



PowerEnJoy
Software Engineering II

Code Inspection

Giovanni Scotti, Marco Trabucchi

Document version: 1
January 10, 2017

Contents

Contents	1
1 Code Description	2
1.1 Assigned Classes	2
1.2 Functional Role of Classes	2
1.2.1 Apache OFBiz	2
1.2.2 OFBiz's DataFile Tools	3
1.2.3 The ModelDataFileReader.java class	3
2 Inspection Results	6
2.1 Notation	6
2.2 Issues	6
2.2.1 ModelDataFileReader Class	6
A Appendix	8
A.1 Software and tools used	8
A.2 Hours of work	8
Bibliography	9

Section 1

Code Description

1.1 Assigned Classes

The assigned class is the following:

- `ModelDataFileReader.java`

The class is located in the `org.apache.ofbiz.datafile` package of the Apache OFBiz project.

1.2 Functional Role of Classes

1.2.1 Apache OFBiz

The class to be reviewed is part of the *Apache OFBiz* open-source project.

Apache OFBiz is an open-source ERP (Enterprise Resource Planning) and CRM (Customer Relationship Management) system that provides a suite of applications that are useful to integrate and automate most business processes of an enterprise (e.g. accounting, e-commerce, project management, order processing...).

The applications provided in the suite are meant to be loosely coupled and are built on a common architecture. Because of this and the project being open-source, they can be customized easily.

The *Wikipedia* page on the project [4] gives some interesting information about the overall structure: *OFBiz* is structured in three distinct layers, namely the **Presentation Layer**, the **Business Layer** and the **Data Layer**.

The **Presentation Layer** is responsible for managing the components which take care of the presentation for each of the applications provided in the suite.

The **Business Layer** defines and manages the actual services to be provided to the users; it also manages the work-flows and simple methods involved in the logic of the applications in the suite, as well as the security issues behind them.

The **Data Layer** manages the data access, storage and interface to the database for the applications of the project in a relational way. The assigned class plays a key role for this layer, since it allows the import and management of external data as defined by the user.

1.2.2 OFBiz's DataFile Tools

Of all the functions provided by *Apache OFBiz*, the assigned class belongs to the one involved with data import and external data handling; this is provided by the DataFile Tool, which is specifically included in the `org.apache.ofbiz.datafile` package of the project.

The documentation for the DataFile Tool, as provided on the official *Apache OFBiz Project Open Wiki* [5] and in the official *javadoc* [6], describes the tool as follows:

"There is a data import tool in OFBiz called the DataFile tool. It uses XML files that describe flat file formats (including character delimited, fixed width, etc.) and parses the flat files based on those definitions. It uses a generic object to represent a row in the flat file. It includes features like a line type code for each line and can support hierarchical flat files (i.e. where parent/child relationships are implied by sub-records). An XSD is provided to describe how the data file definition XML file should look."

The tool is based on the concept of **definition files**, XML files following the XSD structure mentioned in the quotation above that contain one or more **data file definitions** each; **data file definitions**, in turn, are a key concept that represent the structure of the flat data files. Finally, another relevant concept is that of **data files**, which are nothing more but fixed-width or character delimited text files containing all data properly formatted.

1.2.3 The `ModelDataFileReader.java` class

The role of the assigned class within the DataFile Tools is that of reading XML definition files and creating `ModelDataFileReader` objects that are used by other classes in the package to access the subsiding flat data files. These

readers are specific to the URL of the XML definition file resource and hence only represent the portion of the database that is addressed by that XML structure.

The following example, taken from the *Apache OFBiz Project Open Wiki*, gives an idea of the structure specification to be followed when taking advantage of the DataFile Tools:

```
021196033702 ,5031BB GLITTER GLUE PENS BRIGH ,1 ,5031BB , 1, 299,
021196043121 ,BB4312 WONDERFOAM ASSORTED ,1 ,BB4312 , 1, 280,
021196055025 ,9905BB PLUMAGE MULTICOLOURED ,1 ,9905BB , 4, 396,
```

Figure 1.1: An example of a fixed-width text file to be managed with the DataFile Tools.

```
<?xml version="1.0" encoding="UTF-8" ?>

<data-files xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/datafiles.xsd">
  <data-file name="posreport" separator-style="fixed-length" type-code="text">
    <record name="tillentry" limit="many">
      <field name="tillCode" type="String" length="16" position="0"/>
      <field name="name" type="String" length="32" position="17"/>
      <field name="prodCode" type="String" length="12" position="63"/>
      <field name="quantity" type="String" length="8" position="76"/>
      <field name="totalPrice" type="String" length="8" position="85"/>
    </record>
  </data-file>
</data-files>
```

Figure 1.2: An example of XML structure compliant with the XSD definition mentioned above.

