

## 1 (a). Write a program to Demonstrate Sessions Using Servlets

**Aim:** To Demonstrate a program on sessions using servlets.

### **Packages:**

java.io

java.util

javax.servlet

javax.servlet.http

**Note:** Using sessions in Servlets is quite straightforward, and involves looking up the session object associated with the current request, creating a new session object when necessary, looking up information associated with a session, storing information in a session, and discarding completed or abandoned sessions.

This is done by calling the getSession method of HttpServletRequest. If this returns null, you can create a new session, but this is so commonly done that there is an option to automatically create a new session if there isn't one already. Just pass true to getSession.

```
HttpSession session = request.getSession(true);
```

- **getId** :This method returns the unique identifier generated for each session. It is sometimes used as the key name when there is only a single value associated with a session, or when logging information about previous sessions.
- **isNew**: This returns true if the client (browser) has never seen the session, usually because it was just created rather than being referenced by an incoming client request. It returns false for preexisting sessions.
- **getCreationTime**: This returns the time, in milliseconds since the epoch, at which the session was made. To get a value useful for printing out, pass the value to the Date constructor or the setTimeInMillis method of GregorianCalendar.
- **getLastAccessedTime**: This returns the time, in milliseconds since the epoch, at which the session was last sent from the client.
- **getMaxInactiveInterval**: This returns the amount of time, in seconds, that a session should go without access before being automatically invalidated. A negative value indicates that the session should never timeout.

You read information associated with a session by using `getAttribute()`. To specify information, you use `setAttribute()` supplying a key and a value.

### **Session.java:**

```
import java.io.*;

import java.util.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class Session extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)

        throws IOException, ServletException

    {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        HttpSession session = request.getSession(true);

        Date created = new Date(session.getCreationTime());

        Date accessed = new Date(session.getLastAccessedTime());

        out.println("ID " + session.getId()+"<br>");

        out.println("Created: " + created+"<br>");

        out.println("Last Accessed: " + accessed+"<br>");

        String dataName = request.getParameter("dataname");

        if (dataName != null && dataName.length() > 0) {

            String dataValue = request.getParameter("datavalue");

            session.setAttribute(dataName, dataValue);

        }

        Enumeration e = session.getAttributeNames();

        while (e.hasMoreElements()) {

            String name = (String)e.nextElement();
```

```
String value = session.getAttribute(name).toString();  
out.println("<br>" + name + " = " + value);  
}  
}  
}
```

### **Session.html**

```
<html>  
<head>  
<title>Sessions Example</title>  
</head>  
<body>  
    <h1> An example for session attributes </h1>  
    <form action="Session" method=GET>  
Name of Session Attribute:  
    <input type=text size=20 name=dataname>  
    <br>  
Value of Session Attribute:  
    <input type=text size=20 name=datavalue>  
    <br>  
    <input type=submit>  
</form>  
</body>  
</html>
```

### **Steps to Execute:**

1. Place Session.class file in C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\classes
2. Place Session.html file in C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT
3. Change web.xml file in C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\

Ex: <servlet>

<servlet-name>Sessiondemo</servlet-name>

<servlet-class>Session</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Sessiondemo</servlet-name>

