Enterprise JavaBeans

Layer:03

Session
Agenda

- Build stateless & stateful session beans.
- Describe the bean's lifecycle.
- Describe the server's swapping mechanism.
I'll use the same example that I used for the socket and RMI sample.

This time we'll modify the Book class to be an EJB instead of an RMI object.
Implementation

- The EJB solution requires the following files:
  - `Book.java`
  - `BookHome.java`
  - `BookBean.java`
  - `ejb-jar.xml`
Bean Class
Purpose

- The *bean class* provides all of the business logic required by the bean's clients.
- Clients of the EJB never directly communicate with objects of this class.
- The EJB server and container maintain the instances of this class.
Structure

- The bean class may extend any java class.
- It *must* implement the `SessionBean` interface.
- It *may* implement the `SessionSynchronization` interface.
- Each session bean must provide one or more `ejbCreate()` methods.
Bean Instance Lifecycle

![Diagram showing the lifecycle of an EJB instance]

- **Does Not Exist**
  - timeout
  - newInstance()
  - setSessionContext()
  - ejbCreate()
- **Method Ready**
  - ejbPassivate()
- **Passive**
  - ejbActivate()
  - business methods
  - remove(), ejbRemove(), timeout
The `ejbCreate()` method acts as the bean's constructor.

Where a constructor is only executed when the bean is first created, the `ejbCreate()` method could be invoked multiple times for a single bean instance.
A session bean's `ejbCreate()` method must meet the following rules:

- It must have a return type of `void`.
- It can be overloaded unless the bean is a stateless session bean in which case only the no-arg `ejbCreate()` is permissible.
- It can throw any kind of exceptions.
The following code sample takes the Book interface, and converts it to be compatible with the EJB framework.

Initially, we'll just look at the `ejbCreate()` methods. Later we'll add other business methods.
1. package se554.layer03.ejb;

2. import javax.ejb.);

3. public class BookBean
4. implements SessionBean {
5. private SessionContext context = null;
6. private String title = null;
7. private String ISBN = null;

8. // include business methods
9. public void ejbCreate(String title,
   String ISBN)
10. throws CreateException {
11.     this.title = title;
13. } // end ejbCreate()

14. // include container-callback methods

15. } // end BookBean
The `SessionBean` interface declares a set of container-callback methods.

These methods must be implemented by the bean class and are used by the container to manage the bean instance.

The complete interface is on the following slide.
SessionBean Interface
(2 of 2)

1. public interface SessionBean
   extends EnterpriseBean {
   2.   public void ejbActivate();
   3.   public void ejbPassivate();
   4.   public void ejbRemove();
   5.   public void
        setSessionContext(SessionContext sc);
   6.   }

ejbActivate
(1 of 2)

- The **ejbActivate()** method is called by the container whenever a bean is restored from a passivated state.
- This is part of the server swapping mechanism.
- This method should "undo" any work performed by the **ejbPassivate()** method.
public void ejbActivate() {
    // TODO: perform any required resource allocations to ensure that the bean is capable of handling business requests.
}

The `ejbPassivate()` method is called by the container whenever a bean is passivated.

This is part of the server swapping mechanism.

This method should set all non-passivated fields to `null`.
ejbPassivate
(2 of 2)

1. public void ejbPassivate() {
2.     // TODO: set all non-persistable fields
3.     // to null.
4. }

The `ejbRemove()` method is called by the container when:

- A client calls the `remove()` method on their remote reference or on the home interface.
- The container times the bean out.

The `ejbRemove()` method should perform any last-minute deallocations of memory and other resources.
public void ejbRemove() {
  // TODO: de-allocate scarce resources and
  // set object references to null to
  // help the garbage collector.
}
The `setSessionFactory()` method is called by the container when the bean is first created.

- Be sure to capture a reference to the `SessionContext` argument for later use.
- Use this method for one-time initialization with the benefit of the `SessionContext`.
public void setSessionContext(SessionContext sc) {
    this.context = sc;
    // TODO: any other one-time actions
}
EJBContext
The **EJBContext** interface declares a set of methods that allow the bean to integrate with the container.

