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
Code **I**nspection
Version 1.1

January 19, 2016

Principal Adviser: Prof. **Di Nitto**

Authors:
Davide Fisicaro 854043
Gianmarco Giummarra 852667
Salvatore Ferrigno 850130

Contents

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 persintence. According to the Javadocs page[1] relevant to the ReadOnlyBeanContainer Class, this Cointainer 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 explicited the relevant point on the checklist, surrounded by

[1] ReadOnlyBeanContainer 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]

```
252     protected void callEJBLoad(EntityBean ejb, EntityContextImpl entityCtx,
253                                boolean activeTx)
254         throws Exception
255     {
256         // ReadOnlyContextImpl should always be used in conjunction with ReadOnlyBeanContainer
257         assert entityCtx instanceof ReadOnlyContextImpl;
258         ReadOnlyContextImpl context = (ReadOnlyContextImpl) entityCtx;
```

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 added: after 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]

```
285             } else if (RELATIVE_TIME_CHECK_MODE && (refreshPeriodInMillis > 0)) { // 0 implies no time based refresh
286                 if ((currentTimeInMillis - robInfo.lastRefreshedAt) > refreshPeriodInMillis) {
287                     if (_logger.isLoggable(Level.FINE)) {
288                         _logger.log(Level.FINE, "REFRESH DUE TO STALE PK:"
289                                 + " robInfo.lastRefreshedAt: " + robInfo.lastRefreshedAt
290                                 + "; current (approx) time is " + currentTimeInMillis);
291                     }
```

5. In the following lines the line break occurs before the operator, but should appear after it: 278, 288, 289, 302, 315, 348; [15]

```
278                     _logger.log(Level.FINE, "REFRESH DUE TO BEAN-LEVEL UPDATE:"
279                                 + " Bean-level sequence num = " +
280                                 beanLevelSequenceNum +
281                                 robInfo + " current time is " + new Date());
282                 }
```

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]

```
301                 if( _logger.isLoggable(Level.FINE) ) {
302                     _logger.log(Level.FINE, " PK-LEVEL REFRESH : "
303                                 + robInfo + " current time is " + new Date());
304                 }
```

8. Javadoc is missing for this method; [23][2]

```
protected void callEJBLoad(javax.ejb.EntityBean ejb,
EntityContextImpl entityCtx, boolean activeTx) throws
Exception

Javadoc not found. Either Javadoc documentation for this
item does not exist or you have not added specified Javadoc
in the Java Platform Manager or the Library Manager.
```

9. 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]

```
287                 if (_logger.isLoggable(Level.FINE)) {
288                     _logger.log(Level.FINE, "REFRESH DUE TO STALE PK:"
289                             + " robInfo.lastRefreshedAt: " + robInfo.lastRefreshedAt
290                             + "; current (approx) time is " + currentTimeInMillis);
291                 }
```

10. In lines 280-281 there is no distinction between "*beanLevelSequenceNum*" and "*robInfo*" in the output. A separator should occur between the two outputs. [43]

```
277                 if( _logger.isLoggable(Level.FINE) ) {
278                     _logger.log(Level.FINE, "REFRESH DUE TO BEAN-LEVEL UPDATE:"
279                             + " Bean-level sequence num = " +
280                             beanLevelSequenceNum +
281                             robInfo + " current time is " + new Date());
282                 }
```

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]

```
419
420         if ( context.isInState(BeanState.DESTROYED) )
421             return;
422
```

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]

```
413
414     // Called from BaseContainer just before invoking a business method
415     // whose tx attribute is TX_NEVER / TX_NOT_SUPPORTED / TX_SUPPORTS
416     // without a client tx.
417     protected void preInvokeNoTx(EjbInvocation inv) {
418         EntityContextImpl context = (EntityContextImpl)inv.context;
```

5. Javadoc is missing for this method; [23]

```
protected void preInvokeNoTx(com.sun.ejb.EjbInvocation inv)

Javadoc not found. Either Javadoc documentation for this
item does not exist or you have not added specified Javadoc
in the Java Platform Manager or the Library Manager.
```

[2]The output is generated from NetBeans when the Javadoc for the method is asked.

