POLITECNICO
MILANO 1863

Apache OFBiz:

Code Inspection of Selected Classes

Version 1.0

Authors:
Mosciatti Simone
Zanzottera Sara

Reference Professor:
Mottola Luca

January 23, 2017

Contents

1 Introduction

This document described the results of a peer review of a few selected classes taken from the Apache OFBiz Project (official website: https://ofbiz.apache.org), an open source product for the automation of enterprise processes that includes framework components and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) and other business-oriented functionalities.

The code inspection has been carried out following the guidelines described in the reference document provided and consulting the official documentation (https://ofbiz.apache.org/documentation.html)

1.1 Reference Documents

- *Code Inspection Assignement Task Description.pdf*
- *Code Inspection Assignement Table.xlsx*
- Official Documentation of the project at https://ofbiz.apache.org/documentation.html
- Source Code of the project at http://mirror.nohup.it/apache/ofbiz/apache-ofbiz-16.11.01.zip
- *Brutish Programming* at http://users.csc.calpoly.edu/jdalbey/SWE/CodeSmells/bonehead.html

2 Assigned Classes

The class assigned to our group is: `MiniLangUtil.java` (`apache-ofbiz-16.11.01/framework/minilang/src/main/java/org/apache/ofbiz/minilang/MiniLangUtil.java`)

2.1 Role of the assigned class

This class provides some utility functions for the MiniLang Engine.

Mini-Language is described on the official documentation as follows: *The Mini-Language concept in Open For Business is [...] to create simple languages that simplify complex or frequently performed tasks.*

The specific purpose of this static class is unclear: it simply provides a set of static, utility methods and cannot be instantiated. It includes some fundamental operations, like `.callMethod()`, that probably needs to not belong to instantiable objects, but are fundamental inside the package.

The documentation does not provide in-depht informations about this section of the OFBiz framework: it is impossible to understand the actual role of the class without a detailed inspection of all the classes of the `minilang` package.

However, the role of the package itself is clearer, as we can see in the following diagram: it is used as an interpreter for the instructions coming from the Service Engine for the Entity Engine.

