Code Inspection

Luca Marzi
Valeria Mazzola
Federico Nigro

February 5, 2017

Contents

1 Introduction **2**
1.1 Purpose and scope 2

2 Subject of the inspection **2**

3 Functional role of assigned set of classes **2**
3.1 MultiSiteRequestWrapper.java 3
3.2 BirtViewHandler.java 3

4 List of issues **4**
4.1 MultiSiteRequestWrapper.java 4
4.1.1 Naming conventions 4
4.1.2 Indention 5
4.1.3 Braces 5
4.1.4 File Organization 5
4.1.5 Wrapping lines 5
4.1.6 Comments 6
4.1.7 Java source files 6
4.1.8 Package and Import statements 6
4.1.9 Class and Interface declarations 7
4.1.10 Initialization and Declarations 7
4.1.11 Method calls 7
4.1.12 Arrays 7
4.1.13 Object comparison 8
4.1.14 Output format 8
4.1.15 Computation, Comparisons and Assignments 8
4.1.16 Exceptions 8
4.1.17 Flow of Control 11
4.1.18 Files 11
4.2 BirtViewHandler.java 11
4.2.1 Naming conventions 11
4.2.2 Indention 11
4.2.3 Braces 11
4.2.4 File Organization 12
4.2.5 Wrapping lines 12
4.2.6 Comments 12
4.2.7 Java source files 12
4.2.8 Package and Import statements 12
4.2.9 Class and Interface declarations 12
4.2.10 Initialization and Declarations 13
4.2.11 Method calls 13
4.2.12 Arrays 13
4.2.13 Object comparison 13
4.2.14 Output format 13

4.2.15 Computation, Comparisons and Assignments 14
4.2.16 Exceptions 14
4.2.17 Flow of Control 14
4.2.18 Files 14

1 Introduction

1.1 Purpose and scope

This document aims to report on the quality status of the code the project Apache OFBiz.

Apache OFBiz® is 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), E-Business / E-Commerce, SCM (Supply Chain Management), MRP (Manufacturing Resource Planning), MMS/EAM (Maintenance Management System/Enterprise Asset Management). Commercial software of such kind can be quite expensive, and the Apache OFBiz framework is currently being used by a great number of businesses directly, as well as extended and integrated by a number of software projects, commercial and non commercial, under a wide variety of software licenses.

2 Subject of the inspection

This document is about the inspection of these specific classes of the project Apache OFBiz:

Namespace Pattern	Name
apache-ofbiz-16.11.01/specialpurpose/ cmssite/src/main/java/org/apache/ ofbiz/cmssite/multisite/	MultiSiteRequestWrapper.java
apache-ofbiz-16.11.01/specialpurpose/ birt/src/main/java/org/apache/ofbiz/ birt/webapp/view/	BirtViewHandler.java

3 Functional role of assigned set of classes

In this section of the document we are going to provide an overall description of the functional role that we have identified for the class cluster of interests (described in the previous section).

The functional analysis will consider the different classes separately and will take into account all the available documentation about the overall project.

3.1 MultiSiteRequestWrapper.java

As the name of the class properly suggests, this class represent a wrapper that allows to access to a specific functionality by means of a different implementation of interface with respect to the one provided by the functionality itself.

In particular, the MultiSiteRequestWrapper provides access to the functionalities specified by the interface HttpServletRequest. We can infer this by looking at the constructor method of the MultiSiteRequestWrapper, which requires an object of type HttpServletRequest to instantiate a private and final attribute of the class.

This object of type HttpServletRequest is then used to provide most of the functionalities of the wrapper; in fact, most of the functionalities are directly provided by the wrapped object.

Inside the entire project we have found out only one reference to the MultiSiteRequestWrapper class (WebSiteFilter.java) hence we believe that this wrapper is meant as a tool for third-part applications that are built on the top of the Apache OFBiz® architecture.

3.2 BirtViewHandler.java

The BirtViewHandler class provides the functionalities specified by the ViewHandler interface. As seen in the javadoc, the main functionality of such interface is to render web pages.

As a view handler, the class can be used in utilizing software to implement web-based Views (in a MVC like architecture). In particular, the Apache OFBiz project aims at aiding the development of enterprise software such as ERP, CRM, SCM, MPR and MMS/EAM.

