcher = null;
}

/***
 * @see
edu.harvard.fas.rregan.requel.project.impl.assistant.UpdatedEntityNoti
fier#entityUpdated(edu.harvard.fas.rregan.requel.project.ProjectOrDoma
in)
 */
@Override
public void entityUpdated(ProjectOrDomain pod) {
    // TODO: this happens on a separate thread when when the

```

```

    // ApplicationInstance activeInstance
    // is null causing an IllegalStateException "Attempt to update state
of
    // application user
    // interface outside of user interface thread."
    // eventDispatcher.dispatchEvent(new UpdateEntityEvent(this, null,
    // pod));
    log.info("updated " + pod);
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.impl.assistant.UpdatedEntityNoti
fier#entityUpdated(edu.harvard.fas.rregan.requel.project.ProjectOrDoma
inEntity)
 */
@Override
public void entityUpdated(ProjectOrDomainEntity entity) {
    // TODO: this happens on a separate thread when when the
    // ApplicationInstance activeInstance
    // is null causing an IllegalStateException "Attempt to update state
of
    // application user
    // interface outside of user interface thread."
    // eventDispatcher.dispatchEvent(new UpdateEntityEvent(this, null,
    // entity));
    log.info("updated " + entity);
}

```

resolveyissuecommand.java

```

/*
 * $Id: ResolveIssueCommand.java,v 1.7 2009/02/13 12:08:01 rregan Exp
$
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.command;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.user.User;

```

```

/**
 * Resolve the supplied issue with the supplied position.
 *
 * @author ron
 */
public interface ResolveIssueCommand extends Command {

    /**
     * @return The annotatable to resolve the issue with
     */
    public Annotatable getAnnotatable();

    /**
     * The annotatable to resolve the issue with
     *
     * @param annotatable
     */
    public void setAnnotatable(Annotatable annotatable);

    /**
     * @return the resolved issue.
     */
    public Issue getIssue();

    /**
     * Set the issue to resolve with the position.
     *
     * @param issue
     */
    public void setIssue(Issue issue);

    /**
     * @param position -
     *          the position that resolves the issue.
     */
    public void setPosition(Position position);

    /**
     * @param user -
     *          the user resolving the issue.
     */
    public void setEditedBy(User user);
}

```

resolveissuecommandimpl.java

```

/*
 * $Id: ResolveIssueCommandImpl.java,v 1.14 2009/02/17 11:50:47 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import edu.harvard.fas.rregan.requel.annotation.Issue;
import edu.harvard.fas.rregan.requel.annotation.Position;
import edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFactory;
import edu.harvard.fas.rregan.requel.annotation.command.ResolveIssueCommand;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
@Controller("resolveIssueCommand")
@Scope("prototype")
public class ResolveIssueCommandImpl extends AbstractEditCommand
implements ResolveIssueCommand {

    private Annotatable annotatable;
    private Issue issue;
    private Position position;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     */
    @Autowired
    public ResolveIssueCommandImpl(CommandHandler commandHandler,

```

```
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository) {
    super(commandHandler, annotationCommandFactory, repository);
}

protected Position getPosition() {
    return position;
}

public void setPosition(Position position) {
    this.position = position;
}

public void setIssue(Issue issue) {
    this.issue = issue;
}

public Issue getIssue() {
    return issue;
}

@Override
public Annotatable getAnnotatable() {
    return annotatable;
}

@Override
public void setAnnotatable(Annotatable annotatable) {
    this.annotatable = annotatable;
}

@Override
public void execute() throws Exception {
    validate();
    User resolvedByUser = getRepository().get(getEditedBy());
    Position position = getRepository().get(getPosition());
    Issue issue = getRepository().get(getIssue());
    position.resolveIssue(issue, resolvedByUser);
    setPositiongetRepository().merge(position));
    setIssuegetRepository().merge(issue));
}

protected void validate() {
    if (getIssue() == null) {
        throw EntityValidationException.emptyRequiredProperty(Issue.class,
getIssue(), "issue",
EntityExceptionActionType.Updating);
    }
}
```

```

* to the project glossary.
*
* @author ron
*/
@Controller("resolveIssueWithAddActorPositionCommandImpl")
@Scope("prototype")
public class ResolveIssueWithAddActorPositionCommandImpl extends
ResolveIssueCommandImpl {

    private final ProjectCommandFactory projectCommandFactory;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param projectCommandFactory
     * @param repository
     */
    @Autowired
    public ResolveIssueWithAddActorPositionCommandImpl(CommandHandler
commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        ProjectCommandFactory projectCommandFactory, AnnotationRepository
repository) {
        super(commandHandler, annotationCommandFactory, repository);
        this.projectCommandFactory = projectCommandFactory;
    }

    @Override
    public LexicalIssue getIssue() {
        return (LexicalIssue) super.getIssue();
    }

    @Override
    protected AddActorPosition getPosition() {
        return (AddActorPosition) super.getPosition();
    }

    protected ProjectCommandFactory getProjectCommandFactory() {
        return projectCommandFactory;
    }

    @Override
    public void execute() throws Exception {
        LexicalIssue issue = getRepository().get(getIssue());
        User resolvedBy = getRepository().get(getEditedBy());
    }
}

```

```

        EditActorCommand command =
getProjectCommandFactory().newEditActorCommand();
        command.setProjectOrDomain((ProjectOrDomain)
issue.getGroupingObject());
        command.setName(issue.getWord());
        if (!issue.getAnnotatables().isEmpty()) {
            Set<ActorContainer> entities = new
HashSet<ActorContainer>(issue.getAnnotatables()
.size());
            for (Annotatable annotatable : issue.getAnnotatables()) {
                if (annotatable instanceof ActorContainer) {
                    entities.add((ActorContainer) annotatable);
                }
            }
            command.setAddActorContainers(entities);
        }
        command.setEditedBy(resolvedBy);
        command = getCommandHandler().execute(command);
        super.execute();
    }
}

```

resolveissuewithaddglossarytermpositioncommandimpl.java

```

/*
 * $Id: ResolveIssueWithAddGlossaryTermPositionCommandImpl.java,v 1.7
2009/02/13 12:07:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.command;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;

```

```

import
edu.harvard.fas.rregan.requel.annotation.impl.command.ResolveIssueComm
andImpl;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import
edu.harvard.fas.rregan.requel.project.command.EditGlossaryTermCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import
edu.harvard.fas.rregan.requel.project.impl.AddGlossaryTermPosition;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * This resolves an issue with the specified position by adding a word
or phrase
 * to the project glossary.
 *
 * @author ron
 */
@Controller("resolveIssueWithAddGlossaryTermPositionCommandImpl")
@Scope("prototype")
public class ResolveIssueWithAddGlossaryTermPositionCommandImpl
extends ResolveIssueCommandImpl {

    private final ProjectCommandFactory projectCommandFactory;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param projectCommandFactory
     * @param repository
     */
    @Autowired
    public
ResolveIssueWithAddGlossaryTermPositionCommandImpl(CommandHandler
commandHandler,
        AnnotationCommandFactory annotationCommandFactory,
        ProjectCommandFactory projectCommandFactory, AnnotationRepository
repository) {
        super(commandHandler, annotationCommandFactory, repository);
        this.projectCommandFactory = projectCommandFactory;
    }

    @Override
    public LexicalIssue getIssue() {
        return (LexicalIssue) super.getIssue();
    }

    /**
     * This resolves an issue with the specified position by adding a word
or phrase
     * to the project glossary.
     *
     * @param issue
     * @param resolvedBy
     * @param addReferers
     */
    public void resolveIssue(LexicalIssue issue, User resolvedBy, boolean
addReferers) {
        AddGlossaryTermPosition position = new AddGlossaryTermPosition();
        position.setIssue(issue);
        position.setResolvedBy(resolvedBy);
        position.setAddReferers(addReferers);

        EditGlossaryTermCommand command =
getProjectCommandFactory().newEditGlossaryTermCommand();
        command.setName(issue.getWord());
        if (!issue.getAnnotatables().isEmpty()) {
            Set<ProjectOrDomainEntity> entities = new
HashSet<ProjectOrDomainEntity>(issue
                .getAnnotatables().size());
            for (Annotatable annotatable : issue.getAnnotatables()) {
                if (annotatable instanceof ProjectOrDomainEntity) {
                    entities.add((ProjectOrDomainEntity) annotatable);
                }
            }
            command.setAddReferers(entities);
        }
        command.setProjectOrDomain((ProjectOrDomain)
issue.getGroupingObject());
        command.setEditedBy(resolvedBy);
        command = getCommandHandler().execute(command);
        super.execute();
    }
}

}

@Override
protected AddGlossaryTermPosition getPosition() {
    return (AddGlossaryTermPosition) super.getPosition();
}

protected ProjectCommandFactory getProjectCommandFactory() {
    return projectCommandFactory;
}

@Override
public void execute() throws Exception {
    LexicalIssue issue = getRepository().get(getIssue());
    User resolvedBy = getRepository().get(getEditedBy());

    EditGlossaryTermCommand command =
getProjectCommandFactory().newEditGlossaryTermCommand();
    command.setName(issue.getWord());
    if (!issue.getAnnotatables().isEmpty()) {
        Set<ProjectOrDomainEntity> entities = new
HashSet<ProjectOrDomainEntity>(issue
            .getAnnotatables().size());
        for (Annotatable annotatable : issue.getAnnotatables()) {
            if (annotatable instanceof ProjectOrDomainEntity) {
                entities.add((ProjectOrDomainEntity) annotatable);
            }
        }
        command.setAddReferers(entities);
    }
    command.setProjectOrDomain((ProjectOrDomain)
issue.getGroupingObject());
    command.setEditedBy(resolvedBy);
    command = getCommandHandler().execute(command);
    super.execute();
}
}

resolveyissuewithaddwordtodictionarypositioncommandimpl
/*
 * $Id: ResolveIssueWithAddWordToDictionaryPositionCommandImpl
1.7 2009/02/17 11:50:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;

```

resolveissuewithaddwordtodictionarypositioncommandimpl.java

```
/*
 * $Id: ResolveIssueWithAddWordToDictionaryPositionCommandImpl.java,v
1.7 2009/02/17 11:50:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditDictionaryWordComman
d;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.EntityValidationException;
import edu.harvard.fas.rregan.requel.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.requel.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.requel.annotation.impl.AddWordToDictionaryPosit
ion;
import edu.harvard.fas.rregan.requel.annotation.impl.LexicalIssue;

/**
 * This resolves an issue with the specified position by adding a word
to the
 * dictionary.
 *
 * @author ron
 */
@Controller("resolveIssueWithAddWordToDictionaryPositionCommand")
@Scope("prototype")
public class ResolveIssueWithAddWordToDictionaryPositionCommandImpl
extends ResolveIssueCommandImpl {

    private final DictionaryCommandFactory dictionaryCommandFactory;

    /**
     * @param commandHandler
     * @param annotationCommandFactory
     * @param repository
     * @param dictionaryCommandFactory
     */
    @Autowired
    public
ResolveIssueWithAddWordToDictionaryPositionCommandImpl(CommandHandler
commandHandler,

```

```

        AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository,
DictionaryCommandFactory dictionaryCommandFactory) {
super(commandHandler, annotationCommandFactory, repository);
this.dictionaryCommandFactory = dictionaryCommandFactory;
}

@Override
public LexicalIssue getIssue() {
return (LexicalIssue) super.getIssue();
}

@Override
protected AddWordToDictionaryPosition getPosition() {
return (AddWordToDictionaryPosition) super.getPosition();
}

@Override
public void execute() throws Exception {
validate();
EditDictionaryWordCommand command =
dictionaryCommandFactory.newEditDictionaryWordCommand();
command.setLemma(getIssue().getWord());
command = getCommandHandler().execute(command);
super.execute();
}

@Override
protected void validate() {
if (getIssue() == null) {
throw
EntityValidationException.emptyRequiredProperty(LexicalIssue.class,
getIssue(),
"issue", EntityExceptionActionType.Updating);
}
}
}

```

resolvissuewithchangespellingpositioncommandimpl.java

```

/*
 * $Id: ResolveIssueWithChangeSpellingPositionCommandImpl.java,v 1.8
2009/02/17 11:50:47 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.annotation.impl.command;
```

```

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

import javax.persistence.Transient;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.validation.EntityValidationException;
import edu.harvard.fas.rregan.validation.annotation.Annotatable;
import edu.harvard.fas.rregan.validation.annotation.AnnotationRepository;
import
edu.harvard.fas.rregan.validation.annotation.command.AnnotationCommandFact
ory;
import
edu.harvard.fas.rregan.validation.annotation.impl.AddWordToDictionaryPosit
ion;
import
edu.harvard.fas.rregan.validation.annotation.impl.ChangeSpellingPosition;
import edu.harvard.fas.rregan.validation.annotation.impl.LexicalIssue;

/**
 * @author ron
 */
@Controller("resolveIssueWithChangeSpellingPositionCommand")
@Scope("prototype")
public class ResolveIssueWithChangeSpellingPositionCommandImpl extends
ResolveIssueCommandImpl {

/**
 * @param commandHandler
 * @param annotationCommandFactory
 * @param repository
 */
@.Autowired
public
ResolveIssueWithChangeSpellingPositionCommandImpl(CommandHandler
commandHandler,
AnnotationCommandFactory annotationCommandFactory,
AnnotationRepository repository) {
super(commandHandler, annotationCommandFactory, repository);
}

```

```

@Override
public LexicalIssue getIssue() {
    return (LexicalIssue) super.getIssue();
}

@Override
protected ChangeSpellingPosition getPosition() {
    return (ChangeSpellingPosition) super.getPosition();
}

@Override
public void execute() throws Exception {
    validate();
    LexicalIssue issue = getAnnotationRepository().get(getIssue());
    if (getAnnotatable() == null) {
        // if a specific entity isn't specified fix the spelling in all of
        // them.
        for (Annotatable annotatable : issue.getAnnotatables()) {
            fixSpelling(annotatable, issue.getWord(),
            getPosition().getProposedWord());
        }
    } else {
        Annotatable annotatable = getRepository().get(getAnnotatable());
        fixSpelling(annotatable, issue.getWord(),
        getPosition().getProposedWord());
    }
    super.execute();
}

@Override
protected void validate() {
    if (getIssue() == null) {
        throw
EntityValidationException.emptyRequiredProperty(LexicalIssue.class,
getIssue(),
        "issue", EntityExceptionActionType.Updating);
    }
    if (getPosition() == null) {
        throw EntityValidationException.emptyRequiredProperty(
        AddWordToDictionaryPosition.class, getPosition(), "position",
        EntityExceptionActionType.Updating);
    }
}

protected void fixSpelling(Annotatable annotatable, String fromWord,
String toWord)
    throws Exception {

```

```

String textToChange = getTextToChange(annotatable);

if (textToChange.contains(fromWord)) {
    // TODO: need better determination for proper case
    // of word changing to.
    if (Character.isUpperCase(fromWord.charAt(0))) {
        toWord = toWord.substring(0, 1).toUpperCase() +
        toWord.substring(1);
    }

    StringBuilder sb = new StringBuilder(textToChange.length() -
fromWord.length())
        + toWord.length());
    int start = textToChange.indexOf(fromWord);
    while (start != -1) {
        sb.setLength(0);
        int end = start + fromWord.length();
        sb.append(textToChange.substring(0, start));
        sb.append(toWord);
        sb.append(textToChange.substring(end));
        textToChange = sb.toString();
        start = textToChange.indexOf(fromWord);
    }
    // TODO: should this use a command or will it always be invoked
from
    // inside a command?
    setChangedText(annotatable, textToChange);
}
}

@Transient
private String getTextToChange(Annotatable annotatable) throws
IllegalAccessException,
    InvocationTargetException {
    Method getter = getPropertyGetter(annotatable, getIssue()
        .getAnnotatableEntityPropertyName());
    getter.setAccessible(true);
    return (String) getter.invoke(annotatable, new Object[] {});
}

@Transient
private void setChangedText(Annotatable annotatable, String
changedText)
    throws IllegalAccessException, InvocationTargetException {
    Method setter = getPropertySetter(annotatable, getIssue()
        .getAnnotatableEntityPropertyName());
    setter.setAccessible(true);
}

```

```

        setter.invoke(annotatable, changedText);
    }

    @Transient
    private Method getPropertyGetter(Annotatable annotatable, String
propertyName) {
        Class<?> annotatableType = annotatable.getClass();
        while (annotatableType != null) {
            try {
                return annotatableType.getDeclaredMethod("get" + propertyName);
            } catch (NoSuchMethodException e) {
                annotatableType = annotatableType.getSuperclass();
            }
        }
        return null;
    }

    @Transient
    private Method getPropertySetter(Annotatable annotatable, String
propertyName) {
        Class<?> annotatableType = annotatable.getClass();
        while (annotatableType != null) {
            try {
                return annotatableType.getDeclaredMethod("set" + propertyName,
String.class);
            } catch (NoSuchMethodException e) {
                annotatableType = annotatableType.getSuperclass();
            }
        }
        return null;
    }
}

```

resourcebundlehelper.java

```

/*
 * $Id: ResourceBundleHelper.java,v 1.5 2009/03/27 07:16:09 rregan Exp
 */
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan;

import java.util.Locale;
import java.util.ResourceBundle;
import org.apache.log4j.Logger;

```

```

/**
 * @author ron
 */
public class ResourceBundleHelper {
    private static final Logger log =
Logger.getLogger(ResourceBundleHelper.class);

    private final String resourceBundleName;
    private ResourceBundle resourceBundle;

    /**
     * @param resourceBundleName
     */
    public ResourceBundleHelper(String resourceBundleName) {
        this.resourceBundleName = resourceBundleName;
        try {
            setLocale(null);
        } catch (ApplicationException e) {
            log.warn("no default resource file with bundle name " +
resourceBundleName);
        }
    }

    /**
     * @param key
     * @param defaultValue
     * @return
     */
    public Integer getInteger(String key, Integer defaultValue) {
        try {
            return (Integer) getResourceBundle().getObject(key);
        } catch (ClassCastException e) {
            try {
                return new Integer(getResourceBundle().getString(key));
            } catch (Exception e2) {
                return defaultValue;
            }
        } catch (Exception e) {
            return defaultValue;
        }
    }

    /**
     * @param key
     * @param defaultValue
     * @return
     */
    public Double getDouble(String key, Double defaultValue) {
        try {
            return (Double) getResourceBundle().getObject(key);
        } catch (ClassCastException e) {
            try {
                return new Double(getResourceBundle().getString(key));
            } catch (Exception e2) {
                return defaultValue;
            }
        } catch (Exception e) {
            return defaultValue;
        }
    }

    /**
     * @param key
     * @param defaultValue
     * @return
     */
    public Boolean getBoolean(String key, Boolean defaultValue) {
        try {
            return (Boolean) getResourceBundle().getObject(key);
        } catch (ClassCastException e) {
            try {
                return new Boolean(getResourceBundle().getString(key));
            } catch (Exception e2) {
                return defaultValue;
            }
        } catch (Exception e) {
            return defaultValue;
        }
    }

    /**
     * @param key
     * @param defaultValue
     * @return
     */
    public String getString(String key, String defaultValue) {
        try {
            return getResourceBundle().getString(key);
        } catch (Exception e) {
            return defaultValue;
        }
    }
}

```

```

/**
 * @param key
 * @return
 */
public String getString(String key) {
    return getResourceBundle().getString(key);
}

/**
 * Set the locale. This causes the resource bundle to be reloaded for
the
 * new locale.
 *
 * @param locale
 */
public void setLocale(Locale locale) {
    // TODO: don't reload the bundle if the same locale is set again
    try {
        if (locale != null) {
            log.debug("setting bundle " + getResourceBundleName() + " locale "
+ locale);
            setResourceBundle(ResourceBundle.getBundle(getResourceBundleName()
, locale));
        } else {
            log.debug("setting bundle " + getResourceBundleName() + " using
default locale");
            setResourceBundle(ResourceBundle.getBundle(getResourceBundleName()
));
        }
    } catch (Exception e) {
        throw
ApplicationException.missingResourceBundle(getResourceBundleName(),
e);
    }
}

protected String getResourceBundleName() {
    return resourceBundleName;
}

private ResourceBundle getResourceBundle() {
    return resourceBundle;
}

private void setResourceBundle(ResourceBundle resourceBundle) {
    this.resourceBundle = resourceBundle;
}

```

```

}
```

retryonlockfailurescommandhandler.java

```

/*
 * $Id: RetryOnLockFailuresCommandHandler.java,v 1.4 2009/03/22
11:08:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.command;

import java.lang.reflect.Method;

import javax.persistence.OptimisticLockException;

import org.apache.log4j.Logger;
import org.hibernate.StaleObjectStateException;
import org.hibernate.exception.LockAcquisitionException;
import org.springframework.dao.CannotAcquireLockException;
import
org.springframework.orm.hibernate3.HibernateOptimisticLockingFailureEx
ception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.Repository;
import edu.harvard.fas.rregan.requell.EntityLockException;
import edu.harvard.fas.rregan.requell.EntityValidationException;

/**
 * A command handler that wraps another command handler and traps
exceptions
 * related to locking and re-executes the original command in the
wrapped
 * handler.
 *
 * @author ron
 */
public class RetryOnLockFailuresCommandHandler implements
CommandHandler {
    private static final Logger log =
Logger.getLogger(RetryOnLockFailuresCommandHandler.class);

    private final CommandHandler commandHandler;
    private final Repository repository;

    /**

```

```

 * @param commandHandler
 * @param repository
 */
public RetryOnLockFailuresCommandHandler(CommandHandler commandHandler,
                                         Repository repository) {
    this.commandHandler = commandHandler;
    this.repository = repository;
}

public <T extends Command> T execute(T command) throws Exception {
    int retries = 0;
    Throwable[] thrown = new Throwable[3];
    while (true) {
        try {
            T returnCommand = commandHandler.execute(command);
            if ((retries > 0) && log.isInfoEnabled()) {
                StringBuilder sb = new StringBuilder();
                sb.append(getCommandInterfaceName(command.getClass()));
                sb.append(" succeeded after retry: ");
                for (int i = 0; i < retries; i++) {
                    sb.append("thrown[" + i + "]");
                }
                log.info(sb.toString());
            }
            return returnCommand;
        } catch (EntityValidationException e) {
            // the command failed during validation, don't retry
            throw e;
        } catch (EntityLockException e) {
            thrown[retries] = e;
            retries++;
            if (retries > 2) {
                throw e;
            } else {
                log.info(getCommandInterfaceName(command.getClass())
                    + " failed due to a failure to get a lock, retrying attempt: "
                    + retries);
            }
        } catch (EntityException e) {
            // the command failed during validation, don't retry
            throw e;
        } catch (LockAcquisitionException e) {
            // must catch hibernate and/or spring exceptions
        }
    }
}

```

```

Object updatedEntity = repository.get(entity);
// figure out what changed

// get the setter on the command and pass the updated
// object
for (Method method : getCommandInterface(command.getClass())
    .getDeclaredMethods()) {
    log.info(method.toGenericString());
    if ((method.getParameterTypes().length == 1)
        && method.getParameterTypes()[0].isAssignableFrom(entity
            .getClass())) {
        method.setAccessible(true);
        method.invoke(command, updatedEntity);
        break;
    }
}
} else {
    log.error(e + " the entity was null in the exception and could "
        + "not be recovered.", e);
}
} catch (StaleObjectStateException e) {
// must catch hibernate and/or spring exceptions
// because they are thrown from the commit and not
// wrapped by the exception adapters on the repositories

// TODO: this is mostly likely caused by the object being edited
// by the command as the command reloads indirect objects.
// Ideally we could check to see if the object properties being
// changed were changed in another transaction, and if not
// reload the object and rerun the command, otherwise fail
thrown[retries] = e;
retries++;
if (retries > 2) {
    throw e;
} else {
    log.info(getCommandInterfaceName(command.getClass())
        + " failed due to a stale object: " + e.getEntityName() + "#"
        + e.getIdentifier() + ", retrying attempt: " + retries);
}
} catch (HibernateOptimisticLockingFailureException e) {
// must catch hibernate and/or spring exceptions
// because they are thrown from the commit and not
// wrapped by the exception adapters on the repositories

// TODO: this is mostly likely caused by the object being edited
// by the command as the command reloads indirect objects.

```

```

// Ideally we could check to see if the object properties being
// changed were changed in another transaction, and if not
// reload the object and rerun the command, otherwise fail
thrown[retries] = e;
retries++;
if (retries > 2) {
    throw e;
} else {
    log.info(getCommandInterfaceName(command.getClass())
        + " failed due to a stale object: " +
        e.getPersistentClassName() + "#"
        + e.getIdentifier() + ", retrying attempt: " + retries);
}
} catch (Exception e) {
    log.error("unhandled exception in command handler: " + e, e);
    throw e;
}
}

private Class<?> getCommandInterface(Class<?> commandClass) {
    while (Command.class.isAssignableFrom(commandClass)) {
        for (Class<?> face : commandClass.getInterfaces()) {
            if (Command.class.isAssignableFrom(face)) {
                return face;
            }
        }
        commandClass = commandClass.getSuperclass();
    }
    return null;
}

private String getCommandInterfaceName(Class<?> commandClass) {
    Class<?> interfaceClass = getCommandInterface(commandClass);
    if (interfaceClass != null) {
        return interfaceClass.getSimpleName();
    }
    return "<non-command>";
}
}
```

scenario.java

```

/*
 * $Id: Scenario.java,v 1.5 2009/02/11 12:49:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.project;

import java.util.List;
import java.util.Set;

/**
 * A scenario defines a sequence of steps to complete a use case. Each
step may
 * contain alternate or exceptional sub-steps.
 *
 * @author ron
 */
public interface Scenario extends Step {

    /**
     * @return An ordered list of steps, each step is represented by a
scenario
     *         that may contain alternate or exception sub-steps.
     */
    public List<Step> getSteps();

    /**
     * @return The use cases that have this scenario as the primary.
     */
    public Set<UseCase> getUsingUseCases();

    /**
     * This function searches through all reachable scenarios at all
levels
     * below this scenario.
     *
     * @param step -
     *         a step or scenario to test.
     * @return true if the supplied step is a step in this scenario or in
any
     *         scenario it uses.
     */
    public boolean usesStep(Step step);
}


```

scenarioassistant.java

```

/*
 * $Id: ScenarioAssistant.java,v 1.4 2009/01/23 09:54:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses a scenario and adds annotations with suggestions.
 *
 * @author ron
 */
public class ScenarioAssistant extends ScenarioStepAssistant {

    /**
     * @param lexicalAssistant -
     *         assistant for analyzing text for spelling, terms and
other
     *         word oriented analysis.
     * @param scenarioStepAssistant -
     *         assistnat for analyzing the individual step text.
     * @param assistantUser -
     *         the user to use as the creator of the annotation
entities.
     */
    public ScenarioAssistant(LexicalAssistant lexicalAssistant, User
assistantUser) {
        super(ScenarioAssistant.class.getName(), lexicalAssistant,
assistantUser);
    }

    /**
     * @param scenario
     */
    @Override
    public void analyze() {
        super.analyze(); // step level analysis
        if (getEntity() instanceof Scenario) {
            Scenario scenario = (Scenario) getEntity();
            for (Step step : scenario.getSteps()) {
                // NOTE: this resets the entity of this analyzer to a child step
                // or scenario. The original is saved in the local scenario
                // variable on the stack. the text of the scenario was already
                // analyzed by super.analyze
                setEntity(step);
                analyze();
            }
        }
    }
}
```

```
    }
}
}
```

scenariocontainer.java

```
/*
 * $Id: ScenarioContainer.java,v 1.1 2008/10/08 21:55:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Comparator;
import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * A thing that can contain/refer to scenarios.
 *
 * @author ron
 */
public interface ScenarioContainer extends Describable, CreatedEntity
{
    /**
     * The scenarios referenced.
     *
     * @return
     */
    public Set<Scenario> getScenarios();

    /**
     * Compare the objects that contain Scenarios by the description.
     */
    public static final Comparator<ScenarioContainer> COMPARATOR = new
    ScenarioContainerComparator();

    /**
     * A Comparator for collections of Scenario containers.
     */
    public static class ScenarioContainerComparator implements
    Comparator<ScenarioContainer> {
        @Override
        public int compare(ScenarioContainer o1, ScenarioContainer o2) {
            return o1.getDescription().compareTo(o2.getDescription());
        }
    }
}
```

```
        return o1.getDescription().compareTo(o2.getDescription());
    }
}
}
```

scenarioeditorpanel.java

```
/*
 * $Id: ScenarioEditorPanel.java,v 1.28 2009/03/26 08:17:34 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import echopointng.tree.DefaultTreeModel;
import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
```

```

import
edu.harvard.fas.rregan.requel.project.command.CopyScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditScenarioCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.ui.annotation.AnnotationsTable;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;

/**
 * @author ron
 */
public class ScenarioEditorPanel extends
AbstractRequelProjectEditorPanel {
private static final Logger log =
Logger.getLogger(ScenarioEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the ScenarioEditorPanel.properties file to set
the
 * label of the name field. If the property is undefined "Name" is
used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the ScenarioEditorPanel.properties file to set
the
 * label of the scenario type field. If the property is undefined
"Scenario
 * Type" is used.
 */
public static final String PROP_LABEL_SCENARIO_TYPE =
"ScenarioType.Label";

```

```

/**
 * The name to use in the ScenarioEditorPanel.properties file to set
the
 * label of the text field. If the property is undefined "Text" is
used.
 */
public static final String PROP_LABEL_TEXT = "Text.Label";

private UpdateListener updateListener;
private Button copyButton;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public ScenarioEditorPanel(CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
this(ScenarioEditorPanel.class.getName(), commandHandler,
projectCommandFactory,
projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public ScenarioEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
super(resourceBundleName, Scenario.class, commandHandler,
projectCommandFactory,
projectRepository);
}

/**

```

```

    * If the editor is editing an existing Scenario the title specified
    in the
    * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
    property is
    * not set it then tries the standard PROP_PANEL_TITLE and if that
    does not
    * exist it defaults to:<br>
    * "Scenario: {0}"<br>
    * Valid variables are:<br>
    * {0} - Scenario name<br>
    * {1} - project/domain name<br>
    * For new Scenario it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
    * PROP_PANEL_TITLE and finally defaults to:<br>
    * "New Scenario"<br>
    *
    * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
    * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
    * @see Panel.PROP_PANEL_TITLE
    * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
/*
@Override
public String getTitle() {
    if (getScenario() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Scenario: {0}"));
        return MessageFormat.format(msgPattern, getScenario().getName(),
            getProjectOrDomain()
                .getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale())
                .getString(PROP_PANEL_TITLE, "New Scenario"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    Scenario scenario = getScenario();
    if (scenario != null) {
        addInput(EditScenarioCommand.FIELD_NAME, PROP_LABEL_NAME, "Name",
new TextField(),

```

```

        new StringDocumentEx(scenario.getName()));
        addInput("scenarioType", PROP_LABEL_SCENARIO_TYPE, "Type", new
SelectField(),
            new CombinedListModel(getScenarioTypeNames(),
scenario.getType().toString(),
            true));
        addInput(EditScenarioCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text",
new TextArea(),
            new StringDocumentEx(scenario.getText()));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABELGLOSSARYTERM,
            "Glossary Terms", new GlossaryTermsTable(this,
                getResourceBundleHelper(getLocale()), scenario));
        addMultiRowInput("scenarioSteps",
ScenarioStepsEditor.PROP_LABEL_SCENARIO_STEPS,
            "Scenario Steps", new ScenarioStepsEditor(getProjectOrDomain(),
this,
                getResourceBundleHelper(getLocale()), scenario));
        addMultiRowInput("scenarioUseCases",
ScenarioUseCasesTable.PROP_LABEL_SCENARIO_USECASES, "Referring
Use Cases",
            new ScenarioUseCasesTable(this,
                getResourceBundleHelper(getLocale()), scenario));
        addMultiRowInput("scenarioScenarios",
ScenarioScenariosTable.PROP_LABEL_SCENARIO_SCENARIOS, "Referring
Scenarios",
            new ScenarioScenariosTable(this,
                getResourceBundleHelper(getLocale()),
                scenario));
        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
            new AnnotationsTable(this, getResourceBundleHelper(getLocale()),
scenario));
        copyButton = addActionButton(new
Button(getResourceBundleHelper(getLocale()).getString(
            PROP_LABEL_COPY_BUTTON, "Copy")));
        copyButton.addActionListener(new CopyListener(this));
        copyButton.setEnabled(!isReadOnlyMode());
    } else {
        addInput(EditScenarioCommand.FIELD_NAME, PROP_LABEL_NAME, "Name",
new TextField(),
            new StringDocumentEx());
        addInput(
            "scenarioType",
            PROP_LABEL_SCENARIO_TYPE,
            "Type",
            new SelectField(),

```

```

        new CombinedListModel(getScenarioTypeNames(),
ScenarioType.Primary.name(), true));
    addInput(EditScenarioCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text",
new TextArea(),
    new StringDocumentEx());
    addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABEL_GLOSSARY_TERM,
    "Glossary Terms", new GlossaryTermsTable(this,
    getResourceBundleHelpergetLocale()), scenario);
    addMultiRowInput("scenarioSteps",
ScenarioStepsEditor.PROP_LABEL_SCENARIO_STEPS,
    "Scenario Steps", new ScenarioStepsEditor(getProjectOrDomain()),
this,
    getResourceBundleHelpergetLocale(), scenario);
    addMultiRowInput("scenarioUseCases",
    ScenarioUseCasesTable.PROP_LABEL_SCENARIO_USECASES, "Referring
Use Cases",
    new ScenarioUseCasesTable(this,
getResourceBundleHelpergetLocale()), scenario);
    addMultiRowInput("scenarioScenarios",
    ScenarioScenariosTable.PROP_LABEL_SCENARIO_SCENARIOS, "Referring
Scenarios",
    new ScenarioScenariosTable(this,
getResourceBundleHelpergetLocale()),
    scenario);
    addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
    new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
null);
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        ScenarioStepsEditor scenarioEditor = (ScenarioStepsEditor)
getInput("scenarioSteps");
        DefaultTreeModel treeModel = getInputModel("scenarioSteps",
DefaultTreeModel.class);
        MutableTreeNode rootNode = (MutableTreeNode) treeModel.getRoot();

        EditScenarioCommand command =
getProjectCommandFactory().newEditScenarioCommand();
        command.setProjectOrDomain(getProjectOrDomain());
        command.setScenario(getScenario());
        command.setEditedBy(getCurrentUser());
        command.setName(getInputValue(EditScenarioCommand.FIELD_NAME,
String.class));
        command.setText(getInputValue(EditScenarioCommand.FIELD_TEXT,
String.class));
        command.setScenarioTypeName(getInputValue("scenarioType",
String.class));
        command.setStepCommands(scenarioEditor.generateStepEditCommands(
            getProjectCommandFactory(), getProjectOrDomain(), rootNode));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
            // TODO: remove other listeners?
        }
    }
}

```

```

}
getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getScenario()));
} catch (EntityException e) {
if (e.isStaleEntity()) {
// TODO: compare the original values before the user edited
// to the current revisions values and if they are the same
// then update the new revision with the user's changes and
// continue, otherwise show the new changed value vs. the users
// new values.
String newName = get inputValue(EditScenarioCommand.FIELD_NAME,
String.class);
String newText = get inputValue(EditScenarioCommand.FIELD_TEXT,
String.class);
Scenario newScenario = getProjectRepository().get(getScenario());

setTargetObject(newScenario);
if (!newName.equals(newScenario.getName()))
|| !newText.equals(newScenario.getText())) {
setGeneralMessage("The scenario was changed by another user and
the value conflicts with your input.");
if (!newName.equals(newScenario.getName())) {
setValidationMessage(EditScenarioCommand.FIELD_NAME, "Your input
"
+ newName + "'");
setInputValue(EditScenarioCommand.FIELD_NAME,
newScenario.getName());
}
if (!newText.equals(newScenario.getText())) {
setValidationMessage(EditScenarioCommand.FIELD_TEXT, "Your input
"
+ newText + "'");
setInputValue(EditScenarioCommand.FIELD_TEXT,
newScenario.getText());
}
} else {
getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
newScenario));
}
} else if ((e.getEntityPropertyNames() != null)
&& (e.getEntityPropertyNames().length > 0)) {
for (String propertyName : e.getEntityPropertyNames()) {
setValidationMessage(propertyName, e.getMessage());
}
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException))

```

```

InvalidStateException ise = (InvalidStateException) e.getCause();
for (InvalidValue invalidValue : ise.getInvalidValues()) {
String propertyName = invalidValue.getPropertyName();
setValidationMessage(propertyName, invalidValue.getMessage());
}
} else {
setGeneralMessage(e.toString());
}

} catch (Exception e) {
log.error("could not save the scenario: " + e, e);
setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
try {
DeleteScenarioCommand deleteScenarioCommand =
getProjectCommandFactory()
.newDeleteScenarioCommand();
deleteScenarioCommand.setEditedBy(getCurrentUser());
deleteScenarioCommand.setScenario(getScenario());
deleteScenarioCommand =
getCommandHandler().execute(deleteScenarioCommand);
deleted = true;
getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getScenario()));
} catch (Exception e) {
setGeneralMessage("Could not delete entity: " + e);
}
}

private Set<String> getScenarioTypeNames() {
Set<String> scenarioTypeNames = new TreeSet<String>();
for (ScenarioType scenarioType : ScenarioType.values()) {
scenarioTypeNames.add(scenarioType.toString());
}
return scenarioTypeNames;
}

private ProjectOrDomain getProjectOrDomain() {
if (getTargetObject() instanceof ProjectOrDomain) {
return (ProjectOrDomain) getTargetObject();
} else if (getTargetObject() instanceof ProjectOrDomainEntity) {
return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
}
}

```

```

}

return null;
}

private Scenario getScenario() {
    if (getTargetObject() instanceof Scenario) {
        return (Scenario) getTargetObject();
    }
    return null;
}

private static class CopyListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ScenarioEditorPanel panel;

    private CopyListener(ScenarioEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            CopyScenarioCommand copyScenarioCommand =
                panel.getProjectCommandFactory()
                    .newCopyScenarioCommand();
            copyScenarioCommand.setEditedBy(panel.getCurrentUser());
            copyScenarioCommand.setOriginalScenario(panel.getScenario());
            copyScenarioCommand =
                panel.getCommandHandler().execute(copyScenarioCommand);
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null,
                    copyScenarioCommand.getNewScenario()));
            panel.getEventDispatcher().dispatchEvent(
                new OpenPanelEvent(this, PanelActionType.Editor,
                    copyScenarioCommand
                        .getNewScenario(), Scenario.class, null));
        } catch (Exception e) {
            panel.setGeneralMessage("Could not copy entity: " + e);
        }
    }
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ScenarioEditorPanel panel;

```

```

private UpdateListener(ScenarioEditorPanel panel) {
    this.panel = panel;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (panel.deleted) {
        return;
    }
    Scenario existingScenario = panel.getScenario();
    if ((e instanceof UpdateEntityEvent) && (existingScenario != null))
    {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        Scenario updatedScenario = null;
        if (event.getObject() instanceof Scenario) {
            updatedScenario = (Scenario) event.getObject();
            if ((event instanceof DeletedEntityEvent)
                && existingScenario.equals(updatedScenario)) {
                panel.deleted = true;
                panel.getEventDispatcher().dispatchEvent(
                    new DeletedEntityEvent(this, panel, existingScenario));
                return;
            }
        } else if (event.getObject() instanceof Goal) {
            Goal updatedGoal = (Goal) event.getObject();
            if (updatedGoal.getReferers().contains(existingScenario)) {
                for (GoalContainer gc : updatedGoal.getReferers()) {
                    if (gc.equals(existingScenario)) {
                        updatedScenario = (Scenario) gc;
                        break;
                    }
                }
            }
        } else if (event.getObject() instanceof Actor) {
            Actor updatedActor = (Actor) event.getObject();
            if (updatedActor.getReferers().contains(existingScenario)) {
                for (ActorContainer ac : updatedActor.getReferers()) {
                    if (ac.equals(existingScenario)) {
                        updatedScenario = (Scenario) ac;
                        break;
                    }
                }
            }
        } else if (event.getObject() instanceof Annotation) {
            Annotation updatedAnnotation = (Annotation) event.getObject();
            if (event instanceof DeletedEntityEvent) {

```

scenarioeditortreenodefactory.java

```
/*
 * $Id: ScenarioEditorTreeNodeFactory.java,v 1.11 2009/01/21 10:20:23
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
```

```
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.DefaultEditorTree
Node;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFac
tory;

/**
 * A factory for created EditorTree nodes for editing scenarios. It
extends
 * ScenarioStepEditorTreeNodeFactory and uses its createTreeNode() to
create the
 * actual node for the scenario.
 *
 * @author ron
 */
public class ScenarioEditorTreeNodeFactory extends
ScenarioStepEditorTreeNodeFactory {

    /**
     *
     */
    public ScenarioEditorTreeNodeFactory() {
        super(ScenarioEditorTreeNodeFactory.class.getName(),
Scenario.class);
    }

    /**
     * Create a tree node for editing the scenario's and adding a sub-
node for
     * editing each step in the scenario.
     */
}
```

```

 * @see
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFactory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.event.Dispatcher,
 *
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree,
 *      java.lang.Object)
 */
@Override
public MutableTreeNode createTreeNode(EventDispatcher eventDispatcher, EditorTree tree,
Object object) {
Scenario scenario = (Scenario) object;
EditorTreeNode scenarioNode = null;
if (scenario.equals(tree.getRootObject())) {
Row wrap = new Row();
wrap.setInsets(new Insets(10, 5, 10, 0));
wrap.add(new Label(scenario.getName()));
scenarioNode = new DefaultEditorTreeNode(eventDispatcher, wrap);
} else {
scenarioNode = (EditorTreeNode)
super.createTreeNode(eventDispatcher, tree, scenario);
}
for (Step step : scenario.getSteps()) {
EditorTreeNodeFactory factory =
tree.getEditorTreeNodeFactory(step);
scenarioNode.add(factory.createTreeNode(eventDispatcher, tree,
step));
}
return scenarioNode;
}
}

```

scenarioimpl.java

```

/*
 * $Id: ScenarioImpl.java,v 1.10 2009/02/11 12:49:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl;

import java.util.ArrayList;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;

```

```

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import org.hibernate.annotations.IndexColumn;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.Scenario")
@XmlRootElement(name = "scenario", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "scenario", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class ScenarioImpl extends StepImpl implements Scenario {
static final long serialVersionUID = 0L;

private List<Step> steps = new ArrayList<Step>();
private Set<UseCase> usedByUseCases = new TreeSet<UseCase>();

```

```

/**
 * Create a scenario.
 *
 * @param projectOrDomain
 * @param createdBy
 * @param name
 * @param text -
 *      short text description
 * @param scenarioType
 */
public ScenarioImpl(ProjectOrDomain projectOrDomain, User createdBy,
String name, String text,
ScenarioType scenarioType) {
super(Scenario.class.getName(), projectOrDomain, createdBy, name,
text, scenarioType);
projectOrDomain.getScenarios().add(this);
}

protected ScenarioImpl() {
// for hibernate
}

// hack Hibernate can't find projectOrDomain through inheritance but
needs
// them for "mappedBy"
// for AbstractProjectOrDomainEntity.scenarios
@Override
@XmlTransient
@ManyToOne(targetEntity = AbstractProjectOrDomain.class, cascade =
{ CascadeType.PERSIST,
CascadeType.REFRESH }, optional = false)
@JoinColumn(insertable = false, updatable = false)
public ProjectOrDomain getProjectOrDomain() {
return super.getProjectOrDomain();
}

@Override
protected void setProjectOrDomain(ProjectOrDomain projectOrDomain) {
super.setProjectOrDomain(projectOrDomain);
}

/**
 * @see edu.harvard.fas.rregan.requel.Describable#getDescription()
 */
@XmlTransient
@Transient

```

```

@Override
public String getDescription() {
return "Scenario: " + getName();
}

@Override
@XmlTransient
@OneToMany(targetEntity = UseCaseImpl.class, cascade =
{ CascadeType.MERGE,
CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "scenario")
public Set<UseCase> getUsingUseCases() {
return usedByUseCases;
}

protected void setUsingUseCases(Set<UseCase> usingUseCases) {
this.usedByUseCases = usingUseCases;
}

@Override
@XmlElementWrapper(name = "steps", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "stepRef", type = StepImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = StepImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "scenario_steps", joinColumns = { @JoinColumn(name =
"scenario_id") }, inverseJoinColumns = { @JoinColumn(name =
"step_id") })
@IndexColumn(name = "step_index", base = 0)
public List<Step> getSteps() {
return steps;
}

protected void setSteps(List<Step> steps) {
this.steps = steps;
}

@Override
public boolean usesStep(Step stepToFind) {
if (this.equals(stepToFind)) {
return true;
}
Deque<Scenario> scenariosToExamine = new LinkedList<Scenario>();
scenariosToExamine.add(this);

```

```

while (!scenariosToExamine.isEmpty()) {
    Scenario scenario = scenariosToExamine.pop();
    for (Step step : scenario.getSteps()) {
        if (stepToFind.equals(step)) {
            return true;
        }
        if (step instanceof Scenario) {
            scenariosToExamine.add((Scenario) step);
        }
    }
}
return false;
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */
@Override
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser,
    Object parent) {
    super.afterUnmarshal(userRepository, defaultCreatedByUser, parent);
    // this is a hack because JAXB doesn't ensure the parent is
completely
    // initialized
    // before calling the patchers and items nested more than one deep
will
    // never get
    // the project or domain set through JAXB so we must reach down the
    // hierarchy and
    // fill them in from a top level scenario
    if (parent instanceof ProjectOrDomain) {
        afterUnmarshalFixupChildren(this, (ProjectOrDomain) parent);
    }
}

```

```

private void afterUnmarshalFixupChildren(Scenario scenario,
ProjectOrDomain projectOrDomain) {
    for (Step step : scenario.getSteps()) {
        ((StepImpl) step).setProjectOrDomain(projectOrDomain);
        ((StepImpl) step).getUsingScenarios().add(this);
        if (step instanceof Scenario) {
            afterUnmarshalFixupChildren((Scenario) step, projectOrDomain);
        }
    }
}

/**
 * This class is used by JAXB to convert the id of an entity into an
xml id
 * string that will be distinct from other entity xml id strings by
the use
 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "SCN_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

scenariornavigatorpanel.java

```

/*
 * $Id: ScenarioNavigatorPanel.java,v 1.4 2009/02/23 07:37:23 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/

```

```

package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.project.UseCase;
import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
e;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class ScenarioNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(ScenarioNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private UpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the ScenarioNavigatorPanel.properties to
     * set the
     * label on the new Scenario button.
     */
    public static final String PROP_NEW_STORY_BUTTON_LABEL =
"NewScenarioButton.Label";

    /**
     * Property name to use in the ScenarioNavigatorPanel.properties to
     * set the
     * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * Property name to use in the ScenarioNavigatorPanel.properties to
     * set the
     * label on the edit Scenario button in each row of the table.
     */
    public static final String PROP_EDIT_STORY_BUTTON_LABEL =
>EditScenarioButton.Label";

```

```

/**
 * Property name to use in the ScenarioNavigatorPanel.properties to
set the
 * label on the view Scenario button in each row of the table when
the user
 * doesn't have edit permission.
 */
public static final String PROP_VIEW_STORY_BUTTON_LABEL =
"ViewScenarioButton.Label";

/**
 */
public ScenarioNavigatorPanel() {
    super(ScenarioNavigatorPanel.class.getName(), Project.class,
        ProjectManagementPanelNames.PROJECT_SCENARIOS_NAVIGATOR_PANEL_NAME
    );
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column, int row) {
            Scenario scenario = (Scenario) model.getBackingObject(row);
            String buttonLabel = null;
            if (isReadOnlyMode()) {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_VIEW_STORY_BUTTON_LABEL, "View");
            } else {
                buttonLabel = getResourceBundleHelper(getLocale()).getString(
                    PROP_EDIT_STORY_BUTTON_LABEL, "Edit");
            }
            NavigationEvent openEditorEvent = new OpenPanelEvent(this,
                PanelActionType.Editor, scenario, Scenario.class, null,
                WorkflowDisposition.NewFlow);
            NavigatorButton openEditorButton = new
                NavigatorButton(buttonLabel,
                    getEventDispatcher(), openEditorEvent);
            openEditorButton.setStyleName(STYLE_NAME_PLAIN);
            RowLayoutData rld = new RowLayoutData();
            rld.setAlignment(Alignment.ALIGN_CENTER);
            openEditorButton.setLayoutData(rld);
            return openEditorButton;
        }
    }));
}

```

```

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Top
Level",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column, int row) {
                Scenario scenario = (Scenario) model.getBackingObject(row);
                return (scenario.getUsingScenarios().size() == 0) ? "Yes" :
"No";
            }
        }));
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column, int row) {
                Scenario scenario = (Scenario) model.getBackingObject(row);
                return scenario.getName();
            }
        }));
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Type",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column, int row) {
                Scenario scenario = (Scenario) model.getBackingObject(row);
                return scenario.getType().toString();
            }
        }));
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column, int row) {
                AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                    .getBackingObject(row);
                return entity.getCreatedBy().getUsername();
            }
        }));
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",

```

```

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
    int row) {
        AbstractProjectOrDomainEntity entity =
        (AbstractProjectOrDomainEntity) model
        .getBackingObject(row);
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
        return format.format(entity.getDateCreated());
    }
});

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
 * or
 * domain, by default the pattern is "Scenarios: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "Scenarios: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
}

    updateListener = null;
}
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
    Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
    getEventDispatcher(),
        closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);

    if (!isReadOnlyMode()) {
        String newScenarioButtonLabel =
getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_STORY_BUTTON_LABEL, "Add");
        NavigationEvent openScenarioEditor = new OpenPanelEvent(this,
        PanelActionType.Editor,
            getProjectOrDomain(), Scenario.class, null,
        WorkflowDisposition.NewFlow);
        NavigatorButton newScenarioButton = new
NavigatorButton(newScenarioButtonLabel,
            getEventDispatcher(), openScenarioEditor);
        newScenarioButton.setStyleName(STYLE_NAME_DEFAULT);
        buttonsWrapper.add(newScenarioButton);
    }

    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
    }
    updateListener = new UpdateListener(this);
}

```

```

getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.class, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Scenario.class,
                StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(((ProjectOrDomain)
targetObject).getScenarios());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ScenarioNavigatorPanel panel;

    private UpdateListener(ScenarioNavigatorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {

```

```

                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
                            && annotatable.equals(panel.getProjectOrDomain())) {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                            if
                                (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())))
                                {
                                    updatedPod = entity.getProjectOrDomain();
                                    break;
                                }
                            }
                        }
                    }
                }
            }
            if (panel.getProjectOrDomain().equals(updatedPod)) {
                panel.setTargetObject(updatedPod);
            }
        }
    }
}
}

```

scenarioscenriostable.java

```

/*
 * $Id: ScenarioScenariosTable.java,v 1.4 2009/01/08 06:48:45 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

```

```

import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCellValueFactory;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColumnConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractComponentManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.ComponentManipulators;

/**
 * A table of the scenarios that use the supplied scenario.
 */
public class ScenarioScenariosTable extends AbstractRequelNavigatorTable {

```

```

    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(ScenarioScenariosTable.class,
            new ScenarioScenariosTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
     * the
     * ScenarioScenariosTable to define the label of the Story containers
     * field.
     * If the property is undefined the panel should use a sensible
     * default such
     * as "Scenario Scenarios".
     */
    public static final String PROP_LABEL_SCENARIO_SCENARIOS =
"ScenarioScenarios.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the view button in the scenario use cases edit table column. If
     * the
     * property is undefined "View" is used.
     */
    public static final String PROP_VIEW_USECASE_BUTTON_LABEL =
"ViewScenarioScenarios.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the edit button in the scenario use cases edit table column. If
     * the
     * property is undefined "Edit" is used.
     */
    public static final String PROP_EDIT_USECASE_BUTTON_LABEL =
>EditScenarioScenarios.Label";

    private Scenario scenario;
    private final NavigatorTable table;

    /**
     * @param editMode
     * @param resourceBundleHelper
     */

```

```

public ScenarioScenariosTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
super(editMode, resourceBundleHelper);
ColumnLayoutData layoutData = new ColumnLayoutData();
layoutData.setAlignment(Alignment.ALIGN_CENTER);
table = new NavigatorTable(getTableConfig());
table.setLayoutData(layoutData);
add(table);
}

protected Scenario getScenario() {
return scenario;
}

protected void setScenario(Scenario scenario) {
this.scenario = scenario;
if (scenario != null) {
table.setModel(new NavigatorTableModel((Collection)
scenario.getUsingScenarios()));
} else {
table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
}
}

private NavigatorTableConfig getTableConfig() {
NavigatorTableConfig tableConfig = new NavigatorTableConfig();
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("", 
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
Scenario scenario = (Scenario) model.getBackingObject(row);
String buttonLabel = null;
if (isReadOnlyMode()) {
buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_VIEW_USECASE_BUTTON_LABEL, "View");
} else {
buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_EDIT_USECASE_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor, scenario, scenario.getClass(), null,
WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
}
});
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
Scenario scenario = (Scenario) model.getBackingObject(row);
return scenario.getName();
}
}));
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
Scenario scenario = (Scenario) model.getBackingObject(row);
return scenario.getCreatedBy().getUsername();
}
}));
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
@Override
public Object getValueAt(NavigatorTableModel model, int column,
int row) {
Scenario scenario = (Scenario) model.getBackingObject(row);
DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
return formatter.format(scenario.getDateCreated());
}
}));
}
return tableConfig;
}

private static class ScenarioScenariosTableManipulator extends
AbstractComponentManipulator {

```

```

protected ScenarioScenariosTableManipulator() {
    super();
}

@Override
public Object getModel(Component component) {
    return getValue(component, Scenario.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getScenario());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setScenario((Scenario) value);
}

private ScenarioScenariosTable getComponent(Component component) {
    return (ScenarioScenariosTable) component;
}
}

```

scenarioselectorpanel.java

```

/*
 * $Id: ScenarioSelectorPanel.java,v 1.4 2009/02/23 07:37:23 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;

```

```

import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;

```

```

import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class ScenarioSelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(ScenarioSelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the ScenarioNavigatorPanel.properties to
     * set the
     * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * @param projectRepository
     */
    public ScenarioSelectorPanel(ProjectRepository projectRepository) {
        super(ScenarioSelectorPanel.class.getName(), Project.class,
              ProjectManagementPanelNames.PROJECT_SCENARIO_SELECTOR_PANEL_NAME);
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.setRowLevelSelection(true);

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Scenario scenario = (Scenario) model.getBackingObject(row);
                    return scenario.getName();
                }
            }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Top
Level",

```

```

            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Scenario scenario = (Scenario) model.getBackingObject(row);
                    return (scenario.getUsingScenarios().size() == 0) ? "Yes" :
"No";
                }
            }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Type",
            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Scenario scenario = (Scenario) model.getBackingObject(row);
                    return scenario.getType().toString();
                }
            }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Scenario scenario = (Scenario) model.getBackingObject(row);
                    return scenario.getCreatedBy().getUsername();
                }
            }));
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Scenario scenario = (Scenario) model.getBackingObject(row);
                    DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
                    return format.format(scenario.getDateCreated());
                }
            }));
        setTableConfig(tableConfig);
    }
}

```

```

 * Create a title for panel with dynamic information from the project
or
 * domain, by default the title is "Select Scenario"<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    return
getResourceBundleHelpergetLocale()).getString(PROP_PANEL_TITLE,
"Select Scenario");
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);
    add(buttonsWrapper);
}

```

```

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Scenario.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 *
 * @return an adapter to get the collection of items to select from
from the
 *         target object.
 */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
    return new NavigatorTableModelAdapter() {
        private ProjectOrDomain targetObject;

        @Override
        public Collection<Object> getCollection() {
            return (Collection) targetObject.getScenarios();
        }

        @Override
        public void setTargetObject(Object targetObject) {
            this.targetObject = (ProjectOrDomain) targetObject;
        }
    };
}

```

```

};

}

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the Scenario
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final ScenarioSelectorPanel panel;

    private UpdateListener(ScenarioSelectorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
&& annotatable.equals(panel.getProjectOrDomain())))
{

```

```
        updatedPod = (ProjectOrDomain) annotatable;
        break;
    } else if ((annotatable instanceof ProjectOrDomainEntity)) {
        ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
        if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
            updatedPod = entity.getProjectOrDomain();
            break;
        }
    }
}
}
}
if (panel.getProjectOrDomain().equals(updatedPod)) {
    panel.setTargetObject(updatedPod);
}
}
}
}
}
```

scenariostepassistant.java

```
/*
 * $Id: ScenarioStepAssistant.java,v 1.9 2009/02/09 10:12:31 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import java.util.LinkedList;
import java.util.Map;
import java.util.Queue;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.impl.srl.SemanticRoleCollector;
import
edu.harvard.fas.rregan.nlp.impl.srl.SemanticRoleCollectorFunction;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
```

```

import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses a scenario step and adds annotations with suggestions.
 *
 * @author ron
 */
public class ScenarioStepAssistant extends TextEntityAssistant {
    private static final Logger log =
Logger.getLogger(ProjectOrDomainEntityAssistant.class);

    /**
     * The name of the property in the ScenarioStepAssistant.properties
     * file for
     * the note text indicating the assistant couldn't analyze the
     * structure of
     * the step.
     */
    public static final String PROP_COULD_NOT_ANALYZE_STEP_MSG =
"CouldNotAnalyzeStepMessage";
    public static final String PROP_COULD_NOT_ANALYZE_STEP_MSG_DEFAULT =
"The assistant could not analyze the structure of the step text. It
may be too complex or didn't have an identifiable syntactic subject.";

    /**
     * The name of the property in the ScenarioStepAssistant.properties
     * file for
     * the issue text indicating the subject of the step doesn't match a
     * known
     * actor.
     */
    public static final String PROP SUBJECT DOESNT MATCH ACTOR MSG =
"SubjectDoesntMatchActorMessage";
    public static final String
PROP SUBJECT DOESNT MATCH ACTOR MSG DEFAULT = "The subject of the step
text \"\{0\}\" does not match a known actor.";

    /**
     * The name of the property in the ScenarioStepAssistant.properties
     * file for
     * the issue text indicating the assistant couldn't find.
     */
    public static final String PROP_ACTOR_NOT_IN_USECASE_MSG =
"ActorNotInUseCase";

```

```

        public static final String PROP_ACTOR_NOT_IN_USECASE_MSG_DEFAULT =
"The actor of the step \"\{0\}\" is not associated to the use
case \"\{1\}\\".';

        /**
         * @param resourceBundleName -
         *          the full class name to use for the resource bundle.
         * @param lexicalAssistant -
         *          assistant for analyzing text for spelling, terms and
other
         *          word oriented analysis.
         * @param assistantUser -
         *          the user to use as the creator of the annotation
entities.
         */
        protected ScenarioStepAssistant(String resourceBundleName,
LexicalAssistant lexicalAssistant,
        User assistantUser) {
        super(resourceBundleName, lexicalAssistant, assistantUser);
    }

    /**
     * @param step
     */
    @Override
    public void analyze() {
        super.analyze(); // analyze name and text
        if (getEntity() instanceof Step) {
            analyzeStep((Step) getEntity());
        }
    }

    private void analyzeStep(Step step) {
        // The step name should be of the form "<actor> does something"
        // identify the actor
        NLPText text = getPropertyNlpText(PROP_NAME);
        Map<SemanticRole, NLPText> roles = new SemanticRoleCollector(
            new
        SemanticRoleCollectorFunction(text.getPrimaryVerb())).process(text);
        NLPText agentText = roles.get(SemanticRole.AGENT);
        if ((agentText != null) && (agentText.getText().length() > 0)) {
            String subject = agentText.getText();

            boolean matchesExistingActor = false;
            for (Actor actor : step.getProjectOrDomain().getActors()) {
                if (subject.equals(actor.getName())) {
                    matchesExistingActor = true;
                }
            }
        }
    }
}

```

```

        Queue<Scenario> scenarios = new
LinkedList<Scenario>(step.getUsingScenarios());
for (Scenario scenario : scenarios) {
    scenarios.addAll(scenario.getUsingScenarios());
    for (UseCase useCase : scenario.getUsingUseCases()) {
        if (!useCase.getActors().contains(actor)
            && !actor.equals(useCase.getPrimaryActor())) {
            try {
                addSimpleIssue(step.getProjectOrDomain(), getAssistantUser(),
                    step, createMessage(PROP_ACTOR_NOT_IN_USECASE_MSG,
                        PROP_ACTOR_NOT_IN_USECASE_MSG_DEFAULT, actor
                            .getName(), useCase.getName()));
            } catch (Exception e2) {
                log.error("failed to add note indicating actor"
                    + " isn't associated with the usecase.", e2);
            }
        }
    }
}
if (!matchesExistingActor) {
    try {
        addSimpleIssue(step.getProjectOrDomain(), getAssistantUser(),
            step,
            createMessage(PROP_SUBJECT_DOESNT_MATCH_ACTOR_MSG,
                PROP_SUBJECT_DOESNT_MATCH_ACTOR_MSG_DEFAULT, subject));
    } catch (Exception e2) {
        log.error("failed to add note indicating step subject is not an
actor.", e2);
    }
}
} else {
    try {
        addNote(step.getProjectOrDomain(), getAssistantUser(), step,
            createMessage(
                PROP_COULD_NOT_ANALYZE_STEP_MSG,
                PROP_COULD_NOT_ANALYZE_STEP_MSG_DEFAULT));
    } catch (Exception e2) {
        log.error("failed to add note indicating failure of step
analysis.", e2);
    }
}
}
}

```

scenariostepeditor.java

```

/*
 * $Id: ScenarioStepEditor.java,v 1.10 2009/01/21 10:20:23 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import static
edu.harvard.fas.rregan.uiframework.panel.Panel.STYLE_NAME_DEFAULT;

import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.layout.RowStyleData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelComponent;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * An editor component for editing a scenario step's type and name.
This is
 * intended to be embedded in a larger component such as a Tree or
Table for
 * editing a collection of steps.
 *
 * @author ron
 */

```

```

public class ScenarioStepEditor extends AbstractRequestComponent {
    private static final Logger log =
Logger.getLogger(ScenarioStepEditor.class);
    static final long serialVersionUID = 0L;

    /**
     * The style name for step name text editor.
     */
    public static final String STYLE_NAME_STEP_NAME_EDITOR =
"ScenarioStepEditor.NameEditor";

    static {
        ComponentManipulators.setManipulator(ScenarioStepEditor.class,
            new ScenarioStepEditorManipulator());
    }

    private ScenarioStepEditorModel model;
    private SelectField scenarioTypeEditor;
    private TextArea nameEditor;
    private final Row layoutContainer = new Row();
    private final RowLayoutData rowLayoutTop = new RowLayoutData();

    /**
     * Create a scenario step editor for creating a new step.
     *
     * @param editMode
     *      see {@linkEditMode}
     * @param resourceBundleHelper
     */
    public ScenarioStepEditor(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
        rowLayoutTop.setAlignment(Alignment.ALIGN_TOP);
        setup();
        setModel(new ScenarioStepEditorModel(getScenarioTypeNames()));
    }

    /**
     * Create a scenario step editor for editing an existing step.
     *
     * @param editMode
     * @param resourceBundleHelper
     * @param step
     */
    public ScenarioStepEditor(EditMode editMode, ResourceBundleHelper
resourceBundleHelper,
        Step step) {

```

```

        super(editMode, resourceBundleHelper);
        setup();
        setModel(new ScenarioStepEditorModel(getScenarioTypeNames(), step));
    }

    private void setup() {
        layoutContainer.setCellSpacing(new Extent(5));
        scenarioTypeEditor = new SelectField();
        scenarioTypeEditor.setStyleName(STYLE_NAME_DEFAULT);
        scenarioTypeEditor.setLayoutData(rowLayoutTop);
        layoutContainer.add(scenarioTypeEditor);
        nameEditor = new TextArea();
        nameEditor.setStyleName(STYLE_NAME_STEP_NAME_EDITOR);
        nameEditor.setLayoutData(rowLayoutTop);
        // TODO: don't hard code this - it reflects the name db column size
        nameEditor.setMaximumLength(255);
        layoutContainer.add(nameEditor);
        add(layoutContainer);
    }

    /**
     * @return
     */
    public ScenarioStepEditorModel getModel() {
        return model;
    }

    /**
     * @param model
     */
    public void setModel(ScenarioStepEditorModel model) {
        this.model = model;
        scenarioTypeEditor.setModel(model.getScenarioTypeModel());
        scenarioTypeEditor.setSelectionModel(model.getScenarioTypeModel());
        nameEditor.setDocument(model.getNameModel());
    }

    private Set<String> getScenarioTypeNames() {
        Set<String> scenarioTypeNames = new TreeSet<String>();
        for (ScenarioType scenarioType : ScenarioType.values()) {
            scenarioTypeNames.add(scenarioType.toString());
        }
        return scenarioTypeNames;
    }

    // TODO: this may not be needed

```

```

private static class ScenarioStepEditorManipulator extends
AbstractComponentManipulator {

protected ScenarioStepEditorManipulator() {
    super();
}

@Override
public ScenarioStepEditorModel getModel(Component component) {
    return getComponent(component).getModel();
}

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setModel((ScenarioStepEditorModel)
valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // TODO
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getModel(component).getStep());
}

@Override
public void setValue(Component component, Object value) {
    getModel(component).setStep((Step) value);
}

private ScenarioStepEditor getComponent(Component component) {
    return (ScenarioStepEditor) component;
}
}
}

```

scenariostepeditormodel.java

```

/*
 * $Id: ScenarioStepEditorModel.java,v 1.2 2009/01/20 10:26:02 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.ui.project;

import java.util.Arrays;
import java.util.Collection;
import java.util.Set;
import java.util.TreeSet;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;

/**
 * A composite model of the models used to edit a step in the step
editor.
 *
 * @author ron
 */
public class ScenarioStepEditorModel {

private Step step;
private CombinedListModel scenarioTypeModel;
private StringDocumentEx nameModel;
private final Set<String> scenarioTypeNames = new TreeSet<String>();
private final String defaultScenarioTypeName =
ScenarioType.Primary.name();

public ScenarioStepEditorModel(Step step) {
    setStep(step);
}

public ScenarioStepEditorModel(Collection<String> scenarioTypeNames)
{
    setScenarioTypeNames(scenarioTypeNames);
    scenarioTypeModel = new CombinedListModel(getScenarioTypeNames(),
defaultScenarioTypeName,
    true);
    nameModel = new StringDocumentEx("");
}

public ScenarioStepEditorModel(Collection<String> scenarioTypeNames,
Step step) {
    setScenarioTypeNames(scenarioTypeNames);
    scenarioTypeModel = new CombinedListModel(getScenarioTypeNames(),
defaultScenarioTypeName,

```

```

        true);
nameModel = new StringDocumentEx("");
setStep(step);
}

public Step getStep() {
    return step;
}

public void setStep(Step step) {
    this.step = step;
    if (step != null) {
        scenarioTypeModel.setSelectedItem(step.getType().name());
        nameModel.setText(step.getName());
    } else {
        scenarioTypeModel.clearSelection();
        scenarioTypeModel.setSelectedItem(defaultScenarioTypeName);
        nameModel.setText("");
    }
}

public String getName() {
    return nameModel.getText();
}

public String getScenarioTypeName() {
    return (String)
scenarioTypeModel.get(scenarioTypeModel.getMinSelectedIndex());
}

public StringDocumentEx getNameModel() {
    return nameModel;
}

public CombinedListModel getScenarioTypeModel() {
    return scenarioTypeModel;
}

public void setScenarioTypeNames(String[] scenarioTypeNames) {
    setScenarioTypeNames(Arrays.asList(scenarioTypeNames));
}

protected void setScenarioTypeNames(Collection<String>
scenarioTypeNames) {
    this.scenarioTypeNames.clear();
    for (String scenarioTypeName : scenarioTypeNames) {
        this.scenarioTypeNames.add(scenarioTypeName);
    }
}

