Software Engineering 2 Project: PowerEnJoy

Code **I**nspection

Marco Ieni, Francesco Lamonaca, Marco Miglionico
Politecnico di Milano, A.A. 2016/2017

February 5, 2017
v1.0

Contents

Chapter 1

Assignment

In this document we will review a selected number of classes and methods. The goal of this inspection is to find bugs, lines of code that could be written in a better or more readable way and whatever may damage the maintainability and reusability of the code.

1.1 Assigned class

The assignment of the inspection is composed of the following classes:

- CmsEvents;
- OfbizCurrencyTransform.

Both classes belongs to the Apache OFBiz Project[1], an open source product for the automation of enterprise processes that includes framework components and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) and other business-oriented functionalities. The considered version is the 16.11.01.

The full path of the classes are respectively:

- `apache-ofbiz-16.11.01/applications/content/src/main/java/org/apache/ofbiz/content/cms/CmsEvents.java`
- `apache-ofbiz-16.11.01/framework/webapp/src/main/java/org/apache/ofbiz/webapp/ftl/OfbizCurrencyTransform.java`

[1] `https://ofbiz.apache.org/`

Chapter 2

Functional role

In the following, we will describe the functional role of the assigned set of classes.

2.1 Cms Events

The Cms Events class helps to manage the incoming http requests and to properly set the equivalent response.

In the public wiki, this class is mentioned in the page "OFBiz Content Management How to"[1]. Here, you can understand that this class is useful if you want to set-up a content driven website. In fact, to do a quickly initial set-up, you can add to the controller.xml file the following (or similar) entries:

```
1 <default-request request-uri="cms" />
2 <request-map uri="main">
3     <security https="false" auth="false" />
4     <response name="success" type="request" value="cms" />
5 </request-map>
6 <request-map uri="cms">
7     <security https="false" auth="false" />
8     <event type="java" path="org.ofbiz.content.cms.CmsEvents"
          invoke="cms" />
9     <response name="success" type="none" />
10    <response name="error" type="view" value="error" />
11 </request-map>
```

In this way, by default all the incoming requests will be dispatched to the CmsEvents event. This event will use the data in the Content data model to generate the content of the page and will return it back to the browser.

[1] https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Content+Management+How+to

In this way it will be possible to add new pages just editing the data in the Content data model and without editing the controller.xml file.

In the following we will see more details about each methods of the class.

2.1.1 cms method

Given an http request and an http response, the cms method checks if the request is valid or not and it assigns the proper status code to the http response. The status code is provided by the verifyContentToWebSite function, that can return three status codes: "OK", "NOT FOUND" and "GONE". If the status code is not "OK", then the function cms tries to find a specific error page for this website, concerning the status code. If this is not possible, then it tries to find a generic (not related to this current website) content error page concerning the status code.

Furthermore, if the status code found with the verifyContentToWebSite function was "OK" or we found an error page with one of the two previous researches, then we call the function ContentWorker.renderContentAsText. One of the parameter of this function is "content id". If the status code of the response is "OK", then the content id passed to the function will refer to the one specified in the request, otherwise it will refer to the error page that was found.

2.1.2 verifyContentToWebSite method

This method takes as input a delegator (used for queries), a Web Site ID and a Content ID. If the Content ID refers to an actual content of the website referred by the website id the function returns "OK". If not it checks if that content is a sub content of any publish point of the website, calling the third method of this class.

If the result is positive it returns "OK". Otherwise the function returns "GONE", if the content ID refers to a content no more active, otherwise returns "NOT FOUND".

2.1.3 verifySubContent method

This method takes as input a delegator (used for queries), and two content ID. It verifies if the first content is an actual sub content of the second or if it belongs to the tree with the second content as a root.

It returns "OK" if that is true, "GONE" if the content is no more active and "NOT FOUND" if there is no connection between the two contents.

2.2 OfbizCurrencyTransform

This section provides an overview of the functional role of the OfbizCurrency-Transform class and its methods. In this case we are considering a Freemarker Transform for content links. As defined by the Apache Freemarker website[2]:

"Apache FreeMarker is a template engine: a Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language (not a full-blown programming language like PHP). You meant to prepare the data to display in a real programming language, like issue database queries and do business calculations, and then the template displays that already prepared data. In the template you are focusing on how to present the data, and outside the template you are focusing on what data to present."