4 List of issues

4.1 MultiSiteRequestWrapper.java

4.1.1 Naming conventions

Issue number	Line of Code	Description
1	51	Private class attribute “request” should be renamed in “ _request”
2	76	Method parameter “arg0” should be renamed with a meaningful name
3	77	Method parameter “arg0” should be renamed with a meaningful name
4	91	Method parameter “arg0” should be renamed with a meaningful name
5	96	Method parameter “arg0” should be renamed with a meaningful name
6	164	Method parameter “arg0” should be renamed with a meaningful name
7	194	Method parameter “arg0” should be renamed with a meaningful name
8	199	Method parameter “arg0” should be renamed with a meaningful name
9	259	Method parameter “arg0” should be renamed with a meaningful name
10	274	Method parameter “arg0” should be renamed with a meaningful name
11	289	Method parameter “arg0” should be renamed with a meaningful name
12	309	Method parameter “arg0” should be renamed with a meaningful name
13	334	Method parameter “arg0” should be renamed with a meaningful name
14	339	Method parameter “arg0”and “arg1” should be renamed with a meaningful name
15	344	Method parameter “arg0” should be renamed with a meaningful name
16	349	Method parameter “arg0” should be renamed with a meaningful name
17	354	Method parameter “arg0” should be renamed with a meaningful name
18	364	Method parameter “arg0”and “arg1” should be renamed with a meaningful name
19	404	Method parameter “arg0”and “arg1” should be renamed with a meaningful name

4.1.2 Indention

Issue number	Line of Code	Description
20	117	The return statement shall be brought at the same indention level of the if statement of line 111.

4.1.3 Braces

Issue number	Line of Code	Description
21	58	The implementation of the method is on the same level of its declaration: neither “Allman” nor “Kernighan” styles are applied.
23	118	The “else” statement should be brought to the next line, and so all the consecutive lines of code (according to the “Allman” and “Kernighan” styles too).

4.1.4 File Organization

Issue number	Line of Code	Description
24	18	No comments are applied for the description of the used packages section.
25	18	No blank lines of space are applied between the first description regarding the Licence and the declaration of the packages used in the Class.
26	48	No comments are applied for the description of the class section.
27	50	No comments are applied for the description of the attribute section.
28	56	No comments are applied for the description of the methods section.
29	(from 56)	No comments are applied for the description of the single method section (in particular the what to do for the override property, why overriding/deprecating the method, principal methods specifications and purposes).

4.1.5 Wrapping lines

Issue number	Line of Code	Description
30	117	The return statement shall be brought at the same indention level of the if statement of line 111.

4.1.6 Comments

Issue number	Line of Code	Description
31	(from 56)	No comments are applied for the description of the single method section (in particular the what to do for the override property, why overriding the method, principal methods specifications and purposes).
32	104	A brief explanation for the method "getPathInfo()" of line 106 can be useful for better understanding the meaning of the method which seems long.
33	182	A brief explanation of why the method "isRequestedSessionIdFromUrl()" is now deprecated and how to find the same solution now; can be useful for better maintaining the code in the future and the decision to take towards this method.
34	288	A brief explanation of why the method "getRealPath(String arg0)" is now deprecated and how to find the same solution now; can be useful for better maintaining the code in the future and the decision to take towards this method.

4.1.7 Java source files

No issues to be reported.

4.1.8 Package and Import statements

No issues to be reported.

4.1.9 Class and Interface declarations

Issue number	Line of Code	Description
35	48	No documentation comment is left for the class "MultiSiteRequestWrapper"
36	—	There is not a specific, functional, order for the implementation of the methods. It is important to maintain in the class a priority order for the methods inside it. A guide line for the order to give to the methods could be: 1. declaring the login / logout methods 2. declaring the "getters" methods 3. declaring the "stating methods" (like "isAsyncSupported()" and so on) 4. declaring the "setters" methods.
37	106	the "getPathInfo()" method seems too long. Verify if there could be another way for refactoring it (declaring for instance a private method in which implementing diffently the same code).

4.1.10 Initialization and Declarations

Issue number	Line of Code	Description
38	53	The private final attribute "request" of the class is initialized, by the constructor method, with the reference to an object of type HttpServletRequest without checking if the reference is a Null pointer. The constructor should check this condition (and raise an InvalidParameterException) in order to avoid all the possible "NullPointerException" that may raise from the subsequent method calls/attribute accesses on the attribute "request".