```

```

    }

    protected Set<String> getScenarioTypeNames() {
        return scenarioTypeNames;
    }
}

```

scenariostepeditortreenodefactory.java

```

/*
 * $Id: ScenarioStepEditorTreeNodeFactory.java,v 1.8 2009/01/21
09:23:18 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.requel.project.Step;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.AbstractEditorTre
eNodeFactory;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.DefaultEditorTree
Node;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFac
tory;

/**
 * A factory for created EditorTree nodes for editing scenario steps.
It creates
 * a ScenarioStepEditor and decorates it with a button for adding sub-
nodes
 * below this node and for removing the node from its parent.
*/

```

```

/*
 * @author ron
 */
public class ScenarioStepEditorTreeNodeFactory extends
AbstractEditorTreeNodeFactory {
private static final Logger log =
Logger.getLogger(ScenarioStepEditorTreeNodeFactory.class);

/**
 * The style name for the add child node button.
 */
public static final String STYLE_NAME_ADD_CHILD_NODE_BUTTON =
"ScenarioStepEditor.AddChildNodeButton";

/**
 * The style name for the remove node button.
 */
public static final String STYLE_NAME_REMOVE_NODE_BUTTON =
"ScenarioStepEditor.RemoveNodeButton";

/**
 *
 */
public ScenarioStepEditorTreeNodeFactory() {
this(ScenarioStepEditorTreeNodeFactory.class.getName(), Step.class);
}

/**
 * @param resourceBundleName
 * @param targetClass
 */
protected ScenarioStepEditorTreeNodeFactory(String
resourceBundleName, Class<?> targetClass) {
super(resourceBundleName, targetClass);
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFac
tory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.even
t.EventDispatcher,
*
edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree,
*
java.lang.Object)
*/
@Override

```

```

public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, EditorTree tree,
Object object) {
Step step = (Step) object;
ScenarioStepEditor stepEditor = new
ScenarioStepEditor(tree.getEditMode(),
getResourceBundleHelper(tree.getLocale()), step);
EditorTreeNode stepNode = new DefaultEditorTreeNode(eventDispatcher,
stepEditor);

// add a button that adds child nodes
stepNode = addEditorTreeNodeActionButtonDecorator(tree, stepNode,
"",
STYLE_NAME_ADD_CHILD_NODE_BUTTON, new AddChildActionListener(tree,
stepNode,
Step.class));

// add a button that removes this node
stepNode = addEditorTreeNodeActionButtonDecorator(tree, stepNode,
"",
STYLE_NAME_REMOVE_NODE_BUTTON, new RemoveNodeActionListener(tree,
stepNode));

return stepNode;
}

/**
 * A listener for adding new tree nodes.
*
 * @author ron
*/
public static class AddChildActionListener implements ActionListener
{
static final long serialVersionUID = 0L;

private final EditorTree editorTree;
private final EditorTreeNode treeNode;
private final Class<?> childTargetType;

protected AddChildActionListener(EditorTree tree, EditorTreeNode
treeNode,
Class<?> childTargetType) {
this.editorTree = tree;
this.treeNode = treeNode;
this.childTargetType = childTargetType;
}
}
```

```

@Override
public void actionPerformed(ActionEvent e) {
    EditorTreeNodeFactory factory =
editorTree.getEditorTreeNodeFactory(childTargetType);
    MutableTreeNode newNode =
factory.createTreeNode(treeNode.getEventDispatcher(),
    editorTree, null);
    editorTree.getModel().insertNodeInto(newNode, treeNode,
treeNode.getChildCount());
    // TODO: maybe the decorators cause this not to work?
    // editorTree.expandPath(new
    // TreePath(editorTree.getModel().getPathToRoot(treeNode)));
    editorTree.expandAll();
}
}

/**
 * A listener for removing a node from the tree.
 *
 * @author ron
 */
public static class RemoveNodeActionListener implements
ActionListener {
    static final long serialVersionUID = 0L;

    private final EditorTree editorTree;
    private final EditorTreeNode treeNode;

    protected RemoveNodeActionListener(EditorTree tree, EditorTreeNode
treeNode) {
        this.editorTree = tree;
        this.treeNode = treeNode;
    }

@Override
public void actionPerformed(ActionEvent e) {
    try {
        editorTree.getModel().removeNodeFromParent(treeNode);
        // TODO: maybe the decorators cause this not to work?
        // editorTree.expandPath(new
        // TreePath(editorTree.getModel().getPathToRoot(treeNode)));
        editorTree.expandAll();
    } catch (Exception ex) {
        log.error("Exception removing tree node: " + treeNode, ex);
    }
}
}

```

```

}
```

scenariostepseditor.java

```

/*
 * $Id: ScenarioStepsEditor.java,v 1.16 2009/03/26 08:17:34 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Button;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowHeaders;
import echopointng.tree.MutableTreeNode;
import echopointng.tree.TreePath;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.project.command.ConvertStepToScenarioCom
mand;
import edu.harvard.fas.rregan.requel.project.command.EditScenarioCommand;
import edu.harvard.fas.rregan.requel.project.command.EditScenarioStepCommand;
import edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelComponent;
```

```

import edu.harvard.fas.rregan.requel.ui.AbstractRequelController;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractComponentManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.ComponentManipulator;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.ComponentManipulators;
import edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTree;
import edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNode;
import edu.harvard.fas.rregan.uiframework.panel.editor.tree.EditorTreeNodeFactory;

/**
 * A table of the scenarios that use the supplied scenario.
 */
public class ScenarioStepsEditor extends AbstractRequelComponent {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(ScenarioStepsEditor.class,
            new ScenarioStepsEditorManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
     * the
     * ScenarioStepsEditor to define the label of the editor. If the
     * property is
     * undefined the panel should use a sensible default such as "Steps
     * Editor".
     */
    public static final String PROP_LABEL_SCENARIO_STEPS =
        "ScenarioSteps.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the add new step button in the scenario step editor. If the
     * property
     * is undefined "Add Step" is used.
     */
    public static final String PROP_ADD_STEP_BUTTON_LABEL =
        "AddStep.Label";

    /**
     * The name to use in the containing panels properties file to set
     * the label
     * of the add scenario button in the scenario step editor. If the
     * property
     * is undefined "Add Scenario" is used.
     */
    public static final String PROP_ADD_SCENARIO_BUTTON_LABEL =
        "AddScenario.Label";

    /**
     * TreeNodeFactories for creating scenario and step nodes in the
     * steps
     * editor.<br>
     * TODO: these should be injected through Spring
     */
    public static final Set<EditorTreeNodeFactory>
        scenarioTreeNodeFactories;
    static {
        Set<EditorTreeNodeFactory> treeNodeFactories = new
        HashSet<EditorTreeNodeFactory>();
        treeNodeFactories.add(new ScenarioEditorTreeNodeFactory());
        treeNodeFactories.add(new ScenarioStepEditorTreeNodeFactory());
        scenarioTreeNodeFactories =
        Collections.unmodifiableSet(treeNodeFactories);
    }
}

```

```

private Scenario scenario;
private final EditorTree editorTree;
private final Button addStepButton;
private final NavigatorButton addScenarioButton;
private final AddScenarioController addScenarioController;
private final Label messages;

/**
 * @param projectOrDomain
 * @param editMode
 * @param resourceBundleHelper
 * @param projectCommandFactory
 * @param commandHandler
 */
public ScenarioStepsEditor(ProjectOrDomain projectOrDomain, EditMode editMode,
    ResourceBundleHelper resourceBundleHelper) {
    super(editMode, resourceBundleHelper);
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    editorTree = new EditorTree(this, getEventDispatcher(),
        scenarioTreeNodeFactories, true,
        false, !editMode.isReadOnlyMode());
    add(editorTree);
    RowLayoutData buttonLayoutData = new RowLayoutData();
    buttonLayoutData.setAlignment(Alignment.ALIGN_LEFT);
    Row buttonLayout = new Row();
    buttonLayout.setCellSpacing(new Extent(5));
    buttonLayout.setInsets(new Insets(0, 5));
    addStepButton = new
    Button(resourceBundleHelper.getLocale().getString(
        PROP_ADD_STEP_BUTTON_LABEL, "Add Step"));
    addStepButton.setStyleName(Panel.STYLE_NAME_DEFAULT);

    // when the add button gets clicked, add a new step editor node to
    // the
    // bottom of the root of the tree.
    addStepButton.addActionListener(new ActionListener() {
        static final long serialVersionUID = 0L;

        @Override
        public void actionPerformed(ActionEvent e) {
            messages.setText("");
            MutableTreeNode rootNode = editorTree.getRootNode();
            EditorTreeNodeFactory factory =
                editorTree.getEditorTreeNodeFactory(Step.class);

```

```

            MutableTreeNode newNode =
factory.createTreeNode(getEventDispatcher(), editorTree,
    null);
editorTree.getModel().insertNodeInto(newNode, rootNode,
rootNode.getChildCount());
editorTree.expandPath(new
TreePath(editorTree.getModel().getPathToRoot(rootNode)));
}
});
buttonLayout.add(addStepButton);

String buttonLabel = getResourceBundleHelper(getLocale()).getString(
    PROP_ADD_SCENARIO_BUTTON_LABEL, "Add Scenario");

addScenarioButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
addScenarioButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
addScenarioButton.setVisible(false);

NavigationEvent openScenarioSelectorEvent = new OpenPanelEvent(this,
    PanelActionType.Selector, projectOrDomain, Project.class,
    ProjectManagementPanelNames.PROJECT_SCENARIO_SELECTOR_PANEL_NAME,
    WorkflowDisposition.ContinueFlow);

addScenarioButton.setEventToFire(openScenarioSelectorEvent);
addScenarioButton.setVisible(true);
AddScenarioController = new
AddScenarioController(getEventDispatcher(), this, editorTree);
getEventDispatcher().addEventTypeActionListener(SelectEntityEvent.cl
ass,
    addScenarioController, this);
buttonLayout.add(addScenarioButton);
add(buttonLayout);
messages = new Label();
messages.setStyleName(Panel.STYLE_NAME_VALIDATION_LABEL);
add(messages);
}

protected Scenario getScenario() {
    return scenario;
}

protected void setScenario(Scenario scenario) {
    this.scenario = scenario;
    editorTree.setRootObject(scenario);
    addScenarioController.setRootScenario(scenario);
}
}

```

```

/**
 * iterate over the scenario steps from the given node generating
commands
 * for editing the step or scenario.
 *
 * @param projectCommandFactory
 * @param projectOrDomain
 * @param stepEditorModel
 * @param rootNode
 * @return
 */
public List<EditScenarioStepCommand> generateStepEditCommands(
    ProjectCommandFactory projectCommandFactory, ProjectOrDomain
projectOrDomain,
    MutableTreeNode rootNode) {
    List<EditScenarioStepCommand> stepEditCommands = new
ArrayList<EditScenarioStepCommand>();
    if (rootNode != null) {
        Enumeration<MutableTreeNode> enm = rootNode.children();
        while (enm.hasMoreElements()) {
            MutableTreeNode node = enm.nextElement();
            if (node instanceof EditorTreeNode) {
                EditorTreeNode editorNode = (EditorTreeNode) node;
                Component editor = editorNode.getEditor();
                ComponentManipulator man =
ComponentManipulators.getManipulator(editor);
                ScenarioStepEditorModel stepModel = (ScenarioStepEditorModel) man
                    .getModel(editor);
                Step step = man.getValue(editor, Step.class);
                if (node.getChildCount() > 0) {
                    // if there are children then this is a scenario
                    if (step instanceof Scenario) {
                        EditScenarioCommand command = projectCommandFactory
                            .newEditScenarioCommand();
                        command.setProjectOrDomain(projectOrDomain);
                        command.setEditedBy(getCurrentUser());
                        command.setName(stepModel.getName());
                        command.setScenarioType(stepModel.getScenarioType());
                        command.setScenario((Scenario) step);
                        command.setStepCommands(generateStepEditCommands(projectCommand
Factory,
                            projectOrDomain, node));
                        stepEditCommands.add(command);
                    } else {
                        ConvertStepToScenarioCommand command = projectCommandFactory
                            .newConvertStepToScenarioCommand();

```

```

                        command.setProjectOrDomain(projectOrDomain);
                        command.setOriginalScenarioStep(step);
                        command.setEditedBy(getCurrentUser());
                        command.setName(stepModel.getName());
                        command.setScenarioType(stepModel.getScenarioType());
                        command.setStepCommands(generateStepEditCommands(projectCommand
Factory,
                            projectOrDomain, node));
                        stepEditCommands.add(command);
                    }
                } else {
                    // a step
                    EditScenarioStepCommand command = projectCommandFactory
                        .newEditScenarioStepCommand();
                    command.setProjectOrDomain(projectOrDomain);
                    command.setEditedBy(getCurrentUser());
                    command.setName(stepModel.getName());
                    command.setScenarioType(stepModel.getScenarioType());
                    command.setStep(step);
                    stepEditCommands.add(command);
                }
            }
        }
    }
    return stepEditCommands;
}

/**
 * Controller to add an existing scenario from the scenario selector,
 * initiated from the addScenario button.
 *
 * @author ron
 */
private static class AddScenarioController extends
AbstractRequelController {
    static final long serialVersionUID = 0L;

    private final ScenarioStepsEditor editor;
    private final EditorTree editorTree;
    private Scenario rootScenario;

    /**
     * @param eventDispatcher
     * @param editorTree
     * @param rootScenario
     */

```

```

public AddScenarioController(EventDispatcher eventDispatcher,
ScenarioStepsEditor editor,
    EditorTree editorTree) {
super(eventDispatcher);
this.editorTree = editorTree;
this.editor = editor;
}

public void setRootScenario(Scenario rootScenario) {
this.rootScenario = rootScenario;
}

@Override
public void actionPerformed(ActionEvent event) {
editor.messages.setText("");
if (event instanceof SelectEntityEvent) {
SelectEntityEvent selectEntityEvent = (SelectEntityEvent) event;
if (selectEntityEvent.getObject() instanceof Scenario) {
Scenario scenario = (Scenario) selectEntityEvent.getObject();
// make sure the selected scenario or its steps don't use
// the
// original root scenario, NOTE: the rootScenario may be
// null
// if this is a new scenario. In that case no need to check.
if ((rootScenario != null) && scenario.usesStep(rootScenario)) {
editor.messages
.setText("This scenario is a sub-step of the selected scenario
"
+ "and would add a loop in the steps, so it can't be add.");
return;
}
MutableTreeNode rootNode = editorTree.getRootNode();
EditorTreeNodeFactory factory = editorTree
.getEditorTreeNodeFactory(Scenario.class);
MutableTreeNode newNode =
factory.createTreeNode(getEventDispatcher(),
editorTree, scenario);
editorTree.getModel().insertNodeInto(newNode, rootNode,
rootNode.getChildCount());
editorTree.expandPath(new TreePath(editorTree.getModel()
.getPathToRoot(rootNode)));
}
}
}
}

```

```
private static class ScenarioStepsEditorManipulator extends AbstractComponentManipulator {

    protected ScenarioStepsEditorManipulator() {
        super();
    }

    @Override
    public Object getModel(Component component) {
        return getComponent(component).editorTree.getModel();
    }

    @Override
    public void setModel(Component component, Object valueModel) {
        setValue(component, valueModel);
    }

    @Override
    public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
        // this gets handled by the individual step editors.
    }

    @Override
    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getComponent(component).getScenario());
    }

    @Override
    public void setValue(Component component, Object value) {
        getComponent(component).setScenario((Scenario) value);
    }

    private ScenarioStepsEditor getComponent(Component component) {
        return (ScenarioStepsEditor) component;
    }
}
}
```

scenariotype.java

```

/**
 * The type/disposition of a scenario or step, such as a primary,
alternative,
* or exceptional case.
*
* @author ron
*/
public enum ScenarioType {

    /**
     * Indicate that this is a precursor to the scenario. If a
precondition
     * isn't meet the scenario cannot begin.
    */
    PreCondition,

    /**
     * A successful scenario or step that is the primary expected case.
    */
    Primary,

    /**
     * An optional step in a scenario.
    */
    Optional,

    /**
     * A successful scenario or step that is an alternative to the
primary
     * expected case.
    */
    Alternative,

    /**
     * A scenario or step indicating a failed interaction.
    */
    Exception;

    private ScenarioType() {
    }
}

```

scenariousecasetable.java

```
/*
```

```

 * $Id: ScenarioUseCasesTable.java,v 1.4 2009/01/08 06:48:46 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;

```

```

import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component to add to Panels of Story container entity editors to
enable
 * editing of the Stories of the entity.
 *
 * @author ron
 */
public class ScenarioUseCasesTable extends
AbstractRequestNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(ScenarioUseCasesTable.class,
            new ScenarioUseCasesTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * ScenarioUseCasesTable to define the label of the Story containers
field.
     * If the property is undefined the panel should use a sensible
default such
     * as "Scenario UseCases".
    */
    public static final String PROP_LABEL_SCENARIO_USECASES =
"ScenarioUseCases.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the view button in the scenario use cases edit table column. If
the
     * property is undefined "View" is used.
    */
    public static final String PROP_VIEW_USECASE_BUTTON_LABEL =
"ViewScenarioUseCases.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the edit button in the scenario use cases edit table column. If
the

```

```

     * property is undefined "Edit" is used.
    */
    public static final String PROP_EDIT_USECASE_BUTTON_LABEL =
>EditScenarioUseCases.Label";

    private Scenario scenario;
    private final NavigatorTable table;

    /**
     * @param editMode
     * @param resourceBundleHelper
    */
    public ScenarioUseCasesTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
        super(editMode, resourceBundleHelper);
        ColumnLayoutData layoutData = new ColumnLayoutData();
        layoutData.setAlignment(Alignment.ALIGN_CENTER);
        table = new NavigatorTable(getTableConfig());
        table.setLayoutData(layoutData);
        add(table);
    }

    protected Scenario getScenario() {
        return scenario;
    }

    protected void setScenario(Scenario scenario) {
        this.scenario = scenario;
        if (scenario != null) {
            table.setModel(new NavigatorTableModel((Collection)
scenario.getUsingUseCases()));
        } else {
            table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
        }
    }

    private NavigatorTableConfig getTableConfig() {
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                UseCase usecase = (UseCase) model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {

```

```

buttonLabel = getResourceBundleHelper(getLocale()).getString(
    PROP_VIEW_USECASE_BUTTON_LABEL, "View");
} else {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_EDIT_USECASE_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor, usecase, usecase.getClass(), null,
    WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            UseCase useCase = (UseCase) model.getBackingObject(row);
            return useCase.getName();
        }
    });
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            UseCase useCase = (UseCase) model.getBackingObject(row);
            return useCase.getCreatedBy().getUsername();
        }
    });
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            UseCase useCase = (UseCase) model.getBackingObject(row);
            Date dateCreated = useCase.getDateCreated();
            String formattedDate = dateCreated != null ? dateCreated.format("yyyy-MM-dd") : null;
            return formattedDate;
        }
    });
);

private static class ScenarioUseCasesTableManipulator extends
AbstractComponentManipulator {

    protected ScenarioUseCasesTableManipulator() {
        super();
    }

    @Override
    public Object getModel(Component component) {
        return getValue(component, Scenario.class);
    }

    @Override
    public void setModel(Component component, Object valueModel) {
        setValue(component, valueModel);
    }

    @Override
    public void addListenerToDetectChangesToInput(EditMode editMode,
        Component component) {
        // nothing to do.
    }

    @Override
    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getComponent(component).getScenario());
    }

    @Override
    public void setValue(Component component, Object value) {
        getComponent(component).setScenario((Scenario) value);
    }

    private ScenarioUseCasesTable getComponent(Component component) {
        return (ScenarioUseCasesTable) component;
    }
}
}

```

screen.java

```
/*
 * $Id: Screen.java,v 1.2 2008/02/27 08:04:16 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.screen;

import nextapp.echo2.app.event.ActionListener;
import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;

/**
 * @author ron
 */
public interface Screen extends ActionListener {

    public void addActionListener(ActionListener actionListener);

    public void removeActionListener(ActionListener actionListener);

    public EventDispatcher getEventDispatcher();

    public UIFrameworkApp getApp();

    /**
     * This method gets called by the UIFrameworkApp setCurrentScreen()
     * when a
     * new screen replaces the original screen.<br/>NOTE: This method
     * should
     * not be called manually.
     */
    public void dispose();

    /**
     * This method gets called by the UIFrameworkApp setCurrentScreen()
     * when a
     * new screen is set as the current screen.<br/>NOTE: This method
     * should
     * not be called manually.
     */
    public void setup();
}
```

selectentityevent.java

```
/*
 * $Id: SelectEntityEvent.java,v 1.3 2008/09/12 00:15:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation.event;

import edu.harvard.fas.rregan.uiframework.panel.Panel;

/**
 * An event fired by a selection panel to indicate the object selected
 * and the
 * desired destination of the selected object if required.
 *
 * @author ron
 */
public class SelectEntityEvent extends ClosePanelEvent {
    static final long serialVersionUID = 0;

    private final Object selectedObject;

    /**
     * @param source - the panel firing the event, which should also be
     * closed after firing.
     * @param selectedObject - the object that was selected.
     * @param destinationObject - the orginal object that requested the
     * selection of an object,
     *      this will be used to dispatch the event to the appropriate
     * requestor in case there
     *      are other listeners for selection of other objects. If this
     * is null all listeners
     *      for select events will recieve the event.
     */
    public SelectEntityEvent(Panel source, Object selectedObject, Object
destinationObject) {
        this(source, source, selectedObject, destinationObject);
    }

    /**
     * Create a select event when the source is not the panel to be
     * closed, possibly a controller or button on the panel.
     * @param source - the originator of the event
     * @param panelToClose - the panel to close after the event is fired.
     * @param selectedObject
     * @param destinationObject
     */
}
```

```

*/
public SelectEntityEvent(Object source, Panel panelToClose, Object
selectedObject,
    Object destinationObject) {
    this(source, SelectEntityEvent.class.getName(), panelToClose,
selectedObject,
        destinationObject);
}

protected SelectEntityEvent(Object source, String command, Panel
panelToClose,
    Object selectedObject, Object destinationObject) {
    super(source, command, panelToClose, destinationObject);
    this.selectedObject = selectedObject;
}

public Object getObject() {
    return selectedObject;
}
}

```

selectorbutton.java

```

/*
 * $Id: SelectorButton.java,v 1.10 2008/10/11 08:22:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation;

import java.util.HashSet;
import java.util.Set;

import nextapp.echo2.app.Column;
import nextapp.echo2.app.Extent;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Label;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

```

```

import
edu.harvard.fas.rregan.uiframework.AbstractAppAwareActionListener;
import edu.harvard.fas.rregan.uiframework.UIFrameworkApp;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.SelectorB
uttonManipulator;

/**
 * @author ron
 */
public class SelectorButton extends Column {
    private static final Logger log =
Logger.getLogger(SelectorButton.class);
    static final long serialVersionUID = 0;

    static {
        ComponentManipulators.setManipulator(SelectorButton.class, new
SelectorButtonManipulator());
    }

    private final Set<ChangeListener> changeListeners = new
HashSet<ChangeListener>();

    private Label selectedItemIndicator;
    private final SelectionIndicatorLabelAdapter
selectionIndicatorLabelAdapter;
    private NavigatorButton openSelectorButton;
    private ActionListener selectionListener;
    private final Class<?> selectionType;
    private Object selectedObject;
    private final String labelForNoSelectedItem = "<nothing selected>";

    /**
     * @param selectionIndicatorLabelAdapter
     * @param selectionType
     */
    public SelectorButton(SelectionIndicatorLabelAdapter
selectionIndicatorLabelAdapter,
        Class<?> selectionType) {

```

```

super();
this.selectionIndicatorLabelAdapter =
selectionIndicatorLabelAdapter;
this.selectionType = selectionType;
createComponents();
setSelectedObject(null);
setSelectionCriteria(null, null, null);
}

/**
 * @param selectionIndicatorLabelAdapter
 * @param selectionType
 * @param selectedObject
 */
public SelectorButton(SelectionIndicatorLabelAdapter
selectionIndicatorLabelAdapter,
Class<?> selectionType, Object selectedObject) {
super();
this.selectionIndicatorLabelAdapter =
selectionIndicatorLabelAdapter;
this.selectionType = selectionType;
createComponents();
setSelectedObject(selectedObject);
setSelectionCriteria(null, null, null);
}

/**
 * @param selectionIndicatorLabelAdapter
 * @param selectionType
 * @param selectedObject
 * @param selectionCriteriaType
 * @param selectionCriteria
 * @param panelName
 */
public SelectorButton(SelectionIndicatorLabelAdapter
selectionIndicatorLabelAdapter,
Class<?> selectionType, Object selectedObject, Class<?>
selectionCriteriaType,
Object selectionCriteria, String panelName) {
super();
this.selectionIndicatorLabelAdapter =
selectionIndicatorLabelAdapter;
this.selectionType = selectionType;
createComponents();
setSelectedObject(selectedObject);
setSelectionCriteria(selectionCriteriaType, selectionCriteria,
panelName);
}

}

/**
 * @return
 */
public Object getSelectedObject() {
return selectedObject;
}

/**
 * @param selectedObject
 */
public void setSelectedObject(Object selectedObject) {
this.selectedObject = selectedObject;
if (selectedObject != null) {
selectedItemIndicator.setText(selectionIndicatorLabelAdapter
.getLabelString(selectedObject));
} else {
selectedItemIndicator.setText((labelForNoSelectedItem == null ? ""
: labelForNoSelectedItem));
}
fireChangeEvent();
}

/**
 * @param selectionCriteriaType
 * @param selectionCriteria
 * @param panelName
 */
public void setSelectionCriteria(Class<?> selectionCriteriaType,
Object selectionCriteria,
String panelName) {
OpenPanelEvent opeSelectorEvent;

// the "source" of the event will be set as the destination of the
// SelectEntityEvent
// fired by the selector panel. The SelectObjectListener created in
// createComponents()
// will catch the select entity event and set the selected object on
the
// button.
if (selectionCriteria != null) {
opeSelectorEvent = new OpenPanelEvent(this,
PanelActionType.Selector,
selectionCriteria, selectionCriteriaType, panelName,
WorkflowDisposition.ContinueFlow);
} else {
}
}

```

```

    opeSelectorEvent = new OpenPanelEvent(this,
PanelActionType.Selector, null,
    selectionType, null, WorkflowDisposition.ContinueFlow);
}
setOpenSelectorEvent(opeSelectorEvent);
}

public void addChangeListener(ChangeListener l) {
    changeListeners.add(l);
}

public void removeChangeListener(ChangeListener l) {
    changeListeners.remove(l);
}

@Override
public void dispose() {
    changeListeners.clear();
    if (selectionListener != null) {
        getApp().getEventDispatcher().removeEventTypeActionListener(SelectE
ntityEvent.class,
            selectionListener, this);
        selectionListener = null;
    }
    super.dispose();
}

protected void fireChangeEvent() {
    if (!changeListeners.isEmpty()) {
        ChangeEvent e = new ChangeEvent(this);
        for (ChangeListener l : changeListeners) {
            l.stateChanged(e);
        }
    }
}

private void createComponents() {
    RowLayoutData rld = new RowLayoutData();
    rld.setInsets(new Insets(new Extent(0), new Extent(0), new
Extent(5), new Extent(0)));
    selectedIndicator = new Label("");
    selectedIndicator.setStyleName("Default");
    selectedIndicator.setLayoutData(rld);

    openSelectorButton = new NavigatorButton("Select",
getApp().getEventDispatcher());
    openSelectorButton.setStyleName("Default");
}

```

```

Row row = new Row();
row.add(selectedIndicator);
row.add(openSelectorButton);
add(row);
// The SelectObjectListener will catch the select entity event and
set
// the selected
// object on the button. The listener is registered explicitly for
// events with the
// destination of this button
selectionListener =
getApp().getEventDispatcher().addEventTypeActionListener(
    SelectEntityEvent.class, new SelectObjectListener(selectionType,
this, getApp()),
    this);
}

private void setOpenSelectorEvent(OpenPanelEvent eventToFire) {
    openSelectorButton.setEventToFire(eventToFire);
}

private UIFrameworkApp getApp() {
    return UIFrameworkApp.getApp();
}

/**
 * An interface that needs to be implemented to return a string that
 * describes the selected object attached to the button.
 *
 * @author ron
 */
public static interface SelectionIndicatorLabelAdapter {
    /**
     * @param selectedObject
     * @return a string describing the supplied selected object to a
user.
     */
    public String getLabelString(Object selectedObject);
}

/**
 * This listens for a select object event from a selection panel and
sets
 * the selected object on the button.
 *
 * @author ron
 */

```

```
private static class SelectObjectListener extends
AbstractAppAwareActionListener {
    static final long serialVersionUID = 0;

    private final Class<?> selectionType;
    private final SelectorButton button;

    public SelectObjectListener(Class<?> selectionType, SelectorButton
button,
        UIFrameworkApp app) {
        super(app);
        this.selectionType = selectionType;
        this.button = button;
    }

    public void actionPerformed(ActionEvent e) {
        if (e instanceof SelectEntityEvent) {
            SelectEntityEvent event = (SelectEntityEvent) e;
            if (this.button.equals(event.getDestinationObject()) &&
(event.getObject() != null)) {
                if (selectionType.isAssignableFrom(event.getObject().getClass()))
{
                    button.setSelectedObject(event.getObject());
                }
            }
        }
    }
}
```

selectorbuttonmanipulator.java

```
/*
 * $Id: SelectorButtonManipulator.java,v 1.8 2008/10/13 22:58:58
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.manipulators;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.ChangeEvent;
import nextapp.echo2.app.event.ChangeListener;
import edu.harvard.fas.rregan.uiframework.navigation.SelectorButton;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**

```

```
* A generic interface for manipulating a SelectorButton control.  
*  
* @author ron  
*/  
public class SelectorButtonManipulator extends AbstractComponentManipulator {  
  
    public void addListenerToDetectChangesToInput(final EditMode editMode, Component component) {  
        getComponent(component).addChangeListener(new ChangeListener() {  
            static final long serialVersionUID = 0L;  
  
            public void stateChanged(ChangeEvent e) {  
                editMode.setStateEdited(true);  
            }  
        });  
    }  
  
    public <T> T getValue(Component component, Class<T> type) {  
        return type.cast(getComponent(component).getSelectedObject());  
    }  
  
    public void setValue(Component component, Object value) {  
        getComponent(component).setSelectedObject(value);  
    }  
  
    private SelectorButton getComponent(Component component) {  
        return (SelectorButton) component;  
    }  
}
```

selectortablepanel.java

```
/*
 * $Id: SelectorTablePanel.java,v 1.5 2008/09/11 09:47:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Collection;

import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
```

```

import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel;
1;

/**
 * @author ron
 */
public class SelectorTablePanel extends AbstractPanel implements
ActionListener {
    static final long serialVersionUID = 0L;

    private NavigatorTable table;
    private NavigatorTableConfig tableConfig;

    /**
     * @param tableConfig
     * @param supportedContentType
     * @param panelName
     */
    protected SelectorTablePanel(Class<?> supportedContentType, String
panelName) {
        this(SelectorTablePanel.class.getName(), supportedContentType,
panelName);
    }

    /**
     * @param resourceBundleName
     * @param supportedContentType
     * @param panelName
     */
    protected SelectorTablePanel(String resourceBundleName, Class<?>
supportedContentType,
        String panelName) {
        super(resourceBundleName, PanelActionType.Selector,
supportedContentType, panelName);
    }

    @Override
    public void setup() {
        super.setup();
        setTable(new NavigatorTable(getTableConfig()));
        add(getTable());
        setTableModel(getTargetObject());
        getTable().addSelectionListener(this);
    }

    @Override
    public void setTargetObject(Object targetObject) {
        super.setTargetObject(targetObject);
        if (getTable() != null) {
            setTableModel(targetObject);
        }
    }

    private void setTableModel(Object targetObject) {
        if (targetObject != null) {
            NavigatorTableModelAdapter adapter =
getTargetNavigatorTableModelAdapter();
            adapter.setTargetObject(targetObject);
            getTable().setModel(new
NavigatorTableModel(adapter.getCollection()));
        }
    }

    /**
     * This method should be overridden to return a collection when the
target
     * of the panel is not a collection.
     *
     * @return an adapter to get the collection of items to select from
from the
     * target object.
     */
    protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
        return new NavigatorTableModelAdapter() {
            private Collection<Object> targetObject;

            @Override
            public Collection<Object> getCollection() {
                return targetObject;
            }

            @Override
            public void setTargetObject(Object targetObject) {
                this.targetObject = (Collection) targetObject;
            }
        };
    }
}

```

```

};

}

@Override
public void dispose() {
    super.dispose();
    getTable().removeSelectionListener(this);
    setTable(null);
    setTableConfig(null);
}

@Override
public void setStyleName(String newValue) {
    super.setStyleName(newValue);
    getTable().setStyleName(newValue);
}

protected NavigatorTableConfig getTableConfig() {
    return tableConfig;
}

protected void setTableConfig(NavigatorTableConfig tableConfig) {
    this.tableConfig = tableConfig;
}

protected NavigatorTable getTable() {
    return table;
}

protected void setTable(NavigatorTable table) {
    this.table = table;
}

@Override
public void actionPerformed(ActionEvent e) {
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
    getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}
}

```

semanticrelatednesssenserelationinfo.java

```
/*
```

```

 * $Id: SemanticRelatednessSenseRelationInfo.java,v 1.1 2008/12/14
11:36:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * @author ron
 */
public class SemanticRelatednessSenseRelationInfo extends
AbstractSenseRelationInfo {

    protected SemanticRelatednessSenseRelationInfo(Sense sense1, Sense
sense2, double relatedness,
        String reason) {
        super(sense1, sense2, relatedness, reason);
    }

    @Override
    public String toString() {
        return "relatedness(" + getSense1() + ", " + getSense2() + ") -> " +
getRank() + " "
            + getReason() + "}";
    }
}

```

semanticrole.java

```

/*
 * $Id: SemanticRole.java,v 1.4 2009/02/10 10:26:04 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp;

/**
 * @author ron
 */
public enum SemanticRole {
    /**
     *
     */
    PARTICIPANT(),

```

```

/*
 *
 */
AGENT(PARTICIPANT),  

/**/  

 *  

 */
ACTOR(AGENT),  

/**/  

 *  

 */
ACTOR1(ACTOR),  

/**/  

 *  

 */
ACTOR2(ACTOR),  

/**/  

 *  

 */
EXPERIENCER(AGENT),  

/**/  

 *  

 */
CAUSE(AGENT),  

/**/  

 *  

 */
PATIENT(PARTICIPANT),  

/**/  

 *  

 */
PATIENT1(PATIENT),  

/**/  

 *  

 */
PATIENT2(PATIENT),  

/**/  

 *  

 */
BENEFICIARY(PATIENT),  

/**/  

 *  

 */
PRODUCT(PATIENT),  

/**/  

 *  

 */
RECIPIENT(PATIENT),  

/**/  

 *  

 */
THEME(PARTICIPANT),  

/**/  

 *  

 */
THEME1(THEME),  

/**/  

 *  

 */
THEME2(THEME),  

/**/  

 *  

 */
INSTRUMENT(THEME),  

/**/  

 *  

 */
MATERIAL(THEME),  

/**/  

 *  

 */
STIMULUS(THEME),  

/**/  

 *  

 */

```

```

LOCATION(THEME),
/**
 *
 */
DESTINATION(LOCATION),
/**
 *
 */
SOURCE(LOCATION),
/**
 *
 */
ASSET(THEME),
/**
 *
 */
ATTRIBUTE(),
/**
 *
 */
PREDICATE(ATTRIBUTE),
/**
 *
 */
VALUE(),
/**
 *
 */
EXTENT(VALUE),
/**
 *
 */
TIME(VALUE),
/**
 * TODO: should this be a theme?
 */
TOPIC(),
/***
 *
 */
PROPOSITION(TOPIC),
/**
 * I don't know what this represents in VerbNet
 */
OBLIQUE(),
/**
 * The primary verb or a sentence or clause.
 */
VERB();

private final SemanticRole parent;

private SemanticRole(SemanticRole parent) {
    this.parent = parent;
}

private SemanticRole() {
    this.parent = null;
}

/**
 * @return the super type of semantic role or null if this is a base
 * role.
 */
public SemanticRole getParent() {
    return parent;
}

/**
 * @param role
 * @return true if this role is a descendant of, or is the supplied
 * role.
 */
public boolean isA(SemanticRole role) {
    SemanticRole thisOrAncestor = this;
    do {
        if (thisOrAncestor.equals(role)) {
            return true;
        }
        thisOrAncestor = thisOrAncestor.getParent();
    } while (thisOrAncestor != null);
    return false;
}

```

```
}
```

semanticrolecollector.java

```
/*
 * $Id: SemanticRoleCollector.java,v 1.1 2009/02/08 13:25:11 rregan
 * Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.impl.AbstractNLPTextWalker;
import edu.harvard.fas.rregan.nlp.impl.NLPTextWalkerFunction;

/**
 * @author ron
 */
public class SemanticRoleCollector extends
    AbstractNLPTextWalker<Map<SemanticRole, NLPText>,
Collection<NLPText>> {

    private NLPText root;

    /**
     * @param function
     */
    public
    SemanticRoleCollector(NLPTextWalkerFunction<Collection<NLPText>>
function) {
        super(function);
    }

    @Override
    public Map<SemanticRole, NLPText> process(NLPText text) {
        root = text;
        return super.process(text);
    }

    @Override
```

```
        public Map<SemanticRole, NLPText>
        transformResults(Collection<NLPText> result) {
            Map<SemanticRole, NLPText> map = new HashMap<SemanticRole,
NLPText>();
            for (NLPText roleFiller : result) {
                map.put(roleFiller.getSemanticRole(root.getPrimaryVerb()),
roleFiller);
            }
            return map;
        }
    }
```

semanticrolecollectorfunction.java

```
/*
 * $Id: SemanticRoleCollectorFunction.java,v 1.2 2009/02/09 10:12:29
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.Collection;
import java.util.HashSet;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.impl.NLPTextWalkerFunction;

/**
 * An NLPTextWalkerFunction that takes an NLPText and returns a
 * Collection of
 * the NLPTexts that fill SemanticRoles of the primary verb.
 *
 * @see {@link SemanticRole}
 * @author ron
 */
public class SemanticRoleCollectorFunction implements
NLPTextWalkerFunction<Collection<NLPText>> {

    private final Collection<NLPText> roleFillers;
    private final NLPText verb;

    /**
     * Create a tree printer with an initial buffer size of 1000
     * characters and
```

```

 * pretty printing on that adds tabs, spaces, and new lines to make
the tree
 * easier to read.
 *
 * @param verb -
 *          the verb to get the semantic role for.
 */
public SemanticRoleCollectorFunction(NLPText verb) {
    this(verb, new HashSet<NLPText>());
}

/**
 * Create a tree printer with the specified initial buffer size and
pretty
 * printing on that adds tabs, spaces, and new lines to make the tree
easier
 * to read.
 *
 * @param verb -
 *          the verb to get the semantic role for.
 * @param bufferSize -
 *          the initial size of the buffer for building the tree.
 */
public SemanticRoleCollectorFunction(NLPText verb,
Collection<NLPText> roleFillers) {
    this.verb = verb;
    this.roleFillers = roleFillers;
}

@Override
public void init() {
}

@Override
public void begin(NLPText text) {
    if (text.getSemanticRole(verb) != null) {
        roleFillers.add(text);
    }
}

@Override
public Collection<NLPText> end(NLPText t) {
    return roleFillers;
}
}

```

semanticrolelabeler.java

```

/*
 * $Id: SemanticRoleLabeler.java,v 1.8 2009/02/11 09:02:54 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalRelation;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.VerbalNetFrame;
import edu.harvard.fas.rregan.nlp.dictionary.VerbalNetFrameRef;

/**
 * Identify the child elements of a sentence that fill specific
 * thematic/semantic roles.<br>
 * <br>
 * Roles:<br>
 * Verb: The action.<br>
 * Agent: The causer of the action.<br>
 * Patient: The target/recipient of the action.<br>
 * Theme: The thing being acted upon.<br>
 * <br>
 * Example:<br>
 * I gave Bob fifty dollars.<br>
 * Verb: gave<br>
 * Agent: I<br>
 * Patient: Bob<br>
 * Theme: fifty dollars.<br>
 *

```

```

* @author ron
*/
@Component("semanticRoleLabeler")
public class SemanticRoleLabeler implements NLPPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(SemanticRoleLabeler.class);
    private static final Map<String, SyntaxMatchingContext>
compiledContexts = new HashMap<String, SyntaxMatchingContext>();
    private final DictionaryRepository dictionaryRepository;
    private final VerbNetFrameSyntaxParser vnFrameSyntaxParser;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public SemanticRoleLabeler(DictionaryRepository dictionaryRepository,
        VerbNetFrameSyntaxParser vnFrameSyntaxParser) {
        this.dictionaryRepository = dictionaryRepository;
        this.vnFrameSyntaxParser = vnFrameSyntaxParser;
    }

    /**
     * @param text
     */
    @Override
    public NLPText process(NLPText text) {
        if (text.in(GrammaticalStructureLevel.PARAGRAPH)) {
            for (NLPText sentence : text.getChildren()) {
                process(sentence);
            }
        } else {
            if (text.getPrimaryVerb() != null) {
                NLPText primaryVerb = text.getPrimaryVerb();
                boolean rolesAssigned = false;
                if ((primaryVerb.getDictionaryWordSense() != null)
                    &&
                (primaryVerb.getDictionaryWordSense().getVerbNetFrames().size() > 0))
                {
                    rolesAssigned = assignRolesByVerbNetFrames(text, primaryVerb);
                }
                if (!rolesAssigned) {
                    assignRolesBySyntaxDependencies(text, primaryVerb);
                }
            }
        }
        return text;
    }
}

```

```

protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}

protected GrammaticalRelation findPrimarySubjectRelation(NLPText
text) {
    for (GrammaticalRelation relation : text.getGrammaticalRelations())
{
    GrammaticalRelationType relType = relation.getType();
    if (relType.isA(GrammaticalRelationType.NOMINAL SUBJECT)
        || relType.isA(GrammaticalRelationType.AGENT)
        || relType.isA(GrammaticalRelationType.NOMINAL PASSIVE SUBJECT))
{
        return relation;
    }
}
throw SemanticRoleLabelerException.nominalSubjectNotFound(text);

protected synchronized SyntaxMatchingContext
getSyntaxMatchingContext(VerbNetFrameRef frameRef) {
    VerbNetFrame frame = frameRef.getFrame();
    SyntaxMatchingContext context =
compiledContexts.get(frame.getSyntax());
    if (context == null) {
        context = new SyntaxMatchingContext(frameRef, vnFrameSyntaxParser
            .parseVerbNetFrameSyntax(frameRef));
        compiledContexts.put(frame.getSyntax(), context);
    }
    return context;
}

protected boolean assignRolesByVerbNetFrames(NLPText sentence,
NLPText primaryVerb) {
    for (VerbNetFrameRef frameRef :
primaryVerb.getDictionaryWordSense().getVerbNetFrameRefs()) {
        SyntaxMatchingContext context = getSyntaxMatchingContext(frameRef);
        if (context.matches(getDictionaryRepository(), sentence,
primaryVerb)) {
            log.debug("matched verbnet frameRef " + frameRef.getId() + "
syntax: "
+ frameRef.getFrame().getSyntax());
            return true;
        }
    }
    return false;
}

```

```

}

private void assignRolesBySyntaxDependencies(NLPText sentence,
NLPText primaryVerb) {
    boolean passive = false;
    NLPText subject = null;
    NLPText directObject = null;
    NLPText indirectObject = null;
    NLPText prepObject = null;

    for (GrammaticalRelation relation :
sentence.getGrammaticalRelations()) {
        log.debug(relation);
        GrammaticalRelationType relType = relation.getType();
        if (relType.isA(GrammaticalRelationType.SUBJECT)) {
            if (relType.isA(GrammaticalRelationType.NOMINAL_PASSIVE SUBJECT))
{
                passive = true;
            }
            subject = relation.getDependent();
            if ((subject.getParent() != null) &&
subject.getParent().is(ParseTag.NP)) {
                subject = subject.getParent();
            }
        }
        if (relType.isA(GrammaticalRelationType.DIRECT_OBJECT)) {
            if ((sentence.getPrimaryVerb() == null)
|| sentence.getPrimaryVerb().equals(relation.getGovernor())) {
                directObject = relation.getDependent();
                if ((directObject.getParent() != null)
&& directObject.getParent().is(ParseTag.NP)) {
                    directObject = directObject.getParent();
                }
            }
        } else if (relType.isA(GrammaticalRelationType.INDIRECT_OBJECT)) {
            if ((sentence.getPrimaryVerb() == null)
|| sentence.getPrimaryVerb().equals(relation.getGovernor())) {
                indirectObject = relation.getDependent();
                if ((indirectObject.getParent() != null)
&& indirectObject.getParent().is(ParseTag.NP)) {
                    indirectObject = indirectObject.getParent();
                }
            }
        } else if
(relType.isA(GrammaticalRelationType.PREPOSITIONAL_OBJECT)) {
            // use the object of a prepositional phrase if it modifies
            // the primary verb
        }
    }
}

```

```

Set<GrammaticalRelation> prepRels =
relation.getGovernor().getDependentOfType(
    GrammaticalRelationType.PREPOSITIONAL_MODIFIER);
if (prepRels.size() == 1) {
    NLPText verbPred = prepRels.iterator().next().getGovernor();
    if ((sentence.getPrimaryVerb() == null)
|| sentence.getPrimaryVerb().equals(verbPred)) {
        // TODO: Should the object be the whole preposition?
        prepObject = relation.getDependent();
        if ((prepObject.getParent() != null)
&& prepObject.getParent().is(ParseTag.NP)) {
            prepObject = prepObject.getParent();
        }
    }
}
if (passive) {
    if ((subject != null) && (directObject != null)) {
        // if the verb is preceded by "was" or is in perfect
        subject.setSemanticRole(primaryVerb, SemanticRole.PATIENT);
        directObject.setSemanticRole(primaryVerb, SemanticRole.AGENT);
    } else {
        if (subject != null) {
            subject.setSemanticRole(primaryVerb, SemanticRole.AGENT);
        }
        if (directObject != null) {
            directObject.setSemanticRole(primaryVerb, SemanticRole.PATIENT);
        }
    } else {
        if (subject != null) {
            subject.setSemanticRole(primaryVerb, SemanticRole.AGENT);
        }
        if (directObject != null) {
            directObject.setSemanticRole(primaryVerb, SemanticRole.PATIENT);
        }
    }
    if (indirectObject != null) {
        indirectObject.setSemanticRole(primaryVerb, SemanticRole.PATIENT);
    }
    if (prepObject != null) {
        prepObject.setSemanticRole(primaryVerb, SemanticRole.PARTICIPANT);
    }
}
}

```

semanticrolelabelerexception.java

```
/*
 * $Id: SemanticRoleLabelerException.java,v 1.5 2009/02/10 10:26:03
rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.List;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetFrameRef;
import edu.harvard.fas.rregan.nlp.impl.NLPProcessorException;

/**
 * @author ron
 */
public class SemanticRoleLabelerException extends
NLPProcessorException {
    static final long serialVersionUID = 0;

    protected static final String MSG_FORMAT_MATCHED_FAILED = "Frame
match failed at rule '%s' with word '%s'";
    protected static final String MSG_FORMAT_MATCHED_FAILED_SELRES =
"Frame match failed at rule '%s' with word '%s' by selectional
restriction.";
    protected static final String MSG_FORMAT_UNMATCHED_SENTENCE_ELEMENTS =
"Unmatched sentence elements '%s'";
    protected static final String MSG_FORMAT_UNMATCHED_RULES = "Unmatched
rules '%s'";
    protected static final String MSG_FORMAT_UNKNOWN_SEMANTIC_ROLE = "The
role '%s' in syntax '%s' for class '%s' is not known.";
    protected static final String MSG_FORMAT_SEMANTIC_ROLE_NOT_IN_CLASS =
"The role '%s' was not expected in syntax '%s' for class '%s'";
    protected static final String MSG_FORMAT_VERB_NOT_FOUND = "Could not
find primary verb in text '%s';

    /**
     * @param rule -
     *          the rule that wasn't matched.
     * @return a SemanticRoleLabelerException with a message that
matching
     *          failed on the supplied rule.
     */
}
```

```
    protected static SemanticRoleLabelerException
matchFailed(SyntaxMatchingRule rule, NLPText word) {
    return new SemanticRoleLabelerException(MSG_FORMAT_MATCHED_FAILED,
rule, word);
}

    protected static SemanticRoleLabelerException
matchFailedBySelectionalRestriction(
    SyntaxMatchingRule rule, NLPText word) {
    return new
SemanticRoleLabelerException(MSG_FORMAT_MATCHED_FAILED_SELRES, rule,
word);
}

    /**
     * @return a SemanticRoleLabelerException with a message that all the
rules
     *          were matched and there were unmatched sentence elements
     *          remaining.
     */
    protected static SemanticRoleLabelerException
unmatchedSentenceElementsRemaining(
    String remainSentenceText) {
    return new
SemanticRoleLabelerException(MSG_FORMAT_UNMATCHED_SENTENCE_ELEMENTS,
remainSentenceText);
}

    /**
     * @return a SemanticRoleLabelerException with a message that all the
rules
     *          were matched and there were unmatched sentence elements
     *          remaining.
     */
    protected static SemanticRoleLabelerException
unmatchedRulesRemaining(
    List<SyntaxMatchingRule> remaingRules) {
    return new SemanticRoleLabelerException(MSG_FORMAT_UNMATCHED_RULES,
remaingRules.toString());
}

    /**
     * @param semanticRole
     * @param frameRef
     * @return a SemanticRoleLabelerException with a message indicating
an
     *          unexpected semantic role was found.
     */
}
```

```

*/
protected static SemanticRoleLabelerException
unknownSemanticRole(String semanticRole,
    VerbNetFrameRef frameRef) {
    return new
SemanticRoleLabelerException(MSG_FORMAT_UNKNOWN_SEMANTIC_ROLE,
semanticRole,
    frameRef.getFrame().getSyntax(), frameRef.getClass());
}

/**
 * @param semanticRole
 * @param frameRef
 * @return a SemanticRoleLabelerException with a message indicating a
 *         semantic role was not expected in this class.
 */
protected static SemanticRoleLabelerException roleNotInClass(String
semanticRole,
    VerbNetFrameRef frameRef) {
    return new
SemanticRoleLabelerException(MSG_FORMAT_SEMANTIC_ROLE_NOT_IN_CLASS,
semanticRole, frameRef.getFrame().getSyntax(),
frameRef.getVnClass());
}

/**
 * @param text
 * @return
 */
protected static SemanticRoleLabelerException
nominalSubjectNotFound(NLPText text) {
    return new SemanticRoleLabelerException(MSG_FORMAT_VERB_NOT_FOUND,
text.getText());
}

/**
 * @param format
 * @param args
 */
protected SemanticRoleLabelerException(String format, Object... args)
{
    super(format, args);
}

/**
 * @param cause
 * @param format

```

```

        * @param args
    */
protected SemanticRoleLabelerException(Throwable cause, String
format, Object... args) {
    super(cause, format, args);
}

```

semanticroleprinter.java

```

/*
 * $Id: SemanticRolePrinter.java,v 1.3 2009/02/12 02:21:17 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl.srl;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.impl.NLPTextWalkerFunction;

/**
 * An NLPTextWalkerFunction that takes an NLPText and prints out the
semantic
 * roles.
 *
 * <pre>
 * </pre>
 *
 * <p>
 *
 * @see {@link SemanticRole}
 * @author ron
 */
public class SemanticRolePrinter implements
NLPTextWalkerFunction<StringBuilder> {
    private static final Logger log =
Logger.getLogger(SemanticRolePrinter.class);

    private final NLPText verb;
    private final int bufferSize;
    private StringBuilder sb;

```

```

public SemanticRolePrinter() {
    this(null);
}

/**
 * @param verb -
 *          the verb to get the semantic role for.
 */
public SemanticRolePrinter(NLPText verb) {
    this(verb, 1000);
}

/**
 * Create a tree printer with the specified initial buffer size and
 * pretty
 * printing on that adds tabs, spaces, and new lines to make the tree
 * easier
 * to read.
 *
 * @param verb -
 *          the verb to get the semantic role for.
 * @param bufferSize -
 *          the initial size of the buffer for building the tree.
 */
public SemanticRolePrinter(NLPText verb, int bufferSize) {
    this.verb = verb;
    this.bufferSize = bufferSize;
}

@Override
public void init() {
    sb = new StringBuilder(bufferSize);
}

@Override
public void begin(NLPText text) {
    if (verb != null) {
        appendSemanticRoleForVerb(verb, text);
    } else {
        for (NLPText verb : text.getSupportedVerbs()) {
            appendSemanticRoleForVerb(verb, text);
        }
    }
}

@Override
public StringBuilder end(NLPText t) {
}

```

```

    return sb;
}

private void appendSemanticRoleForVerb(NLPText verb, NLPText text) {
    SemanticRole thisSemanticRole = text.getSemanticRole(verb);
    if (thisSemanticRole != null) {
        sb.append(verb);
        sb.append("[");
        sb.append(thisSemanticRole);
        sb.append("]");
        sb.append(text);
        sb.append("\n");
    }
}

```

semanticsimilaritysenserelationinfo.java

```

/*
 * $Id: SemanticSimilaritySenseRelationInfo.java,v 1.1 2008/12/14
 * 11:36:15 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * @author ron
 */
public class SemanticSimilaritySenseRelationInfo extends
AbstractSenseRelationInfo {

    protected SemanticSimilaritySenseRelationInfo(Sense sense1, Sense
sense2, double similarity,
                                                String reason) {
        super(sense1, sense2, similarity, reason);
    }

    @Override
    public String toString() {
        return "similarity(" + getSense1() + ", " + getSense2() + ") -> " +
getRank() + " {" +
            + getReason() + "}";
    }
}

```

```
}
```

semcorfile.java

```
/*
 * $Id: SemcorFile.java,v 1.2 2009/01/03 10:24:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.IndexColumn;

import edu.harvard.fas.rregan.nlp.dictionary.Word.IdAdapter;

/**
 * @author ron
 */
@Entity
@Table(name = "semcor_file")
@XmlRootElement(name = "semcorFile")
public class SemcorFile implements Comparable<SemcorFile>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;

    // the Semcor folder: brown1, brown2, or brownv
    private String corpusSectionFolder;
```

```
// the Semcor tagfile filename in the tagfiles folder: br-a01, br-
j17, ...
private String corpusFileName;
private List<SemcorSentence> sentences = new
ArrayList<SemcorSentence>();

/**
 * @param corpusSectionFolder
 * @param corpusFileName
 */
public SemcorFile(String corpusSectionFolder, String corpusFileName)
{
    setCorpusSectionFolder(corpusSectionFolder);
    setCorpusFileName(corpusFileName);
}

protected SemcorFile() {
}

@Id
@Column(name = "id", unique = true, nullable = false)
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@Column(name = "section")
public String getCorpusSectionFolder() {
    return corpusSectionFolder;
}

protected void setCorpusSectionFolder(String corpusSectionFolder) {
    this.corpusSectionFolder = corpusSectionFolder;
}

@Column(name = "file")
public String getCorpusFileName() {
    return corpusFileName;
}
```

```

protected void setCorpusFileName(String corpusFileName) {
    this.corpusFileName = corpusFileName;
}

@OneToMany(mappedBy = "file", targetEntity = SemcorSentence.class,
fetch = FetchType.LAZY, cascade = CascadeType.ALL)
@IndexColumn(name = "snum", base = 1)
public List<SemcorSentence> getSentences() {
    return sentences;
}

protected void setSentences(List<SemcorSentence> sentences) {
    this.sentences = sentences;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + getCorpusSectionFolder().hashCode();
            result = prime * result + getCorpusFileName().hashCode();
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
    proxy
    if (!(obj instanceof SemcorFile)) {
        return false;
    }
    final SemcorFile other = (SemcorFile) obj;
}

```

```

if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}

if (getCorpusSectionFolder() == null) {
    if (other.getCorpusSectionFolder() != null) {
        return false;
    }
} else if (!
getCorpusSectionFolder().equals(other.getCorpusSectionFolder())) {
    return false;
}
if (getCorpusFileName() == null) {
    if (other.getCorpusFileName() != null) {
        return false;
    }
} else if (!getCorpusFileName().equals(other.getCorpusFileName())) {
    return false;
}
return true;
}

@Override
public int compareTo(SemcorFile o) {
    if (!getCorpusSectionFolder().equals(o.getCorpusSectionFolder())) {
        return
getCorpusSectionFolder().compareTo(o.getCorpusSectionFolder());
    }
    return getCorpusFileName().compareTo(o.getCorpusFileName());
}

```

semcorinitializer.java

```

/*
 * $Id: SemcorInitializer.java,v 1.3 2009/01/26 10:19:01 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import org.springframework.beans.factory.annotation.Autowired;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

```

```

import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.ImportSemcorFileCommand
import edu.harvard.fas.rregan.request.NoSuchEntityException;
import edu.mit.jsemcor.element.IContext;
import edu.mit.jsemcor.element.IContextID;
import edu.mit.jsemcor.main.IConcordance;
import edu.mit.jsemcor.main.IConcordanceSet;
import edu.mit.jsemcor.main.Semcor;

/**
 * Load the dictionary from sql if no words exist.
 *
 * @author ron
 */
// @Component("semcorInitializer")
// @Scope("prototype")
public class SemcorInitializer extends AbstractSystemInitializer {

    /**
     * The name of the property in the DictionaryInitializer.properties
     * file
     * that contains the path to the dictionary xml data file, if not
     * supplied
     * the default file is "resources/nlp/dictionary/dictionary.xml.gz"
     */
    public static final String PROP_SEMCOR_BASE_DICTIONARY =
"SemcorBaseDirectory";
    public static final String PROP_SEMCOR_BASE_DICTIONARY_DEFAULT =
"resources/nlp/semcor/";

    private final DictionaryRepository dictionaryRepository;
    private final CommandHandler commandHandler;
    private final DictionaryCommandFactory dictionaryCommandFactory;

    /**
     * @param dictionaryRepository
     * @param commandHandler
     * @param dictionaryCommandFactory
     */
    @Autowired
    public SemcorInitializer(DictionaryRepository dictionaryRepository,
        CommandHandler commandHandler, DictionaryCommandFactory
        dictionaryCommandFactory) {
        super(10);
        this.dictionaryRepository = dictionaryRepository;
        this.commandHandler = commandHandler;
        this.dictionaryCommandFactory = dictionaryCommandFactory;
    }

    @Override
    public void initialize() {
        try {
            // TODO: this disables the initializer and shouldn't be hard coded
            if (true) {
                return;
            }
            // scan through the semcor files and load them if missing. assume
            if
                // any sentence
                // exists from a file then all of them do.
                ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                    SemcorInitializer.class.getName());
                String semcorBaseDirectory = resourceBundleHelper.getString(
                    PROP_SEMCOR_BASE_DICTIONARY,
                    PROP_SEMCOR_BASE_DICTIONARY_DEFAULT);
                log
                    .info("loading semcor corpus from: "
                        + getClass().getClassLoader().getResource(semcorBaseDirectory)
                        .toExternalForm());

                IConcordanceSet semcor = new
Semcor(getClass().getClassLoader().getResource(
                    semcorBaseDirectory));
                semcor.open();

                // TODO: add a semcore file that holds all the sentences in the
                file
                // and only load files that don't exist in the database.
                for (IConcordance section : semcor.values()) {
                    for (IContextID id : section.getContextIDs()) {
                        IContext context = section.getContext(id);
                        try {
                            dictionaryRepository.findSemcorFile(section.getName(), context
                                .getFilename());
                        } catch (NoSuchEntityException e) {
                            try {
                                log.debug("loading " + section.getName() + " " +
context.getFilename());
                                ImportSemcorFileCommand command = dictionaryCommandFactory
                                    .newImportSemcorFileCommand();
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```
        command.setSection(section);
        command.setContext(context);
        commandHandler.execute(command);
    } catch (Exception ee) {
        log.error("failed to load semcor file " + section.getName() + "
"
        + context.getFilename(), ee);
    }
}
}
}
}
} catch (Exception e) {
log.error("failed to initialize semcor: " + e, e);
}
}
}
```

semcorsentence.java

```
/*
 * $Id: SemcorSentence.java,v 1.2 2009/01/03 10:24:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.IndexColumn;
```

```
import edu.harvard.fas.rregan.nlp.dictionary.Word.IdAdapter;

/**
 * A sentence of the Semcor Corpus.
 *
 * @author ron
 */
@Entity
@Table(name = "semcor_sentence")
@XmlRootElement(name = "semcorSentence")
public class SemcorSentence implements Comparable<SemcorSentence>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;

    private SemcorFile file;
    // the snum from the s element wrapping the sentence in the file.
    private Long sentenceIndex;

    private List<SemcorSentenceWord> words = new
ArrayList<SemcorSentenceWord>();

    /**
     * Create a new sentence. This is used by the semcor db loader.
     *
     * @param corpusSectionFolder -
     *          the Semcor folder: brown1, brown2, or brownv
     * @param corpusFileName -
     *          the Semcor tagfile filename in the tagfiles folder
under the
     *          section folder. for example: br-a01, br-j17, ...
     * @param sentenceIndex -
     *          the sentence index from Semcor based on the value of
snum
     */
    public SemcorSentence(SemcorFile file, Long sentenceIndex) {
        setFile(file);
        setSentenceIndex(sentenceIndex);
    }

    protected SemcorSentence() {
    }

    @Id
    @Column(name = "id", unique = true, nullable = false)
```

```

@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@ManyToOne(targetEntity = SemcorFile.class, cascade =
CascadeType.ALL, optional = false)
@JoinColumn(name = "file_id")
public SemcorFile getFile() {
    return file;
}

protected void setFile(SemcorFile file) {
    this.file = file;
}

@Column(name = "snum")
public Long getSentenceIndex() {
    return sentenceIndex;
}

protected void setSentenceIndex(Long sentenceIndex) {
    this.sentenceIndex = sentenceIndex;
}

/**
 * @return The list of words in the sentence.
 */
@OneToMany(mappedBy = "sentence", targetEntity =
SemcorSentenceWord.class, fetch = FetchType.LAZY, cascade =
CascadeType.ALL)
@IndexColumn(name = "word_index", base = 1)
public List<SemcorSentenceWord> getWords() {
    return words;
}

protected void setWords(List<SemcorSentenceWord> words) {
    this.words = words;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + getFile().hashCode();
            result = prime * result + getSentenceIndex().hashCode();
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
    proxy
    if (!(obj instanceof SemcorSentence)) {
        return false;
    }
    final SemcorSentence other = (SemcorSentence) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }

    if (getFile() == null) {
        if (other.getFile() != null) {
            return false;
        }
    } else if (!getFile().equals(other.getFile())) {
        return false;
    }
    if (getSentenceIndex() == null) {
        if (other.getSentenceIndex() != null) {
            return false;
        }
    }
}

```

```

} else if (!getSentenceIndex().equals(other.getSentenceIndex())) {
    return false;
}
return true;
}

@Override
public int compareTo(SemcorSentence o) {
    if (!getFile().equals(o.getFile())) {
        return getFile().compareTo(o.getFile());
    }
    return getSentenceIndex().compareTo(o.getSentenceIndex());
}

@Override
public String toString() {
    return "" + getWords();
}
}

```

semcorsentenceword.java

```

/*
 * $Id: SemcorSentenceWord.java,v 1.2 2009/01/03 10:24:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlID;

```

```

import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;
import org.hibernate.annotations.Index;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.dictionary.Word.IdAdapter;

/**
 * A word in a sentence of the Semcor Corpus.
 *
 * @author ron
 */
@Entity
@Table(name = "semcor_sentence_word")
@org.hibernate.annotations.Table(appliesTo = "semcor_sentence_word",
indexes = {
    @Index(name = "index_ssw_word", columnNames = { "word_id" }),
    @Index(name = "index_ssw_synset", columnNames = { "sense_id" }) })
@XmlRootElement(name = "word")
public class SemcorSentenceWord implements Comparable<SemcorSentenceWord>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;

    private SemcorSentence sentence;
    private Integer index;
    private String text;
    private ParseTag parseTag;
    private Category properNounCategory;
    private Sense sense;

    /**
     * Create a word that doesn't have a sense, such as text tagged with
     DT, IN,
     * WDT, CC, ...
     *
     * @param sentence -
     *          the semcor sentence this word is part of.
     * @param index -
     *          the index of the word (zero based) in the semcor
     sentence.
     * @param text -
     *          the text of the word.
     * @param parseTag -
     *          the parse tag.
     */

```

```

/*
public SemcorSentenceWord(SemcorSentence sentence, Integer index,
String text, ParseTag parseTag) {
    setSentence(sentence);
    setIndex(index);
    setText(text);
    setParseTag(parseTag);
}

/**
 * Create a word with a sense.
 *
 * @param sentence -
 *          the semcor sentence this word is part of.
 * @param index -
 *          the index of the word (zero based) in the semcor
sentence.
 * @param text -
 *          the text of the word.
 * @param parseTag -
 *          the parse tag.
 * @param sense -
 *          the wordnet Sense.
 */
public SemcorSentenceWord(SemcorSentence sentence, Integer index,
String text,
    ParseTag parseTag, Sense sense) {
    setSentence(sentence);
    setIndex(index);
    setText(text);
    setParseTag(parseTag);
    setSense(sense);
}

/**
 * Create a word that is a named entity, for example
 * City_Executive_Committee, City_of_Atlanta, or
 * Superior_Court_Judge_Durwood_Pye
 *
 * @param sentence -
 *          the semcor sentence this word is part of.
 * @param index -
 *          the index of the word (zero based) in the semcor
sentence.
 * @param text -
 *          the text of the word.
 * @param parseTag -
 */

```

```

        the parse tag.
* @param sense -
*          the wordnet Sense.
* @param properNounCategory -
*          the wordnet category of an entity mapped from the pn
value in
*          Semcor such as group, location or person.
*/
public SemcorSentenceWord(SemcorSentence sentence, Integer index,
String text,
    ParseTag parseTag, Sense sense, Category properNounCategory) {
    setSentence(sentence);
    setIndex(index);
    setText(text);
    setParseTag(parseTag);
    setProperNounCategory(properNounCategory);
    setSense(sense);
}

protected SemcorSentenceWord() {
    // for hibernate
}

@Id
@Column(name = "id", unique = true, nullable = false)
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@ManyToOne(targetEntity = SemcorSentence.class, cascade =
CascadeType.ALL, optional = false)
@JoinColumn(name = "sentence_id")
public SemcorSentence getSentence() {
    return sentence;
}

protected void setSentence(SemcorSentence sentence) {
    this.sentence = sentence;
}

```

```

@Column(name = "word_index")
public Integer getIndex() {
    return index;
}

protected void setIndex(Integer index) {
    this.index = index;
}

@Column(name = "text")
public String getText() {
    return text;
}

protected void setText(String text) {
    this.text = text;
}

@Transient
public boolean isIgnored() {
    return getSense() == null;
}

@Enumerated(EnumType.STRING)
@Column(name = "parse_tag")
public ParseTag getParseTag() {
    return parseTag;
}

protected void setParseTag(ParseTag parseTag) {
    this.parseTag = parseTag;
}

@Transient
public boolean isNamedEntity() {
    return properNounCategory != null;
}

@ManyToOne(targetEntity = Category.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "category_id")
public Category getProperNounCategory() {
    return properNounCategory;
}

protected void setProperNounCategory(Category properNounCategory) {
    this.properNounCategory = properNounCategory;
}

@ManyToOne(targetEntity = Sense.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumns( { @JoinColumn(name = "sense_id"), @JoinColumn(name =
"word_id") })
public Sense getSense() {
    return sense;
}

protected void setSense(Sense sense) {
    this.sense = sense;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + getSentence().hashCode();
            result = prime * result + getIndex().hashCode();
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
proxy
    if (!(obj instanceof SemcorSentenceWord)) {
        return false;
    }
    final SemcorSentenceWord other = (SemcorSentenceWord) obj;
}

