POLITECNICO
MILANO 1863

MyTaxiService

Code Inspection Document

Davide Citterio, Lorenzo Cunial, Massimo Beccari

January 4, 2016

Contents

1 Assigned classes

The class that has been assigned to our group is class **InjectionManagerImpl**, located in *appserver/common/container-common/src/main/java/com/sun/ /enterprise/container/common/impl/util*.
This class is the implementation of the InjectionManager interface, located in *appserver/common/container-common/src/main/java/com/sun/ /enterprise/container/common/spi/util*.

In particular, it's our task to inspect the following methods (contained in InjectionManagerImpl.java):

1. **createManagedObject**(Class<T> clazz, boolean invokePostConstruct)
2. **destroyManagedObject**(Object managedObject, boolean validate)
3. **inject**(final Class clazz, final Object instance, JndiNameEnvironment envDescriptor, String componentId boolean invokePostConstruct)
4. **invokePreDestroy**(final Class clazz, final Object instance, JndiNameEnvironment envDescriptor)
5. **invokePostConstruct**(final Class clazz, final Object instance, JndiNameEnvironment envDescriptor)
6. **__inject**(final Class clazz, final Object instance, String componentId, List <InjectionCapable> injectableResources)

2 Functional role of assigned classes

The class that has been assigned to us, as suggested by its name, deal with the management of the Java EE feature of Injection. In particular, it takes care of the Resource Injection, as declared in the InjectionManager interface Javadoc, which our class implements. It has been possible to identify the functional role of this class also by looking at the public methods that the class implements, which perform the injection of classes and instances. This class also provides the generic lifecycle callback methods for the injected classes and instances (the invocation of PostConstruct and PreDestroy methods).

3 List of issues

Here follow all the issues we have found in the methods assigned to us, divided by the method's name in which the issue has been found.
The issues are coded with a number that refers to the issue's type listed in the code inspection checklist given to us to perform the code inspection.

3.1 Method createManagedObject

Problems identified:

- [**1**] At line 408, *noArgCtor* is not a immediately meaningful name;
- [**13**] The line 387 is composed by 81 char;
- [**18**] Ortography error at line 406: annotated instead of annoated.

3.2 Method destroyManagedObject

Problems identified:

- [**13**] The line 453 is composed by 95 char;
- [**13**] The line 455 is composed by 81 char.

3.3 Method inject

Problems identified:

- [**1**] The first parameter, final Class clazz, hasn't got a meaningful name;
- [**15**] At the end of line 514, a line break split the method's name *getPostConstructMethod* from the method's arguments *(injInfo, nextClass).*

3.4 Method invokePreDestroy

Problems identified:

- [**1**] The first parameter, final Class clazz, hasn't got a meaningful name;
- [**15**] At the end of line 559, a line break split the method's name *getPreDestroyMethod* from the method's arguments *(injInfo, nextClass)*;
- [**23**] The Javadoc specifies only the *instance* parameter
  (it could be accettable considering that this method is private).

3.5 Method invokePostConstruct

Problems identified:

- **[15]** At line 600, a line break split the method's name *getPostConstructMethod* from the method's arguments *(injInfo, nextClass)*;
- issue **[17]** at line 596;
- **[22]** In the Javadoc the interface PostConstruct is not linked;

3.6 Method _inject

Problems identified:

- **[6]** At line 621, the first word is not a verb but an underscore plus a verb;
- **[8]** No indention (line 627);
- **[8]** Only two spaces are used for indention (lines 683, 684, 720);
- **[8]** At lines 723-724 there are wrong indentions caused by the one in the row 722;
- **[8]** Too long indention at line 637;
- **[11]** At line 647, the instruction isn't covered by braces;
- **[12]** The line 718 isn't separated from the one before by a blank line;
- issue **[13]** at lines 638, 655, 657, 663, 664, 665, 673, 697, 698, 704, 705, 706, 713;
- issue **[14]** at line 698;
- issue **[17]** at lines 681, 721.

3.7 Class issues

Problems identified:

- **[25]** At line 88, the static variable *localStrings* is declared after the instance variable *_logger.*

4 Other problems

We haven't identified any other problem about the methods that were assigned to us.

5 Work time

To produce this document, each member have worked around 9 hours.