EJB - ACCESS DATABASE

EJB 3.0, persistence mechanism is used to access the database in which container manages the database related operations. Developers can access database using jdbc api call directly in ejb business methods.

To demonstrate database access in ejb, we're going to do the following tasks.

- Step 1. Create table in database.
- Step 2. Create a stateless ejb having business me.
- Step 3. Update stateless ejb. Add methods to add records and get records from database via entity manager.
- Step 4. 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 a model class

```
public class Book implements Serializable{

   private int id;
   private String name;

   public Book(){
   }

   public int getId() {
      return id;
   }
   ...
}
```

Create Stateless EJB

```
@Stateless
public class LibraryPersistentBean implements LibraryPersistentBeanRemote {

   public void addBook(Book book) {
     //persist book using jdbc calls
   }

   public List<Book> getBooks() {
     //get books using jdbc calls
   }
   ...
}
```

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 database access 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 data access 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	Clean and Build the application to make sure business logic is working as per the requirements.
5	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.
6	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;

public class Book implements Serializable{

   private int id;
   private String name;

   public Book(){
   }

   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.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;

@Stateless
public class LibraryPersistentBean implements LibraryPersistentBeanRemote {

   public LibraryPersistentBean(){
   }

   public void addBook(Book book) {
      Connection con = null;
      String url = "jdbc:postgresql://localhost:5432/postgres";
      String driver = "org.postgresql.driver";

      String userName = "sa";
      String password = "sa";
      List<Book> books = new ArrayList<Book>();
      try {

         Class.forName(driver).newInstance();
         con = DriverManager.getConnection(url , userName, password);

         PreparedStatement st =
         con.prepareStatement("insert into book(name) values(?)");
         st.setString(1,book.getName());

         int result = st.executeUpdate();

      } catch (SQLException ex) {
         ex.printStackTrace();
      } catch (InstantiationException ex) {
         ex.printStackTrace();
      } catch (IllegalAccessException ex) {
         ex.printStackTrace();
      } catch (ClassNotFoundException ex) {
         ex.printStackTrace();
      }
   }

   public List<Book> getBooks() {
      Connection con = null;
      String url = "jdbc:postgresql://localhost:5432/postgres";
      String driver = "org.postgresql.driver";

      String userName = "sa";
      String password = "sa";
      List<Book> books = new ArrayList<Book>();
```

```
      try {

         Class.forName(driver).newInstance();
         con = DriverManager.getConnection(url , userName, password);

         Statement st = con.createStatement();
         ResultSet rs = st.executeQuery("select * from book");

         Book book;
         while (rs.next()) {
            book = new Book();
            book.setId(rs.getInt(1));
            book.setName(rs.getString(2));
            books.add(book);
         }
      } catch (SQLException ex) {
         ex.printStackTrace();
      } catch (InstantiationException ex) {
         ex.printStackTrace();
      } catch (IllegalAccessException ex) {
         ex.printStackTrace();
      } catch (ClassNotFoundException ex) {
         ex.printStackTrace();
      }
      return books;
   }
}
```

- 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 - "LibraryStatelessSessionBean/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 retrives books using stateless session bean getBooks method and exits.
- Then another jndi lookup is done with name - "LibraryStatelessSessionBean/remote" to obtain the remote business object *statefulejb* 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)
```