POLITECNICO DI MILANO

Department of Electronic, Information and Biomedical Engineering

Master Degree course in Computer Science Engineering

Project of Software Engineering 2

PowerEnJoy

Code Inspection

(CI)

Version 1.0 (28 January 2017)

Reference professor: Prof. Elisabetta Di Nitto

Authors:

Davide Anghileri matr. 879194

Antonio Paladini matr. 879118

Contents

1 Classes assigned to the group

The methods assigned us belong all to the class EntityDataLoader.java

- Class: BaseContainer.java
- Package: framework/entity/src/main/java/org/apache/ofbiz/entity/util/EntityDataLoader.java
- Methods:
  - **public static String getPathsString** (String helperName)
  - **public static List<URL> getUrlList**(String helperName)
  - **public static List<URL> getUrlList**(String helperName, String componentName)
  - **public static <E> List<URL> getUrlList**(String helperName, List<E> readerNames)
  - **public static <E> List<URL> getUrlList**(String helperName, String componentName, List<E> readerNames)
  - **public static List<URL> getUrlByComponentList**(String helperName, List<String> components, List<String> readerNames)
  - **public static List<URL> getUrlByComponentList** (String helperName, List<String> components)
  - **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages) throws GenericEntityException
  - **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout) throws GenericEntityException
  - **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout, boolean dummyFks, boolean

maintainTxs, boolean tryInsert) throws GenericEntityException

- **public static int generateData**(Delegator delegator, List<Object> errorMessages) throws GenericEntityException

2 Functional role of assigned set of classes

The role of this class is to provide some utilities to load seed Data. Now we're going to give a briefly description of what, in our opinion, each method inside this class does.

➢ **public static String getPathsString** (String helperName)

This method is used to get the path of an Entity Data in the form of a string providing its name.

➢ **public static List<URL> getUrlList** (String helperName)

This method is only used to call another method with the same name that overloads this one. Moreover, it adds a null parameter and a list of readers.

➢ **public static List<URL> getUrlList**(String helperName, String componentName)

Also this method is only used to call another method with the same name that overloads this one. It adds a list of readers obtained by the name that it receives as input.

- **public static <E> List<URL> getUrlList**(String helperName, List<E> readerNames)

Also this method is only used to call another method with the same name that overloads this one. It adds a null value as second parameter.

- **public static <E> List<URL> getUrlList** (String helperName, String componentName, List<E> readerNames)

This method is the one called by the previous three methods and that overload them. Since there is no documentation which explains the purpose of this method and there are really few comments to explain how does it work we have tried to understand its behaviour analysing the code and the variables used.
First it reads all the files contained into the main resources (as specified in the entityengine.xml file) and into each main component or entity; then it reads all the files contained in the directories specified in the paths obtained by the HelperName and it converts all this files in their URL and finally returns a list containing all the computed URLs.

- **public static List<URL> getUrlByComponentList**(String helperName, List<String> components, List<String> readerNames)

This method receives in input a component list together with a helperName and a list of readerNames. Then for each cuple of readerName and component it retrieves the URL list related to them through the getUrlList() method explained above.

- **public static List<URL> getUrlByComponentList** (String helperName, List<String> components)

This method is created to call the previous one, which overload it. Before to do this, since it isn't provided as a parameter, it manages to retrieve the readerNames list through the helper Name.

- **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages) throws GenericEntityException

This method is only used to call the following one which overload this one. In the calling of the overloader method it sets the txTimeout to -1.

- **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout) throws GenericEntityException

Also this method is designed to call another method which overload this one. The method adds to the calling three new parameters all set to false.

- **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout, boolean dummyFks, boolean maintainTxs, boolean tryInsert) throws GenericEntityException

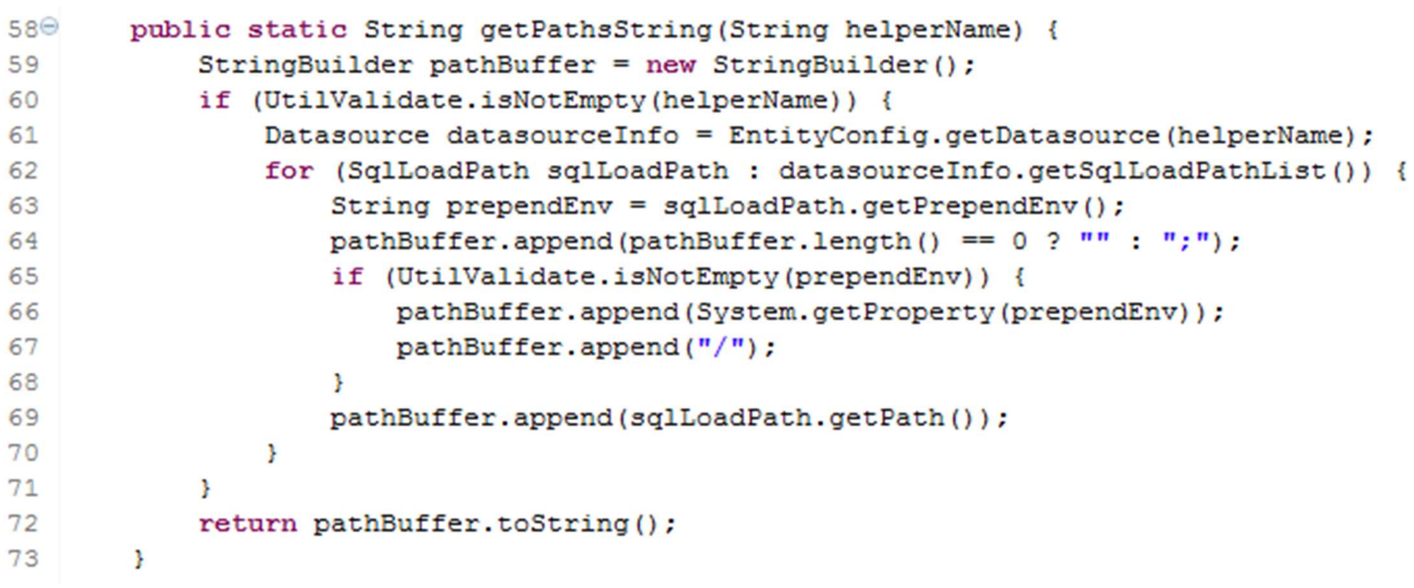
This method has the purpose to load an XML resource given a URL variable called dataUrl. First of all, it is checked whether this variable is set to null, in this case the function returns 0. Otherwise an EntitySaxReader object is used to load data from the dataUrl parameter and return the number of rows changed, which is also the return value of this function.

- **public static int generateData**(Delegator delegator, List<Object> errorMessages) throws GenericEntityException

This method goal is to generate data starting from a Delegator object. For each entity retrieved from the ModelReader of delegator the method saves the table-name in a string (baseName). After this the delegator creates an Entity in the form of a GenericValue class and store it saving the number of row changed, which is the return value of this method.

3 List of issues found by applying the checklist

3.1 BaseContainer.getPathsString(...)

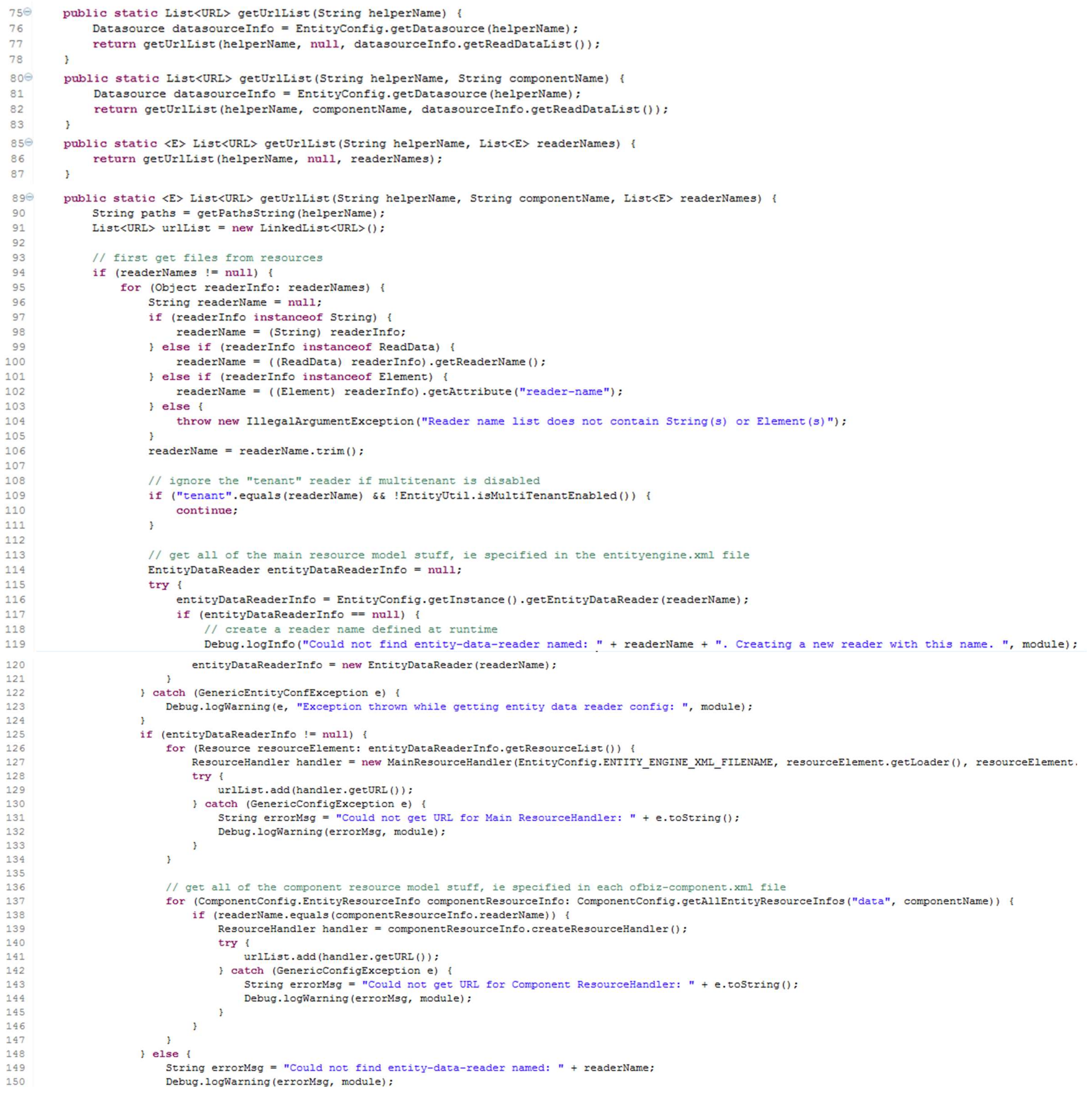


```
58      public static String getPathsString(String helperName) {
59          StringBuilder pathBuffer = new StringBuilder();
60          if (UtilValidate.isNotEmpty(helperName)) {
61              Datasource datasourceInfo = EntityConfig.getDatasource(helperName);
62              for (SqlLoadPath sqlLoadPath : datasourceInfo.getSqlLoadPathList()) {
63                  String prependEnv = sqlLoadPath.getPrependEnv();
64                  pathBuffer.append(pathBuffer.length() == 0 ? "" : ";");
65                  if (UtilValidate.isNotEmpty(prependEnv)) {
66                      pathBuffer.append(System.getProperty(prependEnv));
67                      pathBuffer.append("/");
68                  }
69                  pathBuffer.append(sqlLoadPath.getPath());
70              }
71          }
72          return pathBuffer.toString();
73      }
```