1.3.3 invokeTargetBeanMethod(...)

This method handles the invocation of the method for the Bean. Depending on the "invocationInfo.startsWithFind" 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]

```
552             FinderResultsValue value = finderResultsCache.get(key);
553             if (value != null) {
554                 if (RELATIVE_TIME_CHECK_MODE && (refreshPeriodInMillis > 0)) {
```

2. Lines 581 and 582, the closed curly braces have to be shifted of four characters backward; [10]

```
578                     } else {
579                             returnValue = value.value
580                     }
581                     }
582                 }
583
584         } else {
585             returnValue =  super.invokeTargetBeanMeth
```

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]

```
539     }
540
541     protected Object invokeTargetBeanMethod(Method beanClassMethod, EjbInvocation inv,
542                                             Object target, Object[] params,
543                                             com.sun.enterprise.security.SecurityManager mgr)
```

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]

```
protected Object invokeTargetBeanMethod(Method
beanClassMethod, com.sun.ejb.EjbInvocation inv, Object
target, Object[] params, SecurityManager mgr) throws
Throwable

Javadoc not found. Either Javadoc documentation for this
item does not exist or you have not added specified Javadoc
in the Java Platform Manager or the Library Manager.
```

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] [3] and [33] [4]

```
575                         beanClassMethod, inv, target, params, mgr);
576                     finderResultsCache.put(key, new FinderResultsValue(returnValue,
577                         currentTimeInMillis));
578                 } else {
579                         returnValue = value.value;
```

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 exists, 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]

```
768             // created is the "truth" for this pk.
769             robInfo = (otherRobInfo == null) ? newRobInfo : otherRobInfo;
770         }
```

2. The line 729 has to be removed because it's useless; [12]

```
728             // puts.
729
730             ReadOnlyBeanInfo newRobInfo = new ReadOnlyBeanInfo();
```

3. In order not to exceed the 80 characters line length, the method header (line 715) needs a line break after the first comma; [13]

```
714
715     private ReadOnlyBeanInfo addToCache(Object primaryKey, boolean incrementRefCount) {
716
717         // Optimize for the cache where the cache item already
```

4. Javadoc is missing for this method, this of course made difficult to understand the method's behaviour; [23]

```
private ReadOnlyBeanInfo addToCache(Object primaryKey,
boolean incrementRefCount)

Javadoc not found. Either Javadoc documentation for this
item does not exist or you have not added specified Javadoc
in the Java Platform Manager or the Library Manager.
```

[3] Check that all object references are initialized before use.
[4] Declarations appear at the beginning of blocks.

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 identation, but from the line 827 to 833 are used more than four spaces; [8]

```
826             if ( primaryKeys instanceof Enumeration ) {
827                 Enumeration e = (Enumeration) primaryKeys;
828                 while ( e.hasMoreElements() ) {
829                     Object primaryKey = e.nextElement();
830                     if( primaryKey != null ) {
831                         updateRobInfoAfterFinder(primaryKey);
832                     }
833                 }
```

2. The open curly bracket should be moved to line 805, in order to follow the Kernighan and Ritchie style; [10]

```
                        Object[] findParams)
        throws FinderException
    {
```

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]

```
803     public Object postFind(EjbInvocation inv, Object primaryKeys,
804                            Object[] findParams)
805         throws FinderException
```

5. Javadoc is missing for this method; [23]

```
public Object postFind(com.sun.ejb.EjbInvocation inv, Object
primaryKeys, Object[] findParams) throws FinderException

Javadoc not found. Either Javadoc documentation for this
item does not exist or you have not added specified Javadoc
in the Java Platform Manager or the Library Manager.
```

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]

```
937             if ((params.length == other.params.length)
938                     && (finderMethod.equals(other.finderMethod))) {
```

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]

```
936             FinderResultsKey other = (FinderResultsKey) o;
937             if ((params.length == other.params.length)
938                     && (finderMethod.equals(other.finderMethod))) {
```

1.4 Other problems

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