```

```

if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}

if (getSentence() == null) {
    if (other.getSentence() != null) {
        return false;
    }
} else if (!getSentence().equals(other.getSentence())) {
    return false;
}
if (getIndex() == null) {
    if (other.getIndex() != null) {
        return false;
    }
} else if (!getIndex().equals(other.getIndex())) {
    return false;
}
return true;
}

@Override
public int compareTo(SemcorSentenceWord o) {
    if (getSentence().equals(o.getSentence())) {
        getIndex().compareTo(o.getIndex());
    }
    return getSentence().compareTo(o.getSentence());
}

@Override
public String toString() {
    return getId() + ":" + getText();
}
}

```

semlinkref.java

```

/*
 * $Id: Semlinkref.java,v 1.2 2009/01/03 10:24:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.AttributeOverride;

```

```

import javax.persistence.AttributeOverrides;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

/**
 * @author ron
 */
@Entity
@Table(name = "semlinkref")
@XmlRootElement(name = "semlink")
public class Semlinkref implements Comparable<Semlinkref>, Serializable {
    static final long serialVersionUID = 0;

    private SemlinkrefId id;
    private Synset fromSynset;
    private Synset toSynset;
    private Linkdef linkType;

    public Semlinkref() {
    }

    public Semlinkref(SemlinkrefId id) {
        this.id = id;
    }

    @EmbeddedId
    @AttributeOverrides( {
        @AttributeOverride(name = "synset1id", column = @Column(name =
"synset1id", nullable = false)),
        @AttributeOverride(name = "synset2id", column = @Column(name =
"synset2id", nullable = false)),
        @AttributeOverride(name = "linkid", column = @Column(name =
"linkid", nullable = false)),
        @AttributeOverride(name = "distance", column = @Column(name =
"distance", nullable = false)) })
    public SemlinkrefId getId() {
        return this.id;
    }
}

```

```

public void setId(SemlinkrefId id) {
    this.id = id;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "synsetId", insertable = false, updatable =
false)
@XmlTransient
public Synset getFromSynset() {
    return fromSynset;
}

public void setFromSynset(Synset fromSynset) {
    this.fromSynset = fromSynset;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "synset2id", insertable = false, updatable =
false)
@XmlTransient
public Synset getToSynset() {
    return toSynset;
}

public void setToSynset(Synset toSynset) {
    this.toSynset = toSynset;
}

@ManyToOne(targetEntity = Linkdef.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "linkid", insertable = false, updatable = false)
@XmlTransient
public Linkdef getLinkType() {
    return linkType;
}

public void setLinkType(Linkdef linkType) {
    this.linkType = linkType;
}

@Transient
public int getDistance() {
    return getId().getDistance();
}

```

```

protected void setDistance(int distance) {
    getId().setDistance(distance);
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getFromSynset() == null) ? 0 :
getFromSynset().hashCode());
        result = prime * result + ((getToSynset() == null) ? 0 :
getToSynset().hashCode());
        result = prime * result + ((getLinkType() == null) ? 0 :
getLinkType().hashCode());
        result = prime * result + getDistance();
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Semlinkref other = (Semlinkref) obj;
    if ((getId() != null) && (other.getId() != null)) {
        return getId().equals(other.getId());
    }
    if ((getId() != null) && (other.getId() != null)) {
        return getId().equals(other.getId());
    }
}

```

```

if (getFromSynset() == null) {
    if (other.getFromSynset() != null) {
        return false;
    }
} else if (!getFromSynset().equals(other.getFromSynset())) {
    return false;
}

if (getLinkType() == null) {
    if (other.getLinkType() != null) {
        return false;
    }
} else if (!getLinkType().equals(other.getLinkType())) {
    return false;
}

if (getToSynset() == null) {
    if (other.getToSynset() != null) {
        return false;
    }
} else if (!getToSynset().equals(other.getToSynset())) {
    return false;
}

return true;
}

@Override
public int compareTo(Semlinkref o) {
    return getId().compareTo(o.getId());
}
}

```

semlinkrefid.java

```

/*
 * $Id: SemlinkrefId.java,v 1.2 2009/01/03 10:24:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Embeddable;

```

```

/**
 * @author ron
 */
@Embeddable
public class SemlinkrefId implements Comparable<SemlinkrefId>, Serializable {
    static final long serialVersionUID = 0;

    private Long synset1id;
    private Long synset2id;
    private Long linkid;
    private Integer distance;

    public SemlinkrefId() {
    }

    public SemlinkrefId(Long synset1id, Long synset2id, Long linkid, Integer distance) {
        this.synset1id = synset1id;
        this.synset2id = synset2id;
        this.linkid = linkid;
        this.distance = distance;
    }

    @Column(name = "synset1id", nullable = false)
    public Long getSynset1id() {
        return this.synset1id;
    }

    public void setSynset1id(Long synset1id) {
        this.synset1id = synset1id;
    }

    @Column(name = "synset2id", nullable = false)
    public Long getSynset2id() {
        return this.synset2id;
    }

    public void setSynset2id(Long synset2id) {
        this.synset2id = synset2id;
    }

    @Column(name = "linkid", nullable = false)
    public Long getLinkid() {
        return this.linkid;
    }
}

```

```

public void setLinkid(Long linkid) {
    this.linkid = linkid;
}

@Column(name = "distance", nullable = false)
public Integer getDistance() {
    return distance;
}

protected void setDistance(Integer distance) {
    this.distance = distance;
}

@Override
public boolean equals(Object other) {
    if ((this == other)) {
        return true;
    }
    if ((other == null)) {
        return false;
    }
    if (!(other instanceof SemlinkrefId)) {
        return false;
    }
    SemlinkrefId castOther = (SemlinkrefId) other;

    return (this.getSynsetlid() == castOther.getSynsetlid())
        && (this.getSynset2id() == castOther.getSynset2id())
        && (this.getLinkid() == castOther.getLinkid())
        && (this.getDistance() == castOther.getDistance());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        final int prime = 31;
        int result = 1;
        result = prime * result + getSynsetlid().hashCode();
        result = prime * result + getSynset2id().hashCode();
        result = prime * result + getLinkid().hashCode();
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

```

```

@Override
public int compareTo(SemlinkrefId o) {
    if (this.getSynsetlid() != o.getSynsetlid()) {
        return (this.getSynsetlid().intValue() -
o.getSynsetlid().intValue());
    }
    if (this.getSynset2id() != o.getSynset2id()) {
        return (this.getSynset2id().intValue() -
o.getSynset2id().intValue());
    }
    if (this.getLinkid() != o.getLinkid()) {
        return (this.getLinkid().intValue() - o.getLinkid().intValue());
    }
    if (this.getDistance() != o.getDistance()) {
        return (this.getDistance() - o.getDistance());
    }
    return 0;
}

```

sense.java

```

/*
 * $Id: Sense.java,v 1.7 2009/02/08 13:25:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlIDREF;

```

```

import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;

import
edu.harvard.fas.rregan.nlp.dictionary.impl.command.ImportDictionaryCom
mandImpl;

/**
 * Wordnet Word Sense
 */
@Entity
@Table(name = "sense")
@XmlRootElement(name = "sense")
public class Sense implements Comparable<Sense>, Serializable {
    static final long serialVersionUID = 0;

    private SenseId id;
    private Integer rank;
    private Word word;
    private Synset synset;
    private Integer sampleFrequency;
    private String senseKey;
    private Set<VerbNetFrameRef> frameRefs = new
TreeSet<VerbNetFrameRef>();

    /**
     * @param word
     * @param synset
     */
    public Sense(Word word, Synset synset) {
        setId(new SenseId(synset.getId(), word.getId()));
        setWord(word);
        setSynset(synset);
    }

    protected Sense() {
    }

    @EmbeddedId
    @XmlTransient
    // this must be public for a query.
    public SenseId getId() {
        return this.id;
    }
}

```

```

protected void setId(SenseId id) {
    this.id = id;
}

@ManyToOne(targetEntity = Word.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "wordid", insertable = false, updatable = false)
@XmlTransient
public Word getWord() {
    return word;
}

protected void setWord(Word word) {
    this.word = word;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "synsetid", insertable = false, updatable = false)
@XmlIDREF
@XmlAttribute
public Synset getSynset() {
    return synset;
}

protected void setSynset(Synset synset) {
    this.synset = synset;
}

@OneToMany(targetEntity = VerbNetFrameRef.class, cascade =
{ CascadeType.ALL }, fetch = FetchType.LAZY, mappedBy = "sense")
@Sort(type = SortType.NATURAL)
@XmlTransient
public Set<VerbNetFrameRef> getVerbNetFrameRefs() {
    return frameRefs;
}

protected void setVerbNetFrameRefs(Set<VerbNetFrameRef> frameRefs) {
    this.frameRefs = frameRefs;
}

/**
 * @return The VerbNet frames applicable to this sense for supported
verbs,
     *         or an empty set.
 */

```

```

@Transient
@XmlTransient
public Set<VerbNetFrame> getVerbNetFrames() {
    Set<VerbNetFrame> frames = new
    HashSet<VerbNetFrame>(getVerbNetFrameRefs().size());
    for (VerbNetFrameRef ref : getVerbNetFrameRefs()) {
        frames.add(ref.getFrame());
    }
    return frames;
}

@Column(name = "rank", nullable = false)
@XmlAttribute(name = "rank")
public Integer getRank() {
    return this.rank;
}

public void setRank(Integer rank) {
    this.rank = rank;
}

@Column(name = "freq", nullable = true)
@XmlAttribute(name = "frequency")
public Integer getSampleFrequency() {
    return sampleFrequency;
}

public void setSampleFrequency(Integer sampleFrequency) {
    this.sampleFrequency = sampleFrequency;
}

@Column(name = "sense_key", nullable = true)
@XmlAttribute(name = "senseKey")
public String getSenseKey() {
    return senseKey;
}

public void setSenseKey(String senseKey) {
    this.senseKey = senseKey;
}

@Override
public int compareTo(Sense o) {
    if (getWord().equals(o.getWord())) {
        return (getSynset().getPos().compareTo(o.getSynset().getPos()) * 100 + getRank())
            .compareTo(o.getRank());
    }
    return getWord().compareTo(o.getWord());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getRank() == null) ? 0 : getRank().hashCode());
        result = prime * result + ((getSynset() == null) ? 0 : getSynset().hashCode());
        result = prime * result + ((getWord() == null) ? 0 : getWord().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final Sense other = (Sense) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getWord() == null) {
        if (other.getWord() != null) {
            return false;
        }
    } else if (!getWord().equals(other.getWord())) {
        return false;
    }
}

```

```

if (getSynset() == null) {
    if (other.getSynset() != null) {
        return false;
    }
} else if (!getSynset().equals(other.getSynset())) {
    return false;
}
if (getRank() == null) {
    if (other.getRank() != null) {
        return false;
    }
} else if (!getRank().equals(other.getRank())) {
    return false;
}
return true;
}

@Override
public String toString() {
    return getWord().getLemma() + "#" + getSynset().getPos() + "#" +
getRank();
}

/**
 * This is for JAXB to patchup the parent/child relationship and swap
the
 * creator with an existing user.
 *
 * @param parent
 * @see ImportDictionaryCommandImpl.UnmarshallerListener
 */
public void afterUnmarshal(Object parent) {
    setWord((Word) parent);
    setId(new SenseId(getSynset().getId(), getWord().getId()));
}
}

```

senseid.java

```

/*
 * $Id: SenseId.java,v 1.1 2008/12/13 00:40:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

```

```

import javax.persistence.Column;
import javax.persistence.Embeddable;

/**
 * Wordnet Sense composite id based on a Word and a Synset.
 */
@Embeddable
public class SenseId implements Serializable {
    static final long serialVersionUID = 0;

    private Long synsetid;
    private Long wordid;

    public SenseId() {
    }

    public SenseId(Long synsetid, Long wordid) {
        this.synsetid = synsetid;
        this.wordid = wordid;
    }

    @Column(name = "synsetid", nullable = false)
    public Long getSynsetid() {
        return this.synsetid;
    }

    public void setSynsetid(Long synsetid) {
        this.synsetid = synsetid;
    }

    @Column(name = "wordid", nullable = false)
    public Long getWordid() {
        return this.wordid;
    }

    public void setWordid(Long wordid) {
        this.wordid = wordid;
    }

    @Override
    public boolean equals(Object other) {
        if ((this == other)) {
            return true;
        }
        if ((other == null)) {
            return false;
        }
    }
}

```

```

}
if (!(other instanceof SenseId)) {
    return false;
}
SenseId castOther = (SenseId) other;

return (this.getSynsetid() == castOther.getSynsetid())
    && (this.getWordid() == castOther.getWordid());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getSynsetid() == null) ? 0 :
getSynsetid().hashCode());
        result = prime * result + ((getWordid() == null) ? 0 :
getWordid().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}
}

```

senserelationinfo.java

```

/*
 * $Id: SenseRelationInfo.java,v 1.1 2008/12/14 11:36:11 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import edu.harvard.fas.rregan.nlp.dictionary.Sense;

/**
 * @author ron
 */
public interface SenseRelationInfo {
    public Sense getSense1();

    public Sense getSense2();
}

```

```

    public double getRank();
    public String getReason();
}

```

sentencizer.java

```

/*
 * $Id: Sentencizer.java,v 1.5 2009/04/01 09:47:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.List;

import opennlp.tools.sentdetect.SentenceDetector;
import opennlp.tools.sentdetect.SentenceDetectorME;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.ApplicationException;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * Detects and separates sentences in an NLPText.
 *
 * @author ron
 */
@Component("sentencizer")
public class Sentencizer extends AbstractOpenNLPTool<NLPText> {
    private static final Logger log =
Logger.getLogger(Sentencizer.class);

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the open nlp sentence detector model file relative to the
     * classpath. By default the path is
     * "resources/nlp/opennlp-tools/EnglishSD.bin.gz"
     */
}

```

```

public static final String PROP_ENGLISH_SENTENCE_DETECTOR_MODEL_FILE
= "EnglishSentenceDetectorModelFile";

/**
 * The default sentence detector model file.
 */
public static final String
PROP_ENGLISH_SENTENCE_DETECTOR_MODEL_FILE_DEFAULT =
"resources/nlp/opennlp-tools/EnglishSD.bin.gz";

private static SentenceDetector sentenceDetector;

/***
 * create a new Sentencizer, initializing if needed.
 *
 * @throws ApplicationException
 *          if the underlying OpenNLP sentence detector fails to
 *          initialize
 */
public Sentencizer() {
    init();
}

private synchronized void init() {
    if (sentenceDetector == null) {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                Sentencizer.class.getName());

            String modelFile = resourceBundleHelper.getString(
                PROP_ENGLISH_SENTENCE_DETECTOR_MODEL_FILE,
                PROP_ENGLISH_SENTENCE_DETECTOR_MODEL_FILE_DEFAULT);

            sentenceDetector = new
SentenceDetectorME(readGISModel(modelFile));
        } catch (Exception e) {
            sentenceDetector = null;
            throw ApplicationException.failedToInitializeComponent(getClass(),
e);
        }
    }
}

@Override
public NLPText process(NLPText text) {
    NLPTextImpl workingText = (NLPTextImpl) text;

    if (workingText.hasText()
        && GrammaticalStructureLevel.UNKNOWN.equals(workingText
            .getGrammaticalStructureLevel())) {
        List<String> sentences = sentencize(workingText.getText());
        if (sentences.size() > 1) {
            workingText.setGrammaticalStructureLevel(GrammaticalStructureLevel
                .PARAGRAPH);
        }
        int startIndex = 0;
        for (String sentence : sentences) {
            workingText.getChildren().add(
                new NLPTextImpl(workingText, sentence,
                    GrammaticalStructureLevel.SENTENCE));
            startIndex += sentence.length();
        }
    } else {
        workingText.setGrammaticalStructureLevel(GrammaticalStructureLevel
            .SENTENCE);
    }
    return text;
}

private List<String> sentencize(String text) {
    log.debug("text = " + text);
    if (sentenceDetector != null) {
        int[] sentenceOffsets = sentenceDetector.sentPosDetect(text);
        List<String> trimmedSentences = new
ArrayList<String>(sentenceOffsets.length);

        if (sentenceOffsets.length == 0) {
            trimmedSentences.add(text);
        } else {
            // if leftover is true then there is dangling text after the
            // last sentence
            boolean leftover = sentenceOffsets[sentenceOffsets.length - 1] !=
text.length();
            trimmedSentences.add(text.substring(0,
                sentenceOffsets[0]).trim());

            for (int si = 1; si < sentenceOffsets.length; si++) {
                int nextStart = sentenceOffsets[si];
                while (Character.isWhitespace(text.charAt(nextStart - 1))) {
                    nextStart--;
                }
                trimmedSentences.add(text.substring(sentenceOffsets[si - 1],
                    nextStart));
            }
        }
    }
}

```

```

        }
        if (leftover) {
            trimmedSentences.add(text
                .substring(sentenceOffsets[sentenceOffsets.length -
1]).trim());
        }
        log.debug("sentences detected = " + trimmedSentences.size());
        return trimmedSentences;
    }
    return null;
}
}

```

setscreenevent.java

```

/*
 * $Id: SetScreenEvent.java,v 1.3 2008/03/01 02:30:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.navigation.event;

import org.apache.log4j.Logger;

/**
 * A base class for events that cause the UIFrameworkApp's screen to
change.
 * @author ron
 */
public class SetScreenEvent extends NavigationEvent {
    private static final Logger log =
Logger.getLogger(SetScreenEvent.class);
    static final long serialVersionUID = 0;

    private String screenToDisplay;

    /**
     * @param source
     * @param command
     * @param screenToDisplay
     */
    protected SetScreenEvent(Object source, String command, String
screenToDisplay) {
        super(source, command);
        setScreenToDisplay(screenToDisplay);
    }
}

```

```

        }

        /**
         * @return - the new Screen to display
         */
        public String getScreenToDisplay() {
            return screenToDisplay;
        }

        protected void setScreenToDisplay(String screenToDisplay) {
            this.screenToDisplay = screenToDisplay;
        }
    }
}

```

simplelemmatizer.java

```

/*
 * $Id: SimpleLemmatizer.java,v 1.2 2009/01/26 10:19:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl.lemmatizer;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.LemmatizerRule;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * Lemmatizer that uses a set of rules to find a matching lemma in the
* dictionary.<br>
* The lemmatizer rules are configured in the spring
lemmatizerConfig.xml file.
*
* @author ron
*/
@Component("lemmatizer")
public class SimpleLemmatizer implements NLPProcessor<NLPText> {

    private final Set<LemmatizerRule> rules;

    /**

```

```

 * @param rules -
 *          a set of rules for lemmatizing words.
 */
@.Autowired
public SimpleLemmatizer(Set<LemmatizerRule> rules) {
    this.rules = rules;
}

/**
 * @see
edu.harvard.fas.rregan.nlp.NLPProcessor#process(edu.harvard.fas.rregan
.nlp.NLPText)
 */
@Override
public NLPText process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.WORD)) {
        for (LemmatizerRule rule : rules) {
            String lemma = rule.lemmatize(text.getText(),
text.getPartOfSpeech());
            if (lemma != null) {
                text.setLemma(lemma);
                break;
            }
        }
    } else {
        for (NLPText word : text.getLeaves()) {
            process(word);
        }
    }
    return text;
}

```

simpleleskwsd.java

```

/*
 * $Id: SimpleLeskWSD.java,v 1.4 2009/01/26 10:19:05 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wsd;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;

```

```

import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPProcessorFactory;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentence;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentenceWord;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import edu.harvard.fas.rregan.nlp.impl.NLPTextImpl;

/**
 * A dictionary based word sense disambiguator that uses the
simplified lesk
 * algorithm.
 *
 * @author ron
 */
// @Component("SimpleLeskWSD")
public class SimpleLeskWSD implements NLPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(SimpleLeskWSD.class);

    private final DictionaryRepository dictionaryRepository;
    private final NLPProcessor<NLPText> sentencizer;
    private final NLPProcessor<NLPText> parser;
    private final NLPProcessor<NLPText> lemmatizer;
    private final NLPProcessor<NLPText> dictinizer;

    /**
     * @param dictionaryRepository
     * @param processorFactory
     */
    @Autowired
    public SimpleLeskWSD(DictionaryRepository dictionaryRepository,
NLPProcessorFactory processorFactory) {
        this.dictionaryRepository = dictionaryRepository;
        sentencizer = processorFactory.getSentencizer();
        parser = processorFactory.getParser();
        lemmatizer = processorFactory.getLemmatizer();
        dictinizer = processorFactory.getDictinizer();
    }

    @Override
    public NLPText process(NLPText text) {
        if (text.in(GrammaticalStructureLevel.PARAGRAPH)) {
            for (NLPText sentence : text.getChildren()) {
                process(sentence);
            }
        }
    }
}

```

```

    }
} else {
    dictionary.process(text);
for (NLPText word : text.getLeaves()) {
    Word dicWord = word.getDictionaryWord();
    if (dicWord != null) {
        if (dicWord.getSenses().size() == 1) {
            word.setDictionaryWordSense(dicWord.getSenses().iterator().next());
        } else if (dicWord.getSenses().size() > 1) {
            disambiguate(text, word);
        }
    }
}
return text;
}

/**
 * Simplified Lesk algorithm from Speech and Language Processing,
second
 * edition by Daniel Jurafsky and James Martin, page 646.
 *
 * @param text
 * @param word
 */
private void disambiguate(NLPText text, NLPText word) {
    Sense bestSense =
word.getDictionaryWord().getSense(word.getPartOfSpeech(), 1);
    int maxOverlap = 0;
    for (Sense sense :
word.getDictionaryWord().getSenses(word.getPartOfSpeech())) {
        // TODO: preprocess all the senses
        NLPText signature = buildSignature(sense);
        int overlap = computeOverlap(signature, text);
        if (overlap > maxOverlap) {
            maxOverlap = overlap;
            bestSense = sense;
        } else if (overlap == maxOverlap) {
            // for equal overlap use the more common form
            if (sense.getRank() < bestSense.getRank()) {
                bestSense = sense;
            }
        }
    }
    word.setDictionaryWordSense(bestSense);
}

```

```

private NLPText buildSignature(Sense sense) {
    NLPTextImpl definition = new
NLPTextImpl(sense.getSynset().getDefinition());
    sentencizer.process(definition);
    parser.process(definition);
    lemmatizer.process(definition);
    dictionary.process(definition);
    NLPTextImpl signature = new NLPTextImpl(null,
GrammaticalStructureLevel.BAGOFWORDS);
    for (NLPText word : definition.getLeaves()) {
        signature.getChildren().add(word);
    }
    // add all semcor sentence words to build up the signature
    for (SemcorSentence sentence :
dictionaryRepository.findSemcorSentences(sense)) {
        for (SemcorSentenceWord word : sentence.getWords()) {
            if (word.getSense() != null) {
                NLPTextImpl nlpWord = new NLPTextImpl(signature, word.getText(),
GrammaticalStructureLevel.WORD);
                signature.getChildren().add(nlpWord);
                nlpWord.setParseTag(word.getParseTag());
                nlpWord.setPartOfSpeech(word.getParseTag().getPartOfSpeech());
                Sense aSense = word.getSense();
                nlpWord.setDictionaryWordSense(aSense);
                nlpWord.setPartOfSpeech(aSense.getSynset().getPartOfSpeech());
                nlpWord.setDictionaryWord(aSense.getWord());
                nlpWord.setLemma(aSense.getWord().getLemma());
            }
        }
    }
    return signature;
}

private int computeOverlap(NLPText signature, NLPText context) {
    int overlap = 0;
    log.debug("signature: " + signature);
    log.debug("context: " + context);
    for (NLPText sigWord : signature.getLeaves()) {
        for (NLPText contextWord : context.getLeaves()) {
            if ((sigWord.getDictionaryWord() != null)
&&
sigWord.getDictionaryWord().equals(contextWord.getDictionaryWord())) {
                log.info("dictionary match: " + sigWord.getDictionaryWord() +
" " +
contextWord.getDictionaryWord());
                overlap++;
            }
        }
    }
}

```

```

        }
    log.debug("overlap: " + overlap);
    return overlap;
}
}

```

spellingchecker.java

```

/*
 * $Id: SpellingChecker.java,v 1.6 2009/01/28 06:01:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * Check if a word is spelled correctly
 *
 * @author ron
 */
@Component("spellingChecker")
public class SpellingChecker implements NLPPProcessor<Boolean> {
    private static final Logger log =
Logger.getLogger(SpellingChecker.class);

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public SpellingChecker(DictionaryRepository dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }
}

```

```

@Override
public Boolean process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.WORD) && text.hasText()) {
        if (!text.in(PartOfSpeech.PUNCTUATION, PartOfSpeech.NUMBER,
PartOfSpeech.SYMBOL)
        && !text.in(ParseTag.POS)) {
            // TODO: parser error let's -> let us -> (VB let) (POS 's)
            // TODO: handle contractions
            if (text.in(ParseTag.VBZ, ParseTag.PRPO) &&
text.getText().equalsIgnoreCase("s")) {
                // contractions like It's for it is and let's for let us
                // TODO: replace it with "is" or "us"?
            } else if (text.in(ParseTag.MD) &&
text.getText().equalsIgnoreCase("ll")) {
                // contractions like I'll for I will
                // TODO: replace it with "will"?
            } else if (text.in(ParseTag.VBP) &&
text.getText().equalsIgnoreCase("ve")) {
                // contractions like I've for I have
                // TODO: replace it with "have"?
            } else if (text.in(ParseTag.VBZ) &&
text.getText().equalsIgnoreCase("m")) {
                // contractions like I'm for I am
                // TODO: replace it with "am"?
            } else if (text.in(ParseTag.VBP) &&
text.getText().equalsIgnoreCase("re")) {
                // contractions like you're for you are
                // TODO: replace it with "are"?
            } else if (text.in(PartOfSpeech.ADVERB) &&
text.getText().equalsIgnoreCase("n't")) {
                // contractions like don't for do not
                // TODO: replace it with "not"?
                // TODO: what about the lead word like can't -> (MD ca) (RB
                // n't)
            } else {
                return dictionaryRepository.isKnownWord(text.getText());
            }
        } else {
            for (NLPText word : text.getLeaves()) {
                if (!process(word)) {
                    return Boolean.FALSE;
                }
            }
        }
    }
    return Boolean.TRUE;
}

```

```
}
```

spellingsuggester.java

```
/*
 * $Id: SpellingSuggester.java,v 1.5 2008/12/13 00:40:00 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.DictionaryRepository;

/**
 * Find spelling suggestions
 */
@Component("spellingSuggester")
public class SpellingSuggester implements NLPPProcessor<Collection<NLPText>> {
    private static final Logger log =
        Logger.getLogger(SpellingSuggester.class);

    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public SpellingSuggester(DictionaryRepository dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }
}
```

```
}
```

```
@Override
public Collection<NLPText> process(NLPText text) {
    if (text.is(GrammaticalStructureLevel.WORD) && text.hasText()
        && !text.is(PartOfSpeech.PUNCTUATION)) {
        List<NLPText> results = new ArrayList<NLPText>();
        for (String word :
            dictionaryRepository.findSpellingSuggestions(text.getText(), 2)) {
            results.add(new NLPTextImpl(word,
                GrammaticalStructureLevel.WORD));
        }
        return results;
    }
    return Collections.EMPTY_LIST;
}
```

stakeholder.java

```
/*
 * $Id: Stakeholder.java,v 1.10 2009/01/08 05:49:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public interface Stakeholder extends ProjectOrDomainEntity,
    GoalContainer, Comparable<Stakeholder> {

    /**
     * @return true if this is a stakeholder user (basically getUser() != null)
     */
    public boolean isUserStakeholder();

    /**
     * @return if the stakeholder is associated with a user, this will
     *         return that user, otherwise it will return null.
     */
}
```

```

*/
public User getUser();

/**
 * @return the team that this stakeholder is assigned for the
project.
 */
public ProjectTeam getTeam();

/**
 * @return The set of permissions that the stakeholder has for a
project.
 */
public Set<StakeholderPermission> getStakeholderPermissions();

/**
 * @param entityType
 * @param permissionType
 * @return true if the user has the specified permission type on the
 *         specified entity type.
 */
public boolean hasPermission(Class<?> entityType,
StakeholderPermissionType permissionType);

/**
 * @param stakeholderPermission
 * @return true if the user has the specified permission.
 */
public boolean hasPermission(StakeholderPermission
stakeholderPermission);

/**
 * Grant the stakeholder the specified permission.
 *
 * @param stakeholderPermission
 */
public void grantStakeholderPermission(StakeholderPermission
stakeholderPermission);

/**
 * Revoke the specified permission from the stakeholder.
 *
 * @param stakeholderPermission
 */
public void revokeStakeholderPermission(StakeholderPermission
stakeholderPermission);
}

```

stakeholdereditorpanel.java

```

/*
 * $Id: StakeholderEditorPanel.java,v 1.17 2009/03/05 08:50:46 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;
import java.util.Comparator;
import java.util.HashSet;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;

import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.ComboBox;
import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ProjectTeam;
import edu.harvard.fas.rregan.requel.project.ProjectUserRole;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.DeleteStakeholderCommand
;

```

```

import
edu.harvard.fas.rregan.requel.project.command.EditStakeholderCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.UserSet;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSet;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSetModel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedTextListModel;

/**
 * @author ron
 */
public class StakeholderEditorPanel extends
AbstractRequelProjectEditorPanel {
    private static final Logger log =
Logger.getLogger(StakeholderEditorPanel.class);

    static final long serialVersionUID = 0L;

    /**
     * The name to use in the StakeholderEditorPanel.properties file to
     * set the
     * label of the name field. If the property is undefined "Name" is
     * used.
     */
    public static final String PROP_LABEL_NAME = "Name.Label";

    /**
     * The name to use in the StakeholderEditorPanel.properties file to
     * set the
     * label of the user field. If the property is undefined "User" is
     * used.
     */
    public static final String PROP_LABEL_USER = "User.Label";

```

```

    /**
     * The name to use in the StakeholderEditorPanel.properties file to
     * set the
     * label of the team field. If the property is undefined "Team" is
     * used.
     */
    public static final String PROP_LABEL_TEAM = "Team.Label";

    /**
     * The name to use in the StakeholderEditorPanel.properties file to
     * set the
     * label of the team field. If the property is undefined "Team" is
     * used.
     */
    public static final String PROP_LABEL_PERMISSIONS =
"Permissions.Label";

    private final UserRepository userRepository;
    private UpdateListener updateListener;

    // this is set by the DeleteListener so that the UpdateListener can
    ignore
    // events between when the object was deleted and the panel goes
    away.
    private boolean deleted;

    /**
     * @param commandHandler
     * @param projectCommandFactory
     * @param userRepository
     * @param projectRepository
     */
    public StakeholderEditorPanel(CommandHandler commandHandler,
        ProjectCommandFactory projectCommandFactory, UserRepository
        userRepository,
        ProjectRepository projectRepository) {
        this(StakeholderEditorPanel.class.getName(), commandHandler,
            projectCommandFactory,
            userRepository, projectRepository);
    }

    /**
     * @param resourceBundleName
     * @param commandHandler
     * @param projectCommandFactory
     * @param userRepository
     */

```

```

 * @param projectRepository
 */
public StakeholderEditorPanel(String resourceBundleName,
CommandHandler commandHandler,
ProjectCommandFactory projectCommandFactory, UserRepository
userRepository,
ProjectRepository projectRepository) {
super(resourceBundleName, Stakeholder.class, commandHandler,
projectCommandFactory,
projectRepository);
this.userRepository = userRepository;
}

/**
 * If the editor is editing an existing stakeholder the title
specified in
 * the properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property
 * is not set it then tries the standard PROP_PANEL_TITLE and if that
does
 * not exist it defaults to:<br>
* "Stakeholder: {0}"<br>
* Valid variables are:<br>
* {0} - stakeholder name / username<br>
* {1} - project/domain name<br>
* For new stakeholders it first tries PROP_NEW_OBJECT_PANEL_TITLE,
then
 * PROP_PANEL_TITLE and finally defaults to:<br>
* "New Stakeholder"<br>
*
* @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
* @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
* @see Panel.PROP_PANEL_TITLE
* @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
if (getStakeholder() != null) {
String msgPattern = getResourceBundleHelper(getLocale()).getString(
PROP_EXISTING_OBJECT_PANEL_TITLE,
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Stakeholder: {0}"));
return MessageFormat.format(msgPattern,
(getStakeholder().getUser() != null ?
getStakeholder().getUser().getUsername()
: getStakeholder().getName()), getProjectOrDomain().getName());
} else {
String msg = getResourceBundleHelper(getLocale()).getString(
PROP_NEW_OBJECT_PANEL_TITLE,
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"New Stakeholder"));
return msg;
}
}

@Override
public void setup() {
super.setup();
Stakeholder stakeholder = getStakeholder();
if (stakeholder != null) {
addInput("name", PROP_LABEL_NAME, "Name", new TextField(), new
StringDocumentEx(
stakeholder.getName()));
addInput("user", PROP_LABEL_USER, "User", new SelectField(), new
CombinedListModel(
getProjectUsersUsernames(), (stakeholder.getUser() != null ?
stakeholder
.getUser().getUsername() : ""), true));
addInput("teamName", PROP_LABEL_TEAM, "Team", new ComboBox(),
new CombinedTextListModel(getProjectTeamNames(),
(stakeholder.getTeam() != null ?
stakeholder.getTeam().getName() : "")));
addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals", new
GoalsTable(this,
getResourceBundleHelper(getLocale()), getProjectCommandFactory(),
getCommandHandler(), stakeholder));
addInput("stakeholderPermissions", PROP_LABEL_PERMISSIONS,
"Stakeholder Permissions",
new CheckBoxTreeSet(),
createStakeholderPermissionsSelectionTreeModel(
getProjectRepository().findAvailableStakeholderPermissions(),
stakeholder));
addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
new AnnotationsTable(this, getResourceBundleHelper(getLocale())),
stakeholder);
} else {
addInput("name", PROP_LABEL_NAME, "Name", new TextField(), new
StringDocumentEx());
addInput("user", PROP_LABEL_USER, "User", new SelectField(), new
CombinedListModel(
getProjectUsersUsernames(), "", true));
addInput("teamName", PROP_LABEL_TEAM, "Team", new ComboBox(),

```

```

        new CombinedTextListModel(getProjectTeamNames(), ""));
    addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals", new
GoalsTable(this,
    getResourceBundleHelpergetLocale(), getProjectCommandFactory(),
    getCommandHandler(), null);
    addInput("stakeholderPermissions", PROP_LABEL_PERMISSIONS,
"Stakeholder Permissions",
    new CheckBoxTreeSet(),
    createStakeholderPermissionsSelectionTreeModel(
        getProjectRepository().findAvailableStakeholderPermissions(),
    null));
    addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
    new AnnotationsTable(this, getResourceBundleHelpergetLocale()),
null);
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

```

```

    }

@Override
public void save() {
    try {
        super.save();
        EditStakeholderCommand command =
getProjectCommandFactory().newEditStakeholderCommand();
        command.setStakeholder(getStakeholder());
        command.setProjectOrDomain(getProjectOrDomain());
        command.setEditedBy(getCurrentUser());
        command.setName(getInputValue("name", String.class));
        command.setUsername(getInputValue("user", String.class));
        command.setTeamName(getInputValue("teamName", String.class));
        command.setStakeholderPermissions(getStakeholderPermissionKeys(getI
nputValue(
            "stakeholderPermissions", Set.class), getProjectRepository()
                .findAvailableStakeholderPermissions()));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEve
nt.class,
            updateListener);
            updateListener = null;
        }
        // TODO: remove other listeners?
        getEventDispatcher().dispatchEvent(
            new UpdateEntityEvent(this, command.getStakeholder()));
    } catch (EntityException e) {
        if ((e.getEntityPropertyNames() != null) &&
(e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        } else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
            InvalidStateException ise = (InvalidStateException) e.getCause();
            for (InvalidValue invalidValue : ise.getInvalidValues()) {
                String propertyName = invalidValue.getPropertyName();
                setValidationMessage(propertyName, invalidValue.getMessage());
            }
        } else {
            setGeneralMessage(e.toString());
        }
    } catch (Exception e) {
        log.error("could not save the stakeholder: " + e, e);
    }
}

```

```

        setGeneralMessage("Could not save: " + e);
    }

@Override
public void delete() {
    try {
        DeleteStakeholderCommand deleteStakeholderCommand =
getProjectCommandFactory()
            .newDeleteStakeholderCommand();
        deleteStakeholderCommand.setEditedBy(getCurrentUser());
        deleteStakeholderCommand.setStakeholder(getStakeholder());
        deleteStakeholderCommand =
getCommandHandler().execute(deleteStakeholderCommand);
        deleted = true;
        getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getStakeholder()));
    } catch (Exception e) {
        setGeneralMessage("Could not delete entity: " + e);
    }
}

private Set<String> getProjectTeamNames() {
    Set<String> teamNames = new TreeSet<String>();
    teamNames.add("");
    for (ProjectTeam team : getProjectOrDomain().getTeams()) {
        teamNames.add(team.getName());
    }
    return teamNames;
}

private Set<String> getProjectUsersUsernames() {
    Set<String> usernames = new TreeSet<String>();
    usernames.add("");
    UserSet projectUsers =
getUserRepository().findUsersForRole(ProjectUserRole.class);
    for (User user : projectUsers) {
        usernames.add(user.getUsername());
    }
    return usernames;
}

private CheckBoxTreeSetModel
createStakeholderPermissionsSelectionTreeModel(
    Set<StakeholderPermission> availableStakeholderPermissions,
    Stakeholder stakeholder) {

```

```

    Set<String> optionPaths = new TreeSet<String>(new
Comparator<String>() {
    @Override
    public int compare(String o1, String o2) {
        return o2.compareTo(o1);
    }
});
Set<String> selections = new HashSet<String>();

for (StakeholderPermission permission :
availableStakeholderPermissions) {
    optionPaths.add(generatePermissionPath(permission));
    if ((stakeholder != null) && stakeholder.hasPermission(permission))
{
        selections.add(generatePermissionPath(permission));
    }
}
return new CheckBoxTreeSetModel(optionPaths,
    getGrantablePermissionPaths(getProjectRepository()
        .findAvailableStakeholderPermissions()), selections, true);
}

private Set<String> getGrantablePermissionPaths(
    Set<StakeholderPermission> availableStakeholderPermissions) {
    Set<Class<?>> grantable = new HashSet<Class<?>>();
    Set<String> grantablePermissionPaths = new HashSet<String>();
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder currentStakeholder = project.getUserStakeholder(user);
        if (currentStakeholder != null) {
            for (StakeholderPermission permission :
availableStakeholderPermissions) {
                if
(StakeholderPermissionType.Grant.equals(permission.getPermissionType()))
) {
                    if (currentStakeholder.hasPermission(permission)) {
                        grantable.add(permission.getEntityType());
                    }
                }
            }
            for (StakeholderPermission permission :
availableStakeholderPermissions) {
                if (grantable.contains(permission.getEntityType())
                    || project.getCreatedBy().equals(user)) {
                    grantablePermissionPaths.add(generatePermissionPath(permission));
                }
            }
        }
    }
}
;
```

```

        }
    }
}
return grantablePermissionPaths;
}

private Set<String> getStakeholderPermissionKeys(Set<String>
selectedPaths,
Set<StakeholderPermission> availableStakeholderPermissions) {
log.debug("selectedPaths = " + selectedPaths);
Set<String> stakeholderPermissionKeys = new HashSet<String>();
for (StakeholderPermission permission :
availableStakeholderPermissions) {
    String permissionPath = generatePermissionPath(permission);
    if (selectedPaths.contains(permissionPath)) {
        stakeholderPermissionKeys.add(permission.getPermissionKey());
        log.debug("adding " + permission);
    }
}
return stakeholderPermissionKeys;
}

private String generatePermissionPath(StakeholderPermission
permission) {
    String entityTypeName = permission.getEntityType().get SimpleName();
    String permissionTypeName =
    permission.getPermissionType().toString();
    return (entityTypeName + "/" + permissionTypeName);
}

private ProjectOrDomain getProjectOrDomain() {
    ProjectOrDomain pod = null;
    if (getTargetObject() != null) {
        if (getTargetObject() instanceof ProjectOrDomain) {
            pod = (ProjectOrDomain) getTargetObject();
        } else if (getTargetObject() instanceof Stakeholder) {
            pod = getStakeholder().getProjectOrDomain();
        }
    }
    return pod;
}

private Stakeholder getStakeholder() {
    if (getTargetObject() instanceof Stakeholder) {
        return (Stakeholder) getTargetObject();
    }
}

```

```

        return null;
    }

private UserRepository getUserRepository() {
    return userRepository;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final StakeholderEditorPanel panel;

    private UpdateListener(StakeholderEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        Stakeholder existingStakeholder = panel.getStakeholder();
        if ((e instanceof UpdateEntityEvent) && (existingStakeholder != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            Stakeholder updatedStakeholder = null;
            if (event.getObject() instanceof Stakeholder) {
                updatedStakeholder = (Stakeholder) event.getObject();
                if ((event instanceof DeletedEntityEvent)
                    && existingStakeholder.equals(updatedStakeholder)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingStakeholder));
                    return;
                } else if (event.getObject() instanceof Goal) {
                    Goal updatedGoal = (Goal) event.getObject();
                    if (event instanceof DeletedEntityEvent) {
                        if (existingStakeholder.getGoals().contains(updatedGoal)) {
                            existingStakeholder.getGoals().remove(updatedGoal);
                        }
                        updatedStakeholder = existingStakeholder;
                    } else if
(updatedGoal.getReferers().contains(existingStakeholder)) {
                        for (GoalContainer gc : updatedGoal.getReferers()) {
                            if (gc.equals(existingStakeholder)) {
                                updatedStakeholder = (Stakeholder) gc;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```

        break;
    }
}
} else if (event.getObject() instanceof Annotation) {
    Annotation updatedAnnotation = (Annotation) event.getObject();
    if (event instanceof DeletedEntityEvent) {
        if
(existingStakeholder.getAnnotations().contains(updatedAnnotation)) {
            existingStakeholder.getAnnotations().remove(updatedAnnotation);
        }
        updatedStakeholder = existingStakeholder;
    } else if
(updatedAnnotation.getAnnotatables().contains(existingStakeholder)) {
        for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
            if (annotatable.equals(existingStakeholder)) {
                updatedStakeholder = (Stakeholder) annotatable;
                break;
            }
        }
    }
    if ((updatedStakeholder != null) &&
updatedStakeholder.equals(existingStakeholder)) {
        // TODO: check the input fields to see if the user has made
        // a change before resetting the object and updating the
        // input fields.
        panel.setInputValue("name", updatedStakeholder.getName());
        panel.setInputValue("user",
            (updatedStakeholder.getUser() != null ?
updatedStakeholder.getUser()
                .getUsername() : ""));
        panel.setInputValue("teamName",
            (updatedStakeholder.getTeam() != null ?
updatedStakeholder.getTeam()
                .getName() : ""));
        // TODO: this causes an error for echo2 rendering of the
        // tree
        // panel.setInputValue("stakeholderPermissions", panel
        // .getStakeholderPermissionKeys(updatedStakeholder));
        panel.setInputValue("goals", updatedStakeholder);
        panel.setInputValue("annotations", updatedStakeholder);
        panel.setTargetObject(updatedStakeholder);
    }
}
}

```

```

    }
}

```

stakeholderimpl.java

```

/*
 * $Id: StakeholderImpl.java,v 1.39 2009/02/20 07:27:40 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl;

import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.adapters.XmlJavaTypeAdapter;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.Goal;

```

```

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectTeam;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.impl.ProjectTeamImpl.Tea2TeamIm
plAdapter;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchUserException;
import edu.harvard.fas.rregan.requel.user.impl.UserImpl;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "stakeholders", uniqueConstraints =
{ @UniqueConstraint(columnNames =
    "projectordomain_id", "name", "user_id" ) })
@XmlRootElement(name = "stakeholder", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "stakeholder", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class StakeholderImpl extends AbstractProjectOrDomainEntity
implements Stakeholder {
    static final long serialVersionUID = 0L;

    private Set<StakeholderPermission> stakeholderPermission = new
    HashSet<StakeholderPermission>();
    private Set<Goal> goals = new TreeSet<Goal>();
    private User user;
    private ProjectTeam team;

    /**
     * Create a stakeholder for a user.
     *
     * @param projectOrDomain
     * @param user
     * @param createdBy
     * @param name
     */
}

```

```

    public StakeholderImpl(ProjectOrDomain projectOrDomain, User
createdBy, User user) {
    super(projectOrDomain, createdBy, null);
    this.user = user;
    // add to collection last so that sorting in the collection by
entity
    // properties has access to all the properties.
    projectOrDomain.getStakeholders().add(this);
}

/**
 * Create a stakeholder for a non-user entity.
*
 * @param projectOrDomain
 * @param createdBy
 * @param name
 */
public StakeholderImpl(ProjectOrDomain projectOrDomain, User
createdBy, String name) {
    super(projectOrDomain, createdBy, name);
    projectOrDomain.getStakeholders().add(this);
}

protected StakeholderImpl() {
    // for hibernate
}

@Override
@Column(nullable = true, unique = false)
XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return super.getName();
}

// hack for JAXB to set the name, for some reason it won't use the
inherited
// method.
@Override
public void setName(String name) {
    super.setName(name);
}

/**
 * This is for JAXB to get a unique type specific id for use in an
exported
 * file.

```

```

/*
@Transient
@XmlID
@XmlAttribute(name = "id")
public String getXmlId() {
    return "STK_" + getId();
}

/**
 * This is for displaying a description in the user interface
 */
@XmlTransient
@Transient
public String getDescription() {
    if (isUserStakeholder()) {
        return "Stakeholder: " + getUser().getUsername();
    }
    return "Stakeholder: " + getName();
}

@Transient
@XmlTransient
@Override
public boolean isUserStakeholder() {
    return (getUser() != null);
}

// TODO: there should be some way to mark this as optional in the xml
schema
XmlElement(name = "user", type = UserImpl.class, nullable = true,
required = false, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
// @XmlElementRef(type = UserImpl.class)
@ManyToOne(targetEntity = UserImpl.class, cascade =
{ CascadeType.REFRESH }, optional = true)
public User getUser() {
    return user;
}

/**
 * @param user
 */
public void setUser(User user) {
    this.user = user;
}

@XmlIDREF()

```

```

@XmlAttribute(name = "team")
@XmlJavaTypeAdapter(Team2TeamImplAdapter.class)
@Transient
public ProjectTeam getTeam() {
    return getTeamInternal();
}

public void setTeam(ProjectTeam team) {
    if (getTeamInternal() != null) {
        getTeamInternal().getMembers().remove(this);
    }
    setTeamInternal(team);
    if (team != null) {
        team.getMembers().add(this);
    }
}

@ManyToOne(targetEntity = ProjectTeamImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, optional = true)
protected ProjectTeam getTeamInternal() {
    return team;
}

public void setTeamInternal(ProjectTeam team) {
    this.team = team;
}

XmlElementWrapper(name = "projectPermissions", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel", required = false)
@XmlElementRef(type = StakeholderPermissionImpl.class)
@ManyToMany(targetEntity = StakeholderPermissionImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "stakeholders_permissions", joinColumns =
{ @JoinColumn(name = "stakeholder_id") }, inverseJoinColumns =
{ @JoinColumn(name = "stakeholder_permission_id") })
public Set<StakeholderPermission> getStakeholderPermissions() {
    return stakeholderPermission;
}

protected void setStakeholderPermissions(Set<StakeholderPermission>
stakeholderPermission) {
    this.stakeholderPermission = stakeholderPermission;
}

```

```

public boolean hasPermission(Class<?> entityType,
StakeholderPermissionType permissionType) {
    for (StakeholderPermission stakeholderPermission :
getStakeholderPermissions()) {
        if (stakeholderPermission.getPermissionKey().equals(
            StakeholderPermissionImpl.generatePermissionKey(entityType,
            permissionType))) {
            return true;
        }
    }
    return false;
}

public boolean hasPermission(StakeholderPermission
stakeholderPermission) {
    return getStakeholderPermissions().contains(stakeholderPermission);
}

@Override
public void grantStakeholderPermission(StakeholderPermission
stakeholderPermission) {
    getStakeholderPermissions().add(stakeholderPermission);
}

@Override
public void revokeStakeholderPermission(StakeholderPermission
stakeholderPermission) {
    getStakeholderPermissions().remove(stakeholderPermission);
}

/***
 * @see
edu.harvard.fas.rregan.requel.project.GoalContainer#getGoals()
 */
@Override
@OneToMany(targetEntity = GoalImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@XmlElementWrapper(name = "goals", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "goalRef", type = GoalImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public Set<Goal> getGoals() {
    return goals;
}

```

```

protected void setGoals(Set<Goal> goals) {
    this.goals = goals;
}

/**
 * @see java.lang.Comparable#compareTo(java.lang.Object)
 */
@Override
public int compareTo(Stakeholder o) {
    if (this == o) {
        return 0;
    }
    String thisName = (getUser() != null ? getUser().getUsername() :
getName());
    String thatName = (o.getUser() != null ? o.getUser().getUsername() :
o.getName());
    if (thisName == null) {
        return -1;
    }
    return thisName.compareToIgnoreCase(thatName);
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
        result = prime * result + ((getUser() == null) ? 0 :
getUser().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {

```

```

    return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
    return false;
}
final StakeholderImpl other = (StakeholderImpl) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getName() == null) {
    if (other.getName() != null) {
        return false;
    }
} else if (!getName().equals(other.getName())) {
    return false;
}
if (getUser() == null) {
    if (other.getUser() != null) {
        return false;
    }
} else if (!getUser().equals(other.getUser())) {
    return false;
}
return true;
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *          the user to be set as the created by if no user is
supplied.
 * @see UnmarshallerListener
 */
public void afterUnmarshal(final UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {

```

```

        try {
            if (StakeholderImpl.this.getUser() != null) {
                try {
                    User existingUser = userRepository
                        .findUserByUsername(StakeholderImpl.this.getUser()
                            .getUsername());
                    StakeholderImpl.this.setUser(existingUser);
                } catch (NoSuchUserException e) {
                    // new organization
                    userRepository.persist(StakeholderImpl.this.getUser());
                }
            }
            // update the references to goals
            for (Goal goal : getGoals()) {
                goal.getReferers().add(StakeholderImpl.this);
            }
            } catch (RuntimeException e) {
                throw e;
            } catch (Exception e) {
                throw new SAXException2(e);
            }
        });
    }
}

```

stakeholdernavigatorpanel.java

```

/*
 * $Id: StakeholderNavigatorPanel.java,v 1.1 2008/09/12 22:44:16
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

```

```

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
l;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class StakeholderNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(StakeholderNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private StakeholderUpdateListener updateListener;
}

```

```

private ProjectOrDomain pod;

/**
 * Property name to use in the StakeholderNavigatorPanel.properties
to set
 * the label on the new stakeholder button.
 */
public static final String PROP_NEW_STAKEHOLDER_BUTTON_LABEL =
"NewStakeholderButton.Label";

/**
 * Property name to use in the StakeholderNavigatorPanel.properties
to set
 * the label on the edit stakeholder button in each row of the table.
 */
public static final String PROP_EDIT_STAKEHOLDER_BUTTON_LABEL =
>EditStakeholderButton.Label";

/**
 * Property name to use in the StakeholderNavigatorPanel.properties
to set
 * the label on the view goal button in each row of the table when
the user
 * doesn't have edit permission.
 */
public static final String PROP_VIEW_STAKEHOLDER_BUTTON_LABEL =
"ViewStakeholderButton.Label";

/**
 */
public StakeholderNavigatorPanel() {
    super(StakeholderNavigatorPanel.class.getName(), Project.class,
        ProjectManagementPanelNames.PROJECT_STAKEHOLDERS_NAVIGATOR_PANEL_N
AME);
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
                String editStakeholderButtonLabel = null;
                if (isReadOnlyMode()) {
                    editStakeholderButtonLabel =
getResourceBundleHelpergetLocale())

```

```

        .getString(PROP_VIEW_STAKEHOLDER_BUTTON_LABEL, "View");
    } else {
        editStakeholderButtonLabel =
getResourceBundleHelpergetLocale())
        .getString(PROP_EDIT_STAKEHOLDER_BUTTON_LABEL, "Edit");
    }

NavigationEvent openStakeholderEditor = new OpenPanelEvent(this,
    PanelActionType.Editor, stakeholder, Stakeholder.class, null,
    WorkflowDisposition.NewFlow);
NavigatorButton editStakeholderButton = new NavigatorButton(
    editStakeholderButtonLabel, getEventDispatcher(),
    openStakeholderEditor);
editStakeholderButton.setStyleName(STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
editStakeholderButton.setLayoutData(rld);
return editStakeholderButton;
}
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
            String displayValue = null;
            if (stakeholder.getUser() != null) {
                User user = stakeholder.getUser();
                if ((user.getName() != null) && (user.getName().length() > 0))
{
                    displayValue = user.getName() + " [ " + user.getUsername() + "
];
                } else {
                    displayValue = user.getUsername();
                }
            } else {
                displayValue = stakeholder.getName();
            }
            return displayValue;
        }
    });
};

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("User?",
    new NavigatorTableCellValueFactory() {

```

```

        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
            return (stakeholder.getUser() != null ? "yes" : "no");
        }
    }));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Team",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
            return (stakeholder.getTeam() != null ?
stakeholder.getTeam().getName()
: "");
        }
    }));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Email
Address",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
            String displayValue = null;
            if (stakeholder.getUser() != null) {
                User user = stakeholder.getUser();
                displayValue = user.getEmailAddress();
            } else {
                displayValue = "";
            }
            return displayValue;
        }
    }));
};

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Phone
Number",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {

```

```

        Stakeholder stakeholder = (Stakeholder)
model.getBackingObject(row);
String displayValue = null;
if (stakeholder.getUser() != null) {
    User user = stakeholder.getUser();
    displayValue = user.getPhoneNumber();
} else {
    displayValue = "";
}
return displayValue;
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            return entity.getCreatedBy().getUsername();
        }
    });
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    });
);
setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "Stakeholders: {0}"<br>
 * Valid variables are:<br>

```

```

        * {0} - project/domain name<br>
        *
        * @see Panel.PROP_PANEL_TITLE
        * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
        */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "Stakeholders: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_NEW_STAKEHOLDER_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
        closeEvent);

```

```

closeButton.setStyleName(STYLE_NAME_DEFAULT);
buttonsWrapper.add(closeButton);

if (!isReadOnlyMode()) {
    String newStakeholderButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_NEW_STAKEHOLDER_BUTTON_LABEL, "Add");
    NavigationEvent openStakeholderEditor = new OpenPanelEvent(this,
        PanelActionType.Editor, getProjectOrDomain(), Stakeholder.class,
null,
        WorkflowDisposition.NewFlow);
    NavigatorButton newStakeholderButton = new
NavigatorButton(newStakeholderButtonLabel,
    getEventDispatcher(), openStakeholderEditor);
    newStakeholderButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(newStakeholderButton);
}

add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new StakeholderUpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder
                .hasPermission(Stakeholder.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {

```

```
pod = (ProjectOrDomain) targetObject;
super.setTargetObject(((ProjectOrDomain)
targetObject).getStakeholders());
} else {
log.error("unexpected target object " + targetObject);
}
}

protected ProjectOrDomain getProjectOrDomain() {
return pod;
}

private static class StakeholderUpdateListener implements
ActionListener {
static final long serialVersionUID = 0L;

private final StakeholderNavigatorPanel panel;

private StakeholderUpdateListener(StakeholderNavigatorPanel panel) {
this.panel = panel;
}

@Override
public void actionPerformed(ActionEvent e) {
if (e instanceof UpdateEntityEvent) {
UpdateEntityEvent event = (UpdateEntityEvent) e;
if (event.getObject() instanceof ProjectOrDomain) {
ProjectOrDomain updatedPod = (ProjectOrDomain) event.getObject();
if (panel.getProjectOrDomain().equals(updatedPod)) {
panel.setTargetObject(updatedPod);
}
} else if (event.getObject() instanceof Stakeholder) {
Stakeholder updatedStakeholder = (Stakeholder) event.getObject();
ProjectOrDomain updatedPod =
updatedStakeholder.getProjectOrDomain();
if (panel.getProjectOrDomain().equals(updatedPod)) {
panel.setTargetObject(updatedPod);
}
}
}
}
}
```

stakeholderpermission.java

```
/*
 * $Id: StakeholderPermission.java,v 1.3 2008/09/06 09:31:59 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

/**
 * Permissions at the project level assigned to stakeholders for
acting on
 * project elements.
 *
 * @author ron
 */
public interface StakeholderPermission extends
Comparable<StakeholderPermission> {

    /**
     * @return
     */
    public String getPermissionKey();

    /**
     * @return The class of the project entity the permission applies to.
     */
    public Class<?> getEntityType();

    /**
     * @return The type of permissions like Edit or Grant
     */
    public StakeholderPermissionType getPermissionType();
}
```

stakeholderpermissionimpl.java

```
/*
 * $Id: StakeholderPermissionImpl.java,v 1.11 2008/08/25 02:20:01
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "stakeholder_permissions", uniqueConstraints =
{ @UniqueConstraint(columnNames = {
    "entity_type", "permission_type" }) })
@XmlRootElement(name = "projectPermission", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "projectPermission", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class StakeholderPermissionImpl implements
StakeholderPermission {

    private Long id;
    private Class<?> entityType;
    private StakeholderPermissionType permissionType;

    /**

```

```

* @param entityType
* @param permissionType
*/
public StakeholderPermissionImpl(Class<?> entityType,
StakeholderPermissionType permissionType) {
super();
setEntityType(entityType);
setPermissionType(permissionType);
}

protected StakeholderPermissionImpl() {
// for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
protected Long getId() {
return id;
}

protected void setId(Long id) {
this.id = id;
}

@Transient
public String getPermissionKey() {
return generatePermissionKey(getEntityType(), getPermissionType());
}

@Column(name = "entity_type", nullable = false, updatable = false)
@XmlAttribute(name = "entityType")
@XmlJavaTypeAdapter(StakeholderEntityTypeAdapter.class)
public Class<?> getEntityType() {
return entityType;
}

protected void setEntityType(Class<?> entityType) {
this.entityType = entityType;
}

@Enumerated(EnumType.STRING)
@Column(name = "permission_type", nullable = false, updatable =
false)
@XmlAttribute(name = "permissionType")
@XmlJavaTypeAdapter(StakeholderPermissionTypeAdapter.class)
public StakeholderPermissionType getPermissionType() {
return permissionType;
}

}

protected void setPermissionType(StakeholderPermissionType
permissionType) {
this.permissionType = permissionType;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
if (tmpHashCode == null) {
if (getId() != null) {
tmpHashCode = new Integer(getId().hashCode());
}
final int prime = 31;
int result = 1;
result = prime * result + ((getEntityType() == null) ? 0 :
getEntityType().hashCode());
result = prime * result
+ (getPermissionType() == null) ? 0 :
getPermissionType().hashCode();
tmpHashCode = new Integer(result);
}
return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
if (this == obj) {
return true;
}
if (obj == null) {
return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
return false;
}
final StakeholderPermissionImpl other = (StakeholderPermissionImpl)
obj;
if ((getId() != null) && getId().equals(other.getId())) {
return true;
}
if (getEntityType() == null) {
if (other.getEntityType() != null) {
return false;
}
}
}

```

```

} else if (!getEntityType().equals(other.getEntityType())) {
    return false;
}
if (getPermissionType() == null) {
    if (other.getPermissionType() != null) {
        return false;
    }
} else if (!getPermissionType().equals(other.getPermissionType())) {
    return false;
}
return true;
}

@Override
public int compareTo(StakeholderPermission o) {
    int entityTypeCompare = (getEntityType() == null ? -1 :
getEntityType().getName()
        .compareTo(o.getEntityType().getName()));
    int permissionTypeCompare = (getPermissionType() == null ? -1 :
getPermissionType()
        .compareTo(o.getPermissionType()));
    return (entityTypeCompare == 0 ? permissionTypeCompare :
entityTypeCompare);
}

@Override
public String toString() {
    return getPermissionKey();
}

public static final String generatePermissionKey(Class<?> entityType,
    StakeholderPermissionType permissionType) {
    return entityType.getName() + "[" + permissionType.toString() + "]";
}

/**
 * This class is used by JAXB to convert the
StakeholderPermissionType of a
 * StakeholderPermission into a string for an attribute in the xml
file and
 * the reverse when unmarshaling.
 *
 * @author ron
 */
@XmlTransient
public static class StakeholderPermissionTypeAdapter extends
    XmlAdapter<String, StakeholderPermissionType> {

```

```

    @Override
    public StakeholderPermissionType unmarshal(String typeString) throws
Exception {
        return StakeholderPermissionType.valueOf(typeString);
    }

    @Override
    public String marshal(StakeholderPermissionType type) throws
Exception {
        return type.toString();
    }

    /**
     * This class is used by JAXB to convert the
StakeholderPermissionType of a
     * StakeholderPermission into a string for an attribute in the xml
file and
     * the reverse when unmarshaling.
     *
     * @author ron
     */
    @XmlTransient
    public static class StakeholderEntityTypeAdapter extends
        XmlAdapter<String, Class<?>> {

        @Override
        public Class<?> unmarshal(String className) throws Exception {
            return Class.forName(className);
        }

        @Override
        public String marshal(Class<?> type) throws Exception {
            return type.getName();
        }

        /**
         * This is for JAXB to patchup the parent/child relationship and to
patchup
         * existing persistent objects for the objects that are attached
directly to
         * this object.
         *
         * @param projectRepository
         * @param parent -
        
```

```

        *           the stakeholder that should be granted the permission.
        * @see UnmarshallerListener
        */
public void afterUnmarshal(final ProjectRepository projectRepository,
final Object parent) {
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {
                StakeholderPermission permission =
                    StakeholderPermissionImpl.this;
                ((StakeholderImpl)
parent).getStakeholderPermissions().remove(permission);
                StakeholderPermission existingPermission = projectRepository
                    .findStakeholderPermission(permission.getEntityType(),
                permission
                    .getPermissionType());
                ((StakeholderImpl)
parent).getStakeholderPermissions().add(existingPermission);
            } catch (RuntimeException e) {
                throw e;
            } catch (Exception e) {
                throw new SAXException2(e);
            }
        }
    });
}
}

```

stakeholderpermissionsinitializer.java

```

/*
 * $Id: StakeholderPermissionsInitializer.java,v 1.15 2009/03/22
11:08:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.repository.init;

import java.util.ArrayList;
import java.util.Collection;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

```

```

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.GlossaryTerm;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.ReportGenerator;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import edu.harvard.fas.rregan.requel.project.StakeholderPermission;
import edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.impl.StakeholderPermissionImpl;

/**
 * @author ron
 */
@Component("stakeholderPermissionsInitializer")
@Scope("prototype")
public class StakeholderPermissionsInitializer extends
AbstractSystemInitializer {

    private final ProjectRepository projectRepository;

    /**
     * @param projectRepository
     */
    @Autowired
    public StakeholderPermissionsInitializer(ProjectRepository
projectRepository) {
        super(10);
        this.projectRepository = projectRepository;
    }

    @Override
    @Transactional(propagation = Propagation.REQUIRED)
    public void initialize() {
        log.debug("update stakeholder permissions...");
        for (StakeholderPermission permission : getPermissionTypes()) {
            try {
                permission = projectRepository.findStakeholderPermission(

```

```

        permission.getEntityType(), permission.getPermissionType());
    log.debug(permission + " is already persistent.");
} catch (EntityException e) {
    log.debug("creating: " + permission);
    permission = projectRepository.persist(permission);
}
}

private Collection<StakeholderPermission> getPermissionTypes() {
    Collection<StakeholderPermission> entityTypes = new
    ArrayList<StakeholderPermission>();
    entityTypes
        .add(new StakeholderPermissionImpl(Project.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(Project.class,
        StakeholderPermissionType.Grant));

    entityTypes.add(new StakeholderPermissionImpl(Annotation.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(Annotation.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(Annotation.class,
        StakeholderPermissionType.Delete));
    entityTypes.add(new StakeholderPermissionImpl(Goal.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(Goal.class,
        StakeholderPermissionType.Grant));
    entityTypes
        .add(new StakeholderPermissionImpl(Goal.class,
        StakeholderPermissionType.Delete));
    entityTypes.add(new StakeholderPermissionImpl(Actor.class,
        StakeholderPermissionType.Edit));
    entityTypes
        .add(new StakeholderPermissionImpl(Actor.class,
        StakeholderPermissionType.Grant));
    entityTypes
        .add(new StakeholderPermissionImpl(Actor.class,
        StakeholderPermissionType.Delete));
    entityTypes.add(new StakeholderPermissionImpl(Stakeholder.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(Stakeholder.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(Stakeholder.class,
        StakeholderPermissionType.Delete));
    entityTypes.add(new StakeholderPermissionImpl(GlossaryTerm.class,
        StakeholderPermissionType.Edit));
}

```

```

    entityTypes.add(new StakeholderPermissionImpl(GlossaryTerm.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(GlossaryTerm.class,
        StakeholderPermissionType.Delete));
    entityTypes.add(new StakeholderPermissionImpl(Story.class,
        StakeholderPermissionType.Edit));
    entityTypes
        .add(new StakeholderPermissionImpl(Story.class,
        StakeholderPermissionType.Grant));
    entityTypes
        .add(new StakeholderPermissionImpl(Story.class,
        StakeholderPermissionType.Delete));

    entityTypes
        .add(new StakeholderPermissionImpl(UseCase.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(UseCase.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(UseCase.class,
        StakeholderPermissionType.Delete));

    entityTypes.add(new StakeholderPermissionImpl(Scenario.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(Scenario.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(Scenario.class,
        StakeholderPermissionType.Delete));

    entityTypes.add(new StakeholderPermissionImpl(ReportGenerator.class,
        StakeholderPermissionType.Edit));
    entityTypes.add(new StakeholderPermissionImpl(ReportGenerator.class,
        StakeholderPermissionType.Grant));
    entityTypes.add(new StakeholderPermissionImpl(ReportGenerator.class,
        StakeholderPermissionType.Delete));

    return entityTypes;
}
}

```

stakeholderpermissiontype.java

```

/*
 * $Id: StakeholderPermissionType.java,v 1.4 2009/02/20 10:26:17
 * rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project;

/**
 * The types of permissions that can be granted to stakeholders for
working with
 * project entities.
 *
 * @author ron
 */
public enum StakeholderPermissionType {

    /**
     * Allow a user to create and edit a project entity.
     */
    Edit(),

    /**
     * Allow a user to delete a project entity.
     */
    Delete(),

    /**
     * Allow a user to grant permissions for working with different
project
     * entities.
     */
    Grant();

    private StakeholderPermissionType() {
    }
}

```

staleobjectstateexceptionadapter.java

```

/*
 * $Id: StaleObjectStateExceptionAdapter.java,v 1.1 2008/12/13
00:41:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.repository.jpa;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.requel.EntityLockException;

/**

```

```

        * @author ron
        */
public class StaleObjectStateExceptionAdapter implements
EntityExceptionAdapter {

    /*
     * (non-Javadoc)
     *
     * @see
edu.harvard.fas.rregan.requel.EntityExceptionAdapter#convert(java.lang
.Throwable,
     *         java.lang.Class, java.lang.Object,
     *         edu.harvard.fas.rregan.requel.EntityExceptionActionType)
     */
@Override
public EntityLockException convert(Throwable original, Class<?>
entityType, Object entity,
    EntityExceptionActionType actionType) {
    return EntityLockException.staleEntity(entityType, entity,
actionType);
}

}

```

stanfordlexicalizedparser.java

```

/*
 * $Id: StanfordLexicalizedParser.java,v 1.3 2009/01/28 10:18:45
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.io.CharArrayReader;
import java.io.ObjectInputStream;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.zip.GZIPInputStream;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;

