PowerEnJoy
Code Inspection

Stefano Brandoli (mat. 875633)
Silvia Calcaterra (mat. 874887)
Samuele Conti (mat. 875708)

February 4, 2017

Version 1.0

Contents

1 Introduction **3**

1.1 Classes Assigned . . . 3

1.2 Functional Role . . . 3

1.2.1 General Apache OFbiz Overview . . . 3

1.2.2 Framework . . . 4

1.2.3 Applications . . . 4

1.2.4 CmsEvent class . . . 5

1.2.5 CommonWorkers class . . . 6

1.3 Reference Links and Documents . . . 6

2 Code Inspection **7**

2.1 Notation . . . 7

2.2 Issues found . . . 7

2.2.1 CmsEvents class . . . 7

2.2.2 CommonWorkers class . . . 9

2.3 Other problems . . . 10

3 Software and tools used **11**

4 Effort spent **11**

1 Introduction

In this document we will perform the code inspection of some classes from the Apache OFBiz® source code.

1.1 Classes Assigned

These are the classes assigned to our team:

- **CmsEvents**
- **CommonWorkers**

1. The first one is located in **/apache-ofbiz-16.11.01/applications /content/src/main/java/org/apache/ ofbiz/content/cms/CmsEvents.java**.
2. The second one is located in **/apache-ofbiz-16.11.01/framework /common/src/main/java/org/apache/ofbiz/common/ CommonWorkers.java**.

1.2 Functional Role

1.2.1 General Apache OFbiz Overview

Apache OFBiz® is an open source product for the automation of enterprise processes that includes framework components and business applications for ERP, CRM, E-Business / E-Commerce, SCM, MRP, MMS/EAM.

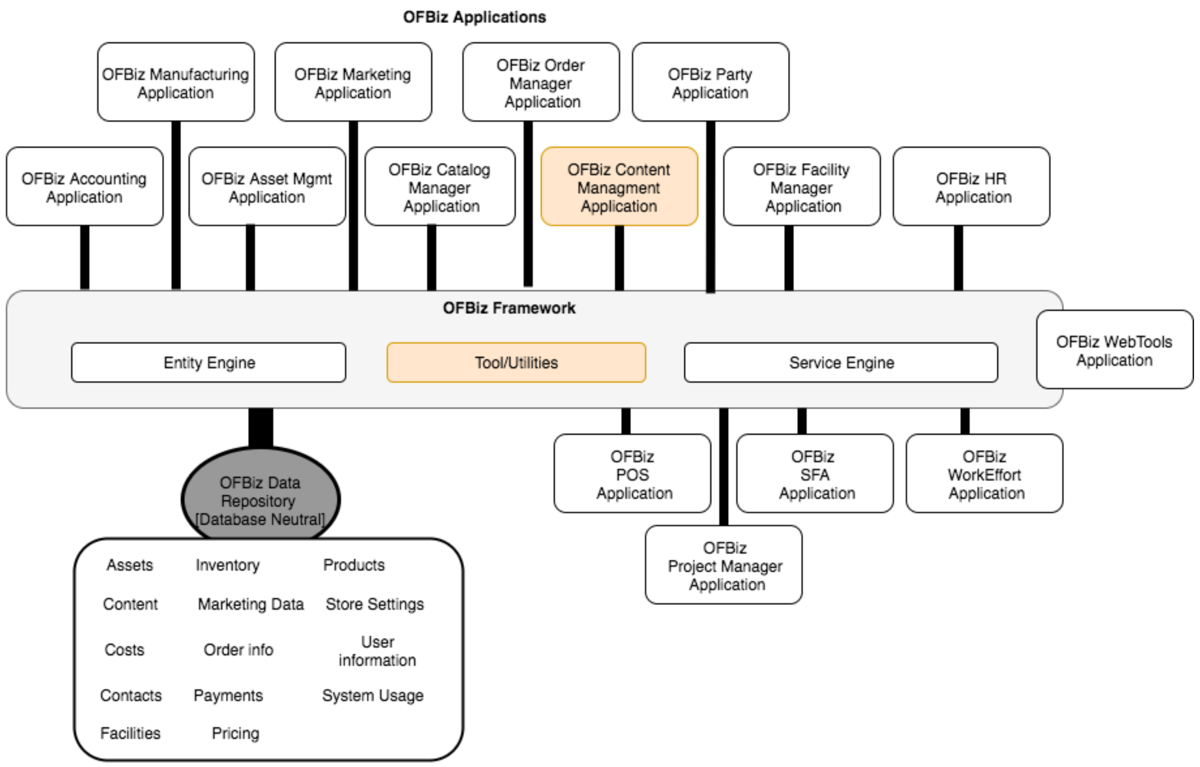
Apache OFBiz is a framework that provides a common data model and a rich set of business processes. All applications are built around a common architecture using common data, logic and process components.

There are a small number of component sets currently in OFBiz (from lowest-level to highest-level):

- Framework
- Applications
- Specialpurpose
- Hot-deploy

Dependencies between components in these sets normally go up the list. For example components in applications can depend on components in framework, but not vice versa.

The following image shows a general schema architecture of Apache OFBiz. Since the classes assigned to us belong to the **framework/common** and **applications/content/cms**, we will mostly concentrate on the areas highlighted in orange in the schema.

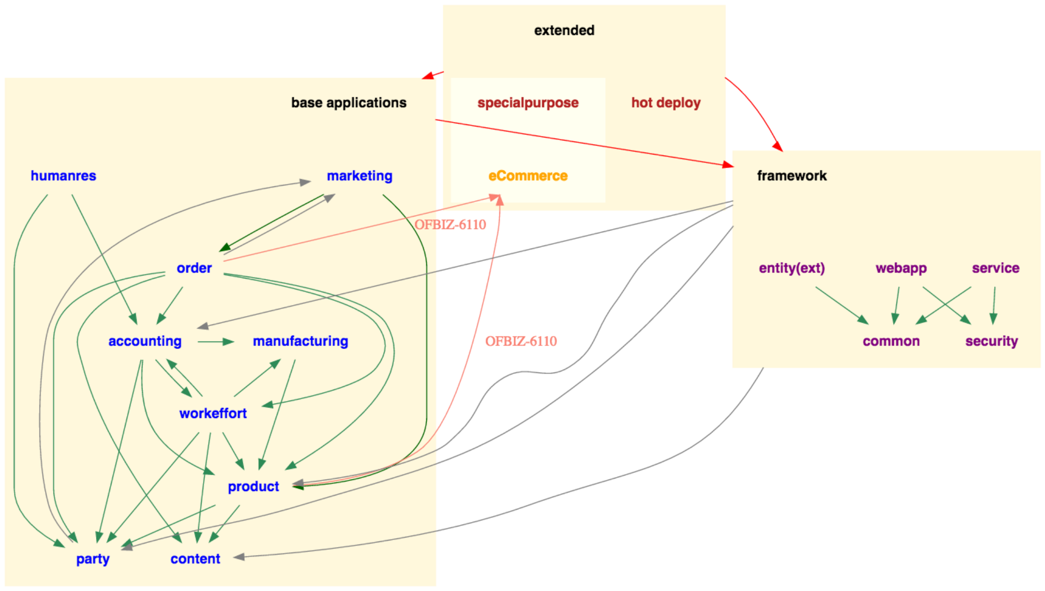

OFBiz Applications
OFBiz Manufacturing Application
OFBiz Marketing Application
OFBiz Order Manager Application
OFBiz Party Application
OFBiz Accounting Application
OFBiz Asset Mgmt Application
OFBiz Catalog Manager Application
OFBiz Content Managment Application
OFBiz Facility Manager Application
OFBiz HR Application
OFBiz Framework
Entity Engine
Tool/Utilities
Service Engine
OFBiz WebTools Application
OFBiz Data Repository [Database Neutral]
OFBiz POS Application
OFBiz SFA Application
OFBiz WorkEffort Application
OFBiz Project Manager Application
Assets
Inventory
Products
Content
Marketing Data
Store Settings
Costs
Order info
User information
Contacts
Payments
System Usage
Facilities
Pricing

1.2.2 Framework

The framework is made by the underlying data structures and utilities that supply the essential common features used by the base applications and the extended components. In this component set, we will perform the code inspection of the class **CommonWorkers.java**.

1.2.3 Applications

The base applications components depend on each other as shown in the following diagram (components higher in the diagram depend on components lower in the diagram). In this component set, we will perform the code inspection of the class **CmsEvent.java**. This class is located in the **content component** in the diagram below, taken from the Apache OFBiz documentation.



1.2.4 CmsEvent class

The **Content Management Application** in the OFBiz application component set supports the creation, modification and management of the content of web pages.

This application is the front-end user interface that allows a user, even with limited expertise, to add, modify, and remove content from a webpage without the intervention of a webmaster.

As stated in OFbiz documentation, the **CmsEvent** class is used to setup a website whose content can be managed using the OFBiz built-in features of the Content Management Application.

