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 Functional Inspection Process

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 classes and methods, 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

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).

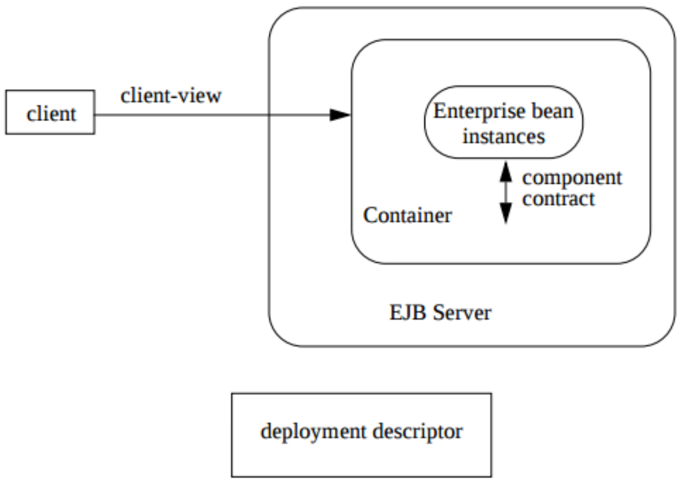
2.3 BaseContainer Class

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

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.

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.

2.4 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.

2.5 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 :

Scope :

2.5.3 initializeEjbInterfaceMethods

Visibility : Private Method

Definition Class : **BaseContainer** class

Called from :

- method in class , row

Usage Analysis :

-

Functional Description :

Scope :

2.5.4 getJaccEjb

Visibility : Public method

Definition Class : **Container** interface.

Called from :

- method in class , row

Usage Analysis :

-

Functional Description :

Scope :

2.5.5 assertValidLocalObject

Visibility : Public method

Definition Class : **Container** interface

Called from :

- method in class , row

Usage Analysis :

-

Functional Description :

Scope :

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

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: