Advanced Rice Training Exercises

Leo Przybylski

Contents

Preface

Copyright

Copyright Holder

Disclaimer

About the Trainer

Leo started working with the Kuali Foundation in 2005 as a developer on the Kuali Financial System. Since then, he has worked as a *Development Manager* on the Kuali Financial System, *Lead Developer* on the Kuali Coeus project, *Software Architect* on the University of Arizona KFS implementation, and now is a *Release Engineer* for the Kuali Foundation for the Rice Project.

Leo has given six presentations on KFS, KC, and Rice on to separate Kuali Days occassions.

One significant contribution he has made to the Kuali Community is his Rice LDAP Integration module.

Using these Exercises

VirtualBox Appliance

Exercise instructions are included in this document. All software and examples are available on the VirtualBox appliance distributed during class. To install the VirtualBox appliance:

1. Copy the **Ubuntu.ova** from the distributed USB drive to your hard disk.

2. Also, copy the VirtualBox installer for your operating system from the USB drive to your hard disk.

3. Execute the VirtualBox installer to install the software.

4. Double-click on the **Ubuntu.ova**. This will begin the VM import process.

Virtual Machine Manifest

The VirtualBox appliance is an Ubuntu Linux distribution. Within it is the software we will use for this class:

Eclipse Indigo the IDE used for class. Includes Subclipse, the m2eclipse plugin, and pre-installed projects with examples.

Oracle jrockit JVM the JVM used for executing/testing examples.

soapUI Utility for debugging soap communication

Maven 3 used to build Rice applications, run tests, and start the Tomcat6 application

OpenLDAP slapd the OpenLDAP project LDAP server used for the LDAP integration examples.

ldap-utils the OpenLDAP project utilities for working with Directory Services.

Oracle MySQL Database Server where the Rice applications will store persistent information.

Credentials

User Account is **rice** with the password **rice**. This is used to unlock the VM after it has suspended, gone to sleep, or locked. The password is also required for executing commands as **root** which may on occassion be required. The user account home directory is located at **/home-/rice** and will frequently be referred to during the training.

Database Account uses the jdbc connection string **jdbc:mysql://localhost:3306/kuldemo** and the username/password **kuldemo/kuldemo**. These are the default credentials and database connection information as defined in kul-cfg-dbs.

Structure

The Eclipse workspace is located at **/home/rice/workspace**. In it are three projects used during the training: **cas**, **rice-src**, and **trnapp**.

Other examples and exercises from the **Basic Kuali Rice Training** can be found in **/home/rice/Aug2011**. You may feel free to go through these as well at your leisure.

Training Overview

Day 1
Introductions and Overview of Rice Basics

Exercise 1
Advanced Rice Customization Patterns

Description

This exercise is designed to help developers explore different patterns within Kuali Rice for development. There are many different areas for customization within Rice itself. There are also just as many different ways of doing the same thing. This can be confusing, daunting, and troublesome. In this exercise, I will explore some preferred methods for some advanced level customizations like creating your own module within Rice or adding your own Constants to be accessed from the webapp.

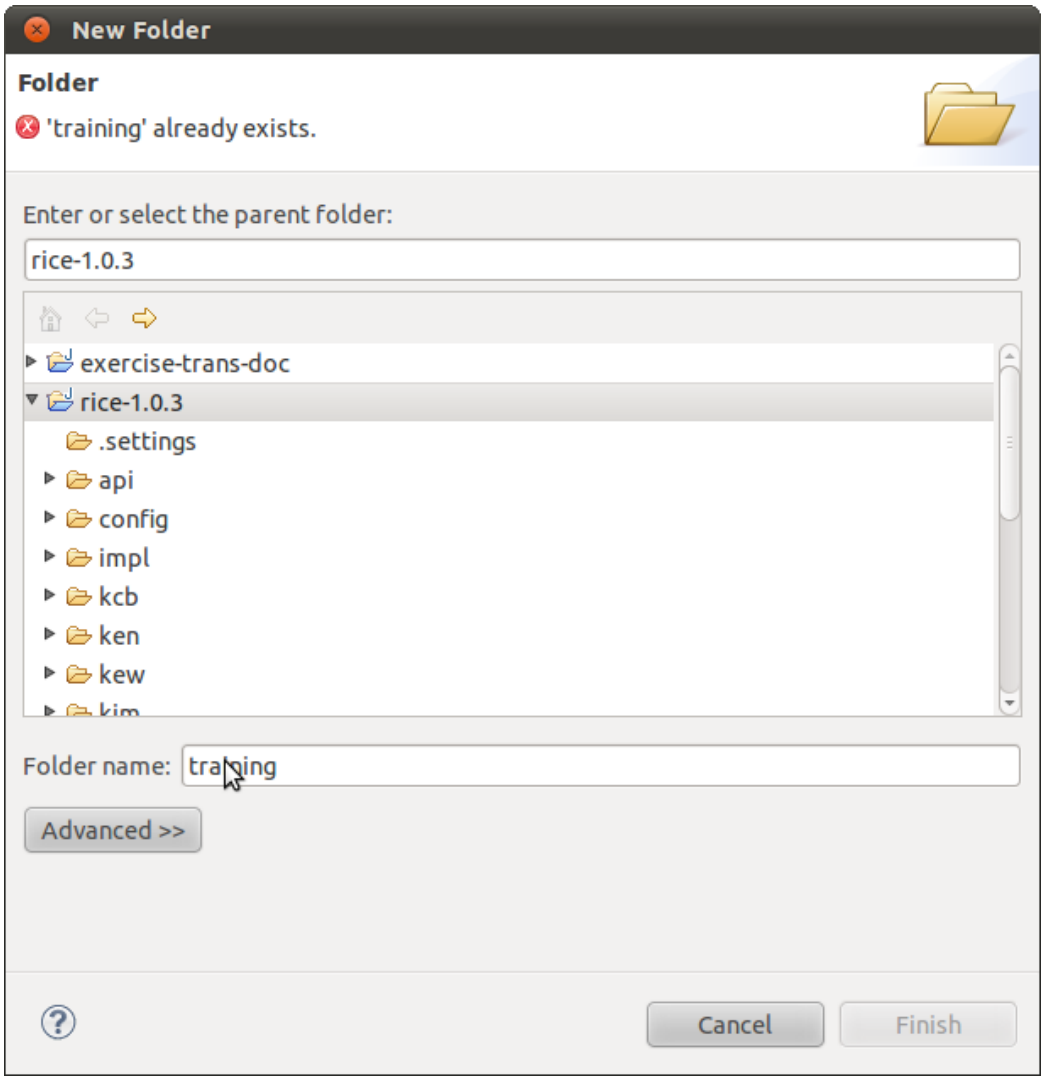
Goals

- Create a custom module.
- Create a custom DD Control class.
- See interesting ways to customize the portal.
- Create a custom JSP/JSTL function.
- Create a custom JstlListener and see how that works.
- Create custom parameters for your application-config.xml configuration
- Add an extension to a business object

1 Creating a Custom Module

1.1 Browse the Rice Source Project

1.2 Create a New Directory



Create a new directory in *rice-src* called *training*

Listing 1: Directory creation for Linux users

```
mkdir training
```

In Eclipse, Refresh your project.

1.3 Add the Standard Structure

1.3.1 Create Directories

Listing 2: Directory creation for Linux users

```
mkdir -p src/main/java/
mkdir -p src/main/resources/
mkdir -p src/main/config/
mkdir -p src/main/webapp/
mkdir -p src/test/java/
mkdir -p src/test/resources/
```

In Eclipse, Refresh your project.

1.3.1 Setup the Build Path

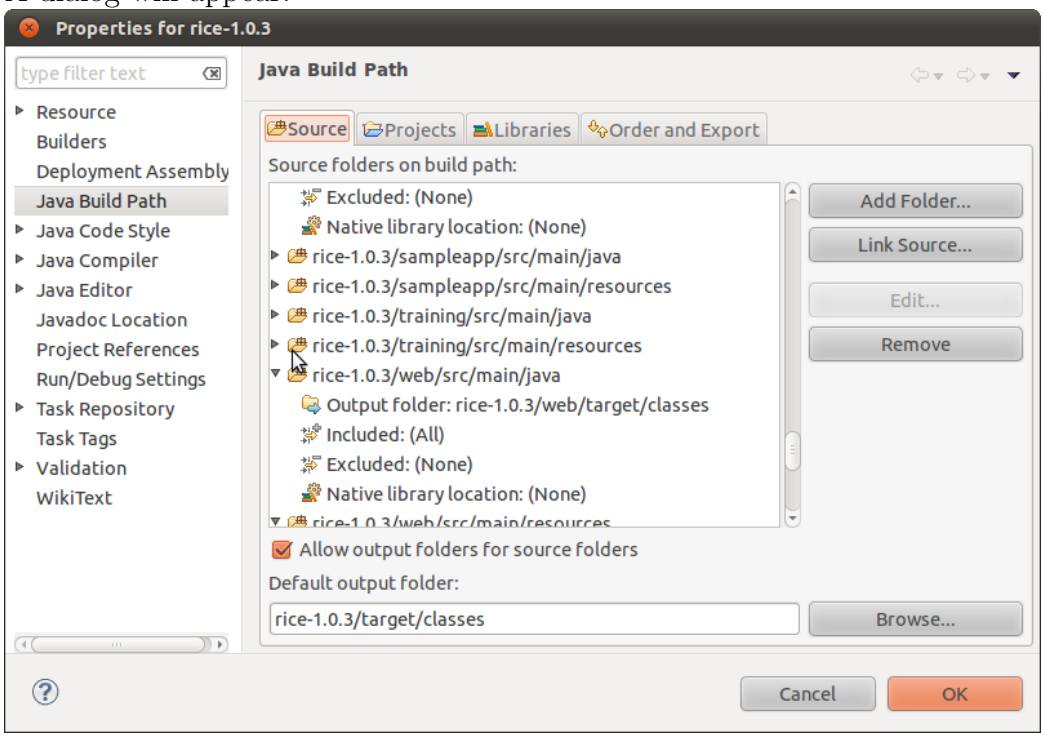
In Eclipse:

1. Expand the training folder
2. Expand the src folder
3. Expand the main folder
4. Click your right mouse button on the java folder. A context menu will appear.
5. Go to **Build Path → Use as Source Folder**
6. Click your right mouse button on the resources folder. A context menu will appear.
7. Go to **Build Path → Use as Source Folder**
8. Expand the test folder
9. Click your right mouse button on the test/java folder. A context menu will appear.
10. Go to **Build Path → Use as Source Folder**
11. Click your right mouse button on the test/resources folder. A context menu will appear.
12. Go to **Build Path → Use as Source Folder**

13. Click your right mouse button on the training folder. A context menu will appear.

14. Go to **Build Path** → **Configure Build Path**

15. A dialog will appear.



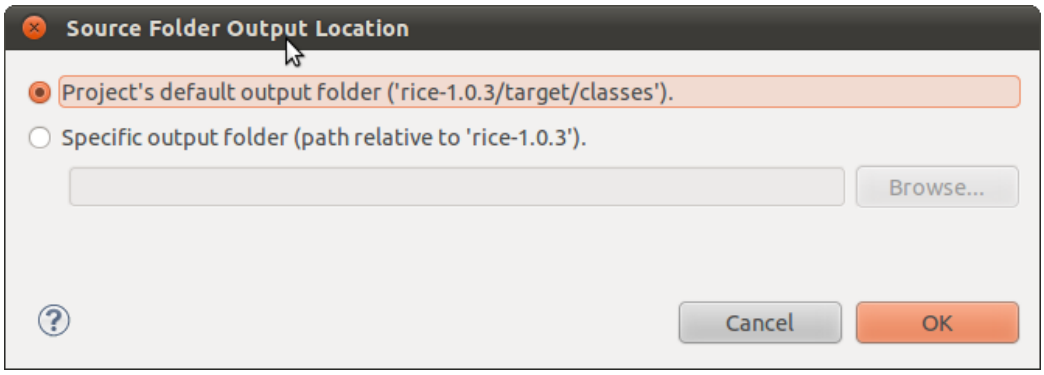
16. Click on the **Source** tab.

