Abgabetermin: Fr. 16. Dez. 2016, 23:55 Uhr

Abgabe: Über Moodle, eine entsprechende Aufgabe ist dort eingerichtet.

Benennung der ZIP-Datei: Nachname_Vorname.zip. Darin sollen die jeweiligen Projekte als Ordner enthalten sein.

Erstellen Sie ein PDF Dokument welches den relevanten Sourcecode (z.B. Code für Validierung, Code für Generierung, XText Syntax) enthält (übersichtlich formatiert und nach einzelnen Aufgaben gegliedert) und entsprechende Testfälle (Screenshots).

Entwicklung einer DSL (24 Punkte + 6 Zusatzpunkte)

Überlegen Sie sich selbstständig eine Domänen-Spezifische Sprache (DSL). Die DSL sollte einen möglichst sinnvollen Einsatz ermöglichen. Überlegen Sie, ob sie ev. aus ihrem beruflichen Umfeld (falls vorhanden) eine Idee generieren können. Versuchen sie so weit als möglich die Konzepte von Ecore/XText einzusetzen.

Konkrete Aufgaben:

- Beschreiben Sie kurz den Zweck und das Einsatzgebiet ihrer DSL sowie die Kernkonzepte hinter Ihrer DSL in schriftlicher Form (ca ½ - 1 Seite).
- Entwickeln Sie für Ihre DSL eine passende textuelle Syntax mit Hilfe von XText. Dokumentieren Sie ihre Designentscheidungen (warum sieht Ihre textuelle Syntax so aus wie sie aussieht).
- Beschreiben Sie schriftlich Einschränkungen (Constraints) für Ihre DSL und begründen Sie, warum die jeweilige Einschränkung nicht (oder nicht einfach) im Metamodell bzw. mit XText direkt verwirklicht werden kann.
- Implementieren Sie mindestens jeweils 5 eigene, unterschiedliche Validierungen und dazu passende Quickfixes
- Implementieren sie geeignete Formatierungshilfen für Ihre DSL.
- Generieren sie für Ihre DSL passende Artefakte. Dies kann z.B. wie in der Übung Java Code sein. Lässt sich für Ihre DSL kein Code generieren, so kann z.B. auch eine Dokumentation als PDF (verwenden Sie eine Java PDF Bibliothek) oder ähnliches generiert werden. Dokumentieren Sie auch hier kurz, wie welches Element abgebildet wird und die Intention hinter den jeweiligen Regeln.
- Erstellen Sie auch ein Beispielmodell, das die Anwendung Ihrer textuellen Syntax zeigt. Zeigen Sie dabei auch mittels passender Screenshots die Validierungs- und Formatierungshilfen, die Sie eingebaut haben. Verwenden Sie das Beispielmodell auch zum Generieren des Codes.

Abzugeben ist die Dokumentation sowie der gesamte Source Code und das Beispielmodell als zip-File über Moodle.

Zusatzteil (6 Extrapunkte)

Entwickeln Sie für Ihre DSL auch eine grafische Syntax mit SIRIUS. Überlegen Sie passende Layer und Conditional Styles um die Anzeige entsprechend anpassen zu können. Überlegen Sie sich auch einen passenden zweiten Viewport (z.B. Tabellen Ansicht wie in der Übung). Dokumentieren Sie sämtliche Designentscheidungen wieder kurz. Erstellen Sie ein Beispielmodell, dass die Anwendung zeigt

1 Beschreibung

In dieser Übung wird eine *DSL* entwickelt, mit der ein Modul für eine bestehende Anwendung beschrieben werden kann. Mit dieser Beschreibung wird eine Maven-Projektstruktur sowie Java Quelltexte wie folgt aufgelistet erstellt:

- ***[module_key]****/message/pom.xml*
  Das Projekt der *Message Bundles*.
  - */message/../com.clevercure.**[module_key]**.message.**[bundle_name]**MessageBundle.java*
    Der Aufzählungsdatentyp mit den Schlüsseln der sprachspezifischen Texte.
  - */message/../resources/META-INF/resource-bundles/**[bundle_name]**_**[locale]**.properties*
    Die *Locale*-spezifische *Properties*-Datei mit den sprachspezifischen Texten.
- ***[module_key]****/model/pom.xml*
  Parent für alle Modell Projekte.
- ***[module_key]****/model/jpa/pom.xml*
  Das Projekt der *JPA*-Modells.
  - */jpa/../com.clevercure.**[module_key]**.jpa.observer.Observer.java*
    Die *Observer*-Klasse, die auf die definierten *Events* reagiert.
  - */jpa/../resources/META-INF/**[module_key]**Orm.xml*
    Die *JPA 2.2 orm.xml* Konfigurationsdatei.
  - */jpa/../resources/META-INF/beans.xml*
    Die *CDI 1.1* Konfigurationsdatei, die *CDI* für dieses Artefakt aktiviert.
- ***[module_key]****/service/pom.xml*
  Parent für alle Service Projekte.
- ***[module_key]****/service/api/pom.xml*
  Das Projekt mit der Service Spezifikation
- ***[module_key]****/service/impl/pom.xml*
  Das Projekt mit der Service Implementierung.
  - */service/impl/../com.clevercure.**[module_key]**.service.impl.observer.Observer.java*
    Die *Observer*-Klasse, die auf die definierten *Events* reagiert.
  - */service/impl/../resources/META-INF/beans.xml*
    Die *CDI 1.1* Konfigurationsdatei, die *CDI* für dieses Artefakt aktiviert.

Es können folgende Aspekte eines Moduls beschrieben werden:

- ***MessageBundles*** sind Beschreibungen von Klassen, die sprachspezifische Texte für einen Schlüssel und eine *Locale* verwalten.
- ***Observers*** sind Beschreibungen von Beobachtermethoden, die mittels einen *Delegate* auf einem definierten *CDI*-Event reagieren können.
- ***JpaConfig*** ist die Beschreibung des *JPA*-Projekts, dem *Observer* und *MessageBundles* hinzugefügt werden können.
- ***ServiceConfig*** ist die Beschreibung der *Service*-Projekte, dem *Observer* und *MessageBundles* hinzugefügt werden können.

Das Ziel dieser *DSL* ist es den initialen Aufwand beim Erstellen eines Moduls für eine bestehende Anwendung zu erleichtern. Ein Module besteht aus mehreren *Maven*-Projekten und Konfigurationsdateien, die Mühsam erstellt werden müssen. Mit dieser *DSL* kann ein Modul einfach beschrieben werden und daraus einen *Maven*-Projektstruktur, mit Konfigurationsdateien und Java Quelltexten zu erstellen.