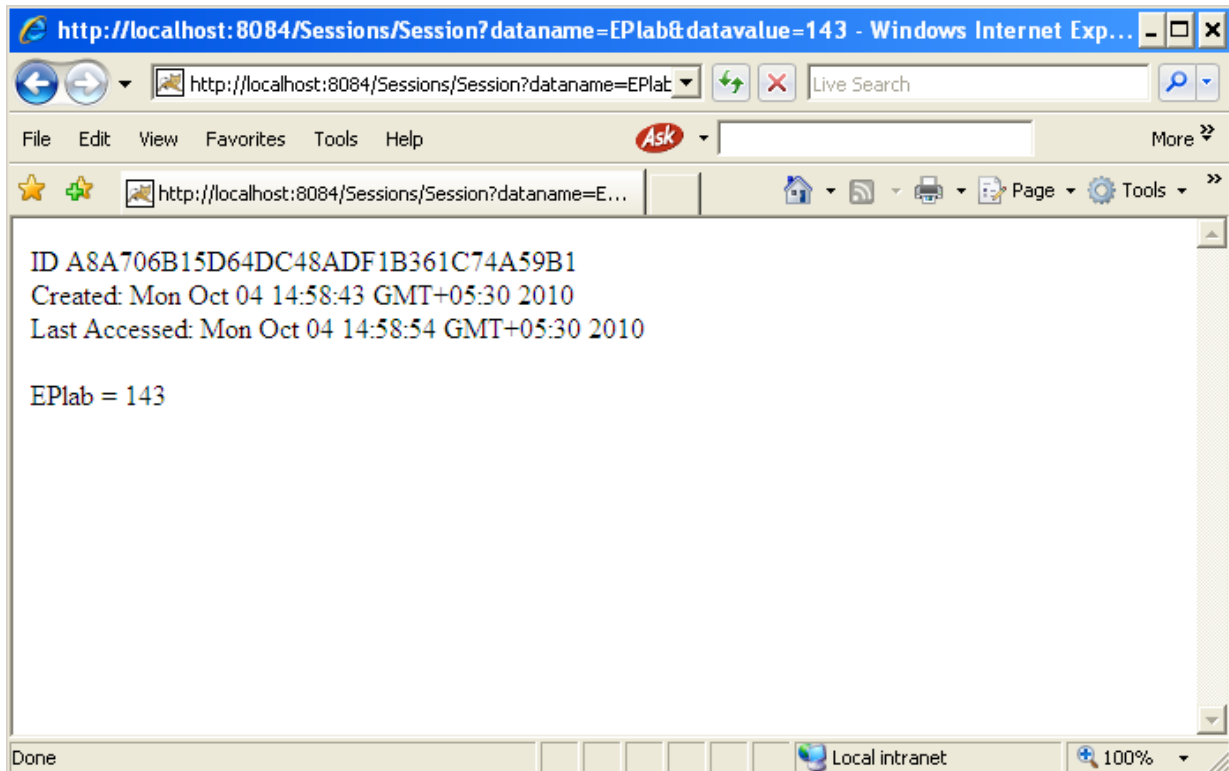
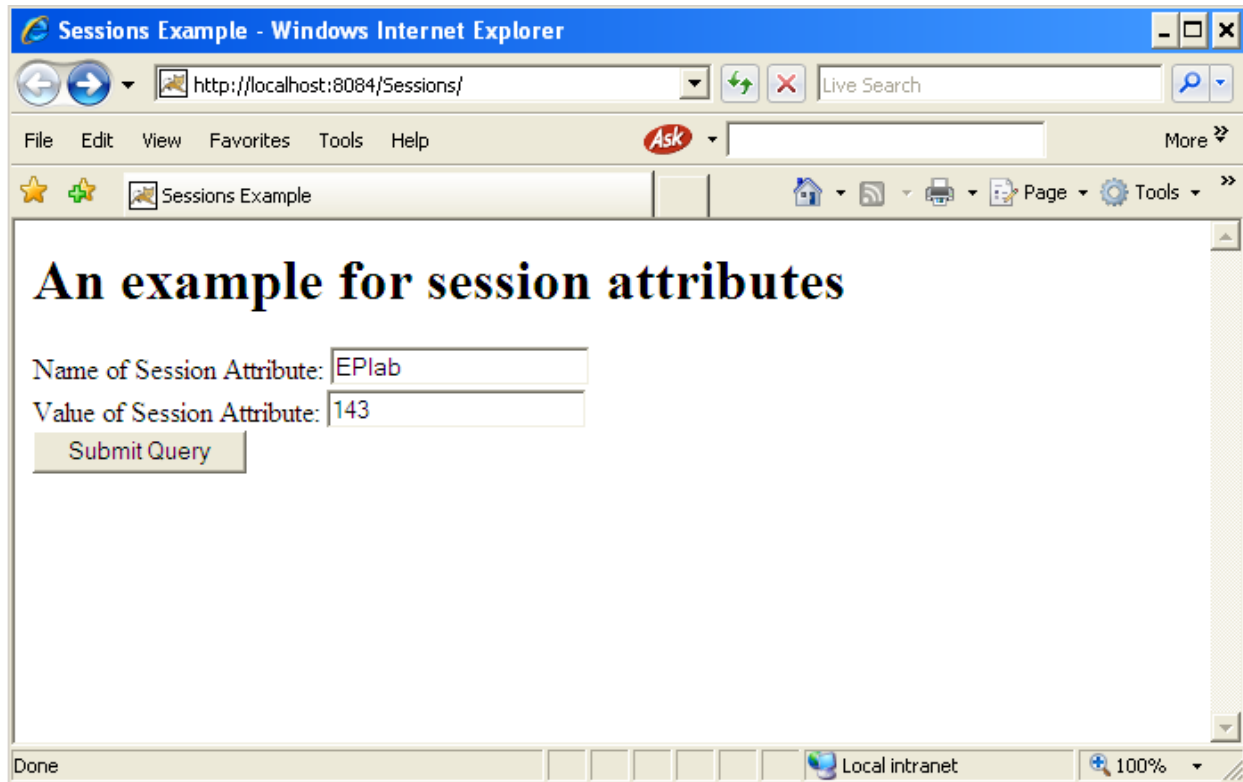
<url-pattern>/Session</url-pattern>

