

Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Computer Science and Engineering
A.Y. 2015-2016



Software Engineering 2 Project
“myTaxiService”
Code Inspection

January 5, 2016

Principal Adviser: Prof. **Di Nitto**

Authors:

Davide Fisicaro 854043

Gianmarco Giummarra 852667

Salvatore Ferrigno 850130

Contents

1	Code Inspection	1
1.1	Assigned Class	1
1.2	Functional role of the assigned Class	1
1.3	List of issues found by applying the checklist	1
1.3.1	callEJBLoad(...)	2
1.3.2	preInvokeNoTx(...)	2
1.3.3	invokeTargetBeanMethod(...)	3
1.3.4	addToCache(...)	4
1.3.5	postFind(...)	4
1.3.6	equals(...)	4
1.4	Other problems	5

1 Code Inspection

1.1 Assigned Class

The Class to check for the writing of this document is the **ReadOnlyBeanContainer.java** class, located in the following path: *appserver/persistence/entitybean-container/src/main/java/org/glassfish/persistence/ejb/entitybean/container/ReadOnlyBeanContainer.java*.

The actual purpose of this code inspection is to check some of the methods of the class mentioned above.

Here is the complete list:

- **callEJBLoad**(EntityBean ejb , EntityContextImpl entityCtx , boolean activeTx)
- **preInvokeNoTx**(EjbInvocation inv)
- **invokeTargetBeanMethod**(Method beanClassMethod , EjbInvocation inv , Object target , Object [] params , com . sun . enterprise . security . SecurityManager mgr)
- **addToCache**(Object primaryKey , boolean incrementRefCount)
- **postFind**(EjbInvocation inv , Object primaryKeys , Object [] findParams)
- **equals**(Object o)

1.2 Functional role of the assigned Class

The actual scope of this Class is to model a Container which manages the ReadOnly Java Beans. A Container is an interface, implemented in four main classes: StatefulSessionContainer, StatelessSessionContainer, EntityContainer, and MessageBeanContainer; in particular, the assigned class extends **EntityContainer**. The EntityContainer class is responsible for instance and **lifecycle management for BMP and CMP EntityBeans**, so it has an important role in the data persistence. According to the Javadocs page¹ relevant to the ReadOnlyBeanContainer Class, this Container should block the calls of the `ejbStore()` and selectively perform the `ejbLoad()`. The way this behavior is adopted is implementing the `callEJBStore()` method, because of the relevant interface implementation of the extended EntityContainer Class, but not performing any actions in there, neither throwing any exceptions.

1.3 List of issues found by applying the checklist

The following sub-sections show the list of issues found by applying the checklist; for each issue is explicated the relevant point on the checklist, surrounded by

¹ReadOnlyBeanContainer Javadoc on javadocs.com

square brackets. In the analyzed methods the chosen used bracing style is the “Kernighan and Ritchie”.

The reference to the lines are absolute and don’t take in account the various proposed changes.

1.3.1 calleJBLoad(...)

This method is used to keep the local cache in sync with the primary copy of the data.

1. The open curly bracket should be moved to line 254, in order to follow the Kernighan and Ritchie style; [10]
2. Blank lines have to be removed in the following lines: 276, 283, 292, 300, 305, 307, 313, 318, 320, 330, 332, 339, 345; [12]
3. The following blank lines have to be removed because useless: 308, 321; [12]
4. The following lines should be divided in order not to exceed the 80 characters line length: 256, 285, 286, 289, 290. [13]
5. In the following lines the line break occurs before the operator, but should appear after it: 278, 288, 289, 302, 315, 348; [15]
6. There is a non necessary line break in the line 253/254; [15]
7. In the following lines, a high level break would be preferred, but in this way the lines will exceed 80 characters: 278-281, 288-290, 302-303, 315-316; [16]
8. Javadoc is missing for this method; [23]
9. Variables “pkLevelSequenceNum” (line 268) should be declared with private visibility, due to the fact that they are used only in this method. [28]
10. Line 289 contains a non-comprehensive output: the description of the printed variable shouldn’t be the path of the variable. In fact “robInfo-lastRefreshedAt:” should be changed into “last refreshed at:”; [42]
11. In lines 280-281 there is no distinction between “beanLevelSequenceNum” and “robInfo” in the output. A separator should occur between the two outputs. [43]

1.3.2 preInvokeNoTx(...)

