Code Inspection

Software Engineering 2

February 5,2017

POLITECNICO
MILANO 1863

Authors:
Claudio Salvatore Arcidiacono Matr 879109
Antonio Di Bello Matr 878786
Denis Dushi Matr 879115

Contents

1 Class that was assigned to the group

../apache-ofbiz-16.11.01/specialpurpose/passport/src/main/java/org/apache/ofbiz/passport
/user/LinkedInAuthenticator.java

2 Functional role of assigned set of classes

The class allows a user to authenticate in the OFBiz application using his LinkedIn credentials. The first time the user logs in, a new OFBiz account is created using the LinkedIn information. Every other time the user authenticates with his LinkedIn credentials the OFBiz account's information are synchronized with the LinkedIn ones.

The class should provide also the possibility to log out and update the password but these functionalities haven't been implemented yet.

Reading the code and following the methods calls helped us inferring the functional role of the class. In particular have been useful as a clue the names of the attributes and the Javadoc documentation, even if incomplete.

3 Issues

3.1 Naming Conventions

3.1.1 Not meaningful names:

- In line 213 the method "getLinkedInUserinfo" does not respect the convention, the last word that compose the method's name should be capitalized.
- The attribute "responseString" in line 404 should have been called only "response", it's a bad habit to include the type in the variable name.

3.1.2 Constants not declared using all uppercase:

- Line 66, "module"
- Line 68, "props"
- Line 70, "resource"

3.2 Indention

The Indention is made up to four blank space and it's used consistently. No tabs are present.

3.3 Braces

3.3.1 Inconsistent bracing style:

- In this class is used consistently the "Kernighan and Ritchie" bracing style.

3.4 File Organization

3.4.1 Blank lines and optional comments to separate section

- Lines 18-19: Misses a blank line between comment section and import section.

3.4.2 Lines longer than 80 characters

- Line: 18, 101, 105, 107, 111, 112, 131, 146, 157, 164, 177, 192, 197, 207, 213, 217, 219, 223, 248, 255, 259, 277, 282, 287, 297, 304, 309, 315, 319, 326, 334, 339, 346, 359, 382, 396, 399, 404, 406, 409, 420, 441, 452, 453, 456, 457, 460, 461.

3.4.3 Lines longer than 120 characters

- Line: 101, 107, 111, 164, 219, 223, 287, 399, 409, 420.

3.5 Wrapping Lines

Code lines are never wrapped even if they exceed 120 characters. Specifically the lines are: 101, 107, 111, 164, 219, 223, 287, 399, 409, 420

3.6 Comments

3.6.1 Comments that don't reflect what the code is doing

- Functions at lines 347, 368, 377, 386 don't reflect the effect described in the relative comment.

3.6.2 Commented out code that doesn't contain a reason for being commented out

- Line 270 contains commented code, and there isn't a valid reason for this.
- Line 406 contains commented code that it's needed for debugging purpose.

3.7 Java Source Files

3.7.1 External program interfaces implemented inconsistently with what is described in the javadoc

- The parameter "userLoginId" of the method "authenticate" in line 101 is not consistently reported in the javadoc. Instead is reported as "username" (line 94).

3.7.2 Incomplete javadoc

- Javadoc documentation is totally absent for methods in lines : 245, 399, 415, 434. These methods are public and it is expected an external usage of them, therefore the lack of documentation is very concerning.
- Private methods in lines 213, 255, 346 are not documented as well.
- In line 148 the Javadoc documentation misses the description of the parameter "userLoginId"

3.8 Package and Import Statements

Package statements are the first non-comment statements. Import statements follow.

3.9 Class and Interface Declarations

3.9.1 Class (static) variables order:

- Line 66...70 : the order (public, protected, package, private) is not respected, the private variable *module* is written before the two public variables *props* and *resource*.

3.9.2 Methods not grouped by functionality:

- the methods *getLinkedInUserId, getUserInfo* and the methods *getLinkedInUserinfo* have the same functionality (get user information) so they should be grouped together.

3.9.3 Code duplicates, long methods, big classes, breaking encapsulation:

- The method *createUser* (lines 255...311) is too long and it does several functions. It should be split into more private methods, one for each function.
- The code of the methods *authenticate* line 101 and *getLinkedInUserinfo* line 213 are quite the same. So the correct from would be *authenticate* that calls *getLinkedInUserinfo* and then do the rest of the computation.

3.10 Initialization and Declarations

3.10.1 Variables and class members not of the correct type or visibility:

Attributes:

- the static attribute *resource* (line 70) is declared as public but is used only in the class LinkedInAuthenticator so its visibility should be private.
- the attribute *dispatcher* (line 72) is declared as protected but is used only in the class LinkedInAuthenticator so its visibility should be private.
- the attribute *delegator* (line 74) is declared as protected but is used only in the class LinkedInAuthenticator so its visibility should be private.

