A stateless session bean is a type of enterprise bean which is normally used to do independent operations. A stateless session bean as per its name does not have any associated client state, but it may preserve its instance state. EJB Container normally creates a pool of few stateless bean’s objects and use these objects to process client's request. Because of pool, instance variable values are not guaranteed to be same across lookups/method calls.

Following are the steps required to create a stateless 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 stateless session bean implementing the above interface.
- Use @Stateless annotation to signify it a stateless bean. EJB Container automatically creates the relevant configurations or interfaces required by reading this annotation during deployment.

**Remote Interface**

```java
import javax.ejb.Remote;

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

**Stateless EJB**

```java
@Stateless
public class LibrarySessionBean implements LibrarySessionBeanRemote {
    //implement business method
}
```

**Example Application**

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a project with a name <em>EjbComponent</em> under a package <em>com.tutorialspoint.stateless</em> as explained in the <em>EJB - Create Application</em> chapter. You can also use the project created in <em>EJB - Create Application</em> chapter as such for this chapter to understand stateless ejb concepts.</td>
</tr>
<tr>
<td>2</td>
<td>Create <em>LibrarySessionBean.java</em> and <em>LibrarySessionBeanRemote</em> as explained in the <em>EJB - Create Application</em> chapter. Keep rest of the files unchanged.</td>
</tr>
</tbody>
</table>
Clean and Build the application to make sure business logic is working as per the requirements.

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.

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**

**LibrarySessionBeanRemote.java**

```java
package com.tutorialspoint.stateless;
import java.util.List;
import javax.ejb.Remote;

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

**LibrarySessionBean.java**

```java
package com.tutorialspoint.stateless;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;

@Stateless
public class LibrarySessionBean implements LibrarySessionBeanRemote {

    private List<String> bookShelf;

    public LibrarySessionBean(){
        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 - LibrarySessionBean/remote.
- We'll using this lookup string to get remote business object of type - com.tutorialspoint.stateless.LibrarySessionBeanRemote

**JBoss Application server log output**

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

**jndi.properties**

```java
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**

```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.testStatelessEjb();
    }
}
```
private void showGUI(){
    System.out.println("************************");
    System.out.println("Welcome to Book Store");
    System.out.println("************************");
    System.out.println("Options 
1. Add Book
2. Exit 
Enter Choice: ");
}

private void testStatelessEjb(){
    try {
        int choice = 1;
        LibrarySessionBeanRemote libraryBean =
            (LibrarySessionBeanRemote)ctx.lookup("LibrarySessionBean/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++;
        }
        LibrarySessionBeanRemote libraryBean1 =
            (LibrarySessionBeanRemote)ctx.lookup("LibrarySessionBean/remote");
        List<String> booksList1 = libraryBean1.getBooks();
        System.out.println("***Using second lookup to get library stateless 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 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 its instance variable.

If user enters 2, system retrieves books using stateless session bean getBooks method and exits.

Then another jndi lookup is done with name - "LibrarySessionBean/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
***Using second lookup to get library stateless 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 stateless object***
Book(s) entered so far: 1
1. Learn Java
BUILD SUCCESSFUL (total time: 12 seconds)
```

- Output shown above may vary depending upon how many stateless ejb object JBoss is maintaining.

- In case a single stateless ejb object is maintained, you may see the same list of books after each lookup.
- EJB Container may return same stateless ejb object for every lookup.
- Stateless ejb bean is keeping value of instance variable till the server is not restarted.