This method is called from BaseContainer just before invoking a business method whose tx attribute is TX_NEVER / TX_NOT_SUPPORTED / TX_SUPPORTS without a client tx.

1. The if statement in the line 420 is not surrounded by curly braces; [11]
2. Blank lines have to be removed in the following lines: 422, 432, 439; [12]
3. Blank lines have to be added between lines 417 and 418; [12]
4. The initial comments is not sufficient to explain what the method is doing; [18]
5. Javadoc is missing for this method; [23]

1.3.3 `invokeTargetBeanMethod(...)`

This method handles the invocation of the method for the Bean. Depending on the “`invocationInfo.startsFind`” attribute, the method calls the superclass’ `invokeTargetBeanMethod(...)` or manages in a different way the actions to do, by creating a new object of the private Class “`FinderResultsKey`” and assigning to a new “`FinderResultValue`” object a yet existing value (`CuncurrentHashMap`).

1. In the line 552, could be better to give a different name than “value” to an object which will have an attribute called value: see the lines 557, 561 and 579 for the “... `value.value`;” statement; [1]
2. Lines 581 and 582, the closed curly braces have to be shifted of four characters backward; [10]
3. Blank lines have to be added in the following position: 552, 554, 555, 565, 566, 568, 570, 572, 583; [12]
4. In order not to exceed the 80 characters line length, the method header (line 541) needs a line break after the first comma; [13]
5. In this method there are no comments; a little explanation could help the maintenance; [18]
6. Javadoc is missing for this method, this of course made difficult to understand the method’s behaviour; [23]
7. In the line 576 the constructor of `FinderResultValue` class is called, and this call is the parameter of another method. This is not a best practice and an object should be created through the constructor, then passed to the `put(...)` method; [31]² and [33]³
8. Objects are often compared using the “`==`” operator instead of the “`equals`” function. For instance at line 553, 565 and 573; [40]
9. This methods throws “`Throwable`” but nor Errors or Exceptions are managed. [52] and [53]

²Check that all object references are initialized before use.

³Declarations appear at the beginning of blocks.

1.3.4 addToCache(...)

This method manages the Read Only Beans cache, and it optimizes the operations whenever the cache item already exists. If the item doesn't exist, the method creates a new one and provides and initialize all the information needed.

1. The if statement in line 769 doesn't follow the "Kernighan and Ritchie" style like the rest of the method; [10]
2. The line 729 has to be removed because it's useless; [12]
3. In order not to exceed the 80 characters line length, the method header (line 541) needs a line break after the first comma; [13]
4. Javadoc is missing for this method, this of course made difficult to understand the method's behaviour; [23]

1.3.5 postFind(...)

This method always calls parent to convert pks to ejbobjects/ejblocalobjects.

The "PrimaryKeys" parameter can be instance of Collection or Enumeration etc. Basing on which of type the parameter is instance of, method "updateRobInfoAfterFinder" is invoked.

1. Four Spaces are used for indentation, but from the line 827 to 833 are used more than four spaces; [8]
2. The open curly bracket should be moved to line 806, in order to follow the Kernighan and Ritchie style; [10]
3. Blank lines have to be removed in the following lines: 807, 849; [12]
4. There is a non necessary line break in the line 804/805; [15]
5. Javadoc is missing for this method; [23]

1.3.6 equals(...)

This method indicates whether an object is equal to another or not. The equals method implements an equivalence relation on non-null object references

1. Automatic tabulation is used two times at line 938 (last two indentations); [9]
2. The following blank lines could be removed, because they do not separate any section: 935, 939, 943, 948, 951, 953, 956, 960, 967, 969; [12]
3. In the line 937 the line break occurs before the operator, but should appear after it; [15]
4. Variable "equal" (line 932) should be declared with private visibility, due to the fact that it's used only in this method; [28]

5. In the line 830, 839, 844 the objects are compared with “!=” instead of “equals”; [40]
6. This methods throws “FinderException” but no Exceptions are managed. [52] and [53]

1.4 Other problems

1. In the line 125 the variable “RELATIVE_TIME_CHECK_MODE” name only upper case characters; [6]