SE2 Inspection Document

Edoardo Giacomello Mattia Fontana

December 30, 2015

Contents

1 Assigned Classes and Methods

Assigned Class: BaseContainer.java **Location**:
appserver/ejb/ejb-container/src/main/java/com/sun/ejb/containers/BaseContainer.java
Package: com.sun.ejb.containers Methods to Inspect:

1. **Name**:mapLocal3xException(Throwable t)
   - **Start Line**:2337
2. **Name**: authorize(EjbInvocation inv)
   - **Start Line**:2362
3. **Name**:initializeEjbInterfaceMethods( )
   - **Start Line**:2408
4. **Name**:getJaccEjb(EjbInvocation inv)
   - **Start Line**:2676
5. **Name**:assertValidLocalObject(Object o)
   - **Start Line**:2725

2 Functional Roles

This section will explain what is the functional role of the class and methods we analysed and will describe the process that have been used in order to discover these functional roles.

2.1 WorkFlow

For getting a better understanding of the analysed component functional roles and the general context, the following steps have been followed:

1. Javadoc inspection of the assigned class, with respect to implemented interfaces, subclasses and implementers.
2. Reading of the document "Enterprise JavaBeansTM Specification Version 2.0", in particular of the section regarding the container contract and functionalities overview
3. Finding the usage of the methods to analyse by using the grep tool

4. Documentation inspection and usage analysis of caller methods and their classes
5. Documentation inspection of the methods that have been assigned for code review
6. Functional Inspection for the code of the methods that have been assigned
7. Definition of the Scope for the methods that have been assigned
8. Code inspection

2.2 Package overview

The package com.sun.ejb.containers provides all the classes needed for implementing an EJB container, which can be either **Stateful** or **Stateless**, an Entity Bean container, or Message Bean Container.
It also provides classes that implement the container Home interface, which defines the methods for the client to create, remove, and find EJB objects of the same type (EJBHomeImpl class).

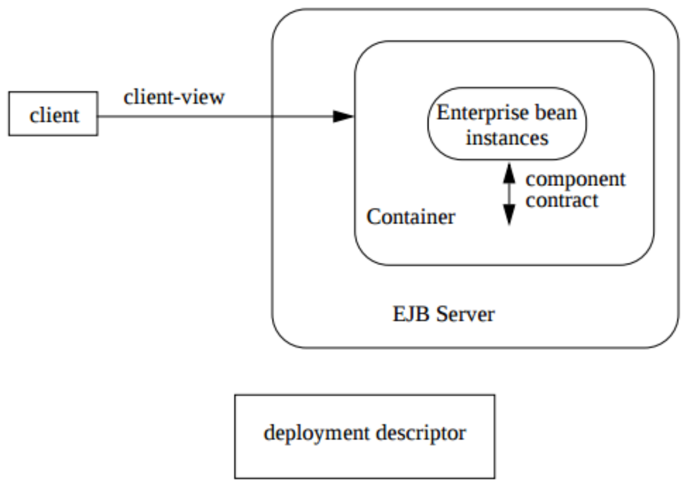
2.3 BaseContainer Class

In this section will be described the main scope of the class that contains the analysed methods.

The **BaseContainer** class implements the Container interfaces as stated in the ***EJB 2.0 specifications***. It hosts the code that is shared between the *Session Beans*, *Entity Beans* and *Message Driven Beans*.

The scope of this class is therefore to provide a common interface between the different types of Java Bean Containers. The context of operation can be inferred by the following diagram, included in the Java Bean Specification document:

Figure 1 Enterprise JavaBeans Contracts



Note that while the figure illustrates only a remote client running outside of the Container, the client-view APIs are also applicable to local clients and to remote clients that are enterprise Beans deployed in the same Container.

An analysis of the code revealed that this class in particular manages the object that contains an EJB Method invocation in several context such as Authorization, Initialization, Pre-Invoking, Post-Invoking etc.

2.3.1 Interfaces

This class implements directly the following interfaces:

Container : This interface is the main contract for a EJB Container implementation. In this case the container is a specific implementation of this interface (see BaseContainer subclasses) and it is responsible for managing the lifecycle, state management, concurrency, transactions etc, by interposing actions before and after invocations on EJBs. The methods that have been analysed are specified in this interface.

JavaEEContainer : The javadoc does not specifies a description for this interface, but the method names suggest that it provides some utility methods for all the JEE containers, such the retrieval of the component Id and the container descriptor.

EjbContainerFacade : This interface provides ejb-specific methods for iiop middleware integration, which is a protocol for distributed systems that supports the mapping between TCP/IP and Inter-Object Request Broker messages.

2.3.2 Subclasses

The **BaseContainer** class is derived by the following classes, each of them implementing a different type of EJB container.

EntityContainer : This class represents a container for an Entity Bean and It is responsible for their instances and lifecycle management. In particular, this type of container (*EJB Spec 2.0, section 10.5.9*) does not ensure that the instance has exclusive access to the state of the object in persistence storage, and the container must therefore synchronize the instance's state at the beginning of a transaction.

MessageBeanContainer This class provides container functionality specific to message-driven EJBs. At deployment time, one instance of the MessageDrivenBeanContainer is created for each message-driven bean in an application. (*Class Javadoc*)

StatefulSessionContainer This class provides container functionality specific to stateful SessionBeans. At deployment time, one instance of the StatefulSessionContainer is created for each stateful SessionBean type (i.e. deployment descriptor) in a JAR. (*Class Javadoc*)

StatelessSessionContainer This class provides container functionality specific to stateless SessionBeans. At deployment time, one instance of the StatelessSessionContainer is created for each stateless SessionBean type (i.e. deployment descriptor) in a JAR.
This container services invocations using a pool of EJB instances. An instance is returned to the pool immediately after the invocation completes, so the number of instances needed = number of concurrent invocations.

A Stateless Bean can hold open DB connections across invocations. Its assumed that the Resource Manager can handle multiple incomplete transactions on the same connection.

AbstractSingletonContainer Called from the JarManager at deployment time.

2.3.3 Class Body

The **BaseContainer** class includes the following nested Classes:

ContainerInfo This class contains strings for monitoring the container information.

ContainerType This enum specifies the type of the container, that can be Entity, MessageDriven, ReadOnly, Singleton, Stateful or Stateless

PreInvokeException This is a wrapper for the exceptions thrown from BaseContainer.preInvoke, so it indicates that the bean's method will not be called. *(from Javadoc)* The preInvokeMethod is a method which is called from the EJB home or object before the invocation of the bean method.

2.4 Terminology and other Components

This section contains all the specific terminology and components that have been referred to during the functional description of the analysed code.

2.4.1 Local and Remote Clients

From Oracle Documentation:
A **local client** has these characteristics.

