My Taxi Service

Code **I**nspection

Authors:

Andrea DONATI {andrea4.donati@mail.polimi.it}

Gabriele CARASSALE {gabriele.carassale@mail.polimi.it}

Manuel DELEO {manuel.deleo@mail.polimi.it}

Prof: Elisabetta Di Nitto

Milan, January 5, 2016

Contents

1 Introduction

The goal of this Code Inspection document is to check the correctness of the given code of the GlassFish Server with respect to a defined list of rules.
In particular the code is reviewed from a formal point of view, focusing on the correctness of the style, in order to avoid the use of "bad constructs" or the presence of other errors.

2 Assigned Class

We have been assigned with eight methods contained in the EntityContainer class. EntityContainer is part of the "org.glassfish.persistence.ejb.entitybean.container" package.

2.1 Functional Role

The EntityContainer class is responsible for instance and lifecycle management for BMP and CMP EntityBeans. It also handles the concurrency when there are multiple concurrent transactions on a EntityBean.

3 Found Issues

3.1 Class and Variables Declarations

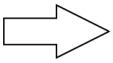
- line 192: private variable should go after package-level variables
- line 198: la variabile static final Logger logger dovrebbe essere scritta in maiuscolo
- static variables should appear before other variables
- lines 243-245: protected variables should go before package level variables
- lines 252-255: protected variables should go before package level variables

EntityContainer is the only public class in the EntityContainer.java file. EntityContainer and the other not-public classes respect all the declaration rules. The package declaration is the first statement, then there are all the package imports and finally the class declaration. On the other hand we found some issues in the declarations of the class variables, not respecting the naming rules and the following order:

- Static Class Variables
  - public
  - protected
  - package level
  - private
- Instance Variables
  - public
  - protected
  - package level
  - private

Here it's presented a before/after representation in order to understand all the corrections applied to the code:

```
public class EntityContainer
    extends BaseContainer
    implements CacheListener
{

    private ThreadLocal ejbServant = new ThreadLocal() {
        protected Object initialValue() {
            return null;
        }
    };

    static final Logger _logger =
        LogDomains.getLogger(EntityContainer.class, LogDomains.EJB_LOGGER);

    static final int POOLED=1, READY=2, INVOKING=3,
            INCOMPLETE_TX=4, DESTROYED=5;
    protected static final int HIGH_WATER_MARK=100;

    private static final int DEFAULT_TX_CACHE_BUCKETS = 16;

    // table of EJBObjects, indexed by primary key.
    // Note: Hashtable methods are synchronized.
    protected EJBObjectCache ejbObjectStore;

    // table of EJBLocalObjectImpls, indexed by primary key.
    // Note: Hashtable methods are synchronized.
    protected EJBObjectCache ejbLocalObjectStore;

    //protected  LIFOChannel channel = null;
    protected Stack         passivationCandidates = new Stack();

    // table of EJBs (Contexts) in READY state, key is primary key
    protected Cache readyStore;

    //Pool of free EntityContexts
    protected AbstractPool  entityCtxPool;

    protected boolean isReentrant;
    protected boolean isContainerManagedPers;

    protected final float DEFAULT_LOAD_FACTOR = 0.75f;
    protected final int DEFAULT_CACHE_SIZE = 8192;
    protected int _maxBuckets = 8;

    protected IASEjbExtraDescriptors iased = null;
    protected BeanCacheDescriptor beanCacheDes = null;
    protected BeanPoolDescriptor beanPoolDes = null;
    protected EjbContainer ejbContainer;
    boolean largeCache = false;

    CacheProperties cacheProp = null;
    PoolProperties poolProp = null;
    Object asyncTaskSemaphore = new Object();
    boolean addedASyncTask = false;

    // a timer task to trim the beans idle in readyStore
    protected IdleBeansPassivator   idleEJBObjectPassivator;
    protected IdleBeansPassivator   idleLocalEJBObjectPassivator;
    protected boolean               defaultCacheEJBO = true;

    IdleBeansPassivator idleBeansPassivator;
    boolean timerValid = true;
    long idleTimeout;


    protected int ejboRemoved;

    protected int   totalPassivations;
    protected int   totalPassivationErrors;

    private EntityCacheStatsProvider    cacheStatsProvider;
```



```
public class EntityContainer
    extends BaseContainer
    implements CacheListener
{
    protected static final int HIGH_WATER_MARK=100;
    static final int POOLED=1, READY=2, INVOKING=3,
            INCOMPLETE_TX=4, DESTROYED=5;

    static final Logger _logger =
        LogDomains.getLogger(EntityContainer.class, LogDomains.EJB_LOGGER);

    private static final int DEFAULT_TX_CACHE_BUCKETS = 16;

    // table of EJBObjects, indexed by primary key.
    // Note: Hashtable methods are synchronized.
    protected EJBObjectCache ejbObjectStore;

    // table of EJBLocalObjectImpls, indexed by primary key.
    // Note: Hashtable methods are synchronized.
    protected EJBObjectCache ejbLocalObjectStore;

    //protected  LIFOChannel channel = null;
    protected Stack         passivationCandidates = new Stack();

    // table of EJBs (Contexts) in READY state, key is primary key
    protected Cache readyStore;

    //Pool of free EntityContexts
    protected AbstractPool  entityCtxPool;

    protected boolean isReentrant;
    protected boolean isContainerManagedPers;

    protected final float DEFAULT_LOAD_FACTOR = 0.75f;
    protected final int DEFAULT_CACHE_SIZE = 8192;
    protected int _maxBuckets = 8;

    protected IASEjbExtraDescriptors iased = null;
    protected BeanCacheDescriptor beanCacheDes = null;
    protected BeanPoolDescriptor beanPoolDes = null;
    protected EjbContainer ejbContainer;

    // a timer task to trim the beans idle in readyStore
    protected IdleBeansPassivator   idleEJBObjectPassivator;
    protected IdleBeansPassivator   idleLocalEJBObjectPassivator;
    protected boolean               defaultCacheEJBO = true;

    protected int ejboRemoved;

    protected int   totalPassivations;
    protected int   totalPassivationErrors;

    boolean largeCache = false;

    CacheProperties cacheProp = null;
    PoolProperties poolProp = null;
    Object asyncTaskSemaphore = new Object();
    boolean addedASyncTask = false;

    IdleBeansPassivator idleBeansPassivator;
    boolean timerValid = true;
    long idleTimeout;

    private EntityCacheStatsProvider    cacheStatsProvider;

    private ThreadLocal ejbServant = new ThreadLocal() {
        protected Object initialValue() {
            return null;
        }
    };
```

