APACHE OFBIZ

CID

Code Inspection Document

Version: 1.0.0

5th February 2017

Edited by:

Marzia Degiorgi - 878360

Giulia Leonardi - 877491

Valentina Ionata - 808678

Index

1. Classes

The classes assigned to our group are the following:

- ***ProductDocument:***../apache-ofbiz-16.11.01/specialpurpose/lucene/src/main/java/org/apache/ofbiz/content/search/ProductDocument.java*
- ***ModelWidget:***../apache-ofbiz-16.11.01/framework/widget/src/main/java/org/apache/ofbiz/widget/model/ModelWidget.java*

2. Functional Role

ModelWidget is an abstract Class that must be extended by other classes in the model. It defines the general features of the widget characterized by the name, the url (system id) and the starting position (column and line where the widget is defined). It contains classic "get" methods that return the private variables of the classes, and it also includes an abstract method to be implemented by the subclasses, that is "public abstract void accept(ModelWidgetVisitor visitor)" . The deduction comes from the javadoc:

```
/**
 * Widget Library - Widget model class. ModelWidget is a base class that is
 * extended by other widget model classes.
 */
```

ProductDocument is the class that allows to build up the document containing the features of each product (e.g. "productId","productName","description", ...). For each field of the document, some values can be added through a function of this class, pinpointing the weight and the attributes chosen. Since the class is devoid of its javadoc, we have deduced its role within the project from the lines of code, attributes and methods.

3. List of Issues

For each class we are going to report the issues found regarding every main point of the checklist.

3.1 ModelWidget

3.1.1 Naming convention

All the conventions for classes, names of interfaces, variables and methods are respected. The only issue found is the declaration of the variables reported below:

<line 36> *public static final String module = ModelWidget.class.getName();*
<line 41> *public static final String enableBoundaryCommentsParam = "widgetVerbose";*
<line 43> *private final String name;*
<line 44> *private final String systemId;*
<line 45> *private final int startColumn;*
<line 46> *private final int startLine;*

In fact, these variables are all constants, but they are not declared using uppercase letters with words separated by an underscore.

3.1.2 Indention

No issues are found in this section, all the spaces follow the rules and in particular the indentation uses, consistently, four spaces.

3.1.3 Braces

No issues are found in this section. The “Kernighan and Ritchie” style is used for the whole class.

3.1.4 File Organization

Both blank lines and optional comments are used in the entire class, except between the beginning comments and the package statement, where there are not blank lines.

<line 1..19>

```
/*******************************************************************************
* Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
```

* * under the License.*
***/

<line 20> *package org.apache.ofbiz.widget.model;*

The lines length constraints are followed.

3.1.5 Wrapping Lines

No issue found in this section.

3.1.6 Comments

The comments are correctly used, the only issue found is reported below:

<line 126..135>

```
/**
 * Returns <code>true</code> if widget boundary comments are enabled. Widget boundary
 * comments are enabled by setting <code>widget.verbose=true</code> in the
*<code>widget.properties</code> file.
*The <code>true</code> setting can be overridden in <code>web.xml</code> or in the
*screen rendering context. If <code>widget.verbose</code> is set to <code>false</code> in
*the <code>widget.properties</code> file, then that setting will override all other settings
*and disable all widget boundary comments.
* @param context Optional context Map
 **/
```

In fact, the javadoc format for the returned parameters does not follow the convention specifying the "@return" statement.

3.1.7 Java Source File

The javadoc does not cover all the methods in the class, in particular the following abstract method:

<line 70> *public abstract void accept(ModelWidgetVisitor visitor) throws Exception;*

As said in the previous section the javadoc of the method reported below is inconsistent because it does not follow the original structure:

<line 136> *public static boolean widgetBoundaryCommentsEnabled(Map<String, ? extends Object> context)*

3.1.8 Package and Import Statements

No problems have been observed in this section.

3.1.9 Class and Interface Declaration

This class does not follow the grouping defined by functionality: in fact, the following statement, which would be listed with the "get" statements group, next the constructors, is not places there.

<line 122> *public String getBoundaryCommentName(){..}*

3.1.10 Initialization and Declaration

No issues found.

3.1.11 Method Calls

An error regarding the parameters of a method occurs: the method "getPropertyValue" belonging to the class "UtilProperties" needs as first parameter the file "widget.properties", instead of "widget".

<line 137> *boolean result = "true".equals(UtilProperties.getPropertyValue("widget", "widget.verbose"));*

3.1.12 Arrays

No issues found.