</servlet-mapping>

4.Restart Tomcat Server.Start->ControlPanel->Administrative tools->services.

5.Type URL in Addressbar <http://localhost:8080/Session.html>

## OUTPUT:



## 1(b). Write a program to demonstrate cookies using servlets.

**Aim:** To demonstrate a program on cookies using servlets

### **Packages:**

java.io

java.util

javax.servlet

javax.servlet.http

**Note:** Cookies are small bits of textual information that a Web server sends to a browser and that the browser returns unchanged when visiting the same Web site or domain later. By having the server read information it sent the client previously, the site can provide visitors with a number of conveniences:

- Identifying a user during an e-commerce session.
- Avoiding username and password.
- Customizing a site.
- Focusing advertising.

### **1. The Servlet Cookie API**

To send cookies to the client, a servlet would create one or more cookies with the appropriate names and values via new Cookie(name, value) set any desired optional attributes via cookie.setXXX, and add the cookies to the response headers via response.addCookie(cookie). To read incoming cookies, call request.getCookies(), which returns an array of Cookie objects. In most cases, you loop down this array until you find the one whose name (getName) matches the name you have in mind, then call getValue on that Cookie to see the value associated with that name.

#### **1.1 Creating Cookies**

A Cookie is created by calling the Cookie constructor, which takes two strings: the cookie name and the cookie value. Neither the name nor the value should contain whitespace or any of:

[ ] ( ) = , " / ? @ : ;

#### **1.2 Reading and Specifying Cookie Attributes**

Before adding the cookie to the outgoing headers, you can look up or set attributes of the cookie. Here's a summary:

##### **getComment/setComment**

Gets/sets a comment associated with this cookie.

**getDomain/setDomain**

Gets/sets the domain to which cookie applies. Normally, cookies are returned only to the exact hostname that sent them. You can use this method to instruct the browser to return them to other hosts within the same domain. Note that the domain should start with a dot (e.g. .prenhall.com), and must contain two dots for non-country domains like .com, .edu, and .gov, and three dots for country domains like .co.uk and .edu.es.

**getMaxAge/setMaxAge**

Gets/sets how much time (in seconds) should elapse before the cookie expires. If you don't set this, the cookie will last only for the current session (i.e. until the user quits the browser), and will not be stored on disk.

**getName/setName**

Gets/sets the name of the cookie. The name and the value are the two pieces you virtually always care about. Since the getCookies method of HttpServletRequest returns an array of Cookie objects, it is common to loop down this array until you have a particular name, then check the value with getValue.

**getPath/setPath**

Gets/sets the path to which this cookie applies. If you don't specify a path, the cookie is returned for all URLs in the same directory as the current page as well as all subdirectories. This method can be used to specify something more general. For example, someCookie.setPath("/") specifies that all pages on the server should receive the cookie. Note that the path specified must include the current directory.