This class contains the declaration of a particular type of Freemarker, in particular its main function is to extract and transform the Java object received(Map of arguments), that in this case contain currency values and display them in an appropriate format. To do that the class implements the TemplateTransformModel, that offer a common transform model for the FreeMarker, and specify it for the specific case of currency.

In order to accomplish this task, inside the class the are 4 methods:

- getArg
- getAmount
- getInteger
- getWriter

The first three methods are very similar, in fact they receive the same input(a Map of arguments and a String that has the function of key) and return the argument corresponding to the key, if it exists. The only main difference between these three methods is the type of argument returned. The fourth method getWriter use the three methods defined before to get some specific values from the Map of arguments received in input and extract the following values: amount, isoCode and locale. Then the methods handles the rounding of the currency and finally it write on a buffer the data extracted in an appropriate format. This passage in made by overriding the methods write, flush and close of the TemplateTransformModel.

We include here the figure of the behaviour of the Apache Freemarker:

[2]http://freemarker. org/

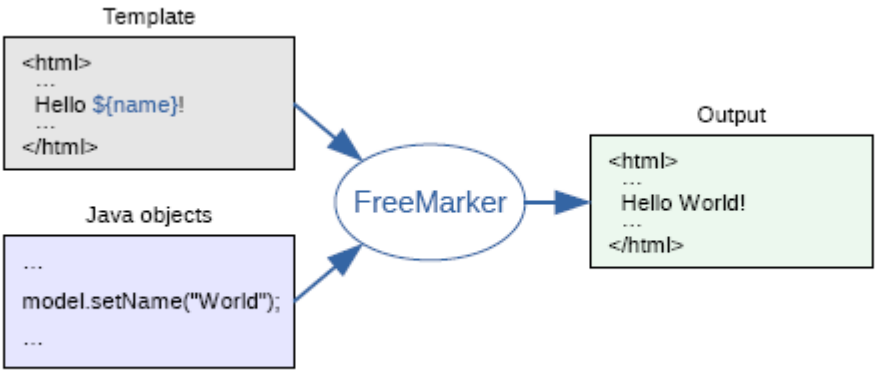

Figure 2.1: Apache FreeMarker

Chapter 3

Checklist Issues

In the following, we will list all the problems that we found by applying the given java checklist.

We will adopt the following notation:

- The items of the code inspection checklist [2] will be referred as follows: **C1**, **C2**, and so on.
- A specific line of code will be referred as follows: L1234.
- An interval of lines of code will be referred as follows: L1234-1256.

3.1 Cms Events

1. **C1** "rd" at L179, "ctx" at L282 and "rh" at L283 are not meaningful names.
2. **C5** The method name at L64 is not a verb. constant at L62 does not follow the naming convention, because it is written in lower case.
3. **C7** The constant at L62 does not follow the naming convention, because it is written in lower case.
4. **C8** After the else statement at L121, the indentation is incorrect until L332, because there 4 spaces less.
5. **C11** The if statements at L196, L198, L233, L247, L251, L371, L424 and L427 are not surrounded by curly braces.
6. **C13** L64, L66, L67, L76, L82, L102-103, L106, L108, L111, L117 L119, L122-123, L129, L136, L139, L143-145, L167, L179, L196, L198, L206,

L212, L221, L225-227, L233, L237-238 and L248 all exceed 80 characters.

7. **C14** L76, L106, L108, L111, L117, L167, L252, L261, L291, L295, L298, L300, L302, L306, L308, L310, L318, L322, L325, L329, L348, L350, L355 and L413 exceed 120 characters.

8. **C18** The code overall results poorly commented. In particular the lines in which the comments are strongly needed are: L106, L108 (there is a call to a function with 10 parameters that has not javadoc comment), L143, L167, L175, L181, L225, L237.

9. **C22** Javadoc is not present.

10. **C23** Javadoc comments for the class and the three methods of this class are missing (even the public one).

11. **C27** In the following we indicate the duplicate literals for which a constant should be defined and the number of times they are repeated:

    - "webSiteId": 10;
    - "_ERROR_MESSAGE_": 6;
    - "error": 7;
    - "WebSiteContent": 5;
    - "webSiteContentTypeId": 5;
    - "text/html": 4;
    - "contentId": 13;
    - "-fromDate": 4.