3.1.13 Object Comparison

No issues found.

3.1.14 Output Format

The following error message does not provide the right guidance to correct the problem since it just pinpoints the class where the problem occurs; furthermore, it is not grammatically correct too ("thrown in" is, in fact, a phrasal verb with the meaning of "including something extra") :

<line 111> *Debug.logWarning(e, "Exception thrown in XmlWidgetVisitor: ", module);*

3.1.15 Computation,Comparison and Assignment

No issues found.

3.1.16 Exception

No issues found.

3.1.17 Flow of Controls

No cycles and switch statement present in the class.

3.1.18 Files

No issues found.

3.2 ProductDocument

3.2.1 Naming convention

The standard constant declaration is not applied: in fact lower case letters instead of uppercase ones separated by an underscore are used in this class:

<line 52> *private static final String module = ProductDocument.class.getName();*
<line 54> *private final Term documentIdentifier;*

3.2.2 Indention

Four spaces are used for indentation,consequently no issue was found.

3.2.3 Braces

The "Kernighan and Ritchie" style is used; the issues found in this section are listed below:

<line 285> *if (fieldName == null) return;*
<line 319> *if (nextValue == null) return currentValue;*
<line 321> *if (currentValue == null) return nextValue;*
<line 323> *if (currentValue.after(nextValue)) return nextValue;*

In these pieces of code the braces for the if statements are not used:

<line 342> *EntityCondition.makeCondition(thruDateName, EntityOperator.GREATER_THAN, UtilDateTime.nowTimestamp())*
<line 343> *));*

- Finally, here a piece of code where there is a brace ')' extra is reported.

3.2.4 File Organization

- In the structure of this class, blank lines and optional comments not separates all the sections. We reported below all the interested parts.

- no separation between beginning comment and package statement:
-

<line 1..18>
*/***/*
***<line 19>**package org.apache.ofbiz.content.search;*

- no separation between class declaration and variable declaration:

***<line 51>** public class ProductDocument implements LuceneDocument {*
***<line 52>** private static final String module = ProductDocument.class.getName();*

- no separation between static and nonstatic variables:

***<line 53>** private static final String NULL_STRING = "NULL";*
***<line 54>** private final Term documentIdentifier;*

- blank separation between all the methods are used but no comments

Beside there are different lines that exceed the limit of 80 characters, among which:

***<line 72>** GenericValue product = EntityQuery.use(delegator).from("Product").where("productId", productId).queryOne();*

***<line 92>** nextReIndex = this.checkSetNextReIndex(product.getTimestamp("introductionDate"), nextReIndex);*
***<line 94>** nextReIndex = this.checkSetNextReIndex(product.getTimestamp("salesDiscontinuationDate"), nextReIndex);*

The following lines exceed also the limit of 120 characters:

***<line 77>**if ("Y".equals(product.getString("isVariant")) && "true".equals(EntityUtilProperties.getPropertyValue("prodsearch", "index.ignore.variants", delegator))) {*

***<line 85>**this.addTextFieldByWeight(doc, "productName", product.getString("productName"), "index.weight.Product.productName", 0, false, "fullText", delegator);*
***<line 86>**this.addTextFieldByWeight(doc, "internalName", product.getString("internalName"), "index.weight.Product.internalName", 0, false, "fullText", delegator);*

<line 87> *this.addTextFieldByWeight(doc, "brandName", product.getString("brandName"), "index.weight.Product.brandName", 0, false, "fullText", delegator);*
<line 88>*this.addTextFieldByWeight(doc, "description", product.getString("description"), "index.weight.Product.description", 0, false, "fullText", delegator);*
<line 89>*this.addTextFieldByWeight(doc, "longDescription", product.getString("longDescription"), "index.weight.Product.longDescription", 0, false, "fullText", delegator);*

<line 91> *doc.add(new LongField("introductionDate", quantizeTimestampToDays(product.getTimestamp("introductionDate")), Field.Store.NO));*
<line 93>*doc.add(new LongField("salesDiscontinuationDate", quantizeTimestampToDays(product.getTimestamp("salesDiscontinuationDate")), Field.Store.NO));*
<line 95> *doc.add(new StringField("isVariant", product.get("isVariant") != null && product.getBoolean("isVariant") ? "true" : "false", Field.Store.NO));*

We do not report all the lines of code with these kind of inequalities due to the high number of them in the entire class.

3.2.5 Wrapping Lines

No issues found in this section.

