EJB - PERSISTENCE

Previous Page
Next Page

EJB 3.0, entity bean used in EJB 2.0 is largely replaced by persistence mechanism. Now entity bean is a simple POJO having mapping with table.

Following are the key actors in persistence API

- **Entity** - A persistent object representing the data-store record. It is good to be serializable.
- **EntityManager** - Persistence interface to do data operations like add/delete/update/find on persistent object*entity*. It also helps to execute queries using **Query** interface.
-
- **Persistence unit** ***persistence. xml*** - Persistence unit describes the properties of persistence mechanism.
-
- **Data Source** ***∗ ds. xml*** - Data Source describes the data-store related properties like connection url. user-name,password etc.

To demonstrate ejb persistence mechanism, we're going to do the following tasks.

- Step 1. Create table in database.
- Step 2. Create Entity class corresponding to table.
- Step 3. Create Data Source and Persistence Unit
- Step 4. Create a stateless ejb having EntityManager instance.
- Step 5. Update stateless ejb. Add methods to add records and get records from database via entity manager.
- Step 6. A console based application client will access the stateless ejb to persist data in database.

Create table

Create a table **books** in default database **postgres**.

```
CREATE TABLE books (
   id     integer PRIMARY KEY,
   name   varchar(50)
);
```

Create Entity class

```
//mark it entity using Entity annotation
//map table name using Table annoation
@Entity
@Table(name="books")
public class Book implements Serializable{

   private int id;
   private String name;

   public Book(){
   }

   //mark id as primary key with autogenerated value
```

```
   //map database column id with id field
   @Id
   @GeneratedValue(strategy= GenerationType.IDENTITY)
   @Column(name="id")
   public int getId() {
      return id;
   }
   ...
}
```

Create DataSource and persistence unit

DataSource *jboss – ds. xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
   <local-tx-datasource>
      <jndi-name>PostgresDS</jndi-name>
      <connection-url>jdbc:postgresql://localhost:5432/postgres</connection-url>
      <driver-class>org.postgresql.driver</driver-class>
      <user-name>sa</user-name>
      <password>sa</password>
      <min-pool-size>5</min-pool-size>
      <max-pool-size>20</max-pool-size>
      <idle-timeout-minutes>5</idle-timeout-minutes>
   </local-tx-datasource>
</datasources>
```

Persistence Unit *persistence. xml*

```
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
   <persistence-unit name="EjbComponentPU" transaction-type="JTA">
      <jta-data-source>java:/PostgresDS</jta-data-source>
      <exclude-unlisted-classes>false</exclude-unlisted-classes>
      <properties/>
   </persistence-unit>
   <persistence-unit name="EjbComponentPU2" transaction-type="JTA">
      <provider>org.hibernate.ejb.HibernatePersistence</provider>
      <jta-data-source>java:/PostgresDS</jta-data-source>
      <exclude-unlisted-classes>false</exclude-unlisted-classes>
      <properties>
         <property name="hibernate.hbm2ddl.auto" value="update"/>
      </properties>
   </persistence-unit>
</persistence>
```

Create Stateless EJB having EntityManager instance

```
@Stateless
public class LibraryPersistentBean implements LibraryPersistentBeanRemote {

   //pass persistence unit to entityManager.
   @PersistenceContext(unitName="EjbComponentPU")
   private EntityManager entityManager;

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

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

After building the ejb module, we need a client to access the stateless bean which we'll be going to create in next section.

Example Application

Let us create a test EJB application to test EJB persistence mechanism.

Step	Description
1	Create a project with a name *EjbComponent* under a package *com.tutorialspoint.entity* as explained in the *EJB - Create Application* chapter. You can also use the project created in *EJB - Create Application* chapter as such for this chapter to understand ejb persistence concepts.
2	Create *Book.java* under package *com.tutorialspoint.entity* and modify it as shown below.
3	Create *LibraryPersistentBean.java* and *LibraryPersistentBeanRemote* as explained in the *EJB - Create Application* chapter and modify them as shown below.
4	Create *jboss-ds.xml* in **EjbComponent > setup** folder and *persistence.xml* in **EjbComponent > src > conf** folder. These folder can be seen in files tab in Netbeans. Modify these files as shown above.
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**. Modify it as shown below.

EJBComponent *EJBModule*

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;
   }
}
```

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();

}
```

LibraryPersistentBean.java

```
package com.tutorialspoint.stateless;

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

@Stateless
public class LibraryPersistentBean implements LibraryPersistentBeanRemote {

   public LibraryPersistentBean(){
   }

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

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

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

- 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=LibraryPersistentBeanRemote,service=EJB3
16:30:02,723 INFO  [EJBContainer] STARTED EJB:
com.tutorialspoint.stateless.LibraryPersistentBeanRemote ejbName: LibraryPersistentBean
16:30:02,731 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
...
```

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.stateless.LibraryPersistentBeanRemote;
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 testStatefulEjb method, jndi lookup is done with name - "LibraryStatefulSessionBean/remote" to obtain the remote business object *statefulejb*.
- 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 persisting the book in database via EntityManager call.
- If user enters 2, system retrieves books using stateful session bean getBooks method and exits.
- Then another jndi lookup is done with name - "LibraryStatelessSessionBean/remote" to obtain the remote business object *statelessejb* again and listing of books is done.

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: 15 seconds)
```

Run Client again to access EJB.

Restart the JBoss before accessing the 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 Spring
**********************
Welcome to Book Store
**********************
Options
1. Add Book
2. Exit
Enter Choice: 2
Book(s) entered so far: 2
1. learn java
2. Learn Spring
BUILD SUCCESSFUL (total time: 15 seconds)
```

- Output shown above states that books are getting stored in persistent storage and are retrieved from database.