Politecnico di Milano

PowerEnjoy Service - Code Inspection Assignment

February 3, 2017

Version 1.1

Authors:

- Domenico FAVARO (Mat. 837995)
- Matheus FIM (Mat. 876069)
- Caio ZULIANI (Mat. 877266)

Prof. Elisabetta DI NITTO

Contents

1 Class assigned - SeoConfigUtil

The Java Class assigned to this document is `SeoConfigUtil`, a class that is part of the `org.apache.ofbiz.product.category` package of the Apache OFBIZ (Open For Business) Framework.

1.1 Functional Role of SeoConfigUtil

The function of this Class as its name states is SEO Configuration File Utility, SEO stands for Search Engine Optimization. Its main function is to parse a XML Cofiguration File `SeoConfig.xml` and store all the elements, URL patters, Categories and filters stated on that configuration document, using a Perl Compiler for Regular Expressions, and then offers methods that answer wether the Category URLs are enabled, wether they can be accesed by Users or Anonymous IDs on the Server and other important values for the SEO of the Web Page.

For this the Class consists on an `init()` method that initializes the class by parsing the XML file, and other access methods that return, if the initialization was successful, the elements of the SEO Configuration.

2 Issues - Code Inspection Checklist

2.1 Naming Conventions

SeoConfigUtil respects naming conventions throughout the file. To illustrate some important points:

1. All method names are meaningful and describe exactly what the method does. Ex:

   - (l.409) `isCategoryNameEnabled()` checks whether category name is enabled.
   - (l.436) `isJSessionIdUserEnabled()` checks whether jsessionid is enabled for user.

   With the exception of maybe the `init()` method that consists on the initialization of the class, but does a lot of parsing of the Configuration File, in fact a 230 lines method that does not call any other methods inside the Class could be considered for break up on sub methods. However this is a method that should be called once for the entire Class and having an `init()` method that just calls an equally long `parseConfigFile()` may be over complicating.

2. One character variables are used only in temporary variables, inside `for` loops (`i,j`) or to catch exceptions (`e`). Other temporary variables that do not include loops or exceptions have meaningful names inside the context they are used, like `pattern` that is initialized as the compiled URL pattern of the Perl Compiler.
3. The only Class name declared in this file is `SeoConfigUtil` and it respects the capitalized letter camelCase for its noun.
4. No Naming Conventions for Interfaces for No Interfaces are declared in this file.
5. All methods are verbs, almost all are getters like `getSpecialProductId()`. Those that return a `boolean` value are posed as a question like `isInitialed()` with the exception of `checkUseUrlRegexp()` and `checkCategoryUrl()` that to respect the naming convention should be called `canUseUrlRegexp()` and `isCategoryUrlEnabled()`
6. All class variables are declared as camelCase. That is mixed case, starting with a lowercase first letter and all theremaining words in the variable name have their first letter capitalized. Ex: `userExceptionPatterns`.
7. All constants, mostly Strings are declared with the all Caps snake case or `SCREAMING_SNAKE_CASE` Ex: `ELEMENT_URL_CONFIG`.

2.2 Identation

For identation four spaces are used consistently and no tabs are used to indent.

2.3 Braces

The Braces ident style used in this Class is the K&R ("Kernighan and Ritchie's") style, first brace is on the same line of the instruction that opens the new block, with the Java variant, that is also for class or method declarations. Ex:

```
public static boolean isInitialed() {
    return isInitialed;
}
```

2.4 File Organization

The Class follows the separation of each method by comments and even inside the `init()` method comments are used to separate the different parts.
It does not respect however the 80 or either the 120 characters line length limit, due to the highly explicative variable names and numerous Debug logs in the `init()` method, making it not an easy method to read.

2.5 Wrapping Lines

As no limit is respected for the line lenght, no wrapping of lines is used, no line breaks. All lines finish at the end of their current expression with ; or { }

2.6 Comments

The Comments in the file are adequately explaning the function of the classes. They are also used to separate and explain the diferentes parts of the class `init()` .

2.7 Java Source Files

The file has one first top-level public class named as the file's name. It also has consistency when becomes of the javadoc in such parameters as that all the classes of the file have the parameter that the javadoc expects also in the reference of the external program interfaces.

2.8 Package and Import Statements

The class assigned requires one package and as was expected it is placed as the first non-coment statement of the file.

2.9 Class and Interface Declarations

2.10 Initialization and Declarations

All the variables were declared in accordance with the classes types and were declared in the correct scope. Moreover when a object is needed there are constructor positioned in the correct form also the variables used for the these objects were initialized in the beggining of the code as is recomended. All variables were declared and initialized in the beggining of the blocks.

2.11 Method Calls

The use of Methods in the file are in accordance with the expected. The ones that have parameters are presented in a correct order as well as the ones that are void don't have any parameters, in other words void methods are not passing parameters. In the calling of the methods they are correct, they refer to existing methods with correspondent assignature. All the returns of the methods are returning values as expected as the following example points were the return 'specialProductIds.containsKey(productId)' were declared as boolean.

```
public static boolean isSpecialProductId(String productId) {
        return specialProductIds.containsKey(productId);
    }
```

2.12 Arrays

The file assigned does not deal with arrays.

2.13 Object Comparison

2.14 Output Format

2.15 Computation, Comparisons and Assignments

2.16 Exceptions

2.17 Flow of Control

2.18 Files

3 Others Problems

4 References

4.1 Used Tools

- **GitHub:** for Version Control
- **Notepad++:** for Java Code Editing
- **TeXworks:** for LaTex editing of this Document

4.2 Effort Spent

Date	Domenico	Caio	Matheus
30/01/17	1h	1h	1h
31/01/17	2h	-	-
01/02/17	-	-	-
02/02/17	2h	-	-
03/02/17	-	-	-
04/02/17	-	-	-
05/02/17	-	-	-

5 Changelog

As the project and design decisions may change during the development this document is also prone to change. We'll document every version in this part.

- **Version 1.1:** 05/02/2017