Politecnico di Milano

Software Engineering 2 Project

a.y. 2016-2017

Code Inspection Document (CI)

Apache OFBiz project

Alessandro Perini, Federico Saini, Ali Merd Türkçapar

Version: 1.0 Release date: 05/02/2017

Contents

Chapter 1
Introduction

1.1 Purpose

This is the Code Inspection document, whose main goal is to examine a specific class of the Apache OFBiz project in order to discover bugs, verify that coding conventions are respected and look for potential issues. This is necessary to improve the quality of the code and to ensure the absence of errors in it.
The inspection has been performed following the checklist in the Code Inspection Assignment Document [4].

1.2 List of Reference Documents

Documents listed below are available here:
https://github.com/AlessandroPerini/PowerEnJoy/tree/master/referenceDocs

[1] Project Website: https://ofbiz.apache.org

[2] Apache OFBiz source code: http://mirror.nohup.it/apache/ofbiz/apache-ofbiz-16.11.01.zip

[3] Apache OFBiz Javadoc: https://ci.apache.org/projects/ofbiz/site/javadocs/

[4] Code Inspection Assignment Document: *"Code Inspection Assignment"*

Chapter 2
Assigned Classes

2.1 Description

Brief description of the assigned class to be inspected.

Name:	ServiceUtil.java
Location:	apache-ofbiz-16.11.01/framework/service/src/main/java/org/apache/ofbiz/service
Package:	org.apache.ofbiz.service
Modifier:	public final class

2.1 Functional Role of the Assigned Class

The class to inspect is a utility class, so his role is to provide a set of services that the program uses.

The class is mainly associated with the message system within each error and success dialog. It inserts messages and returns them as errors to the users. It also has some functions that optimize the software and improve the code efficiency in general.

Methods are grouped by functionality:

- Methods concerning errors and problems.
- Methods concerning messages.
- Methods concerning jobs.
- Generic methods.

Chapter 3
List of Issues and Problems

Task 2.1 - Naming Conventions

5. Every method name has the first letter of each addition word capitalized, but only one method doesn't start with a verb.

- **Line 612**: method name **genericDateCondition** doesn't start with a verb.

6. Attributes names:

- **Lines 483/484/485**: variables are named in different ways. One intending runtime as a single word and the other using run time as separated words. **runtimeData** and **runTimeDataIt**. Names must be standardized.
- **Line 486**: variable name **jobsandBoxCount** must be renamed with **jobsAndBoxCount**.

Task 2.2 - Indentation

8. Three or four spaces are used for indentation and done so consistently.

- **Line 375**: one extra space, the line is not aligned.

Task 2.3 - Braces

10. Kernighan and Ritchie style is used for all the braces. Only the constructor has curly braces on the same line but because it is empty.

- **Line 436**: the if closing brace is shifted right. One space to be canceled.

11. Statements not surrounded by braces:

- **Lines 618, 619 and 653**.

Task 2.4 - File Organization

13. There are multiple cases where the line length is more than 80 characters, most of these are required do not pass 120 characters.

- **Lines 70, 94, 117, 142**.

14. Also there are multiple times where the line length is more than 120 characters which presents worse readability.

- **Line 116**: 200 characters.
- **Line 120**: 221 characters.

Task 2.5 - Wrapping Lines

15. Not all line breaks occur after a comma or an operator.

- **Line 425-431**: line breaks occur after method's name and methods parameter.

17. A new statement is aligned with the beginning of the expression at the same level as the previous line.

- **Line 495**: the new parameter is not aligned with the others, after the line break.

Task 2.9 - Class and Interface Declarations

26. Some methods should change places:

- **Line 659**: **resetJob** should be moved in the section of job related methods.

27. Class is not overly long (739 lines), and methods are cohesively. The code is free of duplicates but there are some long methods:

- **Line 374**: method **purgeOldJobs** is too long (161 lines) but due to the great number of exceptions that the method can raise.

Task 2.10 - Class and Interface Declarations

33. Not all variables are declared at the beginning of the block.

- **Line 306**: declaration of the variables is not at the beginning of the method.

Task 2.15 - Computation, Comparisons and Assignments

44. There are a couple of “brutish” programming practices (Task 5):

- **Line 82**: If statement is not necessary.

```
public static boolean isSuccess(Map<String, ? extends Object> results) {
    if (ServiceUtil.isError(results) || ServiceUtil.isFailure(results)) {
        return false;
    }
    return true;
}
```

can be written as

```
public static boolean isSuccess(Map<String, ? extends Object> results) {
    return !(ServiceUtil.isError(results) || ServiceUtil.isFailure(results));
}
```

Chapter 4
Other Problems and Comments

4.1 Other Problems

- Empty constructor. The class contains an empty constructor that can be removed since the compiler automatically generates it.
- Description of the class is not exhausted. It should definitely be more than “generic service class” which indeed is nothing more than placeholder. It must be replaced with something more meaningful rather than reciting the class name.

4.2 Comments

Task numbers refer to the checklist in the Code Inspection Assignment Document [3].

Task 2.1 - Naming Conventions

1. All names are meaningful.
2. All one-character variables are used as temporary variables. Single character found are: “e” used for exceptions and “i” index used only inside for statements.
3. The only class to inspect has a correct name: ServiceUtil.java.
4. No interfaces to inspect.
5. *Problem Found - See Chapter 3.*
6. *Problem Found - See Chapter 3*.
7. No constants declared.

Task 2.2 - Indentation

8. *Problem Found - See Chapter 3.*
9. No tabs are used.

Task 2.3 - Braces

10. *Problem Found - See Chapter 3.*
11. *Problem Found - See Chapter 3.*

Task 2.4 - File Organization

12. Blank lines and optional comments are used to separate sections.
13. *Problem Found - See Chapter 3.*
14. *Problem Found - See Chapter 3.*

Task 2.5 - Wrapping Lines

15. *Problem Found - See Chapter 3.*
16. All line breaks occur after a semicolon or curly brackets.
17. All the new statements are aligned with the previous one.

Task 2.6 - Comments

18. Not all methods or code blocks are commented but they are comprehensible.
19. There's no commented out code.

Task 2.7 - Java Source Files

20. One class per source file.
21. The public class is the first in the file.
22. There is no external programming interface used in the class.
23. Javadoc is complete, covering all the methods in the class.

Task 2.8 - Package and Import Statements

24. Package statements are the first lines followed by the import statements.

Task 2.9 - Class and Interface Declarations

25. Correct declarations.
26. *Problem Found - See Chapter 3.*
27. *Problem Found - See Chapter 3.*

Task 2.10 - Initialization and Declarations

28. All variables are of the correct type and have the right visibility.
29. All variables are declared in the proper scope.
30. Constructors are used properly.
31. All objects are initialized before the use.
32. Some variables are initialized when they are declared.
33. *Problem Found - See Chapter 3.*

Task 2.11 - Method Calls

34. Parameters are present in the correct order for every called method.
35. All methods are correctly called.
36. Returned values are used properly.

Task 2.12 - Arrays

37. There are no off-by-one errors in array index.
38. Arrays limits are well defined.
39. When a new object is created the constructor is called.

Task 2.13 - Object Comparison

40. Object comparisons are done with equals expression when dealing with non-integer objects.

Task 2.14 - Output Format

41. No grammar errors in displayed outputs.
42. Clear error messages.
43. Output is formatted correctly.

Task 2.15 - Computation, Comparisons and Assignments

44. *Problem Found - See Chapter 3.*
45. There is no error in precedence of operators and the order is correct.
46. The parentheses are used liberally except for the one line if statements.
47. No divisions in the code.
48. There are no arithmetic operations in the class.
49. All comparisons and Boolean operators are correct.
50. Throw and catch expressions complete each other and error conditions seem valid.
51. There are no implicit type conversion.

Task 2.16 - Exceptions

52. All relevant exceptions are caught.
53. Each exception is correctly managed.

Task 2.17 - Flow of Control

54. No switches in the code.
55. No switches in the code.
56. All loops are correctly formed.

Task 2.18 - Files

57. There is no file opened in this class.
58. Therefore file closures are needed.
59. There are no cases where EOF can occur.
60. No exceptions can occur associated to files.