```

```

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.ParserException;
import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.ling.TaggedWord;
import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
import edu.stanford.nlp.process.Tokenizer;
import edu.stanford.nlp.trees.GrammaticalStructure;
import edu.stanford.nlp.trees.GrammaticalStructureFactory;
import edu.stanford.nlp.trees.PennTreebankLanguagePack;
import edu.stanford.nlp.trees.Tree;
import edu.stanford.nlp.trees.TreebankLanguagePack;
import edu.stanford.nlp.trees.TypedDependency;

/**
 * Wrapper for Stanford Parser.
 *
 * @author ron
 */
@Component("stanfordParser")
public class StanfordLexicalizedParser implements
NLPPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(StanfordLexicalizedParser.class);

    /**
     * The name of the property in the Parser.properties file that
contains the
     * path to the serialized/zipped parser model relative to the
classpath. By
     * default the path is "resources/nlp/stanford-
parser/englishPCFG.ser.gz"
    */
    public static final String PROP_PARSER_FILE =
"StanfordParserFileModel";

    /**
     * The default parser model
    */
    public static final String PROP_PARSER_FILE_DEFAULT = "resources/nlp/
stanford-parser/englishPCFG.ser.gz";

    /**
     * The name of the property in the Parser.properties file for the
parameters
     */

    /**
     * to pass to the LexicalizedParser. The parameters are expected to
be of
     * the form "-param1 param1value -param2 param2value ..." where each
token
     * is delimited by whitespace. The parameters will be split into a
String
     * array and passed to the parser via the setOptionsFlags. By default
no
     * parameters are supplied to the parser.
     */
    public static final String PROP_OPTION_FLAGS = "optionFlags";

    private static LexicalizedParser parser;
    private static final TreebankLanguagePack tlp = new
PennTreebankLanguagePack();
    private static final GrammaticalStructureFactory gsf =
tlp.grammaticalStructureFactory();

    static {
        try {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                StanfordLexicalizedParser.class.getName());
            String parserFile =
resourceBundleHelper.getString(PROP_PARSER_FILE,
                PROP_PARSER_FILE_DEFAULT);

            // TODO: remove empty option elements
            String[] optionFlags =
resourceBundleHelper.getString(PROP_OPTION_FLAGS, "").split(
                "\t\n");
            parser = new LexicalizedParser(new ObjectInputStream(new
GZIPInputStream(
                StanfordLexicalizedParser.class.getClassLoader()
                    .getResourceAsStream(parserFile))));
            if ((optionFlags != null) && (optionFlags.length > 0) &&
(optionFlags[0].length() > 0)) {
                parser.setOptionFlags(optionFlags);
            }
        } catch (Exception e) {
            parser = null;
            throw new ExceptionInInitializerError(e);
        }
    }
}

```

```

/*
 */
public StanfordLexicalizedParser() {
}

@Override
public NLPText process(NLPText text) {
    if (text.hasText()) {
        if
(GrammaticalStructureLevel.PARAGRAPH.equals(text.getGrammaticalStructu
reLevel())) {
            for (NLPText sentence : text.getChildren()) {
                process(sentence);
            }
        } else if (text.is(GrammaticalStructureLevel.SENTENCE)
            && (text.getChildren().size() == 0)) {
            parse(text);
        }
    }
    return text;
}

private static synchronized void parse(NLPText text) throws
ParserException {
    List<? extends HasWord> tokens = null;
    if (text.getLeaves().isEmpty()) {
        String sentence = prepareSentence(text.getText());
        Tokenizer<? extends HasWord> tokenizer =
        tlp.getTokenizerFactory().getTokenizer(
            new CharArrayReader(sentence.toCharArray()));
        tokens = tokenizer.tokenize();
    } else {
        List<TaggedWord> xtokens = new ArrayList<TaggedWord>();
        for (NLPText word : text.getLeaves()) {
            if (word.getParseTag() != null) {
                xtokens.add(new TaggedWord(word.getText(),
                word.getParseTag().getText()));
            } else {
                xtokens.add(new TaggedWord(word.getText()));
            }
        }
        tokens = xtokens;
    }
    if (parser.parse(tokens)) {
        Tree stanfordTree = parser.getBestParse();
        copyStanfordTreeToNLPText((NLPTextImpl) text, stanfordTree,
stanfordTree.getLeaves(),
    }
}

new Counter()));

GrammaticalStructure gs =
gsf.newGrammaticalStructure(stanfordTree);
Collection<TypedDependency> typedDependencies =
gs.typedDependencies();
List<NLPText> leaves = text.getLeaves();
for (TypedDependency dependency : typedDependencies) {
    NLPTextImpl governor = (NLPTextImpl)
leaves.get(dependency.gov().index() - 1);
    NLPTextImpl dependent = (NLPTextImpl)
leaves.get(dependency.dep().index() - 1);
    GrammaticalRelationType type = GrammaticalRelationType
        .getGrammaticalRelationByShortName(dependency.reln().getShortNam
e());
    text.getGrammaticalRelations().add(
        new GrammaticalRelationImpl(type, governor, dependent));
}
} else {
    throw ParserException.parseFailed();
}
}

private static class Counter {
    private int count = 0;

    protected void incr() {
        count++;
    }

    protected int getCount() {
        return count;
    }
}

private static void copyStanfordTreeToNLPText(NLPTextImpl parent,
Tree stanfordTree,
    final List<Tree> stanfordLeaves, Counter wordIndexCounter) {
    ParseTag tag = ParseTag.tagOf(stanfordTree.value());

    NLPTextImpl node;
    if (stanfordTree.isPreTerminal()) {
        // word level tags
        String word = stanfordTree.firstChild().value();
        node = new NLPTextImpl(parent, wordIndexCounter.getCount(), word,
tag);
    }
}

```

```

parent.getChildren().add(node);
wordIndexCounter.incr();
} else {
if (ParseTag.ROOT.equals(tag)) {
// skip the root
parent.setParseTag(tag);
node = parent;
} else {
// phrase and clause level tags
node = new NLPTextImpl(parent, tag);
parent.getChildren().add(node);
}
for (Tree child : stanfordTree.children()) {
copyStanfordTreeToNLPText(node, child, stanfordLeaves,
wordIndexCounter);
}
}

private static String prepareSentence(String sentence) {
sentence = sentence.replaceAll(" *, *", " , ");
sentence = sentence.replaceAll(" *\\". *, " \\\". ");
sentence = sentence.replaceAll(" *\\\"! *", " \\\"! ");
sentence = sentence.replaceAll(" *\\\"? *", " \\\"? ");
sentence = sentence.replaceAll(" *: *", " : ");
sentence = sentence.replaceAll(" *; *", " ; ");
sentence = sentence.replaceAll(" *\\\"( *", " \\\"( ");
sentence = sentence.replaceAll(" *\\\") *", " \\\") ");
sentence = sentence.replaceAll(" *\\\"[ *", " \\\"[ ");
sentence = sentence.replaceAll(" *\\\"] *", " \\\"] ");
sentence = sentence.replaceAll(" *\\\"{ *", " \\\"{ ");
sentence = sentence.replaceAll(" *\\\"} *", " \\\"} ");
sentence = sentence.replaceAll(" *< *", " < ");
sentence = sentence.replaceAll(" *> *", " > ");
sentence = removeBrackets(sentence);
return sentence;
}

private static String removeBrackets(String sentence) {
sentence = removeBrackets(sentence, "(", ")");
sentence = removeBrackets(sentence, "[", "]");
sentence = removeBrackets(sentence, "{", "}");
sentence = removeBrackets(sentence, "<", ">");
return sentence;
}

```

```

private static String removeBrackets(String sentence, String
openBracket, String closeBracket) {
int openIdx = 0;
while (openIdx != -1) {
openIdx = sentence.indexOf(openBracket);
if (openIdx == -1) {
return sentence;
}
int closeIdx = sentence.indexOf(closeBracket, openIdx);
if (closeIdx == -1) {
closeIdx = sentence.length() - 1;
}
String start = sentence.substring(0, openIdx);
String end = sentence.substring(closeIdx + 1, sentence.length());
sentence = start + end;
}
return sentence;
}
}

```

stanfordnameentityrecognizer.java

```

/*
 * $Id: StanfordNameEntityRecognizer.java,v 1.4 2009/02/10 03:30:46
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import java.util.ArrayList;
import java.util.List;
import java.util.zip.GZIPInputStream;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParserException;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;

```

```

import edu.harvard.fas.rregan.nlp.dictionary.Word;
import edu.stanford.nlp.ie.AbstractSequenceClassifier;
import edu.stanford.nlp.ie.crf.CRFClassifier;
import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.ling.TaggedWord;
import edu.stanford.nlp.ling.CoreAnnotations.AnswerAnnotation;
import edu.stanford.nlp.ling.CoreAnnotations.PositionAnnotation;
import edu.stanford.nlp.trees.PennTreebankLanguagePack;
import edu.stanford.nlp.trees.TreebankLanguagePack;

/**
 * Wrapper for Stanford Named Entity Recognizer.
 *
 * @author ron
 */
@Component("stanfordNameEntityRecognizer")
public class StanfordNameEntityRecognizer implements
NLPProcessor<NLPText> {
    private static final Logger log =
Logger.getLogger(StanfordNameEntityRecognizer.class);

    /**
     * The name of the property in the
StanfordNameEntityRecognizer.properties
     * file that contains the path to the serialized/zipped model
relative to
     * the classpath. By default the path is
     * "resources/nlp/stanford-ner/ner-eng-ie.crf-3-all2008-
distsim.ser.gz"
    */
    public static final String PROP_MODEL_FILE =
"StanfordNameEntityRecognizerModelFile";

    /**
     * The default parser model
    */
    public static final String PROP_MODEL_FILE_DEFAULT =
"resources/nlp/stanford-ner/ner-eng-ie.crf-3-all2008-distsim.ser.gz";

    private static AbstractSequenceClassifier classifier;
    private static final TreebankLanguagePack tlp = new
PennTreebankLanguagePack();

    static {
        try {

```

```

            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
        StanfordLexicalizedParser.class.getName());

            String parserFile = resourceBundleHelper.getString(PROP_MODEL_FILE,
PROP_MODEL_FILE_DEFAULT);

            classifier = CRFClassifier.getClassifier(new GZIPInputStream(
                StanfordNameEntityRecognizer.class.getClassLoader().getResourceAs
Stream(
                    parserFile)));

        } catch (Exception e) {
            classifier = null;
            throw new ExceptionInInitializerError(e);
        }
    }

    private final DictionaryRepository dictionaryRepository;
    private Sense personSense;
    private Sense locationSense;
    private Sense organizationSense;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public StanfordNameEntityRecognizer(DictionaryRepository
dictionaryRepository) {
        this.dictionaryRepository = dictionaryRepository;
    }

    @Override
    public NLPText process(NLPText text) {
        if (!initialized()) {
            init();
        }

        if (text.hasText()) {
            if
(GrammaticalStructureLevel.PARAGRAPH.equals(text.getGrammaticalStructu
reLevel())) {
                for (NLPText sentence : text.getChildren()) {
                    process(sentence);
                }
            } else if (GrammaticalStructureLevel.SENTENCE.equals(text
.getGrammaticalStructureLevel())) {

```

```

        ner(text);
    }
    return text;
}

/**
 * The entity recognizer is created before the database is
initialized so
 * initialization must be done after it is created.
 */
private void init() {
    Word personWord = dictionaryRepository.findWord("person",
PartOfSpeech.NOUN);
    personSense = personWord.getSense(PartOfSpeech.NOUN, 1);

    Word locationWord = dictionaryRepository.findWord("location",
PartOfSpeech.NOUN);
    locationSense = locationWord.getSense(PartOfSpeech.NOUN, 1);

    Word organizationWord =
dictionaryRepository.findWord("organization", PartOfSpeech.NOUN);
    organizationSense = organizationWord.getSense(PartOfSpeech.NOUN, 1);
}

private boolean initialized() {
    return !((personSense == null) || (locationSense == null) ||
(organizationSense == null));
}

private synchronized void ner(NLPText text) throws ParserException {
    List<TaggedWord> xtokens = new ArrayList<TaggedWord>();
    for (NLPText word : text.getLeaves()) {
        if (word.getParseTag() != null) {
            xtokens.add(new TaggedWord(word.getText(),
word.getParseTag().getText()));
        } else {
            xtokens.add(new TaggedWord(word.getText()));
        }
    }
    List<? extends HasWord> tokens = xtokens;

    for (CoreLabel label : classifier.testSentence(tokens)) {
        String entityType = label.get(AnswerAnnotation.class);
        String positionText = label.get(PositionAnnotation.class);
        int position = Integer.parseInt(positionText);
        NLPText word = text.getLeaves().get(position);

```

```

        if ("PERSON".equals(entityType)) {
            word.setDictionaryWord(personSense.getWord());
            word.setDictionaryWordSense(personSense);
            word.setNamedEntity(true);
        } else if ("ORGANIZATION".equals(entityType)) {
            word.setDictionaryWord(organizationSense.getWord());
            word.setDictionaryWordSense(organizationSense);
            word.setNamedEntity(true);
        } else if ("LOCATION".equals(entityType)) {
            word.setDictionaryWord(locationSense.getWord());
            word.setDictionaryWordSense(locationSense);
            word.setNamedEntity(true);
        }
    }
}

private static String prepareSentence(String sentence) {
    sentence = sentence.replaceAll(" *, *, *, *");
    sentence = sentence.replaceAll(" *\\". *, " \\. *");
    sentence = sentence.replaceAll(" *\\"! *, " \\\\"!");
    sentence = sentence.replaceAll(" *\\"? *, " \\\\"?");
    sentence = sentence.replaceAll(" *: *, " : ");
    sentence = sentence.replaceAll(" *; *, " ; ");
    sentence = sentence.replaceAll(" *\\( *, " \\( ");
    sentence = sentence.replaceAll(" *\\) *, " \\) ");
    sentence = sentence.replaceAll(" *\\[ *, " \\[ ");
    sentence = sentence.replaceAll(" *\\] *, " \\] ");
    sentence = sentence.replaceAll(" *\\{ *, " \\{ ");
    sentence = sentence.replaceAll(" *\\} *, " \\} ");
    sentence = sentence.replaceAll(" *< *, " < ");
    sentence = sentence.replaceAll(" *> *, " > ");
    sentence = removeBrackets(sentence);
    return sentence;
}

private static String removeBrackets(String sentence) {
    sentence = removeBrackets(sentence, "(", ")");
    sentence = removeBrackets(sentence, "[", "]");
    sentence = removeBrackets(sentence, "{", "}");
    sentence = removeBrackets(sentence, "<", ">");
    return sentence;
}

private static String removeBrackets(String sentence, String
openBracket, String closeBracket) {
    int openIdx = 0;
    while (openIdx != -1) {

```

```

openIdx = sentence.indexOf(openBracket);
if (openIdx == -1) {
    return sentence;
}
int closeIdx = sentence.indexOf(closeBracket, openIdx);
if (closeIdx == -1) {
    closeIdx = sentence.length() - 1;
}
String start = sentence.substring(0, openIdx);
String end = sentence.substring(closeIdx + 1, sentence.length());
sentence = start + end;
}
return sentence;
}
}

```

stemmer.java

```

/*
 * $Id: Stemmer.java,v 1.1 2008/08/03 09:45:52 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;

/**
 * @author ron
 */
@Component("stemmer")
public class Stemmer implements NLPPProcessor<NLPText> {
    private static final Logger log = Logger.getLogger(Stemmer.class);

    private final static PorterStemmer stemmer = new PorterStemmer();

    /**
     */
    public Stemmer() {
    }
}

```

```

@Override
public NLPText process(NLPText text) {
    NLPTextImpl workingText = (NLPTextImpl) text;
    if (text.is(GrammaticalStructureLevel.WORD)) {
        ((NLPTextImpl)
text).setLemma(stemmer.stripAffixes(text.getText()));
    } else {
        for (NLPText word : workingText.getLeaves()) {
            process(word);
        }
    }
    return text;
}

/*
 * author: Fotis Lazarinis (actually I translated from C to Java)
date: June
 * 1997 address: Psilovraxou 12, Agrinio, 30100 comments: Compile it,
import
 * the Porter class into you program and create an instance. Then use
the
 * stripAffixes method of this method which takes a String as input
and
 * returns the stem of this String again as a String.
 */

static class NewString {
    public String str;

    NewString() {
        str = "";
    }
}

static class PorterStemmer {

    public String stripAffixes(String str) {
        str = str.toLowerCase();
        str = Clean(str);

        if ((str != "") && (str.length() > 2)) {
            str = stripPrefixes(str);

            if (str != "") {
                str = stripSuffixes(str);
            }
        }
    }
}

```

```

}

return str;
} // stripAffixes

private String Clean(String str) {
    int last = str.length();

    Character ch = new Character(str.charAt(0));
    String temp = "";

    for (int i = 0; i < last; i++) {
        if (ch.isLetterOrDigit(str.charAt(i))) {
            temp += str.charAt(i);
        }
    }

    return temp;
} // clean

private boolean hasSuffix(String word, String suffix, NewString
stem) {

    String tmp = "";

    if (word.length() <= suffix.length()) {
        return false;
    }
    if (suffix.length() > 1) {
        if (word.charAt(word.length() - 2) != suffix.charAt(suffix.length() - 2)) {
            return false;
        }
    }

    stem.str = "";

    for (int i = 0; i < word.length() - suffix.length(); i++) {
        stem.str += word.charAt(i);
    }
    tmp = stem.str;

    for (int i = 0; i < suffix.length(); i++) {
        tmp += suffix.charAt(i);
    }
}

```

```

if (tmp.compareTo(word) == 0) {
    return true;
} else {
    return false;
}

private boolean vowel(char ch, char prev) {
    switch (ch) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            return true;
        case 'y': {
            switch (prev) {
                case 'a':
                case 'e':
                case 'i':
                case 'o':
                case 'u':
                    return false;
                default:
                    return true;
            }
        }
        default:
            return false;
    }
}

private int measure(String stem) {

    int i = 0, count = 0;
    int length = stem.length();

    while (i < length) {
        for (; i < length; i++) {
            if (i > 0) {
                if (vowel(stem.charAt(i), stem.charAt(i - 1))) {
                    break;
                }
            } else {

```

```

        if (vowel(stem.charAt(i), 'a')) {
            break;
        }
    }

    for (i++; i < length; i++) {
        if (i > 0) {
            if (!vowel(stem.charAt(i), stem.charAt(i - 1))) {
                break;
            }
        } else {
            if (!vowel(stem.charAt(i), '?')) {
                break;
            }
        }
        if (i < length) {
            count++;
            i++;
        }
    } // while

    return (count);
}

private boolean containsVowel(String word) {

    for (int i = 0; i < word.length(); i++) {
        if (i > 0) {
            if (vowel(word.charAt(i), word.charAt(i - 1))) {
                return true;
            }
        } else {
            if (vowel(word.charAt(0), 'a')) {
                return true;
            }
        }
    }

    return false;
}

private boolean cvc(String str) {
    int length = str.length();

    if (length < 3) {
        return false;
    }
}

if (vowel(stem.charAt(i), 'a')) {
    break;
}
}

for (i++; i < length; i++) {
    if (i > 0) {
        if (!vowel(stem.charAt(i), stem.charAt(i - 1))) {
            break;
        }
    } else {
        if (!vowel(stem.charAt(i), '?')) {
            break;
        }
    }
    if (i < length) {
        count++;
        i++;
    }
} // while

return (count);
}

private boolean containsVowel(String word) {

    for (int i = 0; i < word.length(); i++) {
        if (i > 0) {
            if (vowel(word.charAt(i), word.charAt(i - 1))) {
                return true;
            }
        } else {
            if (vowel(word.charAt(0), 'a')) {
                return true;
            }
        }
    }

    return false;
}

private boolean cvc(String str) {
    int length = str.length();

    if (length < 3) {
        return false;
    }
}

if ((!vowel(str.charAt(length - 1), str.charAt(length - 2)))
    && (str.charAt(length - 1) != 'w') && (str.charAt(length - 1) != 'x'))
    && (str.charAt(length - 1) != 'y')
    && (vowel(str.charAt(length - 2), str.charAt(length - 3)))) {
    if (length == 3) {
        if (!vowel(str.charAt(0), '?')) {
            return true;
        } else {
            return false;
        }
    } else {
        if (!vowel(str.charAt(length - 3), str.charAt(length - 4))) {
            return true;
        } else {
            return false;
        }
    }
}

return false;
}

private String step1(String str) {
    NewString stem = new NewString();

    if (str.charAt(str.length() - 1) == 's') {
        if ((hasSuffix(str, "sses", stem)) || (hasSuffix(str, "ies",
        stem))) {
            String tmp = "";
            for (int i = 0; i < str.length() - 2; i++) {
                tmp += str.charAt(i);
            }
            str = tmp;
        } else {
            if ((str.length() == 1) && (str.charAt(str.length() - 1) == 's')) {
                str = "";
                return str;
            }
            if (str.charAt(str.length() - 2) != 's') {
                String tmp = "";

```

```

        if (cvc(str)) {
            str += "e";
        }
    }
}

if (hasSuffix(str, "y", stem)) {
    if (containsVowel(stem.str)) {
        String tmp = "";
        for (int i = 0; i < str.length() - 1; i++) {
            tmp += str.charAt(i);
        }
        str = tmp + "i";
    }
}
return str;
}

private String step2(String str) {

    String[][] suffixes = { { "ational", "ate" }, { "tional", "tion" },
"enci", "ence" },
        { "anci", "ance" }, { "izer", "ize" }, { "iser", "ize" },
"abli", "able" },
        { "alli", "al" }, { "entli", "ent" }, { "eli", "e" }, { "ousli",
"ous" },
        { "ization", "ize" }, { "isation", "ize" }, { "ation", "ate" },
{ "ator", "ate" }, { "alism", "al" }, { "iveness", "ive" },
{ "fulness", "ful" }, { "ousness", "ous" }, { "aliti", "al" },
{ "iviti", "ive" }, { "biliti", "ble" } };
    NewString stem = new NewString();

    for (int index = 0; index < suffixes.length; index++) {
        if (hasSuffix(str, suffixes[index][0], stem)) {
            if (measure(stem.str) > 0) {
                str = stem.str + suffixes[index][1];
                return str;
            }
        }
    }
}

return str;
}

```

```

private String step3(String str) {
    String[][] suffixes = { { "icate", "ic" }, { "ative", "" },
    { "alize", "al" }, { "alise", "al" }, { "iciti", "ic" }, { "ical", "ic" }, { "ful",
    "" },
    { "ness", "" } };
    NewString stem = new NewString();

    for (int index = 0; index < suffixes.length; index++) {
        if (hasSuffix(str, suffixes[index][0], stem)) {
            if (measure(stem.str) > 0) {
                str = stem.str + suffixes[index][1];
                return str;
            }
        }
    }
    return str;
}

private String step4(String str) {

    String[] suffixes = { "al", "ance", "ence", "er", "ic", "able",
    "ible", "ant", "ement",
    "ment", "ent", "sion", "tion", "ou", "ism", "ate", "iti", "ous",
    "ive", "ize",
    "ise" };

    NewString stem = new NewString();

    for (int index = 0; index < suffixes.length; index++) {
        if (hasSuffix(str, suffixes[index], stem)) {

            if (measure(stem.str) > 1) {
                str = stem.str;
                return str;
            }
        }
    }
    return str;
}

private String step5(String str) {

    if (str.charAt(str.length() - 1) == 'e') {
        if (measure(str) > 1) /* */
            * measure(str)==measure(stem) if ends

```

```

            * in vowel
            */
        String tmp = "";
        for (int i = 0; i < str.length() - 1; i++) {
            tmp += str.charAt(i);
        }
        str = tmp;
    } else if (measure(str) == 1) {
        String stem = "";
        for (int i = 0; i < str.length() - 1; i++) {
            stem += str.charAt(i);
        }

        if (!cvc(stem)) {
            str = stem;
        }
    }

    if (str.length() == 1) {
        return str;
    }
    if ((str.charAt(str.length() - 1) == 'l') &&
    (str.charAt(str.length() - 2) == 'l'))
        && (measure(str) > 1)) {
        if (measure(str) > 1) /* */
            * measure(str)==measure(stem) if ends
            * in vowel
            */
        String tmp = "";
        for (int i = 0; i < str.length() - 1; i++) {
            tmp += str.charAt(i);
        }
        str = tmp;
    }
    return str;
}

private String stripPrefixes(String str) {

    String[] prefixes = { "kilo", "micro", "milli", "intra", "ultra",
    "mega", "nano",
    "pico", "pseudo" };

    int last = prefixes.length;
    for (int i = 0; i < last; i++) {

```

```

if (str.startsWith(prefixes[i])) {
    String temp = "";
    for (int j = 0; j < str.length() - prefixes[i].length(); j++) {
        temp += str.charAt(j + prefixes[i].length());
    }
    return temp;
}

return str;
}

private String stripSuffixes(String str) {

    str = step1(str);
    if (str.length() >= 1) {
        str = step2(str);
    }
    if (str.length() >= 1) {
        str = step3(str);
    }
    if (str.length() >= 1) {
        str = step4(str);
    }
    if (str.length() >= 1) {
        str = step5(str);
    }

    return str;
}
} // class
}

```

step.java

```

/*
 * $Id: Step.java,v 1.1 2008/10/09 22:55:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

import edu.harvard.fas.rregan.requel.project.TextEntity;

/**

```

```

 * A Step in a Scenario, which may be a Scenario itself.
 *
 * @author ron
 */
public interface Step extends TextEntity, Comparable<Step> {

    /**
     * @return The type/disposition of a scenario or step, such as a primary,
     * alternative, or exceptional case.
     */
    public ScenarioType getType();

    /**
     *
     * @param scenarioType - the new type
     */
    public void setType(ScenarioType scenarioType);

    /**
     *
     * @return The set of scenarios that have this scenario as a step.
     */
    public Set<Scenario> getUsingScenarios();
}

```

stepimpl.java

```

/*
 * $Id: StepImpl.java,v 1.8 2009/02/11 11:00:15 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;

```

```

import javax.persistence.FetchType;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.ScenarioType;
import edu.harvard.fas.rregan.requel.project.Step;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "scenarios", uniqueConstraints =
{ @UniqueConstraint(columnNames =
"projectordomain_id", "name" ) })
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "type", discriminatorType =
DiscriminatorType.STRING, length = 255)
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.project.Step")
@XmlRootElement(name = "step", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "step", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")

```

```

public class StepImpl extends AbstractTextEntity implements Step {
    static final long serialVersionUID = 0L;

    private ScenarioType scenarioType = ScenarioType.Primary;
    private Set<Scenario> usedBy = new TreeSet<Scenario>();
    private String type;

    /**
     * @param projectOrDomain
     * @param createdBy
     * @param name
     * @param text
     * @param scenarioType
     */
    public StepImpl(ProjectOrDomain projectOrDomain, User createdBy,
String name, String text,
    ScenarioType scenarioType) {
        this(Step.class.getName(), projectOrDomain, createdBy, name, text,
scenarioType);
    }

    protected StepImpl(String type, ProjectOrDomain projectOrDomain, User
createdBy, String name,
    String text, ScenarioType scenarioType) {
        super(projectOrDomain, createdBy, name, text);
        setInstanceType(type);
        setType(scenarioType);
    }

    protected StepImpl() {
        // for hibernate
    }

    @Override
    @Column(nullable = false, unique = false)
    @NotEmpty(message = "a unique name is required.")
    @XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
    public String getName() {
        return super.getName();
    }

    // hack for JAXB to set the name, for some reason it won't use the
    inherited
    // method.
    @Override
    public void setName(String name) {

```

```

super.setName(name);
}

// access to the desriminator
@Column(name = "type", insertable = false, updatable = false)
protected String getInstanceType() {
    return type;
}

protected void setInstanceType(String type) {
    this.type = type;
}

@Override
@XmlTransient
@ManyToMany(targetEntity = ScenarioImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY,
mappedBy = "steps")
@JoinTable(name = "scenario_steps", joinColumns = { @JoinColumn(name
= "step_id") }, inverseJoinColumns = { @JoinColumn(name =
"scenario_id") })
@Sort(type = SortType.NATURAL)
public Set<Scenario> getUsingScenarios() {
    return usedBy;
}

protected void setUsingScenarios(Set<Scenario> usedBy) {
    this.usedBy = usedBy;
}

@Override
@Enumerated(EnumType.STRING)
@XmlAttribute(name = "scenarioType")
@Column(name = "scenario_type")
@XmlJavaTypeAdapter(ScenarioTypeAdapter.class)
public ScenarioType getType() {
    return scenarioType;
}

public void setType(ScenarioType scenarioType) {
    this.scenarioType = scenarioType;
}

/***
 * @see
edu.harvard.fas.rregan.requell.project.ProjectOrDomainEntity#getXmlId()
 */
@Transient
@XmlID
@XmlAttribute(name = "id")
@Override
public String getXmlId() {
    // TODO Auto-generated method stub
    return "SCN_" + getId();
}

@Override
@Transient
@XmlTransient
public String getDescription() {
    // TODO Auto-generated method stub
    return "Step: " + getName();
}

@Override
public int compareTo(Step o) {
    return getName().compareToIgnoreCase(o.getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
supplied.
 * @param parent
 * @see UnmarshallerListener
 */
@Override
public void afterUnmarshal(UserRepository userRepository, User
defaultCreatedByUser,
Object parent) {
    if (parent instanceof ProjectOrDomainEntity) {
        super.afterUnmarshal(userRepository, defaultCreatedByUser, null);
    } else if (parent instanceof ProjectOrDomain) {
        super.afterUnmarshal(userRepository, defaultCreatedByUser, parent);
    } else {
        throw new RuntimeException("Unexpected parent type "

```

```

+ parent.getSimpleClassName() + " for " +
getClassName().getSimpleClassName()
+ " named: " + getName());
}
}

/**
 * This class is used by JAXB to convert the ScenarioType of a
Scenario into
 * a string for an attribute in the xml file and the reverse when
 * unmarshalling.
 *
 * @author ron
 */
@XmlTransient
public static class ScenarioTypeAdapter extends XmlAdapter<String,
ScenarioType> {

    @Override
    public ScenarioType unmarshal(String typeString) throws Exception {
        return ScenarioType.valueOf(typeString);
    }

    @Override
    public String marshal(ScenarioType type) throws Exception {
        return type.toString();
    }
}
}

```

storiesTable.java

```

/*
 * $Id: StoriesTable.java,v 1.7 2009/01/27 09:30:19 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.Insets;

```

```

import nextapp.echo2.app.Row;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
e;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;
*/

```

```

 * A component to add to Panels of story container entity editors to
enable
 * editing of the Stories of the entity.
 *
 * @author ron
 */
public class StoriesTable extends AbstractRequelNavigatorTable {
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(StoriesTable.class, new
StoriesTableManipulator());
    }

    /**
     * The name to use in the properties file of the panel that includes
the
     * StoriesTable to define the label of the Stories field. If the
property is
     * undefined the panel should use a sensible default such as
"Stories".
    */
    public static final String PROP_LABEL_STORIES = "Stories.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the view button in the story edit table column. If the property
is
     * undefined "View" is used.
    */
    public static final String PROP_VIEW_STORY_BUTTON_LABEL =
"ViewStory.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the remove button in the story edit table column. If the
property is
     * undefined "Remove" is used.
    */
    public static final String PROP_REMOVE_STORY_BUTTON_LABEL =
"RemoveStory.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the edit button in the story edit table column. If the property
is
     * undefined "Edit" is used.
    */
    public static final String PROP_EDIT_STORY_BUTTON_LABEL =
>EditStory.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the new story button under the Stories table. If the property
is
     * undefined "New" is used.
    */
    public static final String PROP_NEW_STORY_BUTTON_LABEL =
>NewStory.Label";

    /**
     * The name to use in the containing panels properties file to set
the label
     * of the find story button under the Stories table. If the property
is
     * undefined "Find" is used.
    */
    public static final String PROP_FIND_STORY_BUTTON_LABEL =
"FindStory.Label";

    private StoryContainer storyContainer;
    private final NavigatorTable table;
    private final NavigatorButton openStoryEditorButton;
    private final NavigatorButton openStoriesSelectorButton;
    private final ProjectCommandFactory projectCommandFactory;
    private final CommandHandler commandHandler;
    private AddStoryToStoryContainerController addStoryController;
    private RemoveStoryFromStoryContainerController removeStoryController;

    /**
     * @param editMode
     * @param resourceBundleHelper
     * @param projectCommandFactory -
     *          passed to the add/remove story to story container
controller
     * @param commandHandler -
     *          passed to the add/remove story to story container
controller
    */
}

```

```

public StoriesTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper,
ProjectCommandFactory projectCommandFactory, CommandHandler
commandHandler) {
super(editMode, resourceBundleHelper);
this.projectCommandFactory = projectCommandFactory;
this.commandHandler = commandHandler;
ColumnLayoutData layoutData = new ColumnLayoutData();
layoutData.setAlignment(Alignment.ALIGN_CENTER);
table = new NavigatorTable(getTableConfig());
table.setLayoutData(layoutData);
add(table);

Row buttons = new Row();
buttons.setLayoutData(layoutData);

String buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_NEW_STORY_BUTTON_LABEL, "New");
openStoryEditorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openStoryEditorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
openStoryEditorButton.setVisible(false);
buttons.add(openStoryEditorButton);

buttonLabel =
getResourceBundleHelpergetLocale().getString(PROP_FIND_STORY_BUTTON_
LABEL,
"Find");
openStoriesSelectorButton = new NavigatorButton(buttonLabel,
getEventDispatcher());
openStoriesSelectorButton.setStyleName(Panel.STYLE_NAME_DEFAULT);
openStoriesSelectorButton.setVisible(false);
buttons.add(openStoriesSelectorButton);

add(buttons);
}

protected StoryContainer getStoryContainer() {
return storyContainer;
}

protected void setStoryContainer(StoryContainer storyContainer) {
this.storyContainer = storyContainer;

if (storyContainer != null) {
table.setModel(new NavigatorTableModel((Collection)
storyContainer.getStories()));
}
}

```

```

if (!isReadOnlyMode()) {
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
PanelActionType.Editor,
getStoryContainer(), Story.class, null,
WorkflowDisposition.NewFlow);
openStoryEditorButton.setEventToFire(openEditorEvent);
openStoryEditorButton.setVisible(true);

ProjectOrDomain pod = null;
if (getStoryContainer() instanceof ProjectOrDomain) {
pod = (ProjectOrDomain) getStoryContainer();
} else if (getStoryContainer() instanceof ProjectOrDomainEntity) {
pod = ((ProjectOrDomainEntity)
getStoryContainer()).getProjectOrDomain();
}
if (pod != null) {
NavigationEvent openStoriesSelectorEvent = new
OpenPanelEvent(this,
PanelActionType.Selector, pod, Project.class,
ProjectManagementPanelNames.PROJECT_STORY_SELECTOR_PANEL_NAME,
WorkflowDisposition.ContinueFlow);

openStoriesSelectorButton.setEventToFire(openStoriesSelectorEvent
);
openStoriesSelectorButton.setVisible(true);
} else {
openStoriesSelectorButton.setVisible(false);

// use the the story table (this) as the destination because it
// is used as the source to the open panel events created above
if (addStoryController != null) {
getEventDispatcher().removeEventTypeActionListener(SelectEntityEv
ent.class,
addStoryController, this);
}
addStoryController = new
AddStoryToStoryContainerController(getEventDispatcher(),
projectCommandFactory, commandHandler, storyContainer);
getEventDispatcher().addEventTypeActionListener(SelectEntityEvent.
class,
addStoryController, this);

// use the the story table (this) as the destination because it
// is used as the source to the open panel events created above
if (removeStoryController != null) {
getEventDispatcher().removeEventTypeActionListener(

```

```

        RemoveStoryFromStoryContainerEvent.class,
removeStoryController, this);
    }
    removeStoryController = new
RemoveStoryFromStoryContainerController(
    getEventDispatcher(), projectCommandFactory, commandHandler,
storyContainer);
    getEventDispatcher().addEventTypeActionListener(
        RemoveStoryFromStoryContainerEvent.class, removeStoryController,
this);

}
} else {
    table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    openStoryEditorButton.setVisible(false);
    openStoriesSelectorButton.setVisible(false);
}
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Row buttonsContainer = new Row();
                RowLayoutData buttonLayout = new RowLayoutData();
                buttonLayout.setAlignment(Alignment.ALIGN_CENTER);
                buttonLayout.setInsets(new Insets(5, 0));

                Story story = (Story) model.getBackingObject(row);
                String buttonLabel = null;
                if (isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_VIEW_STORY_BUTTON_LABEL, "View");
                } else {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_EDIT_STORY_BUTTON_LABEL, "Edit");
                }
                NavigationEvent openEditorEvent = new
OpenPanelEvent(StoriesTable.this,
                Panel ActionType.Editor, story, story.getClass(), null,
                WorkflowDisposition.NewFlow);
                NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,

```

```

        getEventDispatcher(), openEditorEvent);
                openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                openEditorButton.setLayoutData(buttonLayout);
                buttonsContainer.add(openEditorButton);

                if (!isReadOnlyMode()) {
                    buttonLabel = getResourceBundleHelper(getLocale()).getString(
                        PROP_REMOVE_STORY_BUTTON_LABEL, "Remove");
                    NavigationEvent removeStoryEvent = new
RemoveStoryFromStoryContainerEvent(
                        StoriesTable.this, story, storyContainer, StoriesTable.this);
                    NavigatorButton removeStoryButton = new
NavigatorButton(buttonLabel,
                        getEventDispatcher(), removeStoryEvent);
                    openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
                    openEditorButton.setLayoutData(buttonLayout);
                    buttonsContainer.add(removeStoryButton);
                }
                return buttonsContainer;
            }
        }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Story story = (Story) model.getBackingObject(row);
                return story.getName();
            }
        }));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
        new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                Story story = (Story) model.getBackingObject(row);
                return story.getCreatedBy().getUsername();
            }
        }));

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
        new NavigatorTableCellValueFactory() {
            @Override

```

```

    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
    Story story = (Story) model.getBackingObject(row);
    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
    return formatter.format(story.getDateCreated());
}
});

return tableConfig;
}

private static class StoriesTableManipulator extends
AbstractComponentManipulator {

protected StoriesTableManipulator() {
    super();
}

@Override
public Object getModel(Component component) {
    return getValue(component, StoryContainer.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getStoryContainer());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setStoryContainer((StoryContainer) value);
}

private StoriesTable getComponent(Component component) {
    return (StoriesTable) component;
}
}

```

```

    }
}
```

story.java

```

/*
 * $Id: Story.java,v 1.5 2008/09/06 09:31:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Set;

/**
 * @author ron
 */
public interface Story extends TextEntity, GoalContainer,
ActorContainer, Comparable<Story> {

/**
 * Return the project entities that have a direct reference to this
story.
 *
 * @return
 */
public Set<StoryContainer> getReferers();

/**
 * @return The type of story: Success or Exceptional
 */
public StoryType getStoryType();

/**
 * Set the type of story.
 *
 * @param storyType
 */
public void setStoryType(StoryType storyType);
}
```

story2storyimpladapter.java

```

/*
```

```

 * $Id: Story2StoryImplAdapter.java,v 1.1 2008/09/06 09:31:57 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl;

import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import edu.harvard.fas.rregan.requel.project.Story;

/**
 * Adapter for JAXB to convert interface Story to class StoryImpl and
back.
 *
 * @author ron
 */
@XmlTransient
public class Story2StoryImplAdapter extends XmlAdapter<StoryImpl,
Story> {

@Override
public StoryImpl marshal(Story story) throws Exception {
    return (StoryImpl) story;
}

@Override
public Story unmarshal(StoryImpl story) throws Exception {
    return story;
}
}

```

storyassistant.java

```

/*
 * $Id: StoryAssistant.java,v 1.4 2009/01/23 09:54:24 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses stories and adds annotations with suggestions.

```

```

 *
 * @author ron
 */
public class StoryAssistant extends TextEntityAssistant {

/**
 * @param lexicalAssistant -
 *         assistant for analyzing text for spelling, terms and
other
 *         word oriented analysis.
 * @param updatedEntityNotifier -
 *         after an entity is analyzed it is passed to the
notifier to
 *         tell the UI components that reference the entity to
refresh
 * @param assistantUser -
 *         the user to use as the creator of the annotation
entities.
 */
public StoryAssistant(LexicalAssistant lexicalAssistant, User
assistantUser) {
    super(StoryAssistant.class.getName(), lexicalAssistant,
assistantUser);
}
}

```

storycontainer.java

```

/*
 * $Id: StoryContainer.java,v 1.1 2008/09/03 02:56:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

import java.util.Comparator;
import java.util.Set;

import edu.harvard.fas.rregan.requel.CreatedEntity;
import edu.harvard.fas.rregan.requel.Describable;

/**
 * A thing that can contain/refer to stories.
 *
 * @author ron
 */
public interface StoryContainer extends Describable, CreatedEntity {

```

```

/**
 * The stories referenced.
 */
@return
*/
public Set<Story> getStories();

/**
 * Compare the objects that contain Storys by the description.
 */
public static final Comparator<StoryContainer> COMPARATOR = new
StoryContainerComparator();

/**
 * A Comparator for collections of Story containers.
 */
public static class StoryContainerComparator implements
Comparator<StoryContainer> {
    @Override
    public int compare(StoryContainer o1, StoryContainer o2) {
        return o1.getDescription().compareTo(o2.getDescription());
    }
}
}

```

storycontainerstable.java

```

/*
 * $Id: StoryContainersTable.java,v 1.5 2009/01/08 06:48:44 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;
import java.util.Collections;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.layout.ColumnLayoutData;
import nextapp.echo2.app.layout.RowLayoutData;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;

```

```

import edu.harvard.fas.rregan.requel.ui.AbstractRequelNavigatorTable;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTable;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import edu.harvard.fas.rregan.uiframework.panel.Panel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.AbstractC
omponentManipulator;
import
edu.harvard.fas.rregan.uiframework.panel.editor.manipulators.Component
Manipulators;

/**
 * A component to add to Panels of Story container entity editors to
enable
 * editing of the Stories of the entity.
 *
 * @author ron
 */
public class StoryContainersTable extends AbstractRequelNavigatorTable
{
    static final long serialVersionUID = 0L;

    static {
        ComponentManipulators.setManipulator(StoryContainersTable.class,
            new StoryContainersTableManipulator());
    }
}

```

```

/**
 * The name to use in the properties file of the panel that includes
the
 * StoryContainersTable to define the label of the Story containers
field.
 * If the property is undefined the panel should use a sensible
default such
 * as "Story Referers".
 */
public static final String PROP_LABEL_STORY_CONTAINERS =
"StoryContainers.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the view button in the Story containers edit table column. If
the
 * property is undefined "View" is used.
 */
public static final String PROP_VIEW_STORY_CONTAINER_BUTTON_LABEL =
"ViewStoryContainer.Label";

/**
 * The name to use in the containing panels properties file to set
the label
 * of the edit button in the Story container edit table column. If
the
 * property is undefined "Edit" is used.
 */
public static final String PROP_EDIT_STORY_CONTAINER_BUTTON_LABEL =
>EditStoryContainer.Label";

private Story story;
private final NavigatorTable table;

/**
 * @param editMode
 * @param resourceBundleHelper
 */
public StoryContainersTable(EditMode editMode, ResourceBundleHelper
resourceBundleHelper) {
    super(editMode, resourceBundleHelper);
    ColumnLayoutData layoutData = new ColumnLayoutData();
    layoutData.setAlignment(Alignment.ALIGN_CENTER);
    table = new NavigatorTable(getTableConfig());
    table.setLayoutData(layoutData);
}

```

```

        add(table);
    }

protected Story getStory() {
    return story;
}

protected void setStory(Story story) {
    this.story = story;
    if (story != null) {
        table.setModel(new NavigatorTableModel((Collection)
story.getReferers()));
    } else {
        table.setModel(new NavigatorTableModel(Collections.EMPTY_SET));
    }
}

private NavigatorTableConfig getTableConfig() {
    NavigatorTableConfig tableConfig = new NavigatorTableConfig();
    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        StoryContainer storyContainer = (StoryContainer) model
            .getBackingObject(row);
        String buttonLabel = null;
        if (isReadOnlyMode()) {
            buttonLabel = getResourceBundleHelpergetLocale().getString(
                PROP_VIEW_STORY_CONTAINER_BUTTON_LABEL, "View");
        } else {
            buttonLabel = getResourceBundleHelpergetLocale().getString(
                PROP_EDIT_STORY_CONTAINER_BUTTON_LABEL, "Edit");
        }
        NavigationEvent openEditorEvent = new OpenPanelEvent(this,
            PanelActionType.Editor, storyContainer,
            storyContainer.getClass(),
            null, WorkflowDisposition.NewFlow);
        NavigatorButton openEditorButton = new
        NavigatorButton(buttonLabel,
            getEventDispatcher(), openEditorEvent);
        openEditorButton.setStyleName(Panel.STYLE_NAME_PLAIN);
        RowLayoutData rld = new RowLayoutData();
        rld.setAlignment(Alignment.ALIGN_CENTER);
        openEditorButton.setLayoutData(rld);
        return openEditorButton;
    }
})
}

```

```

        }
    });

tableConfig.addColumnConfig(new
NavigatorTableColumnConfig("Description",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            StoryContainer StoryContainer = (StoryContainer) model
                .getBackingObject(row);
            return StoryContainer.getDescription();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            StoryContainer StoryContainer = (StoryContainer) model
                .getBackingObject(row);
            return StoryContainer.getCreatedBy().getUsername();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
int row) {
            StoryContainer StoryContainer = (StoryContainer) model
                .getBackingObject(row);
            DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            return formatter.format(StoryContainer.getDateCreated());
        }
    }));
}

return tableConfig;
}

private static class StoryContainersTableManipulator extends
AbstractComponentManipulator {

protected StoryContainersTableManipulator() {

```

```

        super();
    }

@Override
public Object getModel(Component component) {
    return getValue(component, Story.class);
}

@Override
public void setModel(Component component, Object valueModel) {
    setValue(component, valueModel);
}

@Override
public void addListenerToDetectChangesToInput(EditMode editMode,
Component component) {
    // nothing to do.
}

@Override
public <T> T getValue(Component component, Class<T> type) {
    return type.cast(getComponent(component).getStory());
}

@Override
public void setValue(Component component, Object value) {
    getComponent(component).setStory((Story) value);
}

private StoryContainersTable getComponent(Component component) {
    return (StoryContainersTable) component;
}
}
}

```

storyeditorpanel.java

```

/*
 * $Id: StoryEditorPanel.java,v 1.19 2009/03/23 11:02:55 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;
import java.util.Set;
import java.util.TreeSet;

```

```

import nextapp.echo2.app.Button;
import nextapp.echo2.app.SelectField;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.StoryType;
import edu.harvard.fas.rregan.requel.project.command.CopyStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.DeleteStoryCommand;
import edu.harvard.fas.rregan.requel.project.command.EditStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;

/**
 * @author ron

```

```

/*
public class StoryEditorPanel extends AbstractRequelProjectEditorPanel
{
    private static final Logger log =
Logger.getLogger(StoryEditorPanel.class);

    static final long serialVersionUID = 0L;

    /**
     * The name to use in the StoryEditorPanel.properties file to set the
label
     * of the name field. If the property is undefined "Name" is used.
     */
    public static final String PROP_LABEL_NAME = "Name.Label";

    /**
     * The name to use in the StoryEditorPanel.properties file to set the
label
     * of the story type field. If the property is undefined "Story Type"
is
     * used.
     */
    public static final String PROP_LABEL_STORY_TYPE = "StoryType.Label";

    /**
     * The name to use in the StoryEditorPanel.properties file to set the
label
     * of the text field. If the property is undefined "Text" is used.
     */
    public static final String PROP_LABEL_TEXT = "Text.Label";

    private UpdateListener updateListener;
    private Button copyButton;

    // this is set by the DeleteListener so that the UpdateListener can
ignore
    // events between when the object was deleted and the panel goes
away.
    private boolean deleted;

    /**
     * @param commandHandler
     * @param projectCommandFactory
     * @param projectRepository
     */
    public StoryEditorPanel(CommandHandler commandHandler,

```

```

    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    this(StoryEditorPanel.class.getName(), commandHandler,
projectCommandFactory,
    projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public StoryEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    super(resourceBundleName, Story.class, commandHandler,
projectCommandFactory,
    projectRepository);
}

/**
 * If the editor is editing an existing Story the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Story: {0}"<br>
 * Valid variables are:<br>
 * {0} - Story name<br>
 * {1} - project/domain name<br>
 * For new Story it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New Story"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    if (getStory() != null) {

```

```

        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Story: {0}"));
        return MessageFormat.format(msgPattern, getStory().getName(),
getProjectOrDomain()
    .getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"New Story"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    Story story = getStory();
    if (story != null) {
        addInput(EditStoryCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
TextField(),
            new StringDocumentEx(story.getName()));
        addInput("storyType", PROP_LABEL_STORY_TYPE, "Type", new
SelectField(),
            new CombinedListModel(getStoryTypeNames(),
story.getStoryType().toString(),
            true));
        addInput(EditStoryCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text", new
TextArea(),
            new StringDocumentEx(story.getText()));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABELGLOSSARYTERM,
            "Glossary Terms", new GlossaryTermsTable(this,
                getResourceBundleHelper(getLocale()), story));
        addMultiRowInput("goals", GoalsTable.PROP_LABELGOALS, "Goals", new
GoalsTable(this,
            getResourceBundleHelper(getLocale()), getProjectCommandFactory(),
            getCommandHandler(), story));
        addMultiRowInput("actors", ActorsTable.PROP_LABELACTORS, "Actors",
new ActorsTable(
            this, getResourceBundleHelper(getLocale()),
getProjectCommandFactory(),
            getCommandHandler(), story));
        addMultiRowInput("storyContainers",
StoryContainersTable.PROP_LABELSTORYCONTAINERS,

```

```

    "Referring Entities", new StoryContainersTable(this,
        getResourceBundleHelper(getLocale()), story);
    addMultiRowInput("annotations",
        AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
        new AnnotationsTable(this, getResourceBundleHelper(getLocale())),
        story);
    copyButton = addActionButton(new
        Button(getResourceBundleHelper(getLocale()).getString(
            PROP_LABEL_COPY_BUTTON, "Copy")));
    copyButton.addActionListener(new CopyListener(this));
    copyButton.setEnabled(!isReadOnlyMode());
} else {
    addInput(EditStoryCommand.FIELD_NAME, PROP_LABEL_NAME, "Name", new
    TextField(),
        new StringDocumentEx());
    addInput("storyType", PROP_LABEL_STORY_TYPE, "Type", new
    SelectField(),
        new CombinedListModel(getStoryTypeNames(), "", true));
    addInput(EditStoryCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text", new
    TextArea(),
        new StringDocumentEx());
    addMultiRowInput("glossaryTerms",
        GlossaryTermsTable.PROP_LABEL_GLOSSARY_TERM,
        "Glossary Terms", new GlossaryTermsTable(this,
            getResourceBundleHelper(getLocale()), story));
    addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals", new
    GoalsTable(this,
        getResourceBundleHelper(getLocale()), getProjectCommandFactory(),
        getCommandHandler(), null));
    addMultiRowInput("actors", ActorsTable.PROP_LABEL_ACTORS, "Actors",
        new ActorsTable(
            this, getResourceBundleHelper(getLocale()),
            getProjectCommandFactory(),
            getCommandHandler(), null),
        addMultiRowInput("storyContainers",
            StoryContainersTable.PROP_LABEL_STORY_CONTAINERS,
            "Referring Entities", new StoryContainersTable(this,
                getResourceBundleHelper(getLocale()), null));
        addMultiRowInput("annotations",
            AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
            new AnnotationsTable(this, getResourceBundleHelper(getLocale())),
            null);
    }

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        EditStoryCommand command =
getProjectCommandFactory().newEditStoryCommand();
        command.setStory(getStory());
        command.setStoryContainer(getStoryContainer());
        command.setEditedBy(getCurrentUser());
        command.setName(getInputValue(EditStoryCommand.FIELD_NAME,
String.class));

        command.setText(getInputValue(EditStoryCommand.FIELD_TEXT,
String.class));
        command.setStoryTypeName(getInputValue("storyType", String.class));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {

```

```

        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
            updateListener);
        // TODO: remove other listeners?
    }
    getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
        command.getStory()));
} catch (EntityException e) {
    if (e.isStaleEntity()) {
        // TODO: compare the original values before the user edited
        // to the current revisions values and if they are the same
        // then update the new revision with the user's changes and
        // continue, otherwise show the new changed value vs. the users
        // new values.
        String newName = getInputValue(EditStoryCommand.FIELD_NAME,
String.class);
        String newText = getInputValue(EditStoryCommand.FIELD_TEXT,
String.class);
        Story newStory = getProjectRepository().get(getStory());

        setTargetObject(newStory);
        if (!newName.equals(newStory.getName()) || !
newText.equals(newStory.getText())) {
            setGeneralMessage("The story was changed by another user and the
value conflicts with your input.");
            if (!newName.equals(newStory.getName())) {
                setValidationMessage(EditStoryCommand.FIELD_NAME, "Your input '" +
newName
                    + "'");
                setInputValue(EditStoryCommand.FIELD_NAME, newStory.getName());
            }
            if (!newText.equals(newStory.getText())) {
                setValidationMessage(EditStoryCommand.FIELD_TEXT, "Your input '" +
newText
                    + "'");
                setInputValue(EditStoryCommand.FIELD_TEXT, newStory.getText());
            }
        } else {
            getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
newStory));
        }
    } else if ((e.getEntityPropertyNames() != null)
        && (e.getEntityPropertyNames().length > 0)) {
        for (String propertyName : e.getEntityPropertyNames()) {
            setValidationMessage(propertyName, e.getMessage());
        }
    } else if ((e.getCause() != null) && (e.getCause() instanceof
        InvalidStateException)) {
        InvalidStateException ise = (InvalidStateException) e.getCause();
        for (InvalidValue invalidValue : ise.getInvalidValues()) {
            String propertyName = invalidValue.getPropertyName();
            setValidationMessage(propertyName, invalidValue.getMessage());
        }
    } else {
        setGeneralMessage(e.toString());
    }
} catch (Exception e) {
    log.error("could not save the story: " + e, e);
    setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
try {
    DeleteStoryCommand deleteStoryCommand = getProjectCommandFactory()
        .newDeleteStoryCommand();
    deleteStoryCommand.setEditedBy(getCurrentUser());
    deleteStoryCommand.setStory(getStory());
    deleteStoryCommand =
getCommandHandler().execute(deleteStoryCommand);
    deleted = true;
    getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getStory()));
} catch (Exception e) {
    setGeneralMessage("Could not delete entity: " + e);
}
}

private Set<String> getStoryTypeNames() {
Set<String> storyTypeNames = new TreeSet<String>();
for (StoryType storyType : StoryType.values()) {
    storyTypeNames.add(storyType.toString());
}
return storyTypeNames;
}

private ProjectOrDomain getProjectOrDomain() {
if (getTargetObject() instanceof ProjectOrDomain) {
    return (ProjectOrDomain) getTargetObject();
} else if (getTargetObject() instanceof ProjectOrDomainEntity) {
}
}

```

```
        return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
    }
    return null;
}

private StoryContainer getStoryContainer() {
if (getTargetObject() instanceof StoryContainer) {
    return (StoryContainer) getTargetObject();
}
return null;
}

private Story getStory() {
if (getTargetObject() instanceof Story) {
    return (Story) getTargetObject();
}
return null;
}

private static class CopyListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final StoryEditorPanel panel;

    private CopyListener(StoryEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            CopyStoryCommand copyStoryCommand =
panel.getProjectCommandFactory()
                .newCopyStoryCommand();
            copyStoryCommand.setEditedBy(panel.getCurrentUser());
            copyStoryCommand.setOriginalStory(panel.getStory());
            copyStoryCommand =
panel.getCommandHandler().execute(copyStoryCommand);
            panel.getEventDispatcher().dispatchEvent(
                new UpdateEntityEvent(this, null,
copyStoryCommand.getNewStory()));
            panel.getEventDispatcher().dispatchEvent(
                new OpenPanelEvent(this, PanelActionType.Editor,
copyStoryCommand
                    .getNewStory(), Story.class, null));
        } catch (Exception e) {

```

```

        panel.setGeneralMessage("Could not copy entity: " + e);
    }
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final StoryEditorPanel panel;

    private UpdateListener(StoryEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        Story existingStory = panel.getStory();
        if ((e instanceof UpdateEntityEvent) && (existingStory != null)) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            Story updatedStory = null;
            if (event.getObject() instanceof Story) {
                updatedStory = (Story) event.getObject();
                if ((event instanceof DeletedEntityEvent) &&
existingStory.equals(updatedStory)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingStory));
                    return;
                }
            } else if (event.getObject() instanceof Goal) {
                Goal updatedGoal = (Goal) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if (existingStory.getReferers().contains(updatedGoal)) {
                        existingStory.getReferers().remove(updatedGoal);
                    }
                    updatedStory = existingStory;
                } else if (updatedGoal.getReferers().contains(existingStory)) {
                    for (GoalContainer gc : updatedGoal.getReferers()) {
                        if (gc.equals(existingStory)) {
                            updatedStory = (Story) gc;
                            break;
                        }
                    }
                }
            }
        }
    }
}

```

```

} else if (event.getObject() instanceof Actor) {
    Actor updatedActor = (Actor) event.getObject();
    if (event instanceof DeletedEntityEvent) {
        if (existingStory.getReferers().contains(updatedActor)) {
            existingStory.getReferers().remove(updatedActor);
        }
        updatedStory = existingStory;
    } else if (updatedActor.getReferers().contains(existingStory)) {
        for (ActorContainer ac : updatedActor.getReferers()) {
            if (ac.equals(existingStory)) {
                updatedStory = (Story) ac;
                break;
            }
        }
    }
} else if (event.getObject() instanceof Annotation) {
    Annotation updatedAnnotation = (Annotation) event.getObject();
    if (event instanceof DeletedEntityEvent) {
        if (existingStory.getAnnotations().contains(updatedAnnotation)) {
            existingStory.getAnnotations().remove(updatedAnnotation);
        }
        updatedStory = existingStory;
    } else if
(updatedAnnotation.getAnnotatables().contains(existingStory)) {
        for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
            if (annotatable.equals(existingStory)) {
                updatedStory = (Story) annotatable;
                break;
            }
        }
    }
}
if ((updatedStory != null) && updatedStory.equals(existingStory))
{
    // TODO: check the input fields to see if the user has made
    // a change before resetting the object and updating the
    // input fields.
    panel.setInputValue(EditStoryCommand.FIELD_NAME,
updatedStory.getName());
    panel.setInputValue("storyType",
updatedStory.getStoryType().toString());
    panel.setInputValue(EditStoryCommand.FIELD_TEXT,
updatedStory.getText());
    panel.setInputValue("glossaryTerms", updatedStory);
    panel.setInputValue("goals", updatedStory);
}

```

```
        panel.setInputValue("actors", updatedStory);
        panel.setInputValue("storyContainers", updatedStory);
        panel.setInputValue("annotations", updatedStory);
        panel.setTargetObject(updatedStory);
    }
}
}
}



## storyimpl.java



```
package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Enumerated;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapte

import org.hibernate.annotations.AnyMetaDef;
import org.hibernate.annotations.ManyToOne;
import org.hibernate.annotations.MetaValue;
import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;
```


```

```

import org.hibernate.validator.NotNull;
import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.StoryType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * A story describes an interaction with the system as prose.
 *
 * @author ron
 */
@Entity
@Table(name = "stories", uniqueConstraints =
{ @UniqueConstraint(columnNames =
{ "projectordomain_id", "name" }) })
@XmlRootElement(name = "story", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "story", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class StoryImpl extends AbstractTextEntity implements Story {
    static final long serialVersionUID = 0;

    private Set<StoryContainer> referers = new
TreeSet<StoryContainer>(StoryContainer.COMPARATOR);
    private StoryType storyType;
    private Set<Goal> goals = new TreeSet<Goal>();
    private Set<Actor> actors = new TreeSet<Actor>();

    /**
     * @param projectOrDomain
     * @param name
     * @param createdBy
     * @param text
     * @param storyType

```

```

    */

    public StoryImpl(ProjectOrDomain projectOrDomain, User createdBy,
String name, String text,
    StoryType storyType) {
        super(projectOrDomain, createdBy, name, text);
        // add to collection last so that sorting in the collection by
entity
        // properties has access to all the properties.
        projectOrDomain.getStories().add(this);
        setStoryType(storyType);
    }

    protected StoryImpl() {
        // for hibernate
    }

    @Override
    @Column(nullable = false, unique = true)
    @NotEmpty(message = "a unique name is required.")
    @XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
    public String getName() {
        return super.getName();
    }

    // hack for JAXB to set the name, for some reason it won't use the
inherited
    // method.
    @Override
    public void setName(String name) {
        super.setName(name);
    }

    @Transient
    @XmlID
    @XmlAttribute(name = "id")
    public String getXmlId() {
        return "STRY_" + getId();
    }

    @Transient
    public String getDescription() {
        return "Story: " + getName();
    }

    @XmlTransient

```

```

@ManyToMany(fetch = FetchType.LAZY, metaColumn = @Column(name =
"storycontainer_type", length = 255, nullable = false))
@AnyMetaDef(idType = "long", metaType = "string", metaValues = {
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Project",
targetEntity = ProjectImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Actor",
targetEntity = ActorImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.Goal",
targetEntity = GoalImpl.class),
    @MetaValue(value = "edu.harvard.fas.rregan.requel.project.UseCase",
targetEntity = UseCaseImpl.class) })
@JoinTable(name = "story_storycontainers", joinColumns =
{ @JoinColumn(name = "story_id") }, inverseJoinColumns = {
    @JoinColumn(name = "storycontainer_type"), @JoinColumn(name =
"storycontainer_id") })
@Sort(type = SortType.COMPARATOR, comparator =
StoryContainer.StoryContainerComparator.class)
public Set<StoryContainer> getReferers() {
    return referers;
}

protected void setReferers(Set<StoryContainer> referers) {
    this.referers = referers;
}

@Override
@Enumerated(EnumType.STRING)
@XmlAttribute(name = "storyType")
@XmlJavaTypeAdapter(StoryTypeAdapter.class)
@Column(nullable = false)
@NotNull(message = "a type is required.")
public StoryType getStoryType() {
    return storyType;
}

@Override
public void setStoryType(StoryType storyType) {
    this.storyType = storyType;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.GoalContainer#getGoals()
 */
@Override
XmlElementWrapper(name = "goals", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")

```

```

@XmlIDREF
@XmlElement(name = "goalRef", type = GoalImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = GoalImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "story_goals", joinColumns = { @JoinColumn(name =
"story_id") }, inverseJoinColumns = { @JoinColumn(name = "goal_id") })
@Sort(type = SortType.NATURAL)
public Set<Goal> getGoals() {
    return goals;
}

protected void setGoals(Set<Goal> goals) {
    this.goals = goals;
}

XmlElementWrapper(name = "actors", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "actorRef", type = ActorImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = ActorImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
    CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "story_actors", joinColumns = { @JoinColumn(name =
"story_id") }, inverseJoinColumns = { @JoinColumn(name =
"actor_id") })
@Sort(type = SortType.NATURAL)
public Set<Actor> getActors() {
    return actors;
}

protected void setActors(Set<Actor> actors) {
    this.actors = actors;
}

@Override
public int compareTo(Story o) {
    return getName().compareToIgnoreCase(o.getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to

```

```

* this object.
*
* @param userRepository
* @param defaultCreatedByUser -
*          the user to be set as the created by if no user is
supplied.
* @see UnmarshallerListener
*/
public void afterUnmarshal(final UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {
                // update the references to goals
                for (Goal goal : getGoals()) {
                    goal.getReferers().add(StoryImpl.this);
                }
                for (Actor actor : getActors()) {
                    actor.getReferers().add(StoryImpl.this);
                }
            } catch (RuntimeException e) {
                throw e;
            } catch (Exception e) {
                throw new SAXException2(e);
            }
        }
    });
}

/**
 * This class is used by JAXB to convert the StoryType of a Story
into a
 * string for an attribute in the xml file and the reverse when
 * unmarshalling.
 *
 * @author ron
 */
@XmlTransient
public static class StoryTypeAdapter extends XmlAdapter<String,
StoryType> {

    @Override
    public StoryType unmarshal(String typeString) throws Exception {

```

```

        return StoryType.valueOf(typeString);
    }

    @Override
    public String marshal(StoryType type) throws Exception {
        return type.toString();
    }
}

```

storynavigatorpanel.java

```

/*
 * $Id: StoryNavigatorPanel.java,v 1.2 2009/02/23 07:37:24 rregan Exp
$ 
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEnti
ty;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
e1;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * TODO: integrate the StoriesTable with this.
 *
 * @author ron
 */
public class StoryNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(StoryNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private UpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the StoryNavigatorPanel.properties to set
     * the
     * * label on the new Story button.
     */

```

```

    public static final String PROP_NEW_STORY_BUTTON_LABEL =
"NewStoryButton.Label";

    /**
     * Property name to use in the StoryNavigatorPanel.properties to set
     * the
     * * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * Property name to use in the StoryNavigatorPanel.properties to set
     * the
     * * label on the edit Story button in each row of the table.
     */
    public static final String PROP_EDIT_STORY_BUTTON_LABEL =
>EditStoryButton.Label";

    /**
     * Property name to use in the StoryNavigatorPanel.properties to set
     * the
     * * label on the view Story button in each row of the table when the
     * user
     * * doesn't have edit permission.
     */
    public static final String PROP_VIEW_STORY_BUTTON_LABEL =
"ViewStoryButton.Label";

    /**
     */
    public StoryNavigatorPanel() {
        super(StoryNavigatorPanel.class.getName(), Project.class,
ProjectManagementPanelNames.PROJECT_STORIES_NAVIGATOR_PANEL_NAME);
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

            new NavigatorTableCellValueFactory() {
                @Override
                public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                    Story story = (Story) model.getBackingObject(row);
                    String buttonLabel = null;
                    if (isReadOnlyMode()) {
                        buttonLabel = getResourceBundleHelpergetLocale().getString(
PROP_VIEW_STORY_BUTTON_LABEL, "View");
                    } else {

```

```

buttonLabel = getResourceBundleHelper(getLocale()).getString(
    PROP_EDIT_STORY_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor, story, Story.class, null,
    WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
}
));
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Story story = (Story) model.getBackingObject(row);
            return story.getName();
        }
    }));
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Type",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            Story Story = (Story) model.getBackingObject(row);
            return Story.getStoryType().toString();
        }
    }));
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            return entity.getCreatedBy().getUsername();
        }
    }));
    });
});
tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    }));
setTableConfig(tableConfig);
}
/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "Storys: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "Stories: {0}");
    ProjectOrDomain pod = getProjectOrDomain();
    if (pod != null) {
        name = pod.getName();
    }
    return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
}

```

```

removeAll();
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
    updateListener);
    updateListener = null;
}
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);

    if (!isReadOnlyMode()) {
        String newStoryButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_NEW_STORY_BUTTON_LABEL, "Add");
        NavigationEvent openStoryEditor = new OpenPanelEvent(this,
PanelActionType.Editor,
        getProjectOrDomain(), Story.class, null,
WorkflowDisposition.NewFlow);
        NavigatorButton newStoryButton = new
NavigatorButton(newStoryButtonLabel,
        getEventDispatcher(), openStoryEditor);
        newStoryButton.setStyleName(STYLE_NAME_DEFAULT);
        buttonsWrapper.add(newStoryButton);
    }

    add(buttonsWrapper);

    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(Story.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject((ProjectOrDomain)
targetObject).getStories());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final StoryNavigatorPanel panel;

    private UpdateListener(StoryNavigatorPanel panel) {
        this.panel = panel;
    }

    @Override