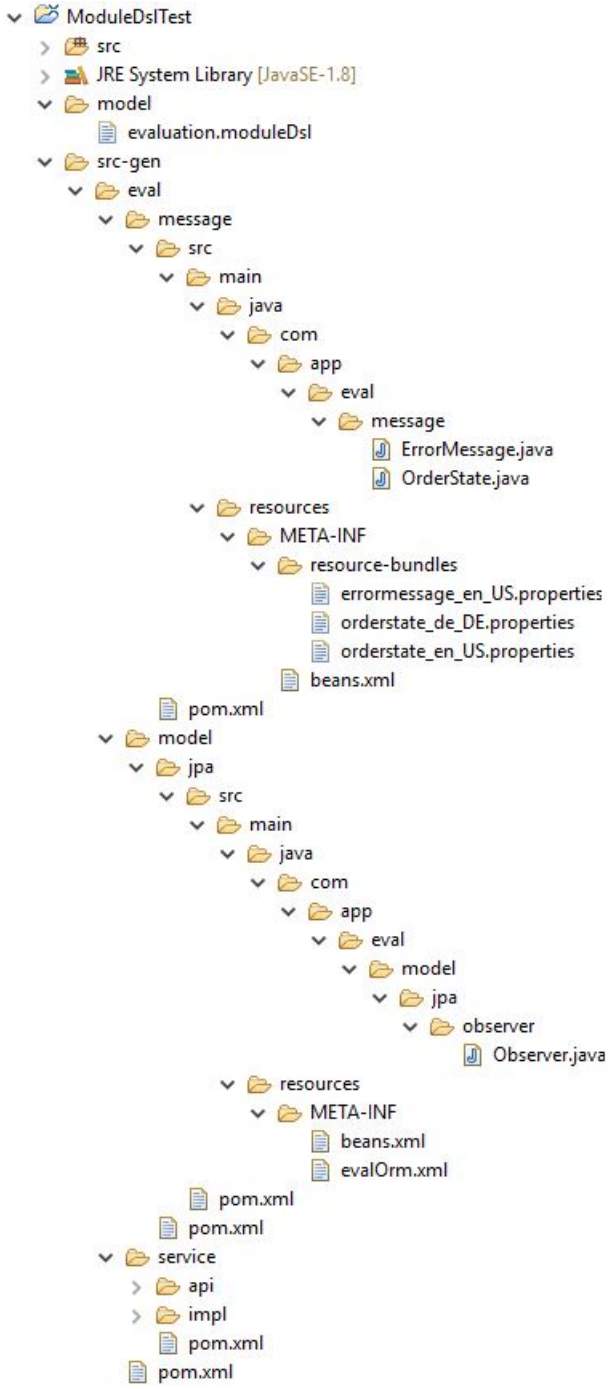

Abbildung 1: Testprojekt für das Modul mit generierten Ressourcen

2 XTEXT Grammatik

Listing 1: ProjectGeneratorDsl.xtext

```
grammar at.ooe.fh.mdm.herzog.dsl.proj.ProjectGenerator with org.eclipse.xtext.common.Terminals

generate projectGenerator "http://www.ooe.at/fh/mdm/herzog/dsl/proj/ProjectGenerator"

Module:
    'module' name=ID '{'
    'key' key=STRING';'
    'cdiEnabled' cdiEnabled=Boolean';'
    ('messageBundles' '{' (messageBundles+=Localized+) '}' ';')?
    ('observers' '{' (observers+=Observer+) '}' ';')?
    'jpaConfig' jpaConfig=JpaConfig';'
    'serviceConfig' serviceConfig=ServiceConfig';'
    '}';

// Service configuration
ServiceConfig: '{'
    ('observers' '{' (observers+=[Observer]+) '}' ';')?
    ('messageBundles' '{' (messageBundles+=[Localized]+) '}' ';')
    '}';

Observer:
    name=ID '{'
    'type' type=CLASSNAME';'
    'during' during=During';'
    'notifyObserver' notify=Notify';'
    'delegate' className=CLASSNAME';'
    ('qualifier' qualifier=CLASSNAME';')?
    '}'';';

// Jpa configuration
JpaConfig: '{'
    'localizedEnums' '{' (localizedEnums+=[Localized]+) '}'';'
    ('observers' '{' (observers+=[Observer]+) '}' ';')?
    '}';

// Common
Localized:
    name=ID '{'
    ('values' '{' (values+=LocalizedEntry+) '}' ';')?
    '}'';';

LocalizedEntry: '{'
    'key' localizedKey=LOCALIZEDKEY';'
    'values' '{' (values+=LocalizedValue+) '}' ';'
    ('args' '{' (args+=STRING+) '}' ';')?
    '}'';';

LocalizedValue: '{'
    'locale' locale=Locale';'
    'value' value=STRING';'
    '}'';';

// Constants
enum Locale: DE_DE='de_DE' | EN_US='en_US';
enum Boolean: TRUE='true' | FALSE='false';
enum During: IN_PROG='InProgress' | AFTER_COMPLETION='AfterCompleition';
enum Notify: ALWAYS='Always' | EXISTS='Exists';
terminal fragment UAZ: 'A'..'Z';
terminal fragment LAZ: 'a'..'z';
```

```
60 terminal fragment UAZN: 'A'..'Z'|'0'..'9';
61 terminal CLASSNAME: (LAZ+ '.')+ (UAZ LAZ*)+;
62 terminal LOCALIZEDKEY: UAZN+ ('_' UAZN)*;
```

2.1 Regeln

Die gesamte Konfiguration befindet sich im Wurzelobjekt des Typ Modul, das ein Modul beschreibt. Objekte, die in mehreren Objekten referenziert werden können, werden auf Ebene des Moduls einmalig beschrieben und dann von anderen Objekten referenziert *(Observer, Localized)*. Sofern ein Name benötigt wird, wurde das Attribut Name den Regeln, die Objekte beschreiben, hinzugefügt, wobei der Name ebenfalls als Schlüssel für diese Objekte fungiert, damit diese Objekte referenziert werden können. Die einzelnen Regeln bilden Objekte, dessen *Java*-Klassenbibliotheken bei der Validierung, Quickfixes und Generatoren verwendet werden.

2.2 Konstanten

Die Konstanten, wie die unterstützten *Locale*, boolsche Zustände und *Observer* spezifische Konstanten wurden als Aufzählungen abgebildet. Es wurde darauf geachtet, das über die Zeichenkettenrepräsentation der Aufzählungen sich leicht die *Java*-Datentypen erstellen lassen.

2.3 Terminal Regeln

Die Terminal Regeln wurden eingeführt, damit die Klassennamen für die *Observer*-Beschreibungen einen gültigen voll qualifizierten *Java*-Klassennamen darstellen und damit die Schlüssel der sprachspezifischen Einträge einer *Localized*-Beschreibung, der Konvention von Schlüsseln einer *Properties*-Datei folgen.

3 Constraint

Folgende Auflistung beschreibt die implementierten Validierungen:

- ***Leere Beschreibung***: Es wurde eine Validierung eingeführt, die auf eine leere Beschreibung eines Moduls prüft, damit ein *Quickfix* eine initiale Beschreibung erstellen kann.
- ***Duplikate Namen***: Da die Namen vom Datentyp *ID* sind, wurde zusätzlich eine Prüfung eingeführt, ob die Namen bereits vergeben wurden. *(Observer, Localized)*
- ***Camel case Namen***: Da die Namen vom Datentyp *ID* sind, wurde zusätzlich eine Prüfung eingeführt, ob die Namen in *camel case* sind.
- ***Duplikate LocaleEntry Einträge***: Es wurde eine Validierung eingeführt, die prüft ob es Duplikate sprachspezifischen Einträge mit demselben Schlüssel gibt.
- ***Doppelte Verwendung von* Locale**: Es wurde eine Validierung eingeführt, die prüft ob eine *Locale* innerhalb eines *Bundles* definiert wurde.
- ***Duplikate Referenzen***: Es wurde eine Validierung eingeführt, die prüft ob es Duplikate bei den gesetzten Referenzen gibt.
- ***Ungenutzte Objekte***: Es wurde eine Validierung eingeführt, die prüft Objekte wie *Observer* oder *Localized* zwar definiert wurden aber nirgends referenziert werden.
- ***Doppelte Objekt Ids***: Es wurde eine Validierung eingeführt, die prüft ob Objekt Ids doppelt vergeben wurden *(Observer, Localized)* Namen.

Mit einer Grammatik kann nicht beschrieben werden, dass es keine Duplikate bei den verwendeten Namen von Objekten in einer Auflistung gibt. Das ist so, da es sich hierum Semantik handelt und nicht Grammatik, ob ein Name nur einmalig oder mehrmalig vergeben werden darf.

Wenn das Attribut Name bereits als *ID* festgesetzt wurde, dann kann nicht zusätzlich eine Terminalregel angewendet werden, die sicherstellt, das der Name auch z.B.: in *camel case* ist.

Wenn eine Auflistung definiert wurde wie *(localizedEnums+=[Localized]+)*, dann kann mit der Grammatik nicht verhindert werden, das Duplikate bei den Referenzen angeben werden, da es sich hier auch um Semantik handelt. Das Attribut *localizedEnums* wird als Liste abgebildet.

Das ein *Localized* für eine *Locale* nur einmal einen Eintrag definieren kann, ist eine Semantik, die auch nicht über die Grammatik abgebildet werden kann.