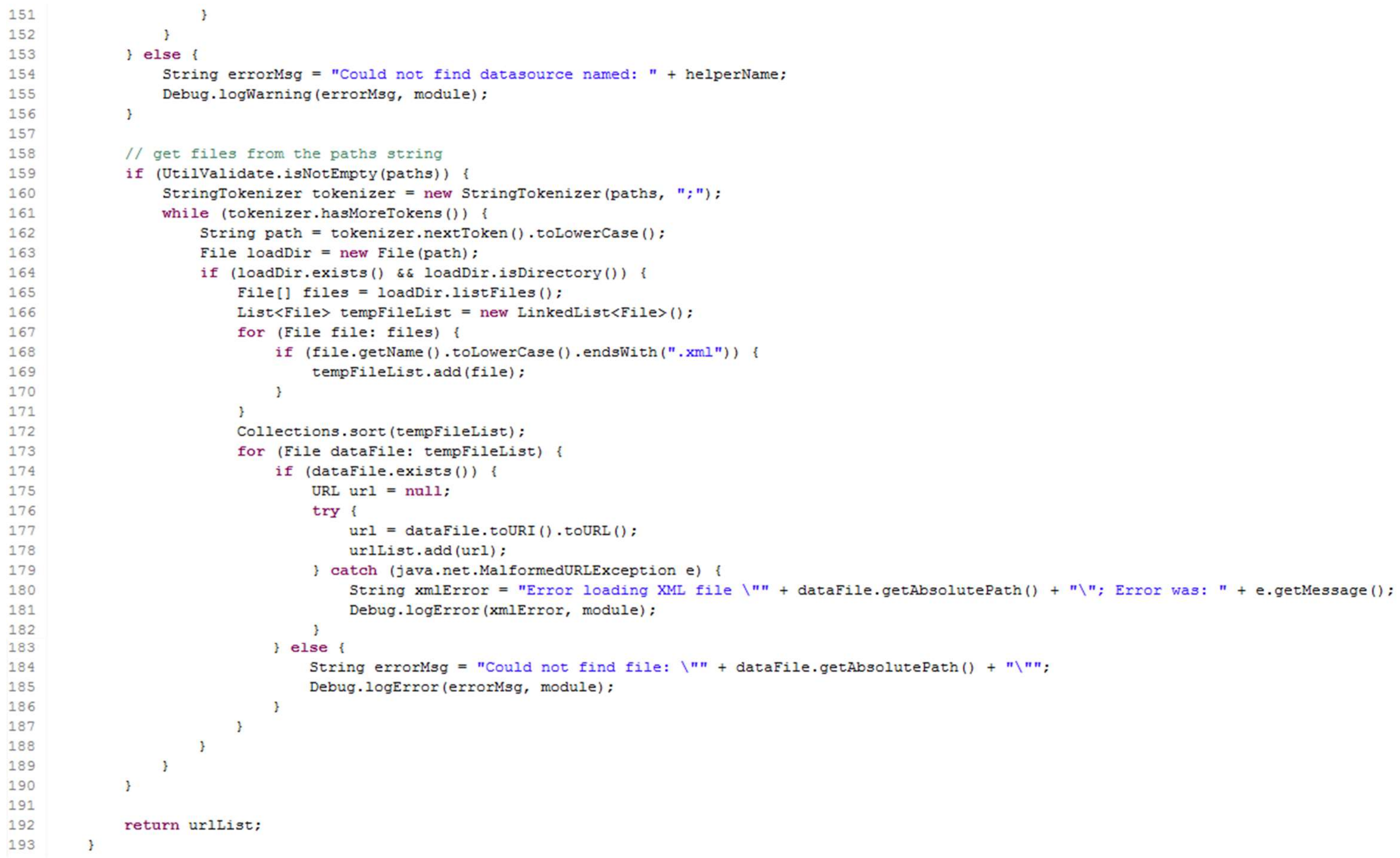
[Rule 18]: In this method there are no comments

