EJB - CALLBACKS

Callback is a mechanism by which life cycle of an enterprise bean can be intercepted. EJB 3.0 specification has specified callbacks for which callback handler methods are to be created. EJB Container calls these callbacks. We can define callback methods in the ejb class itself or in a separate class. EJB 3.0 has provided many annotations for callbacks

Following is the list of callback annotations for stateless bean.

Annotation	Description
@PostConstruct	method is invoked when a bean is created for the first time
@PreDestroy	method is invoked when a bean is removed from the bean pool or is destroyed.

Following is the list of callback annotations for stateful bean.

Annotation	Description
@PostConstruct	method is invoked when a bean is created for the first time
@PreDestroy	method is invoked when a bean is removed from the bean pool or is destroyed.
@PostActivate	method is invoked when a bean is loaded to be used.
@PrePassivate	method is invoked when a bean is put back to bean pool.

Following is the list of callback annotations for message driven bean.

Annotation	Description
@PostConstruct	method is invoked when a bean is created for the first time
@PreDestroy	method is invoked when a bean is removed from the bean pool or is destroyed.

Following is the list of callback annotations for entity bean.

Annotation	Description
@PrePersist	method is invoked when an entity is created in database.
@PostPersist	method is invoked after an entity is created in database.
@PreRemove	method is invoked when an entity is deleted from the database.
@PostRemove	method is invoked after an entity is deleted from the database.
@PreUpdate	method is invoked before an entity is to be updated in the database.
@PostLoad	method is invoked when a record is fetched from database and loaded into the entity.

Example Application

Let us create a test EJB application to test various callbacks in EJB.

Step	Description
1	Create a project with a name *EjbComponent* under a package *com.tutorialspoint.stateless* as explained in the *EJB - Create Application* chapter. You can also use the project created in *EJB - Persistence* chapter as such for this chapter to add various callbacks to ejbs.
2	Create *LibrarySessionBean.java* and *LibrarySessionBeanRemote* as explained in the *EJB - Create Application* chapter. Keep rest of the files unchanged.
3	Use Beans created in the *EJB - Persistence* chapter. Add callback methods as shown below. Keep rest of the files unchanged.
4	Create a java class *BookCallbackListener* under package *com.tutorialspoint.callback*. This class will demonstrates the seperation of callback methods.
5	Clean and Build the application to make sure business logic is working as per the requirements.
6	Finally, deploy the application in the form of jar file on JBoss Application Server. JBoss Application server will get started automatically if it is not started yet.
7	Now create the ejb client, a console based application in the same way as explained in the *EJB - Create Application* chapter under topic **Create Client to access EJB**.

EJBComponent *EJBModule*

BookCallbackListener.java

```
package com.tutorialspoint.callback;

import javax.persistence.PrePersist;
import javax.persistence.PostLoad;
import javax.persistence.PostPersist;
import javax.persistence.PostRemove;
import javax.persistence.PostUpdate;
import javax.persistence.PreRemove;
import javax.persistence.PreUpdate;
import com.tutorialspoint.entity.Book;

public class BookCallbackListener {

   @PrePersist
   public void prePersist(Book book){
      System.out.println("BookCallbackListener.prePersist:"
         + "Book to be created with book id: "+book.getId());
   }

   @PostPersist
   public void postPersist(Object book){
      System.out.println("BookCallbackListener.postPersist::"
         + "Book created with book id: "+((Book)book).getId());
   }

   @PreRemove
   public void preRemove(Book book)
   {
      System.out.println("BookCallbackListener.preRemove:"
         + " About to delete Book: " + book.getId());
   }
```

```
   @PostRemove
   public void postRemove(Book book)
   {
      System.out.println("BookCallbackListener.postRemove::"
         + " Deleted Book: " + book.getId());
   }

   @PreUpdate
   public void preUpdate(Book book)
   {
      System.out.println("BookCallbackListener.preUpdate::"
         + " About to update Book: " + book.getId());
   }

   @PostUpdate
   public void postUpdate(Book book)
   {
      System.out.println("BookCallbackListener.postUpdate::"
         + " Updated Book: " + book.getId());
   }

   @PostLoad
   public void postLoad(Book book)
   {
      System.out.println("BookCallbackListener.postLoad::"
         + " Loaded Book: " + book.getId());
   }
}
```

Book.java

```
package com.tutorialspoint.entity;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EntityListeners;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="books")
public class Book implements Serializable{

   private int id;
   private String name;

   public Book(){
   }

   @Id
   @GeneratedValue(strategy= GenerationType.IDENTITY)
   @Column(name="id")
   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
```

```
        this.name = name;
    }
}
```

LibraryStatefulSessionBean.java

```
package com.tutorialspoint.stateful;

import java.util.ArrayList;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.PostActivate;
import javax.ejb.PrePassivate;
import javax.ejb.Stateful;

@Stateful
public class LibraryStatefulSessionBean
   implements LibraryStatefulSessionBeanRemote {
   List<String> bookShelf;

   public LibraryStatefulSessionBean(){
      bookShelf = new ArrayList<String>();
   }

   public void addBook(String bookName) {
      bookShelf.add(bookName);
   }

   public List<String> getBooks() {
      return bookShelf;
   }

   @PostConstruct
   public void postConstruct(){
      System.out.println("LibraryStatefulSessionBean.postConstruct::"
         + " bean created.");
   }

   @PreDestroy
   public void preDestroy(){
      System.out.println("LibraryStatefulSessionBean.preDestroy:"
         + " bean removed.");
   }

   @PostActivate
   public void postActivate(){
      System.out.println("LibraryStatefulSessionBean.postActivate:"
         + " bean activated.");
   }

   @PrePassivate
   public void prePassivate(){
      System.out.println("LibraryStatefulSessionBean.prePassivate:"
         + " bean passivated.");
   }
}
```

LibraryStatefulSessionBeanRempote.java

```
package com.tutorialspoint.stateful;

import java.util.List;
import javax.ejb.Remote;

@Remote
public interface LibraryStatefulSessionBeanRemote {
   void addBook(String bookName);
```

```
   List getBooks();
}
```

LibraryPersistentBean.java

```
package com.tutorialspoint.stateless;

import com.tutorialspoint.entity.Book;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

@Stateless
public class LibraryPersistentBean
   implements LibraryPersistentBeanRemote {

   public LibraryPersistentBean(){}

   @PersistenceContext(unitName="EntityEjbPU")
   private EntityManager entityManager;

   public void addBook(Book book) {
      entityManager.persist(book);
   }

   public List<Book> getBooks() {
      return entityManager.createQuery("From Book")
         .getResultList();
   }

   @PostConstruct
   public void postConstruct(){
      System.out.println("postConstruct:: LibraryPersistentBean session bean"
         + " created with entity Manager object: ");
   }

   @PreDestroy
   public void preDestroy(){
      System.out.println("preDestroy: LibraryPersistentBean session"
      + " bean is removed ");
   }
}
```

LibraryPersistentBeanRemote.java

```
package com.tutorialspoint.stateless;

import com.tutorialspoint.entity.Book;
import java.util.List;
import javax.ejb.Remote;

@Remote
public interface LibraryPersistentBeanRemote {

   void addBook(Book bookName);

   List<Book> getBooks();

}
```

- As soon as you deploy the EjbComponent project on JBOSS, notice the jboss log.
- JBoss has automatically created a JNDI entry for our session bean - **LibraryPersistentBean/remote**.

- We'll using this lookup string to get remote business object of type - **com.tutorialspoint.stateless.LibraryPersistentBeanRemote**

JBoss Application server log output

```
...
16:30:01,401 INFO  [JndiSessionRegistrarBase] Binding the following Entries in Global
JNDI:
   LibraryPersistentBean/remote - EJB3.x Default Remote Business Interface
   LibraryPersistentBean/remote-com.tutorialspoint.stateless.LibraryPersistentBeanRemote
- EJB3.x Remote Business Interface
16:30:02,723 INFO  [SessionSpecContainer] Starting
jboss.j2ee:jar=EjbComponent.jar,name=LibraryPersistentBean,service=EJB3
16:30:02,723 INFO  [EJBContainer] STARTED EJB:
com.tutorialspoint.stateless.LibrarySessionBeanRemote ejbName: LibraryPersistentBean
...
```

EJBTester *EJBClient*

jndi.properties

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=localhost
```

- These properties are used to initialize the InitialContext object of java naming service
- InitialContext object will be used to lookup stateless session bean

EJBTester.java

```
package com.tutorialspoint.test;

import com.tutorialspoint.stateful.LibrarySessionBeanRemote;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;
import java.util.Properties;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class EJBTester {

   BufferedReader brConsoleReader = null;
   Properties props;
   InitialContext ctx;
   {
      props = new Properties();
      try {
         props.load(new FileInputStream("jndi.properties"));
      } catch (IOException ex) {
         ex.printStackTrace();
      }
      try {
         ctx = new InitialContext(props);
      } catch (NamingException ex) {
         ex.printStackTrace();
      }
      brConsoleReader =
      new BufferedReader(new InputStreamReader(System.in));
   }

   public static void main(String[] args) {
```

```
      EJBTester ejbTester = new EJBTester();

      ejbTester.testEntityEjb();
   }

   private void showGUI(){
      System.out.println("**********************");
      System.out.println("Welcome to Book Store");
      System.out.println("**********************");
      System.out.print("Options \n1. Add Book\n2. Exit \nEnter Choice: ");
   }

   private void testEntityEjb(){
      try {
      int choice = 1;

      LibraryPersistentBeanRemote libraryBean =
      (LibraryPersistentBeanRemote)
      ctx.lookup("LibraryPersistentBean/remote");

      while (choice != 2) {
         String bookName;
         showGUI();
         String strChoice = brConsoleReader.readLine();
         choice = Integer.parseInt(strChoice);
         if (choice == 1) {
            System.out.print("Enter book name: ");
            bookName = brConsoleReader.readLine();
            Book book = new Book();
            book.setName(bookName);
            libraryBean.addBook(book);
         } else if (choice == 2) {
            break;
         }
      }

      List<Book> booksList = libraryBean.getBooks();

      System.out.println("Book(s) entered so far: " + booksList.size());
      int i = 0;
      for (Book book:booksList) {
         System.out.println((i+1)+". " + book.getName());
         i++;
      }

      } catch (Exception e) {
         System.out.println(e.getMessage());
         e.printStackTrace();
      }finally {
         try {
            if(brConsoleReader !=null){
               brConsoleReader.close();
            }
         } catch (IOException ex) {
            System.out.println(ex.getMessage());
         }
      }
   }
}
```

EJBTester is doing the following tasks.

- Load properties from jndi.properties and initialize the InitialContext object.
- In testStatelessEjb method, jndi lookup is done with name - "LibrarySessionBean/remote" to obtain the remote business object *statelessejb*.
- Then user is shown a library store User Interface and he/she is asked to enter choice.

- If user enters 1, system asks for book name and saves the book using stateless session bean addBook method. Session Bean is storing the book in the database.
- If user enters 2, system retrieves books using stateless session bean getBooks method and exits.

Run Client to access EJB

Locate EJBTester.java in project explorer. Right click on EJBTester class and select **run file**.

Verify the following output in Netbeans console.

```
run:
**********************
Welcome to Book Store
**********************
Options
1. Add Book
2. Exit
Enter Choice: 1
Enter book name: Learn Java
**********************
Welcome to Book Store
**********************
Options
1. Add Book
2. Exit
Enter Choice: 2
Book(s) entered so far: 1
1. Learn Java
BUILD SUCCESSFUL (total time: 13 seconds)
```

JBoss Application server log output

You can find the following callback entries in JBoss log

```
14:08:34,293 INFO  [STDOUT] postConstruct:: LibraryPersistentBean session bean created
with entity Manager object
...
16:39:09,484 INFO  [STDOUT] BookCallbackListener.prePersist:: Book to be created with
book id: 0
16:39:09,531 INFO  [STDOUT] BookCallbackListener.postPersist:: Book created with book id:
1
16:39:09,900 INFO  [STDOUT] BookCallbackListener.postLoad:: Loaded Book: 1
...
```