In particular, the CmsEvent class is used in relation with the definition of the **controller.xml** file of a web page. Here is an example coming from the OFbiz documentation:

```
<default-request request-uri="cms"/>
<request-map uri="main">
     <security https="false" auth="false"/>
     <response name="success" type="request" value="cms"/>
</request-map>
<request-map uri="cms">
     <security https="false" auth="false"/>
     <event type="java" path="org.ofbiz.content.cms.CmsEvents"
                              invoke="cms"/>
```

```
    <response name="success" type="none"/>
    <response name="error" type="view" value="error"/>
</request-map>
```

With the above setup, by default all the incoming requests will be dispatched to the CmsEvents class, which can be seen as part of a content delivery application (CDA). The event generated will use the data in the Content data model to generate the content of the webpage and will return it back to the browser. In this way it is possible to add new pages by just editing the data in the Content data model, without editing the controller.xml file.

1.2.5 CommonWorkers class

CommonWorkers is a class situated in the **ofbiz.common package**, contained inside the framework component set of the system: most of the import statements of the class refers to the **ofbiz.entity package**, indicating that CommonWorkers is probably a sort of utility class used by different classes in the entity package.

In particular, CommonWorkers is strictly connected to the Delegator class: every method in CommonWorkers receives an object Delegator as argument and returns a list of states or countries related to the delegator. The scope of CommonWorkers seems to be associated to a geographical utility provided to the **ofbiz.entity package**.

1.3 Reference Links and Documents

The following are the links to the main documentation resources we've used to try to understand the operational context of the classes assigned to us. Given the absense of JavaDocs, we had to extract the majority of the informations from the more general documentation of the Apache OFBiz project.

- https://ofbiz.apache.org/documentation.html
- https://cwiki.apache.org/confluence/display/OFBIZ/Component+and+Component+Set+Dependencies
- https://cwiki.apache.org/confluence/display/OFBIZ/Base+Application+stack
- https://cwiki.apache.org/confluence/display/OFBIZ/Content+Management
- https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Content+Management+How+to

2 Code Inspection

This section highlights all the issues that we have found during the code inspection of the classes **CmsEvents** and **CommonWorkers**. The issues found will follow the checklist provided. The other problems found which were not present in the checklist will be pointed out in the "Other Problems" section.

2.1 Notation

- A single line of code are denoted by **L.012**
- Intervals of lines of code are denoted as **L.100-200**
- Checklist inspection points: they are denoted as **Ci, i = 1...60**

2.2 Issues found

2.2.1 CmsEvents class

- **C1**:
  - L.64: "cms" is not a meaningful name for a method, since it has to handle a request/response interaction with HTTP servlets.
  - Other not meaningful names:
    - L.68 "session" is not a good name for an HTTP session
    - L.179 "rd" is too short to be meaningful for a RequestDispatcher
    - L.282 "ctx" is not a meaningful name for a ServletContext
    - L.283 "rh" is not a meaningful name for a RequestHandler
- **C5**: L.64 "cms" is not a verb
- **C7**:
  - L.76 and L.111 "_ERROR_MESSAGE_", is not valid constant name
  - L.123 "_CURRENT_CHAIN_VIEW_", is not valid constant name
- **C8**: indentation missing L.122-136, L.333.
- **C11**: if, while, do-while, try-catch, and for statements having only one statement and not surrounded by curly braces: L.233, L.247, L.251, L.371. L.196, L.198, L.424, L.427, 'if' construct with only one statement is not surrounded by curly braces.
- **C12**: L.258, L.276, L.296, L.356, L.374-375
- **C13**: multiple lines exceeding 80 characters, reducing readability: L.66-68, L.76, L.82, L.102-123, L.129, L.136, L.139, L.143-145, L.167, L.179, L.196, L.198, L.212, L.225-227, L.233-263, L.278-310, L.318-329, L.337, L.348, L.350, L.355-356, L.374-377, L.382, L.397, L.401-403, L.412-418, L.424. This is mainly due to:

  - The deep nesting of if-else statements, which are a lot since the class was not separated properly in sub functionalities.
  - Deep method calls chains.