3.2 passivateAndPoolEJB

signature: passivateAndPoolEJB(EntityContextImpl context)

This method is called from "addReadyEJB" and "afterCompletion". If the given Bean is not DESTROYED or POOLED, the method will passivate and add it to the pool.
The complete checklist with all the found issues relative to this method is provided below.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are used

Indention

8. ERROR: lines 2171-2177 are not indented properly
9. OK

Braces

10. OK: The bracing style is "Kernighan and Ritchie" is used in a consistent way
11. OK

File Organization

12. OK
13. ERROR: lines 2161 and 2171 are longer than 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. OK
19. OK

Java Source Files

20. //
21. //
22. //
23. //

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declaration

28. OK
29. OK
30. OK
31. OK
32. OK
33. ERROR: Variables declared in lines 2171 and 2179 are not at the beginning of their blocks

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK: comparisons with NULL are performed with "==" operator

Output Format

41. OK
42. OK

43. OK

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK: No try-catch expressions are used
51. OK

Exceptions

52. OK: No exceptions are thrown or caught
53. OK: No exceptions are thrown or caught

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
// called from addReadyEJB and afterCompletion
protected void passivateAndPoolEJB(EntityContextImpl context) {
    if ( context.isInState(BeanState.DESTROYED) ||
            context.isInState(BeanState.POOLED) ) {
        return;
    }

    // missing REASON and DATE
    // if ( context.isPooled() ) {
    // context.isPooled(false);
    // return;
    // }
    EntityBean ejb = (EntityBean) context.getEJB();
    synchronized ( context ) {
        Object primaryKey;
        EjbInvocation inv = super.createEjbInvocation(ejb, context);
        inv.method = ejbPassivateMethod;
        invocationManager.preInvoke(inv);

        try {
            ejb.ejbPassivate();
        } catch ( Exception ex ) {
            _logger.log(Level.FINE,"Exception in passivateAndPoolEJB()",ex);
            forceDestroyBean(context);
            return;
        } finally {
            invocationManager.postInvoke(inv);
        }