4 Validation und Quickfix

Listing 2: ProjectGeneratorValidator.xtend

```
/*
 * generated by Xtext 2.10.0
 */
package at.ooe.fh.mdm.herzog.dsl.proj.validation

import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.JpaConfig
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Localized
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Module
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Observer
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.ProjectGeneratorPackage
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.ServiceConfig
import java.util.Objects
import java.util.regex.Pattern
import org.eclipse.xtext.validation.Check

import static java.util.stream.Collectors.*
import java.util.List
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.LocalizedEntry

/**
 * This class contains custom validation rules.
 *
 * See https://www.eclipse.org/Xtext/documentation/303_runtime_concepts.html#validation
 */
class ProjectGeneratorValidator extends AbstractProjectGeneratorValidator {

  /**
   * The Class holding the validator ids.
   */
  public static class ValidatorId {
    // Module
    public static final String MODULE_EMPTY = "MODULE_EMPTY";
    public static final String MODULE_NAME_CAMEL_CASE = "MODULE_NAME_CAMEL_CASE";
    public static final String MODULE_KEY_UPPER_CASE = "MODULE_KEY_UPPER_CASE";
    public static final String MODULE_OBSERVER_UNUSED = "MODULE_OBSERVER_UNUSED";
    public static final String MODULE_LOCALIZED_UNUSED = "MODULE_LOCALIZED_UNUSED";
    // Observer
    public static final String OBSERVER_NAME_CAMEL_CASE = "OBSERVER_NAME_CAMEL_CASE";
    public static final String OBSERVER_NAME_DUPLICATE = "OBSERVER_NAME_UNIQUE";
    // Localized
    public static final String LOCALIZED_NAME_CAMEL_CASE = "LOCALIZED_NAME_CAMEL_CASE";
    public static final String LOCALIZED_NAME_DUPLICATE = "LOCALIZED_NAME_UNIQUE";
    public static final String LOCALIZED_ENTRY_DUPLICATE = "LOCALIZED_ENTRY_DUPLICATE";
    public static final String LOCALIZED_ENTRY_LOCALE_DUPLICATE =
 ↪   "LOCALIZED_ENTRY_LOCALE_DUPLICATE";
    public static final String LOCALIZED_ENTRY_UNDEFINED = "LOCALIZED_ENTRY_UNDEFINED";
    // ServiceConfig
    public static final String SERVICE_CONFIG_MESSAGE_BUNDLE_DUPLICATE =
 ↪   "SERVICE_CONFIG_MESSAGE_BUNDLE_DUPLICATE";
    public static final String SERVICE_CONFIG_OBSERVERS_DUPLICATE =
 ↪   "SERVICE_CONFIG_OBSERVERS_DUPLICATE";
    // JpaConfig
    public static final String JPA_LOCALIZED_ENUMS_DUPLICATE = "JPA_LOCALIZED_ENUMS_DUPLICATE";
    public static final String JPA_OBSERVERS_DUPLICATE = "JPA_OBSERVERS_DUPLICATE";
  }

  static Pattern CAMEL_CASE_PATTERN = Pattern.compile("([A-Z]{1}[a-z0-9]+)+");

  @Check
```

```
57   def checkForEmptyModule(Module module) {
58     if (module.key == null && module.messageBundles.empty && module.observers.empty &&
     ↪  module.jpaConfig == null &&
59       module.serviceConfig == null) {
60       val errorMsg = "Module is empty";
61       error(errorMsg, ProjectGeneratorPackage.Literals.MODULE__NAME, ValidatorId.MODULE_EMPTY);
62     }
63   }
64
65   // Module: Name must be camel case
66   @Check
67   def checkForCamelCaseModuleName(Module _module) {
68     if (!CAMEL_CASE_PATTERN.matcher(_module.name).matches) {
69       val errorMsg = "Module name must be a camel case string (e.g.: MyModuleName)";
70       error(errorMsg, ProjectGeneratorPackage.Literals.MODULE__NAME,
     ↪  ValidatorId.MODULE_NAME_CAMEL_CASE);
71     }
72   }
73
74   // Module: Key must be upper case
75   @Check
76   def checkForUpperCaseModuleKey(Module _module) {
77     for (Character c : _module.key.toCharArray) {
78       if (Character.isLowerCase(c)) {
79         val errorMsg = "Module key must be upper case";
80         error(errorMsg, ProjectGeneratorPackage.Literals.MODULE__KEY,
     ↪  ValidatorId.MODULE_KEY_UPPER_CASE);
81         return;
82       }
83     }
84   }
85
86   // Module: Observer unused
87   @Check
88   def checkForUnusedObserver(Module _module) {
89     val List<Observer> observers = newArrayList(_module.observers);
90     if (_module.serviceConfig != null) {
91       observers.removeAll(_module.serviceConfig.observers);
92     }
93     if (_module.jpaConfig != null) {
94       observers.removeAll(_module.jpaConfig.observers);
95     }
96     if (!observers.isEmpty) {
97       val warnMsg = "Some observers are unused";
98       warning(warnMsg, ProjectGeneratorPackage.Literals.MODULE__OBSERVERS,
     ↪  ValidatorId.MODULE_OBSERVER_UNUSED,
99         observers.stream.map[name].toArray(size|newArrayOfSize(size)));
100    }
101  }
102
103  // Module: Localized unused
104  @Check
105  def checkForUnusedMessageBundle(Module _module) {
106    val List<Localized> localized = newArrayList(_module.messageBundles);
107    if (_module.serviceConfig != null) {
108      localized.removeAll(_module.serviceConfig.messageBundles);
109    }
110    if (_module.jpaConfig != null) {
111      localized.removeAll(_module.jpaConfig.localizedEnums);
112    }
113    if (!localized.isEmpty) {
114      val warnMsg = "Some message bundles are unused";
115      warning(warnMsg, ProjectGeneratorPackage.Literals.MODULE__MESSAGE_BUNDLES,
     ↪  ValidatorId.MODULE_LOCALIZED_UNUSED,
```

```
116         localized.stream.map[name].toArray(size|newArrayOfSize(size)));
117       }
118     }
119
120     // Localized: Name must be camel case
121     @Check
122     def checkForCamelCaseLocalizedName(Localized _localized) {
123       if (!CAMEL_CASE_PATTERN.matcher(_localized.name).matches) {
124         val errorMsg = "Localized name must be a camel case string (e.g.: MyLocalizedName)";
125         error(errorMsg, ProjectGeneratorPackage.Literals.LOCALIZED__NAME,
    ↪  ValidatorId.LOCALIZED_NAME_CAMEL_CASE);
126       }
127     }
128
129     // Localized: Name must be unique
130     @Check
131     def checkForUniqueLocalizedName(Localized _localized) {
132       val module = _localized.eContainer as Module;
133       Objects.requireNonNull(module, "eContainer should be an instance of Module");
134       val count = module.messageBundles.stream.filter[name.equals(_localized.name)].distinct.count;
135       if (count > 1) {
136         val errorMsg = "Localized name is used by '" + (count - 1) + "' other Localized instances";
137         error(errorMsg, ProjectGeneratorPackage.Literals.LOCALIZED__NAME,
    ↪  ValidatorId.LOCALIZED_ENTRY_DUPLICATE);
138       }
139     }
140
141     // Localized: must contain at least one localized entry
142     @Check
143     def checkForDefinedLocaleEntries(Localized localized) {
144       if (localized.values.empty) {
145         error("If attribute 'values' is defined, then at least one localized values must be given",
146           ProjectGeneratorPackage.Literals.LOCALIZED__VALUES,
    ↪  ValidatorId.LOCALIZED_ENTRY_UNDEFINED);
147       }
148     }
149
150     // LocalizedEntry: Duplicate locale entries for a localized key not allowed
151     @Check
152     def checkForDuplicateLocaleEntries(Localized _localized) {
153       val duplicateEntries =
    ↪  _localized.values.stream.collect(groupingBy[localizedKey]).entrySet.stream.filter [
154         value.size > 1
155       ].map[key].distinct.collect(toList);
156
157       if (!duplicateEntries.empty) {
158         val errorMsg = "Duplicate locale entries found. " +
    ↪  duplicateEntries.stream.collect(joining(",", "[", "]"));
159         error(errorMsg, ProjectGeneratorPackage.Literals.LOCALIZED__VALUES,
    ↪  ValidatorId.LOCALIZED_ENTRY_DUPLICATE,
160           duplicateEntries.toArray(newArrayOfSize(duplicateEntries.size)));
161       }
162     }
163
164     // LocalizedEntry: Duplicate locale entries for a locale not allowed
165     @Check
166     def checkForDuplicateLocaleEntriesLocales(LocalizedEntry _localized) {
167       val duplicateLocales =
    ↪  _localized.values.stream.collect(groupingBy[locale.toString]).entrySet.stream.filter [
168         value.size > 1
169       ].map[key].distinct.collect(toList);
170
171       if (!duplicateLocales.isEmpty) {
```

```
172       val errorMsg = "Duplicate locale found. " + duplicateLocales.stream.collect(joining(",",
    ↪  "[", "]"));
173       error(errorMsg, ProjectGeneratorPackage.Literals.LOCALIZED_ENTRY__VALUES,
    ↪  ValidatorId.LOCALIZED_ENTRY_LOCALE_DUPLICATE,
174         duplicateLocales.toArray(newArrayOfSize(duplicateLocales.size)));
175     }
176   }
177
178
179   // Observer: Name must be camel case
180   @Check
181   def checkForCamelCaseObserverName(Observer _observer) {
182     if (!CAMEL_CASE_PATTERN.matcher(_observer.name).matches) {
183       val errorMsg = "Observer name must be a camel case string (e.g.: MyObserverName)";
184       error(errorMsg, ProjectGeneratorPackage.Literals.OBSERVER__NAME,
    ↪  ValidatorId.OBSERVER_NAME_CAMEL_CASE);
185     }
186   }
187
188   // Observer: Name must be unique
189   @Check
190   def checkForUniqueObserverName(Observer _observer) {
191     val module = _observer.eContainer as Module;
192     Objects.requireNonNull(module, "eContainer should be an instance of Module");
193     val count = module.observers.stream.filter[name.equals(_observer.name)].distinct.count;
194     if (count > 1) {
195       val errorMsg = "Observer name is used by '" + (count - 1) + "' other Observer instances";
196       error(errorMsg, ProjectGeneratorPackage.Literals.OBSERVER__NAME,
    ↪  ValidatorId.OBSERVER_NAME_DUPLICATE);
197     }
198   }
199
200   // ServiceConfig: Message Bundles must be unique
201   @Check
202   def checkForUniqueServiceConfigMessageBundles(ServiceConfig _serviceConfig) {
203     val duplciateBundles =
    ↪  _serviceConfig.messageBundles.stream.collect(groupingBy[name]).entrySet.stream.filter [
204       value.size > 1
205     ].map[key].distinct.collect(toList);
206     if (!duplciateBundles.empty) {
207       val errorMsg = "Duplicate message bundles found. " +
208         duplciateBundles.stream.collect(joining(",", "[", "]"));
209       error(errorMsg, ProjectGeneratorPackage.Literals.SERVICE_CONFIG__MESSAGE_BUNDLES,
210         ValidatorId.SERVICE_CONFIG_MESSAGE_BUNDLE_DUPLICATE,
211         duplciateBundles.toArray(newArrayOfSize(duplciateBundles.size)));
212     }
213   }
214
215   // ServiceConfig: Observers must be unique
216   @Check
217   def checkForUniqueServiceConfigObservers(ServiceConfig _serviceConfig) {
218     val duplciateBundles =
    ↪  _serviceConfig.observers.stream.collect(groupingBy[name]).entrySet.stream.filter [
219       value.size > 1
220     ].map[key].distinct.collect(toList);
221     if (!duplciateBundles.empty) {
222       val errorMsg = "Duplicate observers found. " +
223         duplciateBundles.stream.collect(joining(",", "[", "]"));
224       error(errorMsg, ProjectGeneratorPackage.Literals.SERVICE_CONFIG__OBSERVERS,
225         ValidatorId.SERVICE_CONFIG_OBSERVERS_DUPLICATE,
226         duplciateBundles.toArray(newArrayOfSize(duplciateBundles.size)));
227     }
228   }
```

```
229
230   // JpaConfig: Message Bundles must be unique
231   @Check
232   def checkForUniqueJpaConfigMessagebundles(JpaConfig _jpaConfig) {
233     val duplciateBundles =
      ↪  _jpaConfig.localizedEnums.stream.collect(groupingBy[name]).entrySet.stream.filter [
234       value.size > 1
235     ].map[key].distinct.collect(toList);
236     if (!duplciateBundles.empty) {
237       val errorMsg = "Duplicate localized enums found. " +
238         duplciateBundles.stream.collect(joining(",", "[", "]"));
239       error(errorMsg, ProjectGeneratorPackage.Literals.JPA_CONFIG__LOCALIZED_ENUMS,
240         ValidatorId.JPA_LOCALIZED_ENUMS_DUPLICATE,
241         duplciateBundles.toArray(newArrayOfSize(duplciateBundles.size)));
242     }
243   }
244
245   // JpaConfig: Name must be unique
246   @Check
247   def checkForUniqueJpaConfigObserverName(JpaConfig _jpaConfig) {
248     val duplciateBundles =
      ↪  _jpaConfig.observers.stream.collect(groupingBy[name]).entrySet.stream.filter [
249       value.size > 1
250     ].map[key].distinct.collect(toList);
251     if (!duplciateBundles.empty) {
252       val errorMsg = "Duplicate observers found. " +
253         duplciateBundles.stream.collect(joining(",", "[", "]"));
254       error(errorMsg, ProjectGeneratorPackage.Literals.JPA_CONFIG__OBSERVERS,
255         ValidatorId.JPA_OBSERVERS_DUPLICATE,
256         duplciateBundles.toArray(newArrayOfSize(duplciateBundles.size)));
257     }
258   }
259 }
```