17. Locate rice-1.0.3/training build path configuration by scrolling

18. Expand **training/src/main/java**

19. Double-click on **Default output folder** next to the **Output Folder:**

20. A dialog will appear.



21. Toggle **Specific output folder (path relative to 'rice-1.0.3').**
22. Enter **target/classes**
23. Expand **training/src/main/resources**
24. Double-click on **Default output folder** next to the **Output Folder:**
25. A dialog will appear.
26. Toggle **Specific output folder (path relative to 'rice-1.0.3').**
27. Enter **target/classes**
28. Expand **training/src/test/java**
29. Double-click on **Default output folder** next to the **Output Folder:**
30. A dialog will appear.
31. Toggle **Specific output folder (path relative to 'rice-1.0.3').**
32. Enter **target/test-classes**
33. Expand **training/src/test/resources**
34. Double-click on **Default output folder** next to the **Output Folder:**
35. A dialog will appear.
36. Toggle **Specific output folder (path relative to 'rice-1.0.3').**
37. Enter **target/test-classes**

1.4 Create a pom.xml for the New Module

1.4.1 Stub out training/pom.xml

Listing 3: training/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <name>Advanced Rice Training</name>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.kuali.rice</groupId>
    <artifactId>rice</artifactId>
    <version>1.0.3</version>
  </parent>
  <artifactId>training</artifactId>
  <packaging>war</packaging>
</project>
```

1.4.1 Add Basic Dependencies

Listing 4: training/pom.xml

```
  <dependencies>

    <dependency>
      <groupId>${project.groupId}</groupId>
      <artifactId>rice-impl</artifactId>
      <version>${project.version}</version>
    </dependency>

  </dependencies>
```

1.4.2 Add the Maven Overlay Plugin

Listing 5: training/pom.xml

```
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.1.1</version>
        <configuration>
          <overlays>
            <overlay>
              <groupId>org.kuali.rice</groupId>
              <artifactId>rice-web</artifactId>
            </overlay>
```

```
          </overlays>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

1.4.2 Add the Dependency to rice-web

Listing 6: training/pom.xml

```
    <dependency>
      <groupId>${project.groupId}</groupId>
      <artifactId>rice-web</artifactId>
      <version>${project.version}</version>
      <type>jar</type>
      <scope>compile</scope>
      <exclusions>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>servlet-api</artifactId>
        </exclusion>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>jstl</artifactId>
        </exclusion>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>jsp-api</artifactId>
        </exclusion>
      </exclusions>
    </dependency>

    <dependency>
      <groupId>${project.groupId}</groupId>
      <artifactId>rice-web</artifactId>
      <version>${project.version}</version>
      <type>war</type>
      <scope>runtime</scope>
      <exclusions>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>servlet-api</artifactId>
        </exclusion>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>jstl</artifactId>
        </exclusion>
        <exclusion>
          <groupId>javax.servlet</groupId>
          <artifactId>jsp-api</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
```

1.4.3 Add the Tomcat Maven Plugin

Listing 7: training/pom.xml

```
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>tomcat-maven-plugin</artifactId>
        <!-- tomcat 6.0.26 -->
        <version>1.0</version>
        <configuration>
          <path>\$\{default.context.path\}</path>
        </configuration>
        <dependencies>
          <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>\$\{mysql.version\}</version>
            <scope>runtime</scope>
          </dependency>
        </dependencies>
      </plugin>
```

1.4.4 Add the Jetty Maven Plugin

Listing 8: training/pom.xml

```
      <plugin>
        <groupId>org.mortbay.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>${jetty.version}</version>
        <configuration>
          <webAppConfig>
            <contextPath>${default.context.path}</contextPath>
          </webAppConfig>
        </configuration>
        <dependencies>
          <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>\$\{mysql.version\}</version>
            <scope>runtime</scope>
          </dependency>
        </dependencies>
      </plugin>
```

1.4.5 Add the Maven Properties

Listing 9: training/pom.xml

```
  <properties>
    <build.environment>dev</build.environment>
```

```
    <default.context.path>/\$\{project.artifactId\}--\${build.environment\}<
        /default.context.path>
    <jetty.plugin.version>7.4.5.v20110725</jetty.plugin.version>
    <mysql.version>5.1.13</mysql.version>
  </properties>
```

1.4.6 Add the Maven Plugin Repositories

Listing 10: training/pom.xml

```
  <pluginRepositories>
    <pluginRepository>
      <id>kuali.nexus</id>
      <name>Nexus Repository Manager</name>
      <url>http://nexus.kuali.org/content/groups/public</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
```

1.4.6 Install rice-web.jar

```
rice@rice-VirtualBox: ~/workspace/rice-src/training
File  Edit  View  Search  Terminal  Help
  Try downloading the file manually from the project website.

  Then, install it using the command:
      mvn install:install-file -DgroupId=org.kuali.rice -DartifactId=rice-web -D
version=1.0.3 -Dpackaging=jar -Dfile=/path/to/file

  Alternatively, if you host your own repository you can deploy the file there:
      mvn deploy:deploy-file -DgroupId=org.kuali.rice -DartifactId=rice-web -Dve
rsion=1.0.3 -Dpackaging=jar -Dfile=/path/to/file -Durl=[url] -DrepositoryId=[id]

  Path to dependency:
        1) org.kuali.rice:training:war:1.0.3
        2) org.kuali.rice:rice-web:jar:1.0.3

----------
2 required artifacts are missing.

for artifact:
  org.kuali.rice:training:war:1.0.3

from the specified remote repositories:
  central (http://repo1.maven.org/maven2),
  kuali (https://test.kuali.org/maven)



[INFO] ------------------------------------------------------------------------
[INFO] For more information, run Maven with the -e switch
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 11 seconds
[INFO] Finished at: Mon Aug 22 02:27:37 MST 2011
[INFO] Final Memory: 28M/68M
[INFO] ------------------------------------------------------------------------
rice@rice-VirtualBox:~/workspace/rice-src/training$
```

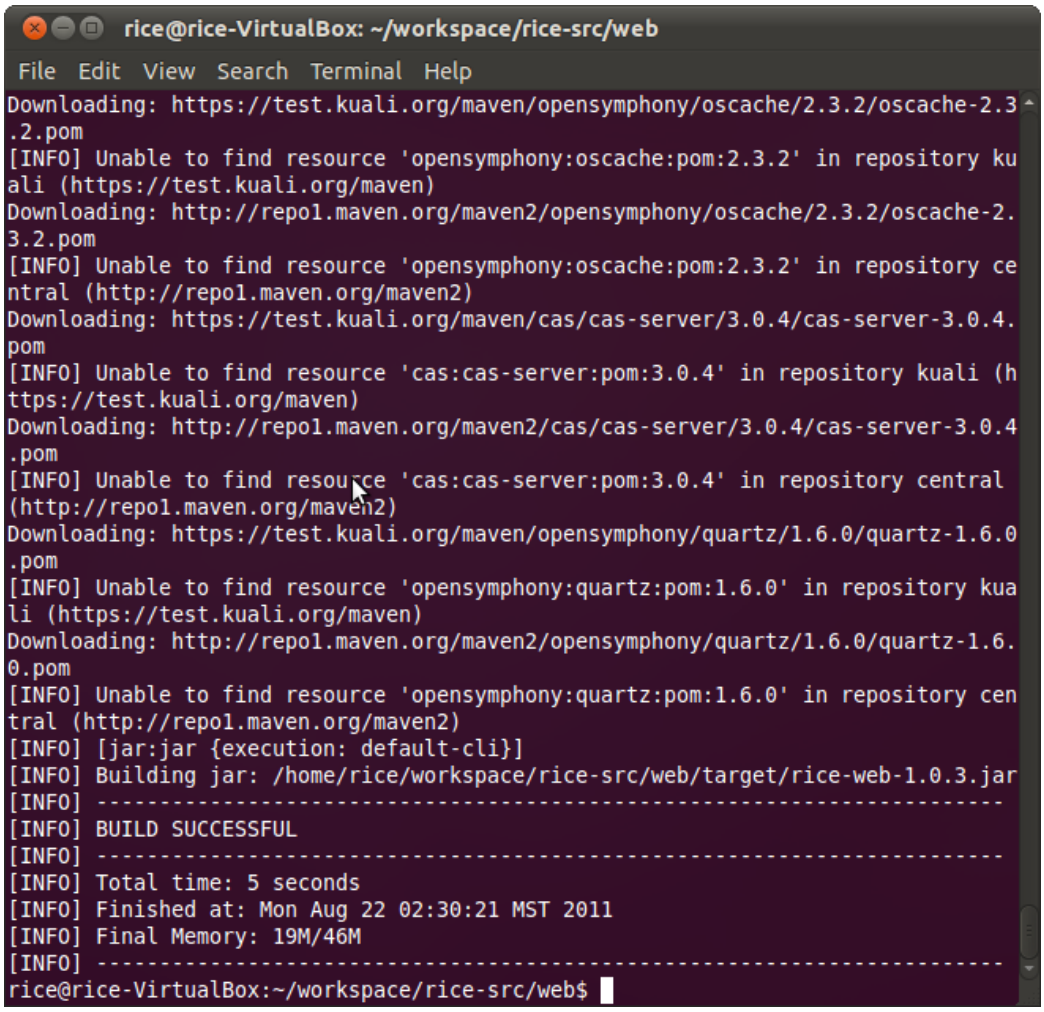
The above is what happens at this point if you try and install. The rice-web.jar dependency cannot be fulfilled because it does not exist. We need to create it.

Listing 11: Maven commands

```
cd /home/rice/workspace/rice-src/web
mvn -Dmaven.test.skip=true install
```

Listing 12: Maven commans

```
mvn jar:jar
```



Listing 13: Maven commands

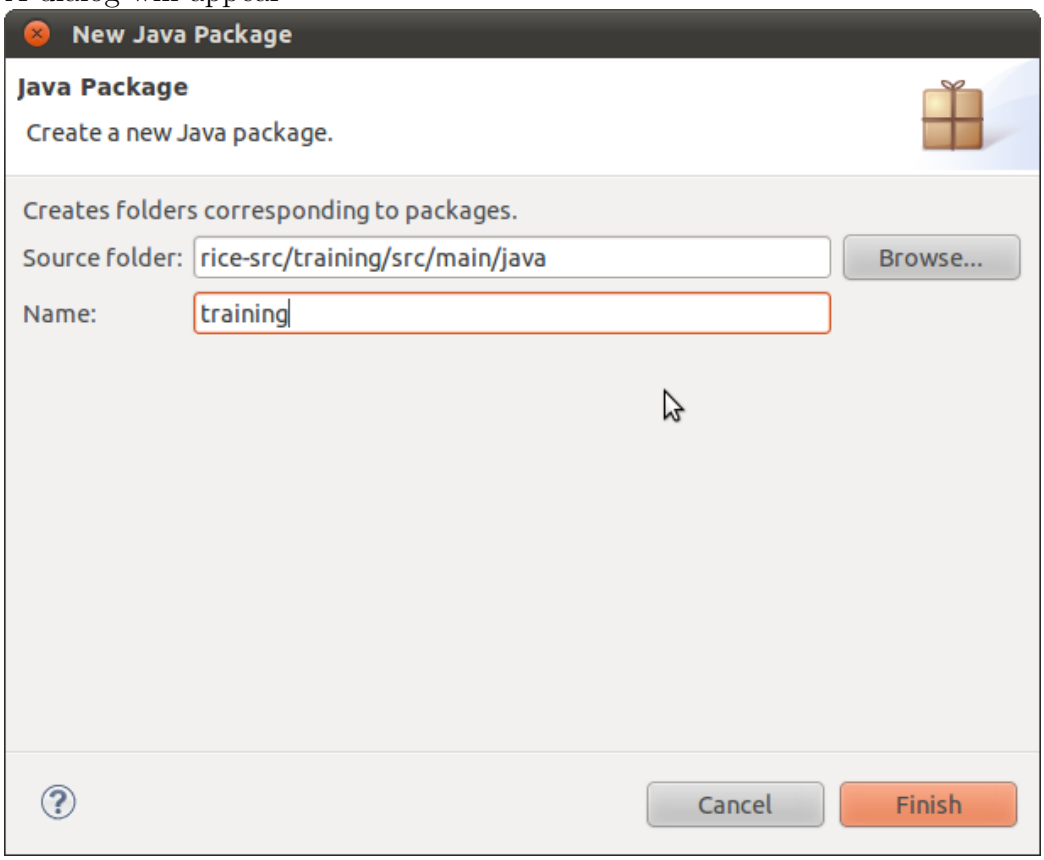
```
mvn install:install-file -DgroupId=org.kuali.rice
-DartifactId=rice-web -Dversion=1.0.3.2 -Dpackaging=jar -Dfile=target/rice-
    web-1.0.3.2.jar
```

1.5 Add a Base Package

In Eclipse:

1. Click your right mouse button on the java folder. A context menu will appear.

2. Go to **New** → **Package**
3. A dialog will appear



4. For Name, use **training**
5. Click Finish

1.6 Create Base Spring Beans

1.6.1 Stub out src/main/resources/training/SpringModuleBeans.xml

Listing 14: Base SpringModuleBeans.xml

```
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
                              http://www.springframework.org/schema/beans/
    spring-beans-2.0.xsd
                              http://www.springframework.org/schema/tx
                              http://www.springframework.org/schema/tx/spring-
    tx-2.0.xsd
                              http://www.springframework.org/schema/aop
                              http://www.springframework.org/schema/aop/spring-
    aop-2.0.xsd">
</beans>
```

1.6.1 Add the module bean with a parent

Listing 15: Base SpringBeans.xml

```

  <bean id="bookstoreModuleCconfiguration"
     parent="bookstoreModuleConfiguration-parentBean" />

  <bean id="bookstoreModuleConfiguration-parentBean" abstract="true"
      class="org.kuali.rice.kns.bo.ModuleConfiguration">
    <property name="namespaceCode" value="bookstore"/>
    <property name="initializeDataDictionary" value="true"/>
    <property name="dataDictionaryPackages">
      <list>
        <value>classpath:train/bookstore/bo/datadictionary/</value>
      </list>
    </property>
    <property name="databaseRepositoryFilePaths">
      <list>
            <value>OJB-repository-bookstore.xml</value>
      </list>
    </property>
    <property name="packagePrefixes">
      <list>
        <value>train.bookstore.bo</value>
      </list>
    </property>
  </bean>
```

1.6.1 Add Spring datasource Configuration

Listing 16: Spring datasource setup

```
  <bean id="trainingDatasource" class="org.kuali.rice.core.database.
      PrimaryDataSourceFactoryBean" lazy-init="true">
    <property name="preferredDataSourceParams">
      <list>
        <value>training.datasource</value>
      </list>
    </property>
    <property name="preferredDataSourceJndiParams">
```

```
      <list>
        <value>training.datasource.jndi.location</value>
      </list>
    </property>
    <property name="serverDataSource" value="false"/>
  </bean>

  <bean id="trainingOjbConfigurer" class="org.kuali.rice.core.ojb.BaseOjbConfigurer">
    <property name="jcdAliases">
      <list>
        <value>trainingDataSource</value>
      </list>
    </property>
    <property name="metadataLocation" value="classpath:training/OJB-repository-bookstore.xml" />
  </bean>
```

1.6.1 Setup platformAwareDao

Listing 17: Spring datasource setup src/main/resources/training/OJB-repository-bookstore.xml

```
  <bean id="platformAwareDao" abstract="true" class="org.kuali.rice.kns.dao.impl.PlatformAwareDaoBaseOjb">
    <property name="jcdAlias" value="trainingDataSource" />
    <property name="dbPlatform" ref="dbPlatform" />
  </bean>
```

1.7 Stub Base OJB Mapping

Listing 18: Stubbed OJB Descriptor file src/main/resources/OJB-repository-training.xml

```
<descriptor-repository version="1.0">

  <jdbc-connection-descriptor jcd-alias="trainingDataSource" default-connection="false" jdbc-level="3.0" eager-release="false" batch-mode="false"
      useAutoCommit="0" ignoreAutoCommitExceptions="false">
    <sequence-manager className="org.kuali.rice.core.ojb.ConfigurableSequenceManager">
      <attribute attribute-name="property.prefix" attribute-value="datasource.ojb.sequenceManager" />
    </sequence-manager>
    <object-cache class="org.apache.ojb.broker.cache.ObjectCachePerBrokerImpl" />
  </jdbc-connection-descriptor>
</descriptor>
```

1.8 Build it

1.8.1 Open a Shell to the Project Directory

Listing 19: Change directory to the project in Linux

```
cd workspace/rice-src/training
```

In Eclipse, Refresh your project.

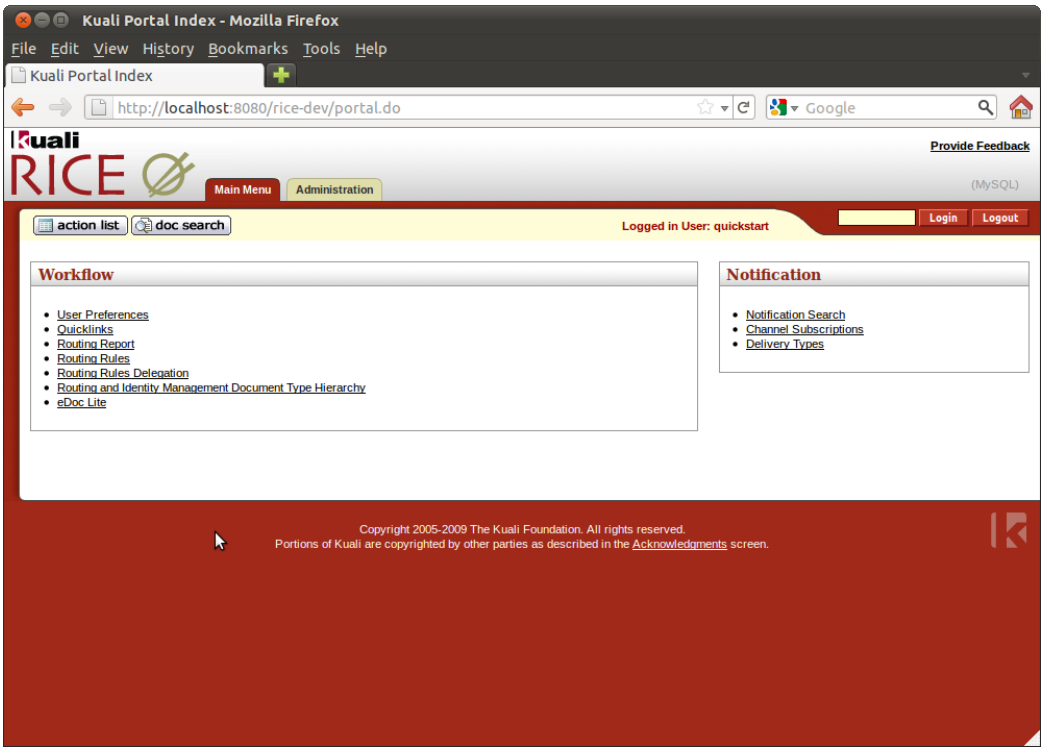
1.8.2 Go

Listing 20: Maven commands

```
cd /home/rice/workspace/rice-src/training
mvn -Dmaven.test.skip=true war:inplace
mvn -Dmaven.test.skip=true tomcat:run
```

```
rice@rice-VirtualBox: ~/workspace/rice-src/training
File  Edit  View  Search  Terminal  Help
rt.XmlWebApplicationContext - Refreshing org.springframework.web.context.support
.XmlWebApplicationContext@1035b2a: display name [WebApplicationContext for names
pace 'kcb-servlet']; startup date [Mon Aug 22 04:52:14 MST 2011]; root of contex
t hierarchy
2011-08-22 04:52:14,995 [main] u:/d: INFO  org.springframework.beans.factory.xml
.XmlBeanDefinitionReader - Loading XML bean definitions from ServletContext reso
urce [/kcb/WEB-INF/kcb-servlet.xml]
2011-08-22 04:52:14,998 [main] u:/d: INFO  org.springframework.web.context.suppo
rt.XmlWebApplicationContext - Bean factory for application context [org.springfr
amework.web.context.support.XmlWebApplicationContext@1035b2a]: org.springframewo
rk.beans.factory.support.DefaultListableBeanFactory@e7403
2011-08-22 04:52:15,001 [main] u:/d: INFO  org.springframework.beans.factory.sup
port.DefaultListableBeanFactory - Pre-instantiating singletons in org.springfram
ework.beans.factory.support.DefaultListableBeanFactory@e7403: defining beans [me
ssageSource,recipientPreferenceService,messageDelivererRegistryService,kenIntegr
ationService,org.springframework.beans.factory.config.PropertyPlaceholderConfigu
rer#0,org.springframework.web.context.support.ServletContextAttributeExporter#0,
viewResolver,urlMapping,org.springframework.web.servlet.handler.SimpleMappingExc
eptionResolver#0,prefsController]; root of factory hierarchy
2011-08-22 04:52:15,001 [main] u:/d: INFO  org.springframework.web.context.suppo
rt.ServletContextAttributeExporter - Exported ServletContext attribute with name
 'kcb.prefix'
2011-08-22 04:52:15,013 [main] u:/d: INFO  org.springframework.web.servlet.Dispa
tcherServlet - FrameworkServlet 'kcb': initialization completed in 19 ms
Aug 22, 2011 4:52:15 AM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
Aug 22, 2011 4:52:15 AM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
```

You should see the above after starting tomcat

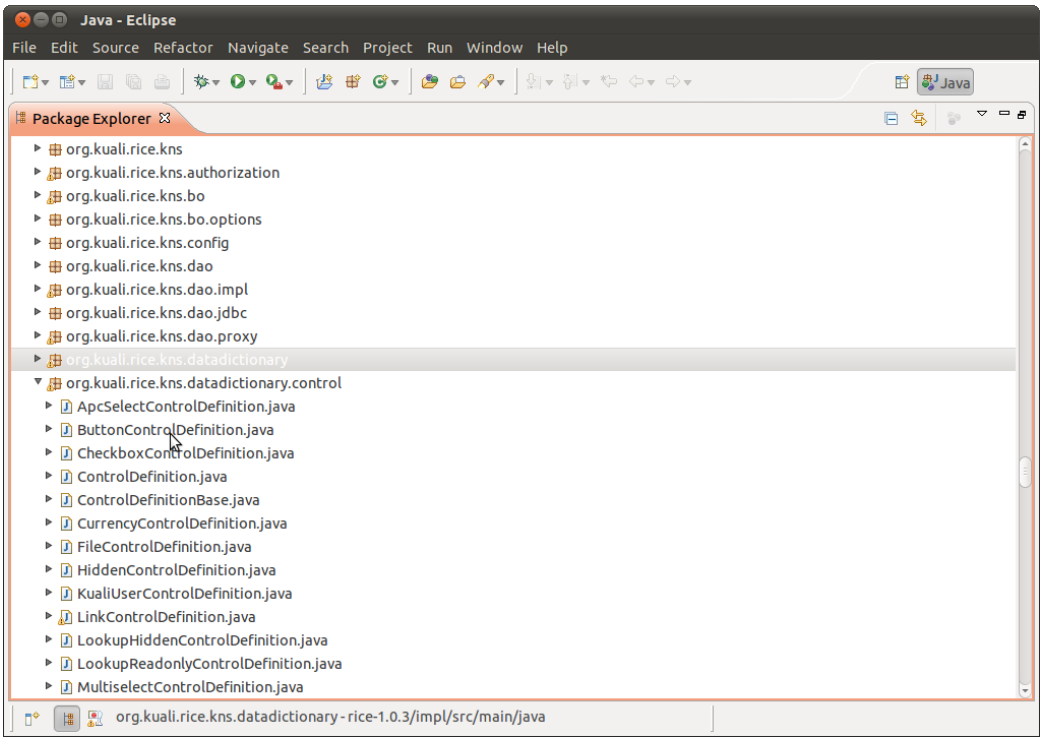


Browse to http://localhost:8080/rice-dev/. Then login as quickstart. You should see the above.

2 Creating a Custom Data Dictionary Control

Control definitions are just Spring beans accessed via the BusinessObjectMetadataService and the DataDictionaryService. Besides controls, you can create or extend any aspect of the DataDictionary including relationships, lookup definitions, and the workflow attributes.

The core DataDictionary classes are located in **impl/src/main/java/org/kuali/rice/kns/datadictionary/** of your rice source code distribution. - Eclipse



.png

2.1 Add a datadictionary Package to Our Module

Listing 21: Directory creation for Linux users

```
mkdir -p training/src/main/java/training/datadictionary/control
```

In Eclipse, Refresh your project.