**getSecure/setSecure**

Gets/sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e. SSL) connections.

**getValue/setValue**

Gets/sets the value associated with the cookie. Again, the name and the value are the two parts of a cookie that you almost always care about, although in a few cases a name is used as a boolean flag, and its value is ignored (i.e the existence of the name means true).

**getVersion/setVersion**

Gets/sets the cookie protocol version this cookie complies with. Version 0, the default, adheres to the original Netscape specification.

### **Cookie.html:**

```
<html>

    <head><title>Example Cookie Program</title></head>

    <body bgcolor="red">

        <form action="Cookie1" method=POST>

            Name: <input type="text" length=20 name="cookieName"><br>
            Value: <input type="text" length=20 name="cookieValue"><br>

                <input type="submit">

        </form>

    </body>

</html>
```

### **Cookie1.java:**

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Cookie1 extends HttpServlet
{
    public void doPost(HttpServletRequest req,HttpServletResponse res) throws
    IOException,ServletException
    {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();

        out.println("<html>");
        out.println("<head>");
```



```
        out.println("<title> example Cookies</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("hello");
        String name=req.getParameter("cookieName");
        if(name !=null && name.length() > 0){
            String value=req.getParameter("cookievalue");
            Cookie c=new Cookie(name,value);
            res.addCookie(c);
        }
        Cookie cookies[]= req.getCookies();
        for(int i=0;i<cookies.length;i++){
            Cookie c=cookies[i];
            String a=c.getName();
            String b=c.getValue();
            out.println(a + "=" + b);
        }
        out.println("</body>");
        out.println("</html>");
    }
}
```

### **Steps to Execute:**

1. Place Cookie1.class file in D:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\classes
2. Place Cookie.html file in D:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT
3. Change web.xml file in D:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\ROOT\WEB-INF\