Listing 3: ProjectGeneratorQuickfixProvider.xtend

```
/*
 * generated by Xtext 2.10.0
 */
package at.ooe.fh.mdm.herzog.dsl.proj.ui.quickfix

import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Boolean
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.During
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.JpaConfig
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Locale
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Localized
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Module
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Notify
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Observer
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.ProjectGeneratorFactory
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.ServiceConfig
import at.ooe.fh.mdm.herzog.dsl.proj.validation.ProjectGeneratorValidator.ValidatorId
import java.util.Objects
import java.util.stream.Collectors
import org.eclipse.xtext.ui.editor.quickfix.DefaultQuickfixProvider
import org.eclipse.xtext.ui.editor.quickfix.Fix
import org.eclipse.xtext.ui.editor.quickfix.IssueResolutionAcceptor
import org.eclipse.xtext.validation.Issue
import static java.util.stream.Collectors.*
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.LocalizedEntry

/**
 * Custom quickfixes.
 *
 * See https://www.eclipse.org/Xtext/documentation/310_eclipse_support.html#quick-fixes
 */
class ProjectGeneratorQuickfixProvider extends DefaultQuickfixProvider {

  /**
   * Generates a default structure.
   */
  @Fix(ValidatorId.MODULE_EMPTY)
  def fixModuleInitGeneration(Issue issue, IssueResolutionAcceptor acceptor) {
    acceptor.accept(issue, 'Initialize default', 'Initialize default', '') [ element, context |
      val module = (element as Module);
      module.key = module.name.toUpperCase;
      module.cdiEnabled = Boolean.TRUE;

      // Default jpa Localized
      val localizedJpa = ProjectGeneratorFactory.eINSTANCE.createLocalized;
      localizedJpa.name = "JpaConstantBundle";
      val localizedJpaEntry = ProjectGeneratorFactory.eINSTANCE.createLocalizedEntry;
      localizedJpaEntry.localizedKey = "KEY_1";
      val localizedJpaValue = ProjectGeneratorFactory.eINSTANCE.createLocalizedValue;
      localizedJpaValue.locale = Locale.EN_US;
      localizedJpaValue.value = "EN_KEY_1";

      localizedJpaEntry.values.add(localizedJpaValue);
      localizedJpa.values.add(localizedJpaEntry);

      // Default service Localized
      val localizedError = ProjectGeneratorFactory.eINSTANCE.createLocalized;
      localizedError.name = "ErrorMessagebundle";
      val localizedErrorEntry = ProjectGeneratorFactory.eINSTANCE.createLocalizedEntry;
      localizedErrorEntry.localizedKey = "ERROR_1";
      val localizedErrorValue = ProjectGeneratorFactory.eINSTANCE.createLocalizedValue;
```

```
61        localizedErrorValue.locale = Locale.EN_US;
62        localizedErrorValue.value = "EN_ERROR_1";
63
64        localizedErrorEntry.values.add(localizedErrorValue);
65        localizedError.values.add(localizedErrorEntry);
66
67        // Default Observer
68        val observer = ProjectGeneratorFactory.eINSTANCE.createObserver;
69        observer.name = "YourObserverName";
70        observer.type = "com.clevercure.common.event.StartedEvent";
71        observer.during = During.IN_PROG;
72        observer.notify = Notify.ALWAYS;
73        observer.className = "com.clevercure." + module.key.toLowerCase +
    ↪  ".event.observer.YourObserverClass";
74
75        // Default jpaConfig
76        val jpaConfig = ProjectGeneratorFactory.eINSTANCE.createJpaConfig;
77        jpaConfig.localizedEnums.add(localizedJpa);
78
79        // Default serviceConfig
80        val serviceConfig = ProjectGeneratorFactory.eINSTANCE.createServiceConfig;
81        serviceConfig.messageBundles.add(localizedError);
82
83        module.messageBundles.add(localizedJpa);
84        module.messageBundles.add(localizedError);
85        module.observers.add(observer);
86        module.jpaConfig = jpaConfig;
87        module.serviceConfig = serviceConfig;
88      ]
89    }
90
91    /**
92     * Module: Make Module.key upper case
93     */
94    @Fix(ValidatorId.MODULE_KEY_UPPER_CASE)
95    def fixModuleKey(Issue issue, IssueResolutionAcceptor acceptor) {
96      acceptor.accept(issue, 'Convert to upper case', 'Convert to upper case', '') [ element,
    ↪   context |
97        val module = (element as Module);
98        module.key = module.key.toUpperCase;
99      ]
100   }
101
102   /**
103    * Module: Observers unused
104    */
105   @Fix(ValidatorId.MODULE_OBSERVER_UNUSED)
106   def fixUnusedObservers(Issue issue, IssueResolutionAcceptor acceptor) {
107     acceptor.accept(issue, 'Remove unused observers', 'Remove unused observers (' +
    ↪   issue.data.join(",") + ')', '') [ element, context |
108       val module = (element as Module);
109       val observerNames = newArrayList(issue.data);
110       module.observers.removeAll(module.observers.stream.filter[observerNames.contains(name)].collect(toList));
111     ]
112   }
113
114   /**
115    * Module: Localized unused
116    */
117   @Fix(ValidatorId.MODULE_LOCALIZED_UNUSED)
118   def fixUnusedLocalized(Issue issue, IssueResolutionAcceptor acceptor) {
119     acceptor.accept(issue, 'Remove unused message bundles', 'Remove unused message bundles (' +
    ↪   issue.data.join(",") + ')', '') [ element, context |
```

```
120       val module = (element as Module);
121       val bundleNames = newArrayList(issue.data);
122       module.messageBundles.removeAll(
123         module.observers.stream.filter[!bundleNames.contains(name)].collect(toList));
124     ]
125   }
126
127   /**
128    * Localized: Remove or rename duplicate Localized matched by their name.
129    */
130   @Fix(ValidatorId.LOCALIZED_NAME_DUPLICATE)
131   def fixDuplicateLocalizedName(Issue issue, IssueResolutionAcceptor acceptor) {
132     // Remove entry
133     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
134       val localized = (element as Localized);
135       Objects.requireNonNull(localized, "Element should be Localized instance");
136       val module = (localized.eContainer as Module);
137       Objects.requireNonNull(module, "eContainer of Localized should be Module instance");
138
139       module.messageBundles.remove(localized);
140     ]
141     // Rename entry
142     acceptor.accept(issue, 'Rename', 'Rename', '') [ element, context |
143       val localized = (element as Localized);
144       Objects.requireNonNull(localized, "Element should be Localized instance");
145       val module = (localized.eContainer as Module);
146       Objects.requireNonNull(module, "eContainer of Localized should be Module instance");
147
148       val duplicates = module.messageBundles.stream.filter[name.equals(localized.name)].collect(
149         Collectors.toList);
150       for (var i = 1; i <= duplicates.size; i++) {
151         val loc = duplicates.get(i);
152         loc.name = loc.name + i;
153       }
154     ]
155   }
156
157   /**
158    * Localized: Remove or rename duplicate LocalizedEntry instances of the given Localized
    ↪  instance.
159    */
160   @Fix(ValidatorId.LOCALIZED_ENTRY_DUPLICATE)
161   def fixLocalizedEntryDuplicates(Issue issue, IssueResolutionAcceptor acceptor) {
162     // Remove entry
163     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
164       val localized = (element as Localized);
165       Objects.requireNonNull(localized, "Element should be Localized instance");
166
167       if ((issue.data != null) && (issue.data.length > 0)) {
168         for (_key : issue.data) {
169           val optionalDuplicate =
    ↪  localized.values.stream.filter[localizedKey.equals(_key)].findFirst;
170           if (optionalDuplicate.isPresent) {
171             localized.values.remove(optionalDuplicate.get);
172           }
173         }
174       }
175     ]
176     // Rename entry
177     acceptor.accept(issue, 'Rename', 'Rename', '') [ element, context |
178       val localized = (element as Localized);
179       Objects.requireNonNull(localized, "Element should be Observer instance");
180
```

```
181       if ((issue.data != null) && (issue.data.length > 0)) {
182         for (_key : issue.data) {
183           val optionalDuplicate =
    ↪  localized.values.stream.filter[localizedKey.equals(_key)].findFirst;
184           if (optionalDuplicate.isPresent) {
185             optionalDuplicate.get.localizedKey = optionalDuplicate.get.localizedKey + "_1";
186           }
187         }
188       }
189     ]
190   }
191
192   /**
193    * Localized: Remove or rename duplicate LocalizedEntry locale.
194    */
195   @Fix(ValidatorId.LOCALIZED_ENTRY_LOCALE_DUPLICATE)
196   def fixLocalizedEntryLocaleDuplicates(Issue issue, IssueResolutionAcceptor acceptor) {
197     // Remove entry
198     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
199       val localized = (element as LocalizedEntry);
200       Objects.requireNonNull(localized, "Element should be Localized instance");
201
202       if ((issue.data != null) && (issue.data.length > 0)) {
203         for (_key : issue.data) {
204           val optionalDuplicate =
    ↪  localized.values.stream.filter[locale.toString.equals(_key)].findFirst;
205           if (optionalDuplicate.isPresent) {
206             localized.values.remove(optionalDuplicate.get);
207           }
208         }
209       }
210     ]
211   }
212
213   /**
214    * Observer: Remove or rename duplicate Observer matched by their name.
215    */
216   @Fix(ValidatorId.OBSERVER_NAME_DUPLICATE)
217   def fixDuplicateObserverName(Issue issue, IssueResolutionAcceptor acceptor) {
218     // Remove entry
219     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
220       val bserver = (element as Observer);
221       Objects.requireNonNull(bserver, "Element should be Observer instance");
222       val module = (bserver.eContainer as Module);
223       Objects.requireNonNull(module, "eContainer of Observer should be Module instance");
224
225       module.observers.remove(bserver);
226     ]
227     // Rename entry
228     acceptor.accept(issue, 'Rename', 'Rename', '') [ element, context |
229       val observer = (element as Observer);
230       Objects.requireNonNull(observer, "Element should be Observer instance");
231       val module = (observer.eContainer as Module);
232       Objects.requireNonNull(module, "eContainer of Observer should be Module instance");
233
234       val duplicates =
    ↪  module.observers.stream.filter[name.equals(observer.name)].collect(Collectors.toList);
235       for (var i = 1; i <= duplicates.size; i++) {
236         val obs = duplicates.get((i - 1));
237         obs.setName(obs.getName() + i);
238       }
239     ]
240   }
```

```
241
242   /**
243    * ServiceConfig: Remove duplicate mapped message bundle matched by their name
244    */
245   @Fix(ValidatorId.SERVICE_CONFIG_MESSAGE_BUNDLE_DUPLICATE)
246   def fixServiceConfigDuplicateMessageBundles(Issue issue, IssueResolutionAcceptor acceptor) {
247     // Remove entry
248     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
249       val config = (element as ServiceConfig);
250       Objects.requireNonNull(config, "Element should be ServiceConfig instance");
251
252       if ((issue.data != null) && (issue.data.length > 0)) {
253         for (_name : issue.data) {
254           val optionalDuplicate =
↪    config.messageBundles.stream.filter[name.equals(_name)].findFirst;
255           if (optionalDuplicate.isPresent) {
256             config.messageBundles.remove(optionalDuplicate.get);
257           }
258         }
259       }
260     ]
261   }
262
263   /**
264    * ServiceConfig: Remove duplicate mapped message bundle matched by their name
265    */
266   @Fix(ValidatorId.SERVICE_CONFIG_OBSERVERS_DUPLICATE)
267   def fixServiceConfigDuplicateObservers(Issue issue, IssueResolutionAcceptor acceptor) {
268     // Remove entry
269     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
270       val config = (element as ServiceConfig);
271       Objects.requireNonNull(config, "Element should be ServiceConfig instance");
272
273       if ((issue.data != null) && (issue.data.length > 0)) {
274         for (_name : issue.data) {
275           val optionalDuplicate = config.observers.stream.filter[name.equals(_name)].findFirst;
276           if (optionalDuplicate.isPresent) {
277             config.observers.remove(optionalDuplicate.get);
278           }
279         }
280       }
281     ]
282   }
283
284
285   /**
286    * JpaConfig: Remove duplicate mapped message bundles mapped by their name.
287    */
288   @Fix(ValidatorId.JPA_LOCALIZED_ENUMS_DUPLICATE)
289   def fixJpaConfigDuplicateMessageBundles(Issue issue, IssueResolutionAcceptor acceptor) {
290     // Remove entry
291     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
292       val config = (element as JpaConfig);
293       Objects.requireNonNull(config, "Element should be JpaConfig instance");
294
295       if ((issue.data != null) && (issue.data.length > 0)) {
296         for (_name : issue.data) {
297           val optionalDuplicate =
↪    config.localizedEnums.stream.filter[name.equals(_name)].findFirst;
298           if (optionalDuplicate.isPresent) {
299             config.localizedEnums.remove(optionalDuplicate.get);
300           }
301         }
```

```
302       }
303     ]
304   }
305
306   /**
307    * JpaConfig: Remove duplicate mapped observers mapped by their name.
308    */
309   @Fix(ValidatorId.JPA_OBSERVERS_DUPLICATE)
310   def fixJpaConfigDuplicateObservers(Issue issue, IssueResolutionAcceptor acceptor) {
311     // Remove entry
312     acceptor.accept(issue, 'Remove', 'Remove', '') [ element, context |
313       val config = (element as JpaConfig);
314       Objects.requireNonNull(config, "Element should be JpaConfig instance");
315
316       if ((issue.data != null) && (issue.data.length > 0)) {
317         for (_name : issue.data) {
318           val optionalDuplicate = config.observers.stream.filter[name.equals(_name)].findFirst;
319           if (optionalDuplicate.isPresent) {
320             config.observers.remove(optionalDuplicate.get);
321           }
322         }
323       }
324     ]
325   }
326 }
```