```

```

public void actionPerformed(ActionEvent e) {
    if (e instanceof UpdateEntityEvent) {
        UpdateEntityEvent event = (UpdateEntityEvent) e;
        ProjectOrDomain updatedPod = null;
        if (event.getObject() instanceof ProjectOrDomain) {
            updatedPod = (ProjectOrDomain) event.getObject();
        } else if (event.getObject() instanceof ProjectOrDomainEntity) {
            ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
                event.getObject();
            updatedPod = updatedEntity.getProjectOrDomain();
        } else if (event.getObject() instanceof Annotation) {
            if (!(event instanceof DeletedEntityEvent)) {
                Annotation updatedAnnotation = (Annotation) event.getObject();
                for (Annotatable annotatable :
                    updatedAnnotation.getAnnotatables()) {
                    if ((annotatable instanceof ProjectOrDomain)
                        && annotatable.equals(panel.getProjectOrDomain())) {
                        updatedPod = (ProjectOrDomain) annotatable;
                        break;
                    } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                        ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
                            annotatable;
                        if
                            (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                                updatedPod = entity.getProjectOrDomain();
                                break;
                            }
                        }
                    }
                }
            if (panel.getProjectOrDomain().equals(updatedPod)) {
                panel.setTargetObject(updatedPod);
            }
        }
    }
}

```

storyselectorpanel.java

```

/*
 * $Id: StorySelectorPanel.java,v 1.4 2009/02/23 07:37:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

```

```

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableMode
l;
import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class StorySelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(StorySelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the StoryNavigatorPanel.properties to set
     * the
     * * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CCancelButton.Label";

    /**
     * @param projectRepository
     */
    public StorySelectorPanel(ProjectRepository projectRepository) {
        super(StorySelectorPanel.class.getName(), Project.class,
ProjectManagementPanelNames.PROJECT_STORY_SELECTOR_PANEL_NAME);
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();

        tableConfig.setRowLevelSelection(true);

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Story story = (Story) model.getBackingObject(row);
        return story.getName();
    }
}));


        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Type",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Story story = (Story) model.getBackingObject(row);
        return story.getStoryType().toString();
    }
}));


        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Story story = (Story) model.getBackingObject(row);
        return story.getCreatedBy().getUsername();
    }
}));


        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        Story story = (Story) model.getBackingObject(row);
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        return format.format(story.getDateCreated());
    }
}));


        setTableConfig(tableConfig);
    }

    /**
     * Create a title for panel with dynamic information from the project
     * or
     * * domain, by default the title is "Select Story"<br>
     * *
     * * @see Panel.PROP_PANEL_TITLE
     * @see
     edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
     */
    @Override

```

```

public String getTitle() {
    return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Select Story");
}

@Override
public void dispose() {
    super.dispose();
removeAll();
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    updateListener = null;
}
}

@Override
public void setup() {
super.setup();

Row buttonsWrapper = new Row();
buttonsWrapper.setInsets(new Insets(10, 5));
buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
NavigationEvent closeEvent = new ClosePanelEvent(this, this);
NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
closeButton.setStyleName(STYLE_NAME_DEFAULT);
buttonsWrapper.add(closeButton);
add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}
}

```

```

protected boolean isReadOnlyMode() {
User user = (User) getApp().getUser();
if (getProjectOrDomain() instanceof Project) {
    Project project = (Project) getProjectOrDomain();
    Stakeholder stakeholder = project.getUserStakeholder(user);
    if (stakeholder != null) {
        return !stakeholder.hasPermission(Story.class,
StakeholderPermissionType.Edit);
    }
}
return true;
}

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 *
 * @return an adapter to get the collection of items to select from
from the
 * target object.
 */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
return new NavigatorTableModelAdapter() {
private ProjectOrDomain targetObject;

@Override
public Collection<Object> getCollection() {
    return (Collection) targetObject.getStories();
}

@Override
public void setTargetObject(Object targetObject) {
    this.targetObject = (ProjectOrDomain) targetObject;
};

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}
}

```

```
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the Story
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final StorySelectorPanel panel;

    private UpdateListener(StorySelectorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
                            && annotatable.equals(panel.getProjectOrDomain())) {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        } else if (((annotatable instanceof ProjectOrDomainEntity)) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                            if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain())) {
                                updatedPod = entity.getProjectOrDomain();
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
        }
    }
}
}
if (panel.getProjectOrDomain().equals(updatedPod)) {
    panel.setTargetObject(updatedPod);
}
}
}
}
}
```

storytype.java

```
/*
 * $Id: StoryType.java,v 1.2 2008/09/26 23:08:58 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

/**
 * The type/disposition of a story, such as a successful or
exceptional case.
 *
 * @author ron
 */
public enum StoryType {

    /**
     * A story with a successful outcome.
     */
    Success,

    /**
     * A story describing a problem case.
     */
    Exception;

    private StoryType() {
    }
}
```

stringnlptextwalker.java

```
/*
 * $Id: StringNLPTextWalker.java,v 1.4 2009/03/22 11:08:22 rregan Exp
$ Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

/**
 * A driver for an NLPTextWalkerFunction that converts the final
StringBuilder
 * returned by the function to a String.
 *
 * @author ron
 */
public class StringNLPTextWalker extends AbstractNLPTextWalker<String,
StringBuilder> {

    /**
     * @param function -
     *          a NLPTextWalkerFunction that returns a StringBuilder
     */
    public StringNLPTextWalker(NLPTextWalkerFunction<StringBuilder>
function) {
        super(function);
    }

    /**
     * @see
edu.harvard.fas.rregan.nlp.impl.AbstractNLPTextWalker#transformResults
(java.lang.Object)
     */
    @Override
    public String transformResults(StringBuilder stringBuilder) {
        return stringBuilder.toString();
    }
}
```

subjectphrasefinder.java

```
/*
 * $Id: SubjectPhraseFinder.java,v 1.2 2008/07/25 01:17:07 rregan Exp
$
```

```
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.ParseTag;

/**
 * @author ron
 */
public class SubjectPhraseFinder implements NLPProcessor<NLPText> {

    /**
     * if the text is a SENTENCE or CLAUSE return the subject noun
phrase.
     */
    @Override
    public NLPText process(NLPText text) {
        NLPText clause = null;
        if (text.is(GrammaticalStructureLevel.SENTENCE)) {
            // first clause holds the subject?
            for (NLPText child : text.getChildren()) {
                if (child.is(GrammaticalStructureLevel.CLAUSE)) {
                    clause = child;
                    break;
                }
            }
        } else if (text.is(GrammaticalStructureLevel.CLAUSE)) {
            clause = text;
        }
        if (clause != null) {
            for (NLPText child : clause.getChildren()) {
                if (child.is(GrammaticalStructureLevel.PHRASE) &&
child.is(ParseTag.NP)) {
                    return child;
                }
            }
        }
        return null;
    }
}
```

synset.java

```
/*
 * $Id: Synset.java,v 1.4 2009/03/27 07:16:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.CollectionOfElements;
import org.hibernate.annotations.IndexColumn;
import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;

import edu.harvard.fas.rregan.nlp.PartOfSpeech;
```

```
/**
 * Wordnet Synset (synonym set)
 */
@Entity
@Table(name = "synset")
@XmlRootElement(name = "synset")
public class Synset implements Comparable<Synset>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;
    private String pos;
    private Category category;
    private String definition;
    private Map<Linkdef, Integer> subsumerCounts = new HashMap<Linkdef, Integer>();
    private Set<Sense> senses = new TreeSet<Sense>();
    private Set<Semlinkref> semlinks = new TreeSet<Semlinkref>();
    private List<SynsetDefinitionWord> words = new
    ArrayList<SynsetDefinitionWord>();

    // private List<SynsetDefinitionWordEntry>
    parsedDefinitionWordEntries;

    protected Synset() {
    }

    @Id
    @Column(name = "synsetid", unique = true, nullable = false)
    @XmlID
    @XmlAttribute(name = "id")
    @XmlJavaTypeAdapter(IdAdapter.class)
    public Long getId() {
        return this.id;
    }

    protected void setId(Long id) {
        this.id = id;
    }

    @Transient
    public PartOfSpeech getPartOfSpeech() {
        return PartOfSpeech.fromWordNetPOS(getPos());
    }

    public boolean isPartOfSpeech(PartOfSpeech pos) {
        return getPartOfSpeech().equals(pos);
    }
```

```

@Column(name = "pos", length = 2)
@XmlAttribute(name = "pos")
public String getPos() {
    return this.pos;
}

public void setPos(String pos) {
    this.pos = pos;
}

@ManyToOne(targetEntity = Category.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "categoryid")
@XmlIDREF
@XmlAttribute(name = "category")
public Category getCategory() {
    return this.category;
}

public void setCategory(Category category) {
    this.category = category;
}

@Column(name = "definition", length = 1000)
XmlElement(name = "definition")
public String getDefinition() {
    return this.definition;
}

public void setDefinition(String definition) {
    this.definition = definition;
}

@Transient
public Integer getSubsumerCount(Linkdef linkType) {
    if (getSubsumerCounts().containsKey(linkType)) {
        return getSubsumerCounts().get(linkType);
    }
    return 0;
}

public void setSubsumerCount(Linkdef linkType, Integer count) {
    getSubsumerCounts().put(linkType, count);
}

/***
 * @return a map of the relation type to count of all synsets
 * subsumed by
 *          this synset in that particular relation.
 */
@CollectionOfElements(targetElement = Integer.class, fetch =
FetchType.LAZY)
@org.hibernate.annotations.MapKey()
@JoinTable(name = "synset_subsumer_counts", joinColumns =
{ @JoinColumn(name = "synsetid") })
public Map<Linkdef, Integer> getSubsumerCounts() {
    return subsumerCounts;
}

public void setSubsumerCounts(Map<Linkdef, Integer> subsumerCounts) {
    this.subsumerCounts = subsumerCounts;
}

@OneToMany(targetEntity = Sense.class, cascade = { CascadeType.ALL },
fetch = FetchType.LAZY)
@JoinColumn(name = "synsetid")
@Sort(type = SortType.NATURAL)
@XmlElementWrapper(name = "senses")
@XmlElementRef(name = "sense")
public Set<Sense> getSenses() {
    return senses;
}

public void setSenses(Set<Sense> senses) {
    this.senses = senses;
}

@OneToMany(targetEntity = Semlinkref.class, cascade =
{ CascadeType.ALL }, fetch = FetchType.LAZY)
@JoinColumn(name = "synsetlid", referencedColumnName = "synsetid")
@Sort(type = SortType.NATURAL)
@XmlElementWrapper(name = "semlinks")
@XmlElementRef(name = "semlink")
public Set<Semlinkref> getSemlinks() {
    return semlinks;
}

public void setSemlinks(Set<Semlinkref> semlinks) {
    this.semlinks = semlinks;
}

@Transient
@XmlTransient

```

```

public Set<Synset> getHyponyms() {
    Set<Synset> hyponyms = new HashSet<Synset>();
    for (Semlinkref semlinkref : getSemlinks()) {
        if ("hyponym".equals(semlinkref.getLinkType().getName())) {
            hyponyms.add(semlinkref.getToSynset());
        }
    }
    return hyponyms;
}

@Transient
@XmlTransient
public Set<Synset> getHyperonyms() {
    Set<Synset> hyperonyms = new HashSet<Synset>();
    for (Semlinkref semlinkref : getSemlinks()) {
        if ("hypernym".equals(semlinkref.getLinkType().getName())) {
            hyperonyms.add(semlinkref.getToSynset());
        }
    }
    return hyperonyms;
}

/**
 * @return The list of words in the sentence.
 */
@OneToMany(mappedBy = "synset", targetEntity =
SynsetDefinitionWord.class, fetch = FetchType.LAZY, cascade =
CascadeType.ALL)
@IndexColumn(name = "word_index", base = 0)
public List<SynsetDefinitionWord> getWords() {
    return words;
}

protected void setWords(List<SynsetDefinitionWord> words) {
    this.words = words;
}

/*
 * @OneToMany(targetEntity = SynsetDefinitionWordEntry.class, cascade =
= {
    * CascadeType.ALL }, fetch = FetchType.LAZY) @JoinColumn(name =
    * "synset_defword_entry_id") @Sort(type = SortType.NATURAL)
    * @XmlElementWrapper(name = "definitionWordEntries")
@XmlElementRef(name =
    * "definitionWordEntry") public List<SynsetDefinitionWordEntry>
    * getParsedDefinitionWordEntries() { return
parsedDefinitionWordEntries; }

```

```

    * public void
setParsedDefinitionWordEntries( List<SynsetDefinitionWordEntry>
    * parsedDefinitionWordEntries) { this.parsedDefinitionWordEntries =
    * parsedDefinitionWordEntries; }
    */
@Override
public int compareTo(Synset o) {
    return getId().compareTo(o.getId());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getCategory() == null) ? 0 :
getCategory().hashCode());
        result = prime * result + ((getDefinition() == null) ? 0 :
getDefinition().hashCode());
        result = prime * result + ((getPos() == null) ? 0 :
getPos().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final Synset other = (Synset) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getDefinition() == null) {

```

```

if (other.getDefinition() != null) {
    return false;
}
} else if (!getDefinition().equals(other.getDefinition())) {
    return false;
}
if (getCategory() == null) {
    if (other.getCategory() != null) {
        return false;
    }
} else if (!getCategory().equals(other.getCategory())) {
    return false;
}
if (getPos() == null) {
    if (other.getPos() != null) {
        return false;
    }
} else if (!getPos().equals(other.getPos())) {
    return false;
}
}
return true;
}

@Override
public String toString() {
    return getPartOfSpeech() + "[" + getId() + "]: " + getDefinition();
}

/**
 * This class is used by JAXB to convert the id of an entity into an
 * xml id
 * string that will be distinct from other entity xml id strings by
 * the use
 * of a prefix.
 *
 * @author ron
 */
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "SYNSET_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {

```

```

        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

synsetdefinitionword.java

```

/*
 * $Id: SynsetDefinitionWord.java,v 1.3 2009/01/03 10:24:32 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Index;

import edu.harvard.fas.rregan.nlp.ParseTag;
import edu.harvard.fas.rregan.nlp.dictionary.Word.IdAdapter;

/**
 * @author ron
 */

```

```

@Entity
@Table(name = "synset_definition_word")
@org.hibernate.annotations.Table(appliesTo = "synset_definition_word",
indexes = {
    @Index(name = "index_sdw_word", columnNames = { "word_id" }),
    @Index(name = "index_sdw_synset", columnNames = { "sense_id" }) })
@XmlRootElement(name = "word")
public class SynsetDefinitionWord implements
Comparable<SynsetDefinitionWord>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;
    private Synset synset;
    private Integer index;
    private String text;
    private ParseTag parseTag;
    private Category properNounCategory;
    private Sense sense;

    /**
     * Create a word that doesn't have a sense, such as text tagged with
     DT, IN,
     * WDT, CC, ...
     *
     * @param synset -
     *          the synset this word is part of.
     * @param index -
     *          the index of the word (one based) in the synset.
     * @param text -
     *          the text of the word.
     * @param parseTag -
     *          the parse tag.
     */
    public SynsetDefinitionWord(Synset synset, Integer index, String
text, ParseTag parseTag) {
        setSynset(synset);
        setIndex(index);
        setText(text);
        setParseTag(parseTag);
    }

    /**
     * Create a word with a sense.
     *
     * @param synset -
     *          the synset this word is part of.
     * @param index -

```

```

     *          the index of the word (one based) in the synset
     definition.
     * @param text -
     *          the text of the word.
     * @param parseTag -
     *          the parse tag.
     * @param sense -
     *          the wordnet Sense.
     */
    public SynsetDefinitionWord(Synset synset, Integer index, String
text, ParseTag parseTag,
        Sense sense) {
        setSynset(synset);
        setIndex(index);
        setText(text);
        setParseTag(parseTag);
        setSense(sense);
    }

    /**
     * Create a word that is a named entity, for example
     * City_Executive_Committee, City_of_Atlanta, or
     * Superior_Court_Judge_Durwood_Pye
     *
     * @param synset -
     *          the synset this word is part of.
     * @param index -
     *          the index of the word (one based) in the synset
     definition.
     * @param text -
     *          the text of the word.
     * @param parseTag -
     *          the parse tag.
     * @param sense -
     *          the wordnet Sense.
     * @param properNounCategory -
     *          the wordnet category of an entity mapped from the pn
     value in
     *          Semcor such as group, location or person.
     */
    public SynsetDefinitionWord(Synset synset, Integer index, String
text, ParseTag parseTag,
        Sense sense, Category properNounCategory) {
        setSynset(synset);
        setIndex(index);
        setText(text);
        setParseTag(parseTag);

```

```

setProperNounCategory(properNounCategory);
setSense(sense);
}

protected SynsetDefinitionWord() {
    // for hibernate
}

@Id
@Column(name = "id", unique = true, nullable = false)
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlElement
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "synset_id")
public Synset getSynset() {
    return synset;
}

public void setSynset(Synset sentence) {
    this.synset = sentence;
}

@Column(name = "word_index")
public Integer getIndex() {
    return index;
}

public void setIndex(Integer index) {
    this.index = index;
}

@Column(name = "text")
public String getText() {
    return text;
}

public void setText(String text) {
    this.text = text;
}

@Transient
public boolean isIgnored() {
    return getSense() == null;
}

@Enumerated(EnumType.STRING)
@Column(name = "parse_tag")
public ParseTag getParseTag() {
    return parseTag;
}

public void setParseTag(ParseTag parseTag) {
    this.parseTag = parseTag;
}

@Transient
public boolean isNamedEntity() {
    return properNounCategory != null;
}

@ManyToOne(targetEntity = Category.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "category_id")
public Category getProperNounCategory() {
    return properNounCategory;
}

public void setProperNounCategory(Category properNounCategory) {
    this.properNounCategory = properNounCategory;
}

@ManyToOne(targetEntity = Sense.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumns( { @JoinColumn(name = "sense_id"), @JoinColumn(name =
"word_id") })
public Sense getSense() {
    return sense;
}

public void setSense(Sense sense) {
    this.sense = sense;
}

```

```

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = getId().hashCode();
        } else {
            final int prime = 31;
            int result = 1;
            result = prime * result + getSynset().hashCode();
            result = prime * result + getIndex().hashCode();
            tmpHashCode = result;
        }
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    // NOTE: getClass().equals(obj.getClass()) fails when the obj is a
proxy
    if (!(obj instanceof SynsetDefinitionWord)) {
        return false;
    }
    final SynsetDefinitionWord other = (SynsetDefinitionWord) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }

    if (getSynset() == null) {
        if (other.getSynset() != null) {
            return false;
        }
    } else if (!getSynset().equals(other.getSynset())) {
        return false;
    }
    if (getIndex() == null) {
        if (other.getIndex() != null) {
            return false;
        }
    }
}

```

```

} else if (!getIndex().equals(other.getIndex())) {
    return false;
}
return true;
}

@Override
public int compareTo(SynsetDefinitionWord o) {
    if (getSynset().equals(o.getSynset())) {
        return getIndex().compareTo(o.getIndex());
    }
    return getSynset().compareTo(o.getSynset());
}

@Override
public String toString() {
    if (getSense() != null) {
        return getSense().toString();
    }
    return getText();
}
}

```

synsethypernymwalkcommand.java

```

/*
 * $Id: SynsetHypernymWalkCommand.java,v 1.1 2008/12/13 00:40:11
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.command;

import java.util.Map;
import java.util.Set;

import edu.harvard.fas.rregan.command.Command;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init.WordNetHypo
nymCountInitializer;
import
edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init.WordNetHypo
nymCountInitializer.Counter;

/**
 */

```

```

* Given a synset, walk up the hypernym graph and update counters.
*
* @author ron
* @see WordNetHypernymCountInitializer
*/
public interface SynsetHypernymWalkCommand extends Command {

    /**
     * set the synset to start from and walk up the graph.
     *
     * @param synset
     */
    public void setSynset(Synset synset);

    /**
     * set the map of counters for all synsets.
     *
     * @param hyponymCounts
     */
    public void setHyponymCounts(Map<Synset, Counter> hyponymCounts);

    /**
     * set the map of ancestors for all synsets.
     *
     * @param hyponymAncestors
     */
    public void setHyponymAncestors(Map<Synset, Set<Synset>>
hyponymAncestors);
}


```

synsethypernymwalkcommandimpl.java

```

/*
 * $Id: SynsetHypernymWalkCommandImpl.java,v 1.1 2008/12/13 00:39:56
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary.impl.command;

import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.command.SynsetHypernymWalkCommand;
import edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init.WordNetHypernymCountInitializer.Counter;

/**
 * @author ron
 */
@Controller("synsetHypernymWalkCommand")
@Scope("prototype")
public class SynsetHypernymWalkCommandImpl extends
AbstractDictionaryCommand implements
SynsetHypernymWalkCommand {

    private Synset synset;
    private Map<Synset, Set<Synset>> hyponymAncestors;
    private Map<Synset, Counter> hyponymCounts;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public SynsetHypernymWalkCommandImpl(DictionaryRepository
dictionaryRepository) {
        super(dictionaryRepository);
    }

    /**
     * @see
edu.harvard.fas.rregan.nlp.dictionary.command.SynsetHypernymWalkCommand#setHyponymAncestors(java.util.Map)
     */
    @Override
    public void setHyponymAncestors(Map<Synset, Set<Synset>>
hyponymAncestors) {
        this.hyponymAncestors = hyponymAncestors;
    }

    /**

```

```

 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.SynsetHypernymWalkCommand
d#setHyponymCounts(java.util.Map)
 */
@Override
public void setHyponymCounts(Map<Synset, Counter> hyponymCounts) {
    this.hyponymCounts = hyponymCounts;
}

/***
 * @see
edu.harvard.fas.rregan.nlp.dictionary.command.SynsetHypernymWalkCommand
d#setSynset(edu.harvard.fas.rregan.nlp.dictionary.Synset)
 */
@Override
public void setSynset(Synset synset) {
    this.synset = synset;
}

/***
 * @see edu.harvard.fas.rregan.command.Command#execute()
 */
@Override
public void execute() throws Exception {
    Synset synset = getDictionaryRepository().get(this.synset);
    if (log.isDebugEnabled()) {
        log.debug(synset);
    }
    hypernymWalk(synset);
}

/***
 * @param synset
 */
private void hypernymWalk(Synset synset) {
    for (Synset hypernym : synset.getHypernyms()) {
        if (!hyponymAncestors.get(synset).contains(hypernym)) {
            hyponymAncestors.get(synset).add(hypernym);
            hyponymCounts.get(hypernym).count++;
        }
        hypernymWalk(hypernym);
    }
}
}

```

syntaxmatchingcontext.java

```

/*
 * $Id: SyntaxMatchingContext.java,v 1.7 2009/02/11 09:02:54 rregan
Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.List;
import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetFrameRef;

/**
 * A context for applying SyntaxMatchingRules to an NLPText. The
context applies
 * a single set of rules set a creation time to various NLPTexts via
the
 * matches() method.
 *
 * @author ron
 */
public class SyntaxMatchingContext {
    private static final Logger log =
Logger.getLogger(SyntaxMatchingContext.class);

    private final VerbNetFrameRef frameRef;
    private final List<SyntaxMatchingRule> rules;

    /**
     * @param frameRef
     * @param rules
     */
    public SyntaxMatchingContext(VerbNetFrameRef frameRef,
List<SyntaxMatchingRule> rules) {
        this.frameRef = frameRef;
        this.rules = rules;
    }
}

```

```

/**
 * @param dictionaryRepository
 * @param nlpText -
 *          the text to match the rules
 * @param primaryVerb -
 *          the primary verb of the supplied nlpText to extract the
 *          thematic roles for.
 * @return true if the supplied nlpText matches the rules in the
context.
 */
public boolean matches(DictionaryRepository dictionaryRepository,
NLPText nlpText,
NLPText primaryVerb) {
List<NLPText> leaves = nlpText.getLeaves();
ListIterator<NLPText> leafIterator = leaves.listIterator();
log.info("start matching: " + frameRef);
try {
for (SyntaxMatchingRule rule : rules) {
skipNoise(leafIterator);
if (!leafIterator.hasNext()) {
throw
SemanticRoleLabelerException.unmatchedRulesRemaining(rules.subList(rul
es
    .indexOf(rule), rules.size()));
}
log.info("current word: " + leaves.get(leafIterator.nextInt()));
rule.match(dictionaryRepository, primaryVerb, leafIterator);
}
// match the ending punctuation
skipNoise(leafIterator);
if (leafIterator.hasNext()) {
throw
SemanticRoleLabelerException.unmatchedSentenceElementsRemaining(nlpTex
t
    .getTextRange(leafIterator.nextInt(), leaves.size() - 1));
}
} catch (SemanticRoleLabelerException e) {
log.info("matching failed: " + frameRef + " error: " +
e.getMessage());
return false;
}
log.info("matching successful: " + frameRef);
return true;
}

protected void skipNoise(ListIterator<NLPText> iter) {
while (iter.hasNext()) {

```

```

NLPText word = iter.next();
if (!word.in(PartOfSpeech.PUNCTUATION, PartOfSpeech.SYMBOL)) {
iter.previous();
return;
}
}
}
}

```

syntaxmatchingrule.java

```

/*
 * $Id: SyntaxMatchingRule.java,v 1.3 2009/02/11 09:02:55 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ListIterator;

import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;

/**
 * A rule derived from the VerbNet frame syntax that matches a
syntactic element
 * of a sentence potentially identifying a semantic role.
 *
 * @author ron
 */
public interface SyntaxMatchingRule {

/**
 * Apply the rule to the current nlpText in the context and return
the
 * matched nlpText or throw a SemanticRoleLabelerException if the
text
 * wasn't matched.
 *
 * @param dictionaryRepository
 * @param verb -
 *          The primary verb of the text being processed.
 * @param textIterator -
 *          an iterator to get the next word level token in the
text or
 *          step back if the next word is not handled by the rule.

```

```

 * @throws SemanticRoleLabelerException -
 *         if the rule can't be applied to the current state of
the
 *
 *         text.
 */
public void match(DictionaryRepository dictionaryRepository, NLPText
verb,
ListIterator<NLPText> textIterator) throws
SemanticRoleLabelerException;
}

```

systemadminuserrole.java

```

/*
 * $Id: SystemAdminUserRole.java,v 1.10 2009/01/10 11:08:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.user;

import java.util.HashSet;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 * Represents a user with system administration privilges. This role
has no user
 * specific properties so a single instance can be shared by all
users.
 *
 * @author ron
 */
@Entity
@DiscriminatorValue(value =
"edu.harvard.fas.rregan.requel.user.SystemAdminUserRole")
@XmlRootElement(name = "systemAdminUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "systemAdminUserRole", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class SystemAdminUserRole extends AbstractUserRole {
    static final long serialVersionUID = 0L;

```

```

static {
    AbstractUserRole.userRoleTypes.add(SystemAdminUserRole.class);
    AbstractUserRole.userRoleTypePermissions.put(SystemAdminUserRole.cl
ss,
        new HashSet<UserRolePermission>());
}

/**
 * @param roleName
 */
public SystemAdminUserRole() {
    super();
}
}

```

systeminitializer.java

```

/*
 * $Id: SystemInitializer.java,v 1.1 2009/01/26 10:19:04 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan;

/**
 * A SystemInitializer does some initialization on system start up
such as
 * creating and configuring built in entities such as users or
permissions;
 * load database data etc.
 *
 * @author ron
 */
public interface SystemInitializer extends
Comparable<SystemInitializer> {
/**
 * The order is a general order value indicating the order that the
 * initializer should be run relative to other initializers. In
general
 * initializers that have no dependencies should use an order of 100,
 * dependent entites such as specific users should use a value of
1000.
 *
 * @return the order to run the initializer
 */
public int getOrder();
}

```

```

    /**
     * create and persist the entities.
     */
    public void initialize();
}


```

tabbedpanelcontainer.java

```

/*
 * $Id: TabbedPaneContainer.java,v 1.15 2008/03/10 23:57:20 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel;

import java.util.Stack;

import nextapp.echo2.app.Component;
import nextapp.echo2.extras.app.TabPane;
import nextapp.echo2.extras.app.layout.TabPaneLayoutData;

import org.springframework.beans.factory.annotation.Autowired;

import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;

/**
 * A PanelContainer that displays panels in multiple tabs that can be
navigated
 * independently.
 *
 * @author ron
 */
public class TabbedPaneContainer extends AbstractPanelContainerPanel
{
    static final long serialVersionUID = 0L;

    private TabPane tabs;
    private final Stack<Integer> tabDisplayOrder = new Stack<Integer>();

    /**
     * Create a TabbedPaneContainer with the default resource bundle
name
     * (using the full class name of this class) and a
DefaultPanelManager.

```

```

    *
    * @param panelManager -
    *          the panel manager with the set of panels available for
display
    *          already attached.
    */
    @Autowired
    public TabbedPaneContainer(PanelManager panelManager) {
        this(TabbedPaneContainer.class.getName(), panelManager);
    }

    /**
     * Create a TabbedPaneContainer with the specified resource bundle
name and
     * PanelManager.
     *
     * @param resourceBundleName
     * @param panelManager
     */
    public TabbedPaneContainer(String resourceBundleName, PanelManager
panelManager) {
        super(resourceBundleName, panelManager);
    }

    @Override
    public void setup() {
        super.setup();
        tabs = new TabPane();
        tabs.setStyleName(STYLE_NAME_DEFAULT);
        add(tabs);
    }

    @Override
    public void dispose() {
        tabs.removeAll();
        tabs.dispose();
        getPanelManager().dispose();
        super.dispose();
        tabs = null;
    }

    @Override
    public void setStyleName(String newValue) {
        super.setStyleName(newValue);
        tabs.setStyleName(newValue);
    }
}

```

```

public void displayPanel(Panel panel, WorkflowDisposition disposition) {
    // TODO: stack panels in the same flow on top of the panel
    // in a window pane

    // add a new tab if the panel isn't displayed yet, otherwise
    // reset the selection to the existing tab
    int tabIndex = tabs.getActiveTabIndex();
    tabIndex = getTabIndexForPanel(panel);
    if (tabIndex == -1) {
        TabPaneLayoutData layoutData = new TabPaneLayoutData();
        layoutData.setTitle(panel.getTitle());
        panel.setLayoutData(layoutData);

        tabs.add((Component) panel);
        tabs.setActiveTabIndex(tabs.getComponentCount() - 1);
        tabDisplayOrder.push(new Integer(tabs.getComponentCount() - 1));
    } else {
        tabs.setActiveTabIndex(tabIndex);
        // move the index from where ever it is in the stack to the top
        tabDisplayOrder.remove(new Integer(tabIndex));
        tabDisplayOrder.push(new Integer(tabIndex));
    }
}

private int getTabIndexForPanel(Panel panel) {
    int index = -1;
    for (int i = 0; i < tabs.getComponentCount(); i++) {
        if (panel.equals(tabs.getComponent(i))) {
            index = i;
            break;
        }
    }
    return index;
}

public void undisplayPanel(Panel panel) {
    int tabIndex = getTabIndexForPanel(panel);
    if (tabIndex > -1) {
        tabs.remove(tabIndex);
        tabDisplayOrder.remove(new Integer(tabIndex));
        if (!tabDisplayOrder.empty()) {
            tabs.setActiveTabIndex(tabDisplayOrder.peek());
        }
    }
}
}

```

textcomponentmanipulator.java

```

/*
 * $Id: TextComponentManipulator.java,v 1.8 2008/10/13 22:58:58 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.panel.editor.manipulators;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.event.DocumentEvent;
import nextapp.echo2.app.event.DocumentListener;
import nextapp.echo2.app.text.Document;
import nextapp.echo2.app.text.TextComponent;
import edu.harvard.fas.rregan.uiframework.panel.editor.EditMode;

/**
 * @author ron
 */
public class TextComponentManipulator extends AbstractComponentManipulator {

    public <T> T getValue(Component component, Class<T> type) {
        return type.cast(getComponent(component).getText());
    }

    public void setValue(Component component, Object value) {
        getComponent(component).setText((String) value);
    }

    public void addListenerToDetectChangesToInput(finalEditMode
editMode, Component component) {
        final TextComponent text = (TextComponent) component;
        text.getDocument().addDocumentListener(new DocumentListener() {
            static final long serialVersionUID = 0L;

            public void documentUpdate(DocumentEvent e) {
                editMode.setStateEdited(true);
            }
        });
    }

    @Override
    public Document getModel(Component component) {
        return getComponent(component).getDocument();
    }
}

```

```

@Override
public void setModel(Component component, Object valueModel) {
    getComponent(component).setDocument((Document) valueModel);
}

private TextComponent getComponent(Component component) {
    return (TextComponent) component;
}
}

```

textentity.java

```

/*
 * $Id: TextEntity.java,v 1.3 2008/03/19 09:57:31 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

/**
 * An entity whose primary content is text, such as a goal or story.
 *
 * @author ron
 */
public interface TextEntity extends ProjectOrDomainEntity {

    /**
     * @return get the text of this entity.
     */
    public String getText();

    /**
     * change the text of this entity.
     *
     * @param text -
     *          new text
     */
    public void setText(String text);
}

```

textentityassistant.java

```

/*
 * $Id: TextEntityAssistant.java,v 1.10 2009/01/23 09:54:24 rregan Exp
 */

```

```

 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project.impl.assistant;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.TextEntity;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Assistant for analyzing entities based on the TextEntity interface,
such as
 * goals and stories. The assistant applies the lexical assistant to
the name
 * and text properties of each entity.
 *
 * @author ron
 */
public class TextEntityAssistant extends
ProjectOrDomainEntityAssistant {
    private static final Logger log =
Logger.getLogger(TextEntityAssistant.class);

    /**
     * @param resourceBundleName -
     *          the full class name to use for the resource bundle.
     * @param lexicalAssistant -
     *          assistant for analyzing text for spelling, terms and
other
     *          word oriented analysis.
     * @param updatedEntityNotifier -
     *          after an entity is analyzed it is passed to the
notifier to
     *          tell the UI components that reference the entity to
refresh
     * @param assistantUser -
     *          the user to use as the creator of the annotation
entities.
     */
    public TextEntityAssistant(String resourceBundleName,
LexicalAssistant lexicalAssistant,
        User assistantUser) {
        super(resourceBundleName, lexicalAssistant, assistantUser);
    }

    @Override

```

```

public void setEntity(ProjectOrDomainEntity entity) throws
IllegalArgumentException {
super.setEntity(entity);
if (entity instanceof TextEntity) {
setPropertyText(PROP_TEXT, ((TextEntity) entity).getText());
}
}

/**
 * Analyze the text of the text property
 *
 * @param entity
 */
@Override
public void analyze() {
super.analyze();
}
}

```

togglebuttonmodelex.java

```

/*
 * $Id: ToggleButtonModelEx.java,v 1.1 2008/03/05 10:52:13 rregan Exp
$ Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.uiframework.panel.editor;

import nextapp.echo2.app.button.DefaultToggleButtonModel;

/**
 * @author ron
 */
public class ToggleButtonModelEx extends DefaultToggleButtonModel {
static final long serialVersionUID = 0;

/**
 * @param selected
 */
public ToggleButtonModelEx(boolean selected) {
super();
setSelected(selected);
}
}

```

typedassistant.java

```

/*
 * $Id: TypedAssistant.java,v 1.1 2009/01/20 10:26:01 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

/**
 * @param <T> -
 *          the type of entity the assistant analyzes.
 * @author ron
 */
public interface TypedAssistant<T> {

/**
 * @return The entity being analyzed.
 */
public T getEntity();

/**
 * @param entity -
 *          the entity to analyze
 */
public void setEntity(T entity);

/**
 * Start the analysis of the supplied entity.
 */
public void analyze();
}

```

uiframeworkapp.java

```

/*
 * $Id: UIFrameworkApp.java,v 1.11 2008/09/12 00:15:12 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import java.util.Map;
import java.util.Set;

```

```

import nextapp.echo2.app.ApplicationInstance;
import nextapp.echo2.app.Component;
import nextapp.echo2.app.ContentPane;
import nextapp.echo2.app.StyleSheet;
import nextapp.echo2.app.TaskQueueHandle;
import nextapp.echo2.app.Window;
import nextapp.echo2.app.componentxml.ComponentXmlException;
import nextapp.echo2.app.componentxml.StyleSheetLoader;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.webcontainer.ContainerContext;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;

import
edu.harvard.fas.rregan.uiframework.controller.AppAwareController;
import edu.harvard.fas.rregan.uiframework.controller.Controller;
import edu.harvard.fas.rregan.uiframework.login.LoginScreen;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import edu.harvard.fas.rregan.uiframework.screen.Screen;

/**
 * @author ron
 */
@org.springframework.stereotype.Component
@org.springframework.context.annotation.Scope("prototype")
public class UIFrameworkApp extends ApplicationInstance {
    private static final Logger log =
Logger.getLogger(UIFrameworkApp.class);
    static final long serialVersionUID = 0;

    private Window mainWindow;
    private final EventDispatcher eventDispatcher;
    private final Map<String, Screen> screens;
    private Screen currentScreen;
    private Object user;
    private StyleSheet styleSheet;

    /**
     * @param eventDispatcher
     * @param controllers
     * @param screens
     * @param initialScreenName
     */
    @Autowired

```

```

        public UIFrameworkApp(EventDispatcher eventDispatcher,
Set<Controller> controllers,
Map<String, Screen> screens) {
super();
log.debug("new UIFrameworkApp");
this.eventDispatcher = eventDispatcher;
this.screens = screens;

// TODO: setup of controllers can go away if the app scope is
changed to
// session and it
// is injected into each AppAwareController, and the eventDispatcher
is
// injected into
// each controller. The controllers would still need to be autowired
and
// injected into
// this method so that they get instantiated or else they need to be
// manually added to
// the spring config.

// register the events the UIFrameworkApp
// register the controller with the dispatcher
for (Controller controller : controllers) {
    // register the EventDispatcher as a listener on the controller so
    // it can dispatch the controllers events
    controller.addActionListener(eventDispatcher);

    if (controller instanceof AppAwareController) {
        ((AppAwareController) controller).setApp(this);
    }
    if (controller.getEventTypesToListenFor().size() > 0) {
        for (Class<? extends ActionEvent> eventType :
controller.getEventTypesToListenFor()) {
            eventDispatcher.addEventTypeActionListener(eventType,
controller);
        }
    } else {
        log.warn("controller '" + controller
        + "' does not have any event types to listen for.");
    }
}

if (log.isDebugEnabled()) {
    for (String screenName : screens.keySet()) {
        log.debug(screenName + ": " +
screens.get(screenName).getClass().getName());
    }
}

```

```

        }

    }

    try {
        styleSheet = StyleSheetLoader.load("Default.stylesheet",
Thread.currentThread()
        .getContextClassLoader());
    } catch (ComponentXmlException e) {
        throw UIFrameworkException.errorInStyleSheet(e.getMessage());
    }
}

/***
 * @return
 */
public EventDispatcher getEventDispatcher() {
    return eventDispatcher;
}

@Override
public Window init() {
    setStyleSheet(styleSheet);
    mainWindow = new Window();
    mainWindow.setTitle("Title");
    setCurrentScreen(LoginScreen.screenName);
    return mainWindow;
}

/***
 * @param screenName
 */
public void setCurrentScreen(String screenName) {
    if (screens.containsKey(screenName)) {
        Screen newScreen = screens.get(screenName);

        // replace the currentScreen in the window
        ContentPane mainContent = mainWindow.getContent();
        // this removes and disposes all components of the last screen.
        mainContent.removeAll();
        mainContent.addComponent(newScreen);

        newScreen.setup();
        currentScreen = newScreen;
    } else {
        log.warn("specified screen '" + screenName + "' does not exist");
    }
}

```

```

    /**
     * @return
     */
    public Object getUser() {
        return user;
    }

    /**
     * @param user
     */
    public void setUser(Object user) {
        this.user = user;
    }

    /**
     * Add a task to Echo that causes the client to poll back to the
     * server in
     * pollTimeMillis milliseconds and execute the task and remove it
     * from the
     * queue.
     *
     * @see RepeatingTask - a task that reregisters itself to have the
     * client
     * repeat polling until the task is explicitly stopped via
     * stopTask()
     * @see AvailableJobsPanel for an example
     * @param taskQueue -
     *          an object returned by getApp().createTaskQueue().
     * @param task -
     *          a Runnable object like
     * @see RepeatingTask
     * @param pollTimeMillis -
     *          the time in milliseconds to wait before polling
     */
    public void enqueueTask(TaskQueueHandle taskQueue, Runnable task, int
pollTimeMillis) {
        enqueueTask(taskQueue, task);
        // set the timeout in the container context
        ContainerContext containerContext = (ContainerContext)
        getContextProperty(ContainerContext.CONTEXT_PROPERTY_NAME);
        containerContext.setTaskQueueCallbackInterval(taskQueue,
pollTimeMillis);
    }

    /**
     * @return the current app instance relative to the session for the
     * current
     */

```

```

/*
 *      user
 */
public static UIFrameworkApp getApp() {
    return (UIFrameworkApp) getActive();
}

uiframeworkconfiguration.java

/*
 * $Id: UIFrameworkConfiguration.java,v 1.3 2008/02/19 09:48:45 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import nextapp.echo2.app.StyleSheet;
import nextapp.echo2.app.componentxml.ComponentXmlException;
import nextapp.echo2.app.componentxml.StyleSheetLoader;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.uiframework.login.LoginScreen;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DefaultEventDispat
cher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import edu.harvard.fas.rregan.uiframework.screen.Screen;

/**
 * TODO: the configuration should be replaced by Spring AutoWiring of
resources
 *
 * @author ron
 */
public class UIFrameworkConfiguration {
    private static final Logger log =
Logger.getLogger(UIFrameworkConfiguration.class);

    private Screen initialScreen;
    private EventDispatcher eventDispatcher;
    private StyleSheet styleSheet;
    private String mainWindowTitle;

    /**

```

```

     * TODO: place holder configuration
     */
    public UIFrameworkConfiguration() {
        try {
            mainWindowTitle = "Title";
            initialScreen = new LoginScreen();
            eventDispatcher = new DefaultEventDispatcher();
            styleSheet = StyleSheetLoader.load("Default.stylesheet",
Thread.currentThread()
                .getContextClassLoader());
        } catch (ComponentXmlException e) {
            throw UIFrameworkException.errorInStyleSheet(e.getMessage());
        } catch (Exception e) {
            throw UIFrameworkException.exceptionInConfig(e);
        }
        log.debug("new UIFrameworkConfiguration");
    }

    /**
     * @return
     */
    public String getMainWindowTitle() {
        return mainWindowTitle;
    }

    /**
     * @return
     */
    public Screen getInitialScreen() {
        return initialScreen;
    }

    /**
     * @return
     */
    public EventDispatcher getEventDispatcher() {
        return eventDispatcher;
    }

    public StyleSheet getStyleSheet() {
        return styleSheet;
    }

    /**
     * TODO: this should be replaced by Spring AutoWiring of Commands
     *
     * @param app

```

```

    */
public void registerCommands(UIFrameworkApp app) {
}
}

```

uiframeworkexception.java

```

/*
 * $Id: UIFrameworkException.java,v 1.2 2009/01/03 10:24:39 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import edu.harvard.fas.rregan.ApplicationException;

/**
 * @author ron
 */
public class UIFrameworkException extends ApplicationException {
    static final long serialVersionUID = 0;

    protected static String MSG_COULD_NOT_CREATE_PANEL = "Could not
create panel for class %s";
    protected static String MSG_UNSAVED_DATA = "The current panel has
unsaved data.";
    protected static String MSG_EXCEPTION_IN_CONFIG = "The configuration
encountered a problem during initialization: %s.";

    /**
     * Create an exception indicating that the Echo2 stylesheet couldn't
be
     * loaded because of an error.
     *
     * @param message -
     *          The message from the stylesheet loader.
     * @return
     */
    public static ApplicationException errorInStyleSheet(String message)
    {
        return new UIFrameworkException(MSG_EXCEPTION_IN_CONFIG, message);
    }

    /**
     * Create an exception indicating an unexpected exception during
application
    
```

```

        * configuration.
        *
        * @param cause
        * @return
        */
    public static ApplicationException exceptionInConfig(Throwable cause)
    {
        return new UIFrameworkException(cause, MSG_EXCEPTION_IN_CONFIG,
cause);
    }

    /**
     * @param panelClass
     * @param cause
     * @return
     */
    public static ApplicationException couldNotCreatePanel(Class<?>
panelClass, Throwable cause) {
        return new UIFrameworkException(cause, MSG_COULD_NOT_CREATE_PANEL,
panelClass.getName());
    }

    /**
     * @return
     */
    public static ApplicationException panelHasUnsavedData() {
        return new UIFrameworkException(MSG_UNSAVED_DATA);
    }

    /**
     * @param format -
     *          a format string appropriate for java.util.Formatter
     * @param args -
     *          variable args list that map to the variables in the
format
     *          *
     *          string
     */
    protected UIFrameworkException(String format, Object... args) {
        super(format, args);
    }

    /**
     * @param cause -
     *          a caught exception that resulted in this exception
     * @param format -
     *          a format string appropriate for java.util.Formatter
     * @param args -
     */

```

```

        *      variable args list that map to the variables in the
format
        *      string
*/
protected UIFrameworkException(Throwable cause, String format,
Object... args) {
    super(format, args, cause);
}
}

```

uiframeworklogoutservlet.java

```

package edu.harvard.fas.rregan.uiframework;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * To properly clear the session of an Echo2 application a separate
servlet is
 * required. This invalidates the session and redirects to the
* UIFrameworkServlet to display the login screen.
*
* @author ron
*/
public class UIFrameworkLogoutServlet extends HttpServlet {
    static final long serialVersionUID = 0;

    @Override
    protected void service(HttpServletRequest request,
    HttpServletResponse response)
        throws ServletException, IOException {
        request.getSession().invalidate();
        // TODO: this presumes that the UIFrameworkServlet is registered at
the
        // root of the context
        response.sendRedirect(request.getContextPath());
    }
}

```

uiframeworkservlet.java

```

/*
 * $Id: UIFrameworkServlet.java,v 1.4 2008/02/26 01:30:26 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework;

import nextapp.echo2.app.ApplicationInstance;
import nextapp.echo2.webcontainer.WebContainerServlet;

import org.springframework.web.context.WebApplicationContext;
import
org.springframework.web.context.support.WebApplicationContextUtils;

/**
 * @author ron
 */
public class UIFrameworkServlet extends WebContainerServlet {
    static final long serialVersionUID = 0;

    /**
     *
     */
    public UIFrameworkServlet() {
        super();
    }

    @Override
    public ApplicationInstance newApplicationInstance() {
        WebApplicationContext ctx =
WebApplicationContextUtils.getWebApplicationContext(getServletContext(
));
        return (ApplicationInstance)
ctx.getAutowireCapableBeanFactory().createBean(UIFrameworkApp.class);
    }
}

```

uimethoddisplayhint.java

```

/*
 * $Id: UIMethodDisplayHint.java,v 1.1 2008/02/15 21:42:02 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

/*
package edu.harvard.fas.rregan.uiframework.reflect;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * @author ron
 */
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface UIMethodDisplayHint {
    public static final int IGNORE = 0;
    public static final int SHORT = 1;
    public static final int LONG = 2;
    public static final int SHORT_OR_LONG = (SHORT | LONG);

    /**
     * An explicit property name for methods that don't follow
     * JavaBean naming conventions.
     *
     * @return
     */
    String propertyName() default "";

    /**
     * An alternate label to use instead of mangling the method name to
     * generate
     * a label for the property.
     * @return
     */
    String label() default "";

    /**
     * Level where this property is appropriate to show:
     * IGNORE - never include this property
     * SHORT - include this property in short displays
     * LONG - include this property in long displays
     * SHORT_OR_LONG - always include this property
     * @return
     */
    int displayLevel() default SHORT_OR_LONG;

    /**

```

```

        * Set this to true if this property should be used in
        * constraining a search.
        *
        * @return
        */
        boolean searchable() default false;

        /**
         * Use the supplied pattern to format the value of the
         * target object for the display value.
         * @return
         */
        String targetPattern() default "";

        /**
         * Use the supplied property of the target object as the
         * display value.
         *
         * @return
         */
        String targetProperty() default "";

    }
}
```

uitypedisplayhint.java

```

/*
 * $Id: UITypeDisplayHint.java,v 1.1 2008/02/15 21:41:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.reflect;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface UITypeDisplayHint {
    /**
     * The name of the property to display when representing an
     * object as a single cell view, such as a table cell or tree
     * cell.
     *
     * @return

```

```

        */
String targetProperty() default "";
}

```

unmarshallerlistener.java

```

/*
 * $Id: UnmarshallerListener.java,v 1.1 2008/12/13 00:40:36 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.Unmarshaller;

import org.apache.log4j.Logger;

/**
 * @author ron
 */
public class UnmarshallerListener extends Unmarshaller.Listener {
    protected static final Logger log =
Logger.getLogger(UnmarshallerListener.class);

    // a list of the possible before/afterUnmarshal method parameters
    private static final List<Class<?>[]>
beforeAndAfterUnmarshalMethodParams = new ArrayList<Class<?>[]>();
    static {
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[] {});
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ DictionaryRepository.class,
    Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ DictionaryRepository.class });
    }
}

```

```

private final DictionaryRepository wordNetRepository;

/**
 * @param wordNetRepository
 */
public UnmarshallerListener(DictionaryRepository wordNetRepository) {
    this.wordNetRepository = wordNetRepository;
}

@Override
public void beforeUnmarshal(Object target, Object parent) {
    super.beforeUnmarshal(target, parent);
    findAndCallUnmarshalMethod("beforeUnmarshal", target, parent);
}

@Override
public void afterUnmarshal(Object target, Object parent) {
    super.afterUnmarshal(target, parent);
    findAndCallUnmarshalMethod("afterUnmarshal", target, parent);
}

/**
 * Search for a method with the supplied name and predefined possible
method
 * parameters, and if found call the method.
 *
 * @param methodName
 * @param target
 * @param parent
 */
private void findAndCallUnmarshalMethod(String methodName, Object
target, Object parent) {
    Class<?> targetType = target.getClass();
    Method afterUnmarshalMethod = null;
    Object[] params = null;
    try {
        // search for the method starting with the most specific class
        // and working up the inheritance hierarchy
        while (targetType != null) {
            for (Class<?>[] methodParamTypes :
beforeAndAfterUnmarshalMethodParams) {
                try {
                    afterUnmarshalMethod = targetType.getDeclaredMethod(methodName,
                        methodParamTypes);
                    afterUnmarshalMethod.setAccessible(true);
                    params = new Object[methodParamTypes.length];
                    for (int i = 0; i < methodParamTypes.length; i++) {

```

```

        if (methodParamTypes[i].equals(DictionaryRepository.class)) {
            params[i] = wordNetRepository;
        } else if (methodParamTypes[i].equals(Object.class)) {
            params[i] = parent;
        }
    }
    afterUnmarshalMethod.invoke(target, params);
} catch (NoSuchMethodException e) {
} catch (SecurityException e) {
    log.error("The afterUnmarshal method for the class " +
targetType.getName()
        + " is not accessible.", e);
}
targetType = targetType.getSuperclass();
} catch (Exception e) {
    log.error("Exception processing afterUnmarshal(" + params + ") on
class " + targetType
        + " for object " + target, e);
}
}
}


```

unmarshallerlistener.java

```

/*
 * $Id: UnmarshallerListener.java,v 1.7 2009/01/07 10:10:04 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.utils.jaxb;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.Unmarshaller;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.repository.Repository;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;

        /**
         * @author ron
         */
public class UnmarshallerListener extends Unmarshaller.Listener {
    protected static final Logger log =
Logger.getLogger(UnmarshallerListener.class);

    // a list of the possible before/afterUnmarshal method parameters
    private static final List<Class<?>[]>
beforeAndAfterUnmarshalMethodParams = new ArrayList<Class<?>[]>();
    static {
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[] {});
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ String.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ Repository.class, Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ ProjectRepository.class,
    UserRepository.class, Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ ProjectRepository.class,
    Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ UserRepository.class, User.class,
    Object.class });
        beforeAndAfterUnmarshalMethodParams
            .add(new Class<?>[] { UserRepository.class, Object.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ ProjectRepository.class });
        beforeAndAfterUnmarshalMethodParams.add(new Class<?>[]
{ UserRepository.class });
        beforeAndAfterUnmarshalMethodParams
            .add(new Class<?>[] { UserRepository.class, User.class });
    }

    private final ProjectRepository projectRepository;
    private final UserRepository userRepository;
    private final User defaultUser;
    private final String projectNameOverride;

    /**
     * @param projectRepository
     * @param userRepository
     * @param defaultUser
     */

```

```
* @param projectNameOverride
*/
public UnmarshallerListener(ProjectRepository projectRepository,
UserRepository userRepository,
    User defaultUser, String projectNameOverride) {
this.projectRepository = projectRepository;
this.userRepository = userRepository;
this.defaultUser = defaultUser;
this.projectNameOverride = projectNameOverride;
}

@Override
public void beforeUnmarshal(Object target, Object parent) {
    super.beforeUnmarshal(target, parent);
    findAndCallUnmarshalMethod("beforeUnmarshal", target, parent);
}

@Override
public void afterUnmarshal(Object target, Object parent) {
    super.afterUnmarshal(target, parent);
    findAndCallUnmarshalMethod("afterUnmarshal", target, parent);
}

/**
 * Search for a method with the supplied name and predefined possible
method
 * parameters, and if found call the method.
 *
 * @param methodName
 * @param target
 * @param parent
 */
private void findAndCallUnmarshalMethod(String methodName, Object
target, Object parent) {
    Class<?> targetType = target.getClass();
    Method afterUnmarshalMethod = null;
    Object[] params = null;
    try {
        // search for the method starting with the most specific class
        // and working up the inheritance hierarchy
        while (targetType != null) {
            for (Class<?>[] methodParamTypes :
beforeAndAfterUnmarshalMethodParams) {
                try {
                    afterUnmarshalMethod = targetType.getDeclaredMethod(methodName,
methodParamTypes);
                    afterUnmarshalMethod.setAccessible(true);
                }
            }
        }
    }
}
```

```

params = new Object[methodParamTypes.length];
for (int i = 0; i < methodParamTypes.length; i++) {
    if (methodParamTypes[i].equals(ProjectRepository.class)) {
        params[i] = projectRepository;
    } else if (methodParamTypes[i].equals(UserRepository.class)) {
        params[i] = userRepository;
    } else if (methodParamTypes[i].equals(Repository.class)) {
        params[i] = userRepository;
    } else if (methodParamTypes[i].equals(User.class)) {
        params[i] = defaultUser;
    } else if (methodParamTypes[i].equals(String.class)) {
        // NOTE: this assumes that only a project will
        // have an before/afterUnmarshal method that
        // takes a string.
        params[i] = projectNameOverride;
    } else if (methodParamTypes[i].equals(Object.class)) {
        params[i] = parent;
    }
}
afterUnmarshalMethod.invoke(target, params);
} catch (NoSuchMethodException e) {
} catch (SecurityException e) {
    log.error("The afterUnmarshal method for the class " +
targetType.getName()
        + " is not accessible.", e);
}
targetType = targetType.getSuperclass();
}
} catch (Exception e) {
    log.error("Exception processing afterUnmarshal(" + params + ") on
class " + targetType
        + " for object " + target, e);
}
}
}

```

untar.java

```
/*
 * $Id: Untar.java,v 1.1 2008/10/20 02:07:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requiel.utils;

import java.io.File;
```

```

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.zip.GZIPInputStream;

import org.apache.commons.compress.archivers.tar.TarEntry;
import org.apache.commons.compress.archivers.tar.TarInputStream;
import org.apache.commons.io.IOUtils;
import org.apache.log4j.Logger;

/**
 * Loosely based on the Apache Ant Untar class.
 *
 * @author ron
 */
public class Untar {
    private static final Logger log = Logger.getLogger(Untar.class);

    private Untar() {
    }

    /**
     * Extract the given tar file to the destination directory.
     *
     * @param archiveFile -
     *        a tar file that may be gzipped.
     * @param destinationDirectory -
     *        a directory to untar the file. If it doesn't exist it
     * will be
     *        created.
     * @throws IOException
     */
    public static void untar(File archiveFile, File destinationDirectory)
        throws IOException {
        log.debug("extracting " + archiveFile + " to " +
destinationDirectory);
        destinationDirectory.mkdirs();
        InputStream archiveInputStream = new FileInputStream(archiveFile);
        if (archiveFile.getName().endsWith(".gz") ||
archiveFile.getName().endsWith(".tgz")) {
            archiveInputStream = new GZIPInputStream(archiveInputStream);
        }
        TarInputStream tis = new TarInputStream(archiveInputStream);
        TarEntry entry = null;
        while ((entry = tis.getNextEntry()) != null) {
            log.debug("extracting " + entry.getName());
            extractFile(destinationDirectory, tis, entry.getName(),
entry.isDirectory());
        }
    }

    /**
     * extract a file to a directory
     *
     * @param destinationDir
     *        the root destination directory to untar the contents of
     * the
     *        archive.
     * @param inputStream
     *        the tar input stream ( apache commons-compress
     * TarInputStream)
     * @param newFileName
     *        the name of the new file or directory
     * @param entryIsDirectory
     *        if this is true the entry is a directory
     * @throws IOException
     *        on error
     */
    private static void extractFile(File destinationDir, TarInputStream
inputStream,
        String newFileName, boolean entryIsDirectory) throws IOException {
        File newFile = new File(destinationDir, newFileName);

        if (entryIsDirectory) {
            newFile.mkdirs();
        } else {
            // create intermediary directories - sometimes zip don't add them
            File newFileParentDir = newFile.getParentFile();
            if (newFileParentDir != null) {
                newFileParentDir.mkdirs();
            }
            FileOutputStream outputStream = null;
            try {
                outputStream = new FileOutputStream(newFile);
                inputStream.copyEntryContents(outputStream);
            } finally {
                IOUtils.closeQuietly(outputStream);
            }
        }
    }
}

```

updatedentitynotifier.java

```
/*
 * $Id: UpdatedEntityNotifier.java,v 1.1 2008/07/31 08:11:11 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;

/**
 * Used by assistants to indicate that an entity was changed.
 *
 * @author ron
 */
public interface UpdatedEntityNotifier {

    /**
     * An Assistant calls this method if it makes changes to a project,
     * including adding annotations to the project.
     *
     * @param pod -
     *          the changed project or domain
     */
    public void entityUpdated(ProjectOrDomain pod);

    /**
     * An Assistant calls this method if it makes changes to a project
     * entity,
     * such as a goal, including adding annotations to the entity.
     *
     * @param entity -
     *          the changed project entity
     */
    public void entityUpdated(ProjectOrDomainEntity entity);
}
```

updateentityevent.java

```
/*
 * $Id: UpdateEntityEvent.java,v 1.7 2008/09/12 01:00:57 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
```

```
/*
 * package edu.harvard.fas.rregan.uiframework.navigation.event;

import edu.harvard.fas.rregan.uiframework.panel.Panel;

/**
 * An event notification indicating that a data object has changed and
 * optionally close a panel.
 *
 * @author ron
 */
public class UpdateEntityEvent extends ClosePanelEvent {
    static final long serialVersionUID = 0;

    private final Object updatedObject;

    /**
     * Create an event notification from the specified panel, indicating
     * to
     * close that panel.
     *
     * @param source
     * @param updatedObject
     */
    public UpdateEntityEvent(Panel source, Object updatedObject) {
        this(source, source, updatedObject);
    }

    public UpdateEntityEvent(Object source, Panel panelToClose, Object
updatedObject) {
        this(source, UpdateEntityEvent.class.getName(), panelToClose,
updatedObject);
    }

    // note the destinationObject parameter to the ClosePanelEvent super
    constructor should
    // always be null so that all update listeners for an object get the
    event.
    protected UpdateEntityEvent(Object source, String command, Panel
panelToClose,
        Object updatedObject) {
        super(source, command, panelToClose, null);
        this.updatedObject = updatedObject;
    }

    public Object getObject() {
        return updatedObject;
    }
}
```

```
}
```

usecase.java

```
/*
 * $Id: UseCase.java,v 1.8 2008/10/10 07:02:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.project;

/**
 * @author ron
 */
public interface UseCase extends TextEntity, GoalContainer,
ActorContainer, StoryContainer,
Comparable<UseCase> {

/**
 * @return
 */
public Actor getPrimaryActor();

/**
 * @return
 */
public Scenario getScenario();
}
```

usecaseassistant.java

```
/*
 * $Id: UseCaseAssistant.java,v 1.6 2009/01/23 09:54:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.project.impl.assistant;

import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * Analyses a use case and its elements and adds annotations with
suggestions.
 *
```

```
 * @author ron
 */
public class UseCaseAssistant extends TextEntityAssistant {

private final ScenarioAssistant scenarioAssistant;
private final ActorAssistant actorAssistant;

/***
 * @param lexicalAssistant -
 *           assistant for analyzing text for spelling, terms and
other
 *           word oriented analysis.
 * @param scenarioAssistant -
 *           assistant for analyzing the use cases scenario.
 * @param actorAssistant -
 *           assistant for analyzing the primary actor.
 * @param updatedEntityNotifier -
 *           after an entity is analyzed it is passed to the
notifier to
 *           tell the UI components that reference the entity to
refresh
 * @param assistantUser -
 *           the user to use as the creator of the annotation
entities.
 */
public UseCaseAssistant(LexicalAssistant lexicalAssistant,
ScenarioAssistant scenarioAssistant,
ActorAssistant actorAssistant, User assistantUser) {
super(UseCaseAssistant.class.getName(), lexicalAssistant,
assistantUser);
this.scenarioAssistant = scenarioAssistant;
this.actorAssistant = actorAssistant;
}

/***
 * @param usecase
 */
@Override
public void analyze() {
super.analyze(); // analyze the name and text
if (getEntity() instanceof UseCase) {
UseCase useCase = (UseCase) getEntity();
actorAssistant.setEntity(useCase.getPrimaryActor());
actorAssistant.analyze();
scenarioAssistant.setEntity(useCase.getScenario());
scenarioAssistant.analyze();
}
}
```

```
}
```

usecaseeditorpanel.java

```
/*
 * $Id: UseCaseEditorPanel.java,v 1.27 2009/03/05 08:50:46 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.MessageFormat;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Button;
import nextapp.echo2.app.TextArea;
import nextapp.echo2.app.TextField;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.ComboBox;
import echopointng.text.StringDocumentEx;
import echopointng.tree.DefaultTreeModel;
import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.ActorContainer;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.GoalContainer;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.StoryContainer;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import
edu.harvard.fas.rregan.requel.project.command.CopyUseCaseCommand;
```

```
import
edu.harvard.fas.rregan.requel.project.command.DeleteUseCaseCommand;
import edu.harvard.fas.rregan.requel.project.command.EditStoryCommand;
import
edu.harvard.fas.rregan.requel.project.command.EditUseCaseCommand;
import
edu.harvard.fas.rregan.requel.project.command.ProjectCommandFactory;
import edu.harvard.fas.rregan.requel.ui.annotation.AnnotationsTable;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedListModel;
import
edu.harvard.fas.rregan.uiframework.panel.editor.CombinedTextListModel;

/**
 * @author ron
 */
public class UseCaseEditorPanel extends
AbstractRequelProjectEditorPanel {
private static final Logger log =
Logger.getLogger(UseCaseEditorPanel.class);

static final long serialVersionUID = 0L;

/**
 * The name to use in the UseCaseEditorPanel.properties file to set
the
 * label of the name field. If the property is undefined "Name" is
used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the UseCaseEditorPanel.properties file to set
the
 * label of the text field. If the property is undefined "Text" is
used.
 */
public static final String PROP_LABEL_TEXT = "Text.Label";
```

```

/**
 * The name to use in the UseCaseEditorPanel.properties file to set
the
 * label of the usecase type field. If the property is undefined
"UseCase
 * Type" is used.
 */
public static final String PROP_LABEL_STORY_TYPE =
"UseCaseType.Label";

/**
 * The name to use in the UseCaseEditorPanel.properties file to set
the
 * label of the primary actor field. If the property is undefined
"Primary
 * Actor" is used.
 */
public static final String PROP_LABEL_PRIMARY_ACTOR =
"PrimaryActor.Label";

private UpdateListener updateListener;
private Button copyButton;

// this is set by the DeleteListener so that the UpdateListener can
ignore
// events between when the object was deleted and the panel goes
away.
private boolean deleted;

/**
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository
 */
public UseCaseEditorPanel(CommandHandler commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    this(UseCaseEditorPanel.class.getName(), commandHandler,
        projectCommandFactory,
        projectRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param projectCommandFactory
 * @param projectRepository

```

```

 */
public UseCaseEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
    ProjectCommandFactory projectCommandFactory, ProjectRepository
projectRepository) {
    super(resourceBundleName, UseCase.class, commandHandler,
        projectCommandFactory,
        projectRepository);
}

/**
 * If the editor is editing an existing UseCase the title specified
in the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "UseCase: {0}"<br>
 * Valid variables are:<br>
 * {0} - UseCase name<br>
 * {1} - project/domain name<br>
 * For new UseCase it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
 * PROP_PANEL_TITLE and finally defaults to:<br>
 * "New UseCase"<br>
 *
 * @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
 * @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    if (getUseCase() != null) {
        String msgPattern = getResourceBundleHelpergetLocale().getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelpergetLocale()
                .getString(PROP_PANEL_TITLE, "UseCase: {0}"));
        return MessageFormat.format(msgPattern, getUseCase().getName(),
            getProjectOrDomain()
                .getName());
    } else {
        String msg = getResourceBundleHelpergetLocale()
            .getString(
                PROP_NEW_OBJECT_PANEL_TITLE,

```

```

        getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE
,
        "New UseCase");
    return msg;
}
}

@Override
public void setup() {
    super.setup();
    UseCase usecase = getUseCase();
    if (usecase != null) {
        addInput(EditUseCaseCommand.FIELD_NAME, PROP_LABEL_NAME, "Name",
new TextField(),
        new StringDocumentEx(usecase.getName()));
        addInput(EditUseCaseCommand.FIELD_TEXT, PROP_LABEL_TEXT, "Text",
new TextArea(),
        new StringDocumentEx(usecase.getText()));
        addInput("primaryActor", PROP_LABEL_PRIMARY_ACTOR, "Primary Actor",
new ComboBox(),
        new CombinedTextListModel(getActorNames(),
usecase.getPrimaryActor().getName()));
        addMultiRowInput("glossaryTerms",
GlossaryTermsTable.PROP_LABEL_GLOSSARY_TERM,
        "Glossary Terms", new GlossaryTermsTable(this,
            getResourceBundleHelpergetLocale()), usecase);
        addMultiRowInput("scenarioSteps",
ScenarioStepsEditor.PROP_LABEL_SCENARIO_STEPS,
        "Scenario", new ScenarioStepsEditor(getProjectOrDomain(), this,
            getResourceBundleHelpergetLocale()), usecase.getScenario());
        setInputValue("scenarioSteps", usecase.getScenario());
        addMultiRowInput("goals", GoalsTable.PROP_LABEL_GOALS, "Goals",
new GoalsTable(this,
            getResourceBundleHelpergetLocale()), getProjectCommandFactory(),
            getCommandHandler(), usecase);
        addMultiRowInput("actors", ActorsTable.PROP_LABEL_ACTORS, "Actors",
new ActorsTable(
            this, getResourceBundleHelpergetLocale(),
            getProjectCommandFactory(),
            getCommandHandler(), null));
        addMultiRowInput("stories", StoriesTable.PROP_LABEL_STORIES,
"Stories",
            new StoriesTable(this, getResourceBundleHelpergetLocale(),
            getProjectCommandFactory(), getCommandHandler(), usecase));
        addMultiRowInput("annotations",
AnnotationsTable.PROP_LABEL_ANNOTATIONS, "Annotations",
            new AnnotationsTable(this, getResourceBundleHelpergetLocale(),
null));
    }
}

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,

```

```

        updateListener);
    }
    updateListener = new UpdateListener(this);
    getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.class, updateListener);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void cancel() {
    super.cancel();
    // TODO: should this call dispose() or will dispose be called already
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
            updateListener);
    }
}

@Override
public void save() {
    try {
        super.save();
        ScenarioStepsEditor scenarioEditor = (ScenarioStepsEditor)
getInput("scenarioSteps");
        DefaultTreeModel treeModel = getInputModel("scenarioSteps",
DefaultTreeModel.class);
        MutableTreeNode rootNode = (MutableTreeNode) treeModel.getRoot();