- **C14**: multiple lines are also exceeding 120 characters, reducing significantly readability: L.76, L.106, L.108, L.111, L.117, L.167, L.252, L.261, L.291-310, L.412, L.318 - 329, L.348, L.350, L.355, L.397, L.401, L.412. This is mainly due to long chains of method calls which are put on the same line.
- **C16**: higher level breaks are not used in lines L.238, L.239, L.240
- **C18**:
  - the JavaDoc of this class is basically inexistent.
  - In the code of the class some comments are present, although not always useful.
- **C22**: it's quite hard to say whether the external interfaces are implemented correctly, since there is basically no JavaDoc for this class.
- **C23**: JavaDoc missing completely.
- **C27**:
  - The "cms" method is huge. It contains almost all the internal logic of the class. It could have been splitted in smaller methods in the CmsEvents class or in some other classes in the /content package, each of which handling a particular sub functionality.
  - Evident duplicated code: L.85-94
- **C29**: except from L.62, all the other variables are in the scope of methods, since they're strictly related to HTTP requests that vary every time.
- **C30**: L.277: method MapStack.create() refers to this:

  ```
  public static <K> MapStack<K> create() {
      MapStack<K> newValue = new MapStack<K>();
      // initialize with a single entry
      newValue.push();
      return newValue;
  }
  ```

  The create() method is used as a kind of static constructor which creates the object and performs a computation on it. The dev should have used directly the constructor and call another method to perform the required computation.
- **C32**: variables aren't initialized when declared: lines L.391.

- **C33**: L.82, L.83, L.97-100, L.102, L.165, L.179, L.197, L.207, L.208, L.235, L.282, L.283, L.291, L.295, L.334, L.335, L.391.

2.2.2 CommonWorkers class

- **C9**: L.150 : even though the class has used 4 spaces for indentation until this line, here two tabs are inserted.
- **C14**: Each of the lines in the following list exceeds 120 characters: L.58, L.59, L.67, L.82, L.93, L.96, L.127, L.149, L.150, L.177, L.180
- **C16**:
  - L.126-130: the canonical 8 spaces indentation after the higher-level break is not used.
  - L.134-138: the canonical 8 spaces indentation after the higher-level break is not used
  - L.150: the canonical 8 spaces indentation after the higher-level break is not used.
  - L.152-156: the canonical 8 spaces indentation after the higher-level break is not used.
- **C18**:
  - L.40: the declaration of the class lacks a comment description.
  - L.46: the first method of the class getCountryList lacks an explanatory comment.
  - L.91: the second method of the class getStateList lacks an explanatory comment. The lack of comments is probabily one of the biggest issue with the CommonWorkers class.
- **C23**:
  - L.40: the declaration of the class lacks a sufficiently explainatory javadoc.
  - L.46: the method getCountryList lacks a sufficiently explainatory javadoc.
  - L.91: the method getStateList lacks a sufficiently explainatory javadoc.
  - L.103: the method getAssociatedStateList lacks a sufficiently explainatory javadoc.
  - L.110: the method getAssociatedStateList lacks a sufficiently explainatory javadoc. Again, the lack of javadocs and documentation in general is one of the biggest issue with the CommonWorkers class.
- **C33**:
  - L.58, the variable exprs is not declared at the beginning of the block.

  - L.59, the variable countriesAvailable is not declared at the beginning of the block.
  - L.65, the variable countriesList is not declared at the beginning of the block.
  - L.119, the variable sortList is not declared at the beginning of the block.
  - L.121, the variable geoList is not declared at the beginning of the block.
  - L.146, the variable stateProvinceFindCond is not declared at the beginning of the block.

- **C53**:
  - L.69: in the catch block only a log is called, no significant action is taken to deal with the exception.
  - L.97: in the catch block only a log is called, no significant action is taken to deal with the exception.
  - L.159: in the catch block only a log is called, no significant action is taken to deal with the exception.

2.3 Other problems

- Some methods in the CmsEvent class have too many parameters. Maybe this is happening since the developers have decided to sacrifice the OO principles for the sake of performance. Howether, it would have been better to define some specific data structures to keep together the parameters that are always used together. An example can be found in CmsEvents class, line L.298.
- In the CmsEvents class, hardcoded strings are reused multiple times: if a modification will be needed, it will have to be propagated through all the class. It would have been better to use an enum or a specific abstract class to contain all the necessary parameters.

3 Software and tools used

- Git (https://github.com/) : for the version controlling of files shared between the team.
- Slack (https://slack.com/): used for group communication.
- GoogleDocs (https://www.google.it/intl/it/docs/about/): to write this document.
- Lyx (http://www.lyx.org/): to format this document.

4 Effort spent

Total hours of work for the Code Inspection:

- Stefano Brandoli: 10 hours
- Silvia Calcaterra: 10 hours
- Samuele Conti: 10 hours