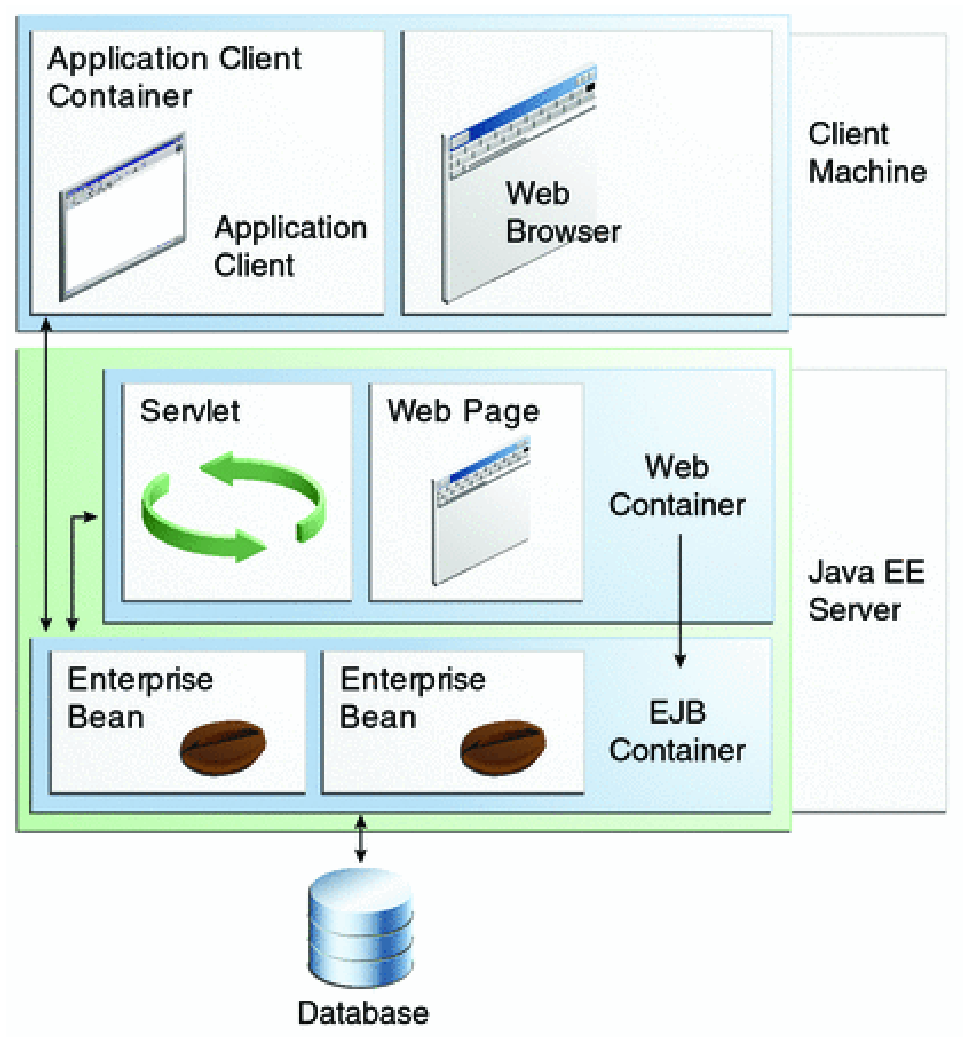
- It must run in the same application as the enterprise bean it accesses.
- It can be a web component or another enterprise bean.
- To the local client, the location of the enterprise bean it accesses is not transparent.

A **remote client** of an enterprise bean has the following traits.

- It can run on a different machine and a different JVM from the enterprise bean it accesses. (It is not required to run on a different JVM.)
- It can be a web component, an application client, or another enterprise bean.
- To a remote client, the location of the enterprise bean is transparent.
- The enterprise bean must implement a business interface. That is, remote clients may not access an enterprise bean through a no-interface view.

2.4.2 EJB Container

Containers are the interface between a component and the low-level platform-specific functionality that supports the component. Before it can be executed, a web, enterprise bean, or application client component must be assembled into a Java EE module and deployed into its container.
A more explicative description is given in the picture below:



2.4.3 EJB Home

From EJBHome javadoc:
The EJB Home is an interface that defines the methods that allow a remote client to create, find, and remove EJB objects.
The remote home interface is defined by the enterprise bean provider and implemented by the enterprise bean container.
Enterprise beans written to the EJB 3.0 and later APIs do not require a home interface.

2.4.4 EJB Local Object

From Oracle JavaDoc:
An enterprise bean's local interface provides the local client view of an EJB object. An enterprise bean's local interface defines the business methods

callable by local clients. The enterprise bean's local interface is defined by the enterprise bean provider and implemented by the enterprise bean container. Enterprise beans written to the EJB 3.0 and later APIs do not require a local interface that extends the EJBLocalObject interface. A local business interface can be used instead.

