Code Inspection Document- PowerEnJoy

POLITECNICO
MILANO 1863

Version 1.0

- Bagna Francesco Matteo (mat. 878556)
- Barletta Carmen (mat. 877129)

Commit date: 05/02/2017

Summary

1: Classes assigned

We had assigned the class WorkEffortContentWrapper, which can be found in

/apache-ofbiz-16.11.01/applications/workeffort/src/main/java/org/apache/ofbiz/workeffort/content/WorkEffortContentWrapper.java

2: Functional role of assigned classes

2.1: Role of the classes

The class WorkEffortContentWrapper provides some methods to extract some information (as the id, the data, the name, the date) of the most current content data of a certain type. It tries to pull out these info, by querying some database entities, when for example there is a request from an Http Servlet that wants to display this info somewhere.

2.2: Proof of the roles

The Javadoc at the beginning of the class says this. Moreover, the methods of the class are for the most part used to get some item, like the contents names or IDs. It is so clear that the class returns items that some other classes might need, especially work effort contents.

3: Issues from checklist

3.1: Naming Conventions

1) The method get (String, boolean, String) on line 80, hasn't got a very meaningful name, indeed its name doesn't clarify the fact that this methods returns a String representing the WorkEffortContent.
   Also the method **public** StringUtil.StringWrapper get (String contentTypeId, String encoderType) on line 89 has the same name but it does something different, because it *gets the most current content data by the defined type.*
2) No one-character values are used
3) The class name begins with a noun in capitalized letter
4) The class implements an interface "ContentWrapper" whose name is in capital letter
5) All methods are getters and one method's name starts with the word "make", so also the conventions about the methods' names has been respected.
6) The four attributes of this class start with a lowercase first letter and all the remaining letters in the variables have their first letter capitalized as (e.g mimeTypeId), but they don't begin with an underscore
7) The constant String module on line 56, is not declared in uppercase letters, but String CACHE_KEY_SEPARATOR on line 57, respects the default rule.

3.2: Indention

8) The indentation is consistent and correct in all the lines of the class
9) No tabs are used to indent

3.3: Braces

10) The WorkEffortContentWrapperWrapper.java has been developed using a consistent bracing style. In particular It has been used a "Kernighan and Ritchie" style (first brace is on the same line of the instruction that opens the new block).
11) All if, while, do-while, try-catch, and for statements that have only one statement to execute are surrounded by curly braces.

3.4: File Organization

12) Blank lines and optional comments are used to separate imports and class (even if there are also blank lines between the imports lines). Only for methods starting on line 89,98,112 and 136 the Javadoc is provided
13) 275, 279,329,204 and many others lines exceed have a length greater than 120 characters. This is a problem because it reduces significantly the code readability
14) In the previous line, has already been said that there're lines with more than 120 characters

3.5: Wrapping Lines

15) Line breaks occur after a comma or an operator

16) No higher-level breaks are used
17) The new statements are aligned with the beginning of the expression at the same level as the previous line

3.6: Comments

18) Unfortunately the code has a very deep lack of comments, and most of methods understanding is left to the developer that reads it
19) As said before, there are very few comments and for this reason there are no unnecessary comments

3.7: Java Source File

20) The WorkEffortContentWrapper.java source file, contains only the WorkEffortContentWrapper public class
21) The WorkEffortContentWrapper public class is the first class in the WorkEffortContentWrapper.java source file
22) The WorkEffortContentWrapper public class, implements only the ContentWrapper interface, whose only method is the one implemented on line 80. Since the Javadoc for the interface is missing, we can say whether the class develops it correctly or not
23) The Javadoc is not complete, because it is provided only for the first four methods, respectively on lines 89,98,112 and 136

3.8: Package and Import Statements

24) The only package statement needed is the one provided on line 19, which is the first non-comment statement, followed by import statements

3.9: Class and Interface Declarations

25) The class declarations follow the default order
26) Methods are grouped (developed in the file) by functionality
27) The class is not free of duplicates: indeed method on line 222 (public static String getWorkEffortContentAsText (…)) contains only a call to a method with the same prototype, and a parameter is set to null. This should be avoided by calling directly the method on line 227 setting the second parameter at null. The same thing appends for the method on line 275 (public static void getWorkEffortContentAsText(…)) which refers to the method on line 279. We can also notice as the cyclomatic complexity of function on line 227 is 20, indeed the method is very long, and the same problem occurs also for function on line 279, which has a cycolomatic complexity of 17

3.10: Initialization and declarations

29) The parameter “workEffortId” is not used in the method. (Line 276)

31) Variable “dispatcher” is not initialized if the second constructor is called (Line 62). Moreover, variables “content” and “dataResource” could not be initialized when they are used if a GeneralException has been invocated in the try-catch construct (Lines 149-157).

33) Variables are not declared at the beginning of a block (Lines 234-235-236-253-255-298-314-315-366).

3.11: Method calls

36) The return values of the "put" methods are not caught (Lines 343-344).

3.12: Arrays

No problems found.

3.13: Object comparison

40) variables are compared with "!=" instead of "!equals"(Lines 101-115-124-139-148-156-164-184-188-196-237-248-261-281-285-299-304-316-320-339-354-358). Moreover, variables are compared with "==" instead of "equals" (Lines 230-281-285-293-317-354-358-362).

3.14: Output format

42) Error message is not comprehensive of a solution (Lines 120-153-161-177-193-266-270-294).

3.15: Computation, comparisons and assignments

49)This if construct will never be used, since at line 316 we can enter the if body only if work effort is not null. There could be a problem with the correctness of the operators (Line 317).

50) The two "catch" have the same body, they should be merged together.

4: Other problems

- The strings "content", "defaultMimeType", "text/html; charset=utf-8", "contentID", "WorkEffortContentTypeId", "workeffortId", "WorkEffortContent" are passed to many methods in the class. It is advisable to create constant String rather than passing always the strings, so to reduce possible errors in typing.
- Method "get" at line 90 overrides a method, but the notation "@Override" is not present. It should be added.
- Method "getList" at line 173 can return null, which could cause some NullPointerExceptions if not used correctly. It's better to return an empty collection.
- Some methods (lines 223, 228, 276, 280) have too many parameters passed, it could be better to split the method in some way.
- Some array or map objects (lines 306, 338,342) are declared by repeating in the "<>" the type of objects; this can be avoided.