Methods:

- the method *getUserInfo* (line 399) is declared as public but is used only in the class LinkedInAuthenticator so we don't understand why the visibility its declared as public(maybe future uses), it should be declared as private.
- the method *parseLinkedInUserInfo* (line 434) is declared as public but is used only in the class LinkedInAuthenticator so we don't understand why the visibility its declared as public(maybe future uses), it should be declared as private.

3.10.2 Variables not declared in the proper scope:

- the variable beganTransaction at line 171 is not declared in the proper scope. the proper scope would be inside the try catch block, at line 173.

3.10.3 Declarations not at the beginning of blocks :

- Due to the fact that the method *createUser* (lines 255...311) does multiple functions and that it should be split into several function we find that the declarations of its variables is sparse around the method and not at the beginning of the block. Looking inside the internal blocks of the methods at line 302 and 332 we have declarations that are not at the beginning of the block.
- the variables *standardProfileRequest* (line 420) and *url* (line 421)in the method *getLinkedInUserId* (lines 415... 432) are not declared at the beginning of the block .
- All the variables from the line 440 to the line 460 of the methods *parseLinkedInUserInfo*(lines 434...465) are not declared at the beginning of the block.

3.11 Method Calls

Methods are correctly called and the returned values are properly used. The parameters are passed in the correct order.

3.12 Arrays

The collections used are Map that are always initialized.
Arrays are used properly.

3.13 Object Comparison

Comparisons between object are executed correctly.

3.14 Output Format

At line 360 the debug message is ambiguous.

3.15 Computation, Comparisons and Assignments

3.15.1 Liberal use of parenthesis:

- At line 315 there is an unnecessary use of parenthesis, more specifically the parenthesis at columns 37...110

3.15.2 Comparisons and Boolean operators:

- At line 417 we have an if condition stating "*persons.getLength() <= 0*" . *persons* is of type NodeList and the method getLenght returns the number of elements in the node list, so the usage of less or equal is inappropriate(a list can't contain a negative number of elements). The correct form would be persons.getLength() == 0 .

3.16 Exceptions

3.16.1 The relevant exceptions are caught

- Line 275 and 284 : line 284 should be moved in the try block, in order to avoid a possible null pointer exception.

3.16.2 Appropriate actions are taken for each catch block

- Line 192: The debug message is wrong, it should uses "begin" instead of "suspend".
- Lines 176, 191, 196, 206, 320: the catch blocks generate only a debug message, without trying to workaround the problem.
- Lines 115 -123 and 226-235: the catch blocks only throw another exception of different type, that will be managed from the caller of the function. All the catch blocks do the same things, so they can be grouped into only one catch.
- Line 173: try block isn't associated at any catch block, is present only the finally block. Furthermore, internally at the try block there are some try/catch block. This may result in confusing code and should have their catch and finally sections merged.

3.17 Flow of Control

All loops are correctly formed. There aren't switch statements.

3.18 Files

Files are not used in this class.

4 Any other problems

- In line 107 to the method UtilMisc.toMap is passed a probably wrong String parameter due to a typo. Should be “LinkedInUser” instead of "linedInUserId". This is the source of a bug because the function can’t derive the information of the LinkedIn user and will not permit the authentication.
- The method "authenticate" (line 101) requires two parameters "String password" and "boolean isServiceAuth" that are never used inside the function. We guess that the password is managed and verified externally. The fact that in lines 272/273 the password of a new user is set "[EXTERNAL]" support that thesis.
- Functions at line 340, 347, 368, 377, 386 have not been implemented.
- In line 366 and 373 are present the following typos : "Authenicator", "synchronzied".It will bring bugs probably.
- The structure of the functions *createUser* at line 245 and 255 is totally wrong because the function at line 245 just call the function at line 255 that is too long and does too many function. The correct form would be to have a public function that calls a set of private function.

5 Effort spent

During the whole project the team worked with the following schedule:

Claudio Salvatore Arcidiacono

- 1 February : 10-18 6 hours (2 hours of break)
- 3 February : 18-21 3 hours

  Total hours: 9 hours.

Antonio Di Bello

- 26 January : 4 hours
- 3 February : 3 hours

  Total hours: 7 hours.

Denis Dushi

- 2 February : 10-18 6 hours (2 hours of break)
- 3 February : 18-21 3 hours

  Total hours: 8 hours.

6 References

6.1 Software and tool used

The tools we used to create this document are:

- Overleaf (for latex writing in parallel)