2.4.5 EJB Invocation

The EjbInvocation object contains the state associated with an invocation on an EJB or EJBHome (local/remote). It is usually created by generated code in *ObjectImpl and *HomeImpl classes. It is passed as a parameter to Container.preInvoke() and postInvoke(), which are called by the EJB(Local)Object/EJB(Local)Home before and after an invocation.

2.4.6 JACC: Java Authorization Contract for Containers

From Oracle Documentation: The Java Authorization Contract for Containers (JACC) specification defines a contract between a Java EE application server and an authorization policy provider. All Java EE containers support this contract.

The JACC specification defines java.security.Permission classes that satisfy the Java EE authorization model. The specification defines the binding of container access decisions to operations on instances of these permission classes. It defines the semantics of policy providers that use the new permission classes to address the authorization requirements of the Java EE platform, including the definition and use of roles.

2.5 Methods

All considered methods are implemented in the **BaseContainer** class.

2.5.1 mapLocal3xException

Visibility : Private Method.

Definition Class : **BaseContainer** class

Called from :

- method **postInvoke** in class **BaseContainer**, row **2124**
- method **mapRemoteException** in class **BaseContainer**, row **2292**

Usage Analysis :

- The first usage occurrence is into the **postInvoke** method of this same class. The postInvoke method is a method which is called from the EJB Home or Container after the invocation of the bean method. In the case an exception is raised and the invocation is not remote, the exception dynamic type is mapped by the **mapLocal3xException** method and stored into the **EJBInvocation** object.
- The second usage occurrence is into the **mapRemoteException** method of this same class. The mapRemoteException method checks if a remote exception invocation is asynchronous: in that case, as stated in comment lines, we are sure that the exception is raised by a remote business interface and not from a 2.x client, so it has to be mapped as a local exception by the **mapLocal3xException** method.

Functional Description : This method consists in a check over the dynamic type of a **Throwable** taken as input. If the instance matches one of the exception type defined, it is re-instantiated as a corresponding non-local exception and returned.

Scope : Although no javadoc is available for this method, through the usage and code analysis, with respect to the Oracle Javadoc of the package javax.ejb it has been possible to deduce that this is an helper method, which maps a Local EJB exception of the 3.x version into another corresponding exception that could be sent to the client.

2.5.2 authorize

Visibility : Public Method

Definition Class : **Container** interface

Called from :

- method **preInvoke** in class **BaseContainer**, row **1959**
- method **authorizeLocalMethod** in class **BaseContainer**, row **2162**
- method **authorizeRemoteMethod** in class **BaseContainer**, row **2185**
- method **invoke** in class **WebServiceInvocationHandler**, row **2185**
- method **authorizeWebService** in class **EjbInvocation**, row **675**

Usage Analysis :

- **preInvoke** is the method which is called from the EJB container before the invocation of the actual EJB method. It checks if the state of the method invocation is legitimate or it would lead to exceptions.
  The call of the authorize method is done in the context of security checking: if the authorize method returns false, an exception is raised which states that the client is not authorized to make that method invocation.
- the **authorizeLocalMethod** and **authorizeRemoteMethod** are called from the local/remote container home or object respectively in order to authorize the execution of a EJB method; the usage of the authorize method is similar to that in the previous point.
- **WebServiceInvocationHandler** is a proxy invocation handler for web service ejb invocations.
  It calls the authorize method for checking if the client is authorized to call a certain method through that proxy
- The **EjbInvocation** is the object that contains the state of an EJB Method invocation. No javadoc is available, but the name and the code suggest that the authorize method is used to authorize a web service method call by accessing the container that own the invocation itself.

Functional Description : The method first try to fetch the method invocation associated information and attaches it to the invocation object, because it would improve performance.
Then it checks if the called method has been called from the business home interface, in that case it will return true; if not the method will call the authorize method of the security manager. If the security manager doesn't authorize the invocation, its context is released.

Scope : The JavaDoc states that this method contains the common code for managing the security manager authorization call.
In practice, this method is useful to assert if the client who calls an EJB method is authorized to make that invocation.
By code inspection it is possible to understand that this method will make a check on the source of the called EJB method and authorize it automatically or invoke the security manager instead.

2.5.3 initializeEjbInterfaceMethods

Visibility : Private Method

Definition Class : **BaseContainer** class

Called from :

- **Constructor** of **BaseContainer** class, row **840**.

Usage Analysis :

- This method is called by the class Constructor during the initialization process of a Container.

Functional Description : This method creates an array of Methods that will contains all methods of the EJB interface, according to its type (Local or Remote, Stateless or Stateful).

Scope : This method adds by reflection the interface methods that the BaseContainer class has implemented and assign the produced array of methods to the local ejb home or local object.

2.5.4 getJaccEjb

Visibility : Public method

Definition Class : **Container** interface.

Called from :

- method **getJaccEjb** in class **EjbInvocation**, row **368**

- Indirectly by method **getEnterpriseBean** in class **EJBPolicy-ContextDelegate**, row **60** (see Usage Analysis)

Usage Analysis :

- The BaseContainer getJaccEjb method is only called in the method with the same name which below to the invocation itself. The javadoc of the invocation version of the method states that the user shall call the getJaccEjb method on the invocation object rather than directly on the EJB field, but it just call the Container method and return its value.

- The EJBPolicyContextDelegate is a delegate for the Policy Context and it calls getJaccEjb for returning the bean that is owned by the invocation object.

Functional Description : The operation flow of this method is based on several assumptions that are specified in the comments.
First of all the method make a check on the invocation passed as parameter, it has to be a business method invocation done through a remote, local or serviceEndpoint interface. Then if the context for the invocation has not been set, it is done and the ejb for that context is returned. There an important consideration about the accessed variable is made in the comments, but it doesn't affect the way the method works.

Scope : This method retrieves the Java Bean from the invocation context. This is necessary for the JACC policy provider (see Terminology section).

2.5.5 assertValidLocalObject

Visibility : Public method

Definition Class : **Container** interface

Called from :

- method **assertValidLocalObject** in class **SunContainerHelper**, row **240**
- Indirectly, method **assertValidLocalObject** in class **CMPHelper**, row **234**
- Indirectly, method **assertValidLocalObjectImpl** in class **JDOEJB20HelperImpl**, row **243**

Usage Analysis :

- This method is used in helper class that manages the Container-Managed persistance for java beans.
  In particular, the JDOEJB20HelperImpl is an helper class that is useful to convert persistance-capable beans from and to single object and collection of these.

Functional Description : This method receives as input an object, and it will raise an exception in the case the object is not a local valid one, or just return otherwise. It starts checking if the object is null and if it's an instance of a EJBLocalObject. In that case, it checks if the container of that object is the same of the object in which the assertValidLocalObject method is called. If the check don't passes, an error message is built and an exception is thrown.

Scope : The scope of this method is to check if the object passed as parameter is a Local object and belong to this container. It is used prevalently for the bean persistence management.

3 Checklist and Issues

4 Other problems

5 References

- Assignment document part 3: Document structure and checklist
- "Brutish Programming", Dr. John Dalbey: Code quality inspection
- http://glassfish.pompel.me/ Javadoc for Glassfish
- http://www.javadocumentation.com/ Javadoc for the specific package we analysed
- Enterprise JavaBeansTM Specification Version 2.0 - Sun Microsystems
- Oracle Documentation: Component descriptions and some JavaDocs

6 Tools

- **SVN**: For the checkout of the source code
- **Gedit**: Text editor for code inspection
- **GrepCode**: Preliminary package analysis
- **grep**: For usage inspection

7 Work Hours

- Mattia Fontana:
- Edoardo Giacomello: