EJB - STATEFUL BEAN

Previous Page
Next Page

A stateful session bean is a type of enterprise bean which preserve the conversational state with client. A stateful session bean as per its name keeps associated client state in its instance variables. EJB Container creates a separate stateful session bean to process client's each request. As soon as request scope is over, statelful session bean is destroyed.

Following are the steps required to create a stateful ejb.

- Create a remote/local interface exposing the business methods.
- This interface will be used by the ejb client application.
-
- Use @Local annotation if ejb client is in same environment where ejb session bean is to be deployed.
-
- Use @Remote annotation if ejb client is in different environment where ejb session bean is to be deployed.
-
- Create a stateful session bean implementing the above interface.
- Use @Stateful annotation to signify it a stateful bean. EJB Container automatically creates the relevant configurations or interfaces required by reading this annotation during deployment.

Remote Interface

```
import javax.ejb.Remote;

@Remote
public interface LibraryStatefulSessionBeanRemote {
   //add business method declarations
}
```

Stateful EJB

```
@Stateful
public class LibraryStatefulSessionBean implements LibraryStatefulSessionBeanRemote {
   //implement business method
}
```

Example Application

Let us create a test EJB application to test stateful EJB.

Step	Description
1	Create a project with a name *EjbComponent* under a package *com.tutorialspoint.stateful* 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 stateful ejb concepts.
2	Create *LibraryStatefulSessionBean.java* and *LibraryStatefulSessionBeanRemote* as explained in the *EJB - Create Application* chapter. Keep rest of the files unchanged.

3 Clean and Build the application to make sure business logic is working as per the requirements.

4 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.

5 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*

LibraryStatefulSessionBeanRemote.java

```
package com.tutorialspoint.stateful;

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

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

LibraryStatefulSessionBean.java

```

package com.tutorialspoint.stateful;

import java.util.ArrayList;
import java.util.List;
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;
   }
}
```

- 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 - **LibraryStatefulSessionBean/remote**.
- We'll using this lookup string to get remote business object of type - **com.tutorialspoint.stateful.LibraryStatefulSessionBeanRemote**

JBoss Application server log output

```
...
16:30:01,401 INFO  [JndiSessionRegistrarBase] Binding the following Entries in Global
JNDI:
   LibraryStatefulSessionBean/remote - EJB3.x Default Remote Business Interface
```

```
LibraryStatefulSessionBean/remote-com.tutorialspoint.stateful.LibraryStatefulSessionBeanRe
mote - EJB3.x Remote Business Interface
16:30:02,723 INFO  [SessionSpecContainer] Starting
jboss.j2ee:jar=EjbComponent.jar,name=LibraryStatefulSessionBean,service=EJB3
16:30:02,723 INFO  [EJBContainer] STARTED EJB:
com.tutorialspoint.stateful.LibraryStatefulSessionBeanRemote ejbName:
LibraryStatefulSessionBean
16:30:02,731 INFO  [JndiSessionRegistrarBase] Binding the following Entries in Global
JNDI:

   LibraryStatefulSessionBean/remote - EJB3.x Default Remote Business Interface

LibraryStatefulSessionBean/remote-com.tutorialspoint.stateful.LibraryStatefulSessionBeanRe
mote - 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 stateful session bean

EJBTester.java

```
package com.tutorialspoint.test;

import com.tutorialspoint.stateful.LibraryStatefulSessionBeanRemote;
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.testStatelessEjb();
   }

   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 testStatelessEjb(){

      try {
         int choice = 1;

         LibraryStatefulSessionBeanRemote libraryBean =

LibraryStatefulSessionBeanRemote)ctx.lookup("LibraryStatefulSessionBean/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++;
         }
         LibraryStatefulSessionBeanRemote libraryBean1 =

(LibraryStatefulSessionBeanRemote)ctx.lookup("LibraryStatefulSessionBean/remote");
         List<String> booksList1 = libraryBean1.getBooks();
         System.out.println(
            "***Using second lookup to get library stateful object***");
         System.out.println(
            "Book(s) entered so far: " + booksList1.size());
         for (int i = 0; i < booksList1.size(); ++i) {
            System.out.println((i+1)+". " + booksList1.get(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 stateful session bean addBook method. Session Bean is storing the book in its instance variable.
- If user enters 2, system retrieves books using stateful session bean getBooks method and exits.
- Then another jndi lookup is done with name - "LibraryStatefulSessionBean/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
***Using second lookup to get library stateful object***
Book(s) entered so far: 0
BUILD SUCCESSFUL (total time: 13 seconds)
```

Run Client again 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: 2
Book(s) entered so far: 0
***Using second lookup to get library stateful object***
Book(s) entered so far: 0
BUILD SUCCESSFUL (total time: 12 seconds)
```

- Output shown above states that for each lookup a different stateful ejb instance is returned.

- Stateful ejb object is keeping value for single session only. As in second run, we're not getting any value of books