        EditUseCaseCommand command =
getProjectCommandFactory().newEditUseCaseCommand();
        command.setUseCase(getUseCase());
        command.setProjectOrDomain(getProjectOrDomain());
        command.setEditedBy(getCurrentUser());
    }
}

```

```

        command.setName(getInputValue(EditUseCaseCommand.FIELD_NAME,
String.class));
        command.setText(getInputValue(EditUseCaseCommand.FIELD_TEXT,
String.class));
        command.setPrimaryActorName(getInputValue("primaryActor",
String.class));
        command.setStepCommands(scenarioEditor.generateStepEditCommands(
            getProjectCommandFactory(), getProjectOrDomain(), rootNode));
        command = getCommandHandler().execute(command);
        setValid(true);
        if (updateListener != null) {
            getEventDispatcher().removeEventTypeActionListener(UpdateEntityEvent.class,
                updateListener);
            // TODO: remove other listeners?
        }
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
command.getUseCase()));
    } catch (EntityException e) {
        if (e.isStaleEntity()) {
            // TODO: compare the original values before the user edited
            // to the current revisions values and if they are the same
            // then update the new revision with the user's changes and
            // continue, otherwise show the new changed value vs. the users
            // new values.
            String newName = getInputValue(EditUseCaseCommand.FIELD_NAME,
String.class);
            String newText = getInputValue(EditStoryCommand.FIELD_TEXT,
String.class);
            UseCase newUseCase = getProjectRepository().get(getUseCase());

            setTargetObject(newUseCase);
            if (!newName.equals(newUseCase.getName())) {
                setGeneralMessage("The usecase was changed by another user and
the value conflicts with your input.");
                if (!newName.equals(newUseCase.getName())) {
                    setValidationMessage(EditUseCaseCommand.FIELD_NAME, "Your input
"
                        + newName + "'");
                    setInputValue(EditUseCaseCommand.FIELD_NAME,
newUseCase.getName());
                }
                if (!newText.equals(newUseCase.getText())) {
                    setValidationMessage(EditUseCaseCommand.FIELD_TEXT, "Your input
"
                        + newText + "'");
                }
            }
        }
    }
}

```

```

        setInputValue(EditUseCaseCommand.FIELD_TEXT,
newUseCase.getText());
    }
} else {
    getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
newUseCase));
}

} else if ((e.getEntityPropertyNames() != null)
&& (e.getEntityPropertyNames().length > 0)) {
for (String propertyName : e.getEntityPropertyNames()) {
    setValidationMessage(propertyName, e.getMessage());
}
} else if ((e.getCause() != null) && (e.getCause() instanceof
InvalidStateException)) {
    InvalidStateException ise = (InvalidStateException) e.getCause();
    for (InvalidValue invalidValue : ise.getInvalidValues()) {
        String propertyName = invalidValue.getPropertyName();
        setValidationMessage(propertyName, invalidValue.getMessage());
    }
} else {
    setGeneralMessage(e.toString());
}

} catch (Exception e) {
log.error("could not save the usecase: " + e, e);
setGeneralMessage("Could not save: " + e);
}
}

@Override
public void delete() {
try {
    DeleteUseCaseCommand deleteUseCaseCommand =
getProjectCommandFactory()
    .newDeleteUseCaseCommand();
    deleteUseCaseCommand.setEditedBy(getCurrentUser());
    deleteUseCaseCommand.setUseCase(getUseCase());
    deleteUseCaseCommand =
getCommandHandler().execute(deleteUseCaseCommand);
    deleted = true;
    getEventDispatcher().dispatchEvent(new DeletedEntityEvent(this,
getUseCase()));
} catch (Exception e) {
    setGeneralMessage("Could not delete entity: " + e);
}
}

```

```

private Set<String> getActorNames() {
Set<String> actorNames = new TreeSet<String>();
for (Actor actor : getProjectOrDomain().getActors()) {
    actorNames.add(actor.getName());
}
return actorNames;
}

private ProjectOrDomain getProjectOrDomain() {
if (getTargetObject() instanceof ProjectOrDomain) {
    return (ProjectOrDomain) getTargetObject();
} else if (getTargetObject() instanceof ProjectOrDomainEntity) {
    return ((ProjectOrDomainEntity)
getTargetObject()).getProjectOrDomain();
}
return null;
}

private UseCase getUseCase() {
if (getTargetObject() instanceof UseCase) {
    return (UseCase) getTargetObject();
}
return null;
}

private static class CopyListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final UseCaseEditorPanel panel;

    private CopyListener(UseCaseEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            CopyUseCaseCommand copyUseCaseCommand =
panel.getProjectCommandFactory()
    .newCopyUseCaseCommand();
            copyUseCaseCommand.setEditedBy(panel.getCurrentUser());
            copyUseCaseCommand.setOriginalUseCase(panel.getUseCase());
            copyUseCaseCommand =
panel.getCommandHandler().execute(copyUseCaseCommand);
            panel.getEventDispatcher().dispatchEvent(

```

```

        new UpdateEntityEvent(this, null,
copyUseCaseCommand.getNewUseCase()));
    panel.getEventDispatcher().dispatchEvent(
        new OpenPanelEvent(this, PanelActionType.Editor,
copyUseCaseCommand
            .getNewUseCase(), UseCase.class, null));
} catch (Exception e) {
    panel.setGeneralMessage("Could not copy entity: " + e);
}
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final UseCaseEditorPanel panel;

    private UpdateListener(UseCaseEditorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (panel.deleted) {
            return;
        }
        UseCase existingUseCase = panel.getUseCase();
        if ((e instanceof UpdateEntityEvent) && (existingUseCase != null))
{
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            UseCase updatedUseCase = null;
            if (event.getObject() instanceof UseCase) {
                updatedUseCase = (UseCase) event.getObject();
                if ((event instanceof DeletedEntityEvent)
                    && existingUseCase.equals(updatedUseCase)) {
                    panel.deleted = true;
                    panel.getEventDispatcher().dispatchEvent(
                        new DeletedEntityEvent(this, panel, existingUseCase));
                    return;
                }
            } else if (event.getObject() instanceof Goal) {
                Goal updatedGoal = (Goal) event.getObject();
                if (event instanceof DeletedEntityEvent) {
                    if (existingUseCase.getGoals().contains(updatedGoal)) {
                        existingUseCase.getGoals().remove(updatedGoal);
                    }
                }
                updatedUseCase = existingUseCase;
            }
        }
    }
}

```

```

} else if (updatedGoal.getReferers().contains(existingUseCase)) {
for (GoalContainer gc : updatedGoal.getReferers()) {
    if (gc.equals(existingUseCase)) {
        updatedUseCase = (UseCase) gc;
        break;
    }
}
} else if (event.getObject() instanceof Actor) {
Actor updatedActor = (Actor) event.getObject();
panel.setInputModel("primaryActor", new CombinedListModel(
    panel.getActorNames(), panel
        .getInputValue("primaryActor", String.class), true));
if (event instanceof DeletedEntityEvent) {
    if (existingUseCase.getActors().contains(updatedActor)) {
        existingUseCase.getActors().remove(updatedActor);
    }
    updatedUseCase = existingUseCase;
} else if (updatedActor.getReferers().contains(existingUseCase)) {

    for (ActorContainer ac : updatedActor.getReferers()) {
        if (ac.equals(existingUseCase)) {
            updatedUseCase = (UseCase) ac;
            break;
        }
    }
}
} else if (event.getObject() instanceof Story) {
Story updatedStory = (Story) event.getObject();
if (event instanceof DeletedEntityEvent) {
    if (existingUseCase.getStories().contains(updatedStory)) {
        existingUseCase.getStories().remove(updatedStory);
    }
    updatedUseCase = existingUseCase;
} else if (updatedStory.getReferers().contains(existingUseCase)) {

    for (StoryContainer ac : updatedStory.getReferers()) {
        if (ac.equals(existingUseCase)) {
            updatedUseCase = (UseCase) ac;
            break;
        }
    }
}
} else if (event.getObject() instanceof Annotation) {
Annotation updatedAnnotation = (Annotation) event.getObject();
if (event instanceof DeletedEntityEvent) {

```

```

if
(existingUseCase.getAnnotations().contains(updatedAnnotation)) {
    existingUseCase.getAnnotations().remove(updatedAnnotation);
}
updatedUseCase = existingUseCase;
} else if
(updatedAnnotation.getAnnotatables().contains(existingUseCase)) {
    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
        if (annotatable.equals(existingUseCase)) {
            updatedUseCase = (UseCase) annotatable;
            break;
        }
    }
}
if ((updatedUseCase != null) &&
updatedUseCase.equals(existingUseCase)) {
    // TODO: check the input fields to see if the user has made
    // a change before resetting the object and updating the
    // input fields.
    panel.setInputValue(EditUseCaseCommand.FIELD_NAME,
updatedUseCase.getName());
    panel.setInputValue("primaryActor",
updatedUseCase.getPrimaryActor().getName());
    panel.setInputValue(EditUseCaseCommand.FIELD_TEXT,
updatedUseCase.getText());
    panel.setInputValue("glossaryTerms", updatedUseCase);
    panel.setInputValue("goals", updatedUseCase);
    panel.setInputValue("actors", updatedUseCase);
    panel.setInputValue("stories", updatedUseCase);
    panel.setInputValue("annotations", updatedUseCase);
    panel.setTargetObject(updatedUseCase);
}
}
}
}

usecaseimpl.java
*/
/* $Id: UseCaseImpl.java,v 1.14 2009/01/23 09:54:27 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.project.impl;

import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlIDREF;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;
import org.hibernate.validator.NotEmpty;
import org.xml.sax.SAXException;

import com.sun.istack.SAXException2;
import com.sun.xml.bind.v2.runtime.unmarshaller.Patcher;
import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.requel.project.Actor;
import edu.harvard.fas.rregan.requel.project.Goal;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.Scenario;
import edu.harvard.fas.rregan.requel.project.Story;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBCreatedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**

```

```

* @author ron
*/
@Entity
@Table(name = "usecases", uniqueConstraints =
{ @UniqueConstraint(columnNames =
"projectordomain_id", "name" ) })
@XmlRootElement(name = "usecase", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "usecase", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class UseCaseImpl extends AbstractTextEntity implements UseCase
{
    static final long serialVersionUID = 0L;

    private Actor primaryActor;
    private Set<Goal> goals = new TreeSet<Goal>();
    private Set<Actor> actors = new TreeSet<Actor>();
    private Set<Story> stories = new TreeSet<Story>();
    private Scenario scenario;

    /**
     * Create a use case.
     *
     * @param projectOrDomain
     * @param primaryActor
     * @param createdBy
     * @param name
     * @param text -
     *           short text description\
     * @param scenario
     */
    public UseCaseImpl(ProjectOrDomain projectOrDomain, Actor
primaryActor, User createdBy,
        String name, String text, Scenario scenario) {
        super(projectOrDomain, createdBy, name, text);
        setPrimaryActor(primaryActor);
        setScenario(scenario);
        projectOrDomain.getUseCases().add(this);
    }

    protected UseCaseImpl() {
        // for hibernate
    }

    @Override
    @Column(nullable = false, unique = false)
    @NotEmpty(message = "a unique name is required.")

```

```

    @XmlElement(name = "name", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
    public String getName() {
        return super.getName();
    }

    // hack for JAXB to set the name, for some reason it won't use the
    inherited
    // method.
    @Override
    public void setName(String name) {
        super.setName(name);
    }

    /**
     * @see
     edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity#getXmlId()
     */
    @Transient
    @XmlID
    @XmlAttribute(name = "id")
    @Override
    public String getXmlId() {
        return "UC_" + getId();
    }

    /**
     * @see edu.harvard.fas.rregan.requel.Describable#getDescription()
     */
    @XmlTransient
    @Transient
    @Override
    public String getDescription() {
        // TODO Auto-generated method stub
        return "UseCase: " + getName();
    }

    /**
     * @see
     edu.harvard.fas.rregan.requel.project.UseCase#getPrimaryActor()
     */
    @XmlIDREF
    @XmlElement(name = "primaryActorRef", type = ActorImpl.class,
    namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
    @ManyToOne(targetEntity = ActorImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
    CascadeType.REFRESH }, optional = false)

```

```

@Override
public Actor getPrimaryActor() {
    return primaryActor;
}

/**
 * @param primaryActor
 */
public void setPrimaryActor(Actor primaryActor) {
    this.primaryActor = primaryActor;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.GoalContainer#getGoals()
 */
@Override
@XmlElementWrapper(name = "goals", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "goalRef", type = GoalImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = GoalImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "usecase_goals", joinColumns = { @JoinColumn(name =
"usecase_id") }, inverseJoinColumns = { @JoinColumn(name =
"goal_id") })
@Sort(type = SortType.NATURAL)
public Set<Goal> getGoals() {
    return goals;
}

protected void setGoals(Set<Goal> goals) {
    this.goals = goals;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.ActorContainer#getActors()
 */
@Override
@XmlElementWrapper(name = "actors", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "actorRef", type = ActorImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = ActorImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "usecase_actors", joinColumns = { @JoinColumn(name =
"usecase_id") }, inverseJoinColumns = { @JoinColumn(name =
"actor_id") })
@Sort(type = SortType.NATURAL)
public Set<Actor> getActors() {
    return actors;
}

protected void setActors(Set<Actor> actors) {
    this.actors = actors;
}

/**
 * @see
edu.harvard.fas.rregan.requel.project.StoryContainer#getStories()
 */
@Override
@XmlElementWrapper(name = "stories", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlIDREF
@XmlElement(name = "storyRef", type = StoryImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToMany(targetEntity = StoryImpl.class, cascade =
{ CascadeType.MERGE, CascadeType.PERSIST,
  CascadeType.REFRESH }, fetch = FetchType.LAZY)
@JoinTable(name = "usecase_stories", joinColumns = { @JoinColumn(name =
"usecase_id") }, inverseJoinColumns = { @JoinColumn(name =
"story_id") })
@Sort(type = SortType.NATURAL)
public Set<Story> getStories() {
    return stories;
}

protected void setStories(Set<Story> stories) {
    this.stories = stories;
}

/**
 */
@Override
@XmlIDREF
@XmlElement(name = "scenarioRef", type = ScenarioImpl.class,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")

```

```
@ManyToOne(targetEntity = ScenarioImpl.class, cascade =
{ CascadeType.MERGE,
  CascadeType.PERSIST, CascadeType.REFRESH }, fetch = FetchType.LAZY)
public Scenario getScenario() {
    return scenario;
}

protected void setScenario(Scenario scenario) {
    this.scenario = scenario;
}

@Override
public int compareTo(UseCase o) {
    return getName().compareToIgnoreCase(o.getName());
}

/**
 * This is for JAXB to patchup the parent/child relationship and to
patchup
 * existing persistent objects for the objects that are attached
directly to
 * this object.
 *
 * @param userRepository
 * @param defaultCreatedByUser -
 *           the user to be set as the created by if no user is
supplied.
 * @see UnmarshallerListener
 */
public void afterUnmarshal(final UserRepository userRepository, User
defaultCreatedByUser) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBCreatedEntityPatcher(userRepository, this,
defaultCreatedByUser));
    UnmarshallingContext.getInstance().addPatcher(new Patcher() {
        @Override
        public void run() throws SAXException {
            try {
                for (Goal goal : getGoals()) {
                    goal.getReferers().add(UseCaseImpl.this);
                }
                for (Actor actor : getActors()) {
                    actor.getReferers().add(UseCaseImpl.this);
                }
                if (getPrimaryActor() != null) {
                    getPrimaryActor().getReferers().add(UseCaseImpl.this);
                }
            }
        }
    });
}
```

```
        for (Story story : getStories()) {
            story.getReferers().add(UseCaseImpl.this);
        }
    } catch (RuntimeException e) {
        throw e;
    } catch (Exception e) {
        throw new SAXException2(e);
    }
}
});
```

usecasenavigatorpanel.java

```
/*
 * $Id: UseCaseNavigatorPanel.java,v 1.3 2009/03/26 10:31:21 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requiel.ui.project;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;
import nextapp.echo2.app.layout.RowLayoutData;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requiel.annotation.Annotatable;
import edu.harvard.fas.rregan.requiel.annotation.Annotation;
import edu.harvard.fas.rregan.requiel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requiel.project.UseCase;
import edu.harvard.fas.rregan.requiel.project.Project;
import edu.harvard.fas.rregan.requiel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requiel.project.Stakeholder;
import
edu.harvard.fas.rregan.requiel.project.StakeholderPermissionType;
```

```

import
edu.harvard.fas.rregan.requel.project.impl.AbstractProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTablePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * TODO: integrate the UseCasesTable with this.
 *
 * @author ron
 */
public class UseCaseNavigatorPanel extends NavigatorTablePanel {
    private static final Logger log =
Logger.getLogger(UseCaseNavigatorPanel.class);
    static final long serialVersionUID = 0;

    private UpdateListener updateListener;
    private ProjectOrDomain pod;

    /**
     * Property name to use in the UseCaseNavigatorPanel.properties to
     * set the
     * * label on the new usecase button.
     */
    public static final String PROP_NEW_USE_CASE_BUTTON_LABEL =
"NewButton.Label";

    /**
     * Property name to use in the UseCaseNavigatorPanel.properties to
     * set the
     * * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * Property name to use in the UseCaseNavigatorPanel.properties to
     * set the
     * * label on the edit usecase button in each row of the table.
     */
    public static final String PROP_EDIT_USE_CASE_BUTTON_LABEL =
>EditButton.Label";

    /**
     * Property name to use in the UseCaseNavigatorPanel.properties to
     * set the
     * * label on the view usecase button in each row of the table when the
     * user
     * * doesn't have edit permission.
     */
    public static final String PROP_VIEW_USE_CASE_BUTTON_LABEL =
"ViewButton.Label";

    /**
     */
    public UseCaseNavigatorPanel() {
        super(UseCaseNavigatorPanel.class.getName(), Project.class,
              ProjectManagementPanelNames.PROJECT_USE_CASES_NAVIGATOR_PANEL_NAME
        );
        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("",

new NavigatorTableCellValueFactory() {
            @Override
            public Object getValueAt(NavigatorTableModel model, int column,
int row) {
                UseCase usecase = (UseCase) model.getBackingObject(row);

```

```

String buttonLabel = null;
if (isReadOnlyMode()) {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_VIEW_USE_CASE_BUTTON_LABEL, "View");
} else {
    buttonLabel = getResourceBundleHelper(getLocale()).getString(
        PROP_EDIT_USE_CASE_BUTTON_LABEL, "Edit");
}
NavigationEvent openEditorEvent = new OpenPanelEvent(this,
    PanelActionType.Editor, usecase, UseCase.class, null,
    WorkflowDisposition.NewFlow);
NavigatorButton openEditorButton = new
NavigatorButton(buttonLabel,
    getEventDispatcher(), openEditorEvent);
openEditorButton.setStyleName(STYLE_NAME_PLAIN);
RowLayoutData rld = new RowLayoutData();
rld.setAlignment(Alignment.ALIGN_CENTER);
openEditorButton.setLayoutData(rld);
return openEditorButton;
});
});

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            UseCase usecase = (UseCase) model.getBackingObject(row);
            return usecase.getName();
        }
    });
);

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Primary
Actor",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            UseCase usecase = (UseCase) model.getBackingObject(row);
            if (usecase.getPrimaryActor() != null) {
                return usecase.getPrimaryActor().getName();
            }
            return "<Not Selected>";
        }
    });
);

    tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            return entity.getCreatedBy().getUsername();
        }
    }));
}

tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
    new NavigatorTableCellValueFactory() {
        @Override
        public Object getValueAt(NavigatorTableModel model, int column,
        int row) {
            AbstractProjectOrDomainEntity entity =
(AbstractProjectOrDomainEntity) model
                .getBackingObject(row);
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
            return format.format(entity.getDateCreated());
        }
    }));
}

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
or
 * domain, by default the pattern is "UseCases: {0}"<br>
 * Valid variables are:<br>
 * {0} - project/domain name<br>
 *
 * @see Panel.PROP_PANEL_TITLE
 * @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
 */
@Override
public String getTitle() {
    String name = "";
    String msgPattern =
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
    "UseCases: {0}");
}

```

```

ProjectOrDomain pod = getProjectOrDomain();
if (pod != null) {
    name = pod.getName();
}
return MessageFormat.format(msgPattern, name);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
    if (updateListener != null) {
        getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
            updateListener);
        updateListener = null;
    }
}

@Override
public void setup() {
    super.setup();

    Row buttonsWrapper = new Row();
    buttonsWrapper.setInsets(new Insets(10, 5));
    buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

    String closeButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
    NavigationEvent closeEvent = new ClosePanelEvent(this, this);
    NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
    closeButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(closeButton);

    if (!isReadOnlyMode()) {
        String newUseCaseButtonLabel =
getResourceBundleHelpergetLocale()).getString(
    PROP_NEW_USE_CASE_BUTTON_LABEL, "Add");
        NavigationEvent openUseCaseEditor = new OpenPanelEvent(this,
PanelActionType.Editor,
        getProjectOrDomain(), UseCase.class, null,
WorkflowDisposition.NewFlow);
    }
}

```

```

    NavigatorButton newUseCaseButton = new
NavigatorButton(newUseCaseButtonLabel,
    getEventDispatcher(), openUseCaseEditor);
    newUseCaseButton.setStyleName(STYLE_NAME_DEFAULT);
    buttonsWrapper.add(newUseCaseButton);
}

add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}

protected boolean isReadOnlyMode() {
    User user = (User) getApp().getUser();
    if (getProjectOrDomain() instanceof Project) {
        Project project = (Project) getProjectOrDomain();
        Stakeholder stakeholder = project.getUserStakeholder(user);
        if (stakeholder != null) {
            return !stakeholder.hasPermission(UseCase.class,
StakeholderPermissionType.Edit);
        }
    }
    return true;
}

@Override
public void setTargetObject(Object targetObject) {
    if (targetObject instanceof ProjectOrDomain) {
        pod = (ProjectOrDomain) targetObject;
        super.setTargetObject(((ProjectOrDomain)
targetObject).getUseCases());
    } else {
        log.error("unexpected target object " + targetObject);
    }
}

protected ProjectOrDomain getProjectOrDomain() {
    return pod;
}

```

```

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final UseCaseNavigatorPanel panel;

    private UpdateListener(UseCaseNavigatorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
                    event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
                        updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
                            && annotatable.equals(panel.getProjectOrDomain())) {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
                                annotatable;
                            if
                                (entity.getProjectOrDomain().equals(panel.getProjectOrDomain())))
                            {
                                updatedPod = entity.getProjectOrDomain();
                                break;
                            }
                        }
                    }
                }
            }
            if (panel.getProjectOrDomain().equals(updatedPod)) {
                panel.setTargetObject(updatedPod);
            }
        }
    }
}

```

```

}
```

usecaseselectorpanel.java

```

/*
 * $Id: UseCaseSelectorPanel.java,v 1.3 2009/02/23 07:37:24 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.project;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Collection;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;
import nextapp.echo2.app.event.ActionEvent;
import nextapp.echo2.app.event.ActionListener;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.annotation.Annotatable;
import edu.harvard.fas.rregan.requel.annotation.Annotation;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomainEntity;
import edu.harvard.fas.rregan.requel.project.UseCase;
import edu.harvard.fas.rregan.requel.project.Project;
import edu.harvard.fas.rregan.requel.project.ProjectOrDomain;
import edu.harvard.fas.rregan.requel.project.ProjectRepository;
import edu.harvard.fas.rregan.requel.project.Stakeholder;
import
edu.harvard.fas.rregan.requel.project.StakeholderPermissionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.event.ClosePanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.DeletedEntityEvent
;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.SelectEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableCell
ValueFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableColu
mnConfig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableConf
ig;
import
edu.harvard.fas.rregan.uiframework.navigation.table.NavigatorTableModel
;
import
edu.harvard.fas.rregan.uiframework.panel.NavigatorTableModelAdapter;
import edu.harvard.fas.rregan.uiframework.panel.SelectorTablePanel;

/**
 * @author ron
 */
public class UseCaseSelectorPanel extends SelectorTablePanel {
    private static final Logger log =
Logger.getLogger(UseCaseSelectorPanel.class);
    static final long serialVersionUID = 0;

    private final ProjectRepository projectRepository;
    private UpdateListener updateListener;

    /**
     * Property name to use in the UseCaseNavigatorPanel.properties to
     * set the
     *   * label for the text of the cancel/reset button.
     */
    public static final String PROP_CANCEL_BUTTON_LABEL =
"CancelButton.Label";

    /**
     * @param projectRepository
     */
    public UseCaseSelectorPanel(ProjectRepository projectRepository) {
        super(UseCaseSelectorPanel.class.getName(), Project.class,
              ProjectManagementPanelNames.PROJECT_USE_CASES_SELECTOR_PANEL_NAME)
        ;
        this.projectRepository = projectRepository;

        NavigatorTableConfig tableConfig = new NavigatorTableConfig();
        tableConfig.setRowLevelSelection(true);
    }
}

```

```

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Name",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        UseCase usecase = (UseCase) model.getBackingObject(row);
        return usecase.getName();
    }
}));

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Created
By",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        UseCase usecase = (UseCase) model.getBackingObject(row);
        return usecase.getCreatedBy().getUsername();
    }
}));

        tableConfig.addColumnConfig(new NavigatorTableColumnConfig("Date
Created",
new NavigatorTableCellValueFactory() {
    @Override
    public Object getValueAt(NavigatorTableModel model, int column,
int row) {
        UseCase usecase = (UseCase) model.getBackingObject(row);
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        return format.format(usecase.getDateCreated());
    }
}));
}

setTableConfig(tableConfig);
}

/**
 * Create a title for panel with dynamic information from the project
* or
* domain, by default the title is "Select UseCase"<br>
*
* @see Panel.PROP_PANEL_TITLE
* @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override

```

```

public String getTitle() {
    return
getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
"Select Use Case");
}

@Override
public void dispose() {
    super.dispose();
removeAll();
if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
    updateListener = null;
}
}

@Override
public void setup() {
super.setup();

Row buttonsWrapper = new Row();
buttonsWrapper.setInsets(new Insets(10, 5));
buttonsWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));

String closeButtonLabel =
getResourceBundleHelper(getLocale()).getString(
    PROP_CANCEL_BUTTON_LABEL, "Close");
NavigationEvent closeEvent = new ClosePanelEvent(this, this);
NavigatorButton closeButton = new NavigatorButton(closeButtonLabel,
getEventDispatcher(),
    closeEvent);
closeButton.setStyleName(STYLE_NAME_DEFAULT);
buttonsWrapper.add(closeButton);
add(buttonsWrapper);

if (updateListener != null) {
    getEventDispatcher().removeEventTypeActionListener(UpdateEntityEven
t.class,
        updateListener);
}
updateListener = new UpdateListener(this);
getEventDispatcher().addEventTypeActionListener(UpdateEntityEvent.cl
ass, updateListener);
}
}

```

```

protected boolean isReadOnlyMode() {
User user = (User) getApp().getUser();
if (getProjectOrDomain() instanceof Project) {
    Project project = (Project) getProjectOrDomain();
    Stakeholder stakeholder = project.getUserStakeholder(user);
    if (stakeholder != null) {
        return !stakeholder.hasPermission(UseCase.class,
StakeholderPermissionType.Edit);
    }
}
return true;
}

/**
 * This method should be overridden to return a collection when the
target
 * of the panel is not a collection.
 *
 * @return an adapter to get the collection of items to select from
from the
 * target object.
 */
@Override
protected NavigatorTableModelAdapter
getTargetNavigatorTableModelAdapter() {
return new NavigatorTableModelAdapter() {
private ProjectOrDomain targetObject;

@Override
public Collection<Object> getCollection() {
    return (Collection) targetObject.getUseCases();
}

@Override
public void setTargetObject(Object targetObject) {
    this.targetObject = (ProjectOrDomain) targetObject;
};

protected ProjectOrDomain getProjectOrDomain() {
    return (ProjectOrDomain) getTargetObject();
}

protected ProjectRepository getProjectRepository() {
    return projectRepository;
}
}

```

```
}

@Override
public void actionPerformed(ActionEvent e) {
    // before returning, initialize the usecase
    SelectEntityEvent selectEvent = new SelectEntityEvent(this,
getTable().getSelectedObject(),
    getDestinationObject());
    getEventDispatcher().dispatchEvent(selectEvent);
}

private static class UpdateListener implements ActionListener {
    static final long serialVersionUID = 0L;

    private final UseCaseSelectorPanel panel;

    private UpdateListener(UseCaseSelectorPanel panel) {
        this.panel = panel;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e instanceof UpdateEntityEvent) {
            UpdateEntityEvent event = (UpdateEntityEvent) e;
            ProjectOrDomain updatedPod = null;
            if (event.getObject() instanceof ProjectOrDomain) {
                updatedPod = (ProjectOrDomain) event.getObject();
            } else if (event.getObject() instanceof ProjectOrDomainEntity) {
                ProjectOrDomainEntity updatedEntity = (ProjectOrDomainEntity)
event.getObject();
                updatedPod = updatedEntity.getProjectOrDomain();
            } else if (event.getObject() instanceof Annotation) {
                if (!(event instanceof DeletedEntityEvent)) {
                    Annotation updatedAnnotation = (Annotation) event.getObject();
                    for (Annotatable annotatable :
updatedAnnotation.getAnnotatables()) {
                        if ((annotatable instanceof ProjectOrDomain)
                            && annotatable.equals(panel.getProjectOrDomain())) {
                            updatedPod = (ProjectOrDomain) annotatable;
                            break;
                        } else if ((annotatable instanceof ProjectOrDomainEntity)) {
                            ProjectOrDomainEntity entity = (ProjectOrDomainEntity)
annotatable;
                            if
(entity.getProjectOrDomain().equals(panel.getProjectOrDomain()))
                                updatedPod = entity.getProjectOrDomain();
                            break;
                        }
                    }
                }
            }
        }
    }
}
```

```
        }
    }
}
}
if (panel.getProjectOrDomain().equals(updatedPod)) {
    panel.setTargetObject(updatedPod);
}
}
}
}
}
```

user.java

```
/*
 * $Id: User.java,v 1.13 2008/08/17 03:29:07 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user;

import java.util.Comparator;
import java.util.Set;

import javax.xml.bind.ValidationException;

import edu.harvard.fas.rregan.requel.OrganizedEntity;
import
edu.harvard.fas.rregan.requel.user.exception.NoSuchRoleForUserException
n;

/**
 * A user of the system.
 *
 * @author ron
 */
public interface User extends Comparable<User>, OrganizedEntity {

/**
 * The users real name useful for identifying the user in documents
or
 * reports generated by the system.
 *
 * @return - the user's name
 */
public String getName();
```

```

/**
 * set the user's name.
 *
 * @param name -
 *          the new name of the user.
 */
public void setName(String name);

/**
 * A system-wide unique text identifier for the user.
 *
 * @return - the user's username TODO: what makes a valid username?
TODO:
 *      can a username be changed?
 */
public String getUsername();

/**
 * Set a new username for this user<br>
 * NOTE: if the new user name is in use by someone else, saving the
user
 * will fail.
 *
 * @param username -
 *          new username value.
 */
public void setUsername(String username);

/**
 * Reset the user's password to the supplied plain text string.
 *
 * @param password -
 *          the user's new password as plain text.
 * @throws ValidationException
 *          if the supplied string doesn't meet the password
criteria
 *      TODO: what is the password criteria?
 */
public void resetPassword(String password);

/**
 * Return true if the supplied plain text password matches the user's
 * password.
 *
 * @param password -
 *          a plain text string to test as the user's password.

```

```

 * @return true if the supplied plain text password matches the
user's
 *          password.
 */
public boolean isPassword(String password);

/**
 * The user's email address. The system will use this for sending
 * communications to the user and may be included in generated
documents or
 * presented to other users of the system for contacting this user.
 *
 * @return the user's email address
 */
public String getEmailAddress();

/**
 * Set the user's email address.
 *
 * @param emailAddress -
 *          the email address to assign
 * @throws ValidationException
 *          if the emailAddress is ill-formed
 */
public void setEmailAddress(String emailAddress);

/**
 * The user's primary contact number. This may be included in
generated
 * documents or presented to other users of the system for contacting
this
 * user.
 *
 * @return
 */
public String getPhoneNumber();

/**
 * Set the user's phone number.
 *
 * @param phoneNumber -
 *          the phone number to assign
 * @throws ValidationException
 *          if the phone number is ill-formed
 */
public void setPhoneNumber(String phoneNumber);

```

```

/**
 * The system administrator can create accounts intended to be shared
by
 * multiple users conceivably for reading the projects status. The
 * administrator can lock these accounts so they can not be edited by
the
 * user.
 *
 * @return true if the account is editable by the user, false if the
account
 *         can only be edited by the administrator
 */
public boolean isEditable();

/**
 * @param edited
 */
public void setEditable(boolean edited);

/**
 * The system level roles that the user has permission to use, such
as
 * project user, domain administrator or system administrator. The
role may
 * includ role specific data for this user, such as project
information.
 *
 * @return
 */
public Set<UserRole> getUserRoles();

/**
 * Return the role object for the specified type. A user can only
have one
 * role per type.
 *
 * @param <T> -
 *         the class of user role being retrieved
 * @param roleType -
 *         The type (class) of the role being requested.
 * @return the role object for the specified type
 * @throws NoSuchRoleForUserException -
 *         if the user doesn't have a role for the supplied type
 */
public <T extends UserRole> T getRoleForType(Class<T> roleType)
    throws NoSuchRoleForUserException;

```

```

/**
 * Return true if the user is assigned to the supplied role type.
 *
 * @param roleType -
 *         The UserRoleType of the role being tested.
 * @return - true if the user is assigned to the supplied role type.
 */
public boolean hasRole(Class<? extends UserRole> roleType);

/**
 * grant the specified role to the user.
 *
 * @param userRoleType -
 *         the type of role to grant
 */
public void grantRole(Class<? extends UserRole> userRoleType);

/**
 * revoke the specified role from the user.
 *
 * @param userRoleType -
 *         the type of role to revoke
 */
public void revokeRole(Class<? extends UserRole> userRoleType);

/**
 * Test if a user is equal to another user by internal id. This is
needed to
 * figure out if a user object is equal to another when the username
 * changes.
 *
 * @param other
 * @return
 */
public boolean equalsById(User other);

/**
 * a Comparator for comparing two users, ordered by username
 */
public static Comparator<User> UserComparator = new
Comparator<User>() {
    public int compare(User o1, User o2) {
        // this catches the case of when a user's username has changed
        // TODO: if the new username sorts before the original then the
        // username comparator may terminate the sorting before the actual
        // user is found.
        if (o1.equals(o2)) {

```

```

        return 0;
    }
    return UsernameComparator.compare(o1.getUsername(),
o2.getUsername());
}
};

/**
 * Compare two username strings.
 */
public static Comparator<String> UsernameComparator = new
Comparator<String>() {
    public int compare(String o1, String o2) {
        return o1.toLowerCase().compareTo(o2.toLowerCase());
    }
};
}

```

user2userimpladapter.java

```

/*
 * $Id: User2UserImplAdapter.java,v 1.2 2008/06/25 09:58:49 rregan Exp
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl;

import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.adapters.XmlAdapter;

import edu.harvard.fas.rregan.requel.user.User;

/**
 * Adapter for JAXB to convert interface User to class UserImpl and
back.
 *
 * @author ron
 */
@XmlTransient
public class User2UserImplAdapter extends XmlAdapter<UserImpl, User> {

    @Override
    public UserImpl marshal(User user) throws Exception {
        return (UserImpl) user;
    }
}

```

```

@Override
public User unmarshal(UserImpl user) throws Exception {
    return user;
}

}

```

useradminnavigatorpanel.java

```

/*
 * $Id: UserAdminNavigatorPanel.java,v 1.1 2008/09/12 22:44:14 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.user;

import java.util.Set;

import nextapp.echo2.app.Alignment;
import nextapp.echo2.app.Insets;
import nextapp.echo2.app.Row;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.uiframework.navigation.NavigatorButton;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory;
import edu.harvard.fas.rregan.uiframework.panel.NavigatorTreePanel;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
public class UserAdminNavigatorPanel extends NavigatorTreePanel {
    private static final Logger log =
Logger.getLogger(UserAdminNavigatorPanel.class);
}

```

```

static final long serialVersionUID = 0;

/**
 * Property name to use in the UserAdminNavigatorPanel.properties to
set the
 * lable on the new user button.
 */
public static final String PROP_NEW_USER_BUTTON_LABEL =
"NewUserButton.Label";

/**
 * @param treeNodeFactories
 */
public UserAdminNavigatorPanel(Set<NavigatorTreeNodeFactory>
treeNodeFactories) {
    super(UserAdminNavigatorPanel.class.getName(), treeNodeFactories,
SystemAdminUserRole.class);
}

@Override
public void dispose() {
    super.dispose();
    removeAll();
}

@Override
public void setup() {
    String newUserButtonLabel =
getResourceBundleHelper(getLocale()).getString(
PROP_NEW_USER_BUTTON_LABEL, "New User");

    NavigationEvent openUserEditor = new OpenPanelEvent(this,
PanelActionType.Editor, null,
    User.class, null, WorkflowDisposition.NewFlow);
    NavigatorButton newUserButton = new
NavigatorButton(newUserButtonLabel,
    getEventDispatcher(), openUserEditor);
    newUserButton.setStyleName(STYLE_NAME_DEFAULT);
    Row newUserButtonWrapper = new Row();
    newUserButtonWrapper.setInsets(new Insets(5));
    newUserButtonWrapper.setAlignment(new Alignment(Alignment.CENTER,
Alignment.DEFAULT));
    newUserButtonWrapper.add(newUserButton);
    add(newUserButtonWrapper);
    super.setup();
}
}

```

usercollectionnavigatortreenodefactory.java

```

/*
 * $Id: UserCollectionNavigatorTreeNodeFactory.java,v 1.1 2008/09/12
22:44:14 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.user;

import java.util.Enumeration;

import nextapp.echo2.app.Label;
import nextapp.echo2.app.event.ActionEvent;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserSet;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeUpdateListener;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNode;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

```

```

/**
 * @author ron
 */
@Component("userCollectionNavigatorTreeNodeFactory")
@Scope("singleton")
public class UserCollectionNavigatorTreeNodeFactory extends
AbstractNavigatorTreeNodeFactory {

    /**
     * The property name to use to control the label on the primary node
     * generated by the factory.
     */
    public final static String PROP_USERS_NODE_LABEL = "UsersNodeLabel";

    /**
     * @param eventDispatcher
     */
    public UserCollectionNavigatorTreeNodeFactory() {
        super(UserCollectionNavigatorTreeNodeFactory.class.getName(),
              UserSet.class);
    }

    /**
     * @see
     edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
     ctory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.tre
     e.NavigatorTree,
     *      java.lang.Object)
     */
    public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
        Object object) {
        UserSet users = (UserSet) object;
        String usersNodeLabel =
getResourceBundleHelper(tree.getLocale()).getString(
    PROP_USERS_NODE_LABEL, "Users");

        // TODO: fire an event to a controller that reloads the
        // users and then fires an open panel event? or have the
        // NavigatorTreeNodeUpdateListener reset the event when
        // users are updated
        NavigationEvent openUserList = new OpenPanelEvent(tree,
PanelActionType.Navigator, users,
        UserSet.class, null, WorkflowDisposition.NewFlow);
    }
}

```

```

    NavigatorTreeNode usersTreeNode = new
NavigatorTreeNode(eventDispatcher, users, new Label(
    usersNodeLabel), openUserList);

    usersTreeNode
        .setUpdateListener(new
NavigatorTreeNodeUserUpdateListener(usersTreeNode, tree));

    // TODO: if there are lots of users, create a nested tree with
    // intermediate nodes representing the starting letters of user
    names
    // such that a node has no more than N nodes and no nodes are empty.
    for (User user : users) {
        NavigatorTreeNodeFactory factory =
tree.getNavigatorTreeNodeFactory(user);
        usersTreeNode.add(factory.createTreeNode(eventDispatcher, tree,
user));
    }
    return usersTreeNode;
}

private static class NavigatorTreeNodeUserUpdateListener extends
AbstractNavigatorTreeNodeUpdateListener {
    static final long serialVersionUID = 0L;

    protected NavigatorTreeNodeUserUpdateListener(NavigatorTreeNode
navigatorTreeNode,
        NavigatorTree tree) {
        super(navigatorTreeNode, tree);
    }

    public void actionPerformed(ActionEvent event) {
        if (event instanceof UpdateEntityEvent) {
            UpdateEntityEvent uee = (UpdateEntityEvent) event;
            if ((uee.getObject() != null) && (uee.getObject() instanceof
User)) {
                User user = (User) uee.getObject();
                NavigatorTreeNode thisNode = getNavigatorTreeNode();
                UserSet users = (UserSet) thisNode.getTargetObject();
                if (!users.contains(user)) {
                    addUserNode(thisNode, user);
                } else {
                    removeUserNode(thisNode, user);
                    addUserNode(thisNode, user);
                }
                users.add(user);
            }
        }
    }
}

```

```

}

private void addUserNode(MutableTreeNode thisNode, User user) {
    // add a new node for this user
    int indexToInsert = 0;
    Enumeration<MutableTreeNode> enm = thisNode.children();
    while (enm.hasMoreElements()) {
        MutableTreeNode node = enm.nextElement();
        if (node instanceof NavigatorTreeNode) {
            Object targetObject = ((NavigatorTreeNode)
node).getTargetObject();
            if ((targetObject != null) && (targetObject instanceof User)) {
                User nodeUser = (User) targetObject;
                if (User.UserComparator.compare(user, nodeUser) < 1) {
                    break;
                }
                indexToInsert++;
            }
        }
    }
    // see http://echo.nextapp.com/site/node/1625
    getTree().getModel().insertNodeInto(
        getTree().getNavigatorTreeNodeFactory(user).createTreeNode(
            getNavigatorTreeNode().getEventDispatcher(), getTree(), user),
        thisNode, indexToInsert);
}

private void removeUserNode(MutableTreeNode thisNode, User user) {
    // remove the node with the user.
    Enumeration<MutableTreeNode> enm = thisNode.children();
    while (enm.hasMoreElements()) {
        MutableTreeNode node = enm.nextElement();
        if (node instanceof NavigatorTreeNode) {
            NavigatorTreeNode ntn = (NavigatorTreeNode) node;
            Object targetObject = ntn.getTargetObject();
            if ((targetObject != null) && (targetObject instanceof User)) {
                User nodeUser = (User) targetObject;
                if (User.UserComparator.compare(user, nodeUser) == 0) {
                    getTree().getModel().removeNodeFromParent(node);
                    break;
                }
            }
        }
    }
}

```

```

}

```

usercommandfactory.java

```

/*
 * $Id: UserCommandFactory.java,v 1.3 2008/12/13 00:40:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.user.command;

import edu.harvard.fas.rregan.command.CommandFactory;

/**
 * @author ron
 */
public interface UserCommandFactory extends CommandFactory {

    /**
     * @return a new LoginCommand for authenticating a user of the
     * system.
     */
    public LoginCommand newLoginCommand();

    /**
     * @return a new EditUserCommand for creating or editing a user of
     * the
     *      system.
     */
    public EditUserCommand newEditUserCommand();
}

```

usercommandfactoryimpl.java

```

/*
 * $Id: UserCommandFactoryImpl.java,v 1.6 2008/12/13 00:41:57 rregan
 * Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.command;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;

```

```

import org.springframework.stereotype.Controller;

import edu.harvard.fas.rregan.command.AbstractCommandFactory;
import edu.harvard.fas.rregan.command.CommandFactoryStrategy;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import edu.harvard.fas.rregan.requel.user.command.LoginCommand;
import edu.harvard.fas.rregan.requel.user.command.UserCommandFactory;

/**
 * @author ron
 */
@Controller("userCommandFactory")
@Scope("singleton")
public class UserCommandFactoryImpl extends AbstractCommandFactory
implements UserCommandFactory {

/**
 * @param creationStrategy -
 *          the strategy to use for creating new Command instances
 */
@.Autowired
public UserCommandFactoryImpl(CommandFactoryStrategy
creationStrategy) {
    super(creationStrategy);
}

@Override
public LoginCommand newLoginCommand() {
    return (LoginCommand)
getCreationStrategy().newInstance(LoginCommandImpl.class);
}

@Override
public EditUserCommand newEditUserCommand() {
    return (EditUserCommand)
getCreationStrategy().newInstance(EditUserCommandImpl.class);
}
}

```

usereditorpanel.java

```

/*
 * $Id: UserEditorPanel.java,v 1.13 2009/03/29 02:08:34 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

```

```

package edu.harvard.fas.rregan.requel.ui.user;

import java.text.MessageFormat;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import nextapp.echo2.app.Component;
import nextapp.echo2.app.PasswordField;
import nextapp.echo2.app.TextField;

import org.apache.log4j.Logger;
import org.hibernate.validator.InvalidStateException;
import org.hibernate.validator.InvalidValue;

import echopointng.ComboBox;
import echopointng.text.StringDocumentEx;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.SystemAdminUserRole;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRole;
import edu.harvard.fas.rregan.requel.user UserRolePermission;
import edu.harvard.fas.rregan.requel.user.command.EditUserCommand;
import edu.harvard.fas.rregan.requel.user.command.UserCommandFactory;
import edu.harvard.fas.rregan.uiframework.login.InitAppEvent;
import edu.harvard.fas.rregan.uiframework.navigation.event.UpdateEntityEvent;
import edu.harvard.fas.rregan.uiframework.panel.editor.AbstractEditorPanel;
import edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSet;
import edu.harvard.fas.rregan.uiframework.panel.editor.CheckBoxTreeSetModel;
import edu.harvard.fas.rregan.uiframework.panel.editor.CombinedTextListModel;

/**
 * @author ron
 */
public class UserEditorPanel extends AbstractEditorPanel {

```

```

private static final Logger log =
Logger.getLogger(UserEditorPanel.class);
static final long serialVersionUID = 0L;

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the username field. If the property is undefined "Username" is
used.
 */
public static final String PROP_LABEL_USERNAME = "Username.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the password field. If the property is undefined "Password" is
used.
 */
public static final String PROP_LABEL_PASSWORD = "Password.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the second password field. If the property is undefined "Retype
* Password" is used.
 */
public static final String PROP_LABEL_RE_PASSWORD =
"RePassword.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the name field. If the property is undefined "Name" is used.
 */
public static final String PROP_LABEL_NAME = "Name.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the organization field. If the property is undefined
"Organization" is
 * used.
 */
public static final String PROP_LABEL_ORGANIZATION =
"Organization.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the email address field. If the property is undefined "Email
Address"
 * is used.
 */
public static final String PROP_LABEL_EMAIL = "EmailAddress.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the phone number field. If the property is undefined "Phone
Number" is
 * used.
 */
public static final String PROP_LABEL_PHONE = "PhoneNumber.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the user roles checkbox fields. If the property is undefined
"User
* Roles" is used.
 */
public static final String PROP_LABEL_ROLES = "UserRoles.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the project user permissions checkbox fields. If the property
is
 * undefined "Project User Permissions" is used.
 */
public static final String PROP_LABEL_PROJECT_USER_PERMS =
"ProjectUserPermissions.Label";

/**
 * The name to use in the UserEditorPanel.properties file to set the
label
 * of the checkbox to control whether the user(s) that use this
account can
 * change the values field.
 */
public static final String PROP_LABEL_EDITABLE = "CanBeEdited.Label";

private final UserCommandFactory userCommandFactory;
private final UserRepository userRepository;

```

```

private final CommandHandler commandHandler;

/**
 * @param commandHandler
 * @param userCommandFactory
 * @param userRepository
 */
public UserEditorPanel(CommandHandler commandHandler,
UserCommandFactory userCommandFactory,
 UserRepository userRepository) {
    this(UserEditorPanel.class.getName(), commandHandler,
userCommandFactory, userRepository);
}

/**
 * @param resourceBundleName
 * @param commandHandler
 * @param userCommandFactory
 * @param userRepository
 */
public UserEditorPanel(String resourceBundleName, CommandHandler
commandHandler,
 UserCommandFactory userCommandFactory, UserRepository
userRepository) {
    super(resourceBundleName, User.class);
    this.userCommandFactory = userCommandFactory;
    this.userRepository = userRepository;
    this.commandHandler = commandHandler;
}

@Override
protected boolean isShowDelete() {
    return false;
}

/**
 * If the editor is editing an existing user the title specified in
the
 * properties file as PROP_EXISTING_OBJECT_PANEL_TITLE If that
property is
 * not set it then tries the standard PROP_PANEL_TITLE and if that
does not
 * exist it defaults to:<br>
 * "Edit User: {0}"<br>
 * Valid variables are:<br>
 * {0} - username<br>
 * {1} - name <br>
 */

```

```

* For new users it first tries PROP_NEW_OBJECT_PANEL_TITLE, then
* PROP_PANEL_TITLE and finally defaults to:<br>
* "New User"<br>
*
* @see AbstractEditorPanel.PROP_EXISTING_OBJECT_PANEL_TITLE
* @see AbstractEditorPanel.PROP_NEW_OBJECT_PANEL_TITLE
* @see Panel.PROP_PANEL_TITLE
* @see
edu.harvard.fas.rregan.uiframework.panel.AbstractPanel#getTitle()
*/
@Override
public String getTitle() {
    if (getUser() != null) {
        String msgPattern = getResourceBundleHelper(getLocale()).getString(
            PROP_EXISTING_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "Edit User: {0}"));
        return MessageFormat.format(msgPattern, getUser().getUsername(),
getUser().getName());
    } else {
        String msg = getResourceBundleHelper(getLocale()).getString(
            PROP_NEW_OBJECT_PANEL_TITLE,
            getResourceBundleHelper(getLocale()).getString(PROP_PANEL_TITLE,
                "New User"));
        return msg;
    }
}

@Override
public void setup() {
    super.setup();
    User editingUser = (User) getApp().getUser();
    User user;
    if (editingUser.hasRole(SystemAdminUserRole.class)) {
        user = getUser();
    } else {
        // a non-admin user can only edit their own account info.
        user = editingUser;
    }
    if (user != null) {
        Component username = addInput("username", PROP_LABEL_USERNAME,
"Username",
            new TextField(), new StringDocumentEx(user.getUsername()));
        addInput("password", PROP_LABEL_PASSWORD, "Password", new
PasswordField(),
            new StringDocumentEx());
    }
}

```

```

    addInput("repassword", PROP_LABEL_RE_PASSWORD, "Retype Password",
new PasswordField(),
    new StringDocumentEx());
    addInput("name", PROP_LABEL_NAME, "Name", new TextField(), new
StringDocumentEx(user
    .getName()));
    addInput("organizationName", PROP_LABEL_ORGANIZATION,
"Organization", new ComboBox(),
    new
CombinedTextListModel(getUserRepository().getOrganizationNames(), user
    .getOrganization().getName());
    addInput("emailAddress", PROP_LABEL_EMAIL, "Email Address", new
TextField(),
    new StringDocumentEx(user.getEmailAddress()));
    addInput("phoneNumber", PROP_LABEL_PHONE, "Phone Number", new
TextField(),
    new StringDocumentEx(user.getPhoneNumber()));
if (editingUser.hasRole(SystemAdminUserRole.class)) {
    username.setEnabled(true);
    addInput("userRoles", PROP_LABEL_ROLES, "User Roles", new
CheckBoxTreeSet(),
        createUserRoleSelectionTreeModel(getUserRepository().findUserRol
eTypes(),
            user));
    // addInput("editable", PROP_LABEL_EDITABLE, "Editable User?",
    // new CheckBox(),
    // new ToggleButtonModelEx(user.isEditable()));
} else {
    username.setEnabled(false);
}
} else {
    addInput("username", PROP_LABEL_USERNAME, "Username", new
TextField(),
    new StringDocumentEx());
    addInput("password", PROP_LABEL_PASSWORD, "Password", new
PasswordField(),
    new StringDocumentEx());
    addInput("repassword", PROP_LABEL_RE_PASSWORD, "Retype Password",
new PasswordField(),
    new StringDocumentEx());
    addInput("name", PROP_LABEL_NAME, "Name", new TextField(), new
StringDocumentEx());
    addInput("organizationName", PROP_LABEL_ORGANIZATION,
"Organization", new ComboBox(),
    new
CombinedTextListModel(getUserRepository().getOrganizationNames(),
 ""));
}

    addInput("emailAddress", PROP_LABEL_EMAIL, "Email Address", new
TextField(),
    new StringDocumentEx());
    addInput("phoneNumber", PROP_LABEL_PHONE, "Phone Number", new
TextField(),
    new StringDocumentEx());
    addInput("userRoles", PROP_LABEL_ROLES, "User Roles", new
CheckBoxTreeSet(),
        createUserRoleSelectionTreeModel(getUserRepository().findUserRole
Types(), null));
    // addInput("editable", PROP_LABEL_EDITABLE, "Editable User?", new
    // CheckBox(),
    // new ToggleButtonModelEx(true));
}
}

@Override
public void cancel() {
    super.cancel();
}

@Override
public void save() {
    try {
        super.save();
        User editedBy = (User) getApp().getUser();
        EditUserCommand command =
getUserCommandFactory().newEditUserCommand();
        command.setUser(getUser());
        if (editedBy.hasRole(SystemAdminUserRole.class)) {
            command.setUsername(getInputValue("username", String.class));
            Set<String> userRoles = getInputBorderValue("userRoles", Set.class);
            command.setUserRoleNames(getUserRoleNames(userRoles,
getUserRepository()
    .findUserRoleTypes()));
            command.setUserRolePermissionNames(getUserRolePermissionNames(user
Roles,
                getUserRepository().findUserRoleTypes()));
            command.setEditable(Boolean.TRUE);
        }
        command.setPassword(getInputBorderValue("password", String.class));
        command.setRepassword(getInputBorderValue("repassword", String.class));
        command.setName(getInputBorderValue("name", String.class));
        command.setEmail(getInputBorderValue("emailAddress",
String.class));
        command.setPhoneNumber(getInputBorderValue("phoneNumber", String.class));
    }
}

```

```

        command.setOrganizationName(getInputValue("organizationName",
String.class));
        command.setEditedBy(editedBy);
        command = getCommandHandler().execute(command);
        User user = command.getUser();
        setValid(true);
        if (getApp().getUser().equals(user)) {
            getEventDispatcher().dispatchEvent(new InitAppEvent(this, user));
        }
        getEventDispatcher().dispatchEvent(new UpdateEntityEvent(this,
user));
    } catch (EntityException e) {
        if ((e.getCause() != null) && (e.getCause() instanceof
        InvalidStateException)) {
            InvalidStateException ise = (InvalidStateException) e.getCause();
            for (InvalidValue invalidValue : ise.getInvalidValues()) {
                String propertyName = invalidValue.getPropertyName();
                if ("hashedPassword".equals(propertyName)) {
                    propertyName = "password";
                }
                if
(Organization.class.isAssignableFrom(invalidValue.getBeanClass())
                && propertyName.equals("name")) {
                    propertyName = "organizationName";
                }
                setValidationMessage(propertyName, invalidValue.getMessage());
            }
        } else if ((e.getEntityPropertyNames() != null)
        && (e.getEntityPropertyNames().length > 0)) {
            for (String propertyName : e.getEntityPropertyNames()) {
                setValidationMessage(propertyName, e.getMessage());
            }
        } else {
            setGeneralMessage(e.toString());
        }
    } catch (Exception e) {
        log.error("could not save the user: " + e, e);
        setGeneralMessage("Could not save: " + e);
    }
}

private Set<String> getUserRoleNames(Set<String> selectedPaths,
    Set<Class<? extends UserRole>> userRoleTypes) {
    log.debug("selectedPaths = " + selectedPaths);
    Set<String> userRoleNames = new HashSet<String>();
    for (Class<? extends UserRole> userRoleType : userRoleTypes) {
        String userRoleName = AbstractUserRole.getRoleName(userRoleType);

```

```

        log.debug("userRoleName = " + userRoleName);
        if (selectedPaths.contains(userRoleName)) {
            userRoleNames.add(userRoleName);
            log.debug("adding " + userRoleName);
        }
    }
    return userRoleNames;
}

private Map<String, Set<String>>
getUserRolePermissionNames(Set<String> selectedPaths,
    Set<Class<? extends UserRole>> userRoleTypes) {
    log.debug("selectedPaths = " + selectedPaths);
    Map<String, Set<String>> userRolePermissionNames = new
HashMap<String, Set<String>>();
    for (Class<? extends UserRole> userRoleType : userRoleTypes) {
        String userRoleName = AbstractUserRole.getRoleName(userRoleType);
        log.debug("userRoleName = " + userRoleName);
        userRolePermissionNames.put(userRoleName, new HashSet<String>());
        if (selectedPaths.contains(userRoleName)) {
            for (UserRolePermission userRolePermission : getUserRepository()
                .findUserRolePermissions(userRoleType)) {
                log.debug("userRolePermission = " + userRolePermission);
                if (selectedPaths.contains(userRoleName + "/" +
userRolePermission.getName())) {
                    log.debug("adding " + userRolePermission);
                    userRolePermissionNames.get(userRoleName).add(userRolePermission
                .getName());
                }
            }
        }
    }
    return userRolePermissionNames;
}

private CheckBoxTreeSetModel createUserRoleSelectionTreeModel(
    Set<Class<? extends UserRole>> userRoleTypes, User user) {
    Set<String> optionPaths = new TreeSet<String>(new
Comparator<String>() {
    @Override
    public int compare(String o1, String o2) {
        return o2.compareTo(o1);
    }
});
    Set<String> initialSelection = new HashSet<String>();
    for (Class<? extends UserRole> userRoleType : userRoleTypes) {

```

```

String userRoleTypePath =
AbstractUserRole.getRoleName(userRoleType);
optionPaths.add(userRoleTypePath);
if ((user != null) && user.hasRole(userRoleType)) {
    initialSelection.add(userRoleTypePath);
}
for (UserRolePermission permission :
getRepository().findUserRolePermissions(
    userRoleType)) {
    String userRolePermissionPath = userRoleTypePath + "/" +
permission.getName();
    optionPaths.add(userRolePermissionPath);
    if ((user != null) && user.hasRole(userRoleType)) {
        UserRole role = user.getRoleForType(userRoleType);
        if (role.hasUserRolePermission(permission)) {
            initialSelection.add(userRolePermissionPath);
        }
    }
}
return new CheckBoxTreeSetModel(optionPaths, initialSelection);
}

private CommandHandler getCommandHandler() {
    return commandHandler;
}

private UserCommandFactory getUserCommandFactory() {
    return userCommandFactory;
}

private UserRepository getUserRepository() {
    return userRepository;
}

private User getUser() {
    return (User) getTargetObject();
}
}

```

userentityexception.java

```

/*
 * $Id: UserEntityException.java,v 1.2 2008/12/13 00:41:36 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

*/
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRole;

/**
 * @author ron
 */
public class UserEntityException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_EXCEPTION_GRANTING_ROLE = "Exception
granting role '%s' to user '%s': %s";

    protected static String MSG_EXCEPTION_REVOKING_ROLE = "Exception
revoking role '%s' from user '%s': %s";

    /**
     * @param userRoleType -
     *          the type of role being granted
     * @param user -
     *          the user the role was being granted to
     * @param cause -
     *          the exception that occurred during the grant
     * @return
     */
    public static UserEntityException exceptionGrantingRole(Class<?
extends UserRole> userRoleType,
    User user, Exception cause) {
        return new UserEntityException(cause, User.class, user, "userRoles",
userRoleType,
            EntityExceptionActionType.Creating, MSG_EXCEPTION_GRANTING_ROLE,
userRoleType
                .getSimpleName(), user.getUsername(), cause);
    }

    /**
     * @param userRoleType -
     *          the type of role being revoked
     * @param user -
     *          the user the role was being revoked from
     * @param cause -
     *          the exception that occurred during the revoke.
     */

```

```

 * @return
 */
public static UserEntityException exceptionRevokingRole(Class<?
extends UserRole> userRoleType,
    User user, Exception cause) {
    return new UserEntityException(cause, User.class, user, "userRoles",
userRoleType,
        EntityExceptionActionType.Creating, MSG_EXCEPTION_REVOKING_ROLE,
userRoleType
        .getSimpleName(), user.getUsername(), cause);
}

/**
 * @param format
 * @param args
 */
protected UserEntityException(Class<?> entityType, Object entity,
String entityPropertyName,
    Object entityValue, EntityExceptionActionType actionType, String
format,
    Object... messageArgs) {
    super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected UserEntityException(Throwable cause, Class<?> entityType,
Object entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
    super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
        messageArgs);
}
}

```

userimpl.java

```

/*
 * $Id: UserImpl.java,v 1.38 2009/03/22 11:08:23 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.

```

```

 */
package edu.harvard.fas.rregan.requel.user.impl;

import java.io.Serializable;
import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.Version;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import org.hibernate.validator.Email;
import org.hibernate.validator.NotEmpty;
import org.hibernate.validator.Pattern;
import org.hibernate.validator.Size;

import com.sun.xml.bind.v2.runtime.unmarshaller.UnmarshallingContext;

import edu.harvard.fas.rregan.HashUtils;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.Organization;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user UserRole;

```

```

import
edu.harvard.fas.rregan.requel.user.exception.NoSuchRoleForUserException;
import
edu.harvard.fas.rregan.requel.user.exception.UserEntityException;
import
edu.harvard.fas.rregan.requel.utils.jaxb.JAXBOrganizedEntityPatcher;
import edu.harvard.fas.rregan.requel.utils.jaxb.UnmarshallerListener;

/**
 * @author ron
 */
@Entity
@Table(name = "users")
@XmlRootElement(name = "user", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "user", propOrder = { "username", "hashedPassword",
"name", "emailAddress",
"phoneNumber", "organization", "userRoles", "editable" }, namespace
= "http://www.people.fas.harvard.edu/~rregan/requel")
public class UserImpl implements User, Serializable {
    static final long serialVersionUID = 0L;

    private Long id;
    private String name;
    private String username;
    private String hashedPassword;
    private String emailAddress;
    private String phoneNumber;
    private Organization organization;
    private boolean editable = true;
    private Set<UserRole> userRoles;
    // start at 1 so hibernate recognizes the new
    // instance as the initial value and not stale.
    private int version = 1;

    /**
     * @param username
     * @param password
     * @param emailAddress
     * @param organization
     */
    public UserImpl(String username, String password, String
emailAddress, Organization organization) {
        setUsername(username);
        setUserRoles(new
TreeSet<UserRole>(UserRole UserRoleComparator.INSTANCE));
    }
}

```

```

    resetPassword(password);
    setEmailAddress(emailAddress);
    setOrganization(organization);
}

/**
 * @param username
 * @param password
 * @param repassword
 * @param emailAddress
 * @param organization
 */
public UserImpl(String username, String password, String repassword,
String emailAddress,
Organization organization) {
    setUsername(username);
    setUserRoles(new
TreeSet<UserRole>(UserRole UserRoleComparator.INSTANCE));
    resetPassword(password, repassword);
    setEmailAddress(emailAddress);
    setOrganization(organization);
}

/**
 * @param username
 * @param password
 * @param repassword
 * @param name
 * @param emailAddress
 * @param phoneNumber
 * @param organization
 * @param editable
 */
public UserImpl(String username, String password, String repassword,
String name,
String emailAddress, String phoneNumber, Organization organization,
Boolean editable) {
    this(username, password, repassword, emailAddress, organization);
    setName(name);
    setPhoneNumber(phoneNumber);
    setEditable(editable);
}

protected UserImpl() {
    // for hibernate
}

```

```

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(Adapter.class)
protected Long getId() {
    return id;
}

protected void setId(Long id) {
    this.id = id;
}

@Version
protected int getVersion() {
    return version;
}

protected void setVersion(int version) {
    this.version = version;
}

@XmlElement(name = "name", defaultValue = "", required = true,
namespace = "http://www.people.fas.harvard.edu/~rregan/requel")
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

@Column(unique = true, nullable = false)
@NotEmpty(message = "username is required.")
@XmlElement(name = "username", required = true, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

@Transient
public boolean isPassword(String password) {
    return
getHashedPassword().equals(HashUtils.getMD5HashDigestString(password))
;
}

/**
 * Reset the user's password if the password and repassword match.
 *
 * @param password
 * @param repassword
 */
public void resetPassword(String password, String repassword) {
    if ((password != null) && (password.trim().length() > 0)) {
        if (password.equals(repassword)) {
            setPassword(password);
        } else {
            // This is needed because hibernate checks nullability before
            // doing validation and throws a PropertyValueException for
            // the password without validating the rest of the properties.
            setHashedPassword("");
        }
    } else {
        // This is needed because hibernate checks nullability before
        // doing validation and throws a PropertyValueException for
        // the password without validating the rest of the properties.
        if (getHashedPassword() == null) {
            setHashedPassword("");
        }
    }
}

public void resetPassword(String password) {
    if ((password != null) && (password.trim().length() > 0)) {
        setHashedPassword(HashUtils.getMD5HashDigestString(password));
    } else {
        // This is needed because hibernate checks nullability before
        // doing validation and throws a PropertyValueException for
        // the password without validating the rest of the properties.
        if (getHashedPassword() == null) {
            setHashedPassword("");
        }
    }
}

@Column(nullable = false)
@NotEmpty(message = "password is required and both fields must
match.")

```

```

@XmlElement(name = "password", required = true, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
protected String getHashedPassword() {
    return hashedPassword;
}

protected void setHashedPassword(String hashedPassword) {
    this.hashedPassword = hashedPassword;
}

@Column(nullable = false)
@Email
@NotEmpty(message = "email address is required.")
@XmlElement(name = "emailAddress", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getEmailAddress() {
    return emailAddress;
}

public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
}

@Pattern(regex = "^(?:\\([2-9]\\d{2})\\)|[2-9]\\d{2}(?:(?:[- ]?)\\d{4})?", message = "must be a valid
10 digit phone number or empty.")
@XmlElement(name = "phoneNumber", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

@XmlElementRef(type = OrganizationImpl.class, namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@ManyToOne(targetEntity = OrganizationImpl.class, cascade =
{ CascadeType.MERGE,
    CascadeType.PERSIST, CascadeType.REFRESH }, fetch =
FetchType.EAGER, optional = false)
public Organization getOrganization() {
    return organization;
}

public void setOrganization(Organization organization) {

```

```

    this.organization = organization;
}

@XmlElement(name = "editable", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public boolean isEditable() {
    return editable;
}

public void setEditable(boolean editable) {
    this.editable = editable;
}

@XmlElementWrapper(name = "userRoles", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlElementRef(type = AbstractUserRole.class)
@OneToMany(targetEntity = AbstractUserRole.class, cascade =
{ CascadeType.MERGE,
    CascadeType.PERSIST, CascadeType.REFRESH }, fetch =
FetchType.EAGER)
@JoinTable(name = "users_user_roles", joinColumns =
{ @JoinColumn(name = "user_id") }, inverseJoinColumns =
{ @JoinColumn(name = "role_id") })
@Size(min = 1, message = "one or more roles must be selected.")
public Set<UserRole> getUserRoles() {
    return userRoles;
}

protected void setUserRoles(Set<UserRole> userRoles) {
    this.userRoles = userRoles;
}

@Transient
public <T extends UserRole> T getRoleForType(Class<T> userRoleType)
    throws NoSuchRoleForUserException {
    for (UserRole role : getUserRoles()) {
        if (userRoleType.isAssignableFrom(role.getClass())) {
            return userRoleType.cast(role);
        }
    }
    throw NoSuchRoleForUserException.forUserRoleTypeName(this,
userRoleType);
}

@Transient
public boolean hasRole(Class<? extends UserRole> userRoleType) {
    try {

```