    At L85-L94 there is a piece of duplicate code, in fact it would be better to implement a method for the if-else block and call this method two times, the first for "targetRequest" and the second for "actualRequest".

    Furthermore, the cms method is very long and does too many things. It would be better to split the different tasks into different private methods.

12. **C32** At L391 the variable "responseCode" is not initialized.

13. **C33** The declarations at the following lines do not appear at the beginning of a block: L82-83, L97-98, L100, L102, L106, L165, L179, L207-208, L235, L291, L295, L334-335, L391.

14. **C43** At L248 there is a space before the character ':'.

15. **C44** At L366 the boolean variable hadContent is set true if the condition in the if clause at L365 is true (UtilValidate.isNotEmpty(publishPoints)), otherwise is set to false. The if statement is not necessary but could be written as hadContent = UtilValidate.isNotEmpty(publishPoints). The exact situation is repeated at L407.

16. **C53** The catch at L114, L146, L168, L228, L243, L269, L327, L345 only log the error.

3.2 Ofbiz Currency Transform

1. **C2** The one character variable "o" declared at L53, L72 and L91 is not used as temporary "throwaway" variable.

2. **C5** The name of method "formatCurrency" at L176, L178, L181 is not a verb.

3. **C7** The constant at L49 does not follow the naming convention, because it is written in lower case.

4. **C11** The if statements at L55, L78, L92, L143, L147, L169 are not surrounded by curly braces.

5. **C13** L55, L78, L92, L140, L142, L143, L169, L176, L178, L181 exceed 80 characters. In particular in L55, L78, L143, L169 this can be easily avoid.

6. **C18** The code overall results poorly commented. In particular the lines in which the comments are strongly needed are: L140, L142 and L175. Also, large pieces of code are left uncommented or only have extremely brief explanations of their purpose. This makes the process of verifying the behavior of the code very hard, as it's not clear what the expected result should be and what's the rationale behind it.

7. **C22** Javadoc is not present.

8. **C23** Javadoc comments for the class and the four methods of this class are missing.

9. **C27** The method "getWriter" L116-L188 is very long. This can be avoid by declaring some additional methods that encapsulate some

functions of this long method or by splitting the method, in this way the whole class will be more readable.

10. **C41** At L169 there is a grammatical error in the output, in fact "parms" should be renamed as "params".

11. **C43** At L55, L78, L92 there is a space before the character ':'.

12. **C53** The catch at L132, L183 only log the error.

Chapter 4

Other problems

In the following, we will list all the problems that we found without applying the checklist.

4.1 Cms Events

- The return value of the method cms is used only to communicate the outcome of the function. In fact it is a string and the only values that can be returned are "error", "success" or null. We know that in java this is a bad practice, because if you want to signal an error you should not return a special value, but throwing an exception. However, even in the case when you want to return an "error" or "success" value, it would be better to return a Boolean.
- The assignments at L69, L100 and L207 are useless.
- The following lines nest more than 3 if/for/while/switch/try statements: L131, L175, L196, L198, L200, L211, L224, L232, L259, L288, L317, L383, L416.
- The catch at L185 can be combined with the one at L182, which has the same body.

4.2 Ofbiz Currency Transform

- The following line nest more than 3 if/for/while/switch/try statements: L143.
- At L149 the if statement can be merged with the enclosing one.

- At L157 the array designator can be moved from the variable to the type(minor error).

Appendix A

Appendix

A.1 Used software and tools

- LaTeX [1], for typesetting this document.
- Texmaker[2], for the writing of this document.
- GitHub[3] for version control and distributed work.
- GitHub desktop[4] used to collaborate in the team and to keep track of the changes.
- Eclipse[5] used to analyze the code.

A.2 Work hours

The statistics about commits and code contribution are available on the GitHub repository of the project[6].

These are our estimation of the work hours spent on this project:

- Marco Ieni: 14 hours
- Francesco Lamonaca: 14 hours
- Marco Miglionico: 14 hours

[1] https://www.latex-project.org/
[2] http://www.xm1math.net/texmaker/
[3] https://github.com/
[4] https://desktop.github.com/
[5] https://eclipse.org/
[6] https://github.com/marcomiglionico94/Software-Engineering-2-Project