3.2.6 Comments

In this class very few comments are used and no javadoc is present to explain methods, blocks of code and constructor.
There are also a piece of code that is commented out and any details are provided about the reason why. We report below the block of code we are talking about:

<line 90>
//doc.add(new StringField("introductionDate", checkValue(product.getString("introductionDate")), Store.NO));

3.2.7 Java Source File

The major issue found is the lack of the javadocs about every objects in the class. This prevents us from checking the consistence of the external interfaces.

3.2.8 Package and Import Statements

No issues found, the order of the packages and import statements is correct.

3.2.9 Class and Interface Declaration

The usual order is followed, except for the class documentation (javadoc), that is not present at all. Furthermore, the following static variable:

<line 328..332>
private static final EntityCondition THRU_DATE_ONLY_CONDITION = EntityCondition.makeCondition(
EntityCondition.makeCondition("thruDate", EntityOperator.EQUALS, null),
EntityOperator.OR,
EntityCondition.makeCondition("thruDate", EntityOperator.GREATER_THAN, UtilDateTime.nowTimestamp())
);
should be declared with the other private static type at the beginning of the class.

Concerning the methods, they are rightly grouped by functionality, except for the following ones:
<line 61> *public String toString()*
<line 65> *public Term getDocumentIdentifier()*

Furthermore, the code has very long methods, such as:

<line 69..281> *"public Document prepareDocument(Delegator delegator)"*.

A case of duplication of code appears, for the first time, at

<lines 124-126> *for (GenericValue productFeatureGroupAppl : productFeatureGroupAppls){*

fromDate = productFeatureGroupAppl.getTimestamp("fromDate");

thruDate = productFeatureGroupAppl.getTimestamp("thruDate");

}

This actions about timestamps is repeated 9 times within the same class: gathering these lines in a method and revoking it every following times it is required would tangibly improve the style of the written code and decrease its length.

An other duplication of code is present in

<lines 85-89> *this.addTextFieldByWeight(doc, "productName", product.getString("productName"), "index.weight.Product.productName", 0, false, "fullText", delegator);*

this.addTextFieldByWeight(doc, "internalName", product.getString("internalName"), "index.weight.Product.internalName", 0, false, "fullText", delegator);

this.addTextFieldByWeight(doc, "brandName", product.getString("brandName"), "index.weight.Product.brandName", 0, false, "fullText", delegator);

this.addTextFieldByWeight(doc, "description", product.getString("description"), "index.weight.Product.description", 0, false, "fullText", delegator);

this.addTextFieldByWeight(doc, "longDescription", product.getString("longDescription"), "index.weight.Product.longDescription", 0, false, "fullText", delegator);

and again in <lines 115-120>, <lines 145-146> and more. With this solution it is easy to make grammatical or typographical mistakes and incorrectly typed letters or numbers could produce incorrect results. Furthermore,it is not easy to modify. Finally, it requires many unnecessary lines of code. The construct could be replaced with just a method, which scans a serie of lists containing all the parameter thanks to a simple “for” cycle. This would permit an easy modification of the parameters taken by the method, if required, and would decrease consistently the number of lines of codes employed.

3.2.10 Initialization and Declaration

No issues found: all the variables are correctly initialized, all the classes and variables have the right visibility, and all the other subjects concerning the “initialization and declaration” topic are correct too.

3.2.11 Method Calls

No issues found.

3.2.12 Arrays

No issues found: there are no index out of bounds, neither off-by-one errors. Constructors are correctly used.

3.2.13 Object Comparison

No issues found.

3.2.14 Output Format

The following error message does not provide any suggestion or guidance to understand and correct the problem:

<lines 277-279>
```
catch (GenericEntityException e) {
Debug.logError(e, module);
}
```

No other grammatical or formatting errors.

3.2.15 Computation,Comparison and Assignment

No issues have been found.

3.2.16 Exception

All the relevant exceptions are caught with try/catch block.

3.2.17 Flow of Controls

There are no switch statements and the cycles are correctly formed.

3.2.18 Files

No files are present in this class.

4. Other Problems

The only additional issue that could be pointed up is about the *ProductDocument* class. In fact, several methods belonging to thi class have a very high cyclomatic complexity, due to the frequent presence of *"for"* cycles and *"if"* statements, and also due to the length and to common repetitions that may create bugs in the code.

No others specific problems were found in the assigned classes.

5. Revision History

Version	Date	Summary
1.0.0	**05/02/2017**	Initial Release