4.1.11 Method calls

No issues to be reported.

4.1.12 Arrays

No issues to be reported.

4.1.13 Object comparison

Issue number	Line of Code	Description
38	109	The second term of the comparison (“pathInfo != null”) should be changed in “StringUtils.isEmpty(pathInfo)” or “pathInfo.equals(null)”.
39	111	“nextPathSegmentStart == -1” should be changed in “nextPathSegmentStart.equals(-1)”.
40	113	“nextPathSegmentStart == -1” should be changed in “nextPathSegmentStart.equals(-1)”.

4.1.14 Output format

No issues to be reported.

4.1.15 Computation, Comparisons and Assignments

Issue number	Line of Code	Description
41	118	The else statement is unnecessary for the this case. Return the “pathInfo” variable directly cutting the else block.
42	111-115	There could be a better way to refactor the code. For instance creating a lighter solution coupling the last two if staments and reorganizing the code.

4.1.16 Exceptions

Issue number	Line of Code	Description
43	107	Manage the exception that can be generated by the casting of variable. It is convenient to do it by a try catch block or throwing the exception from the method. In this last case is also important to understanding where the exception will be managed inside the program.

44	229	IOException should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct.
45	284	IOException should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct.
46	344	UnsupportedEncodingException should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct.
47	349	ServletException, IOException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct. This has to be valid for both the found exceptions.

48	354	ServletException, IOException, IllegalStateException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct. This has to be valid for all the found exceptions.
49	359	ServletException, IOException, IllegalStateException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct. This has to be valid for all the found exceptions.
50	364	ServletException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct.
51	369	ServletException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct.

52	409	ServletException, IOException, should be caught, it can be done in two ways: 1. returning something specific to be used by the invokator 2. verifying if all invokators handle the exception and if the level in which the exception is handled is correct. This has to be valid for both the found exceptions.
53	—	Verify for each important method of the interface if there is the necessity of throwing/managing the NullPointerException for the returned parameters and, for single case, where they will be in the program.

4.1.17 Flow of Control

No issues to be reported.

4.1.18 Files

No issues to be reported.

4.2 BirtViewHandler.java

4.2.1 Naming conventions

No issues to be reported.

4.2.2 Indention

Issue number	Line of Code	Description
1	90	Indentation is inconsistent: should be 12 spaces instead of 13

4.2.3 Braces

No issues to be reported.

4.2.4 File Organization

Issue number	Line of Code	Description
2	96	An additional line may be used to visually separate semantically related parts of the code
3	116	Line exceeds 120 characters length
4	119	An additional line may be used to visually separate the *try* and *catch* blocks

4.2.5 Wrapping lines

No issues to be reported.

4.2.6 Comments

Issue number	Line of Code	Description
5	50	A class documentation comment is missing

4.2.7 Java source files

No issues to be reported.

4.2.8 Package and Import statements

No issues to be reported.

4.2.9 Class and Interface declarations

Issue number	Line of Code	Description
6	-	At least one class constructor should be declared explicitly. If the constructor is not meant to be called, it should be declared as private.

4.2.10 Initialization and Declarations

Issue number	Line of Code	Description
7	85	Variable declarations should be at the beginning of a block
8	88	Variable declarations should be at the beginning of a block
9	90	Variable declarations should be at the beginning of a block
10	97	Variable declarations should be at the beginning of a block
11	103	Variable declarations should be at the beginning of a block
12	109	Variable declarations should be at the beginning of a block
13	115	Variable declarations should be at the beginning of a block
14	116	Variable declarations should be at the beginning of a block

Important note Although the variable declarations mentioned above do not follow the rule of appearing at the beginning of a block, it is true that their current position in the code concurs in achieving a greater clarity and cleanness in the code. These issues have been included in the present document for completeness and do not constitute a severe infraction of coding standards.

4.2.11 Method calls

No issues to be reported.

4.2.12 Arrays

No issues to be reported.

4.2.13 Object comparison

No issues to be reported.

4.2.14 Output format

Issue number	Line of Code	Description
15	120	Error message should be phrased in a more understable fashion
16	124	Error message should be more descriptive of the problem

4.2.15 Computation, Comparisons and Assignments

No issues to be reported.

4.2.16 Exceptions

No issues to be reported.

4.2.17 Flow of Control

No issues to be reported.

4.2.18 Files

No issues to be reported.