`ModelDataFileReader` objects are instantiated by classes that need access to the data files (e.g. `DataFile`) via the `getModelDataFileReader(URL readerURL)` static method which, depending on which readers are already instantiated, either returns a new reader or the reference to an existing one.

If a new reader has to be instantiated, the constructor method is called and the initialization of the actual `ModelDataFiles` - which are the objects containing the structure for the data files - takes place. This happens through

a call from the constructor to the `createModelDataFiles()` method, which takes care of filling the `modelDataFiles` map attribute of the reader with all the `ModelDataFile` objects associated with it.

The above operation is an iterative procedure over the - possibly many - data file definitions that takes advantage, within its code, of the `createModelDataFile(Element dataFileElement)` method, which is responsible for properly structuring and formatting the individual `ModelDataFile` objects to be inserted in the Map `modelDataFiles` attribute.

In turn, said procedure calls the `createModelRecord(Element recordElement)` method to perform a similar operation over the records (one or more) contained within a single `ModelDataFile`; the same happens within said method, which calls the `createModelField(Element fieldElement)` which creates the structure of single `ModelField` objects.

A `ModelRecord` is an object indicating the structure of the individual data blocks within the data files. A `ModelField` is an object representing the structure of the individual data "tuples" contained in each record.

The other methods of the class are interfacing methods, and provide a way for other classes to access the data structure encapsulated in the class by protecting the real data representation of textual files. In detail, there are:

- A `getDataFileNames()`, which returns a collection containing the names of the data files listed and described in the XML descriptor file;
- A `getDataFileNamesIterator()`, which returns an iterator on the String names of the same data files;
- A `getModelDataFile(String DataFileName)`, which returns a single `ModelDataFile` based on its name in the XML descriptor file;
- A `getModelDataFiles()`, which returns the map containing all the `ModelDataFiles` accessible by the reader.

It is interesting to note that the assigned class is marked as `final`, and hence can not be extended by other classes. This was done, most probably, as a design choice: the class was not designed for being extendible because the functionalities it provides must not be changed or, in fact, extended by other readers.

Section 2

Inspection Results

This chapter contains the result of the code inspection carried out on the assigned class and its methods. All the points of the given checklist [2] have been checked.

2.1 Notation

The following notations have been used to draw up this document:

- A specific line of code is referred as follows: **L.123**
- Specific items of the code inspection checklist [2] are referred as follows: **C1, C2, ... Cn**
- An interval of lines of code is referred as follows: **L.123-L.456**

2.2 Issues

2.2.1 ModelDataFileReader Class

1. **C18.** Comments are NOT used to adequately explain what the class, blocks of code and methods do except for the following methods:
 - `getDataFileNames()`
 - `getDataFileNamesIterator()`
 - `getModelDataFile(String dataFileName)`

Moreover the provided JavaDoc is very limited.

2. **C22.** JavaDoc for the public method `getModelDataFiles` is missing, therefore it is impossible to check whether the interface has been implemented consistently or not.
3. **C40.** No object is compared to another one. There are only objects compared to `null` by using the `"=="` operator correctly.
4. **C42.** The following lines contain log error messages that do not provide guidance or hints on how to correct the problem at all: **L.102**, **L.115**, **L.129-130**, **L.136-137**, **L.142-143**, **L.158-159**.

Appendix A

Appendix

A.1 Software and tools used

- L^AT_EX, used as typesetting system to build this document.
- draw.io - <https://www.draw.io> - used to draw diagrams and mock-ups.
- GitHub - <https://github.com> - used to manage the different versions of the document and to make the distributed work much easier.
- GitHub Desktop, the GitHub official application that offers a seamless way to contribute to projects.

A.2 Hours of work

The absolute major part of the document was produced in group work. The approximate number of hours of work for each member of the group is the following:

- Giovanni Scotti:
- Marco Trabucchi:

NOTE: indicated hours include the time spent in group work.

Bibliography

- [1] AA 2016/2017 Software Engineering 2 - *Project goal, schedule and rules*
- [2] AA 2016/2017 Software Engineering 2 - *Code Inspection Assignment - Task Description*
- [3] OFBiz - *Apache OFBiz Documentation*
- [4] Wikipedia article on OFBiz - https://en.wikipedia.org/wiki/Apache_OFBiz
- [5] *Apache OFBiz Project Open Wiki* article on the DataFile Tools - <https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz%27s+Data+File+Tools>
- [6] Official Apache OFBiz JavaDoc - <https://ci.apache.org/projects/ofbiz/site/javadocs/>