Ex: <servlet>

<servlet-name>Cookiedemo</servlet-name>

<servlet-class>Cookie1</servlet-class>

</servlet>

<servlet-mapping>

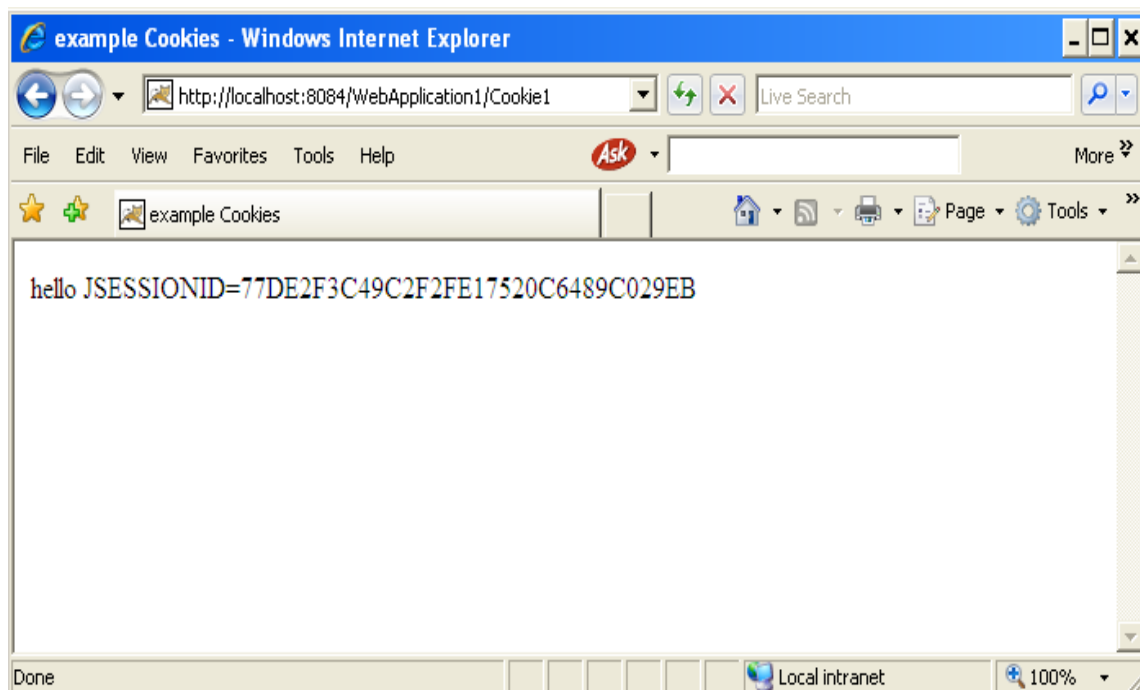
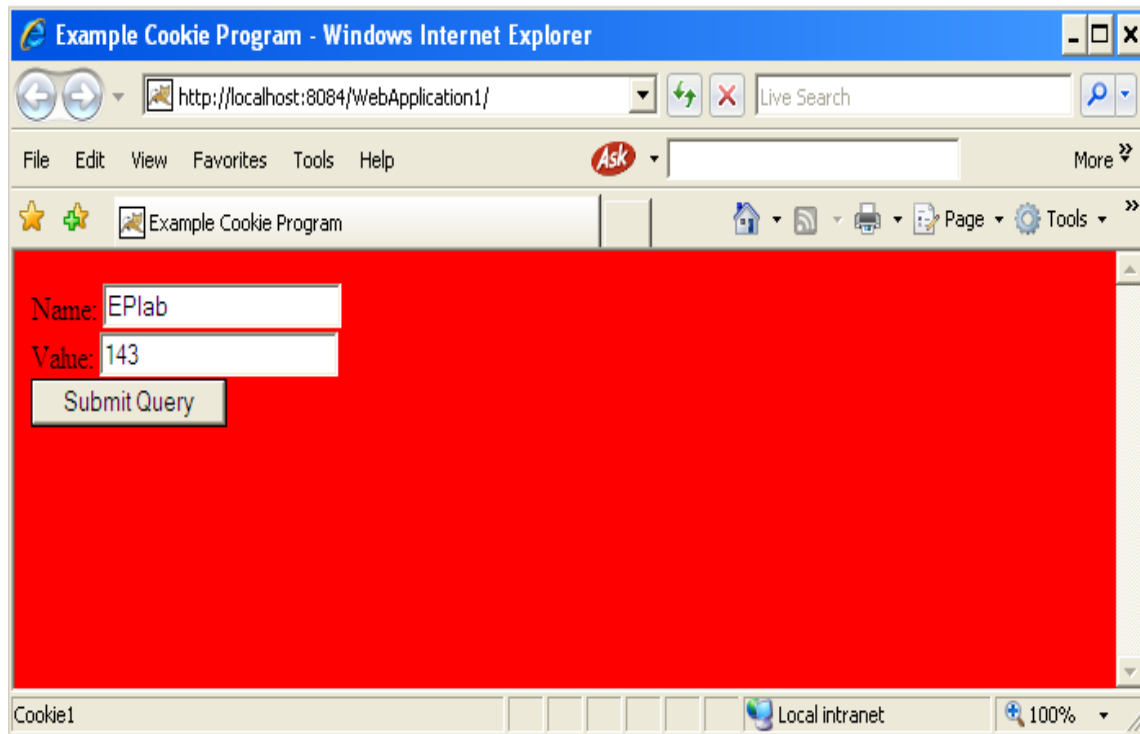
<servlet-name> Cookiedemo </servlet-name>

<url-pattern>/Cookie</url-pattern>

</servlet-mapping>

4. Restart Tomcat Server.Start->ControlPanel->Administrative tools->services.
5. Type URL in Addressbar <http://localhost:8080/Cookie.html>

## **OUTPUT:**



## **2. Write a Java Web application to integrate JSP & Servlets.**

**AIM:** To demonstrate a java web application to integrate Servlets and JSPs.

### **Procedure:**

- 1) Open NetBeans IDE.
- 2) Click on File, Choose New Project.
- 3) Select Web Application under Java Web and Click Next.
- 4) Choose a meaningful name for the project and specify the location for the project and Click Next.
- 5) Choose Server, Java EE Version and Context Path for the application and Click Finish.
- 6) This will create a new Web Application.
- 7) Right Click on Project, Select New and Select JSP.
- 8) Choose File Name and Click Finish.
- 9) Right Click on Project->New->Other->JavaBeans Objects->JavaBeans Component and Click Next.
- 10) Choose the Name, Package for the Bean file and Click Finish.
- 11) .Add Servlet to the Application.
- 12) Right click on project and New and Select Servlet.
- 13) Specify Name and Package for the servlet and Click Next.
- 14) Check the mark Add information to deployment descriptor and Click Finish.
- 15) This will add the new Servlet to the Project.
- 16) If you want to add a new java class to the existing project, then right click on project and New and Select Java Class.
- 17) Choose the name for the class, Specify the package and click Finish.
- 18) Connect to your Database.
- 19) Create table UNTITLED for accessing user details from Database and for INSERTING the details of corresponding data entry.
- 20) After the completion of project, Right Click on it and select Clean and Build.
- 21) Go to Services tab, expand Servers and Right Click on Glassfish Server and select Start.
- 22) Now Right Click on Project and Click Run.
- 23) This will open your application in browser.

### **Index.html:**

```
<html>

  <head>

    <title>TODO supply a title</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

  </head>

  <body>

    <form action="NewServlet">

      Enter roll :<input type="text" name="roll"/>

      Enter name :<input type="text" name="name"/>

      Enter mobno :<input type="text" name="mobno"/>

      <input type="submit" />

    </form>

  </body>

</html>
```

### **NewServlet.java:**

```
import abc.NewSessionBeanLocal;

import java.io.IOException;

import java.io.PrintWriter;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

```

public class NewServlet extends HttpServlet {

    @EJB

    private NewSessionBeanLocal n;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet NewServlet</title>");
            out.println("</head>");
            out.println("<body>");

            int roll= Integer.parseInt(request.getParameter("roll"));

            String name=request.getParameter("name");

            String mobno=request.getParameter("mobno");

            boolean k= n.insert(roll, name, mobno);

            if (k==true)

                request.getRequestDispatcher("/login.jsp").include(request, response);

            else

                request.getRequestDispatcher("/loginfail.jsp").include(request, response);
        }
    }
}

```

```

        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    } // </editor-fold>
}

```

### **login.jsp:**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>JSP Page</title>

    </head>

    <body>

        <%

            out.println("<h1>new record had been inserted</h1>");

        %>

    </body>

</html>
```

### **loginfail.jsp:**

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>JSP Page</title>

    </head>

    <body>

        <%

            out.println("<h1>insertion failed !! Try again !!</h1>");

        %>

    </body>

</html>
```



```
%>
</body>
</html>
```

### **NewSessionBeanLocal.java:**

```
package abc;

import javax.ejb.Local;

@Local

public interface NewSessionBeanLocal {

    public boolean insert(int roll,String name,String mob);

}
```

### **NewSessionBean.java:**

```
package abc;

import static java.lang.Character.UnicodeBlock.forName;

import javax.ejb.Stateless;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

@Stateless

public class NewSessionBean implements NewSessionBeanLocal {

    @Override

    public boolean insert(int roll,String name,String mob)

    {
```

```

int i=0;

try {
    Class.forName("org.apache.derby.jdbc.ClientDriver");
} catch (ClassNotFoundException ex) {
    Logger.getLogger(NewSessionBean.class.getName()).log(Level.SEVERE, null, ex);
}

try {
    Connection con;

    con = DriverManager.getConnection("jdbc:derby://localhost:1527/it","it","it");

    PreparedStatement ps=con.prepareStatement("insert into UNTITLED values (?,?,?)");

    ps.setInt(1,roll);

    ps.setString(2,name);

    ps.setString(3,mob);

    i=ps.executeUpdate();

} catch (SQLException ex) {
    Logger.getLogger(NewSessionBean.class.getName()).log(Level.SEVERE, null, ex);
}

if (i==1)
{
    return true;
}

else

    return false;
}
}