- It's the parent of the **SessionContext** interface provided to each session bean in its `setSessionContext()` method.
- The class that implements this interface is provided by the EJB server provider.
Structure

1. public interface EJBContext {
2.   public Principal getCallerPrincipal();
3.   public EJBHome getEJBHome();
4.   public boolean getRollbackOnly();
5.   public UserTransaction getUserTransaction()
       getTransaction();
6.   public boolean isCallerInRole();
7.   public void setRollbackOnly();
8. }
getCallerPrincipal

- Returns the minimally authenticated user that invoked the current method on the bean.
- This method isn't used to enforce security, but to customize processing.
- We'll cover EJB security later in the course (time permitting).
getEJBHome

- Returns the home interface that was used to create the bean instance.
- This method isn't commonly used.
- We'll see the home interface and corresponding home object later in this lecture.
getRollbackOnly

- Returns **true** if the current transaction has been marked for rollback and **false** otherwise.
- This method can only be called by an EJB using container-managed transactions.
- We'll discuss EJB transactions later in the course.
**getUserTransaction**

- Returns a reference to the EJB server's *transaction service*.
- This method can only be used by EJBs using bean-managed transactions.
- We'll discuss EJB transactions in a later lecture.
isCallerInRole

- Returns **true** if the invoker of the method belongs to the specified EJB role and **false** otherwise.

- We'll cover EJB security later in the course (time permitting).
setRollbackOnly

- Marks the current transaction so that the server issues a *rollback* instead of a *commit*.
- This method can only be called by an EJB using container-managed transactions.
- We'll discuss EJB transactions later on in the course.
SessionContext
Purpose

- The `SessionContext` interface inherits from the `EJBCo<text width="100%">ntext`.  
- It provides an additional method that is critical for proper use of EJBs as arguments to methods.
Structure

1. public interface SessionContext extends EJBContext {
2.     public EJBOBJECT getEJBOBJECT();
3. }

Business Methods
Purpose

- The bean's business methods are used to provide the actual business functionality that will be requested by clients of the bean.
Structure

- Business methods are implemented within the bean class.
- Their signatures must be declared in the remote interface.
- They can accept any RMI-IIOP compliant arguments and throw any exceptions.
1. package se554.layer03.ejb;

2. import javax.ejb.*;

3. public class BookBean
   implements SessionBean {

4.   private SessionContext context = null;
5.   private String title = null;
6.   private String ISBN = null;

7.   public String getTitle() {
8.     return this.title;
9.   } // end getTitle()
10. public String getISBN() {
11.     return this.ISBN;
12. } // end getISBN()

13. public void 
14.     ejbCreate(String title,String ISBN) 
15.     throws CreateException {
16.         this.title = title;
17.         this.ISBN = ISBN;
18.     } // end ejbCreate()

19. public void ejbActivate() {
20. } // end ejbActivate()
19.  public void ejbPassivate() {  
20.   }  // end ejbPassivate()

21.  public void ejbRemove() {  
22.   this.title = null;
23.   this.ISBN = null;
24.   this.context = null;
25.   }  // end ejbRemove()
26. public void
    setSessionContext(SessionContext sc) {
27.        this.context = sc;
28.    } // end setSessionContext()
29.} // end BookBean
Scalability
If a stateful session bean has remained idle for a period of time, the container may decide to *passivate* the bean to disk.

Once the bean's state has been saved, the container can loan that bean instance out to another client.

This is called *swapping*. 
Swapping
(2 of 2)

- The `ejbPassivate()` method is called before the bean's state is saved.
- The `ejbActivate()` method is called once the bean's data has been restored from the passivated state.
- Swapping is transparent to the client.
Pooling (1 of 2)

- Good containers may allocate several EJB instances for stateless beans.
- These instances are placed into a pool.
- When a request involving a stateless EJB is received, an arbitrary instance is retrieved from the pool to satisfy the request and is returned to the pool afterward.
Pooling allows a much higher degree of scalability since only a relatively few number of bean instances can be used to satisfy a large number of requests.
Exceptions
Categories

- Bean methods can throw exceptions just as any other class method.
- Exceptions are categorized as:
  - System
  - Application
System Exceptions

- Any `RuntimeException` or `EJBException` exception is considered a system exception.

- System exceptions are returned to the client as a `RemoteException`.

- The container destroys the bean instance that threw the exception.
Application Exceptions

- Any exception not inherited from `RuntimeException` or `EJBException` is considered an application exception.
- Application exceptions are returned to the client "as is."
- The container takes no further action.
ejbCreate Revisited

1. public void 
ejbCreate(String title, String ISBN) 
   throws CreateException {
2.     if ( title == null ) {
3.         throw new CreateException();
4.     } 
5.     if ( ISBN == null ) {
6.         throw new CreateException();
7.     } 
8.     this.title = title;
10. } // end ejbCreate()
Summary

- Build stateless & stateful session beans.
- Describe the bean's lifecycle.
- Describe the server's swapping mechanism.
Next Steps

- Please read chapter 4, pp.111-125 and chapter 7.
- Layer:03 homework is available on the course website.