5 Formatter

Listing 4: ProjectGeneratorFormatter.xtend

```
/*
 * generated by Xtext 2.10.0
 */
package at.ooe.fh.mdm.herzog.dsl.proj.formatting2

import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Localized
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.LocalizedEntry
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.LocalizedValue
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Module
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.ServiceConfig
import at.ooe.fh.mdm.herzog.dsl.proj.services.ProjectGeneratorGrammarAccess
import com.google.inject.Inject
import org.eclipse.xtext.formatting2.AbstractFormatter2
import org.eclipse.xtext.formatting2.IFormattableDocument
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Observer

class ProjectGeneratorFormatter extends AbstractFormatter2 {

  @Inject extension ProjectGeneratorGrammarAccess

  def dispatch void format(Module module, extension IFormattableDocument document) {
    val open = module.regionFor.keyword("{").prepend[noSpace].append[newLine];
    val close = module.regionFor.keyword("}").prepend[newLine];
    interior(open, close)[indent]

    module.regionFor.keyword("key").prepend[newLine];
    module.regionFor.keyword("cdiEnabled").prepend[newLine];
    module.regionFor.keyword("messageBundles").prepend[newLine];

    module.regionFor.keyword("observers").prepend[newLine];
    val openObs = module.regionFor.keyword("observers {");
    val closeObs = module.regionFor.keyword("}");
    interior(openObs, closeObs)[indent];

    module.regionFor.keyword(";").prepend[noSpace];

    // TODO: format HiddenRegions around keywords, attributes, cross references, etc.
    module.messageBundles.forEach[format];
    module.observers.forEach[format];
    module.getJpaConfig.format;
    module.getServiceConfig.format;
  }

  def dispatch void format(Localized localized, extension IFormattableDocument document) {
    val open = localized.regionFor.keyword("{").prepend[noSpace].append[newLine];
    val close = localized.regionFor.keyword("}").prepend[newLine];
    interior(open, close)[indent]

    localized.regionFor.keyword("values").prepend[newLine];
    localized.regionFor.keyword(";").prepend[noSpace].append[newLine];

    localized.values.forEach[format];
  }

  def dispatch void format(LocalizedEntry localizedEntry, extension IFormattableDocument document)
↪  {
    val open = localizedEntry.regionFor.keyword("{").prepend[newLine].append[newLine];
    val close = localizedEntry.regionFor.keyword("}").prepend[newLine];
    interior(open, close)[indent]
```

```
59
60      localizedEntry.regionFor.keyword("key").prepend[newLine];
61      localizedEntry.regionFor.keyword("values").prepend[newLine];
62      localizedEntry.regionFor.keyword(";").prepend[noSpace];
63
64      localizedEntry.values.forEach[format];
65    }
66
67    def dispatch void format(LocalizedValue localizedValue, extension IFormattableDocument document)
      ↪  {
68      val open = localizedValue.regionFor.keyword("{").prepend[newLine].append[newLine];
69      val close = localizedValue.regionFor.keyword("}").prepend[newLine];
70      interior(open, close)[indent]
71
72      localizedValue.regionFor.keyword("locale").prepend[newLine];
73      localizedValue.regionFor.keyword("value").prepend[newLine];
74      localizedValue.regionFor.keyword(";").prepend[noSpace];
75    }
76
77    def dispatch void format(ServiceConfig serviceConfig, extension IFormattableDocument document) {
78      serviceConfig.observers.forEach[format];
79    }
80
81    def dispatch void format(Observer observer, extension IFormattableDocument document) {
82      val open = observer.regionFor.keyword("{").prepend[noSpace].append[newLine];
83      val close = observer.regionFor.keyword("}").prepend[newLine];
84      interior(open, close)[indent; indent; indent]
85
86      observer.regionFor.keyword("type").prepend[newLine];
87      observer.regionFor.keyword("during").prepend[newLine];
88      observer.regionFor.keyword("notify").prepend[newLine];
89      observer.regionFor.keyword("delegate").prepend[newLine];
90      observer.regionFor.keyword(";").prepend[noSpace].append[newLine];
91    }
92
93  // TODO: implement for JpaConfig, Localized, LocalizedEntry
94  }
```