```

getRoleForType(userRoleType);
return true;
} catch (NoSuchRoleForUserException e) {
return false;
}
}

public void grantRole(Class<? extends UserRole> userRoleType) {
if (!hasRole(userRoleType)) {
UserRole role = null;
Constructor<? extends UserRole> constructor = null;
try {
constructor = userRoleType.getConstructor(User.class);
role = constructor.newInstance(this);
} catch (NoSuchMethodException e) {
try {
constructor = userRoleType.getConstructor();
role = constructor.newInstance();
} catch (Exception e2) {
UserEntityException.exceptionGrantingRole(userRoleType, this,
e2);
}
} catch (Exception e) {
UserEntityException.exceptionGrantingRole(userRoleType, this, e);
}
getUserRoles().add(role);
}
}

public void revokeRole(Class<? extends UserRole> userRoleType) {
if (hasRole(userRoleType)) {
UserRole role = getRoleForType(userRoleType);
getUserRoles().remove(role);
try {
Method setUser = role.getClass().getDeclaredMethod("setUser",
User.class);
setUser.setAccessible(true);
setUser.invoke(role, new Object[] { null });
} catch (NoSuchMethodException e) {
// expected if the role is shared by multiple users
} catch (Exception e) {
UserEntityException.exceptionRevokingRole(userRoleType, this, e);
}
}
}

@Override

```

```

public boolean equalsById(User other) {
if ((getId() != null) && (((UserImpl) other).getId() != null)) {
return getId().equals(((UserImpl) other).getId());
}
return false;
}

@Override
public boolean equals(Object obj) {
if (this == obj) {
return true;
}
if (obj == null) {
return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
return false;
}
final UserImpl other = (UserImpl) obj;
if ((getId() != null) && getId().equals(other.getId())) {
return true;
}
if (getUsername() == null) {
if (other.getUsername() != null) {
return false;
}
} else if (!getUsername().equals(other.getUsername())) {
return false;
}
return true;
}

@Override
public int compareTo(User o) {
return getUsername().compareTo(o.getUsername());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
if (tmpHashCode == null) {
if (getId() != null) {
tmpHashCode = new Integer(getId().hashCode());
}
final int prime = 31;
int result = 1;

```

```

        result = prime * result + ((getUsername() == null) ? 0 :
getUsername().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public String toString() {
    return User.class.getName() + "[" + getId() + "]: " + getUsername();
}

/**
 * This is for JAXB to patchup existing persistent objects for the
objects
 * that are attached directly to this object.
 *
 * @param userRepository
 * @see UnmarshallerListener
 */
public void afterUnmarshal(UserRepository userRepository) {
    UnmarshallingContext.getInstance().addPatcher(
        new JAXBOrganizedEntityPatcher(userRepository, this));
}

/**
 * This class is used by JAXB to convert the id of an entity into an
xml id
 * string that will be distinct from other entity xml id strings by
the use
 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "USR_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
    }
}

```

```

        }
        return "";
    }
}
}

```

usernameinuseexception.java

```

/*
 * $Id: UsernameInUseException.java,v 1.4 2008/12/13 00:41:36 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.user.User;

/**
 * @author ron
 */
public class UsernameInUseException extends UserEntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_NAME = "A user already exists with
username '%s'";

    /**
     * @param username -
     *          the username already in use
     * @return
     */
    public static UsernameInUseException forName(String username) {
        return new UsernameInUseException(User.class, null, "username",
username,
            EntityExceptionActionType.Creating, MSG_FOR_NAME, username);
    }

    /**
     * @param format
     * @param args
     */
    protected UsernameInUseException(Class<?> entityType, Object entity,
String entityPropertyName,
        Object entityValue, EntityExceptionActionType actionType, String
format,

```

```

    Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected UsernameInUseException(Throwable cause, Class<?>
entityType, Object entity,
String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
messageArgs);
}
}

```

usernavigatorTreeNodeFactory.java

```

/*
 * $Id: UserNavigatorTreeNodeFactory.java,v 1.1 2008/09/12 22:44:15
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.ui.user;

import nextapp.echo2.app.Label;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import echopointng.tree.MutableTreeNode;
import edu.harvard.fas.rregan.requel.user.User;
import
edu.harvard.fas.rregan.uiframework.navigation.WorkflowDisposition;
import
edu.harvard.fas.rregan.uiframework.navigation.event.EventDispatcher;
import
edu.harvard.fas.rregan.uiframework.navigation.event.NavigationEvent;
import
edu.harvard.fas.rregan.uiframework.navigation.event.OpenPanelEvent;

```

```

import
edu.harvard.fas.rregan.uiframework.navigation.tree.AbstractNavigatorTr
eeNodeFactory;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTree;
import
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNode;
import edu.harvard.fas.rregan.uiframework.panel.PanelActionType;

/**
 * @author ron
 */
@Component("userNavigatorTreeNodeFactory")
@Scope("singleton")
public class UserNavigatorTreeNodeFactory extends
AbstractNavigatorTreeNodeFactory {

/**
 * Create a new NavigatorTreeNodeFactory appropriate for
SystemAdminUserRole
 * objects.
 */
public UserNavigatorTreeNodeFactory() {
super(UserNavigatorTreeNodeFactory.class.getName(), User.class);
}

/**
 * @see
edu.harvard.fas.rregan.uiframework.navigation.tree.NavigatorTreeNodeFa
ctory#createTreeNode(edu.harvard.fas.rregan.uiframework.navigation.tre
e.NavigatorTree,
* java.lang.Object)
*/
@Override
public MutableTreeNode createTreeNode(EventDispatcher
eventDispatcher, NavigatorTree tree,
Object object) {
User user = (User) object;
NavigationEvent openUserEditor = new OpenPanelEvent(tree,
PanelActionType.Editor, user,
User.class, null, WorkflowDisposition.NewFlow);
// TODO: create a NavigatorTreeNodeUpdateListener to listen for
// UpdateEvents for the user.
return new NavigatorTreeNode(eventDispatcher, user, new
Label(user.getUsername()),
openUserEditor);
}
}

```

```
}
```

userpropertyvalueexceptionadapter.java

```
/*
 * $Id: UserPropertyValueExceptionAdapter.java,v 1.2 2009/02/17
11:50:50 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.repository.jpa;
import org.hibernate.PropertyValueException;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.repository.EntityExceptionAdapter;
import edu.harvard.fas.rregan.requell.user.EntityValidationException;

/**
 * @author ron
 */
public class UserPropertyValueExceptionAdapter implements
EntityExceptionAdapter {

/**
 * @see
edu.harvard.fas.rregan.repository.EntityExceptionAdapter#convert(java.
lang.Throwable,
 *      java.lang.Object,
 *      edu.harvard.fas.rregan.repository.EntityExceptionActionType)
*/
@Override
public EntityException convert(Throwable original, Class<?>
entityType, Object entity,
EntityExceptionActionType actionType) {
PropertyValueException pve = (PropertyValueException) original;
String propertyName = pve.getPropertyName();
if ("hashedPassword".equals(propertyName)) {
propertyName = "password";
}
if (original.getMessage().startsWith(
"not-null property references a null or transient value")) {
return EntityValidationException.emptyRequiredProperty(entityType,
entity,

```

```
        propertyName, actionType);
} else {
return EntityException.forUnknownProblem(original, entityType,
entity, propertyName,
null, actionType);
}
}
}
```

userrepository.java

```
/*
 * $Id: UserRepository.java,v 1.16 2008/12/13 00:41:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requell.user;

import java.util.Set;

import edu.harvard.fas.rregan.repository.Repository;
import
edu.harvard.fas.rregan.requell.user.exception.NoSuchOrganizationExcepti
on;
import
edu.harvard.fas.rregan.requell.user.exception.NoSuchUserException;

/**
 * @author ron
 */
public interface UserRepository extends Repository {

/**
 * Get a specific organization by name.
 *
 * @param name -
 *      the name of the organization
 * @return the organization with the specified name
 * @throws NoSuchOrganizationException
 */
public Organization findOrganizationByName(String name) throws
NoSuchOrganizationException;

/**
 * @return all the organizations in the system.
 */

```

```

public Set<Organization> findOrganizations();

/**
 * @return all the names of the organizations in the system.
 */
public Set<String> getOrganizationNames();

/**
 * Get a specific user by username from the repository.
 *
 * @param username -
 *          the username of the user to retrieve from the
repository.
 * @return The user for the username
 * @throws NoSuchUserException -
 *          if the supplied username doesn't correspond to a user
in the
 *          repository.
 */
public User findUserByUsername(String username) throws
NoSuchUserException;

/**
 * @return all the users of the system in the repository.
 */
public UserSet findUsers();

/**
 * @param roleType
 * @return a set of users that have the supplied role type.
 */
public UserSet findUsersForRole(Class<? extends UserRole> roleType);

/**
 * @return the available types of user roles
 */
public Set<Class<? extends UserRole>> findUserRoleTypes();

/**
 * @param userRoleType
 * @param name
 * @return
 */
public UserRolePermission findUserRolePermission(Class<? extends
UserRole> userRoleType,
String name);

```

```

/**
 * @param userRoleType
 * @return
 */
public Set<UserRolePermission> findUserRolePermissions(Class<?
extends UserRole> userRoleType);
}

```

userrole.java

```

/*
 * $Id: UserRole.java,v 1.8 2008/03/26 10:39:53 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user;

import java.util.Comparator;
import java.util.Set;

/**
 * @author ron
 */
public interface UserRole {

    /**
     * @return the name of the role.
     */
    public String getRoleName();

    /**
     * @return the permissions available for this role
     */
    public Set<UserRolePermission> getAvailableUserRolePermissions();

    /**
     * Grant the specified permission to the user.
     *
     * @param permission
     */
    public void grantUserRolePermission(UserRolePermission permission);

    /**
     * Revoke the specified permission from the user.
     *
     * @param permission
     */
}

```

```

public void revokeUserRolePermission(UserRolePermission permission);

/**
 * @param permission
 * @return true if the user has the specified permission with this
role.
 */
public boolean hasUserRolePermission(UserRolePermission permission);

/**
 * Compare user roles by name.
 *
 * @author ron
 */
public static class UserRoleComparator implements
Comparator<UserRole> {
    public static final UserRoleComparator INSTANCE = new
UserRoleComparator();

    public int compare(UserRole o1, UserRole o2) {
        return o1.getRoleName().compareTo(o2.getRoleName());
    }

    @Override
    public boolean equals(Object o) {
        return (o instanceof UserRoleComparator);
    }
}

```

userroleexistsexception.java

```

/*
 * $Id: UserRoleExistsException.java,v 1.4 2008/12/13 00:41:36 rregan
Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.exception;

import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.repository.EntityExceptionActionType;
import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserRole;

/**
 * @author ron

```

```

/*
public class UserRoleExistsException extends EntityException {
    static final long serialVersionUID = 0;

    protected static String MSG_FOR_NAME = "The user role '%s' already
exists.";
    protected static String MSG_FOR_NAME_AND_USER = "The user role '%s'
for user '%s' already exists.";

    /**
     * @param userRole -
     *          the userRole that already exists.
     * @return
     * @deprecated
     */
    @Deprecated
    public static UserRoleExistsException forRole(UserRole userRole) {
        return new UserRoleExistsException(UserRole.class, userRole, null,
null,
            EntityExceptionActionType.Creating, MSG_FOR_NAME,
userRole.getRoleName());
    }

    /**
     * @param userRole -
     *          the userRole that already exists for the supplied user.
     * @param user -
     *          the user that the role already exists for.
     * @return
     * @deprecated
     */
    @Deprecated
    public static UserRoleExistsException forRoleAndUser(UserRole
userRole, User user) {
        return new UserRoleExistsException(User.class, user, "userRoles",
userRole,
            EntityExceptionActionType.Updating, MSG_FOR_NAME_AND_USER,
userRole.getRoleName(),
            user.getUsername());
    }

    /**
     * @param format
     * @param args
     */
    protected UserRoleExistsException(Class<?> entityType, Object entity,

```

```

    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
super(entityType, entity, entityPropertyName, entityValue,
actionType, format, messageArgs);
}

/**
 * @param cause
 * @param format
 * @param args
 */
protected UserRoleExistsException(Throwable cause, Class<?>
entityType, Object entity,
    String entityPropertyName, Object entityValue,
EntityExceptionActionType actionType,
    String format, Object... messageArgs) {
super(cause, entityType, entity, entityPropertyName, entityValue,
actionType, format,
    messageArgs);
}
}

```

userrolepermission.java

```

/*
 * $Id: UserRolePermission.java,v 1.8 2009/01/10 11:08:59 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.user;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.adapters.XmlAdapter;

```

```

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

/**
 * @author ron
 */
@Entity
@Table(name = "user_role_permissions", uniqueConstraints =
{ @UniqueConstraint(columnNames =
    "name", "role_type" ) })
@XmlRootElement(name = "userPermission", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
@XmlType(name = "userPermission", namespace =
"http://www.people.fas.harvard.edu/~rregan/requel")
public class UserRolePermission implements
Comparable<UserRolePermission>, Serializable {
static final long serialVersionUID = 0L;

private Long id;
private String userRoleType;
private String name;

/**
 * @param userRoleType
 * @param permissionName
 */
public UserRolePermission(Class<? extends UserRole> userRoleType,
String permissionName) {
setUserRoleType(userRoleType.getName());
setName(permissionName);
}

protected UserRolePermission() {
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
protected Long getId() {
return id;
}

protected void setId(Long id) {
this.id = id;
}

/***

```

```

 * @return The name of the permission for display
 */
@Column(name = "name", nullable = false, length = 50)
@XmlAttribute(name = "name")
public String getName() {
    return name;
}

protected void setName(String name) {
    this.name = name;
}

@Column(name = "role_type", nullable = false, length = 255)
@XmlAttribute(name = "userRoleType")
protected String getUserRoleType() {
    return userRoleType;
}

protected void setUserRoleType(String userRoleType) {
    this.userRoleType = userRoleType;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        // NOTE: this doesn't use Id so that the static permission
        // constants at the top of this class will match the permissions
        // in the database.
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getName() == null) ? 0 :
        getName().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
}

```

```

if (!(obj instanceof UserRolePermission)) {
    return false;
}
final UserRolePermission other = (UserRolePermission) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getName() == null) {
    if (other.getName() != null) {
        return false;
    }
} else if (!getName().equals(other.getName())) {
    return false;
}
return true;
}

@Override
public String toString() {
    return this.getClass().getName() + "[" + getId() + "]:" + +
    getName();
}

@Override
public int compareTo(UserRolePermission o) {
    if ((getId() != null) && getId().equals(o.getId())) {
        return 0;
    }
    return getName().compareTo(o.getName());
}

/**
 * This class is used by JAXB to convert the id of an entity into an
xml id
 * string that will be distinct from other entity xml id strings by
the use
 * of a prefix.
 *
 * @author ron
 */
@XmlTransient
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "PRM_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return null; // new Long(id.substring(prefix.length()));
    }
}

```

```

}

@Override
public String marshal(Long id) throws Exception {
    if (id != null) {
        return prefix + id.toString();
    }
    return "";
}
}

```

userrolepermissionsinitializer.java

```

/*
 * $Id: UserRolePermissionsInitializer.java,v 1.7 2009/01/26 10:19:03
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user.impl.repository.init;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.repository.EntityException;
import edu.harvard.fas.rregan.requel.user.AbstractUserRole;
import edu.harvard.fas.rregan.requel.user.UserRepository;
import edu.harvard.fas.rregan.requel.user.UserRole;
import edu.harvard.fas.rregan.requel.user.UserRolePermission;

/**
 * @author ron
 */
@Component("userRolePermissionsInitializer")
@Scope("prototype")
public class UserRolePermissionsInitializer extends
AbstractSystemInitializer {

    private final UserRepository userRepository;

```

```

    @Autowired
    public UserRolePermissionsInitializer(UserRepository userRepository)
    {
        super(10);
        this.userRepository = userRepository;
    }

    @Override
    @Transactional(propagation = Propagation.REQUIRED)
    public void initialize() {
        log.debug("update role permissions...");
        for (Class<? extends UserRole> userRoleType :
userRepository.findUserRoleTypes()) {
            Set<UserRolePermission> fixedPermissions = new
HashSet<UserRolePermission>();
            for (UserRolePermission permission : AbstractUserRole
                .getAvailableUserRolePermissions(userRoleType)) {
                try {
                    permission = userRepository.findUserRolePermission(userRoleType,
permission
                        .getName());
                    log.debug(permission + " is already persistent.");
                } catch (EntityException e) {
                    log.debug("creating: " + permission);
                    permission = userRepository.persist(permission);
                }
                fixedPermissions.add(permission);
            }
            AbstractUserRole.userRoleTypePermissions.put(userRoleType,
fixedPermissions);
        }
    }
}

```

userset.java

```

/*
 * $Id: UserSet.java,v 1.1 2008/03/27 09:26:03 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.requel.user;

import java.util.Set;

```

```

/**
 * A set of users that recognizes a user even if the username changes.
 *
 * @author ron
 */
public interface UserSet extends Set<User> {
}

usersetimpl.java

/*
 * $Id: UserSetImpl.java,v 1.3 2008/08/28 09:53:36 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.requel.user.impl;

import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import edu.harvard.fas.rregan.requel.user.User;
import edu.harvard.fas.rregan.requel.user.UserSet;

/**
 * A set of users that recognizes a user even if the username changes.
TODO:
 * this may not be needed because of considering the id in equals,
compareTo and
 * hashCode methods.
 *
 * @author ron
 */
public class UserSetImpl implements UserSet {

    Set<User> usersByName = new TreeSet<User>(User.UserComparator);
    Map<Long, User> usersById = new HashMap<Long, User>();

    /**
     * create a new empty UserSet
     */
    public UserSetImpl() {
        }
    }

    /**
     * Create a new UserSet with the given collection of users.
     *
     * @param users
     */
    public UserSetImpl(Collection<?> users) {
        for (Object o : users) {
            UserImpl user = (UserImpl) o;
            User originalUser = usersById.get(user.getId());
            if ((originalUser == null) && usersByName.contains(user)) {
                throw new IllegalArgumentException("user " + user
                    + " conflicts with an existing user " + originalUser);
            }
            usersById.put(user.getId(), user);
            usersByName.add(user);
        }
    }

    /**
     * @see java.util.Collection#add(java.lang.Object)
     * @return true if the user was not in the collection or the user is
not
     *         equal to the copy that was in the collection.
     */
    public boolean add(User u) {
        UserImpl user = (UserImpl) u;
        User originalUser = usersById.get(user.getId());
        if ((originalUser == null) && usersByName.contains(user)) {
            throw new IllegalArgumentException("user " + user + " conflicts
with an existing user "
                + originalUser);
        }
        usersById.put(user.getId(), user);
        usersByName.add(user);
        return !user.equals(originalUser);
    }

    /**
     * @see java.util.Collection#addAll(java.util.Collection)
     */
    public boolean addAll(Collection<? extends User> c) {
        boolean rval = false;
        for (User u : c) {
            rval = (rval || add(u));
        }
        }
    }
}

```

```

    return rval;
}

/**
 * @see java.util.Collection#clear()
 */
public void clear() {
    usersByName.clear();
    usersById.clear();
}

/**
 * @see java.util.Collection#contains(java.lang.Object)
 */
public boolean contains(Object o) {
    if (o instanceof User) {
        UserImpl user = (UserImpl) o;
        return usersById.containsKey(user.getId());
    }
    return false;
}

/**
 * @see java.util.Collection#containsAll(java.util.Collection)
 */
public boolean containsAll(Collection<?> c) {
    for (Object o : c) {
        if (!contains(o)) {
            return false;
        }
    }
    return true;
}

/**
 * @see java.util.Collection#isEmpty()
 */
public boolean isEmpty() {
    return usersById.isEmpty();
}

/**
 * @see java.util.Collection#iterator()
 */
public Iterator<User> iterator() {
    return usersByName.iterator();
}

```

```

    /**
     * @see java.util.Collection#remove(java.lang.Object)
     */
    public boolean remove(Object o) {
        if (o instanceof User) {
            UserImpl u = (UserImpl) o;
            User originalUser = usersById.remove(u.getId());
            if (originalUser != null) {
                usersByName.remove(originalUser);
            }
            return (originalUser != null);
        }
        return false;
    }

    /**
     * @see java.util.Collection#removeAll(java.util.Collection)
     */
    public boolean removeAll(Collection<?> c) {
        boolean rval = false;
        for (Object o : c) {
            rval = (rval || remove(o));
        }
        return rval;
    }

    /**
     * @see java.util.Collection#retainAll(java.util.Collection)
     */
    public boolean retainAll(Collection<?> c) {
        boolean rval = usersByName.retainAll(c);
        usersById.clear();
        for (User u : usersByName) {
            usersById.put(((UserImpl) u).getId(), u);
        }
        return rval;
    }

    /**
     * @see java.util.Collection#size()
     */
    public int size() {
        return usersById.size();
    }
}

```

```

 * @see java.util.Collection#toArray()
 */
public Object[] toArray() {
    return usersByName.toArray();
}

/** 
 * @see java.util.Collection#toArray(T[])
 */
public <T> T[] toArray(T[] a) {
    return usersByName.toArray(a);
}

```

verbmatchingrule.java

```

/*
 * $Id: VerbMatchingRule.java,v 1.6 2009/02/11 09:02:54 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.util.ListIterator;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.GrammaticalRelationType;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.impl.NLPTextImpl;

/**
 * @author ron
 */
public class VerbMatchingRule implements SyntaxMatchingRule {
    private static final Logger log =
        Logger.getLogger(VerbMatchingRule.class);

    /**
     * Create a new rule that matches the primary verb verbs.
     */
    public VerbMatchingRule() {

```

```

    /**
     * @see
     * @see edu.harvard.fas.rregan.nlp.impl.srl.SyntaxMatchingRule#match(edu.harvard.fas.rregan.nlp.impl.srl.SyntaxMatchingContext)
     */
    @Override
    public void match(DictionaryRepository dictionaryRepository, NLPText verb,
                      ListIterator<NLPText> textIterator) throws
        SemanticRoleLabelerException {
        NLPText matchedWords = new NLPTextImpl();
        NLPText word = textIterator.next();

        // http://en.wikipedia.org/wiki/Auxiliary_verb
        // optional modal auxiliary verbs: can/could, may/might,
        shall/should,
        // will/would, must, ought.
        if (word.is(PartOfSpeech.MODAL) && word.isDependentOf(GrammaticalRelationType.AUXILIARY, verb)) {
            matchedWords.addRef(word);
            word = textIterator.next();
            // match for "not"
            if ("not".equalsIgnoreCase(word.getText()) && word.is(PartOfSpeech.ADVERB) && word.isDependentOf(GrammaticalRelationType.NEGATION_MODIFIER, verb)) {
                matchedWords.addRef(word);
                word = textIterator.next();
            }
        }

        // I have/had <verb>
        // optional auxiliary verb: have, had
        if (word.is(PartOfSpeech.VERB) && word.isDependentOf(GrammaticalRelationType.AUXILIARY, verb)) {
            matchedWords.addRef(word);
            word = textIterator.next();
            // match for "not"
            if ("not".equalsIgnoreCase(word.getText()) && word.is(PartOfSpeech.ADVERB) && word.isDependentOf(GrammaticalRelationType.NEGATION_MODIFIER, verb)) {
                matchedWords.addRef(word);
                word = textIterator.next();
            }
        }
    }

```

```

// optional auxiliary verb: be
if (word.is(PartOfSpeech.VERB)
    && word.isDependentOf(GrammaticalRelationType.AUXILIARY, verb)) {
    matchedWords.addRef(word);
    word = textIterator.next();
    // match for "not"
}

// finally the verb!
if (word.is(PartOfSpeech.VERB) && verb.equals(word)) {
    matchedWords.addRef(word);
    log.debug("matched: " + matchedWords.getText());
    return;
}
matchedWords.addRef(word);
throw SemanticRoleLabelerException.matchFailed(this, matchedWords);
}

@Override
public String toString() {
    return getClass().getSimpleName();
}

```

verbnetclass.java

```

/*
 * $Id: VerbNetClass.java,v 1.5 2009/02/11 00:55:08 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;

```

```

/**
 * A VerbNet class. VerbNet frames are organized into a set of classes
 * like
 *   * admire-31.2 and withdraw-82-1. The classes relate the semantic
 * roles
 *   * containing selectional restrictions to the frames that use them.
 *
 * @author ron
 */
@Entity
@Table(name = "vnclass")
public class VerbNetClass {

    private Long id;
    private String name;
    private VerbNetClass parent;
    private Set<VerbNetRoleRef> roleRefs = new HashSet<VerbNetRoleRef>();

    protected VerbNetClass() {
    }

    @Id
    @Column(name = "classid", unique = true, nullable = false)
    public Long getId() {
        return this.id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    @Column(name = "class", length = 32)
    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @ManyToOne(targetEntity = VerbNetClass.class, cascade =
    CascadeType.ALL, optional = true)
    @JoinColumn(name = "parentid", insertable = false, updatable = false,
    nullable = true, unique = false)
    public VerbNetClass getParent() {
        return parent;
    }
}

```

```

}

public void setParent(VerbNetClass parent) {
    this.parent = parent;
}

@OneToMany(targetEntity = VerbNetRoleRef.class, cascade =
{ CascadeType.ALL }, fetch = FetchType.LAZY, mappedBy =
"verbNetClass")
protected Set<VerbNetRoleRef> getRoleRefs() {
    return roleRefs;
}

protected void setRoleRefs(Set<VerbNetRoleRef> roleRefs) {
    this.roleRefs = roleRefs;
}

@Transient
public VerbNetRoleRef getRoleRef(String type) {
    for (VerbNetRoleRef ref : getFullRoleRefs()) {
        if
(type.toLowerCase().equals(ref.getVerbNetRole().getType().toLowerCase(
))) {
            return ref;
        }
    }
    return null;
}

@Transient
private Set<VerbNetRoleRef> getFullRoleRefs() {
    Set<VerbNetRoleRef> fullSet = new HashSet<VerbNetRoleRef>();
    fullSet.addAll(getRoleRefs());
    VerbNetClass ancestor = getParent();
    while (ancestor != null) {
        for (VerbNetRoleRef roleRef : ancestor.getRoleRefs()) {
            if (!fullSet.contains(roleRef)) {
                fullSet.add(roleRef);
            }
        }
        ancestor = ancestor.getParent();
    }
    return fullSet;
}

private Integer tmpHashCode = null;

```

```

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public String toString() {
    return getClass().getSimpleName() + ":" + getName();
}

```

verbnetframe.java

```

/*
 * $Id: VerbNetFrame.java,v 1.2 2009/02/09 10:12:30 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

/**
 * A frame from VerbNET that assigns thematic roles to syntax
structures. The
 * assignment is made through psuedo-XML of the form:<br>
 * <NP value="Agent"><SYNRESTRS/></NP><VERB/><NP
value="Theme"><SYNRESTRS/></NP>
 *
 * @author ron
 */
@Entity

```

```


|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> @Table(name = "vnframedef") public class VerbNetFrame implements Comparable&lt;VerbNetFrame&gt;, Serializable {     static final long serialVersionUID = 0;      private Long id;     private String name;     private String xtagRef;     private String description1;     private String description2;     private String syntax;     private String semantics;      protected VerbNetFrame() {     }      @Id     @Column(name = "frameid", unique = true, nullable = false)     public Long getId() {         return this.id;     }      protected void setId(Long id) {         this.id = id;     }      @Column(name = "number", length = 16)     public String getName() {         return this.name;     }      protected void setName(String name) {         this.name = name;     }      @Column(name = "xtag", length = 16)     public String getXtagRef() {         return xtagRef;     }      protected void setXtagRef(String xtagRef) {         this.xtagRef = xtagRef;     }      @Column(name = "description1", length = 64)     public String getDescription1() {         return description1;     } </pre> | <pre>         }          public void setDescription1(String description1) {             this.description1 = description1;         }          @Column(name = "description2", length = 64)         public String getDescription2() {             return description2;         }          protected void setDescription2(String description2) {             this.description2 = description2;         }          @Column(name = "syntax", nullable = false, length = 16277215)         public String getSyntax() {             return syntax;         }          protected void setSyntax(String syntax) {             this.syntax = syntax;         }          @Column(name = "semantics", nullable = false, length = 16277215)         public String getSemantics() {             return semantics;         }          public void setSemantics(String semantics) {             this.semantics = semantics;         }          @Override         public int compareTo(VerbNetFrame o) {             return getId().compareTo(o.getId());         }          private Integer tmpHashCode = null;          @Override         public int hashCode() {             if (tmpHashCode == null) {                 if (getId() != null) {                     tmpHashCode = new Integer(getId().hashCode());                 }                 final int prime = 31; </pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

```

int result = 1;
result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
result = prime * result + ((getSyntax() == null) ? 0 :
getSyntax().hashCode());
result = prime * result
+ ((getDescription1() == null) ? 0 :
getDescription1().hashCode());
result = prime * result
+ ((getDescription2() == null) ? 0 :
getDescription2().hashCode());
tmpHashCode = new Integer(result);
}
return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
if (this == obj) {
return true;
}
if (obj == null) {
return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
return false;
}
final VerbNetFrame other = (VerbNetFrame) obj;
if ((getId() != null) && getId().equals(other.getId())) {
return true;
}
if (getName() == null) {
if (other.getName() != null) {
return false;
}
} else if (!getName().equals(other.getName())) {
return false;
}
if (getSyntax() == null) {
if (other.getSyntax() != null) {
return false;
}
} else if (!getSyntax().equals(other.getSyntax())) {
return false;
}
if (getDescription1() == null) {
if (other.getDescription1() != null) {
return false;
}
} else if (!getDescription1().equals(other.getDescription1())) {
return false;
}
} else if (!getDescription2().equals(other.getDescription2())) {
return false;
}
if (getDescription2() == null) {
if (other.getDescription2() != null) {
return false;
}
} else if (!getDescription2().equals(other.getDescription2())) {
return false;
}
return true;
}

@Override
public String toString() {
// TODO Auto-generated method stub
return getClass().getSimpleName() + "(" + getName() + "): " +
getSyntax();
}
}

verbnetframeref.java

/*
* $Id: VerbNetFrameRef.java,v 1.3 2009/02/09 10:12:30 rregan Exp $
* Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
*/
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

/**

```

```

* A reference between a VerbNet frame and a WordNet sense.
*
* @author ron
*/
@Entity
@Table(name = "vnframeref")
public class VerbNetFrameRef implements Comparable<VerbNetFrameRef>, Serializable {
    static final long serialVersionUID = 0;
    private Long id;
    private Sense sense;
    private VerbNetFrame frame;
    private VerbNetClass vnClass;

    protected VerbNetFrameRef() {
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "framerefid", unique = true, nullable = false)
    public Long getId() {
        return this.id;
    }

    protected void setId(Long id) {
        this.id = id;
    }

    /**
     * @return The VerbNet frame of this reference.
     */
    @ManyToOne(targetEntity = VerbNetFrame.class, cascade =
    CascadeType.ALL, optional = false)
    @JoinColumn(name = "frameid", insertable = false, updatable = false,
    nullable = true, unique = false)
    public VerbNetFrame getFrame() {
        return frame;
    }

    protected void setFrame(VerbNetFrame frame) {
        this.frame = frame;
    }

    @ManyToOne(targetEntity = VerbNetClass.class, cascade =
    CascadeType.ALL, optional = false)
    @JoinColumn(name = "classid", insertable = false, updatable = false,
    nullable = false, unique = false)

```

```

        public VerbNetClass getVnClass() {
            return vnClass;
        }

        protected void setVnClass(VerbNetClass vnClass) {
            this.vnClass = vnClass;
        }

        /**
         * @return The WordNet sense of this frame reference
         */
        @ManyToOne
        @JoinColumns(
            @JoinColumn(name = "synsetid", referencedColumnName = "synsetid",
            insertable = false, updatable = false, nullable = false, unique =
            false),
            @JoinColumn(name = "wordid", referencedColumnName = "wordid",
            insertable = false, updatable = false, nullable = false, unique =
            false) )
        public Sense getSense() {
            return sense;
        }

        protected void setSense(Sense sense) {
            this.sense = sense;
        }

        @Override
        public int compareTo(VerbNetFrameRef o) {
            return getId().compareTo(o.getId());
        }

        private Integer tmpHashCode = null;

        @Override
        public int hashCode() {
            if (tmpHashCode == null) {
                if (getId() != null) {
                    tmpHashCode = new Integer(getId().hashCode());
                }
                final int prime = 31;
                int result = 1;
                result = prime * result + ((getSense() == null) ? 0 :
                getSense().hashCode());
                result = prime * result + ((getFrame() == null) ? 0 :
                getFrame().hashCode());
                tmpHashCode = new Integer(result);
            }
            return tmpHashCode;
        }
    }
}

```

```

}
return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
if (this == obj) {
    return true;
}
if (obj == null) {
    return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
    return false;
}
final VerbNetFrameRef other = (VerbNetFrameRef) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getSense() == null) {
    if (other.getSense() != null) {
        return false;
    }
} else if (!getSense().equals(other.getSense())) {
    return false;
}
if (getFrame() == null) {
    if (other.getFrame() != null) {
        return false;
    }
} else if (!getFrame().equals(other.getFrame())) {
    return false;
}
return true;
}

@Override
public String toString() {
// TODO Auto-generated method stub
return getClass().getSimpleName() + "(" + getSense() + ":" +
getVnClass() + "):" +
    + getFrame();
}
}

```

verbnetframesyntaxparser.java

```

/*
 * $Id: VerbNetFrameSyntaxParser.java,v 1.7 2009/02/11 09:02:55 rregan
 * Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.io.Reader;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.log4j.Logger;
import org.apache.xpath.XPathAPI;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.traversal.NodeIterator;
import org.xml.sax.InputSource;

import edu.harvard.fas.rregan.nlp.SemanticRole;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetFrame;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetFrameRef;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;

/**
 * Parser for VerbNet frame syntax xml strings.<br>
 * NOTE: selectional restriction elements are ignored.
 *
 * @author ron
 */
@Component("verbNetFrameSyntaxParser")
public class VerbNetFrameSyntaxParser {
    private static final Logger log =
Logger.getLogger(VerbNetFrameSyntaxParser.class);

    private final DictionaryRepository dictionaryRepository;
    private final VerbNetSelectionalRestrictionsParser vnSelResParser;
}
```

```
/**  
 * @param dictionaryRepository  
 * @param vnSelResParser  
 */  
@Autowired  
public VerbNetFrameSyntaxParser(DictionaryRepository  
dictionaryRepository,  
    VerbNetSelectionalRestrictionsParser vnSelResParser) {  
    this.dictionaryRepository = dictionaryRepository;  
    this.vnSelResParser = vnSelResParser;  
}  
  
/**  
 * @param frameRef -  
 *                  a VerbNet Frame reference  
 * @return  
 */  
public List<SyntaxMatchingRule>  
parseVerbNetFrameSyntax(VerbNetFrameRef frameRef) {  
    VerbNetFrame frame = frameRef.getFrame();  
    List<SyntaxMatchingRule> rules = new  
    ArrayList<SyntaxMatchingRule>();  
    try {  
        DocumentBuilder db =  
DocumentBuilderFactory.newInstance().newDocumentBuilder();  
        Reader reader = new StringReader("<syntax>" + frame.getSyntax() +  
"</syntax>");  
        Document verbNetSyntaxXML = db.parse(new InputSource(reader));  
        NodeIterator nl = XPathAPI.selectNodeIterator(verbNetSyntaxXML,  
"/syntax/*");  
        Node node;  
        while ((node = nl.nextNode()) != null) {  
            String tag = node.getNodeName();  
            if ("NP".equals(tag)) {  
                String roleName = null;  
                if ((node.getAttributes() != null)  
                    && (node.getAttributes().getNamedItem("value") != null)) {  
                    roleName =  
node.getAttributes().getNamedItem("value").getNodeValue();  
                    SemanticRole role =  
SemanticRole.valueOf(roleName.toUpperCase());  
                    if (role == null) {  
                        throw  
SemanticRoleLabelerException.unknownSemanticRole(roleName,  
frameRef);  
                }  
            }  
        }  
    }  
}
```

```

VerbNetRoleRef roleRef =
frameRef.getVnClass().getRoleRef(roleName);
if (roleRef == null) {
    throw SemanticRoleLabelerException.roleNotInClass(roleName,
frameRef);
}
if ((roleRef.getRestrictionXML() != null)
    && (roleRef.getRestrictionXML().length() > 0)
    && (roleRef.getSelectionalRestrictions().size() == 0)) {
    roleRef = vnSelResParser.parseSelectionalRestrictions(roleRef);
}
rules.add(new NounPhraseMatchingRule(roleRef));
}
} else if ("VERB".equals(tag)) {
    rules.add(new VerbMatchingRule());
} else if ("ADJ".equals(tag)) {
    rules.add(new AdjectiveMatchingRule());
} else if ("ADV".equals(tag)) {
    rules.add(new AdverbMatchingRule());
} else if ("PREP".equals(tag)) {
    if ((node.getAttributes() != null)
        && (node.getAttributes().getNamedItem("value") != null)) {
        String filter =
node.getAttributes().getNamedItem("value").getNodeValue();
        rules.add(new PrepositionMatchingRule(filter));
    } else {
        rules.add(new PrepositionMatchingRule());
    }
} else if ("LEX".equals(tag)) {
    if ((node.getAttributes() != null)
        && (node.getAttributes().getNamedItem("value") != null)) {
        String filter =
node.getAttributes().getNamedItem("value").getNodeValue();
        rules.add(new LexicalMatchingRule(filter));
    }
} else {
    throw new Exception("Unexpected element type " + tag);
}
}
} catch (Exception e) {
    log.error("could not parse frame " + frame + " syntax: " +
frame.getSyntax(), e);
    rules.clear();
}
return rules;
}

```

```

protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}
}

```

verbnetimporter.java

```

/*
 * $Id: VerbNetImporter.java,v 1.1 2008/06/18 07:55:26 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.verbnet;

import java.io.File;

import javax.xml.XMLConstants;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;

import org.apache.log4j.Logger;

import edu.harvard.fas.rregan.nlp.verbnet.VNCLASS;

/**
 * @author ron
 */
public class VerbNetImporter {
    protected static final Logger log =
        Logger.getLogger(VerbNetImporter.class);

    /**
     * @param verbNetSchemaFile
     * @param verbNetDataFile
     * @return
     * @throws Exception
     */
    public VNCLASS loadVerbNetClass(File verbNetSchemaFile, File
verbNetDataFile) throws Exception {
        JAXBContext context = JAXBContext.newInstance(VNCLASS.class);
        SchemaFactory schemaFactory =
            SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = schemaFactory.newSchema(verbNetSchemaFile);
        Unmarshaller unmarshaller = context.createUnmarshaller();

```

```

unmarshaller.setSchema(schema);
return (VNCLASS) unmarshaller.unmarshal(verbNetDataFile);
}
}

```

verbnetrole.java

```

/*
 * $Id: VerbNetRole.java,v 1.3 2009/02/11 00:47:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import edu.harvard.fas.rregan.nlp.SemanticRole;

/**
 * @author ron
 */
@Entity
@Table(name = "vnroletype")
public class VerbNetRole implements java.io.Serializable,
Comparable<VerbNetRole> {
    static final long serialVersionUID = 0;

    private Long id;
    private String type;
    private SemanticRole semanticRole;

    protected VerbNetRole() {
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "roletypeid", unique = true, nullable = false)
    public Long getId() {
        return this.id;
    }
}

```

```

protected void setId(Long id) {
    this.id = id;
}

@Column(name = "type", unique = true, nullable = false)
public String getType() {
    return this.type;
}

public void setType(String type) {
    this.type = type;
}

@Enumerated(EnumType.STRING)
@Column(name = "semrole", nullable = false)
public SemanticRole getSemanticRole() {
    return semanticRole;
}

public void setSemanticRole(SemanticRole semanticRole) {
    this.semanticRole = semanticRole;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getType() == null) ? 0 :
                getType().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    return getType().equals(getType());
}

@Override

```

```

    public int compareTo(VerbNetRole o) {
        return getType().compareToIgnoreCase(o.getType());
    }
}

```

verbnetroleref.java

```

/*
 * $Id: VerbNetRoleRef.java,v 1.3 2009/02/11 00:47:45 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

/**
 * A reference from a VerbNet class to a VerbNet role including
 * selectional
 * restrictions for the role in that class.
 *
 * @author ron
 */
@Entity
@Table(name = "vnroleref")
public class VerbNetRoleRef implements Serializable {
    static final long serialVersionUID = 0;

    private Long id;
    private VerbNetClass vnClass;
    private VerbNetRole vnRole;
    private String restrictionXML;
}

```

```

private Set<VerbNetSelectionRestriction> restrictions = new
HashSet<VerbNetSelectionRestriction>();

protected VerbNetRoleRef() {
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "rolerefid", unique = true, nullable = false)
public Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@ManyToOne(targetEntity = VerbNetClass.class, cascade =
CascadeType.ALL, optional = false)
@JoinColumn(name = "classid", nullable = false, unique = false)
public VerbNetClass getVerbNetClass() {
    return vnClass;
}

protected void setVerbNetClass(VerbNetClass vnClass) {
    this.vnClass = vnClass;
}

@ManyToOne(targetEntity = VerbNetRole.class, cascade =
CascadeType.ALL, optional = false)
@JoinColumn(name = "roletypeid", nullable = false, unique = false)
public VerbNetRole getVerbNetRole() {
    return vnRole;
}

protected void setVerbNetRole(VerbNetRole role) {
    this.vnRole = role;
}

@Column(name = "selrestrs", nullable = false, unique = false)
public String getRestrictionXML() {
    return restrictionXML;
}

public void setRestrictionXML(String restrictionXML) {
    this.restrictionXML = restrictionXML;
}

```

```

@OneToMany(targetEntity = VerbNetSelectionRestriction.class, cascade =
{ CascadeType.ALL }, fetch = FetchType.EAGER, mappedBy = "parent")
public Set<VerbNetSelectionRestriction> getSelectionalRestrictions()
{
    return restrictions;
}

protected void
setSelectionalRestrictions(Set<VerbNetSelectionRestriction>
restrictions) {
    this.restrictions = restrictions;
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getRestrictionXML() == null) ? 0 :
getRestrictionXML().hashCode());
        result = prime * result + ((getVerbNetClass() == null) ? 0 :
getVerbNetClass().hashCode());
        result = prime * result + ((getVerbNetRole() == null) ? 0 :
getVerbNetRole().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    final VerbNetRoleRef other = (VerbNetRoleRef) obj;
    if (getRestrictionXML() == null) {
        if (other.getRestrictionXML() != null)
            return false;
    }

```

```

} else if (!getRestrictionXML().equals(other.getRestrictionXML()))
    return false;
if (getVerbNetClass() == null) {
    if (other.getVerbNetClass() != null)
        return false;
} else if (!getVerbNetClass().equals(other.getVerbNetClass()))
    return false;
if (getVerbNetRole() == null) {
    if (other.getVerbNetRole() != null)
        return false;
} else if (!getVerbNetRole().equals(other.getVerbNetRole()))
    return false;
return true;
}

}

```

verbnetselectionalrestrictionsparser.java

```

/*
 * $Id: VerbNetSelectionalRestrictionsParser.java,v 1.2 2009/02/09
 * 10:12:28 rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.srl;

import java.io.Reader;
import java.io.StringReader;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.log4j.Logger;
import org.apache.xpath.XPathAPI;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.traversal.NodeIterator;
import org.xml.sax.InputSource;

import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.VerbNetRoleRef;

```

```

import
edu.harvard.fas.rregan.nlp.dictionary.VerbNetSelectionalRestrictionType;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditVerbNetSelectionalRestrictionCommand;

/**
 * Parser for VerbNet thematic role selectional restrictions. Given a
 * VerbNetRoleRef, parse the selection restrictions xml and add a
 * VerbNetSelectionalRestriction to the role reference.
 *
 * @author ron
 */
@Component("verbNetSelectionalRestrictionsParser")
public class VerbNetSelectionalRestrictionsParser {
    private static final Logger log =
Logger.getLogger(VerbNetFrameSyntaxParser.class);

    private final CommandHandler commandHandler;
    private final DictionaryCommandFactory dictionaryCommandFactory;
    private final DictionaryRepository dictionaryRepository;

    /**
     * @param dictionaryRepository
     */
    @Autowired
    public VerbNetSelectionalRestrictionsParser(CommandHandler commandHandler,
                                                DictionaryCommandFactory dictionaryCommandFactory,
                                                DictionaryRepository dictionaryRepository) {
        this.commandHandler = commandHandler;
        this.dictionaryCommandFactory = dictionaryCommandFactory;
        this.dictionaryRepository = dictionaryRepository;
    }

    /**
     * @param roleRef -
     *          the role reference the restriction applies to
     * @param selrestrs -
     *          a VerbNet selrestrs xml string
     * @return the updated roleRef with the selectional restrictions
     * added.
     */

```

```

public VerbNetRoleRef parseSelectionalRestrictions(VerbNetRoleRef
roleRef) {
    try {
        DocumentBuilder db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Reader reader = new StringReader(roleRef.getRestrictionXML());
        Document selrestrsXML = db.parse(new InputSource(reader));
        NodeIterator nl = XPathAPI.selectNodeIterator(selrestrsXML,
"/SELRESTR/*");
        Node node;
        while ((node = nl.nextNode()) != null) {
            String tag = node.getNodeName();
            if ("SELRESTR".equals(tag)) {
                String value = "+";
                String typeName = null;
                if (node.getAttributes() != null) {
                    if (node.getAttributes().getNamedItem("Value") != null) {
                        value =
node.getAttributes().getNamedItem("Value").getNodeValue();
                    }
                    if (node.getAttributes().getNamedItem("type") != null) {
                        typeName =
node.getAttributes().getNamedItem("type").getNodeValue();
                    }
                }
                if ((typeName != null) && ("+".equals(value) ||
"-=".equals(value))) {
                    // TODO: this needs to be a command
                    VerbNetSelectionRestrictionType type =
getDictionaryRepository()
                    .findVerbNetSelectionRestrictionType(typeName);
                    EditVerbNetSelectionRestrictionCommand command =
dictionaryCommandFactory
                    .newEditVerbNetSelectionRestrictionCommand();
                    command.setVerbNetRoleRef(roleRef);
                    command.setVerbNetSelectionRestrictionType(type);
                    command.setInclude(value);
                    command = commandHandler.execute(command);
                    roleRef = command.getVerbNetRoleRef();
                }
            } else {
                throw new Exception("Unexpected element type " + tag);
            }
        }
    } catch (Exception e) {
        log.error("could not parse thematic role selectional restriction
for " + roleRef + ":" +
        + roleRef.getRestrictionXML(), e);
    }
}

```

```

    }
    return roleRef;
}

protected DictionaryRepository getDictionaryRepository() {
    return dictionaryRepository;
}
}

```

verbnetselectionrestriction.java

```

/*
 * $Id: VerbNetSelectionRestriction.java,v 1.4 2009/02/09 10:12:29
rregan Exp $
 * Copyright (c) 2009 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.dictionary;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

/**
 * A restriction includes or excludes a restriction type.
 */
@Entity
@Table(name = "vnroleselres")
public class VerbNetSelectionRestriction {

    private Long id;
    private VerbNetRoleRef parent;
    private VerbNetSelectionRestrictionType type;
    private String include;

    /**
     * @param parent
     * @param type
     */

```

```

 * @param include
 */
public VerbNetSelectionRestriction(VerbNetRoleRef parent,
VerbNetSelectionRestrictionType type,
    String include) {
    setParent(parent);
    setType(type);
    setInclude(include);
}

protected VerbNetSelectionRestriction() {
    // for hibernate
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "roleselresid", unique = true, nullable = false)
public Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@ManyToOne(targetEntity = VerbNetRoleRef.class, cascade =
CascadeType.ALL, optional = false)
@JoinColumn(name = "rolerefid", nullable = false, unique = false)
public VerbNetRoleRef getParent() {
    return parent;
}

protected void setParent(VerbNetRoleRef parent) {
    this.parent = parent;
}

@ManyToOne(targetEntity = VerbNetSelectionRestrictionType.class,
cascade = CascadeType.ALL, optional = false)
@JoinColumn(name = "roletypeid", nullable = false, unique = false)
public VerbNetSelectionRestrictionType getType() {
    return type;
}

protected void setType(VerbNetSelectionRestrictionType type) {
    this.type = type;
}

```

```

public String getInclude() {
    return include;
}

protected void setInclude(String include) {
    this.include = include;
}

@Override
public String toString() {
    return getClass().getSimpleName() + " " + getInclude() + getType();
}

```

verbnetselectionrestrictiontype.java

```

/*
 * $Id: VerbNetSelectionRestrictionType.java,v 1.2 2009/02/09 10:12:30
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlTransient;

/**
 * A selection restriction on a VerbNet semantic role from the frame
syntax
 * element. Each restriction is assigned a synset and link type
indicating the
 * type of entities that fit the restriction.<br>
 * For example:<br>
 * The "location" restriction maps to the "location#n#1" synset with a
link type

```

```

* of "hyponym" meaning the 2000 or so hyponyms of this synset comply
with the
* restriction.<br>
* The "refl" restriction (self reference) maps to the "reflexive"
synset with a
* link type of "hyponym instance" meaning a word that is a reflexive
pronoun is
* appropriate for the restriction, such as himself, herself, or
itself.
*
* @author ron
*/
@Entity
@Table(name = "vnselres")
public class VerbNetSelectionRestrictionType implements
Comparable<VerbNetSelectionRestrictionType>, Serializable {
static final long serialVersionUID = 0;

private Long id;
private String name;
private Synset synset;
private Linkdef linkType;

protected VerbNetSelectionRestrictionType() {
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
@Column(name = "vnselresid", unique = true, nullable = false)
public Long getId() {
    return id;
}

protected void setId(Long id) {
    this.id = id;
}

@Column()
public String getName() {
    return this.name;
}

protected void setName(String name) {
    this.name = name;
}

```

```

@ManyToOne(targetEntity = Synset.class, cascade = CascadeType.ALL,
optional = false)
@JoinColumn(name = "synsetid", insertable = false, updatable = false)
public Synset getSynset() {
    return synset;
}

protected void setSynset(Synset synset) {
    this.synset = synset;
}

@ManyToOne(targetEntity = Linkdef.class, cascade = CascadeType.ALL,
optional = true)
@JoinColumn(name = "linkid", insertable = false, updatable = false)
@XmlTransient
public Linkdef getLinkType() {
    return linkType;
}

protected void setLinkType(Linkdef linkType) {
    this.linkType = linkType;
}

@Override
public int compareTo(VerbNetSelectionRestrictionType o) {
    return getName().compareToIgnoreCase(o.getName());
}

private Integer tmpHashCode = null;

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getName() == null) ? 0 :
getName().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {

```

```

if (this == obj) {
    return true;
}
if (obj == null) {
    return false;
}
if (!getClass().isAssignableFrom(obj.getClass())) {
    return false;
}
final VerbNetFrame other = (VerbNetFrame) obj;
if ((getId() != null) && getId().equals(other.getId())) {
    return true;
}
if (getName() == null) {
    if (other.getName() != null) {
        return false;
    }
} else if (!getName().equals(other.getName())) {
    return false;
}
return true;
}

@Override
public String toString() {
    return getClass().getSimpleName() + " " + getName();
}
}

```

word.java

```

/*
 * $Id: Word.java,v 1.3 2009/01/30 10:36:32 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import java.util.TreeSet;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;

```

```

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Transient;
import javax.persistence.UniqueConstraint;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElementRef;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlID;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.adapters.XmlAdapter;
import javax.xml.bind.adapters.XmlJavaTypeAdapter;

import org.hibernate.annotations.Sort;
import org.hibernate.annotations.SortType;

import edu.harvard.fas.rregan.nlp.PartOfSpeech;

/**
 * Wordnet Word
 */
@Entity
@Table(name = "word", uniqueConstraints =
@UniqueConstraint(columnNames = "lemma"))
@XmlRootElement(name = "word")
public class Word implements Comparable<Word>, Serializable {
    static final long serialVersionUID = 0;

    private Long id;
    private String lemma;
    private String phoneticCode; // for Jazzy spell checker
    private Set<Sense> senses = new TreeSet<Sense>();

    /**
     * Create a new word with the supplied text
     *
     * @param lemma -
     *          the text of the word
     * @param phoneticCode -
     */
    public Word(String lemma, String phoneticCode) {
        setLemma(lemma);
        setPhoneticCode(phoneticCode);
    }
}

```

```

}

protected Word() {
}

@Id
@Column(name = "wordid", unique = true, nullable = false)
@GeneratedValue(strategy = GenerationType.AUTO)
@XmlID
@XmlAttribute(name = "id")
@XmlJavaTypeAdapter(IdAdapter.class)
public Long getId() {
    return this.id;
}

protected void setId(Long id) {
    this.id = id;
}

@Column(name = "lemma", unique = true, nullable = false, length = 80)
@XmlAttribute(name = "lemma")
public String getLemma() {
    return this.lemma;
}

public void setLemma(String lemma) {
    this.lemma = lemma;
}

@Column(name = "phonetic_code", unique = false, nullable = true,
length = 80)
@XmlAttribute(name = "phoneticCode")
public String getPhoneticCode() {
    return this.phoneticCode;
}

public void setPhoneticCode(String phoneticCode) {
    this.phoneticCode = phoneticCode;
}

@OneToMany(targetEntity = Sense.class, cascade = { CascadeType.ALL },
fetch = FetchType.LAZY)
@JoinColumn(name = "wordid")
@Sort(type = SortType.NATURAL)
@XmlElementWrapper(name = "senses")
@XmlElementRef(name = "sense")
public Set<Sense> getSenses() {
    return senses;
}

public void setSenses(Set<Sense> senses) {
    this.senses = senses;
}

/**
 * Get the sense for a particular part of speech.
 *
 * @param pos
 * @return
 */
@Transient
@XmlTransient
public Set<Sense> getSenses(PartOfSpeech pos) {
    Set<Sense> senses = new HashSet<Sense>(getSenses().size());
    for (Sense sense : getSenses()) {
        if (sense.getSynset().isPartOfSpeech(pos)) {
            senses.add(sense);
        }
    }
    return senses;
}

/**
 * @param pos
 * @param rank
 * @return A specific sense by part of speech and rank
 */
@Transient
@XmlTransient
public Sense getSense(PartOfSpeech pos, Integer rank) {
    for (Sense sense : getSenses(pos)) {
        if (rank.equals(sense.getRank())) {
            return sense;
        }
    }
    return null;
}

@Override
public int compareTo(Word o) {
    return getLemma().compareTo(o.getLemma());
}

private Integer tmpHashCode = null;

```

```

@Override
public int hashCode() {
    if (tmpHashCode == null) {
        if (getId() != null) {
            tmpHashCode = new Integer(getId().hashCode());
        }
        final int prime = 31;
        int result = 1;
        result = prime * result + ((getLemma() == null) ? 0 :
getLemma().hashCode());
        tmpHashCode = new Integer(result);
    }
    return tmpHashCode.intValue();
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (!getClass().isAssignableFrom(obj.getClass())) {
        return false;
    }
    final Word other = (Word) obj;
    if ((getId() != null) && getId().equals(other.getId())) {
        return true;
    }
    if (getLemma() == null) {
        if (other.getLemma() != null) {
            return false;
        }
    } else if (!getLemma().equals(other.getLemma())) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return lemma;
}

/**

```

```

     * This class is used by JAXB to convert the id of an entity into an
     * xml id
     * string that will be distinct from other entity xml id strings by
     * the use
     * of a prefix.
     *
     * @author ron
     */
public static class IdAdapter extends XmlAdapter<String, Long> {
    private static final String prefix = "WORD_";

    @Override
    public Long unmarshal(String id) throws Exception {
        return new Long(id.substring(prefix.length()));
    }

    @Override
    public String marshal(Long id) throws Exception {
        if (id != null) {
            return prefix + id.toString();
        }
        return "";
    }
}

```

wordnet.java

```

/*
 * $Id: WordNet.java,v 1.1 2008/12/08 04:38:33 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */

package edu.harvard.fas.rregan.nlp.impl.wordnet;

import java.io.InputStream;
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;

import net.didion.jwnl.JWNL;
import net.didion.jwnl.data.IndexWord;
import net.didion.jwnl.data.POS;
import net.didion.jwnl.dictionary.Dictionary;

import org.apache.log4j.Logger;

```

```

import com.nexagis.jawbone.filter.SimilarFilter;

/**
 * @author ron
 */
public class WordNet {
    private static final Logger log = Logger.getLogger(WordNet.class);

    /**
     * The name of the property in the NLPImpl.properties file that
     * contains the
     * path to the JWNL configuration/properties file relative to the
     * classpath.
     * By default the path is "resources/nlp/jwnl/file_properties.xml"
     */
    public static final String PROP_JWNL_WORDNET_PROP_FILE =
"JWNLPropertyFile";

    private Dictionary jwnlDictionary;

    public WordNet() {
        log.debug("initializing WordNet.");
        String propertiesFile = "resources/nlp/jwnl/file_properties.xml";
        InputStream propertiesStream =
getClass().getClassLoader().getResourceAsStream(
            propertiesFile);
        try {
            JWNL.initialize(propertiesStream);
            jwnlDictionary = Dictionary.getInstance();
        } catch (Exception e) {
            log.error("Problem initializing WordNet: " + e, e);
        }
    }

    // return the POS for the supplied Penn Treebank tag string or words
    "Verb"
    // "Noun" etc.
    protected POS getPartOfSpeechFromString(String partOfSpeech) {
        POS pos = POS.getPOSForLabel(partOfSpeech);
        if (pos == null) {
            if (partOfSpeech.startsWith("V")) {
                pos = POS.VERB;
            } else if (partOfSpeech.startsWith("N")) {
                pos = POS.NOUN;
            } else if (partOfSpeech.startsWith("J")) {
                pos = POS.ADJECTIVE;
            } else if (partOfSpeech.startsWith("RB")) {
                pos = POS.ADVERB;
            }
        }
        log.debug(pos == null ? "null" : pos.toString());
        return pos;
    }

    public void bla(String pos, String lemma) {
        try {
            IndexWord iw =
jwnlDictionary.lookupIndexWord(getPartOfSpeechFromString(pos), lemma);
        } catch (Exception e) {
            log.error(e, e);
        }
    }

    public boolean isSearchablePartOfSpeech(String partOfSpeech) {
        return getPartOfSpeechFromString(partOfSpeech) != null;
    }

    public boolean knownWord(String partOfSpeech, String word) {
        log.debug("partOfSpeech = " + partOfSpeech + " word = " + word);
        if (jwnlDictionary != null) {
            try {
                POS pos = getPartOfSpeechFromString(partOfSpeech);
                return
jwnlDictionary.getMorphologicalProcessor().lookupBaseForm(pos, word) !=
null;
            } catch (Exception e) {
                log.error("Problem in JWNL lookupBaseForm(): " + e, e);
            }
        }
        return false;
    }

    public Set<String> similarWords(String partOfSpeech, String word) {
        return similarWords(partOfSpeech, word, 2, 20);
    }

    public Set<String> similarWords(String partOfSpeech, String word, int
maxDistance,
        int maxResults) {
        log.debug("partOfSpeech = " + partOfSpeech + " word = " + word + "
maxDistance =
        + maxDistance + " maxResults = " + maxResults);
        Set<String> similarWords = new TreeSet<String>();


```

```

if (jwnlDictionary != null) {
    try {
        POS pos = getPartOfSpeechFromString(partOfSpeech);
        // this is slow, but find the most similar words first
        if (pos != null) {
            for (int distance = 0; distance <= maxDistance; distance++) {
                if ((maxResults != 0) && (similarWords.size() < maxResults)) {
                    SimilarFilter filter = new SimilarFilter(word, true, distance);
                    Iterator<IndexWord> iter =
jwnlDictionary.getIndexWordIterator(pos);
                    while (iter.hasNext()) {
                        IndexWord nextWord = iter.next();
                        if (filter.accept(nextWord.getLemma())) {
                            similarWords.add(nextWord.getLemma());
                        }
                    }
                    if ((maxResults > 0) && (similarWords.size() == maxResults)) {
                        break;
                    }
                }
            }
        }
    } catch (Exception e) {
        log.error("Problem in JWNL lookupBaseForm(): " + e, e);
    }
}
return similarWords;
}

public String lemmatize(String partOfSpeech, String word) {
log.debug("partOfSpeech = " + partOfSpeech + " word = " + word);
if (jwnlDictionary != null) {
    try {
        POS pos = getPartOfSpeechFromString(partOfSpeech);
        if (pos != null) {
            IndexWord indexWord = jwnlDictionary.getMorphologicalProcessor()
                .lookupBaseForm(pos, word);
            if (indexWord != null) {
                String lemma = indexWord.getLemma();
                log.debug("partOfSpeech = " + partOfSpeech + " word = " + word
                    + " lemma = " + lemma);
                return lemma;
            }
        }
    } catch (Exception e) {
        log.error("Problem in JWNL lookupBaseForm(): " + e, e);
    }
}

```

```

        }
        return word;
    }
}

```

wordnetdefinitionwordsinitializer.java

```

/*
 * $Id: WordNetDefinitionWordsInitializer.java,v 1.3 2009/01/26
10:19:01 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import java.io.IOException;
import java.io.InputStream;
import java.util.zip.GZIPInputStream;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.LoadWordNetTaggedGlosses
Command;

/**
 * Create SynsetDefinitionWords from the wsd field in the synsets.
 *
 * @author ron
 */
@Component("wordNetDefinitionWordsInitializer")
@Scope("prototype")
public class WordNetDefinitionWordsInitializer extends
AbstractSystemInitializer {

    /**
     * The name of the property in the

```

```

 * WordNetDefinitionWordsInitializer.properties file that contains
the path
 * to the directory containing all the WordNet merged glosstag files,
if not
 * supplied the default file is "resources/nlp/jwnl/wn30/"
 */
public static final String PROP_WORDNET_MERGED_GLOSS_FILES_DIRECTORY
= "WordNetMergedGlossFilesDirectory";
public static final String
PROP_WORDNET_MERGED_GLOSS_FILES_DIRECTORY_DEFAULT =
"resources/nlp/jwnl/wn30/";

/**
 * The name of the property in the
 * WordNetDefinitionWordsInitializer.properties file that contains a
comma
 * delimited list of merged glosstag xml file names. The files are
expected
 * to be in the directory defined by
 * PROP_WORDNET_MERGED_GLOSS_FILES_DIRECTORY. If not supplied the
default
 * list is "categorydef.sql.gz, word.sql.gz, synset.sql.gz,
sense.sql.gz"
 * NOTE: the files may be plain sql files or zipped sql files.
 */
public static final String PROP_WORDNET_MERGED_GLOSS_FILES =
"WordNetMergedGlossFiles";
public static final String PROP_DWORDNET_MERGED_GLOSS_FILES_DEFAULT =
"adj.xml.gz, adv.xml.gz, noun.xml.gz, verb.xml.gz";

private final DictionaryRepository dictionaryRepository;
private final CommandHandler commandHandler;
private final DictionaryCommandFactory dictionaryCommandFactory;

/**
 * @param dictionaryRepository
 * @param commandHandler
 * @param dictionaryCommandFactory
 */
@.Autowired
public WordNetDefinitionWordsInitializer(DictionaryRepository
dictionaryRepository,
CommandHandler commandHandler, DictionaryCommandFactory
dictionaryCommandFactory) {
super(16);
this.dictionaryRepository = dictionaryRepository;
this.commandHandler = commandHandler;
}

```

```

this.dictionaryCommandFactory = dictionaryCommandFactory;
}

@Override
public void initialize() {
try {
// TODO: this disables the initializer and shouldn't be hard coded
if (true) {
return;
}
log.info("initializing WordNet tagged definitions...");
if (dictionaryRepository.buildSynsetDefinitionWords()) {
ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
WordNetDefinitionWordsInitializer.class.getName());
String directory = resourceBundleHelper.getString(
PROP_WORDNET_MERGED_GLOSS_FILES_DIRECTORY,
PROP_WORDNET_MERGED_GLOSS_FILES_DIRECTORY_DEFAULT);
String files =
resourceBundleHelper.getString(PROP_WORDNET_MERGED_GLOSS_FILES,
PROP_DWORDNET_MERGED_GLOSS_FILES_DEFAULT);
for (String file : files.split(",")) {
file = file.trim();
LoadWordNetTaggedGlossesCommand command =
dictionaryCommandFactory
.newWordNetTaggedGlossaryDigester();
command.setInputStream(getDataFileInputStream(directory + file));
// execute the LoadWordNetTaggedGlossesCommand directly
// (without the command handler)
// so that a transaction isn't started. Each inner command
// (batch for one synset) will be
// executed in its own transaction.
command.execute();
// commandHandler.execute(command);
}
}
} catch (Exception e) {
log.error("failed to load WordNet tagged gloss files: " + e, e);
}
}

private InputStream getDataFileInputStream(String dataFilePath)
throws IOException {
log.debug("loading data file " + dataFilePath);
InputStream dataInputStream =
getClass().getClassLoader().getResourceAsStream(dataFilePath);
if (dataFilePath.endsWith(".gz")) {

```

```

    dataInputStream = new GZIPIInputStream(dataInputStream);
}
return dataInputStream;
}
}

```

wordnethyponymcountinitializer.java

```

/*
 * $Id: WordNetHyponymCountInitializer.java,v 1.3 2009/01/26 10:19:01
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import java.util.Collection;
import java.util.Deque;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.command.BatchCommand;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.dictionary.Dictionary;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Linkdef;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSemlinkRefCommand;
import
edu.harvard.fas.rregan.nlp.dictionary.command.EditSynsetCommand;

/**
 * Count all the hyponyms that are subsumed by a synset.
 *
 * @author ron

```

```

*/
@Component("wordNetHyponymCountInitializer")
@Scope("prototype")
public class WordNetHyponymCountInitializer extends
AbstractSystemInitializer {

    private final DictionaryRepository dictionaryRepository;
    private final CommandHandler commandHandler;
    private final DictionaryCommandFactory dictionaryCommandFactory;
    private final Map<Long, SynsetNode> semanticGraph = new HashMap<Long,
SynsetNode>();
    private final Map<Long, Linkdef> semanticLinkTypes = new
HashMap<Long, Linkdef>();

    /**
     * @param dictionaryRepository
     * @param commandHandler
     * @param dictionaryCommandFactory
     */
    @Autowired
    public WordNetHyponymCountInitializer(DictionaryRepository
dictionaryRepository,
        CommandHandler commandHandler, DictionaryCommandFactory
dictionaryCommandFactory) {
        super(20);
        this.dictionaryRepository = dictionaryRepository;
        this.commandHandler = commandHandler;
        this.dictionaryCommandFactory = dictionaryCommandFactory;
    }

    @Override
    public void initialize() {
        try {
            // TODO: this disables the initializer and shouldn't be hard coded
            if (true) {
                return;
            }
            if (dictionaryRepository.buildSynsetLinkPathsAndCounts()) {
                // load the link types in a map for quick references
                for (Linkdef linkType : dictionaryRepository.findLinkdefs()) {
                    semanticLinkTypes.put(linkType.getId(), linkType);
                }
                // getting the dictionary avoids the overhead of wrapping all
                // the synsets in proxy on return from the repository.
                Dictionary dictionary = dictionaryRepository.getDictionary();
                Collection<Synset> synsets = dictionary.getSynsets();
            }
        }
    }
}