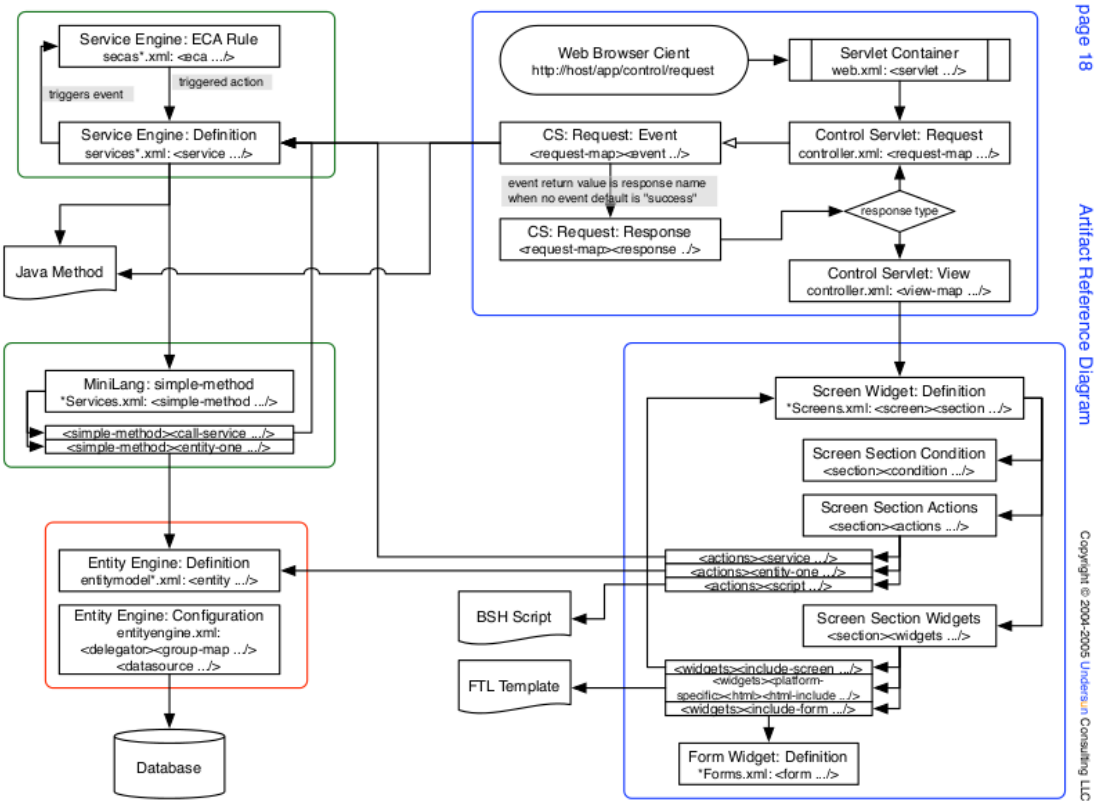

Figure 1: OFBiz Artifact Reference Diagram

2.2 Usage of the class

In order to have a general idea of how and how much the class is used inside the project, we performed a lookup into the repository. `grep` reports these usages of `MiniLangUtil` into the OFBiz source code:

```
1 grep -Ril "MiniLangUtil" [ofbiz-basepath]
2
3 [minilang-package]/method/eventops/RequestParametersToList.java
4 [minilang-package]/method/envops/Now.java
5 [minilang-package]/method/envops/SetOperation.java
6 [minilang-package]/method/envops/ToString.java
7 [minilang-package]/method/envops/SetCalendar.java
8 [minilang-package]/method/callops/CallClassMethod.java
9 [minilang-package]/method/callops/CallObjectMethod.java
```

```
[minilang-package]/method/callops/SetServiceFields.java
[minilang-package]/method/callops/CallScript.java
[minilang-package]/method/otherops/PropertyToField.java
[minilang-package]/method/otherops/Calculate.java
[minilang-package]/method/conditional/ValidateMethodCondition.java
[minilang-package]/method/conditional/CompareCondition.java
[minilang-package]/method/conditional/RegexpCondition.java
[minilang-package]/method/conditional/Compare.java
[minilang-package]/method/conditional/CompareFieldCondition.java
[minilang-package]/SimpleMethod.java
[minilang-package]/MiniLangValidate.java
```

Then we check if all methods included in `MiniLangUtil.java` are actually used in the repo.

```
grep -Ril "MiniLangUtil.containsScript" [ofbiz-basepath]
[minilang-package]/method/envops/SetOperation.java
[minilang-package]/method/envops/SetCalendar.java
[minilang-package]/method/callops/CallScript.java
[minilang-package]/MiniLangValidate.java

grep -Ril "MiniLangUtil.autoCorrectOn" [ofbiz-basepath]
[minilang-package]/method/eventops/RequestParametersToList.java
[minilang-package]/method/envops/Now.java
[minilang-package]/method/envops/SetOperation.java
[minilang-package]/method/envops/SetCalendar.java
[minilang-package]/method/callops/SetServiceFields.java
[minilang-package]/method/callops/CallScript.java
[minilang-package]/method/otherops/PropertyToField.java
[minilang-package]/method/conditional/CompareFieldCondition.java
[minilang-package]/SimpleMethod.java

 grep -Ril "MiniLangUtil.callMethod" [ofbiz-basepath]
[minilang-package]/method/callops/CallClassMethod.java
[minilang-package]/method/callops/CallObjectMethod.java

grep -Ril "MiniLangUtil.convertType" [ofbiz-basepath]
[minilang-package]/method/envops/SetOperation.java
[minilang-package]/method/envops/ToString.java
[minilang-package]/method/envops/SetCalendar.java
[minilang-package]/method/otherops/Calculate.java
[minilang-package]/method/conditional/ValidateMethodCondition.java
[minilang-package]/method/conditional/RegexpCondition.java
[minilang-package]/method/conditional/Compare.java

grep -Ril "MiniLangUtil.flagDocumentAsCorrected" [ofbiz-basepath]
[minilang-package]/method/eventops/RequestParametersToList.java
[minilang-package]/method/envops/Now.java
[minilang-package]/method/envops/SetOperation.java
[minilang-package]/method/envops/SetCalendar.java
[minilang-package]/method/callops/SetServiceFields.java
[minilang-package]/method/callops/CallScript.java
[minilang-package]/method/otherops/PropertyToField.java
[minilang-package]/method/conditional/CompareFieldCondition.java
```

```
[minilang-package]/SimpleMethod.java

grep -Ril "MiniLangUtil.getObjectClassForConversion" [ofbiz-basepath]
[minilang-package]/method/envops/SetOperation.java
[minilang-package]/method/conditional/CompareCondition.java
[minilang-package]/method/conditional/CompareFieldCondition.java

grep -Ril "MiniLangUtil.isConstantAttribute" [ofbiz-basepath]
[minilang-package]/MiniLangValidate.java

grep -Ril "MiniLangUtil.isConstantPlusExpressionAttribute" [ofbiz-basepath]
[minilang-package]/MiniLangValidate.java

grep -Ril "MiniLangUtil.isDocumentAutoCorrected" [ofbiz-basepath]
[minilang-package]/SimpleMethod.java

grep -Ril "MiniLangUtil.writeMiniLangDocument" [ofbiz-basepath]
[minilang-package]/SimpleMethod.java
```

The results support our previous hypotesis: `MiniLangUtil` is an internal class of the `minilang` package. It is never used outside the package and provides support functions for the processes of the `minilang` package.

A completely different analysis should be done for the second class included in this source file, `PlainString`. Such analysis is performed in a dedicated subsection of the next section.

3 Issues Found

Issues are listed first concerning the class as a whole, and then method by method.

3.1 MiniLangUtil class

- Line 60: the public final constant `module` should be named in capital letters.
- Line 65: Sonar reports `new HashSet<String>()` as a minor issue, because a diamond operator could be used here (`new HashSet<>()`)
- File organization shows some issues. A lot of lines exceed the 80-char limits, and a few of them (line 111, 152 and 172) exceed the 120-char limit too. In detail, line 111 breaks after char 152 (the total line size is 260 chars), while line 152 reaches the 138 chars and line 172 reaches 124 chars.
- Javadoc is inconsistently detailed, and lacks fundamental informations for some methods (see next subsections).
- Concerning Java Source Files organization, a peculiar issue has been found. `MiniLangUtil` includes a public inner class, called `PlainString`, whose usage and role into the project looks completely different from `MiniLangUtil`'s ones. See the dedicated section for more details about this issue.

- Being an utility class used only inside the package, `MiniLangUtil` show no need to be a public class. The class definition should be changed from `public final class MiniLangUtil` into `final class MiniLangUtil`.

3.2 containsScript (Line 79)

- Line 80: `str.lenght()` is called, but `str` is not guaranteed to be not null. In addition, the method's Javadoc does not says that the behavior of the method in case of null input is a `NullPointerException` (lacks a `@throws` statement), while it should be the case.

3.3 autoCorrectOn (Line 96)

No issues found.

3.4 callMethod (Line 111)

- This method's Javadoc lacks a lot of informations, starting from parameter's descriptions.
- One of the parameters (`retFieldFma`) has a name that becomes understandable only reading the whole method. It should at least be described in the Javadoc.
- Line 133: `retFieldFma.isEmpty()` is called but `retFieldFma` is not guaranteed to be not null. Javadoc for this method does not mentions that the result for a null `retFieldFma` in input is a `NullPointerException` (lacks a `@throws` statement)
- Line 136: each `Exception` is caught and changed into a `MiniLangRuntimeException`. This can be reasonable considering the context (the code simulates a method call), but looks like a code smell.
- Line 112-123: Sonar reports this block as a duplicate of lines 92-103 of the method `CreateObject.exec()`. We believe this block of code should be moved into a separate method and called by both methods at need.

3.5 convertType (Line 152)

- This method's Javadoc lacks a lot of informations, starting from parameter's description.
- Line 151: `@SuppressWarnings` refers to line 172, where an unchecked cast to `Converter<Object, Object>` is performed. Considering that `Converters` is a class that belongs to the project (see, at line 38, the `import` statement) this is probably due to a misunderstanding between developers that are in charge of these classes. `Converters` should be modified to provide a `Converter<Object, Object>` in output.

- == is often used in place of `.equals()` to compare objects: see lines 153, 159, 165, 169. These == comparisons should be safe, as they are used only for comparing classes, but as a general rule, it is desirable to use `.equals()` in any case.
- Line 173: the declaration of `localizedConverted` could be moved into the `if` block for clarity. The variable is never used outside the block.
- Lines 177, 180, 183: Sonar reports the reassigment of new values to some input variables.
- This method contains a snippet of code that could be moved elsewhere: lines 182 to 184 looks like input validation for the subsequent call to `localizedConverter.convert(obj, locale, timeZone, format)`, that would be better located into that method itself.

3.6 flagDocumentAsCorrected (Line 194)

- This method's Javadoc lacks a lot of informations, starting from parameter's description.
- Line 195: `element` is not guaranteed to be not null, but a call to `element.getOwnerDocument()` is issued. Javadoc for this method does not mention that the result for a null `element` in input is a `NullPointerException` (lacks a `@throws` statement)

3.7 getObjectClassForConversion (Line 211)

- This method's Javadoc, even being a little more detailed that the previous ones, lacks parameter's description and could be a little more detailed about the behavior or the method concerning `Map, List` and `Set` classes.

3.8 isConstantAttribute (Line 235)

- Line 236: A call on `attributeValue.lenght()` is performed, but `attributeValue` is not guaranteed to be not null. This leads to a peculiar behavior: if `attributeValue == ""`, it returns `true`, while if `attributeValue == NULL`, it throws a `NullPointerException` (not stated in the Javadoc).

3.9 isConstantPlusExpressionAttribute (Line 251)

- Line 252: A call on `attributeValue.lenght()` is performed, but `attributeValue` is not guaranteed to be not null. This leads to a peculiar behavior: if `attributeValue == ""`, it returns `true`, while if `attributeValue == NULL`, it throws a `NullPointerException` (not described in the Javadoc).
- Line 254-255: Comments are used in a cryptic way. The statement does not explain why concatenated expressions should be avoided. It does not even contains `TODO` or `FIXME` tags.

3.10 isDocumentAutoCorrected (Line 273)

- Line 174: a call to `document.getUserData()` is issued, but `document` is not guaranteed to be not null. The consequent `NullPointerException` is not even stated in the Javadoc (lacks a `@throws` statement).

3.11 writeMiniLangDocument (Line 284)

- This method's Javadoc lacks a lot of informations, starting from parameter's descriptions.
- Line 289: the path `"component://minilang/config/MiniLang.xslt"` is hardcoded. It should be at least moved into a private constant or made configurable by putting it in a config file.
- Exceptions management in this method is poor. Twice (line 293 and 310) there is a `catch(Exception e)` statement that replaces the original exception with a generic error message.
- Line 307: `xmlURL.getFile()` is called, but `xmlURL` is not guaranteed to be not null. In addition, this statement is put into a `try { } catch(Exception e)` block that can completely mask this trivial issue.
- Line 311: `xmlURL` is concatenated to the error message string without even checking it for being not null. This can raise a new Exception inside the `catch` block.
- Sonar reports that the whole `try` block should be replaced with a more compliant "try-with-resources" block, as files are involved. This is an interesting suggestion that should definitely be implemented, but it is not an issue per se.

3.12 PlainString

`MiniLangUtil` include a public, inner class called `PlainString`. This class, which can be found at line 323, is completely empty (the whole source code of `PlainString` is `public static class PlainString {}`) and shows no JavaDoc.