6 Generator

Listing 5: ProjectGeneratorGenerator.xtend

```
/*
 * generated by Xtext 2.10.0
 */
package at.ooe.fh.mdm.herzog.dsl.proj.generator

import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Module
import java.util.List
import java.util.Map
import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.AbstractGenerator
import org.eclipse.xtext.generator.IFileSystemAccess2
import org.eclipse.xtext.generator.IGeneratorContext
import at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Observer

/**
 * Generates code from your model files on save.
 *
 * See https://www.eclipse.org/Xtext/documentation/303_runtime_concepts.html#code-generation
 */
class ProjectGeneratorGenerator extends AbstractGenerator {

  static val PARENT_ARTIFACT = "parent";
  static val PARENT_GROUP = "com.app";
  static val PACKAGE_PREFIX = "com.app.";

  static val DIR_PACKAGE_PREFIX = "com/app/";
  static val SRC_MAIN_JAVA = "src/main/java/";
  static val SRC_MAIN_RES = "src/main/resources/";

  override void doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) {
    for (e : resource.allContents.toIterable.filter(typeof(Module))) {
      generateParentProjectModule(e, resource, fsa, context);
      generateParentProjectModel(e, resource, fsa, context);
      generateProjectMessage(e, resource, fsa, context);
      generateProjectJpa(e, resource, fsa, context);
      generateParentProjectService(e, resource, fsa, context);
      generateProjectServiceApi(e, resource, fsa, context);
      generateProjectServiceImpl(e, resource, fsa, context);
    }
  }

  /**
   * Generate the parent project for the described module
   */
  def generateParentProjectModule(Module _module, Resource resource, IFileSystemAccess2 fsa,
    IGeneratorContext context) {
    val projectDir = _module.key.toLowerCase + "/";
    fsa.generateFile(projectDir + "/pom.xml", _module.pomParentModule);
  }

  /**
   * Generate the message project
   */
  def generateProjectMessage(Module _module, Resource resource, IFileSystemAccess2 fsa,
↪   IGeneratorContext context) {
    if (_module.messageBundles.isEmpty) {
      return;
    }
    val projectDir = _module.key.toLowerCase + "/message/";
```

```
59      fsa.generateFile(projectDir + "/pom.xml", _module.pomMessage);
60      if (at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Boolean.TRUE.equals(_module.cdiEnabled)) {
61        fsa.generateFile(projectDir + SRC_MAIN_RES + "META-INF/beans.xml", beansXml(null));
62      }
63      for (bundle : _module.messageBundles) {
64        val localeToKeyValueMap = newHashMap();
65        for (entry : bundle.values) {
66          for (value : entry.values) {
67            var keyToValueMap = localeToKeyValueMap.get(value.locale);
68            if (keyToValueMap == null) {
69              localeToKeyValueMap.put(value.locale, newHashMap());
70              keyToValueMap = localeToKeyValueMap.get(value.locale);
71            }
72            keyToValueMap.put(entry.localizedKey, value.value);
73          }
74        }
75        for (entry : localeToKeyValueMap.entrySet) {
76          fsa.generateFile(
77            projectDir + SRC_MAIN_RES + "META-INF/resource-bundles/" + bundle.name.toLowerCase + "_"
    ↪  +
78              entry.key.toString + ".properties", messageBundleProperties(entry.value));
79          fsa.generateFile(
80            projectDir + SRC_MAIN_JAVA + DIR_PACKAGE_PREFIX + _module.key.toLowerCase + "/message/"
    ↪  +
81              bundle.name + ".java",
82            messageBundleEnum((PARENT_GROUP + "." + _module.key.toLowerCase + ".message"),
    ↪  bundle.name,
83              newArrayList(entry.value.keySet)));
84        }
85      }
86    }
87
88    /**
89     * Generate the parent project for the model projects
90     */
91    def generateParentProjectModel(Module _module, Resource resource, IFileSystemAccess2 fsa,
92      IGeneratorContext context) {
93      val projectDir = _module.key.toLowerCase + "/model/";
94      fsa.generateFile(projectDir + "/pom.xml", _module.pomParentModel);
95    }
96
97    /**
98     * Generate the jpa project
99     */
100   def generateProjectJpa(Module _module, Resource resource, IFileSystemAccess2 fsa,
    ↪  IGeneratorContext context) {
101     if (_module.jpaConfig == null) {
102       return;
103     }
104     val projectDir = _module.key.toLowerCase + "/model/jpa/";
105     fsa.generateFile(projectDir + "/pom.xml", _module.pomJpa);
106     if (at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Boolean.TRUE.equals(_module.cdiEnabled)) {
107       fsa.generateFile(projectDir + SRC_MAIN_RES + "META-INF/beans.xml", beansXml(null));
108       fsa.generateFile(projectDir + SRC_MAIN_RES + "META-INF/" + _module.key.toLowerCase +
    ↪  "Orm.xml",
109         beansXml(null));
110       if (!_module.jpaConfig.observers.isEmpty) {
111         fsa.generateFile(projectDir + SRC_MAIN_JAVA + DIR_PACKAGE_PREFIX + _module.key.toLowerCase
    ↪  +
112           "/model/jpa/observer/Observer.java",
113           observerClass(PACKAGE_PREFIX + _module.key.toLowerCase + ".observer",
    ↪  _module.jpaConfig.observers));
114       }
```

```
115       }
116     }
117
118     /**
119      * Generate the parent project for the service projects
120      */
121     def generateParentProjectService(Module _module, Resource resource, IFileSystemAccess2 fsa,
122       IGeneratorContext context) {
123       if (_module.serviceConfig == null) {
124         return;
125       }
126       val projectDir = _module.key.toLowerCase + "/service/";
127       fsa.generateFile(projectDir + "/pom.xml", _module.pomServiceParent);
128     }
129
130     /**
131      * Generate the service api project
132      */
133     def generateProjectServiceApi(Module _module, Resource resource, IFileSystemAccess2 fsa,
134       IGeneratorContext context) {
135       if (_module.serviceConfig == null) {
136         return;
137       }
138       val projectDir = _module.key.toLowerCase + "/service/api/";
139       fsa.generateFile(projectDir + "/pom.xml", _module.pomServiceApi);
140       if (at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Boolean.TRUE.equals(_module.cdiEnabled)) {
141         fsa.generateFile(projectDir + SRC_MAIN_RES + "META-INF/beans.xml", beansXml(null));
142       }
143     }
144
145     /**
146      * Generate the service impl project
147      */
148     def generateProjectServiceImpl(Module _module, Resource resource, IFileSystemAccess2 fsa,
149       IGeneratorContext context) {
150       if (_module.serviceConfig == null) {
151         return;
152       }
153       val projectDir = _module.key.toLowerCase + "/service/impl/";
154       fsa.generateFile(projectDir + "/pom.xml", _module.pomServiceImpl);
155       // TODO: extract decorators
156       if (at.ooe.fh.mdm.herzog.dsl.proj.projectGenerator.Boolean.TRUE.equals(_module.cdiEnabled)) {
157         fsa.generateFile(projectDir + SRC_MAIN_RES + "META-INF/beans.xml", beansXml(null));
158         if (!_module.serviceConfig.observers.isEmpty) {
159           fsa.generateFile(projectDir + SRC_MAIN_JAVA + DIR_PACKAGE_PREFIX + _module.key.toLowerCase
    ↪  +
160             "/service/impl/observer/Observer.java",
161             observerClass(PACKAGE_PREFIX + _module.key.toLowerCase + ".observer",
162               _module.serviceConfig.observers));
163         }
164       }
165     }
166
167     /**
168      * Template for the beans xml.
169      */
170     def beansXml(List<Observer> observers) '''
171       <?xml version="1.0" encoding="UTF-8"?>
172         <beans xmlns="http://java.sun.com/xml/ns/javaee"
    ↪    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ↪    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    ↪    http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
173
```

```
174
175       <!-- Add decorators, interceptors, .. here -->
176     </beans>
177   '''
178
179   /**
180    * Template for the beans xml.
181    */
182   def ormXml() '''
183     <entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm"
184       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
185       xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/orm
      ↪  http://java.sun.com/xml/ns/persistence/orm_2_0.xsd"
186       version="2.0">
187
188       <!-- Add your first entity here -->
189       </entity-mappings>
190   '''
191
192   /**
193    * Template for the observer class.
194    */
195   def observerClass(String packageName, List<Observer> observers) '''
196     package packageName;
197
198     import javax.enterprise.event.*;
199
200     @Dependent
201     public class Observer {
202
203     FOR observer : observers
204       @Inject
205         private Event<observer.type> eventobservers.indexOf(observer);
206         @Inject
207         private observer.className delegateEventobservers.indexOf(observer);
208     ENDFOR
209     FOR observer : observers
210       public void
      ↪   observerEvent_observers.indexOf(observer)(@Observes(during=TransactionPhase.observer.during.toString,
      ↪   notifyObserver = Reception.observer.getNotify().toString()) observer.type evt) {
211         delegateEventobservers.indexOf(observer).observe(evt);
212       }
213     ENDFOR
214     }
215   '''
216
217   /**
218    * Template for the message bundle properties file.
219    */
220   def messageBundleProperties(Map<String, String> _keyTovalueMap) '''
221     FOR entry : _keyTovalueMap.entrySet
222       entry.key=entry.value
223     ENDFOR
224   '''
225
226   /**
227    * Template for the message bundle enumeration for the keys.
228    */
229   def messageBundleEnum(String packageName, String bundleName, List<String> _keys) '''
230     package packageName.message;
231
232     public enum bundleNameMessageBundle {
```

```
233         _keys.join(',\n,');
234       }
235     '''
236
237     /**
238      * Template for the pom.xml of the module parent project
239      */
240     def pomParentModule(Module _module) '''
241       <?xml version="1.0" encoding="UTF-8"?>
242       <project xmlns="http://maven.apache.org/POM/4.0.0"
243                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
244                xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
245           <parent>
246               <artifactId>PARENT_ARTIFACT</artifactId>
247               <groupId>PARENT_GROUP</groupId>
248               <version>0.0.1-SNAPSHOT</version>
249           </parent>
250           <modelVersion>4.0.0</modelVersion>
251           <packaging>pom</packaging>
252           <artifactId>module-_module.key.toLowerCase</artifactId>
253           <name>module-_module.key.toLowerCase-message</name>
254           <description>The parent project holding the module projects</description>
255
256
257             <modules>
258                 <module>message</module>
259                 <module>model</module>
260                 <module>service</module>
261             </modules>
262       </project>
263     '''
264
265     /**
266      * Template for the pom.xml of the message project
267      */
268     def pomMessage(Module _module) '''
269       <?xml version="1.0" encoding="UTF-8"?>
270       <project xmlns="http://maven.apache.org/POM/4.0.0"
271                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
272                xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
273           <parent>
274               <artifactId>PARENT_ARTIFACT</artifactId>
275               <groupId>PARENT_GROUP</groupId>
276               <version>0.0.1-SNAPSHOT</version>
277           </parent>
278           <modelVersion>4.0.0</modelVersion>
279           <packaging>pom</packaging>
280           <artifactId>_module.key.toLowerCase-message</artifactId>
281           <name>_module.key.toLowerCase-message</name>
282           <description>The project holding the localizations</description>
283
284       </project>
285     '''
286
287     /**
288      * Template for the pom.xml of the model parent project
289      */
290     def pomParentModel(Module _module) '''
291       <?xml version="1.0" encoding="UTF-8"?>
292       <project xmlns="http://maven.apache.org/POM/4.0.0"
293                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
294             xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
295         <parent>
296             <artifactId>PARENT_ARTIFACT</artifactId>
297             <groupId>PARENT_GROUP</groupId>
298             <version>0.0.1-SNAPSHOT</version>
299         </parent>
300         <modelVersion>4.0.0</modelVersion>
301         <packaging>pom</packaging>
302         <artifactId>module-_module.key.toLowerCase-model</artifactId>
303         <name>_module.key.toLowerCase-model</name>
304         <description>The main project for all model projects</description>
305 
306         <modules>
307             <module>jpa</module>
308         </modules>
309 
310     </project>
311   '''
312 
313   /**
314    * Template for the pom.xml of the model jpa project
315    */
316   def pomJpa(Module _module) '''
317     <?xml version="1.0" encoding="UTF-8"?>
318     <project xmlns="http://maven.apache.org/POM/4.0.0"
319              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
320              xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
321         <parent>
322             <artifactId>PARENT_ARTIFACT</artifactId>
323             <groupId>PARENT_GROUP</groupId>
324             <version>0.0.1-SNAPSHOT</version>
325         </parent>
326         <modelVersion>4.0.0</modelVersion>
327         <packaging>pom</packaging>
328       <artifactId>_module.key.toLowerCase-model-jpa</artifactId>
329       <name>_module.key.toLowerCase-model-jpa</name>
330       <description>The jpa model project</description>
331 
332       <!-- Dependencies Other projects -->
333         <dependency>
334             <groupId>PARENT_GROUP</groupId>
335             <artifactId>_module.key.toLowerCase-message</artifactId>
336             <version>${project.version}</version>
337         </dependency>
338 
339     </project>
340   '''
341 
342   /**
343    * Template for the pom.xml of the service parent project
344    */
345   def pomServiceParent(Module _module) '''
346     <?xml version="1.0" encoding="UTF-8"?>
347     <project xmlns="http://maven.apache.org/POM/4.0.0"
348              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
349              xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
350         <parent>
351             <artifactId>PARENT_ARTIFACT</artifactId>
352             <groupId>PARENT_GROUP</groupId>
353             <version>0.0.1-SNAPSHOT</version>
```

```
354         </parent>
355         <modelVersion>4.0.0</modelVersion>
356         <packaging>pom</packaging>
357 
358         <modules>
359             <module>jpa</module>
360         </modules>
361 
362         <artifactId>module-|_module.key.toLowerCase|-service</artifactId>
363         <name>|_module.key.toLowerCase|-service</name>
364         <description>The parent project for all service projects</description>
365 
366 
367         <modules>
368             <module>api</module>
369             <module>impl</module>
370         </modules>
371     </project>
372   '''
373 
374   /**
375    * Template for the pom.xml of the service api project
376    */
377   def pomServiceApi(Module _module) '''
378     <?xml version="1.0" encoding="UTF-8"?>
379     <project xmlns="http://maven.apache.org/POM/4.0.0"
380              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
381              xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
382         <parent>
383             <artifactId>|PARENT_ARTIFACT|</artifactId>
384             <groupId>|PARENT_GROUP|</groupId>
385             <version>0.0.1-SNAPSHOT</version>
386         </parent>
387         <modelVersion>4.0.0</modelVersion>
388         <packaging>jar</packaging>
389         <artifactId>|_module.key.toLowerCase|-service-api</artifactId>
390         <name>|_module.key.toLowerCase|-service-api</name>
391         <description>The api for the service logic operations</description>
392 
393 
394            <!-- Dependencies Other projects -->
395            <dependency>
396                <groupId>|PARENT_GROUP|</groupId>
397                <artifactId>|_module.key.toLowerCase|-model-jpa</artifactId>
398                <version>${project.version}</version>
399            </dependency>
400 
401     </project>
402   '''
403 
404   /**
405    * Template for the pom.xml of the service impl project
406    */
407   def pomServiceImpl(Module _module) '''
408     <?xml version="1.0" encoding="UTF-8"?>
409     <project xmlns="http://maven.apache.org/POM/4.0.0"
410              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
411              xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
412         <parent>
413             <artifactId>|PARENT_ARTIFACT|</artifactId>
414             <groupId>|PARENT_GROUP|</groupId>
```

```
            <version>0.0.1-SNAPSHOT</version>
        </parent>
        <modelVersion>4.0.0</modelVersion>
        <packaging>jar</packaging>
        <artifactId>_module.key.toLowerCase-service-impl</artifactId>
        <name>_module.key.toLowerCase-service-impl</name>
        <description>The implementation for the service logic operations</description>


            <!-- Dependencies Other projects -->
            <dependency>
                <groupId>PARENT_GROUP</groupId>
                <artifactId>_module.key.toLowerCase-service-api</artifactId>
                <version>${project.version}</version>
            </dependency>

    </project>
  '''
}
```

