Code Inspection Document

Version 1.0
5 February 2017

Authors:
Patrizia Porati, Tommaso Sardelli

POLITECNICO
MILANO 1863

Contents

1 Introduction

This document report on the quality status of assigned code extracts using the checklist for Java Code Inspection reported on the Appendix of *Code Inspection Assignment Task Description* document.
Code inspection is an examination of computer source code. In particular it aims to find mistakes overlooked during the development phase in order to improve the quality of the software and the development' skills. Reviews are done in various forms such as pair programming, informal walkthroughs, and formal inspections.

This document is divided into n main parts:

Section 1: Introduction this first section gives a short introduction to the Code Inspection document

Section 2: Assigned classes this section is to state the classes that we are going to analyze

Section 3: Functional role this section contains the description of the functional role we have identified for the classes specifies in the previous section

Section 4: List of issues in this section we report the fragment of code that do not fulfill some points in the checklist

Section 7: Working Hours this sections contains the result of our effort, quantified in the number of hours we have needed in order to develop this document.

Section 8: References a list of all the references we took into account when writing this document.

2 Assigned classes

In this chapter we are going to state the classes assigned to us for the code inspection activity.
In particular, there is only one java class which we are going to analyze: that is **VisitHandler.java**, located in the following file *"../apache-ofbiz-16.11.01/framework/webapp/src/main/java/org/apache/ofbiz/webapp/stats/VisitHandler.java"* from a release of the **Apache OFBz project**.
Apache OFBz is an open source product for the automation of enterprise processes that includes framework component and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) and other business-oriented functionalities.

Here are the four methods contained in the **VisitHandler** class:

- `setUserLogin (HttpSession session, GenericValue userLogin, boolean userCreated)`, line 61
- `getVisitId(HttpSession session)`, line 99
- `getVisit(HttpSession session)`, line 78
- `getVisitor(HttpServletRequest request, HttpServletResponse response)`, line 206

3 Functional roles of the assigned class

The VisitHandler class is created to expose static methods useful for website statistics. As the class name suggests, it is used to store informations in the database about users visiting the website. It does so inspecting the current HttpSession and extracting a visitor, that is the logged user visting the website, and a visit, a record of a single visit by a user.

The two entities are loaded with additional information about the current connection. For what concerns the visit, said information includes: webapp name, client locale, client request, client referrer and client user agent. Also network information are stored, like the client hostname and ip address.

The visitor instead is associated to an existing one using cookies informations or a new one is created if no association can be found.

When all informations are collected, the entities as stored in the databse permanently.

4 List of issues

Here is a comprehensive list of all issues we have found according to the checklist reported in the assignment. For the sake of clarity we decided to write down only relevant and meaningful points and to omit the other ones.

4.1 Indentation

[9] Tabs
Four spaces are consistently used to indent except at line 208, where there is a tab.

4.2 Braces

[10] Bracing style
The "Kernighan and Ritchie" style is consistently used.

[11] One statement
All `if`, `while`, `do-while`, `try-catch` and `for` statements that have only one statement to execute should be surrounded bu curly braces.
Line 61 is like this:

```
if (userLogin == null) return;
```

but should be like this:

```
if (userLogin == null) {
        return;
}
```

In the same way, line 229 should be:

```
if (Debug.verboseOn()) {
        Debug.logVerbose("Cookies:" + cookies, module);
}
```

Line 239 should be:

```
if (Debug.infoOn()) {
        Debug.logInfo("Found visitorId [" + cookieVisitorId + "] in cookie",
        module);
}
```

And finally line 257 should be:

```
if (Debug.infoOn()) {
        Debug.logInfo("The visitorId [" + cookieVisitorId +
        "] found in cookie was invalid, creating new Visitor with ID [" +
        visitor.getString("visitorId") + "]", module);
}
```

4.3 File Organization

[13] 80 characters limit
There are a lot of line that exceed 80 characters:
18, 56, 61, 111, 120, 122, 125, 127, 131, 133, 134, 135, 135, 137, 139, 142, 147, 150, 151, 152, 154, 155, 157, 159, 160, 161, 165, 167, 169, 171, 172, 176, 182, 183, 190, 199, 206, 209, 212, 218, 219, 220, 224, 226, 229, 232, 239, 242, 247, 252, 254, 255, 256, 257, 260, 267, 270, 271 and 280.

[14] 120 characters limit
There are some lines that also exceed 120 characters:
131, 142, 150, 151, 152, 169, 209, 224, 239, 252, 254, 257, 260 and 270.

4.4 Comments

[18] Adeguate comments
In this class there are very few comments that explain methods and blocks of code, but in general the code is comprehensible.

4.5 Java Source Files

[23] JavaDoc
In this class, there are very few JavaDoc comments and they are inaccurate.

4.6 Initialization and Declarations

[28] Visibility
`VisitHandler` class has only static methods and variables but doesn't have a private constructor.

[33] Declarations appear at the beginning of blocks

- `visit` variable declared at line 78 is not at the beginning of the block.

```
    [...]
74 } catch (GenericEntityException e) {
75 Debug.logError(e, "Could not update visitor: ", module);
76 }
77
78 GenericValue visit = getVisit(session);
    [...]
```

- `visitor` variable declared at line 163 is not at the beginning of the block.

```
160 visit.set("clientUser", session.getAttribute("_CLIENT_REMOTE_USER_"));
161
162 // get the visitorId
163 GenericValue visitor = (GenericValue) session.getAttribute("visitor");
164 if (visitor != null) {
```

4.7 Computation, Comparisons and Assignments

[44] "Brutish programming"
The following three lines contain a triple way if statement inside the method invocations itself. This makes the whole line pretty long and the code less readable. It would probably have been better to extract this operation in separate methods with a better interface.

```
150 if (initialRequest != null) visit.set("initialRequest",
       initialRequest.length() > 250 ? initialRequest.substring(0, 250)
       : initialRequest);
151 if (initialReferrer != null) visit.set("initialReferrer",
       initialReferrer.length() > 250 ? initialReferrer.substring(0, 250)
       : initialReferrer);
152 if (initialUserAgent != null) visit.set("initialUserAgent",
       initialUserAgent.length() > 250 ? initialUserAgent.substring(0, 250)
       : initialUserAgent);
```

4.8 Exceptions

[53] Check that the appropriate action are taken for each catch block.
While we think that exceptions are caught properly, some log messages saved within this blocks are a bit vague.

```
74 Debug.logError(e, "Could not update visitor: ", module);
94 Debug.logError(e, "Could not update visit: ", module);
190 Debug.logError(e, "Could not create new visit:", module);
247 Debug.logError(e, "Could not create new visitor:", module);
```

Furthermore, a message with a context similar to the previous ones is logged as warning instead of error.

```
280 Debug.logWarning(new Exception(), "Could not find or create the visitor...", module);
```

5 Effort spent

Section	Working hours	Author
Introduction	1	Patrizia Porati
Assigned classes	1	Patrizia Porati
Functional role	2	Tommaso Sardelli
Issues	8	Patrizia Porati, Tommaso Sardelli
Document review	3	Patrizia Porati, Tommaso Sardelli

6 Reference documents

- The assignemnt description: *Code Inspection Assignment Task Description.pdf*
- The assignment of the classes: *Code Inspection Assignment Table.xlsx*