Line 58 : [Rule 1] The name of the variable “helperName” in our opinion is not meaningful and it’s difficult to understand what this variable concern.

3.2 BaseCointainer.getUrlList(…)



```java
public static List<URL> getUrlList(String helperName) {
    Datasource datasourceInfo = EntityConfig.getDatasource(helperName);
    return getUrlList(helperName, null, datasourceInfo.getReadDataList());
}
public static List<URL> getUrlList(String helperName, String componentName) {
    Datasource datasourceInfo = EntityConfig.getDatasource(helperName);
    return getUrlList(helperName, componentName, datasourceInfo.getReadDataList());
}
public static <E> List<URL> getUrlList(String helperName, List<E> readerNames) {
    return getUrlList(helperName, null, readerNames);
}

public static <E> List<URL> getUrlList(String helperName, String componentName, List<E> readerNames) {
    String paths = getPathsString(helperName);
    List<URL> urlList = new LinkedList<URL>();

    // first get files from resources
    if (readerNames != null) {
        for (Object readerInfo: readerNames) {
            String readerName = null;
            if (readerInfo instanceof String) {
                readerName = (String) readerInfo;
            } else if (readerInfo instanceof ReadData) {
                readerName = ((ReadData) readerInfo).getReaderName();
            } else if (readerInfo instanceof Element) {
                readerName = ((Element) readerInfo).getAttribute("reader-name");
            } else {
                throw new IllegalArgumentException("Reader name list does not contain String(s) or Element(s)");
            }
            readerName = readerName.trim();

            // ignore the "tenant" reader if multitenant is disabled
            if ("tenant".equals(readerName) && !EntityUtil.isMultiTenantEnabled()) {
                continue;
            }

            // get all of the main resource model stuff, ie specified in the entityengine.xml file
            EntityDataReader entityDataReaderInfo = null;
            try {
                entityDataReaderInfo = EntityConfig.getInstance().getEntityDataReader(readerName);
                if (entityDataReaderInfo == null) {
                    // create a reader name defined at runtime
                    Debug.logInfo("Could not find entity-data-reader named: " + readerName + ". Creating a new reader with this name. ", module);
                    entityDataReaderInfo = new EntityDataReader(readerName);
                }
            } catch (GenericEntityConfException e) {
                Debug.logWarning(e, "Exception thrown while getting entity data reader config: ", module);
            }
            if (entityDataReaderInfo != null) {
                for (Resource resourceElement: entityDataReaderInfo.getResourceList()) {
                    ResourceHandler handler = new MainResourceHandler(EntityConfig.ENTITY_ENGINE_XML_FILENAME, resourceElement.getLoader(), resourceElement.
                    try {
                        urlList.add(handler.getURL());
                    } catch (GenericConfigException e) {
                        String errorMsg = "Could not get URL for Main ResourceHandler: " + e.toString();
                        Debug.logWarning(errorMsg, module);
                    }
                }

                // get all of the component resource model stuff, ie specified in each ofbiz-component.xml file
                for (ComponentConfig.EntityResourceInfo componentResourceInfo: ComponentConfig.getAllEntityResourceInfos("data", componentName)) {
                    if (readerName.equals(componentResourceInfo.readerName)) {
                        ResourceHandler handler = componentResourceInfo.createResourceHandler();
                        try {
                            urlList.add(handler.getURL());
                        } catch (GenericConfigException e) {
                            String errorMsg = "Could not get URL for Component ResourceHandler: " + e.toString();
                            Debug.logWarning(errorMsg, module);
                        }
                    }
                }
            } else {
                String errorMsg = "Could not find entity-data-reader named: " + readerName;
                Debug.logWarning(errorMsg, module);
```



```
151                 }
152             }
153         } else {
154             String errorMsg = "Could not find datasource named: " + helperName;
155             Debug.logWarning(errorMsg, module);
156         }
157
158         // get files from the paths string
159         if (UtilValidate.isNotEmpty(paths)) {
160             StringTokenizer tokenizer = new StringTokenizer(paths, ";");
161             while (tokenizer.hasMoreTokens()) {
162                 String path = tokenizer.nextToken().toLowerCase();
163                 File loadDir = new File(path);
164                 if (loadDir.exists() && loadDir.isDirectory()) {
165                     File[] files = loadDir.listFiles();
166                     List<File> tempFileList = new LinkedList<File>();
167                     for (File file: files) {
168                         if (file.getName().toLowerCase().endsWith(".xml")) {
169                             tempFileList.add(file);
170                         }
171                     }
172                     Collections.sort(tempFileList);
173                     for (File dataFile: tempFileList) {
174                         if (dataFile.exists()) {
175                             URL url = null;
176                             try {
177                                 url = dataFile.toURI().toURL();
178                                 urlList.add(url);
179                             } catch (java.net.MalformedURLException e) {
180                                 String xmlError = "Error loading XML file \"" + dataFile.getAbsolutePath() + "\"; Error was: " + e.getMessage();
181                                 Debug.logError(xmlError, module);
182                             }
183                         } else {
184                             String errorMsg = "Could not find file: \"" + dataFile.getAbsolutePath() + "\"";
185                             Debug.logError(errorMsg, module);
186                         }
187                     }
188                 }
189             }
190         }
191
192         return urlList;
193     }
```