7 Resultate

7.1 DSL

Listing 6: ProjectGeneratorQuickfixProvider.xtend

```
module Evaluation{
  key 'EVAL';
  cdiEnabled true;
  messageBundles {
        OrderState{
      values {
      {
        key CONFIRMED;
        values {
          {
            locale de_DE;
            value 'Confirmed';
          };
          {
            locale en_US;
            value 'Confirmed';
          };
        };
          };
      {
        key CANCLED;
        values {
          {
            locale en_US;
            value 'Cancled';
          };
        };
          };
      };
    };
        ErrorMessage{
      values {
        {
          key UNKNOWN;
          values {
            {
              locale en_US;
              value 'E00002: Unhandled error detected';
            };
          };
            };
      };
    };
  };
  observers {
        LoginObserver{
      type com.clevercure.event.LoginEvent;
      during InProgress;
      notifyObserver Always;
      delegate com.clevercure.evaluation.observer.LoginEventObserver;
    };
    };
    jpaConfig {
        localizedEnums {
            OrderState
          };
```

```
57          observers {
58            LoginObserver
59          };
60    };
61  serviceConfig {
62      observers {
63          LoginObserver
64        };
65      messageBundles {
66        OrderState ErrorMessage
67        };
68    };
69  }
```

7.2 Validierung und Quickfix

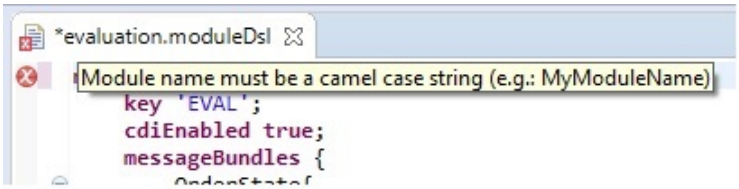

Abbildung 2: Modulname muss *camel case* sein

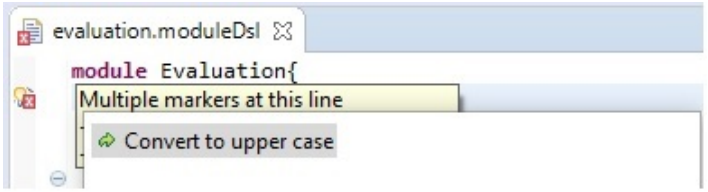

Abbildung 3: Modul *key* muss *upper case* sein

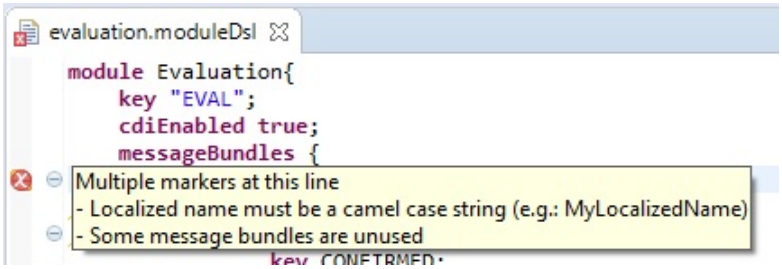

Abbildung 4: MessageBundle name muss *camel case* sein und sollte verwendet werden.

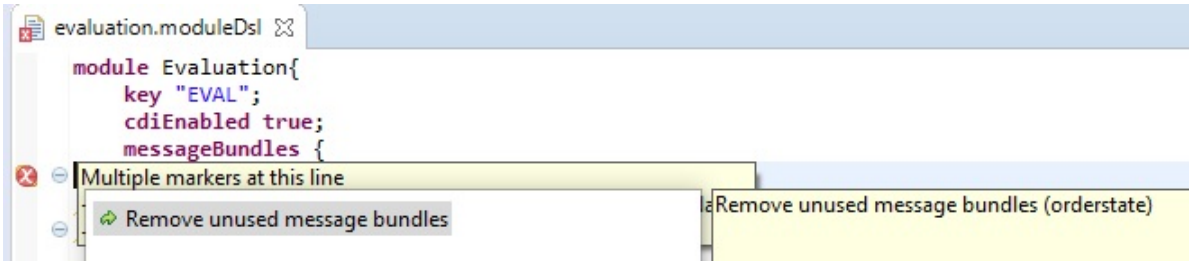

Abbildung 5: MessageBundle *Quickfix* Optionen.