```

```

for (Object[] semanticLink : dictionaryRepository.findSemanticLinks()) {
    // log.info(hypernymLink);
    Long fromSynset = (Long) semanticLink[0];
    Long toSynset = (Long) semanticLink[1];
    Long linkType = (Long) semanticLink[2];
    Integer distance = (Integer) semanticLink[3];

    SynsetNode fromSynsetNode = semanticGraph.get(fromSynset);
    if (fromSynsetNode == null) {
        fromSynsetNode = new SynsetNode(fromSynset);
        semanticGraph.put(fromSynset, fromSynsetNode);
    }
    Set<SemlinkPath> semlinkPaths =
fromSynsetNode.semanticPaths.get(linkType);
    if (semlinkPaths == null) {
        semlinkPaths = new HashSet<SemlinkPath>();
        fromSynsetNode.semanticPaths.put(linkType, semlinkPaths);
    }
    semlinkPaths.add(new SemlinkPath(linkType, toSynset, distance));
    // build a node for the target too, otherwise root nodes
    // in the graph won't have a SynsetNode
    SynsetNode targetNode = semanticGraph.get(toSynset);
    if (targetNode == null) {
        semanticGraph.put(toSynset, new SynsetNode(toSynset));
    }
}

int count = 0;
for (Long synset : semanticGraph.keySet()) {
    count++;
    if (count % 1000 == 0) {
        log.info("walking " + count);
    }
    walkSynsetRelations(synset);
}

count = 0;
BatchCommand batchCommand =
dictionaryCommandFactory.newBatchCommand();
for (Synset synset : synsets) {
    count++;
    if (count % 100 == 0) {
        commandHandler.execute(batchCommand);
        batchCommand = dictionaryCommandFactory.newBatchCommand();
        log.info("generating updates " + count);
    }
}

}
SynsetNode node = semanticGraph.get(synset.getId().intValue());
EditSynsetCommand command =
dictionaryCommandFactory.newEditSynsetCommand();
batchCommand.addCommand(command);
command.setSynset(synset);
if (node != null) {
    Map<Long, Integer> subsumerCounts = new HashMap<Long,
Integer>();
    for (Long linkType : node.subsumerCounts.keySet()) {
        subsumerCounts.put(linkType,
node.subsumerCounts.get(linkType).count);
    }
    command.setSubsumerCounts(subsumerCounts);

    if (node.semanticPaths != null) {
        for (Long linkType : node.semanticPaths.keySet()) {
            for (SemlinkPath spd : node.semanticPaths.get(linkType)) {
                EditSemlinkRefCommand editSemlinkRefCommand =
dictionaryCommandFactory
                    .newEditSemlinkRefCommand();
                editSemlinkRefCommand.setFromSynset(node.synsetId);
                editSemlinkRefCommand.setToSynset(spd.synsetId);
                editSemlinkRefCommand.setLinkDef(spd.linkdefid);
                editSemlinkRefCommand.setDistance(spd.distance);
                batchCommand.addCommand(editSemlinkRefCommand);
            }
        }
    }
}

} else {
    command.setSubsumerCounts(null);
}
}
}
}
} catch (Exception e) {
log.error("failed to initialize wordnet hyponym counts: " + e, e);
} catch (Error e) {
log.error("failed to initialize wordnet hyponym counts: " + e, e);
}
}

/**
 * Walk all the hypernym paths from the startSynset to the root of
the tree
 * that contains it. Increment all the hyponym counts of the
ancestors of

```

```

* the start synset and add the ancestor plus distance to the set of
* ancestor paths. Make sure the start synset is only counted once
for each
    * ancestor even though the ancestor may be contained in multiple
paths.
    *
    * @param startSynset
    */
private void walkSynsetRelations(Long startSynset) {
    SynsetNode startSynsetNode = semanticGraph.get(startSynset);
    for (Long linkType : startSynsetNode.semanticPaths.keySet()) {
        // only expand the paths of link types that are transitive
        if (semanticLinkTypes.get(linkType).getRecurses() && (linkType != 2)) {
            walkSynsetRelations(startSynset, linkType);
        }
    }
}

private void walkSynsetRelations(Long startSynset, Long linkType) {
    int tick = 0;
    SynsetNode startSynsetNode = semanticGraph.get(startSynset);
    Set<SemlinkPath> semanticPaths =
    startSynsetNode.semanticPaths.get(linkType);
    Set<SemlinkPath> synsetsVisited = new HashSet<SemlinkPath>();
    Deque<SemlinkPath> pathsToVisit = new LinkedList<SemlinkPath>();
    // add a path to itself, i.e. a synset subsumes itself, and it is
    // the LCS between itself and all its subsumes
    pathsToVisit.push(new SemlinkPath(linkType, startSynset, 0));
    while (!pathsToVisit.isEmpty()) {
        SemlinkPath currentPath = pathsToVisit.pop();
        if (!synsetsVisited.contains(currentPath)) {
            tick++;
            synsetsVisited.add(currentPath);
            semanticPaths.add(currentPath);
            SynsetNode synsetNode = semanticGraph.get(currentPath.synsetid);
            // paths of the current node being visited
            Set<SemlinkPath> paths = synsetNode.semanticPaths.get(linkType);
            if (paths != null) {
                for (SemlinkPath path : paths) {
                    // only look at the immediate links of the current node.
                    if (path.distance == 1) {
                        Long linkTarget = path.synsetid;
                        SynsetNode linkTargetNode = semanticGraph.get(linkTarget);
                        Counter subsumerCounter =
                            linkTargetNode.subsumerCounts.get(linkType);
                        if (subsumerCounter == null) {

```

```

                            subsumerCounter = new Counter();
                            linkTargetNode.subsumerCounts.put(linkType, subsumerCounter);
                        }
                        subsumerCounter.count++;
                        pathsToVisit.push(new SemlinkPath(linkType, linkTarget,
                            currentPath.distance + 1));
                    }
                }
            }
        }
    }

public static class SynsetNode {
    public Long synsetid;
    public final Map<Long, Set<SemlinkPath>> semanticPaths = new
    HashMap<Long, Set<SemlinkPath>>();
    public final Map<Long, Counter> subsumerCounts = new HashMap<Long,
    Counter>();

    public SynsetNode(Long synsetid) {
        this.synsetid = synsetid;
    }

    @Override
    public boolean equals(Object o) {
        if (o instanceof SynsetNode) {
            return (((SynsetNode) o).synsetid == this.synsetid);
        }
        return false;
    }

    @Override
    public int hashCode() {
        return synsetid.hashCode();
    }

    @Override
    public String toString() {
        return "SynsetNode[" + synsetid + "]";
    }
}

public static class SemlinkPath {
    public Long linkdefid;
    public Long synsetid;
    public Integer distance;
}
```

```

public SemlinkPath(Long linkdefid, Long synsetid, Integer distance)
{
    this.linkdefid = linkdefid;
    this.synsetid = synsetid;
    this.distance = distance;
}

@Override
public boolean equals(Object o) {
    if (o instanceof SemlinkPath) {
        SemlinkPath other = (SemlinkPath) o;
        if (!linkdefid.equals(other.linkdefid)) {
            return false;
        }
        return (synsetid.equals(other.synsetid));
    }
    return false;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((synsetid == null) ? 0 :
synsetid.hashCode());
    result = prime * result + ((linkdefid == null) ? 0 :
linkdefid.hashCode());
    return result;
}

@Override
public String toString() {
    return "SemlinkPath[" + synsetid + ", " + linkdefid + ", " +
distance + "]";
}

/**
 * holds a count that can be incremented.
 *
 * @author ron
 */
public static class Counter {
    public int count = 0;
}
}

```

wordnetsensekeyinitializer.java

```

/*
 * $Id: WordNetSenseKeyInitializer.java,v 1.3 2009/01/26 10:19:01
rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.dictionary.impl.repository.init;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.zip.GZIPInputStream;

import net.didion.jwnl.data.POS;

import org.springframework.beans.factory.annotation.Autowired;

import edu.harvard.fas.rregan.AbstractSystemInitializer;
import edu.harvard.fas.rregan.ResourceBundleHelper;
import edu.harvard.fas.rregan.command.CommandHandler;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import
edu.harvard.fas.rregan.nlp.dictionary.command.DictionaryCommandFactory
;
import edu.harvard.fas.rregan.nlp.dictionary.command.EditSenseCommand;
import edu.harvard.fas.rregan.requel.NoSuchEntityException;

/**
 * Create SynsetDefinitionWords from the wsdl field in the synsets.
 *
 * @author ron
 */
// @Component("wordNetSenseKeyInitializer")
// @Scope("prototype")
public class WordNetSenseKeyInitializer extends
AbstractSystemInitializer {

    /**
     * The name of the property in the
WordNetSenseKeyInitializer.properties
     * file that contains the path of the sense.index file, if not
supplied the

```

```

 * default file is "resources/nlp/jwnl/wn30/index.sense"
 */
public static final String PROP_WORDNET_SENSE_INDEX_FILE =
"WordNetSenseIndexFile";
public static final String PROP_WORDNET_SENSE_INDEX_FILE_DEFAULT =
"resources/nlp/jwnl/wn30/index.sense";

private final DictionaryRepository dictionaryRepository;
private final CommandHandler commandHandler;
private final DictionaryCommandFactory dictionaryCommandFactory;
private net.didion.jwnl.dictionary.Dictionary jwnlDictionary;

/**
 * @param dictionaryRepository
 * @param commandHandler
 * @param dictionaryCommandFactory
 */
@.Autowired
public WordNetSenseKeyInitializer(DictionaryRepository
dictionaryRepository,
    CommandHandler commandHandler, DictionaryCommandFactory
dictionaryCommandFactory) {
    super(15);
    this.dictionaryRepository = dictionaryRepository;
    this.commandHandler = commandHandler;
    this.dictionaryCommandFactory = dictionaryCommandFactory;
}

@Override
public void initialize() {
    // TODO: this disables the initializer and shouldn't be hard coded
    if (true) {
        return;
    }
    log.info("initializing WordNet sense sense keys...");
    try {
        if (dictionaryRepository.buildSenseKeys()) {
            ResourceBundleHelper resourceBundleHelper = new
ResourceBundleHelper(
                WordNetSenseKeyInitializer.class.getName());
            String senseIndexPath = resourceBundleHelper.getString(
                PROP_WORDNET_SENSE_INDEX_FILE,
                PROP_WORDNET_SENSE_INDEX_FILE_DEFAULT);

            BufferedReader reader = new BufferedReader(new InputStreamReader(
                getDataFileInputStream(senseIndexPath)));
            String indexLine;

```

```

while ((indexLine = reader.readLine()) != null) {
    String[] indexEntries = indexLine.split(" ");
    String senseKey = indexEntries[0];
    String lemma = senseKey.substring(0, senseKey.indexOf('%'));
    String synsetTypeCode = senseKey.substring(lemma.length() + 1,
senseKey
        .indexOf(':'));
    String offset = indexEntries[1];
    // TODO: map the type code to the base value like 5 - > 3
    if (synsetTypeCode.equals("5")) {
        synsetTypeCode = "3";
    }
    Long synsetId = new Long(synsetTypeCode + offset);
    try {
        Sense sense =
dictionaryRepository.findSensesByLemmaAndSynsetId(lemma
            .replace('_', ' '), synsetId);
        EditSenseCommand command =
dictionaryCommandFactory.newEditSenseCommand();
        command.setSense(sense);
        command.setSenseKey(senseKey);
        command = commandHandler.execute(command);
        log.info(sense + " -> " + senseKey);
    } catch (NoSuchEntityException e) {
        log.warn("no sense for " + lemma + " " + synsetId);
    }
}
} catch (Exception e) {
    log.error("failed to initialize semcor: " + e, e);
}
}

private InputStream getDataFileInputStream(String dataFilePath)
throws IOException {
    log.debug("loading data file " + dataFilePath);
    InputStream dataInputStream =
getClass().getClassLoader().getResourceAsStream(dataFilePath);
    if (dataFilePath.endsWith(".gz")) {
        dataInputStream = new GZIPInputStream(dataInputStream);
    }
    return dataInputStream;
}

private POS getWordNetPOSFromPartOfSpeech(PartOfSpeech partOfSpeech)
{
    if (partOfSpeech.in(PartOfSpeech.NOUN)) {

```

```

    return POS.NOUN;
} else if (partOfSpeech.in(PartOfSpeech.VERB)) {
    return POS.VERB;
} else if (partOfSpeech.in(PartOfSpeech.ADVERB)) {
    return POS.ADVERB;
} else if (partOfSpeech.in(PartOfSpeech.ADJECTIVE)) {
    return POS.ADJECTIVE;
}
return null;
}
}

```

wordnetwsd.java

```

/*
 * $Id: WordnetWSD.java,v 1.2 2009/03/27 07:16:09 rregan Exp $
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.nlp.impl.wsd;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import edu.harvard.fas.rregan.nlp.GrammaticalStructureLevel;
import edu.harvard.fas.rregan.nlp.NLPPProcessor;
import edu.harvard.fas.rregan.nlp.NLPText;
import edu.harvard.fas.rregan.nlp.PartOfSpeech;
import edu.harvard.fas.rregan.nlp.dictionary.DictionaryRepository;
import edu.harvard.fas.rregan.nlp.dictionary.Linkdef;
import edu.harvard.fas.rregan.nlp.dictionary.SemcorSentenceWord;
import edu.harvard.fas.rregan.nlp.dictionary.Sense;
import edu.harvard.fas.rregan.nlp.dictionary.Synset;
import edu.harvard.fas.rregan.nlp.dictionary.SynsetDefinitionWord;
import edu.harvard.fas.rregan.nlp.dictionary.Word;
import
edu.harvard.fas.rregan.nlp.impl.wsd.ColocationSenseRelationInfo.Colo-
cationSource;
import edu.harvard.fas.rregan.request.NoSuchEntityException;

```

```

    /**
     * A dictionary based word sense disambiguator that uses the
     information content
     * and semantic similarity of WordNet words to disambiguate terms.
     *
     * @author ron
     */
    @Component("WordNetWSD")
    public class WordnetWSD implements NLPPProcessor<NLPText> {
        private static final Logger log = Logger.getLogger(WordnetWSD.class);

        private final DictionaryRepository dictionaryRepository;

        /**
         * @param dictionaryRepository
         * @param processorFactory
         */
        @Autowired
        public WordnetWSD(DictionaryRepository dictionaryRepository) {
            this.dictionaryRepository = dictionaryRepository;
        }

        @Override
        public NLPText process(NLPText text) {
            if (text.in(GrammaticalStructureLevel.PARAGRAPH)) {
                for (NLPText sentence : text.getChildren()) {
                    process(sentence);
                }
            } else {
                // identify all the dictionary words with a single sense for the
                // given part of speech (monosemous words)
                for (NLPText word : text.getLeaves()) {
                    Word dicWord = word.getDictionaryWord();
                    if (dicWord != null) {
                        Set<Sense> senses = dicWord.getSenses(word.getPartOfSpeech());
                        if (senses.size() == 1) {
                            word.setDictionaryWordSense(senses.iterator().next());
                        }
                    }
                }
                // disambiguate the remaining words
                disambiguate(text);
            }
            return text;
        }
    }

```

```

/**
 * disambiguate using Wordnet similarity by shared info content.
 *
 * @param text
 * @param word
 */
public void disambiguate(NLPText text) {
    Map<NLPText, Set<SenseRelationInfo>> senseRelations = new
    HashMap<NLPText, Set<SenseRelationInfo>>();

    Set<PairOfWords> pairOfWords = getPairsOfWords(text);
    log.debug(pairOfWords);
    for (PairOfWords tuple : pairOfWords) {
        Sense bestSense1 = null;
        Sense bestSense2 = null;
        if (!senseRelations.containsKey(tuple.getWord1())) {
            senseRelations.put(tuple.getWord1(), new
        HashSet<SenseRelationInfo>());
        }
        if (!senseRelations.containsKey(tuple.getWord2())) {
            senseRelations.put(tuple.getWord2(), new
        HashSet<SenseRelationInfo>());
        }

        // if both words already have senses skip to next pair
        if ((tuple.getWord1().getDictionaryWordSense() != null)
            && (tuple.getWord2().getDictionaryWordSense() != null)) {
            log.debug("pair " + tuple + " is already disambiguated");
            continue;
        }

        // find co-locations of the word senses in synset definitions
        log.debug("searching for synset definition colocations of " +
        tuple.getWord1()
            + " and " + tuple.getWord2());
        List<Map<NLPText, Sense>> colocations;
        colocations = findSynsetDefinitionColocations(tuple.getWord1(),
        tuple.getWord2());
        log.debug("colocations of " + tuple.getWord1() + " and " +
        tuple.getWord2() + ":" +
        colocations);
        if (!colocations.isEmpty()) {
            for (Map<NLPText, Sense> wordSenseMap : colocations) {
                if (wordSenseMap.containsKey(tuple.getWord1())
                    && wordSenseMap.containsKey(tuple.getWord2())) {
                    bestSense1 = wordSenseMap.get(tuple.getWord1());
                    bestSense2 = wordSenseMap.get(tuple.getWord2());

```

```

                    SenseRelationInfo sri = new ColocationSenseRelationInfo(
                        ColocationSource.WordNetDefinition, bestSense1, bestSense2,
                    "");
                    senseRelations.get(tuple.getWord1()).add(sri);
                    senseRelations.get(tuple.getWord2()).add(sri);
                }
            }
        }

        // find co-locations of the word senses in the semcor corpus
        // sentences
        log.debug("searching for Semcor Corpus sentence colocations of " +
        tuple.getWord1()
            + " and " + tuple.getWord2());
        colocations = findSemcorSentenceColocations(tuple.getWord1(),
        tuple.getWord2());
        log.debug("colocations of " + tuple.getWord1() + " and " +
        tuple.getWord2() + ":" +
        colocations);
        if (!colocations.isEmpty()) {
            for (Map<NLPText, Sense> wordSenseMap : colocations) {
                if (wordSenseMap.containsKey(tuple.getWord1())
                    && wordSenseMap.containsKey(tuple.getWord2())) {
                    bestSense1 = wordSenseMap.get(tuple.getWord1());
                    bestSense2 = wordSenseMap.get(tuple.getWord2());

                    SenseRelationInfo sri = new ColocationSenseRelationInfo(
                        ColocationSource.SemcorCorpus, bestSense1, bestSense2, "");
                    senseRelations.get(tuple.getWord1()).add(sri);
                    senseRelations.get(tuple.getWord2()).add(sri);
                }
            }
        }

        // similarity of word senses or their definitions
        for (Sense word1Sense :
            tuple.getWord1().getDictionaryWord().getSenses(
                tuple.getWord1().getPartOfSpeech())) {
            for (Sense word2Sense :
                tuple.getWord2().getDictionaryWord().getSenses(
                    tuple.getWord2().getPartOfSpeech())) {
                SenseRelationInfo sri;
                if (word1Sense.getSynset().getPartOfSpeech().equals(
                    word2Sense.getSynset().getPartOfSpeech())) {
                    sri = similarity(word1Sense, word2Sense);
                } else {

```

```

    sri = definitionSimilarity(word1Sense, word2Sense);
}
senseRelations.get(tuple.getWord1()).add(sri);
senseRelations.get(tuple.getWord2()).add(sri);

// relatedness
sri = relatedness(word1Sense, word2Sense);
senseRelations.get(tuple.getWord1()).add(sri);
senseRelations.get(tuple.getWord2()).add(sri);
}
}

if (log.isDebugEnabled()) {
for (NLPText word : text.getLeaves()) {
log.debug(word.getText() + " -> " + senseRelations.get(word));
}
}

// build a sense by sense matrix and assign a rank of the
// relationship between each sense.
int senseCount = 0;
Map<Sense, Integer> senseToIndexMap = new HashMap<Sense,
Integer>();
Map<Integer, Sense> indexToSenseMap = new HashMap<Integer,
Sense>();
for (NLPText word : text.getLeaves()) {
if (word.getDictionaryWord() != null) {
for (Sense sense : word.getDictionaryWord().getSenses()) {
senseToIndexMap.put(sense, senseCount);
indexToSenseMap.put(senseCount, sense);
senseCount++;
}
}
}
Counter senseRelationMatrix[][] = new Counter[senseCount]
[senseCount];
for (NLPText word : text.getLeaves()) {
Set<SenseRelationInfo> relations = senseRelations.get(word);
if (relations != null) {
for (SenseRelationInfo sri : relations) {
Sense sense1 = sri.getSense1();
Sense sense2 = sri.getSense2();
int index1 = senseToIndexMap.get(sense1);
int index2 = senseToIndexMap.get(sense2);
Counter rank = senseRelationMatrix[index1][index2];
if (rank == null) {
rank = new Counter();
}
}
}
}

```

```

    senseRelationMatrix[index1][index2] = rank;
}
rank.count += sri.getRank();
}
}
}

// sum up the ranks of each sense
Map<Sense, Counter> senseRankMap = new HashMap<Sense, Counter>();
for (int x = 0; x < senseCount; x++) {
for (int y = 0; y < senseCount; y++) {
if (x != y) {
Sense sense = indexToSenseMap.get(x);
Counter rank = senseRelationMatrix[x][y];
if (rank != null) {
Counter counter = senseRankMap.get(sense);
if (counter == null) {
counter = new Counter();
senseRankMap.put(sense, counter);
}
counter.count += rank.count;
}
}
}
}

// assign the sense of a word with the highest rank
for (NLPText word : text.getLeaves()) {
if (word.getDictionaryWord() != null) {
double maxRank = 0.0;
// choose the highest ranked in WordNet for each word.
Sense bestSense =
word.getDictionaryWord().getSense(word.getPartOfSpeech(), 1);
for (Sense sense : word.getDictionaryWord().getSenses()) {
Counter rankCounter = senseRankMap.get(sense);
if (rankCounter != null) {
double rank = senseRankMap.get(sense).count;
if (rank > maxRank) {
maxRank = rank;
bestSense = sense;
}
}
}
word.setDictionaryWordSense(bestSense);
word.setDictionaryWordSenseRelationInfo(senseRelations.get(word))
;
}
}

```

```

        }

    }

}

/***
 * Example:<br>
 * text -> users access the system.<br>
 * results:<br>
 * user, access<br>
 * user, system<br>
 * access, system<br>
 *
 * @param text
 * @return the distinct pairs of all the dictionary words in the text
not
 *           including pairs of the same word
 */
protected Set<PairOfWords> getPairsOfWords(NLPText text) {
    Set<PairOfWords> pairsOfWords = new HashSet<PairOfWords>();
    for (NLPText word1 : text.getLeaves()) {
        if (word1.getDictionaryWord() != null) {
            for (NLPText word2 : text.getLeaves()) {
                if ((word2.getDictionaryWord() != null)
                    && (!word1.getLemma().equals(word2.getLemma()) || !word1
                        .getPartOfSpeech().equals(word2.getPartOfSpeech())))
                {
                    pairsOfWords.add(new PairOfWords(word1, word2));
                }
            }
        }
    }
    return pairsOfWords;
}

/***
 * Given two words find the senses of the words that collocate in the
synset
 * defintions of WordNet.
 *
 * @param word1
 * @param word2
 * @return a list of the senses of the supplied words that collocate
mapped
 *           by the word. If the words don't collocate an empty list is
 *           returned. If one of the words is already assigned a sense
return
 *           only the senses of the other word that collocate with that
sense.
 */
public List<Map<NLPText, Sense>>
findSynsetDefinitionColocations(NLPText word1, NLPText word2) {
    List<Map<NLPText, Sense>> colocations = new ArrayList<Map<NLPText,
Sense>>();
    List<Synset> synsets;
    if (word1.getDictionaryWordSense() != null) {
        synsets =
dictionaryRepository.findSynsetsWithColocatedDefinitionSenseAndWord(wo
rd1
            .getDictionaryWordSense(), word2.getDictionaryWord());
    } else if (word2.getDictionaryWordSense() != null) {
        synsets =
dictionaryRepository.findSynsetsWithColocatedDefinitionSenseAndWord(wo
rd2
            .getDictionaryWordSense(), word1.getDictionaryWord());
    } else {
        synsets =
dictionaryRepository.findSynsetsWithColocatedDefinitionWords(word1
            .getDictionaryWord(), word2.getDictionaryWord());
    }
    for (Synset synset : synsets) {
        Set<Sense> word1Senses = new HashSet<Sense>();
        Set<Sense> word2Senses = new HashSet<Sense>();

        if (synset.getWords() != null) {
            for (SynsetDefinitionWord defWord : synset.getWords()) {
                if ((defWord != null) &&
defWord.getText().equals(word1.getLemma())
                    && synset.isPartOfSpeech(word1.getPartOfSpeech()))
                {
                    log.debug(word1.getLemma() + "#" + defWord.getSense().getRank()
+ " "
                        + defWord.getSynset().getDefinition());
                    word1Senses.add(defWord.getSense());
                }
                if ((defWord != null) &&
defWord.getText().equals(word2.getLemma())
                    && synset.isPartOfSpeech(word2.getPartOfSpeech()))
                {
                    log.debug(word2.getLemma() + "#" + defWord.getSense().getRank()
+ " "
                        + defWord.getSynset().getDefinition());
                    word2Senses.add(defWord.getSense());
                }
            }
        }
        // make pairs for each of the word1 and word2 senses
        for (Sense sense1 : word1Senses) {

```

```

        for (Sense sense2 : word2Senses) {
            Map<NLPText, Sense> senseMap = new HashMap<NLPText, Sense>();
            senseMap.put(word1, sense1);
            senseMap.put(word2, sense2);
            colocations.add(senseMap);
        }
    }

    return colocations;
}

/**
 * Given two words find the senses of the words that collocate in the
Semcor
 * Corpus sentences.
 *
 * @param word1
 * @param word2
 * @return a list of the senses of the supplied words that collocate
mapped
 *         by the word. If the words don't collocate an empty list is
 *         returned. If one of the words is already assigned a sense
return
 *         only the senses of the other word that collocate with that
sense.
 */
public List<Map<NLPText, Sense>>
findSemcorSentenceColocations(NLPText word1, NLPText word2) {
    List<Map<NLPText, Sense>> colocations = new ArrayList<Map<NLPText,
Sense>>();
    List<SemcorSentenceWord> sentenceWords;
    if (word1.getDictionaryWordSense() != null) {
        sentenceWords = dictionaryRepository
            .findSemcorSentencesWithColocatedDefinitionSenseAndWord(word1
                .getDictionaryWordSense(), word2.getDictionaryWord());
    } else if (word2.getDictionaryWordSense() != null) {
        sentenceWords = dictionaryRepository
            .findSemcorSentencesWithColocatedDefinitionSenseAndWord(word2
                .getDictionaryWordSense(), word1.getDictionaryWord());
    } else {
        sentenceWords =
dictionaryRepository.findSemcorSentencesWithColocatedDefinitionWords(
            word1.getDictionaryWord(), word2.getDictionaryWord());
    }
    Set<Sense> word1Senses = new HashSet<Sense>();
    Set<Sense> word2Senses = new HashSet<Sense>();
}

```

```

        for (SemcorSentenceWord sentenceWord : sentenceWords) {
            log.debug(sentenceWord);
            if
(sentenceWord.getSense().getWord().getLemma().equals(word1.getLemma()))
            ) {
                log.debug(word1.getLemma() + "#" +
sentenceWord.getSense().getRank() + " "
                    + sentenceWord.getSense().getSynset().getDefinition());
                word1Senses.add(sentenceWord.getSense());
            }
            if
(sentenceWord.getSense().getWord().getLemma().equals(word2.getLemma()))
            ) {
                log.debug(word2.getLemma() + "#" +
sentenceWord.getSense().getRank() + " "
                    + sentenceWord.getSense().getSynset().getDefinition());
                word2Senses.add(sentenceWord.getSense());
            }
        }

        // make pairs for each of the word1 and word2 senses
        for (Sense sense1 : word1Senses) {
            for (Sense sense2 : word2Senses) {
                Map<NLPText, Sense> senseMap = new HashMap<NLPText, Sense>();
                senseMap.put(word1, sense1);
                senseMap.put(word2, sense2);
                colocations.add(senseMap);
            }
        }
        return colocations;
    }

    /**
     * The relationship of two senses through derivation, pertainym or
     * entailment.
     *
     * @param sense1
     * @param sense2
     * @return
     */
    public SenseRelationInfo relatedness(Sense sense1, Sense sense2) {
        log.debug("sense1: " + sense1 + " " +
sense1.getSynset().getDefinition());
        log.debug("sense2: " + sense2 + " " +
sense2.getSynset().getDefinition());

        if (sense1.equals(sense2)) {

```

```

        return new SemanticSimilaritySenseRelationInfo(sense1, sense2,
1.0d, "identical.");
    }
    Linkdef linkDef;
    try {
        linkDef = dictionaryRepository.findLinkDef(40L);
        dictionaryRepository.findSemlinkref(sense1.getSynset(),
sense2.getSynset(), linkDef, 1);
        return new SemanticRelatednessSenseRelationInfo(sense1, sense2,
1.0d, linkDef.getName());
    } catch (NoSuchEntityException e) {
    }

    for (Long linkDefId : new Long[] { 71L, 80L, 81L }) {
        try {
            linkDef = dictionaryRepository.findLinkDef(linkDefId);
            dictionaryRepository.findLexlinkref(sense1, sense2, linkDef);
            return new SemanticRelatednessSenseRelationInfo(sense1, sense2,
1.0d, linkDef
                .getName());
        } catch (NoSuchEntityException e) {
        }
    }
    return new SemanticRelatednessSenseRelationInfo(sense1, sense2,
0.0d, "unrelated.");
}

/**
 * The similarity of the words in the definitions of the two word
senses and
 * the two senses themselves.
 *
 * @param sense1
 * @param sense2
 * @return
 */
public SenseRelationInfo definitionSimilarity(Sense sense1, Sense
sense2) {
    log.debug("sense1: " + sense1 + " " +
sense1.getSynset().getDefinition());
    log.debug("sense2: " + sense2 + " " +
sense2.getSynset().getDefinition());

    if (sense1.equals(sense2)) {

```

```

        return new SemanticSimilaritySenseRelationInfo(sense1, sense2,
1.0d, "identical.");
    }

    int sense1Dim = sense1.getSynset().getWords().size() + 1;
    int sense2Dim = sense2.getSynset().getWords().size() + 1;
    SenseRelationInfo senseRelations[][] = new
SenseRelationInfo[sense1Dim][sense2Dim];
    for (SynsetDefinitionWord sense1defWord :
sense1.getSynset().getWords()) {
        Sense sense1defWordSense = sense1defWord.getSense();
        log.debug("sense1defWord = " + sense1defWord);
        if ((sense1defWordSense != null)
            &&
sense1defWordSense.getSynset().getPartOfSpeech().in(PartOfSpeech.NOUN,
PartOfSpeech.VERB)) {
            // compare the sense 1 definition word sense to sense 2
            if (sense1defWordSense.getSynset().getPartOfSpeech().equals(
sense2.getSynset().getPartOfSpeech())) {
                SenseRelationInfo similarity = similarity(sense1defWordSense,
sense2);
                log.debug(" sense1 def word: " + sense1defWordSense + " sense2:
" + sense2
                    + " similarity: " + similarity.getRank());
                senseRelations[sense1.getSynset().getWords().indexOf(sense1defWor
dSense) + 1][0] = similarity;
            }
            for (SynsetDefinitionWord sense2defWord :
sense2.getSynset().getWords()) {
                Sense sense2defWordSense = sense2defWord.getSense();
                log.debug(" sense2defWord = " + sense2defWord);
                if (sense2defWordSense != null) {
                    // compare the sense 2 definition word senses to
                    // sense 1
                    if (sense2defWordSense.getSynset().getPartOfSpeech().equals(
sense1.getSynset().getPartOfSpeech())) {
                        SenseRelationInfo similarity = similarity(sense1,
sense2defWordSense);
                        log.debug(" sense1: " + sense1 + " sense2 def word: "
+ sense2defWordSense + " similarity: " +
similarity.getRank());
                        senseRelations[0][sense2.getSynset().getWords().indexOf(
                            sense1defWordSense) + 1] = similarity;
                    }
                    // compare the sense 2 definition word sense to the
                    // sense 1 definition word sense
                    if (sense1defWordSense.getSynset().getPartOfSpeech().equals(

```

```

        sense2defWordSense.getSynset().getPartOfSpeech())) {
SenseRelationInfo similarity = similarity(sense1defWordSense,
    sense2defWordSense);
log.debug("  sensel def word: " + sense1defWordSense
    + " sense2 def word: " + sense2defWordSense + " similarity:
"
    + similarity.getRank());
senseRelations[sense1.getSynset().getWords()
    .indexOf(sense1defWordSense) + 1]
[sense2.getSynset().getWords()
    .indexOf(sense1defWordSense) + 1] = similarity;
}
}
}
}
double similaritySum = 0.0d;
StringBuilder sb = new StringBuilder();
for (int i = 0; i < sense1Dim; i++) {
    for (int j = 0; j < sense2Dim; j++) {
        if (senseRelations[i][j] != null) {
            similaritySum += senseRelations[i][j].getRank();
            sb.append(senseRelations[i][j].getReason());
            sb.append("\n");
        }
    }
}
return new SemanticRelatednessSenseRelationInfo(sense1, sense2,
similaritySum
    / (sense1Dim + sense2Dim), sb.toString());
}

/**
 * Given two synsets of like part of speech, calculate the similarity
of the
 * words using the information content measure of Seco, Veale and
Hayes and
 * the similarity measure defined by Lin, which is a factor of the
 * commonality of the two terms over their combined specificity.<br>
 * NOTE: in the cases where two synsets have multiple common
ancestors, the
 * highest similarity is returned.
 *
 * @param sensel
 * @param sense2
 * @param linkType
 * @return

```

```

 */
public SenseRelationInfo similarity(Sense sensel, Sense sense2) {
    log.debug("sensel: " + sensel + " " +
sensel.getSynset().getDefinition());
    log.debug("sense2: " + sense2 + " " +
sense2.getSynset().getDefinition());
    Linkdef linkType = dictionaryRepository.findLinkDef(1L);
    Synset synset1 = sensel.getSynset();
    Synset synset2 = sense2.getSynset();

    // TODO: link types available, for example Pieces and Cancer are
    // related as zodiac signs, but use linkType of 4 (instance of) and
not
    // hyponym.
    if (synset1.equals(synset2)) {
        return new SemanticSimilaritySenseRelationInfo(sensel, sense2,
1.0d, "identical.");
    }

    SenseRelationInfo sri = new
SemanticSimilaritySenseRelationInfo(sensel, sense2, 0.0d,
    "no similarity.");
    for (Synset lcs :
dictionaryRepository.getLowestCommonHypernyms(synset1, synset2)) {
        // Similarity using Lin's formula
        double ic1 = dictionaryRepository.infoContent(synset1, linkType);
        double ic2 = dictionaryRepository.infoContent(synset2, linkType);
        double icLCS = dictionaryRepository.infoContent(lcs, linkType);
        double similarity = (2.0d * icLCS) / (ic1 + ic2);
        // Jiang and Conrath 1997 - similarity
        // (2.0d * icLCS) - (ic1 + ic2)
        if (similarity > sri.getRank()) {
            sri = new SemanticSimilaritySenseRelationInfo(sensel, sense2,
similarity, sensel
                + " ic: " + ic1 + " " + sense2 + " ic: " + ic2 + " lcs: " + lcs
+ " ic: "
                + icLCS);
        }
    }
    if (log.isDebugEnabled()) {
        log.debug("synset1: " + synset1 + " synset2: " + synset2 + "
maxSimilarity: "
            + sri.getRank());
    }
    return sri;
}

```

```

protected class PairOfWords {
    private final NLPText word1;
    private final NLPText word2;

    protected PairOfWords(NLPText word1, NLPText word2) {
        // put the words in order of spelling
        if (word1.getLemma().compareTo(word2.getLemma()) < 0) {
            this.word1 = word1;
            this.word2 = word2;
        } else {
            this.word1 = word2;
            this.word2 = word1;
        }
    }

    protected NLPText getWord1() {
        return word1;
    }

    protected NLPText getWord2() {
        return word2;
    }

    @Override
    public int hashCode() {
        return word1.hashCode() + word2.hashCode();
    }

    @Override
    public boolean equals(Object o) {
        if (o instanceof PairOfWords) {
            PairOfWords other = (PairOfWords) o;
            if (((word1.getText()).equals(other.word1.getText()) &&
word1.getPartOfSpeech()
                .equals(other.word1.getPartOfSpeech())))
                && ((word2.getText()).equals(other.word2.getText()) && word2
                    .getPartOfSpeech().equals(other.word2.getPartOfSpeech())))
            {
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return "(" + getWord1() + ", " + getWord2() + ")";
    }
}

```

```

    }

    private class Counter {
        double count = 0.0;
    }
}

```

workflowdisposition.java

```

/*
 * $Id: WorkflowDisposition.java,v 1.2 2008/02/27 11:34:41 rregan Exp
 *
 * Copyright (c) 2008 Ron Regan Jr. All Rights Reserved.
 */
package edu.harvard.fas.rregan.uiframework.navigation;

/**
 * WorkflowDisposition indicates the relationship between panels for a
 * set of
 * work. It is used by events and panel management
 *
 * @author ron
 */
public enum WorkflowDisposition {

    /**
     * If the disposition of an event is NewFlow then the PanelManager
     * should
     * cleanup the state of the previous workflow checking for unsaved
     * data and
     * allowing the user to save or throw away any changes.
     */
    NewFlow(),

    /**
     * If the disposition of an event is ContinueFlow, it indicates to
     * the
     * PanelManager that the state of the last window should be
     * preserved while
     * the user does additional work in the new panel. When the user is
     * finished
     * working in the new window, the previous window will be displayed.
     */
    ContinueFlow();
}

```

```

private WorkflowDisposition() {
}
}

```

XSLT Code

project2html.xslt

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
$Id: project2html.xslt,v 1.9 2009/03/06 02:06:45 rregan Exp $
An XSLT for rendering a Requel Project xml file as an html document.
-->
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/1999/xhtml"
  xmlns:rp="http://www.people.fas.harvard.edu/~rregan/requel"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xi="http://www.w3.org/2001/XInclude" exclude-result-
prefixes="xs xsi rp xsl xi">

<xsl:output method="xml" encoding="UTF-8" indent="yes" />

<xsl:template match="/rp:project">
<html>
  <head>
    <title>
      <xsl:value-of select="rp:name" />
    </title>
    <xsl:call-template name="css"/>
  </head>
  <body>
    <div class="project">
      <h1>Requirements for &quot;<xsl:value-of select="rp:name" />&quot;</h1>
      <h5>Client: <xsl:value-of select="rp:organization/@name" /></h5>
      <h5>Revision: <xsl:value-of select="@revision" /></h5>
      <h4>Table of Contents</h4>
      <div class="toc">
        <ul>
          <xsl:if test="count(rp:stakeholders/rp:stakeholder) > 0">
            <li><a href="#stakeholders">Stakeholders</a>
          </ol>

```

```

          <xsl:apply-templates select="rp:stakeholders/rp:stakeholder"
mode="summary"/>
        </ol>
      </li>
    </xsl:if>
    <xsl:if test="count(rp:teams/rp:team) > 0">
      <li><a href="#teams">Teams</a>
      <ol>
        <xsl:apply-templates select="rp:teams/rp:team" mode="summary"/>
      </ol>
    </li>
  </xsl:if>
  <xsl:if test="count(rp:actors/rp:actor) > 0">
    <li><a href="#actors">Actors</a>
    <ol>
      <xsl:apply-templates select="rp:actors/rp:actor" mode="summary"/>
    </ol>
  </li>
  </xsl:if>
  <xsl:if test="count(rp:goals/rp:goal) > 0">
    <li><a href="#goals">Goals</a>
    <ol>
      <xsl:apply-templates select="rp:goals/rp:goal" mode="summary"/>
    </ol>
  </li>
  </xsl:if>
  <xsl:if test="count(rp:stories/rp:story) > 0">
    <li><a href="#stories">Stories</a>
    <ol>
      <xsl:apply-templates select="rp:stories/rp:story" mode="summary"/>
    </ol>
  </li>
  </xsl:if>
  <xsl:if test="count(rp:usecases/rp:usecase) > 0">
    <li><a href="#usecases">Use Cases</a>
    <ol>
      <xsl:apply-templates select="rp:usecases/rp:usecase"
mode="summary"/>
    </ol>
  </li>
  </xsl:if>
  <xsl:if test="count(rp:scenarios/rp:scenario) > 0">
    <li><a href="#scenarios">Scenarios</a>
    <ol>
      <xsl:apply-templates select="rp:scenarios/rp:scenario"
mode="summary"/>
    </ol>
  </li>

```

```

</ol>
</li>
</xsl:if>
<xsl:if test="count(rp:glossary/rp:term) > 0">
<li><a href="#glossary">Glossary</a>
<ol>
  <xsl:apply-templates select="rp:glossary/rp:term" mode="summary"/>
</ol>
</li>
</xsl:if>
</ul>
</div>
<div class="body">
<xsl:apply-templates select="rp:stakeholders" />
<xsl:apply-templates select="rp:teams" />
<xsl:apply-templates select="rp:actors" />
<xsl:apply-templates select="rp:goals" />
<xsl:apply-templates select="rp:stories" />
<xsl:apply-templates select="rp:usecases" />
<xsl:apply-templates select="rp:scenarios" />
<xsl:apply-templates select="rp:glossary" />
</div>
</div>
</body>
</html>
</xsl:template>

<xsl:template match="rp:stakeholder[count(rp:user/rp:username) = 0] | rp:team | rp:actor | rp:goal | rp:story | rp:usecase | rp:scenario | rp:term" mode="summary">
<li>
  <a href="#{generate-id(.)}">
    <xsl:value-of select="rp:name" />
  </a>
</li>
</xsl:template>
<xsl:template match="rp:stakeholder[count(rp:user/rp:username) > 0]" mode="summary">
<li>
  <a href="#{generate-id(.)}">
    <xsl:value-of select="rp:user/rp:name" /> [<xsl:value-of select="rp:user/rp:username" />]
  </a>
</li>
</xsl:template>

```

```

  <!-- Stakeholders -->
<xsl:template match="rp:stakeholders">
<xsl:if test="count(rp:stakeholder) > 0">
<div class="stakeholders">
  <a name="stakeholders"></a>
  <h2>Stakeholders</h2>
  <table class="stakeholders">
    <thead>
      <tr>
        <th>User?</th>
        <th>Name</th>
        <th>Organization</th>
        <th>Email</th>
        <th>Phone</th>
      </tr>
    </thead>
    <tbody>
      <xsl:apply-templates select="rp:stakeholder" />
    </tbody>
  </table>
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:stakeholder[count(rp:user/rp:username) > 0]">
<tr>
<td>
  Yes
</td>
<td>
  <a name="{generate-id(.)}">
    <xsl:value-of select="rp:user/rp:name" />
    (<xsl:value-of select="rp:user/rp:username" />)
  </a>
</td>
<td>
  <xsl:apply-templates select="rp:user/rp:organization/@name" />
</td>
<td>
  <xsl:apply-templates select="rp:user/rp:emailAddress" />
</td>
<td>
  <xsl:apply-templates select="rp:user/rp:phoneNumber" />
</td>
</tr>
<xsl:if test="count(rp:goals/*) > 0">
<tr>

```

```

<td></td>
<td colspan="4">
<xsl:apply-templates select="rp:goals" />
</td>
</tr>
</xsl:if>
<xsl:if test="count(rp:annotations/*) > 0">
<tr>
<td></td>
<td colspan="4">
<xsl:apply-templates select="rp:annotations" />
</td>
</tr>
</xsl:if>
</xsl:template>

<xsl:template match="rp:stakeholder">
<tr>
<td>
No
</td>
<td>
<a name="{generate-id(.)}">
<xsl:value-of select="rp:name" />
</a>
</td>
<td>
<xsl:apply-templates select="rp:organization/@name" />
</td>
<td>
</td>
<td>
</td>
</tr>
<xsl:if test="count(rp:goals/*) > 0">
<tr>
<td></td>
<td colspan="4">
<xsl:apply-templates select="rp:goals" />
</td>
</tr>
</xsl:if>
<xsl:if test="count(rp:annotations/*) > 0">
<tr>
<td></td>
<td colspan="4">
<xsl:apply-templates select="rp:annotations" />
</td>
</tr>
</xsl:if>
</xsl:template>
</td>
</tr>
</xsl:if>
</xsl:template>

<xsl:template match="rp:stakeholderRef">
<xsl:variable name="ref-id" select=". " />
<li class="stakeholder">
<span class="body">
<a href="#{generate-id(/rp:stakeholder[@id = $ref-id])}">
<xsl:apply-templates select="//rp:stakeholder[@id = $ref-
id]/rp:name" />
</a>
</span>
</li>
</xsl:template>

<!-- Teams -->
<xsl:template match="rp:teams">
<xsl:if test="count(rp:team) > 0">
<a name="teams"></a>
<h2>Teams</h2>
<div class="body">
<xsl:apply-templates select="*"/>
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:team">
<h2><xsl:value-of select="rp:name"/></h2>
<div class="body">
<xsl:apply-templates select="rp:members"/>
<xsl:apply-templates select="rp:annotations" />
</div>
</xsl:template>

<!-- Actors -->
<xsl:template match="rp:project/rp:actors">
<xsl:if test="count(rp:actor) > 0">
<a name="actors"></a>
<h2>Actors</h2>
<div class="body">
<xsl:apply-templates select="*"/>
</div>
</xsl:if>
</xsl:template>

```

```

<xsl:template match="rp:actors">
  <xsl:if test="count(*) > 0">
    <h4>Actors</h4>
    <div class="body">
      <xsl:apply-templates select="*" />
    </div>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:actor">
  <div class="actor">
    <h4>
      <a name="{generate-id(.)}">
        <xsl:value-of select="rp:name" />
      </a>
    </h4>
    <div class="body">
      <xsl:apply-templates select="rp:text" />
      <xsl:apply-templates select="rp:goals" />
      <xsl:apply-templates select="rp:annotations" />
    </div>
  </div>
</xsl:template>

<xsl:template match="rp:actorRef">
  <xsl:variable name="ref-id" select="." />
  <li class="actor">
    <span class="body">
      <a href="#{generate-id(/rp:actor[@id = $ref-id])}">
        <xsl:apply-templates select="/rp:actor[@id = $ref-id]/rp:name" />
      </a>
    </span>
  </li>
</xsl:template>

<xsl:template match="rp:primaryActorRef">
  <xsl:variable name="ref-id" select="." />
  <a href="#{generate-id(/rp:actor[@id = $ref-id])}">
    <xsl:apply-templates select="/rp:actor[@id = $ref-id]/rp:name" />
  </a>
</xsl:template>

<!-- Goals -->
<xsl:template match="rp:project/rp:goals">
  <xsl:if test="count(rp:goal) > 0">
    <div class="goals">
      <a name="goals"></a>

```

```

      <h2>Goals</h2>
      <ul class="goals">
        <xsl:apply-templates select="*" />
      </ul>
    </div>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:goals">
  <xsl:if test="count(*) > 0">
    <h4>Goals</h4>
    <ul class="goals">
      <xsl:apply-templates select="*" />
    </ul>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:goal">
  <li class="goal">
    <div class="goal">
      <h4>
        <a name="{generate-id(.)}">
          <xsl:value-of select="rp:name" />
        </a>
      </h4>
      <div class="body">
        <xsl:value-of select="rp:text" />
      </div>
      <xsl:apply-templates select="rp:goalRelations" />
      <xsl:apply-templates select="rp:annotations" />
    </div>
  </li>
</xsl:template>

<xsl:template match="rp:goalRef">
  <xsl:variable name="ref-id" select="." />
  <li class="goal">
    <span class="body">
      <a href="#{generate-id(/rp:goal[@id = $ref-id])}">
        <xsl:apply-templates select="/rp:goal[@id = $ref-id]/rp:name" />
      </a>
    </span>
  </li>
</xsl:template>

```

```

<xsl:template match="rp:goalRelations">
  <xsl:if test="count(rp:goalRelation) > 0">
    <h4>Goal Relations</h4>
    <ul>
      <xsl:apply-templates select="rp:goalRelation" />
    </ul>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:goalRelation">
  <xsl:variable name="ref-id" select="@toGoal" />
  <li class="goal">
    <div>
      <xsl:value-of select="@relationType" /> -
      <a href="#{generate-id(//rp:goal[@id = $ref-id])}">
        <xsl:apply-templates select="//rp:goal[@id = $ref-id]/rp:name" />
      </a>
    </div>
    <xsl:apply-templates select="rp:annotations" />
  </li>
</xsl:template>


<xsl:template match="rp:project/rp:stories">
  <xsl:if test="count(rp:story) > 0">
    <a name="stories"></a>
    <h2>Stories</h2>
    <div class="body">
      <xsl:apply-templates select="*" />
    </div>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:stories">
  <xsl:if test="count(*) > 0">
    <h4>Stories</h4>
    <xsl:choose>
      <xsl:when test="count(rp:story) > 0">
        <div class="body">
          <xsl:apply-templates select="rp:story" />
        </div>
      </xsl:when>
      <xsl:when test="count(rp:storyRef) > 0">
        <div class="body">
          <ul>
            <xsl:for-each select="rp:storyRef">
              <li><xsl:apply-templates select="." /></li>
            </xsl:for-each>
          </ul>
        </div>
      </xsl:when>
    </xsl:choose>
  </xsl:if>
</xsl:template>

</xsl:for-each>
</ul>
</div>
</xsl:when>
</xsl:choose>
</xsl:if>
</xsl:template>

<xsl:template match="rp:story">
  <div class="story">
    <h4>
      <a name="{generate-id(.)}">
        <xsl:value-of select="@storyType" />
      Story -
      <xsl:value-of select="rp:name" />
    </a>
  </h4>
  <div class="body">
    <xsl:apply-templates select="rp:text" />
  </div>
  <xsl:apply-templates select="rp:actors" />
  <xsl:apply-templates select="rp:goals" />
  <xsl:apply-templates select="rp:annotations" />
</div>
</xsl:template>

<xsl:template match="rp:storyRef">
  <xsl:variable name="ref-id" select="." />
  <a href="#{generate-id(//rp:story[@id = $ref-id])}">
    <xsl:apply-templates select="//rp:story[@id = $ref-id]/rp:name" />
  </a>
</xsl:template>


<xsl:template match="rp:project/rp:usecases">
  <xsl:if test="count(rp:usecase) > 0">
    <a name="usecases"></a>
    <h2>Use Cases</h2>
    <div class="body">
      <xsl:apply-templates select="*" />
    </div>
  </xsl:if>
</xsl:template>

<xsl:template match="rp:usecases">
  <xsl:if test="count(*) > 0">
    <h4>Use Cases</h4>

```

```

<div class="body">
  <xsl:apply-templates select="rp:usecase" />
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:usecaseRef">
  <xsl:variable name="ref-id" select="." />
  <a href="#{generate-id(/rp:usecase[@id = $ref-id])}">
    <xsl:apply-templates select="/rp:usecase[@id = $ref-id]/rp:name" />
  </a>
</xsl:template>

<xsl:template match="rp:usecase">
  <div class="usecase">
    <h4>
      <a name="{generate-id(.)}">
        <xsl:value-of select="rp:name" />
      </a>
    </h4>
    <div class="body">
      <xsl:if test="string-length(rp:text) > 0">
        <h4>Description</h4>
        <xsl:apply-templates select="rp:text" />
      </xsl:if>
      <xsl:apply-templates select="rp:goals" />
      <xsl:apply-templates select="rp:stories" />
      <h4>Primary Actor: <xsl:apply-templates
select="rp:primaryActorRef" /></h4>
      <xsl:apply-templates select="rp:actors" />
      <h4>Scenario</h4>
      <xsl:variable name="scenarioRef" select="string(rp:scenarioRef)" />
      <xsl:apply-templates select="/rp:scenario[@id = $scenarioRef]" mode="usecase" />
      <xsl:apply-templates select="rp:annotations" />
    </div>
  </div>
</xsl:template>

<!-- Scenarios - only include scenarios that aren't part of another
scenario or use case --&gt;
&lt;xsl:template match="rp:project/rp:scenarios"&gt;
  &lt;xsl:variable name="rootScenariosExist"&gt;
    &lt;xsl:for-each select="rp:scenario"&gt;
      &lt;xsl:variable name="scenarioId" select="@id"/&gt;
      &lt;xsl:choose&gt;
        &lt;xsl:when test="count(/rp:scenarioRef[string(text()) =
$scenarioId]) = 0 and count(/rp:stepRef[string(text()) =
$scenarioId]) = 0"&gt;&lt;xsl:value-of select="concat(@id, '"
)" /&gt;&lt;/xsl:when&gt;
        &lt;/xsl:choose&gt;
        &lt;/xsl:for-each&gt;
        &lt;xsl:variable&gt;
          &lt;xsl:comment&gt; &lt;xsl:value-of select="$rootScenariosExist"/&gt;
        &lt;/xsl:comment&gt;
        &lt;xsl:if test="string-length($rootScenariosExist) &gt; 0"&gt;
          &lt;a name="scenarios"&gt;&lt;/a&gt;
          &lt;h2&gt;Scenarios (not used else where)&lt;/h2&gt;
          &lt;div class="body"&gt;
            &lt;xsl:for-each select="rp:scenario"&gt;
              &lt;xsl:variable name="scenarioId" select="@id"/&gt;
              &lt;xsl:if test="count(/rp:scenarioRef[string(text()) =
$scenarioId]) = 0 and count(/rp:stepRef[string(text()) =
$scenarioId]) = 0"&gt;
                &lt;xsl:apply-templates select="." /&gt;
              &lt;/xsl:if&gt;
            &lt;/xsl:for-each&gt;
          &lt;/div&gt;
        &lt;/xsl:if&gt;
      &lt;/xsl:template&gt;

      &lt;xsl:template match="rp:scenarios"&gt;
        &lt;xsl:if test="count(*) &gt; 0"&gt;
          &lt;h4&gt;Scenarios&lt;/h4&gt;
          &lt;div class="body"&gt;
            &lt;xsl:apply-templates select="rp:scenario" /&gt;
          &lt;/div&gt;
        &lt;/xsl:if&gt;
      &lt;/xsl:template&gt;

      &lt;xsl:template match="rp:scenario" mode="usecase"&gt;
        &lt;div class="scenario"&gt;
          &lt;xsl:apply-templates select="rp:steps" /&gt;
        &lt;/div&gt;
      &lt;/xsl:template&gt;

      &lt;xsl:template match="rp:scenario"&gt;
        &lt;div class="scenario"&gt;
          &lt;h4&gt;
            &lt;a name="{generate-id(.)}"&gt;
              &lt;xsl:value-of select="@scenarioType" /&gt;
              -
              &lt;xsl:value-of select="rp:name" /&gt;
            &lt;/a&gt;
          &lt;/h4&gt;
        &lt;/div&gt;
      &lt;/xsl:template&gt;
    </pre>

```

```

</h4>
<xsl:if test="string-length(rp:text) > 0">
  <h4>Description</h4>
  <xsl:apply-templates select="rp:text" />
</xsl:if>
<xsl:apply-templates select="rp:steps" />
<xsl:apply-templates select="rp:annotations" />
</div>
</xsl:template>

<xsl:template match="rp:steps">
  <ol class="scenario">
    <xsl:for-each select="rp:stepRef">
      <li class="step"><xsl:apply-templates select="."/></li>
    </xsl:for-each>
  </ol>
</xsl:template>

<xsl:template match="rp:stepRef">
  <xsl:variable name="stepRefId" select="text()" />
  <xsl:apply-templates select="//rp:scenario[@id = $stepRefId]//rp:step[@id = $stepRefId]" mode="steps"/>
</xsl:template>

<!-- Steps and embedded scenarios -->
<xsl:template match="rp:scenario" mode="steps">
  <xsl:value-of select="@scenarioType" /> - <xsl:value-of
select="rp:name" />
  <xsl:apply-templates select="rp:steps" />
</xsl:template>
<xsl:template match="rp:step" mode="steps">
  <xsl:value-of select="@scenarioType" /> - <xsl:value-of
select="rp:name" />
</xsl:template>

<!-- Glossary -->
<xsl:template match="rp:project/rp:glossary">
  <xsl:if test="count(rp:term) > 0">
    <a name="glossary"></a>
    <h2>Glossary</h2>
    <div class="body">
      <table class="stakeholders">
        <thead>
          <tr>
            <th>Term</th>
            <th>Definition</th>
          </tr>
        </thead>
        <tbody>
          <xsl:apply-templates select="rp:term" />
        </tbody>
      </table>
    </div>
  </xsl:if>
</xsl:template>

</thead>
<tbody>
  <xsl:apply-templates select="rp:term" />
</tbody>
</table>
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:term">
  <tr>
    <td><xsl:value-of select="rp:name" /></td>
    <td><xsl:value-of select="rp:text" /></td>
  </tr>
</xsl:template>

<!-- Annotations -->
<xsl:template match="rp:annotations">
<!--
<xsl:if test="count(*) > 0">
  <h4>Annotations</h4>
  <ul>
    <xsl:apply-templates select="rp:note|rp:issue|rp:lexicalIssue" />
  </ul>
</xsl:if>
-->
</xsl:template>

<xsl:template match="rp:note">
  <li class="note">
    <xsl:apply-templates select="rp:text" />
  </li>
</xsl:template>

<xsl:template match="rp:issue|rp:lexicalIssue">
  <li class="issue">
    <xsl:apply-templates select="rp:text" />
    <xsl:apply-templates select="rp:positions" />
  </li>
</xsl:template>

<xsl:template match="rp:positions">
  <xsl:if test="count(*) > 0">
    <h4>Solutions</h4>
    <div class="body">
      <ul>
        <xsl:apply-templates select="*" />
      </ul>
    </div>
  </xsl:if>
</xsl:template>

```

```

</ul>
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:position|rp:addWordToDictionaryPosition|
rp:changeSpellingPosition|rp:addGlossaryTermPosition|
rp:addActorPosition">
<li class="position">
<xsl:apply-templates select="rp:text" />
<xsl:apply-templates select="rp:arguments" />
</li>
</xsl:template>

<xsl:template match="rp:arguments">
<xsl:if test="count(*) > 0">
<h4>Arguments</h4>
<div class="body">
<h5>Strongly For</h5>
<xsl:apply-templates select="rp:argument[@supportLevel =
'StronglyFor']" />
<h5>For</h5>
<xsl:apply-templates select="rp:argument[@supportLevel = 'For']" />
<h5>Neutral</h5>
<xsl:apply-templates select="rp:argument[@supportLevel = 'Neutral']"
/>
<h5>Against</h5>
<xsl:apply-templates select="rp:argument[@supportLevel = 'Against']"
/>
<h5>Strongly Against</h5>
<xsl:apply-templates select="rp:argument[@supportLevel =
'StronglyAgainst']" />
</div>
</xsl:if>
</xsl:template>

<xsl:template match="rp:argument">
<div class="argument">
<xsl:apply-templates select="rp:text" />
</div>
</xsl:template>

<!-- CSS style -->
<xsl:template name="css">
<style type="text/css">
html {

```

```

height: 100%;
}

body {
height: 100%;
font-family: Arial;
font-size: 10pt;
margin-top: .5in;
margin-left: .5in;
margin-right: .5in;
margin-bottom: .5in;
}

h1 {
text-align: center;
font-family: Arial;
font-size: 14pt;
}

h2 {
text-align: left;
font-family: Arial;
font-size: 12pt;
}

h3, h4, h5, h6 {
text-align: left;
font-family: Arial;
font-size: 11pt;
}

table {
border: solid 1pt black;
border-collapse: collapse;
empty-cells: show;
}

td {
padding: 5pt;
border: solid 1pt black;
border-collapse: collapse;
font-family: Arial;
font-size: 10pt;
vertical-align: top;
text-align: left;

```

```

font-weight: normal;
}

div .project .body {
margin: 0;
}

div .body {
margin-top: .25in;
margin-left: .25in;
margin-right: .25in;
margin-bottom: .25in;
}

.stakeholders table {
margin-top: .25in;
margin-left: .25in;
margin-right: .25in;
margin-bottom: .25in;
}

th {
padding: 4pt;
border: solid 1pt black;
border-collapse: collapse;
font-family: Arial;
font-size: 10pt;
vertical-align: bottom;
text-align: left;
font-weight: bold;
}

.issue {
}

.note {
}

.p {
padding-bottom: 10pt;
}

```

</style>

</xsl:template>

</xsl:stylesheet>

XML

uiMainConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
 - Spring config for Main UI Panels. This collects panels defined in
the other
 - config files and assigns them to the navigation
-->
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:util="http://www.springframework.org/schema/util"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util-2.5.xsd
    ">

<bean id="mainNavigationPanelManager" scope="prototype"
class="edu.harvard.fas.rregan.uiframework.panel.DefaultPanelManager">
<constructor-arg><ref bean="eventDispatcher"/></constructor-arg>
<constructor-arg>
<util:set scope="prototype" value-
type="edu.harvard.fas.rregan.uiframework.panel.Panel">
<ref bean="userAdminNavigatorPanel"/>
<ref bean="projectNavigatorPanel"/>
<!--
<ref bean="nlpNavigatorPanel"/>
-->
</util:set>
</constructor-arg>
</bean>

<bean id="mainNavigationPanelContainer" scope="prototype"
class="edu.harvard.fas.rregan.requel.ui.MainScreenTabbedNavigation">

```

```

<constructor-arg><ref
local="mainNavigationPanelManager"/></constructor-arg>
</bean>

<bean id="mainContentPanelManager" scope="prototype"
class="edu.harvard.fas.rregan.uiframework.panel.DefaultPanelManager">
<constructor-arg><ref bean="eventDispatcher"/></constructor-arg>
<constructor-arg>
<util:set scope="prototype" value-
type="edu.harvard.fas.rregan.uiframework.panel.PanelDescriptor">
  <!-- user panels -->
  <ref bean="userEditorPanelFactory"/>

  <!-- annotation panels -->
  <ref bean="noteEditorPanelFactory"/>
  <ref bean="issueEditorPanelFactory"/>
  <ref bean="positionEditorPanelFactory"/>
  <ref bean="argumentEditorPanelFactory"/>

  <!-- project panels -->
  <ref bean="projectOverviewPanelFactory"/>
  <ref bean="projectOpenIssuesPanelFactory"/>
  <ref bean="projectStakeholdersNavigatorPanelFactory"/>
  <ref bean="stakeholderEditorPanelFactory"/>

  <ref bean="projectGoalsNavigatorPanelFactory"/>
  <ref bean="projectGoalsSelectorPanelFactory"/>
  <ref bean="goalEditorPanelFactory"/>
  <ref bean="goalRelationEditorPanelFactory"/>

  <ref bean="storyEditorPanelFactory"/>
  <ref bean="projectStoriesNavigatorPanelFactory"/>
  <ref bean="projectStoriesSelectorPanelFactory"/>

  <ref bean="actorEditorPanelFactory"/>
  <ref bean="projectActorsNavigatorPanelFactory"/>
  <ref bean="projectActorsSelectorPanelFactory"/>

  <ref bean="useCaseEditorPanelFactory"/>
  <ref bean="projectUseCasesNavigatorPanelFactory"/>
  <ref bean="projectUseCasesSelectorPanelFactory"/>

  <ref bean="scenarioEditorPanelFactory"/>
  <ref bean="projectScenariosNavigatorPanelFactory"/>
  <ref bean="projectScenarioSelectorPanelFactory"/>

  <ref bean="projectGlossaryTermsNavigatorPanelFactory"/>

```

```

<ref bean="projectGlossaryTermsSelectorPanelFactory"/>
<ref bean="glossaryTermEditorPanelFactory"/>

<ref bean="projectReportsNavigatorPanelFactory"/>
<ref bean="reportEditorPanelFactory"/>

<!-- NLP panels
<ref bean="parserPanelFactory" />
-->
</util:set>
</constructor-arg>
</bean>

<bean id="mainContentPanelContainer" scope="prototype"
class="edu.harvard.fas.rregan.uiframework.panel.TabbedPanelContainer">
<constructor-arg><ref
local="mainContentPanelManager"/></constructor-arg>
</bean>
</beans>

```

commandHandlerConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
 - Spring config for navigation setup.
-->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context-2.5.xsd
           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
           http://www.springframework.org/schema/util
           http://www.springframework.org/schema/util/spring-util-2.5.xsd
           "
       >
<!-- create these nested so that only one CommandHandler class is
exposed -->

```

```

<bean id="commandHandler"
      class="edu.harvard.fas.rregan.command.RetryOnLockFailuresCommandHandler"
      scope="singleton">
    <constructor-arg index="0">
      <bean
        class="edu.harvard.fas.rregan.command.ExceptionMappingCommandHandler"
        scope="singleton">
        <constructor-arg index="0">
          <bean
            class="edu.harvard.fas.rregan.repository.jpa.ExceptionMapper"/>
        </constructor-arg>
        <constructor-arg index="1">
          <bean
            class="edu.harvard.fas.rregan.requel.command.AnalysisInvokingCommandHandler"
            scope="singleton">
            <constructor-arg index="0">
              <bean class="edu.harvard.fas.rregan.command.DefaultCommandHandler"
                scope="singleton"/>
              </constructor-arg>
            </bean>
          </constructor-arg>
        </bean>
      </constructor-arg>
      <constructor-arg index="1">
        <ref bean="projectRepository"/>
      </constructor-arg>
    </bean>
  </beans>

```

```

http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">
  <aop:aspectj-autoproxy/>

  <bean id="proxyFactoryBean"
        class="org.springframework.aop.framework.ProxyFactoryBean"/>

  <!-- The Advice for tracing -->
  <bean id="traceInterceptor"
        class="org.springframework.aop.interceptor.SimpleTraceInterceptor"/>

  <aop:config>
    <!-- using the full class name with the '+' got this to work -->
    <aop:pointcut id="repositoryMethods" expression="execution(*
edu.harvard.fas.rregan.repository.jpa.AbstractJpaRepository+.*(..))"/>
    <aop:advisor id="traceInterceptorAdvisor" advice-
      ref="traceInterceptor" pointcut-ref="repositoryMethods"/>
  </aop:config>

  <!-- Activates @Transactional -->
  <tx:annotation-driven transaction-manager="transactionManager"/>

  <!-- Configurer for properties in beans of the form ${xxx} (replaces
with properties from files) -->
  <context:property-placeholder location="WEB-
INF/classes/db.properties"/>

  <!-- generic driver, ok for testing, but creates a separate db
connection every time
    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="${db.driver}"/>
      <property name="url" value="${db.baseUrl}${db.server}:${db.port}/${
db.name}${db.urlParams}"/>
      <property name="username" value="${db.username}"/>
      <property name="password" value="${db.password}"/>
    </bean>
  -->
  <!-- for DBCP
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
      <property name="driverClassName" value="${db.driver}"/>

```

jpaConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
  - Spring config using the
-->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context-2.5.xsd

```

```

<property name="url" value="${db.baseUrl}${db.server}:${db.port}/${
{db.name}${db.urlParams}"/>
<property name="username" value="${db.username}"/>
<property name="password" value="${db.password}"/>
</bean>
-->

<!-- C3P0
<bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
<property name="driverClass" value="${db.driver}"/>
<property name="jdbcUrl" value="${db.baseUrl}${db.server}:${db.port}/${
{db.name}${db.urlParams} " />
<property name="user" value="${db.username} " />
<property name="password" value="${db.password} " />
<property name="maxPoolSize" value="5" />
<property name="minPoolSize" value="1" />
<property name="maxIdleTime" value="5000" />
</bean>
-->
<bean id="dataSource"
class="org.logicalcobwebs.proxool.ProxoolDataSource">
<property name="alias" value="pool1" />
<property name="driver" value="${db.driver} " />
<property name="driverUrl" value="${db.baseUrl}${db.server}:${
{db.port}/ ${db.name}${db.urlParams} " />
<property name="user" value="${db.username} " />
<property name="password" value="${db.password} " />
<property name="maximumConnectionCount" value="10" />
<property name="maximumActiveTime" value="3600000" />
<property name="houseKeepingTestSql" value="select CURRENT_DATE" />
</bean>

<bean id="jdbcTemplate"
class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="dataSource"/>
</bean>

<bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryB
ean">
<property name="dataSource"><ref local="dataSource"/></property>
<property name="persistenceUnitName" value="${
{db.jpa.persistenceUnitName} " />
<property name="persistenceXmlLocation" value="classpath*:META-
INF/persistence.xml"/>
<property name="jpaDialect"><bean class="${
{db.jpa.dialect} " /></property>
<property name="jpaVendorAdapter">
<bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
<property name="databasePlatform" value="${db.hibernate.dialect} " />
<property name="showSql" value="${db.hibernate.showSql} " />
<property name="generateDdl" value="${db.hibernate.generateDdl} " />
</bean>
</property>
<!-- NOTE: don't use SimpleLoadTimeWeaver with hibernate -->
</bean>

<bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
<property name="entityManagerFactory"><ref
local="entityManagerFactory"/></property>
</bean>

<!--
This scans for all spring annotated files except for files related to
the
simple repository.
-->
<context:component-scan base-package="edu.harvard.fas.rregan">
<context:exclude-filter type="regex"
expression=".Simple.*Repository"/>
</context:component-scan>
</beans>

```