2.2 Stub SuggestsBox Control in src/main/java/training/datadictionary/control

Listing 22: Stubbed OJB Descriptor file src/main/resources/OJB-repository-training.xml

```
package training.datadictionary.control;

/**
 *
 */
public class SuggestsBoxDefinition extends ControlDefinitionBase {
    private static final long serialVersionUID = -1L;

```

```
        public SuggestsBoxDefinition() {
    }

    /**
     * @see java.lang.Object#toString()
     */
    public String toString() {
        return "SuggestsBoxDefinition";
    }
}
```

3 Creating a Custom JSTL Function

3.1 Add a web Package to Our Module

Listing 23: Directory creation for Linux users

```
mkdir -p training/src/main/java/training/web/
```

3.2 Stub a TrainingFunctions Class in the web Package

Listing 24: training/web/TrainingFunctions

```
package training.web;

/**
 * Full of static methods for JSTL function access.
 *
 */
public final class TrainingFunctions {
}
```

3.2 Add a Method that Fetches System Parameters

Listing 25: training/web/TrainingFunctions

```
...
...
import static org.kuali.rice.kns.service.KNSServiceLocator.
    getParameterService

public final static boolean paramExists(final String component, final String
name) {
    return getParameterService().parameterExists("KR-TRN", component, name);
}

public final static String getParameter(final String component, final String
name) {
```

```
    return getParameterService().getParameterValue("KR-TRN", component, name);
}

public final static List<String> getParameters(final String component, final String
name) {
    return getParameterService().getParameterValues("KR-TRN", component, name);
}

public final static boolean isEnabled(final String component, final String
name) {
    return getParameterService().getIndicatorParameter("KR-TRN", component, name);
}
```

3.3 Stub a Tag Library Definition for the Module

Listing 26: src/main/webapp/WEB-INF/tlds/trnfunc.tld Tag Library Definition

```
<taglib xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd" version="2.0">
</taglib>
```

3.4 Define the Tag Library

Listing 27: src/main/webapp/WEB-INF/tlds/trnfunc.tld Tag Library Definition

```
    <description>Training functions library</description>
    <display-name>Training functions</display-name>
    <tlib-version>1.0</tlib-version>
    <short-name>fn</short-name>
    <uri>http://www.kuali.org/jsp/jstl/functions</uri>
```

3.5 Add the Functions from TrainingFunctions

Listing 28: src/main/webapp/WEB-INF/tlds/trnfunc.tld Tag Library Definition

```
    <function>
        <description>Access System Parameters from JSTL</description>
        <name>parameterExists</name>
```

```
        <function-class>training.web.TrainingFunctions</function-class>
        <function-signature>boolean parameterExists(java.lang.String, java.lang.String)</function-signature>
        <example>&lt;c:if test="${trn-fn:parameterExists('Document', 'ACTIVE_FILE_TYPES')}"&gt;&lt;/c:if&gt;</example>
    </function>

    <function>
        <description>Access System Parameters from JSTL</description>
        <name>getParameter</name>
        <function-class>training.web.TrainingFunctions</function-class>
        <function-signature>java.lang.String getParameter(java.lang.String, java.lang.String)</function-signature>
        <example>${trn-fn:getParameter('Document', 'ACTIVE_FILE_TYPES')}</example>
    </function>

    <function>
        <description>Access System Parameters from JSTL</description>
        <name>getParameters</name>
        <function-class>training.web.TrainingFunctions</function-class>
        <function-signature>java.util.List getParameters(java.lang.String, java.lang.String)</function-signature>
        <example>${trn-fn:getParameters('Document', 'ACTIVE_FILE_TYPES')}</example>
    </function>

    <function>
        <description>Access System Parameters from JSTL</description>
        <name>isEnabled</name>
        <function-class>training.web.TrainingFunctions</function-class>
        <function-signature>boolean isEnabled(java.lang.String, java.lang.String)</function-signature>
        <example>${trn-fn:isEnabled('Document', 'ALLOW_NEGATIVE_BALANCE_IND')}</example>
    </function>
```

3.6 Test Yourself: What have you learned?

Using the same examples above, add a call to **hasPermission** that you can use to restrict a tab in the portal specifically for FO's.

3 Adding to JstlConstantsInitListener

JstlConstantsInitListener can be found in **impl/src/main/java/org/kuali/rice/kns/web/listener/JstlConstantsInitListener.java** of your rice source distribution.

3.1 Add a listener Package to Our Module

Listing 29: Directory creation for Linux users

```
mkdir -p training/src/main/java/training/web/listener
```

3.1 Stub out a ContextListener

Listing 30: training/web/TrainingFunctions

```
package training.web.listener;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;


/**
 * This class is the JstlContants implementation of the
     ServletContextListener.
 */
public class JstlConstantsInitListener implements ServletContextListener {
}
```

3.1 Add a Constant

Listing 31: training/web/TrainingFunctions

```
package training;


import org.kuali.rice.core.util.JSTLConstants;

public class TrainingConstants extends JSTLConstants {
  public static String CURRENT_PROCESS_DATE_PARAMETER = "
      CURRENT_PROCESS_DATE";
}
```

3.1 Add Constants to the Listener

Listing 32: training/web/TrainingFunctions

```
    import training.TrainingConstants;
    ...
    ...
    public void contextInitialized(ServletContextEvent sce) {

        ServletContext context = sce.getServletContext();
```

```
        // publish application constants into JSP app context with name "Constants"
        context.setAttribute("TrainingConstants", new TrainingConstants());
    }
```

3.1 Add the New InitListener to the web.xml

The listener needs to be added to the Rice web.xml file. There is only one web.xml file for an application. Each rice project has one. It will be located in your project source tree at src/main/webapp/WEB-INF/

In it, you will see a section with listeners that looks like:

Listing 33: src/main/webapp/WEB-INF/web.xml

```
    <listener>
      <listener-class>org.kuali.rice.core.web.listener.StandaloneInitializeListener</listener-class>
    </listener>

    <listener>
      <listener-class>org.kuali.rice.kns.web.listener.JstlConstantsInitListener</listener-class>
    </listener>

    <listener>
      <listener-class>org.kuali.rice.kns.web.listener.KualiHttpSessionListener</listener-class>
    </listener>
```

Just add yours. Now when your application starts, you will be able to access your constants from the JSP/JSTL.

Exercise 2
Portal Customization

Description

Besides a middleware framework and API, Rice supplies a reference implementation portal with administrative focused user interfaces. Most institutions will want to customize this to fit their users and their functional needs.

That will involve some modification at just about any level. This exercise explores advanced techniques for portal customization

Goals

- Learn new ways to use javascript to communicate with SOA
- How to creatively add functionality and rich user interface design to the Kuali Portal

1 Add a New Tab Restricted by Permission

2 Add Tooltips to Channels

Notes

Day 2
Advanced Kuali Enterprise Workflow

Exercise 3
Complex Routing Patterns

Description

This exercise is going to build on another exercise taken from the Basic Kuali Rice Training, and bring it all together as one complete exercise. We are going to create a new type of product for sale that is exactly like the previous.

We are going to use an *ImportedBookDocument* to demonstrate capabilities for *Complex Role-based Routing*, *Complex Split Nodes*, *Custom Document Type Statuses*, and

Goals

- create a split node based on the contents of a document
- route to a responsibility based upon a role that is determined by a user's department
- add a custom status to a document which will reveal more detail about routing
- learn how to make a customized action list

1 Inherited Routing

We will reuse the existing exercise information for our routing. We will first need to create a new **ImportedBookOrderDocument** to handle books imported from outside the US.

1.1 Create a new train.bookstore.document.ImportedBookOrderDocument

Create a new class called **ImportedBookOrderDocument** in the **train.bookstore.document** package. It will inherit from the **train.bookstore.document.BookOrderDocument**

1.2 Create a new ImportedBookOrderDocument Workflow Doctype

Listing 34: New ImportedBookOrderDocumentType

```
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="ns:workflow" xmlns:fo="http://www.w3.org/1999/XSL/Format"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
    ="ns:workflow resource:WorkflowData">
  <documentTypes xmlns="ns:workflow/DocumentType" xsi:schemaLocation="
      ns:workflow/DocumentType resource:DocumentType">
    <documentType>
      <name>ImportedBookOrderDocumentType</name>
      <parent>BookOrderDocumentType</parent>
      <label>Imported Book Order</label>
    </documentType>
  </documentTypes>
</data>
```

1.3 Ingest the New Document Type

1. Start the Training Application
2. Login as the **admin** user
3. Select the **Administration** tab
4. Locate **XML Ingester** under the **Workflow** Channel
5. Ingest the document type here

1.4 Create a new ImportedBookOrderDocument Workflow Doctype

The application should already be started. You can try and test it out. The behavior should be the same for the **ImportedBookOrderDocument** as for the **BookOrderDocument** to *Fiscal Approval*.

1.5 Create a new ImportedBookOrderDocument.xml DataDictionary Entry

We also need to create a DataDictionary entry for the new **ImportedBookOrderDocument**.

1. Edit **src/main/resources/train/bookstore/document/datadictionary/ImportedBookOrderDocument.xml**

2. Create a bean that inherits from the **BookOrderDocument**

Listing 35: ImportBookOrderDocument.xml

```
<bean id="ImportedBookOrderDocument-parentBean" abstract="true"
    parent="BookOrderDocument-parentBean">
      <property name="documentTypeName" value="
          ImportedBookOrderDocumentType"/>
      <property name="documentClass"
    value="train.bookstore.document.ImportedBookOrderDocument"/>
```

The final data dictionary entry looks like:

Listing 36: ImportBookOrderDocument.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http:
    //www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.
    springframework.org/schema/p" xmlns:dd="http://rice.kuali.org/dd"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
    http://rice.kuali.org/dd http://rice.kuali.org/dd/dd.xsd">


  <bean id="ImportedBookOrderDocument" parent="ImportedBookOrderDocument-
      parentBean"/>

  <bean id="ImportedBookOrderDocument-parentBean" abstract="true" parent="
      BookOrderDocument-parentBean">
        <property name="documentTypeName" value="
            ImportedBookOrderDocumentType"/>
        <property name="documentClass"
    value="train.bookstore.document.ImportedBookOrderDocument"/>
</beans>
```

2 Role-Based Routing

For imported books, they will need to first go through Customs.

2.1 Create the Customs Derived Role

To do this we will need a KIM Type. We will make the KIM Type first and then the role.

2.1.1 Stub a CustomsDerivedRoleTypeServiceImpl class

Add the following class to **train.bookstore.identity.service.impl**

Listing 37: Stubbed CustomsDerivedRoleTypeServiceImpl.java

```
public class CustomsDerivedRoleTypeServiceImpl extends
    KimDerivedRoleTypeServiceBase {
    /**
     *
     * @see org.kuali.rice.kim.service.support.impl.KimRoleTypeServiceBase#
         getPrincipalIdsFromApplicationRole(java.lang.String, java.lang.
         String, org.kuali.rice.kim.bo.types.dto.AttributeSet)
     */
    @Override
    public List<RoleMembershipInfo> getRoleMembersFromApplicationRole(String
         namespaceCode, String roleName, AttributeSet qualification) {
        final List<RoleMembershipInfo> members = new ArrayList<
            RoleMembershipInfo>();
        return members
    }
}
```

Currently, this will return just about anyone. What we intend to do is return only add a user to the Customs Role when a person is found in the **CUSTOMS** department. This means we will need to make use of KIM and the **PersonService** to do this.

2.1.2 Inject the PersonService

Listing 38: Stubbed CustomsDerivedRoleTypeServiceImpl.java

```

    private PersonService personService;

    public void setPersonService(final PersonService personService) {
        this.personService = personService;
    }

    protected PersonService getPersonService() {
        return personService;
    }
```

2.1.2 Finish Implementing getRoleMembersFromApplicationRole

So far, this is what we have:

Listing 39: Stubbed customsDerivedRoleTypeServiceImpl.java

```
    public List<RoleMembershipInfo> getRoleMembersFromApplicationRole(String
         namespaceCode, String roleName, AttributeSet qualification) {
```

```
        final List<RoleMembershipInfo> members = new ArrayList<
            RoleMembershipInfo>();
        return members;
    }
```

What we want to do is implement this to use the **findPeople()** method in **PersonService** to lookup all **Person** instances where **primaryDepartmentCode** is *CUSTOMS*.

Once we have those **Person** instances, we are going to need to create **RoleMembershipInfo** instances from those and add them to our list of members. To do that you want to use something like:

Listing 40: New RoleMembershipInfo snippet

```
final RoleMembershipInfo member = new RoleMembershipInfo(null,null,person.
    getPrincipalId(),Role.PRINCIPAL_MEMBER_TYPE,null)
```

2.1.2 Configure the CustomsDerivedRoleTypeServiceImpl

By *configure*, I mean into Spring.

1. Open the **trnapp-BookstoreModuleBeans.xml** file.

2. Find a space near the bottom of the file and add:

Listing 41: trnapp-BookstoreModuleBeans.xml

```
<bean id="customDerivedRoleTypeService"
  class="train.identity.service.impl.CustomDerivedRoleTypeServiceImpl"
  />
```

3. Publish the service on the bus (KSB). On the line just below, type:

Listing 42: trnapp-BookstoreModuleBeans.xml

```
<bean class="org.kuali.rice.ksb.messaging.KSBExporter">
  <property name="serviceDefinition">
    <bean class="org.kuali.rice.ksb.messaging.JavaServiceDefinition">
      <property name="serviceNameSpaceURI" value="" />
      <property name="localServiceName" value="
          customDerivedRoleTypeService" />
      <property name="service" ref="customDerivedRoleTypeService" />
    </bean>
  </property>
</bean>
```

2.1.3 Create a customDerivedRoleTypeService Kim Type Using the Service

Listing 43: New KIM Type

```
INSERT INTO krim_typ_t (
        kim_typ_id, nmspc_cd, nm, srvc_nm, actv_ind, obj_id
) VALUES (
LAST_INSERT_ID(),
'TRNAPP',
'Derived Role: Customs',
'customsDerivedRoleTypeService',
'Y',
UUID());
```

2.1.4 Create a Customs Role Using the Type

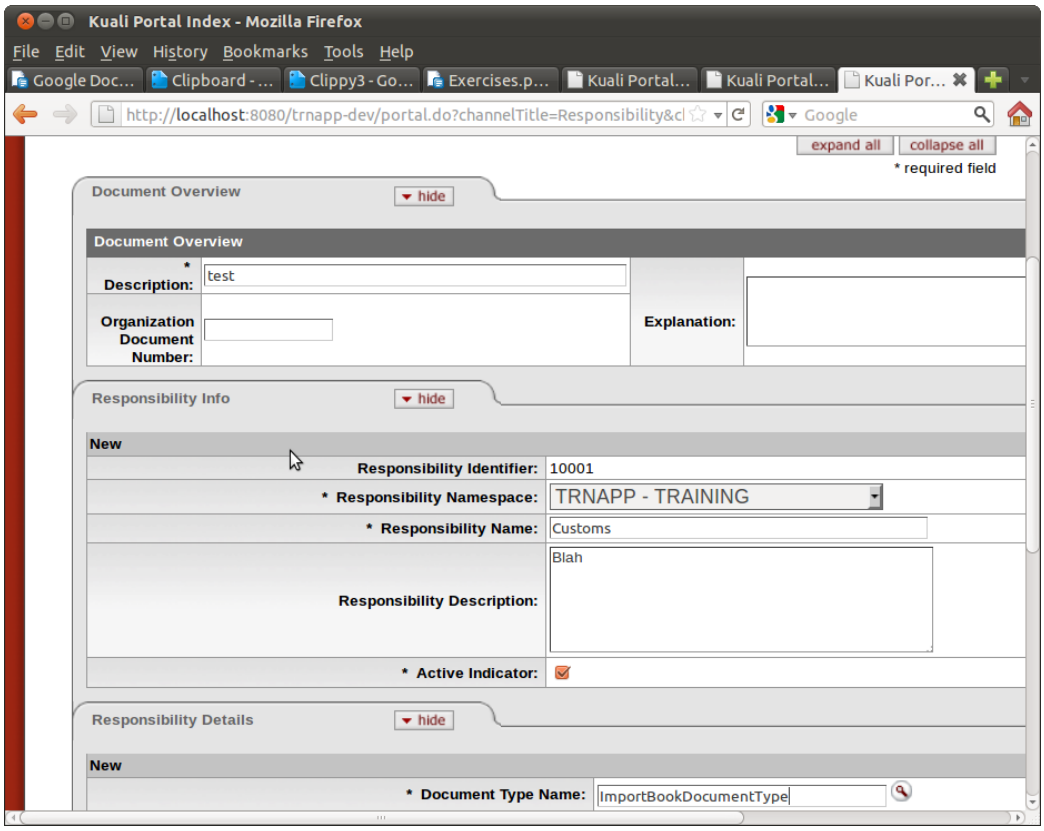
Listing 44: New KIM Type

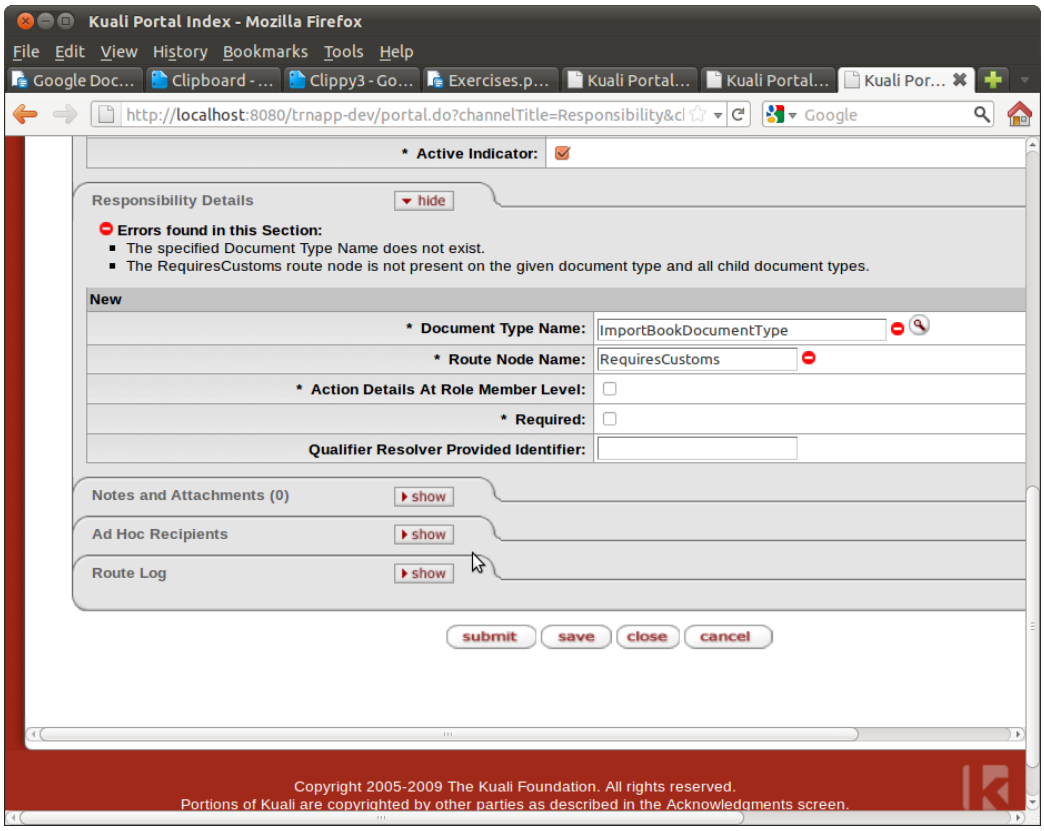
```
INSERT INTO KRIM_ROLE_T (
    ROLE_ID, OBJ_ID, ROLE_NM, NMSPC_CD, DESC_TXT, KIM_TYP_ID,
    ACTV_IND, LAST_UPDT_DT
) VALUES (
LAST_INSERT_ID,
UUID(),
'Customs',
'TRNAPP',
'',
select max(id) from KRIM_TYP_ID_S,
'Y',
SYSDATE());
```

2.2 Create a Customs Responsibilty and Assign it

2.2.1 New Customs Responsibilty

1. Browse the application at `http://localhost:8080/trnapp-dev/portal.do`
2. Click on the **Administration** tab
3. Click on the **Responsibility** lookup link
4. Click on **create new**


Kuali Portal Index - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google Doc... Clipboard - ... Clippy3 - Go... Exercises.p... Kuali Portal... Kuali Portal... Kuali Por...
http://localhost:8080/trnapp-dev/portal.do?channelTitle=Responsibility&cl
Google
expand all
collapse all
* required field
Document Overview
hide
Document Overview
* Description:
test
Organization Document Number:
Explanation:
Responsibility Info
hide
New
Responsibility Identifier:
10001
* Responsibility Namespace:
TRNAPP - TRAINING
* Responsibility Name:
Customs
Responsibility Description:
Blah
* Active Indicator:
Responsibility Details
hide
New
* Document Type Name:
ImportBookDocumentType



2.2.2 Assign Customs Responsibility to the Customs Role

2.3 Add a User to the Customs Role

2.4 Update the ImportedBookOrderDocumentType.xml

Now that the basic information has been created for the the derived role, we need to tell Workflow to use it.

1. Edit the **ImportedBookOrderDocumentType.xml**
2. Add a **routePath**

Listing 45: ImportBookOrderDocument.xml

```
<routePaths>
    <routePath>
        <start name="AdHoc" nextNode="
            Warehouse_Processing" />
```

```
                                <role name="Warehouse Processing"
                                    nextNode="Customs" />
                                <role name="Customs" />
                            </routePath>
                        </routePaths>
```

3. Add a **routeNode**

Listing 46: ImportBookOrderDocument.xml

```
                        <routeNodes>
                            <start name="AdHoc" />
                            <requests name="Warehouse Processing">
                                <activationType>P</activationType>
                                <ruleTemplate>
                                    WarehouseProcessingTemplate</
                                    ruleTemplate>
                            </requests>
                            <role name="Customs">
                                <type>org.kuali.rice.kns.workflow.
                                    attribute.
                                    DataDictionaryQualifierResolver</
                                    type>
                            </role>
                        </routeNodes>
```

2.5 Update the ImportedBookOrderDocument.xml

Listing 47: ImportBookOrderDocument.xml

```
        <bean
                id="DocumentValuePathGroup-ImportedBookOrderDocument-
                    RequiresCustoms-bookOrderEntries"
                parent="org.kuali.rice.kns.workflow.attribute.
                    DataDictionaryQualifierResolver">
                <property name="documentCollectionPath">
                        <bean class="org.kuali.rice.kns.datadictionary.
                            DocumentCollectionPath">
                            <property name="collectionPath"
                value="bookOrderEntries" />
            </bean>
        </property>
    </bean>
```

3 Complex Split Nodes

3.1 Modify the Existing ImportedBookOrderDocumentType.xml

Now we edit the **ImportBookOrderDocumentType.xml** and add the following **routePath** and **routeNode** information.

Listing 48: ImportBookOrderDocumentType.xml

```
<routePaths>
                        <routePath>
                                <start name="AdHoc" nextNode="Warehouse Processing" />
                                <split name="RequiresCustoms">
                    <branch name="False">
                    </branch>
                    <branch name="True">
                                <role name="Warehouse Processing" nextNode="Customs" />
                                <role name="Customs" />
                                </branch>
                    <join name="JoinRequiresCustoms" />
                </split>
                        </routePath>
 </routePaths>
                <routeNodes>
                        <start name="AdHoc" />
                        <split name="RequiresCustoms">
                                <type>
                                        org.kuali.rice.kew.actions.SimpleBooleanSplitNode
                                </type>
                        </split>
                        <requests name="Warehouse Processing">
                                <activationType>P</activationType>
                                <ruleTemplate>WarehouseProcessingTemplate</ruleTemplate>
                        </requests>
                        <join name="JoinRequiresCustoms" />
                        <role name="Customs">
                                <type>org.kuali.rice.kns.workflow.attribute.DataDictionaryQualifierResolver
                                </type>
                        </role>
        </routeNodes>
```

3.2 Modify the ImportedBookOrderDocument.java

The following **answerSplitNodeQuestion()** method must be overridden in the **ImportedBookOrderDocument**

Listing 49: New KIM Type

```
    /**
     * @see org.kuali.kfs.sys.document.FinancialSystemTransactionalDocumentBase#answerSplitNodeQuestion(java.lang.String)
     */
    @Override
    public boolean answerSplitNodeQuestion(String nodeName) throws UnsupportedOperationException {
        if ("RequiresCustoms".equals(nodeName)) {
```

```
            if (getBookOrderEntries().size() > 1) {
                return true;
            }
            else {
                return false;
            }
        }
        throw new UnsupportedOperationException("No split node logic
        defined for split node "+nodeName+" on the Imported Book Order
        document");
    }
```

4 Adding Users to the Customs Derived Role

4.1 Look up a Person and Add Set the primaryDepartmentCode to CUSTOMS

1. Browse the application at `http://localhost:8080/trnapp-dev/portal.do`
2. Backdoor login as **admin**
3. Click on the Administration Tab
4. Go to the Person Lookup
5. Search for **user1** principalName
6. Click the **edit** link
7. Add an **Affiliation** of type **STAFF**
8. Add employment information. Set the **primaryDepartmentCode** to **CUSTOMS**

5 Create and Route a New Imported Book Order Document

5.1 Test the case of creating a book order with 2 entries

Notes

Day 3
Advanced Kuali Identity Management

Exercise 4
ldapsearch Tool and the Query Syntax

Description

Use the ldapsearch tool to try various query examples in order to find specific users with different information.

Goals

- Learn LDAP Query Syntax
- Learn to use OpenLdap tools

Instructions

1. Start by verifying the tools are installed by running ldapsearch on the commandline by with '-h' to access the help information

   Listing 50: Verify the ldapsearch is there

   ```
   ldapsearch -h
   man ldapsearch
   ```

2. Search by connecting to the localhost LDAP server to the **cn-admin,dc=rsmart,dc=com** distinguished name and the **ou=people,dc=rsmart,dc=com** organizational unit, and query for anyone with the **eduPersonPrincipalName** of leo

   Listing 51: Verify the ldapsearch is there

   ```
   ldapsearch -H ldap://localhost -D "cn=admin,dc=rsmart,dc=com" -b
   "ou=people,dc=rsmart,dc=com" -w rice
   "(eduPersonPrincipalName=leo)"
   ```

   This query should result in the following:

Listing 52: Basic query using eduPersonPrincipalName

```
# extended LDIF
#
# LDAPv3
# base <ou=people,dc=rsmart,dc=com> with scope subtree
# filter: (eduPersonPrincipalName=leo)
# requesting: ALL
#

# 10000000, people, rsmart.com
dn: uid=10000000,ou=people,dc=rsmart,dc=com
uid: 10000000
eduPersonPrincipalName: leo
mail: leo@rsmart.com
eduPersonPrimaryAffiliation: student
eduPersonAffiliation: former-employee
eduPersonAffiliation: former-staff
eduPersonAffiliation: member
eduPersonAffiliation: student
departmentNumber: 9507
sn: Przybylski
givenName: Leonard
cn: Leonard Przybylski
employeeStatus: T
employeeZip: 85721-0073
employeeState: AZ
employeeCity: TUCSON
employeePhone: 5206266997
employeeNumber: 133006641
objectClass: top
objectClass: eduPerson
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: rsmartEmployee

# 10000000, people, rsmart.com
dn: cn=10000000,ou=people,dc=rsmart,dc=com
uid: 10000000
eduPersonPrincipalName: leo
mail: leo@rsmart.com
eduPersonPrimaryAffiliation: student
eduPersonAffiliation: former-employee
eduPersonAffiliation: former-staff
eduPersonAffiliation: member
eduPersonAffiliation: student
departmentNumber: 9507
sn: Przybylski
givenName: Leonard
cn: Leonard Przybylski
cn: 10000000
employeeStatus: T
employeeZip: 85721-0073
employeeState: AZ
employeeCity: TUCSON
employeePhone: 5206266997
employeeNumber: 133006641
objectClass: top
```

```
objectClass: eduPerson
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: rsmartEmployee

# search result
search: 2
result: 0 Success

# numResponses: 3
```

Information you want to analyze that will be used later is that we are using

Listing 53: Verify the ldapsearch is there

```
uid: 10000000
eduPersonPrincipalName: leo
```

Later, we will need to adjust our mappings in our integration to be aware of these fields. Knowing what they are is important.

Also, notice that we are using the following LDAP objectClasses.

inetOrgPerson or Internet Organization Person. This schema is a standard in Directory Services containing a number of important attribute fields we use in our data like **employeeNumber**, **departmentNumber**, and **employeeType**.

eduPerson is an internet2 standard for information about higher education people to be shared across institutions. Fields from this that we're using are anything starting with **eduPerson***

rsmartEmployee my own schema I created to fill in gaps for employees like the **employeeStatus** field which is actually really important.

3. Execute ldapsearch again. This time search for a **cn** of 10000000 **OR mail** field of `*b*@rsmart.com`

Listing 54: LDAP Query syntax using OR notation

```
ldapsearch -H ldap://localhost -D "cn=admin,dc=rsmart,dc=com" -b
"ou=people,dc=rsmart,dc=com" -w rice
"(|(cn=10000000)(mail=*b*@rsmart.com))"
```

As in most query syntaxes, **OR** and **AND** are pretty much universal. What sets LDAP apart is how most criteria is contained within **()**. Further, operators are prefixed. That is, they come before the criteria. Notice how the | comes first in `"(|(cn=10000000)(mail=*b*@rsmart.com))"`

Exercise 5
Implementing LDAP Entity Integration

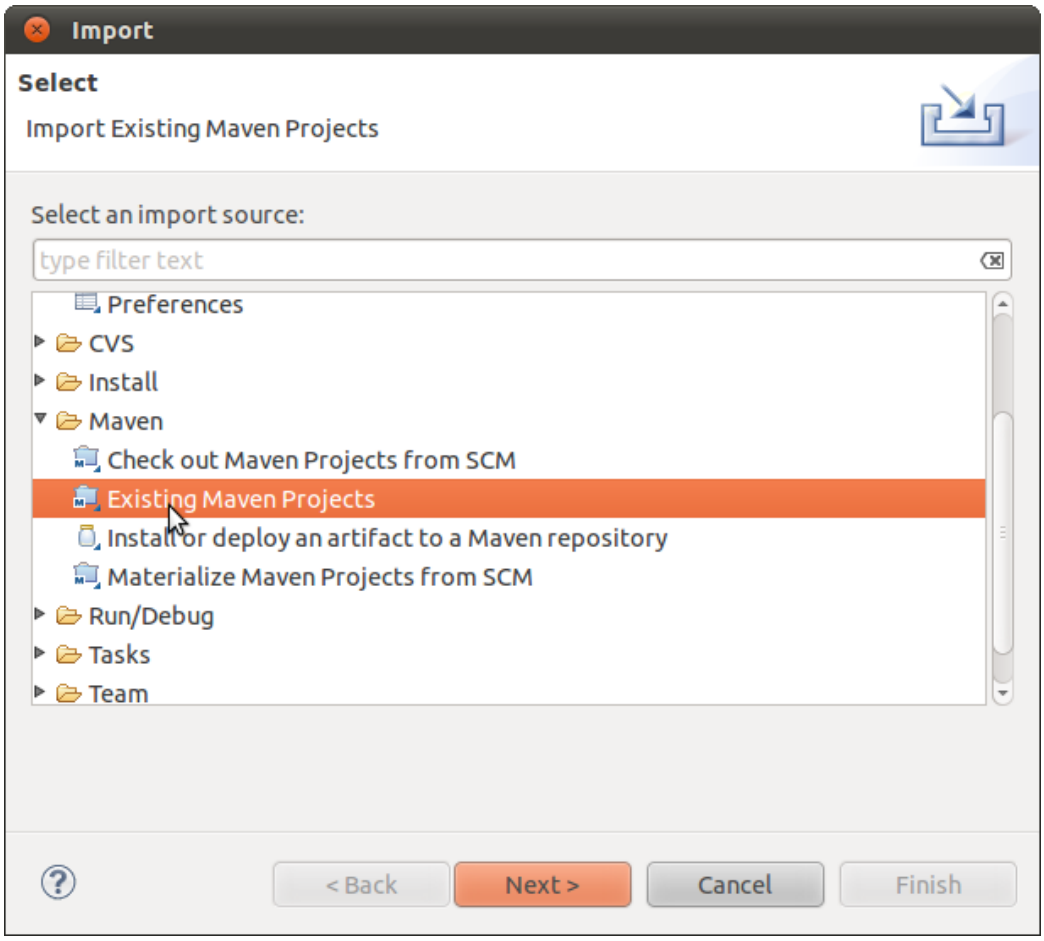
The Kuali Foundation official documentation for the LDAP Integration Module is located at https://wiki.kuali.org/x/FSyREg

1 Setup CAS for LDAP

Up until now, we have been using the `DummyLoginFilter` which is ok, but now that our users exist in CAS, we can no longer use it. Once CAS validates a users, KIM will try to find the user in the system. We will need to integrate both steps at once before the application will function again.

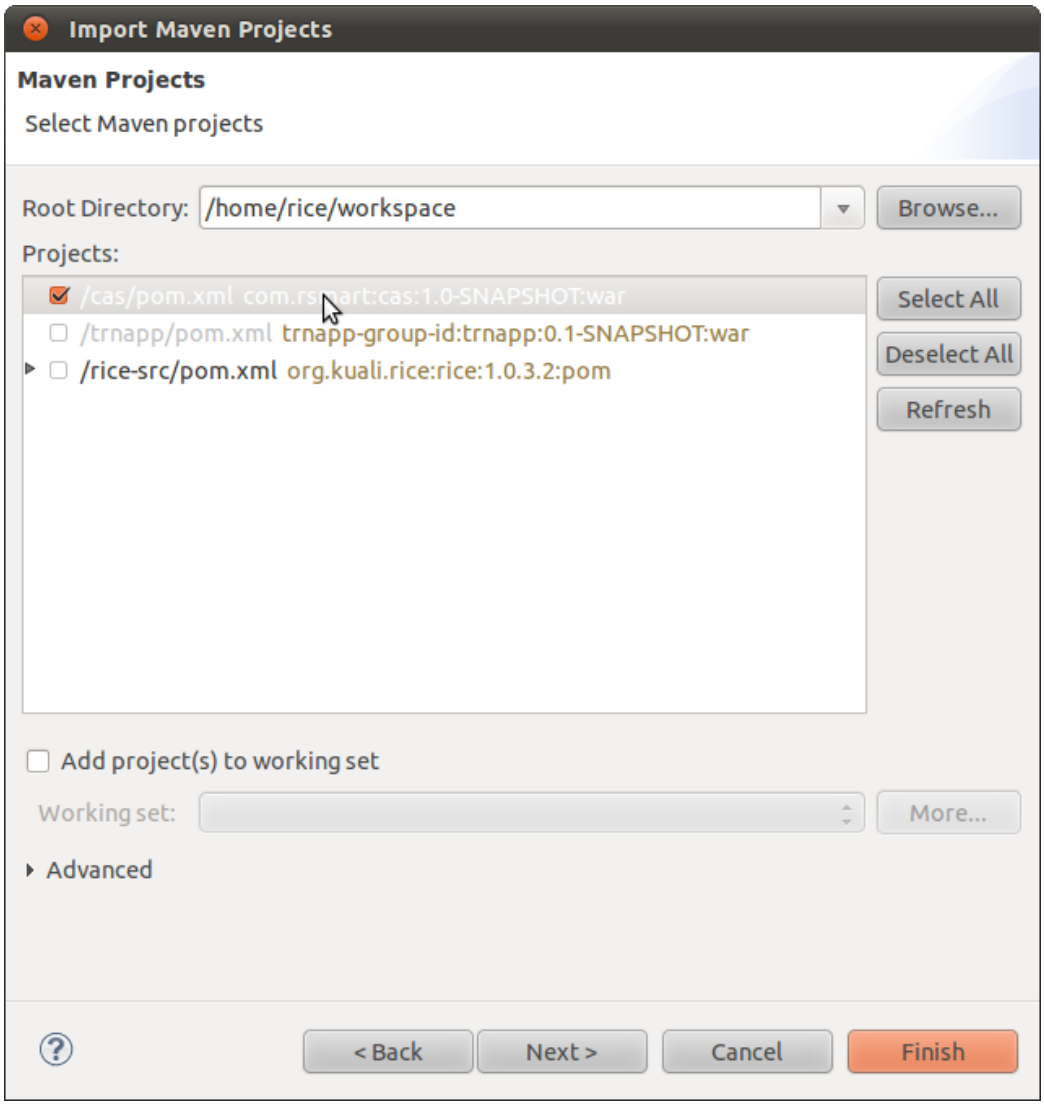
I have included in the workspace of the VM a **CAS** project. You will want to import this into Eclipse as a new Maven project.

1. File → Import → Other

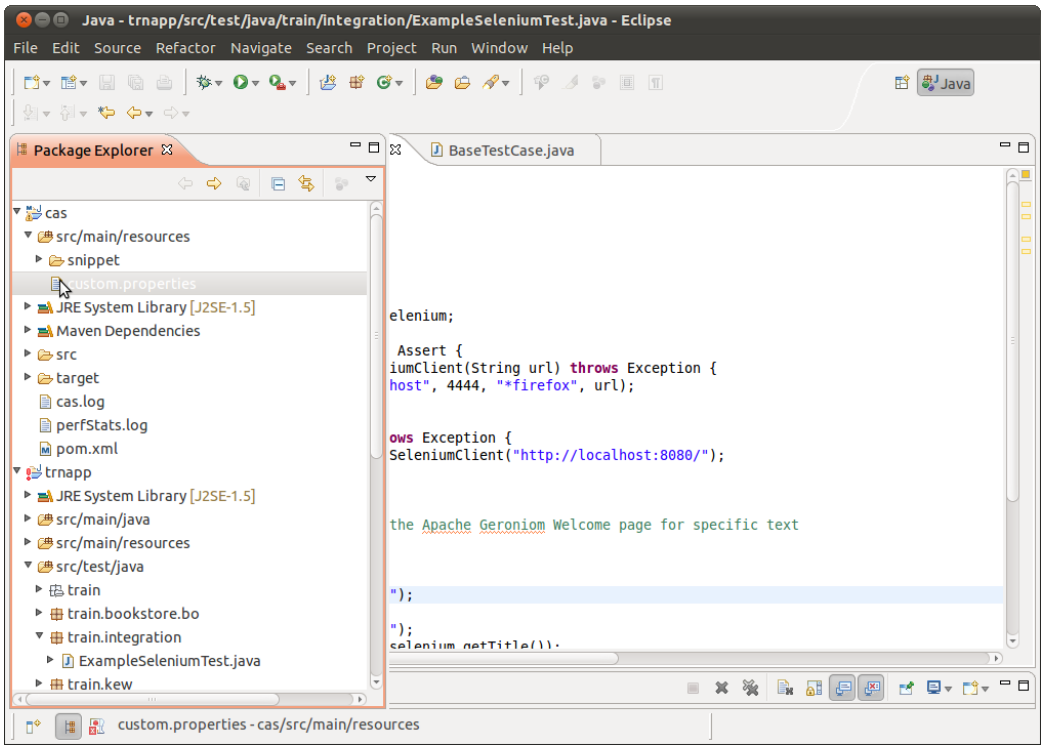


2. Click the **Browse** button

3. Select **cas** from `/home/rice/workspace`



4. Edit `src/main/resources/custom.properties`.



You want it to look like this.

Listing 55: src/main/resources/custom.properties

```
ldap.server.url=ldap://localhost
ldap.server.bind.username=cn=admin,dc=rsmart,dc=com
ldap.server.bind.password=rice
ldap.authentication.filter=eduPersonPrincipalName=%u,ou=people,dc=
    rsmart,dc=com
ldap.searchBase=ou=people,dc=rsmart,dc=com
```

5. Start it up with Maven

Listing 56: src/main/resources/custom.properties

```
mvn war:inplace tomcat:run
```

6. Configure the application for CAS by editing the `/home/rice/kuali/main/dev/trnapp-config.x`

Listing 57: src/main/resources/custom.properties

```
        <param name="rice.ldap.username">cn=admin,dc=rsmart,dc=com</
            param>
    <param name="rice.ldap.password">rice</param>
    <param name="rice.ldap.url">ldap://localhost</param>
```

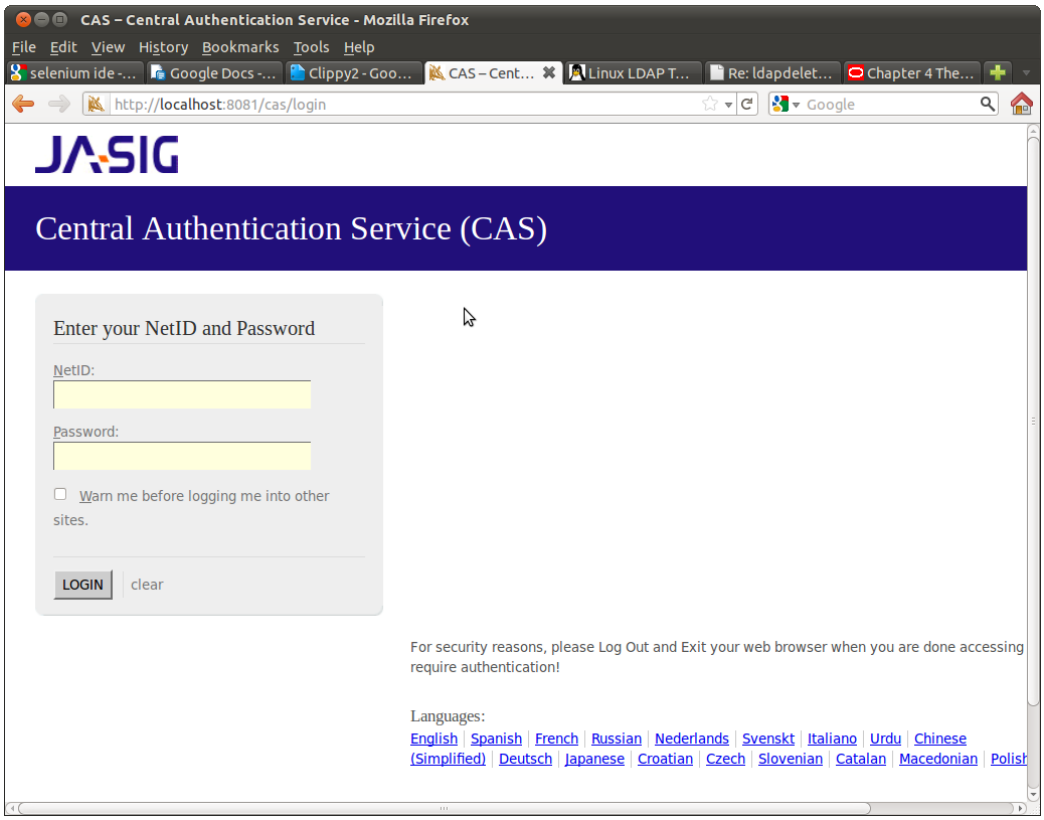
```
    <param name="rice.ldap.base">ou=people,dc=rsmart,dc=com</param>
        <param name="rice.additionalSpringFiles">org/kuali/rice/kim/
            config/KIMLdapSpringBeans.xml</param>

      <param name="cas.rice.server.name">${application.host}:8081</
          param>
      <param name="cas.url">http://${cas.rice.server.name}/${cas.
          context.name}</param>
      <param name="cas.require.https">false</param>
      <param name="cas.validate.password">false</param>
       <param name="filter.login.class">org.jasig.cas.client.
           authentication.AuthenticationFilter</param>
       <param name="filter.login.casServerLoginUrl">${cas.url}/login</
           param>
       <param name="filter.login.serverName">${application.host}:${
           http.port}</param>
       <param name="filtermapping.login.1">/*</param>

       <param name="filter.validation.class">org.jasig.cas.client.
           validation.Cas20ProxyReceivingTicketValidationFilter</param
           >
       <param name="filter.validation.casServerUrlPrefix">${cas.url}</
           param>
       <param name="filter.validation.serverName">${application.host}:
           ${http.port}</param>
       <param name="filtermapping.validation.2">/*</param>

       <param name="filter.caswrapper.class">org.jasig.cas.client.util
           .HttpServletRequestWrapperFilter</param>
       <param name="filtermapping.caswrapper.3">/*</param>
```

7. Browse to http://localhost:8081/cas/login



2 Add the Rice LDAP Integration Module

Add the LDAP Module to a Rice project and customize the mappers.

Goals

1. Attache the LDAP module to Rice 1.0.3 and build it
2. Learn to connect CAS to LDAP
3. Learn to debug and troubleshoot LDAP Queries
4. Learn to map between an LDAP database and KIM objects
5. Learn to modify mappings at all levels
6. Learn about caching in KIM

7. Learn about configuring Rice LDAP Integration

1 Implementation

1.1 Enable LDAP Integration

1. Switch to the `rice-src/ldap/` project
2. Browse to `src/main/config/example-config/`
3. Open the rice-config.xml
4. Copy the contents into `trnapp-config.xml`
5. Change the fields until they are like this:

Listing 58: trnapp-config.xml

```
<param name="rice.ldap.username">cn=admin,ou=people,dc=rsmart,dc=com</param>
<param name="rice.ldap.password">rice</param>
<param name="rice.ldap.url">ldap://localhost</param>
<param name="rice.ldap.base">ou=People,dc=rsmart,dc=edu</param>
<param
name="rice.additionalSpringFiles">org/kuali/rice/kim/config/KIMLdapSpringBeans.xml</param>
```

This will turn the LDAP Integration on and configure the connection.

1.2 Configure System Parameters

These system parameters are so vital, that the application will fail in most instances when not configured properly

KIM_TO_LDAP_FIELD_MAPPINGS Many different lookup for entities storied in LDAP can exist and they are not constrained to field name consistency. This means the field names can vary. In the event that field names do vary, then there cannot be a one-to-one mapping between KIM objects and LDAP objects. Instead, we have to have redundant mappings. This system parameter allows us to do just that.

When creating the System Parameters, use the following

Listing 59: trnapp-config.xml

```
entityId=uid;principalId=uid;principalName=eduPersonPrincipalName;
    givenName=sn;principals.principalName=eduPersonPrincipalName;
    persons.principalName=eduPersonPrincipalName;principals.
    principalId=uid;principals.active=employeeStatus;lastName=sn;
    firstName=givenName;employmentInformation.employeeStatus=
    employeeStatus;employmentInformation.employeeId=emplId,facultyId;
    names.lastName=sn;names.firstName=givenName;employmentInformation.
    employeeStatusCode=employeeStatus;
```

KIM_TO_LDAP_VALUE_MAPPINGS map possible values in the lookup form the results from an LDAP query. For example, employee active/inactive types have Y, N, and Both values. Each value needs to be mapped to a result from the LDAP query results.

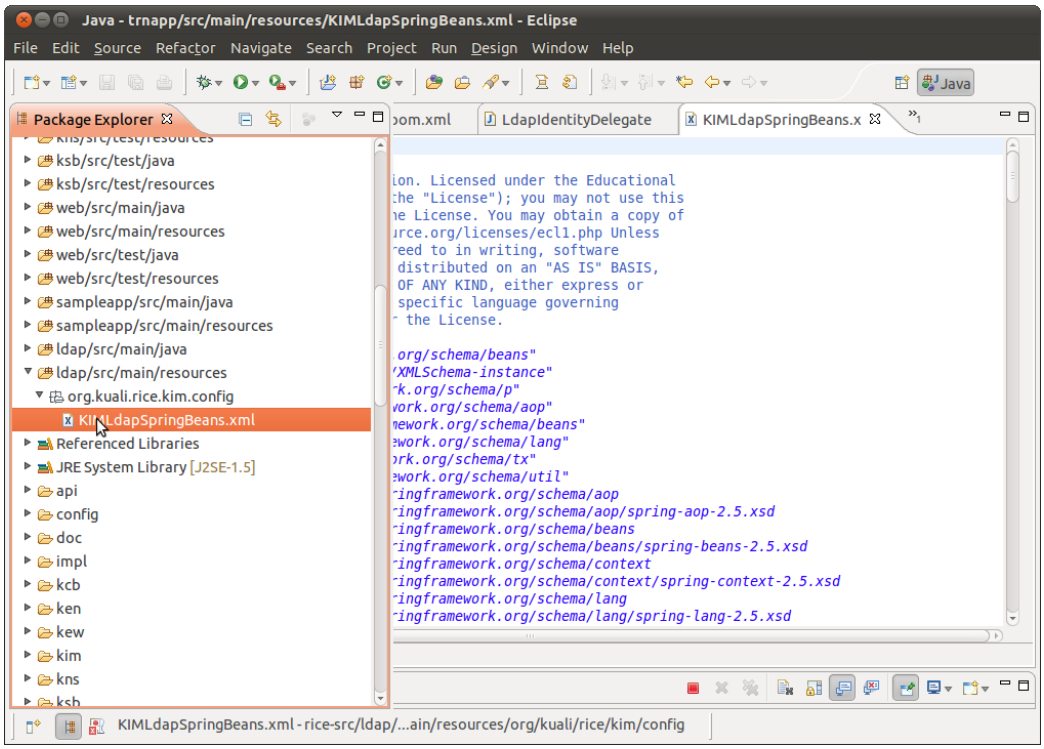
Listing 60: trnapp-config.xml

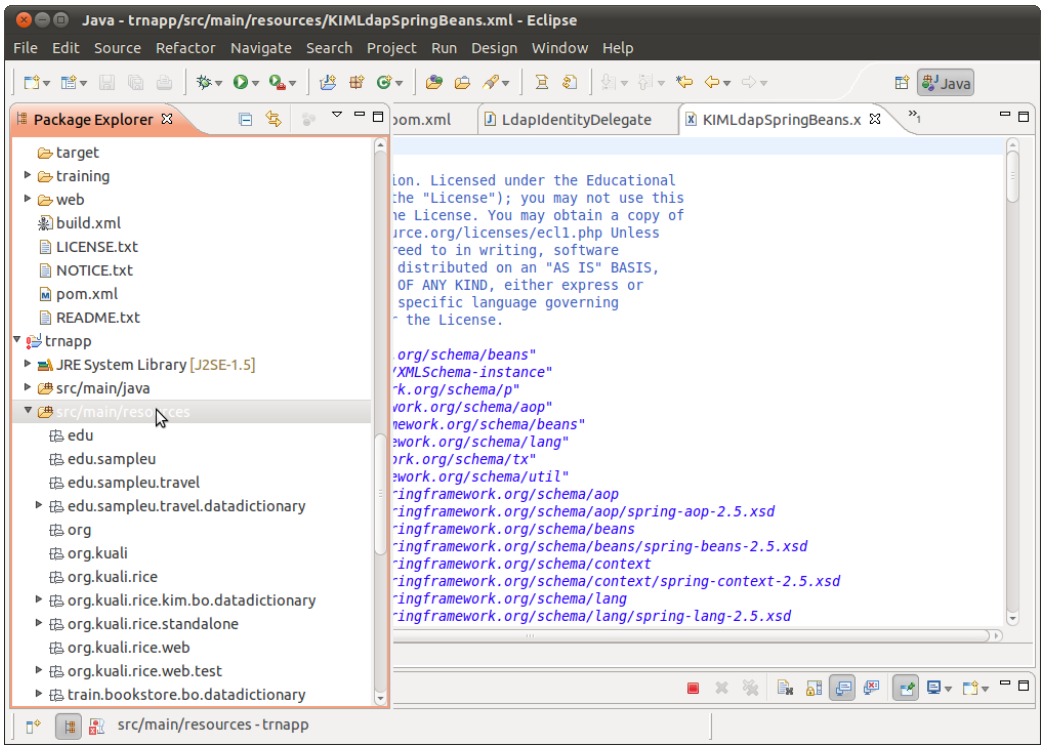
```
principals.active.Y=T;principals.active.N=!T;
```

1.3 Make KIMLdapSpringBeans.xml More Configurable

We want to be able to modify the KIMLdapSpringBeans.xml which is in the ldap module of the Rice project. We do not want to have to rebuild it each time we make a change to the Spring beans.

1. Copy **KIMLdapSpringBeans.xml** from **src/main/resources/org/kuali/rice/ldap/config** in the **ldap** module in **rice-src** to your **src/main/resources** in the **trnapp** project.


Java - trnapp/src/main/resources/KIMLdapSpringBeans.xml - Eclipse
File Edit Source Refactor Navigate Search Project Run Design Window Help
Java
Package Explorer
ksb/src/test/java
ksb/src/test/resources
web/src/main/java
web/src/main/resources
web/src/test/java
web/src/test/resources
sampleapp/src/main/java
sampleapp/src/main/resources
ldap/src/main/java
ldap/src/main/resources
org.kuali.rice.kim.config
KIMLdapSpringBeans.xml
Referenced Libraries
JRE System Library [J2SE-1.5]
api
config
doc
impl
kcb
ken
kew
kim
kns
LdapIdentityDelegate
KIMLdapSpringBeans.x
KIMLdapSpringBeans.xml - rice-src/ldap/...ain/resources/org/kuali/rice/kim/config



2. Edit the new **KIMLdapSpringBeans.xml** to look like:

Listing 61: trnapp/src/main/resources/KIMLdapSpringBeans.xml