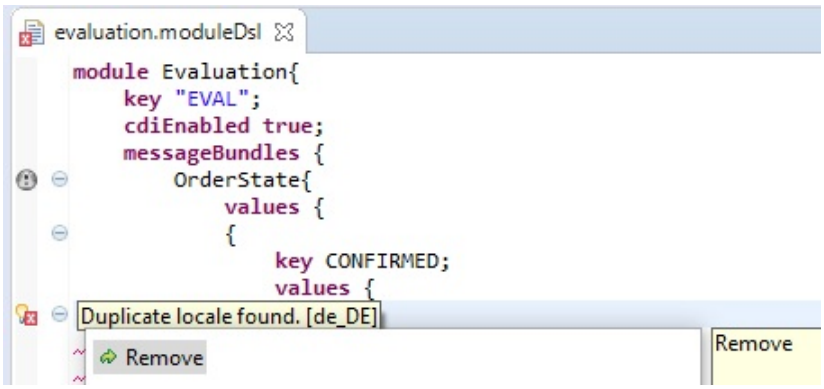

Abbildung 6: Sprachspezifischer Eintrag mit doppeltet vergebener *Locale*

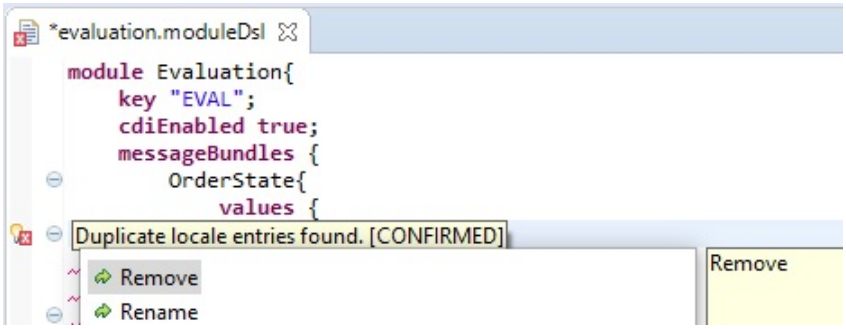

Abbildung 7: Sprachspezifischer Eintrag mit doppeltet vergebenen Schlüssel

Abbildung 8: Doppelter Name eines *Localized*

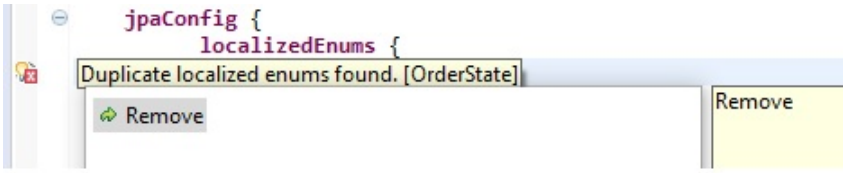

Abbildung 9: *JPA*-Konfiguration mit doppelten *Localized*

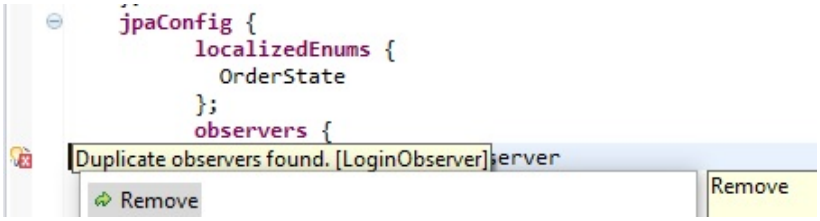

Abbildung 10: *JPA*-Konfiguration mit doppelten *Observer*

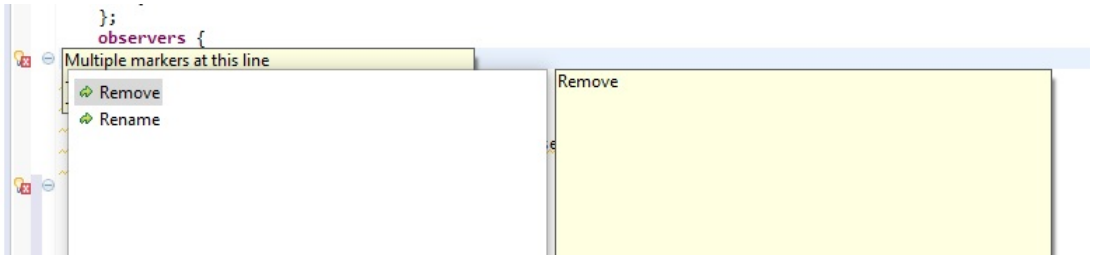

Abbildung 11: Doppelter Names eines *Observers*

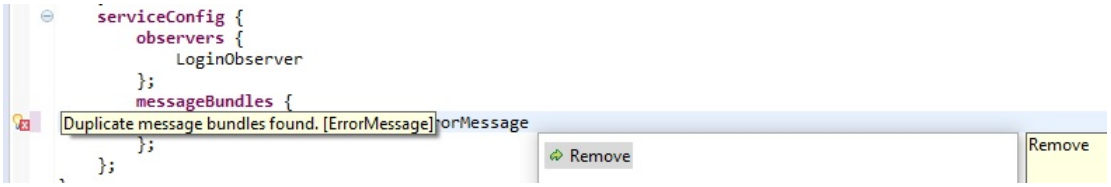

Abbildung 12: *Service*-Konfiguration mit doppelten *Localized*

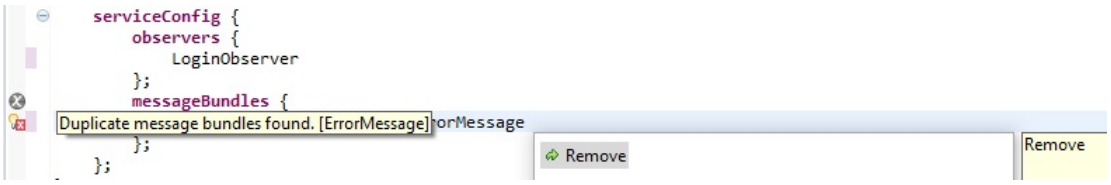

Abbildung 13: *Service*-Konfiguration mit doppelten *Observer*

7.3 Generiert

Für die Inhalte der generierten Ressourcen bitte die *./dsl/evaluation.moduledsl* im Hauptprojekt dazu verwenden, um das Testmodul zu generieren.

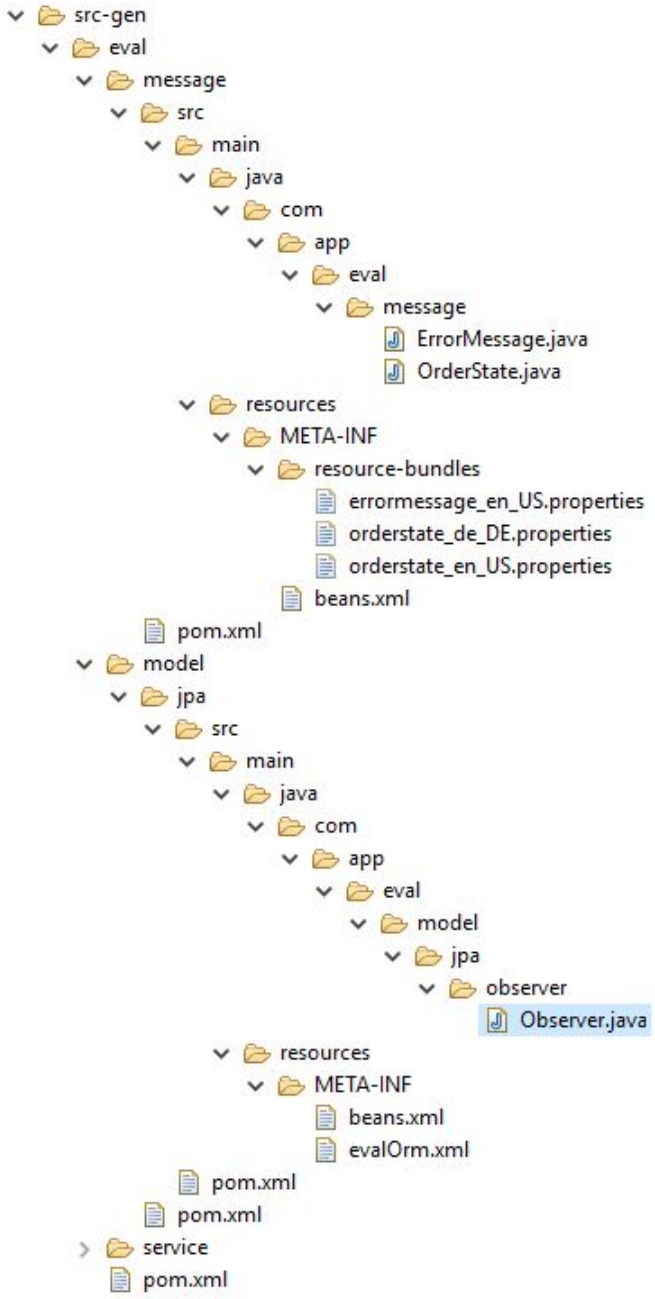

Abbildung 14: Erster Teil der generierten Ressourcen

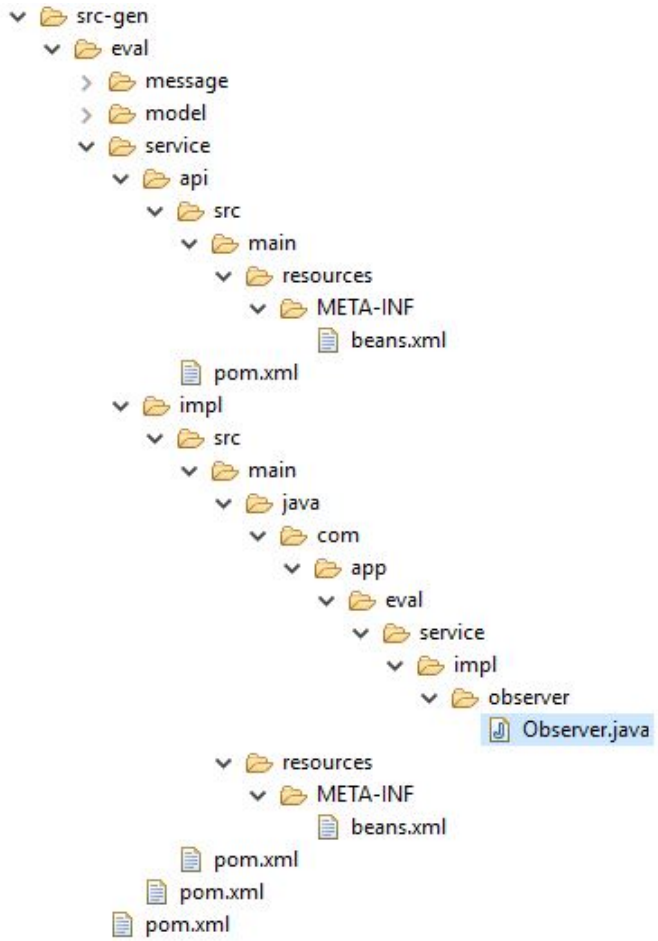

Abbildung 15: Zweiter Teil der generierten Ressourcen