        // remove EJB(Local)Object from ejb(Local)ObjectStore

        primaryKey = context.getPrimaryKey();
        if ( isRemote ) {
            removeEJBObjectFromStore(primaryKey);
        }
        if ( isLocal ) {
            ejbLocalObjectStore.remove(primaryKey);
        }

        addPooledEJB(context);
    }
}
```

3.3 addIncompleteTxEJB

signature: addIncompleteTxEJB(EntityContextImpl context)

This method is called only from "afterBegin". This EJB is invoked either with client's transaction or with a new transaction. In the first case the transaction would already be in the table, but not in the second case.
The complete checklist with all the found issues relative to this method is provided below.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are used

Indention

8. ERROR: lines 2171-2177 are not indented properly
9. OK

Braces

10. OK: The bracing style is "Kernighan and Ritchie" is used in a consistent way
11. OK

File Organization

12. OK
13. ERROR: lines 2161 and 2171 are longer than 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. OK
19. OK

Java Source Files

20. //
21. //
22. //
23. //

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declaration

28. OK
29. OK
30. OK
31. OK
32. OK
33. ERROR: Variables declared in lines 2171 and 2179 are not at the beginning of their blocks

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK: comparisons with NULL are performed with "==" operator

Output Format

41. OK
42. OK

43. OK

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK: No try-catch expressions are used
51. OK

Exceptions

52. OK: No exceptions are thrown or caught
53. OK: No exceptions are thrown or caught

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
/**
 * Called only from afterBegin.
 * This EJB is invoked either with client's tx (in which case
 * it would already be in table), or with new tx (in which case
 * it would not be in table).
 */
private void addIncompleteTxEJB(EntityContextImpl context) {
    Vector beans;
    ActiveTxCache activeTxCache;
    JavaEETransaction current =
            (JavaEETransaction) context.getTransaction();
    if ( current == null ) {
        return;
    }
    if ( (containerStateManager.isNullEJBObject(context)) &&
         (containerStateManager.isNullEJBLocalObject(context)) ) {
        return;
    }

    // Its ok to add this context without checking if its already there.
    activeTxCache =
            (ActiveTxCache) ejbContainerUtilImpl.getActiveTxCache(current);
    if (activeTxCache == null) {
        activeTxCache = new ActiveTxCache(DEFAULT_TX_CACHE_BUCKETS);
        ejbContainerUtilImpl.setActiveTxCache(current, activeTxCache);
    }

    activeTxCache.add(context);

    beans = ejbContainerUtilImpl.getBeans(current);
    beans.add(context);
}
```

3.4 removeIncompleteTxEJB

signature: removeIncompleteTxEJB(EntityContextImpl context, boolean updateTxBeanTable)

This method is called from "releaseContext" if the EJB is removed, from "afterCompletion", and from "passivateEJB". The complete checklist with all the found issues relative to this method is provided below.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are declared

Indention

8. ERROR: lines 2199-2202 are not indented properly
9. ERROR: in lines 2199-2202 tabs are used instead of spaces

Braces

10. OK: The bracing style is "Kernighan and Ritchie" is used in a consistent way
11. OK

File Organization

12. OK
13. ERROR: lines 2189 and 2199 are longer than 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. OK
19. // There's no commented out code

Java Source Files

20. //
21. //
22. OK
23. OK

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declaration

28. OK
29. OK
30. OK
31. OK
32. OK
33. ERROR: Variable "activeTxCache" declared in line 2199 is not at the beginning of its block

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK: comparisons with NULL are performed with "==" operator

Output Format

41. // There's no displayed output
42. // There are no error messages
43. // There's no displayed output

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK: No try-catch expressions are used
51. OK

Exceptions

52. OK: No exceptions are thrown or caught
53. OK: No exceptions are thrown or caught

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
/**
 * Called from releaseContext if ejb is removed, from afterCompletion,
 * and from passivateEJB.
 */
protected void removeIncompleteTxEJB(EntityContextImpl context,
                                     boolean updateTxBeanTable) {
    ActiveTxCache activeTxCache;
    JavaEETransaction current =
            (JavaEETransaction) context.getTransaction();

    if (current == null) {
        return;
    }
    if ( (containerStateManager.isNullEJBObject(context)) &&
         (containerStateManager.isNullEJBLocalObject(context)) ) {
        return;
    }

    activeTxCache =
            (ActiveTxCache) ejbContainerUtilImpl.getActiveTxCache(current);
    if (activeTxCache != null) {
        activeTxCache.remove(this, context.getPrimaryKey());
    }