```
<bean id="kimConstants" class="org.kuali.rice.kim.util.ConstantsImpl">
<property name="kimLdapIdProperty" value="${rice.ldap.param.kimLdapIdProperty}" />
<property name="kimLdapNameProperty" value="${rice.ldap.param.kimLdapNameProperty}" />
<property name="snLdapProperty" value="${rice.ldap.param.snLdapProperty}" />
<property name="givenNameLdapProperty" value="${rice.ldap.param.givenNameLdapProperty}" />
<property name="entityIdKimProperty" value="${rice.ldap.param.entityIdKimProperty}" />
<property name="employeeMailLdapProperty" value="${rice.ldap.param.employeeMailLdapProperty}" />
<property name="employeePhoneLdapProperty" value="${rice.ldap.param.employeePhoneLdapProperty}" />
<property name="defaultCountryCode" value="${rice.ldap.param.defaultCountryCode}" />
<property name="mappedParameterName" value="${mappedParameterName}" />
<property name="mappedValuesName" value="${rice.ldap.param.mappedValuesName}" />
```

```
        <property name="parameterNamespaceCode" value="${rice.ldap.
            param.parameterNamespaceCode}" />
        <property name="parameterDetailTypeCode"    value="${rice.ldap.
            param.parameterDetailTypeCode}" />
        <property name="personEntityTypeCode"    value="${rice.ldap.
            param.personEntityTypeCode}" />
        <property name="employeeIdProperty"      value="${rice.ldap.
            param.employeeIdPropertyd}" />
        <property name="departmentLdapProperty" value="${rice.ldap.
            param.departmentNumber}" />
        <property name="employeeTypeProperty"    value="${rice.ldap.
            param.employeeTypeProperty}" />
        <property name="employeeStatusProperty" value="${rice.ldap.
            param.employeeStatusProperty}" />
        <property name="affiliationLdapProperty"    value="${rice.ldap.
            param.affiliationLdapProperty}" />
        <property name="primaryAffiliationLdapProperty"    value="${
            rice.ldap.param.primaryAffiliationLdapProperty}" />
        <property name="defaultCampusCode"       value="${rice.ldap.
            param.defaultCampusCode}" />
        <property name="defaultChartCode"        value="${rice.ldap.
            param.defaultChartCode}" />
        <property name="taxExternalIdTypeCode"   value="${rice.ldap.
            param.taxExternalIdTypeCode}" />
        <property name="externalIdProperty"      value="${rice.ldap.
            param.externalIdProperty.externalId}" />
        <property name="externalIdTypeProperty" value="${rice.ldap.
            param.externalIdTypeProperty.externalIdentifierTypeCode}"
            />
        <property name="affiliationMappings"
    value="${rice.ldap.param.affiliationMappings}"
      />
        <property name="employeeAffiliationCodes"   value="${rice.ldap.
            param.employeeAffiliationCodes}" />
        </bean>
```

Notice that I prefixed the parameters with **rice.ldap.param**. This is the standard way to create a context for your parameters. Otherwise, it is unpolished and can be vulnerable to a name collision.

3. Create a new file called **ldap-config.xml** in your **trnapp** project. The file will be located in **src/main/resources/META-INF**

4. Copy the contents of **KIMLdapSpringBeans.xml** into it.

5. Then edit the file using search/replace until it looks like a standard XML Config. Here is what I had:

Listing 62: trnapp/src/main/resources/KIMLdapSpringBeans.xml

```
<config>
    <param name="rice.ldap.param.kimLdapIdProperty"              >uid</
        param>
```

```
        <param name="rice.ldap.param.kimLdapNameProperty"          >eduPersonPrincipalName</param>
        <param name="rice.ldap.param.snLdapProperty"               >sn</param>
        <param name="rice.ldap.param.givenNameLdapProperty"        >givenName</param>
        <param name="rice.ldap.param.entityIdKimProperty"          >entityId</param>
        <param name="rice.ldap.param.employeeMailLdapProperty"     >mail</param>
        <param name="rice.ldap.param.employeePhoneLdapProperty"    >employeePhone</param>
        <param name="rice.ldap.param.defaultCountryCode"           >1</param>
        <param name="rice.ldap.param.mappedParameterName"          >KIM_TO_LDAP_FIELD_MAPPINGS</param>
        <param name="rice.ldap.param.mappedValuesName"             >KIM_TO_LDAP_VALUE_MAPPINGS</param>
        <param name="rice.ldap.param.parameterNamespaceCode"       >KR-SYS</param>
        <param name="rice.ldap.param.parameterDetailTypeCode"      >All</param>
        <param name="rice.ldap.param.personEntityTypeCode"         >PERSON</param>
        <param name="rice.ldap.param.employeeIdProperty"           >emplId</param>
        <param name="rice.ldap.param.departmentLdapProperty"       >departmentNumber</param>
        <param name="rice.ldap.param.employeeTypeProperty"         >employeeType</param>
        <param name="rice.ldap.param.employeeStatusProperty"       >employeeStatus</param>
        <param name="rice.ldap.param.affiliationLdapProperty"      >affiliationProperty</param>
        <param name="rice.ldap.param.primaryAffiliationLdapProperty"
            >eduPersonPrimaryAffiliation</param>
        <param name="rice.ldap.param.defaultCampusCode"            >MC</param>
        <param name="rice.ldap.param.defaultChartCode"             >UA</param>
        <param name="rice.ldap.param.taxExternalIdTypeCode"        >TAX</param>
        <param name="rice.ldap.param.externalIdProperty"           >externalIdentifiers.externalId</param>
        <param name="rice.ldap.param.externalIdTypeProperty"       >externalIdentifiers.externalIdentifierTypeCode</param>
        <param name="rice.ldap.param.affiliationMappings"
    >staff=STAFF,faculty=FCLTY,employee=STAFF,student=STDNT,affilate=AFLT</param>
        <param name="rice.ldap.param.employeeAffiliationCodes"     >STAFF,FCLTY</param>
</config>
```

6. Add the **ldap-config.xml** to your **/home/rice/kuali/main/dev/trnapp-config.xml**

Listing 63: trnapp-config.xml

```
<param
name="config.location">classpath:META-INF/ldap-config.xml</param>
\lst{listing}
\item Since we are here, change the
  \textbf{rice.additionalSpringFiles} to refer to our new \textbf{
      KIMLdapSpringBeans.xml}

\begin{lstlisting}[numbers=left,language=xml,basicstyle=\scriptsize,
    backgroundcolor=\color{ubergray},caption={trnapp-config.xml},frame
    =single,breaklines=true]
        <param
    name="rice.additionalSpringFiles">KIMLdapSpringBeans.xml</param>
```

7. Now when we can modify our LDAP configuration from our **trnapp-config.xml**, and not worry about having to redistribute the rice-ldap.jar each time.

2 Integration

We have already configured our Rice LDAP Integration and implemented it, but what haven't done is actually integrate it into our Rice and distribute that yet. As it stands, we probably cannot start our application because of this missing dependency. We will now do the integration and set that up.

2.1 Add the ldap module to rice

1. Open the **rice-src/pom.xml**.
2. Locate the **modules** section
3. Add the `<module>ldap</module>`

Listing 64: rice-src/pom.xml

```
        <modules>
                <module>api</module>
                <module>impl</module>
                <module>ldap</module>
                <module>web</module>
                <module>sampleapp</module>
                <module>ksb</module>
                <module>kcb</module>
                <module>kns</module>
                <module>kim</module>
                <module>kew</module>
                <module>ken</module>
        </modules>
```

4. Now install the module. Execute the following from **rice-src**

Listing 65: Install the LDAP Integration Module

```
mvn clean install
```

2.2 Go

Now you can just start your application. Now you have two tomcat contexts you must start. One for your Rice application, and one for CAS.

1. Go to Run-¿Start CAS
2. Go to Run-¿Start Tomcat
3. Browse to http://localhost:8081/cas/login
4. You are now using LDAP Integration

3 Use Case for Adding Caching

Numerous LDAP Queries (even in a connection pooled environment) can be taxing on the system. Caching can solve this. There are different scenarios for solving the torrent of entity information with a cache. You can implement your own internal cache within your LDAP implementation, or you can utilize the builtin caching for KIM.

Notes

Day 4
Selenium Testing Plus Review from Days 2 and 3

Exercise 6
Download Selenium

Description

In this exercise, we download and install the **Compatility Reporter** plugin and the **Selenium IDE Plugin**.

Goals

- Find necessary plugins
- Install the Selenium IDE Plugin

Instructions

1 Install Compatibility Reporter Plugin

2 Install Selenium IDE Plugin

Exercise 7
Create a Selenium Test with Selenium IDE

Description

Record and save a test that can be replayed again and again using the Selenium IDE Plugin.

Goals

- Learn to record Selenium IDE

Instructions

Exercise 8
Export Selenium Test to Programmatic Selenium Unit Test with Web Driver

Description

Goals

Instructions

Notes

Day 5
External Web Services

Exercise 5

Notes