We undestand that creating a specific source file for an empty class like `PlainString` can be overkill, but anyway this solution is awkward. A public class in general has no reason for being internal, especially into a class like `MiniLangUtil` that could hold a default modifier instead of the public one.

This solution simply makes difficult for developerd to locate `PlainString` into the project.

In addition, being the class empty, we wondered if `PlainString` is even used into the project, or it is a leftover. So we performed a lookup into the code base to locate eventual usages: the results, shown below, are pretty surprising.

```
grep -Ril "PlainString" [ofbiz-basepath]

[ofbiz-basepath]/applications/order/groovyScripts/entry/cart/LookupBulkAddSupplierProducts.groovy
[ofbiz-basepath]/applications/product/src/main/java/org/apache/ofbiz/shipment/thirdparty/ups/UpsServices.java
[ofbiz-basepath]/applications/product/src/main/java/org/apache/ofbiz/shipment/thirdparty/dhl/DhlServices.java
[ofbiz-basepath]/applications/product/src/main/java/org/apache/ofbiz/shipment/thirdparty/usps/UspsServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/cybersource/IcsPaymentServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/verisign/PayflowPro.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/authorizedotnet/AIMPaymentServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/orbital/OrbitalPaymentServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/paypal/PayPalEvents.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/paypal/PayPalServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/clearcommerce/CCPaymentServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/ideal/IdealEvents.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/gosoftware/RitaServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/gosoftware/PcChargeServices.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/thirdparty/worldpay/WorldPayEvents.java
[ofbiz-basepath]/applications/accounting/src/main/java/org/apache/ofbiz/accounting/test/FinAccountTests.java
[ofbiz-basepath]/framework/entity/dtd/entity-eca.xsd
[ofbiz-basepath]/framework/service/dtd/service-eca.xsd
[ofbiz-basepath]/framework/minilang/src/main/java/org/apache/ofbiz/minilang/MiniLangUtil.java
[ofbiz-basepath]/framework/minilang/dtd/simple-methods.xsd
[ofbiz-basepath]/framework/common/src/main/java/org/apache/ofbiz/common/uom/UomWorker.java
[ofbiz-basepath]/framework/widget/dtd/widget-common.xsd
[ofbiz-basepath]/framework/base/src/main/java/org/apache/ofbiz/base/util/ObjectType.java
[ofbiz-basepath]/framework/base/src/main/java/org/apache/ofbiz/base/util/test/ObjectTypeTests.java
```

As we can see, `PlainString` is user thoughout the whole project, also outside the `minilang` package. It is even referenced into some `.xsd` files and into a Groovy script. Seems that `PlainString` is an entity, probably a field type or something similar, even if the number or references into the codebase (24) seems too low to support the idea `PlainString` is a fundamental entity for the project.

In addition, the documentation shows no references about `PlainString`. Also the online version of the Javadoc (found here: https://ci.apache.org/projects/ofbiz/site/javadocs/org/ofbiz/minilang/MiniLangUtil.PlainString.html) is of no use, as the class itself has no Javadoc.

While it remains unclear why developers should feel the need to use such an empty class to (probably) identify a plain string, and if `PlainString` is actually the way OFBiz deals with plain strings, the location of this class looks completely wrong. `PlainString` should have been located with other basic field types, and into a dedicated Java source file, instead of being a nested class of a randomly chosen class of the project.

4 Conclusions

4.1 Tools used

During the development of this document we used the following tools:

- **Github** to version control the project
- **LaTeX** on TeXworks to redact this document
- **SonarQube Scanner 2.8** to double-check the results of the manual code inspection.

4.2 Hours of work

- SZ: 2h on 10/01
- SZ: 6h on 11/01
- SM: 3h on 12/01
- SZ: 7h on 13/01
- SZ: 1h on 14/01
- SM: 1h on 22/01
- SZ: 1h on 23/01