Enterprise JavaBeans

Layer 04:

Interfaces
Agenda

- Discuss the home and remote interfaces.
- Discuss the relationships between the interfaces and the bean class.
- Discuss issues with bean's in general.
I'll continue to use the same example that I used before, specifically the **Book** bean.
Remote Interface
Purpose

- The *remote interface* is used by the client to invoke business methods on an EJB.
- The EJB server generates the classes that implement this interface.
- The stub and skeleton classes that implement this interface are collectively called the *EJBOBJECT*. 
Structure

- The remote interface must extend the **EJBObjekt** interface.

- Any business methods that the client needs access should be declared.

- If you don't want the client to see a method, don't add it to the remote interface.
1. package se554.layer03.ejb;

2. import java.rmi.*;
3. import javax.ejb.*;

4. public interface Book
   extends EJBObject {
5.   public String getTitle()
       throws RemoteException;

6.   public String getISBN()
       throws RemoteException;

7. }

Exceptions

- Each method must throw the `RemoteException`. This is how all system exceptions will be returned to the client.
- Other exceptions can be declared as needed to represent various application exceptions.
EJBOBJECT Interface

- The **EJBOBJECT** interface provides common functionality shared by all **EJBOBJECT** instances.
- Recall that the **EJBOBJECT** is a distributed object. These objects live in the EJB server.
EJBObject

1. public interface EJBObject
   extends Remote {
2.   public EJBHome getEJBHome
       throws RemoteException;
3.   public Handle getHandle()
       throws RemoteException;
4.   public Object getPrimaryKey()
       throws RemoteException;
5.   public boolean isIdentical(EJBObject obj)
       throws RemoteException;
6.   public void remove()
       throws RemoteException, RemoveException;
7. }

Last Revised: 10/2/2001  Copyright (C) 2001
getEJBHome

- Returns a remote reference to the home object that was used to create this EJBObjec
- Generally, we don't use this method.
getHandle

• Returns a handle to the EJB. This handle can be stored by the client code and used to "re-connect" to the bean instance in the event of a client failure.

• Handles are discussed in more detail later.
**getPrimaryKey**

- This method is only valid for entity beans. Attempting to invoke it on a session bean will result in a `RemoteException`.

- Returns the primary key object represented by that bean reference.

- We'll discuss primary keys when we look at entity beans.
isIdentical

• Acts as the `equals()` method for EJB references.

• `isIdentical()` returns true iff:
  – The beans are stateless session beans from the same home.
  – The beans are entity beans from the same home with the same primary key.
- Removes the target EJB from the server.
- This may result in a call to the bean class' `ejbRemove()` method.
- The container may de-allocate the bean class instance and will de-allocate the server side `EJBObject`. 
Home Interface
Purpose

- The *home interface* is used by the client to locate or create references to an EJB.
- The EJB server generates the classes that implement this interface.
- The stub and skeleton classes that implement this interface are collectively called the **EJBHome**.
Structure

- The home interface must extend the \texttt{EJBHome} interface.

- Any creator or finder methods that the client needs to access, should be declared.
1. package se554.layer03.ejb;

2. import java.rmi. *;
3. import javax.ejb. *;

4. public interface BookHome
   extends EJBHome {
5.       public Book create(String title,  
                          String ISBN)  
          throws CreateException, RemoteException;
6. }

BookHome
Creating a Bean
(1 of 3)
Creating a Bean
(2 of 3)
Creating a Bean
(3 of 3)
The `create()` method acts as the bean factory method.

This will be used by the client to gain a reference to the bean's remote interface.

Once this reference has been generated, it can be used by the client to invoke business methods on the bean itself.
A home interface's `create()` methods must meet the following rules:

- It must have a return type of the bean's remote interface.
- It can be overloaded unless the bean is a stateless session bean in which case only the no-arg `create()` is permissible.
- It can throw any kind of exceptions.
EJBHome Interface

- The **EJBHome** interface provides common functionality shared by all home interfaces.
- Recall that the *home object* is a distributed object. These objects live in the EJB server.
1. public interface EJBHome
   extends Remote {

2.   public EJBMetaData getEJBMetaData
        throws RemoteException;

3.   public HomeHandle getHomeHandle()
        throws RemoteException;

4.   public void remove(Handle handle)
        throws RemoteException, RemoveException;

5.   public void remove(Object primaryKey)
        throws RemoteException, RemoveException;

6.   }
getEJBMetaData

- Returns a reference to the EJBMetaData that represents this EJB.
- We'll look at the EJBMetaData interface in a later slide.
- The EJBMetaData interface and its implementation will be provided by the server vendor.
getHomeHandle

- Returns a reference to the `HomeHandle` that represents this EJB's home object.
- The `HomeHandle` can be used to "re-connect" to a home object in the event of a client failure.
- Handles are discussed in more detail later.
The first version of the `remove()` method accepts an EJB's handle and uses that to remove the bean.

This is the same as if you invoke the `remove()` method on the bean's `EJBOBJECT` directly.
The second version of the \texttt{remove()} method accepts an EJB's primary key.

This variation of the method is only useful for entity beans. Attempting to invoke it on a session bean results in a \texttt{RemoteException}.
EJBMetaData

1. public interface EJBMetaData {
2. public EJBHome getEJBHome();
3. public Class getHomeInterfaceClass();
4. public Class getPrimaryKeyClass();
5. public Class getRemoteInterfaceClass();
6. public boolean isSession();
7. public boolean isStatelessSession();
8. }

getEJBHome

- Returns a reference to the EJBHome that created this EJB.
getHomeInterfaceClass

- Returns a `java.lang.Class` reference describing the home interface used to build this EJB.
- This is typically used by tool providers to provide graphical representations of beans.
getPrimaryKeyClass

- Returns a `java.lang.Class` reference describing the primary key class of the EJB.
- This method is only defined for entity beans. Invoking it on a session bean will throw a `RuntimeException`.
- This is typically used by tool providers to provide graphical representations of beans.
getRemoteInterfaceClass

- Returns a `java.lang.Class` reference describing the remote interface used to build this EJB.
- This is typically used by tool providers to provide graphical representations of beans.
isSession

- Returns **true** if the bean is either a stateful or stateless session bean and **false** otherwise.
- This is typically used by tool providers to provide graphical representations of beans.
isStatelessSession

- Returns true if the bean is a stateless session bean and false otherwise.
- This is typically used by tool providers to provide graphical representations of beans.
Purpose

- A *handle* is a unique identifier that represents either an **EJBObject** or an **EJBHome** reference.

- These can be used by a client to re-establish connectivity to specific bean instances after a client shutdown.
Types

- There are two (2) kinds of handles:
  - Handle
  - HomeHandle
- Each type provides its own specific functionality.
public interface HomeHandle extends Serializable {
    public EJBHome getEJBHome()
    throws RemoteException;
}

public interface Handle
   extends Serializable {
   public EJBOBJECT getEJBOBJECT()
      throws RemoteException;
}

Properties

- All handles are serializable.
- The concept is portable, but the implementation is vendor-specific.
- Only required since EJB 1.1.
Pitfalls
The container is responsible for managing the concurrency of an EJB. It will throw a `RemoteException` to the client if:

- Two threads attempt to invoke methods concurrently using the same `EJBObject`.
- This means that `loopbacks` aren't legal.
Summary

- Discuss the home and remote interfaces.
- Discuss the relationships between the interfaces and the bean class.
- Discuss issues with bean's in general.
Next Steps

- Read chapter 7 in your text.
- The assignment for layer:04 is available on the course website.