Line 89 : [Rule 13] The length of the line is 102 characters and it's greater than 80.

Line 104 : [Rule 13] The length of the line is 97 characters and it's greater than 80.

Line 119 : [Rule 14] The length of the line is 126 characters and it's greater than 120.

Line 123 : [Rule 13] The length of the line is 92 characters and it's greater than 80.

Line 127 : [Rule 14] The length of the line is 151 characters and it's greater than 120.

Line 137 : [Rule 14] The length of the line is 131 characters and it's greater than 120.

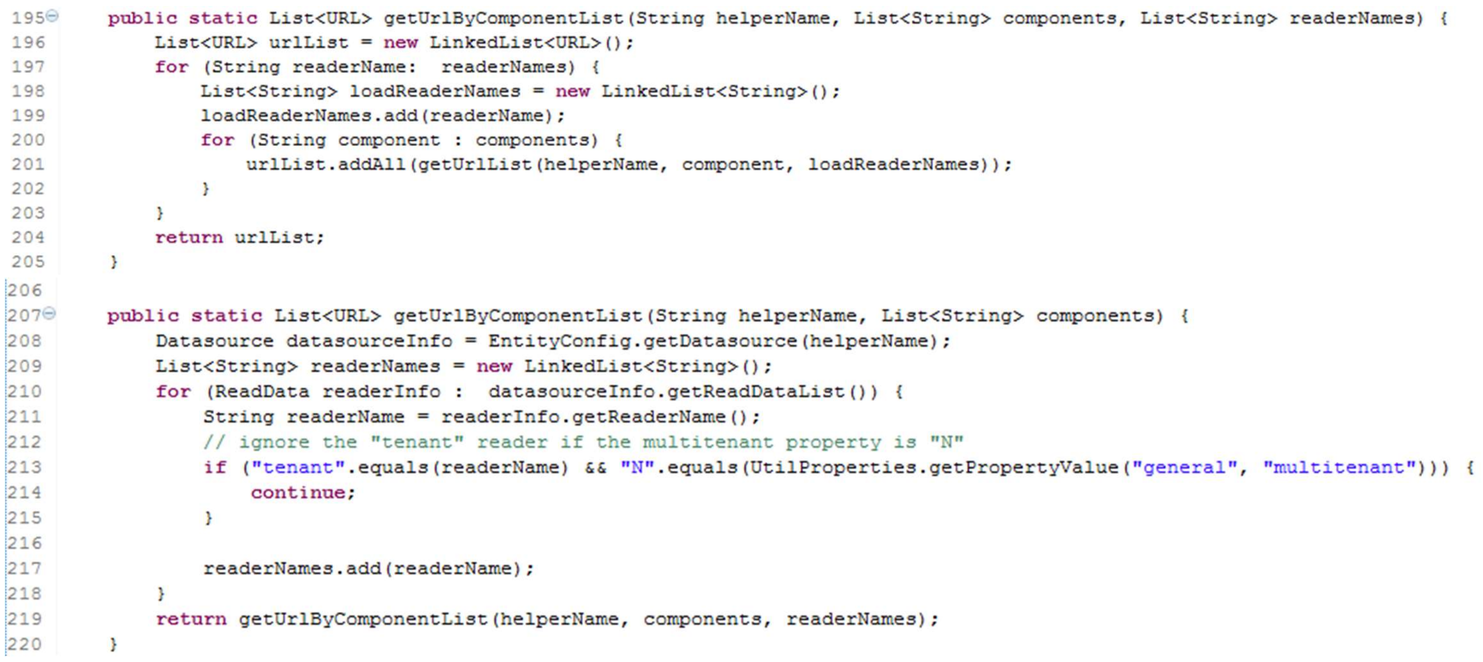
Line 180 : [Rule 13] The length of the line is 114 characters and it's greater than 80.

Line 195 : [Rule 13] The length of the line is 117 characters and it's greater than 80.

Lines 89-193 : [Rule 27] This method is too long (104 lines) and should be divided using sub-functions.

Line 120: [Rule 33] This declaration doesn't appear at the begging of the block in which it is written.

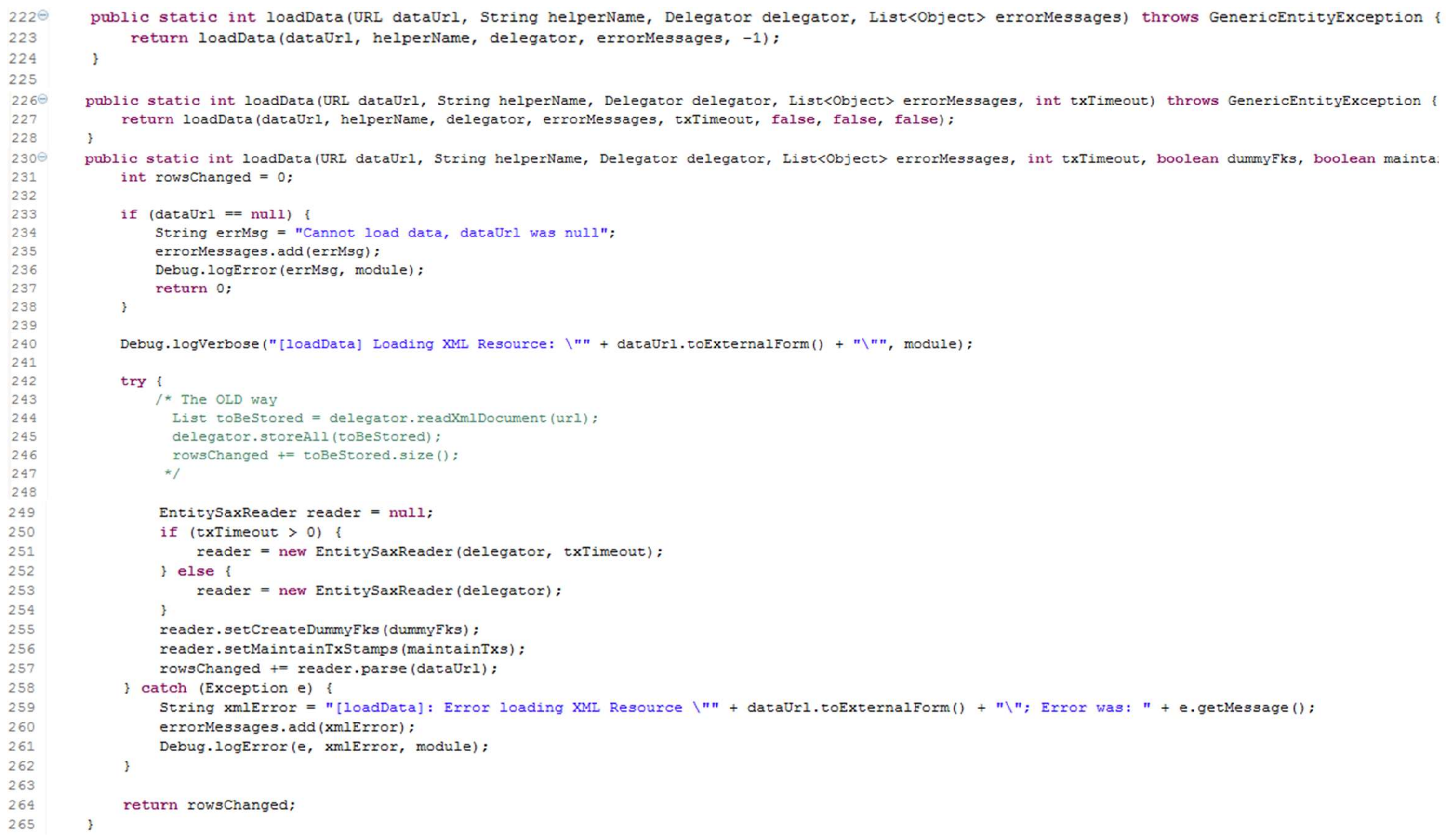
3.3 BaseContainer.getUrlByComponentList (…)



```
195     public static List<URL> getUrlByComponentList(String helperName, List<String> components, List<String> readerNames) {
196         List<URL> urlList = new LinkedList<URL>();
197         for (String readerName:  readerNames) {
198             List<String> loadReaderNames = new LinkedList<String>();
199             loadReaderNames.add(readerName);
200             for (String component : components) {
201                 urlList.addAll(getUrlList(helperName, component, loadReaderNames));
202             }
203         }
204         return urlList;
205     }
206
207     public static List<URL> getUrlByComponentList(String helperName, List<String> components) {
208         Datasource datasourceInfo = EntityConfig.getDatasource(helperName);
209         List<String> readerNames = new LinkedList<String>();
210         for (ReadData readerInfo :  datasourceInfo.getReadDataList()) {
211             String readerName = readerInfo.getReaderName();
212             // ignore the "tenant" reader if the multitenant property is "N"
213             if ("tenant".equals(readerName) && "N".equals(UtilProperties.getPropertyValue("general", "multitenant"))) {
214                 continue;
215             }
216
217             readerNames.add(readerName);
218         }
219         return getUrlByComponentList(helperName, components, readerNames);
220     }
```

[Rule 18]: The definition of this method with three input parameters is not even commented. In the second definition there is only a comment but it doesn't explain what the method does. Line 213 : [Rule 13] The length of the line is 117 characters and it's greater than 80.

3.4 BaseContainer.loadData(…)



```
222    public static int loadData(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages) throws GenericEntityException {
223         return loadData(dataUrl, helperName, delegator, errorMessages, -1);
224    }
225
226    public static int loadData(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout) throws GenericEntityException {
227        return loadData(dataUrl, helperName, delegator, errorMessages, txTimeout, false, false, false);
228    }
230    public static int loadData(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout, boolean dummyFks, boolean mainta
231        int rowsChanged = 0;
232
233        if (dataUrl == null) {
234            String errMsg = "Cannot load data, dataUrl was null";
235            errorMessages.add(errMsg);
236            Debug.logError(errMsg, module);
237            return 0;
238        }
239
240        Debug.logVerbose("[loadData] Loading XML Resource: \"" + dataUrl.toExternalForm() + "\"", module);
241
242        try {
243            /* The OLD way
244              List toBeStored = delegator.readXmlDocument(url);
245              delegator.storeAll(toBeStored);
246              rowsChanged += toBeStored.size();
247             */
248
249             EntitySaxReader reader = null;
250             if (txTimeout > 0) {
251                 reader = new EntitySaxReader(delegator, txTimeout);
252             } else {
253                 reader = new EntitySaxReader(delegator);
254             }
255             reader.setCreateDummyFks(dummyFks);
256             reader.setMaintainTxStamps(maintainTxs);
257             rowsChanged += reader.parse(dataUrl);
258         } catch (Exception e) {
259             String xmlError = "[loadData]: Error loading XML Resource \"" + dataUrl.toExternalForm() + "\"; Error was: " + e.getMessage();
260             errorMessages.add(xmlError);
261             Debug.logError(e, xmlError, module);
262         }
263
264         return rowsChanged;
265    }
```

[Rule 18]: These methods are not commented.

Line 222 : [Rule 14] The length of the line is 140 characters and it's greater than 120.

Line 226 : [Rule 14] The length of the line is 155 characters and it's greater than 120.

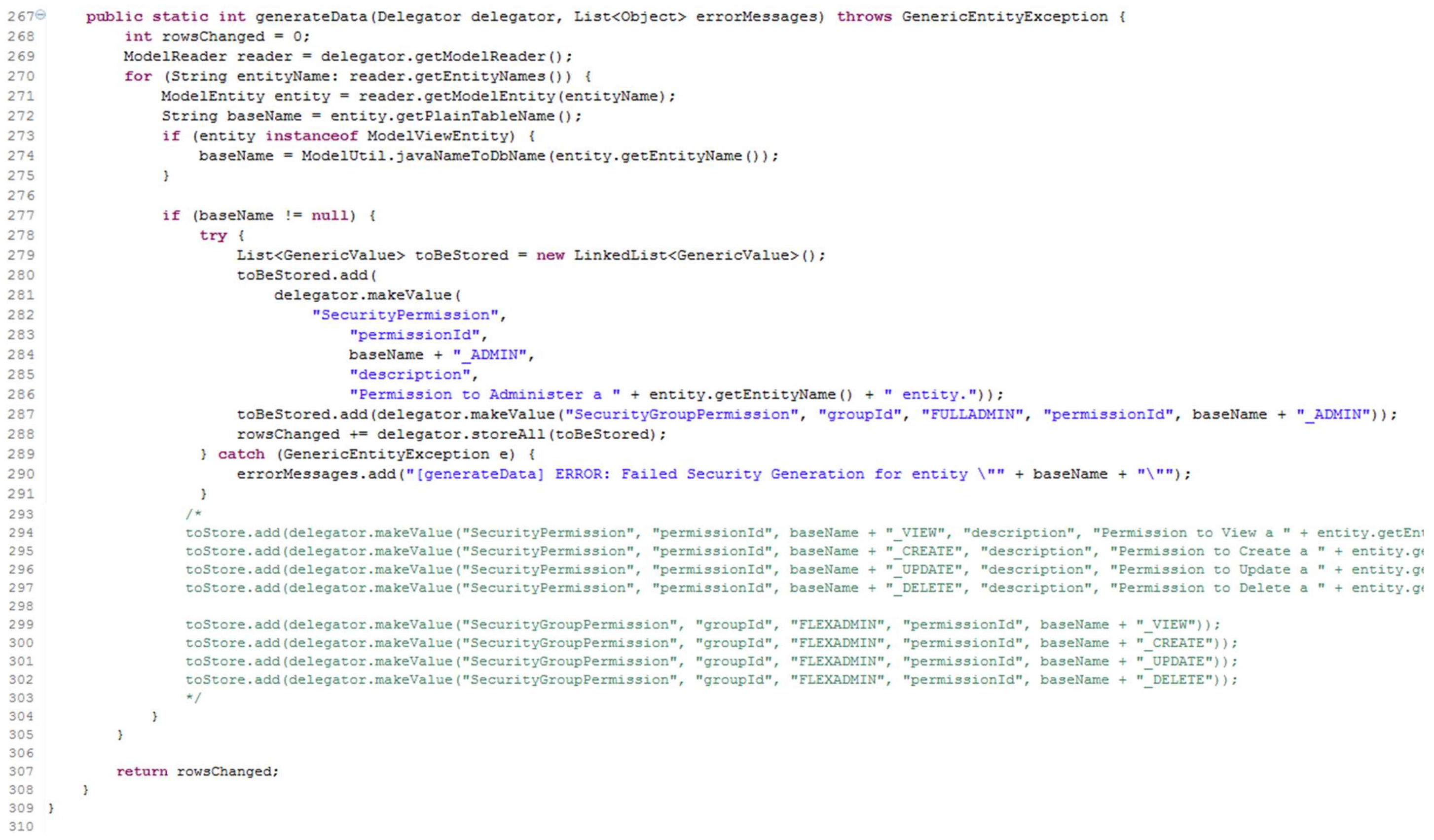
Line 230 : [Rule 14] The length of the line is 213 characters and it's greater than 120.