    if ( updateTxBeanTable ) {
        Vector beans = ejbContainerUtilImpl.getBeans(current);
        beans.remove(context); // this is a little expensive...
    }
}
```

3.5 equals

signature: equals(Object obj)

This public method overrides the standard 'equals' functions and its purpose is to compare two EntityBeans in the INCOMPLETE-TX state by analyzing their primary keys and transaction contexts.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are used

Indention

8. OK
9. OK

Braces

10. OK: The bracing style is "Kernighan and Ritchie" is used in a consistent way
11. OK

File Organization

12. OK
13. OK
14. OK

Wrapping Lines

15. ERROR: Line 2269, line break after parenthesis
16. OK
17. OK

Comments

18. OK
19. OK

Java Source Files

20. //
21. //
22. //
23. //

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declaration

28. OK
29. OK
30. OK
31. OK
32. OK
33. OK

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK

Output Format

41. OK
42. OK
43. OK

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK
51. OK

Exceptions

52. OK
53. OK

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
public final boolean equals(Object obj) {
    if ( !(obj instanceof EJBTxKey) ) {
        return false;
    }
    EJBTxKey other = (EJBTxKey) obj;
    try {
        if ( primaryKey.equals(other.primaryKey) ) {
            if ( (tx == null) && (other.tx == null) ) {
                return true;
            } else if ( (tx != null) && (other.tx != null) &&
                    tx.equals(other.tx) ) {
                return true;
            } else  {
                return false;
            }
        } else {
            return false;
        }
    } catch ( Exception ex ) {
        _logger.log(Level.FINE, "Exception in equals()", ex);
        return false;
    }
}
```

3.6 CacheProperties

signature: CacheProperties(EntityContainer entityContainer)

This static class, called at the time of creation of the EntityContainer, provides all information about the cache, such as the maximum size and the idle timeout.
It should be noted that the instance variables of the class are called directly from other methods; getter methods should be implemented and these variables should have private visibility.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. OK
4. // it's not an interface
5. OK
6. OK
7. OK: no constants are used

Indention

8. OK
9. OK

Braces

10. ERROR: The bracing style "Kernighan and Ritchie" is used, but lines 2309-2310 use the "Allman" style
11. OK

File Organization

12. OK
13. ERROR: Lines 2294-95, 2297, 2301, 2305, 2309 exceed 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. ERROR: Line 2315, useless comment
19. OK

Java Source Files

20. //
21. //
22. //
23. //

Package and Import Statements

24. OK

Class and Interface Declarations

25. OK
26. OK
27. OK

Initialization and Declaration

28. ERROR: No clear visibility for instance variables
29. OK
30. OK
31. OK
32. OK
33. OK

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK

Output Format

41. OK
42. OK
43. OK

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK
51. OK

Exceptions

52. OK: No exceptions are thrown or catched
53. OK: No exceptions are thrown or catched

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
protected static class CacheProperties {

    int maxCacheSize ;
    int numberOfVictimsToSelect ;
    int cacheIdleTimeoutInSeconds ;

    public CacheProperties(EntityContainer entityContainer) {
        numberOfVictimsToSelect =
            Integer.parseInt(
                    entityContainer.ejbContainer.getCacheResizeQuantity());
        maxCacheSize = Integer.parseInt(
                entityContainer.ejbContainer.getMaxCacheSize());
        cacheIdleTimeoutInSeconds =
            Integer.parseInt(
                    entityContainer.ejbContainer.getCacheIdleTimeoutInSeconds());

        if(entityContainer.beanCacheDes != null) {
            int temp = 0;
            if((temp =
                    entityContainer.beanCacheDes.getResizeQuantity()) != -1) {
                numberOfVictimsToSelect = temp;
            }

            if((temp =
                    entityContainer.beanCacheDes.getMaxCacheSize()) != -1) {
                maxCacheSize = temp;
            }

            if ((temp =
                    entityContainer.beanCacheDes.getCacheIdleTimeoutInSeconds()) != -1) {
                cacheIdleTimeoutInSeconds = temp;
            }
        }
    }
}
```

3.7 PoolProperties

signature: PoolProperties(EntityContainer entityContainer)

This class is called from the ContainerFactory during initialization and it provides all information about the pool that has to be created, such as its maximum size, its resize quantity and its idle timeout.
The same arguments about instance variables' visibility presented for the CacheProperties class are valid for the PoolProperties class too.

CHECKLIST
Naming Conventions

1. OK
2. OK: No one-character variables are used
3. OK
4. // it's not an interface
5. OK
6. OK
7. OK

Indention

8. OK
9. OK

Braces

10. OK: The bracing style "Kernighan and Ritchie" is used in a consistent way
11. OK

File Organization

12. OK
13. ERROR: Line 2338 exceeds 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. ERROR: line 2349, useless comment
19. ERROR: Line 2320, unclear comment;

Java Source Files

20. //
21. //
22. //
23. //

Package and Import Statements

24. OK

Class and Interface Declarations

25. OK
26. OK
27. OK

Initialization and Declaration

28. ERROR: No clear visibility for instance variables
29. OK
30. OK
31. OK
32. OK
33. OK

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK

Output Format

41. OK
42. OK
43. OK

Computation, Comparisons and Assignments

44. OK
45. OK
46. OK
47. OK
48. OK
49. OK
50. OK
51. OK

Exceptions

52. OK: No exceptions are thrown or catched
53. OK: No exceptions are thrown or catched

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
private static class PoolProperties {
    int maxPoolSize;
    int poolIdleTimeoutInSeconds;
    int poolResizeQuantity;
    int steadyPoolSize;

    public PoolProperties(EntityContainer entityContainer) {

        maxPoolSize = Integer.parseInt(entityContainer.ejbContainer.getMaxPoolSize());
        poolIdleTimeoutInSeconds = Integer.parseInt(
            entityContainer.ejbContainer.getPoolIdleTimeoutInSeconds());
        poolResizeQuantity = Integer.parseInt(
            entityContainer.ejbContainer.getPoolResizeQuantity());
        steadyPoolSize = Integer.parseInt(
            entityContainer.ejbContainer.getSteadyPoolSize());
        if(entityContainer.beanPoolDes != null) {
            int temp = 0;
            if ((temp =
                    entityContainer.beanPoolDes.getMaxPoolSize()) != -1) {
                maxPoolSize = temp;
            }
            if ((temp =
                    entityContainer.beanPoolDes.getPoolIdleTimeoutInSeconds()) != -1) {
                poolIdleTimeoutInSeconds = temp;
            }
            if ((temp =
                    entityContainer.beanPoolDes.getPoolResizeQuantity()) != -1) {
                poolResizeQuantity = temp;
            }
            if ((temp =
                    entityContainer.beanPoolDes.getSteadyPoolSize()) != -1) {
                steadyPoolSize = temp;
            }
        }
    }
}
```

3.8 isIdentical

signature: isIdentical(EJBObjectImpl ejbObjImpl , EJBObject other)

This method is called to check if an EJBObjectImpl and an EJBOject are the same object. It first compares the stub of ejbObjectImpl, next the primary key.

CHECKLIST
Naming Conventions

1. OK
2. OK: no one-character variables are used
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are used

Indention

8. OK
9. OK

Braces

10. OK
11. ERROR: line 2360: if statement without curly braces

File Organization

12. OK
13. ERROR: line 2371: exceed 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. OK

Comments

18. OK
19. OK // not contain commented code.

Java Source Files

20. //

21. //
22. OK
23. OK

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declarations

28. OK
29. OK
30. OK
31. OK
32. OK
33. OK

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. ERROR: line 2354: two objects are compared with "=="

Output Format

41. OK
42. OK
43. OK

Computation, Comparisons and Assignments

44. OK

45. Ok
46. OK
47. OK
48. OK
49. OK
50. OK
51. OK

Exceptions

52. OK
53. OK

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. // No loops are present in the method

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
protected boolean isIdentical(EJBObjectImpl ejbObjImpl, EJBObject other)
        throws RemoteException {
    if ( other.equals(ejbObjImpl.getStub()) ) {
        return true;
    } else {
        try {
            // EJBObject may be a remote object.
            // Compare homes. See EJB2.0 spec section 9.8.
            if ( !getProtocolManager().isIdentical(ejbHomeStub,
                                                   other.getEJBHome())) {
                return false;
            }

            // Compare primary keys.
            if (!ejbObjImpl.getPrimaryKey().equals(other.getPrimaryKey())) {
                return false;
            }

            return true;
        } catch ( Exception ex ) {
            _logger.log(Level.INFO,
                "entitybean.container.ejb_comparison_exception",
                        logParams);
            _logger.log(Level.INFO, "", ex);
            throw new RemoteException("Exception in isIdentical()", ex);
        }
    }
}
```

3.9 doConcreteContainerShutdown

signature: doConcreteContainerShutdown(boolean appBeingUndeployed)

This method is called to destroy all EJBObject refs and NULL the other variables before making the undeploy.

CHECKLIST
Naming Conventions

1. OK
2. OK // at line 2490 it is used the variable 'e' but it is temporary
3. // it's not a class
4. // it's not an interface
5. OK
6. // it's not a class
7. OK: no constants are used

Indention

8. OK
9. OK

Braces

10. OK
11. OK

File Organization

12. ERROR: line 2462, 2499, 2514 do not require a blank line
13. ERROR: line 2518: exceed 80 characters
14. OK

Wrapping Lines

15. OK
16. OK
17. ERROR: line 2496: the line is not indent

Comments

18. OK
19. OK

Java Source Files

20. //

21. //
22. OK
23. OK

Package and Import Statements

24. // It's not a class

Class and Interface Declarations

25. // It's not a class
26. // It's not a class
27. OK

Initialization and Declarations

28. OK
29. OK
30. OK
31. OK
32. OK
33. OK

Method Calls

34. OK
35. OK
36. OK

Arrays

37. // No arrays are used
38. // No arrays are used
39. // No arrays are used

Object Comparison

40. OK

Output Format

41. OK
42. OK
43. OK

Computation, Comparisons and Assignments

44. OK

45. OK
46. OK
47. OK
48. OK
49. OK
50. OK
51. OK

Exceptions

52. OK
53. OK

Flow of Control

54. // No switch statements are used in the method
55. // No switch statements are used in the method
56. ERROR: loop from line 2456: elements variable is not increased

Files

57. // No files are used in the method
58. // No files are used in the method
59. // No files are used in the method
60. // No files are used in the method

```
protected void doConcreteContainerShutdown(boolean appBeingUndeployed) {

    String ejbName = ejbDescriptor.getName();

    if(_logger.isLoggable(Level.FINE)) {
        _logger.log(Level.FINE,"[EntityContainer]: Undeploying " + ejbName +
            " ...");
    }
    // destroy all EJBObject refs

    try {
        Iterator elements = ejbObjectStore.values();
        while ( elements.hasNext() ) {
            EJBObjectImpl ejbObjImpl = (EJBObjectImpl) elements.next();
            try {
                if ( isRemote ) {
                    remoteHomeRefFactory.destroyReference
                        (ejbObjImpl.getStub(), ejbObjImpl.getEJBObject());
                }
            } catch ( Exception ex ) {
                _logger.log(Level.FINE, "Exception in undeploy()", ex);
            }
        }

        ejbObjectStore.destroy();  //store must set the listern to null
        ejbObjectStore = null;

        ejbLocalObjectStore.destroy(); //store must set the listern to null
        ejbLocalObjectStore = null;

        // destroy all EJB instances in readyStore
        destroyReadyStoreOnUndeploy(); //cache must set the listern to null

        entityCtxPool.close();
        poolProbeListener.unregister();
        if (cacheProbeListener != null) {
            cacheProbeListener.unregister();
        }
```

```
            // stops the idle bean passivator and also removes the link
            // to the cache; note that cancel() method of timertask
            // does not remove the task from the timer's queue
            if (idleBeansPassivator != null) {
                try {
                    idleBeansPassivator.cancel();
                } catch (Exception e) {
                    _logger.log(Level.FINE,
                                "[EntityContainer] cancelTimerTask: ", e);
                }
                this.idleBeansPassivator.cache  = null;
            }
            cancelTimerTasks();
        }
        finally {
            // helps garbage collection
            this.ejbObjectStore          = null;
            this.ejbLocalObjectStore     = null;
            this.passivationCandidates   = null;
            this.readyStore              = null;
            this.entityCtxPool           = null;
            this.iased                   = null;
            this.beanCacheDes            = null;
            this.beanPoolDes             = null;
            this.ejbContainer            = null;
            this.cacheProp               = null;
            this.poolProp                = null;
            this.asyncTaskSemaphore      = null;
            this.idleBeansPassivator     = null;
        }

        if(_logger.isLoggable(Level.FINE)) {
            _logger.log(Level.FINE,
                " [EntityContainer]: Successfully Undeployed " + ejbName);
        }
    }
```

4 Times spent working on this document

Group Member	**Total Hours**
Andrea Donati	12
Gabriele Carassale	11
Manuel Deleo	11