Line 240 : [Rule 13] The length of the line is 104 characters and it's greater than 80.

Line 259 : [Rule 14] The length of the line is 125 characters and it's greater than 120.

Lines 243-247 : [Rule 19] There is code commented out without the date when it can be removed

3.5 BaseContainer.generateData(…)



```
267     public static int generateData(Delegator delegator, List<Object> errorMessages) throws GenericEntityException {
268         int rowsChanged = 0;
269         ModelReader reader = delegator.getModelReader();
270         for (String entityName: reader.getEntityNames()) {
271             ModelEntity entity = reader.getModelEntity(entityName);
272             String baseName = entity.getPlainTableName();
273             if (entity instanceof ModelViewEntity) {
274                 baseName = ModelUtil.javaNameToDbName(entity.getEntityName());
275             }
276
277             if (baseName != null) {
278                 try {
279                     List<GenericValue> toBeStored = new LinkedList<GenericValue>();
280                     toBeStored.add(
281                         delegator.makeValue(
282                             "SecurityPermission",
283                                 "permissionId",
284                                 baseName + "_ADMIN",
285                                 "description",
286                                 "Permission to Administer a " + entity.getEntityName() + " entity."));
287                     toBeStored.add(delegator.makeValue("SecurityGroupPermission", "groupId", "FULLADMIN", "permissionId", baseName + "_ADMIN"));
288                     rowsChanged += delegator.storeAll(toBeStored);
289                 } catch (GenericEntityException e) {
290                     errorMessages.add("[generateData] ERROR: Failed Security Generation for entity \"" + baseName + "\"");
291                 }
293                /*
294                toStore.add(delegator.makeValue("SecurityPermission", "permissionId", baseName + "_VIEW", "description", "Permission to View a " + entity.getEnt
295                toStore.add(delegator.makeValue("SecurityPermission", "permissionId", baseName + "_CREATE", "description", "Permission to Create a " + entity.ge
296                toStore.add(delegator.makeValue("SecurityPermission", "permissionId", baseName + "_UPDATE", "description", "Permission to Update a " + entity.ge
297                toStore.add(delegator.makeValue("SecurityPermission", "permissionId", baseName + "_DELETE", "description", "Permission to Delete a " + entity.ge
298
299                toStore.add(delegator.makeValue("SecurityGroupPermission", "groupId", "FLEXADMIN", "permissionId", baseName + "_VIEW"));
300                toStore.add(delegator.makeValue("SecurityGroupPermission", "groupId", "FLEXADMIN", "permissionId", baseName + "_CREATE"));
301                toStore.add(delegator.makeValue("SecurityGroupPermission", "groupId", "FLEXADMIN", "permissionId", baseName + "_UPDATE"));
302                toStore.add(delegator.makeValue("SecurityGroupPermission", "groupId", "FLEXADMIN", "permissionId", baseName + "_DELETE"));
303                */
304             }
305         }
306
307         return rowsChanged;
308     }
309 }
310
```

[Rule 18]: This method is not commented

Line 267 : [Rule 13] The length of the line is 105 characters and it's greater than 80.

Line 287 : [Rule 14] The length of the line is 133 characters and it's greater than 120.

Line 290 : [Rule 13] The length of the line is 119 characters and it's greater than 80.

Lines 280-281 : [Rule 15] There is a line-break after a '('

Lines 293-303: [Rule 19] There is code commented out without the reason why this code is written there.

3.6 Overall annotations

[Rule 23]: JavaDoc wasn't used to produce the documentation of this code.

4 Other problems

Lines 282-286 : The indentation is not properly used.

We have noticed a fact in the following method:

- **public static int loadData**(URL dataUrl, String helperName, Delegator delegator, List<Object> errorMessages, int txTimeout, boolean dummyFks, boolean maintainTxs, boolean tryInsert)

HelperName and *tryInsert* are not used, we don't know if it was a conscious choice or not but it if the code shouldn't undergo changes they are useless.