```

## **Web.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
```

```
    <servlet>
```

```
        <servlet-name>NewServlet</servlet-name>
```

```
        <servlet-class>NewServlet</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

```
        <servlet-name>NewServlet</servlet-name>
```

```
        <url-pattern>/NewServlet</url-pattern>
```

```
    </servlet-mapping>
```

```
    <session-config>
```

```
        <session-timeout>
```

```
            30
```

```
        </session-timeout>
```

```
    </session-config>
```

```
</web-app>
```

## **OUTPUT:**

← → ↻ ⓘ localhost:8080/EnterpriseApplication1-war/NewServlet?roll=45543&name=pawan&mobno=64677

**Hello World!**

**new record had been inserted**

---

← → ↻ ⓘ localhost:8080/EnterpriseApplication1-war/NewServlet?roll=2&name=tr&mobno=gfcxg

**insertion failed !! Try again !!**

### **3. Write a JSF application to demonstrate Party Planner.**

**AIM:** To demonstrate a JSF application to demonstrate Party Planner.

#### **PROCEDURE:**

1. Open NetBeans IDE.
2. Click on File, Choose New Project.
3. Select Web Application under Java Web and Click Next.
4. Choose a meaningful name for the project and specify the location for the project and Click Next.
5. Choose Server, Java EE Version and Context Path for the application and Click Next.
6. Choose the JavaServer Faces Framework and Click Finish.
7. This will create a new JSF Application.
8. Right Click on Project->New->Other->JavaServer Faces->JSF Managed Bean,Click Next.
9. Specify the Name and Package for the bean file and Click Finish.
10. Add Java class to the Project.
11. If you want to add a new java class to the existing project, then right click on project and New and Select Java Class.
12. Choose the name for the class, Specify the package and click Finish.
13. After the completion of project, Right Click on it and select Clean and Build.
14. Go to Services tab, expand Servers and Right Click on Glassfish Server and select Start.
15. If there is no server, Install one.
16. Now Right Click on Project and Click Run.
17. This will open your application in browser.

## **index.xhtml:**

```
<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

    xmlns:f="http://xmlns.jcp.org/jsf/core"

    xmlns:h="http://xmlns.jcp.org/jsf/html">

    <h:head>

        <title>#{partyBean.name}</title>

    </h:head>

    <h:body>

        <div align="center">

            <h:form>

                <label style="font-weight:bold"> Party Pieces </label> <br/><br/>

                <label>Choose a title: </label>

                <h:inputText value = "#{partyBean.name}" ></h:inputText><br/>

                <label>Choose a picture: </label>

                <h:selectOneMenu value="#{partyBean.imageUri}">

                    <f:selectItem itemValue="party1.jpg" itemLabel="Hats" />

                    <f:selectItem itemValue="party2.jpg" itemLabel="Balloons" />

                </h:selectOneMenu><br/>

                <label>Are parents allowed ? </label>

                <h:selectBooleanCheckbox value="#{partyBean.parentsAllowed}"
            ></h:selectBooleanCheckbox><br/>
```

```

        <h:commandButton value="Submit picture and parent choices" action="index"
/><br/>
        <h:commandButton value="Reset to default" action="#{partyBean.reset}"
/><br/><br/>
        </h:form>
        <br/><br/>
        <label style="font-weight:bold"> Party Summary </label> <br/><br/>
        <h:outputText value="#{partyBean.summary}"/><br/><br/>
        <h:dataTable value="#{partyBean.items}" var="item">
            <h:column >
                #{item.name}
            </h:column>
            <h:column>
                #{item.age}
            </h:column>
        </h:dataTable>
        <br/><br/>
        <h:graphicImage value="#{partyBean.imageUri}" width="200" height="171" />
        <p>
            <h:form>
                <h:commandLink value="Follow me for a surprise..." action="surprise.xhtml" />
            </h:form>
        </p>
    </div>
</h:body>
</html>

```

### **Surprise.xhtml:**

```
<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

    xmlns:h="http://xmlns.jcp.org/jsf/html">

    <h:head>

        <title>Surprise !</title>

    </h:head>

    <h:body>

        <div align='center'>

            <br/>

            Hope the party goes off with a bang !

            <p>

                <h:form>

                    <h:commandLink value="Return to #{partyBean.name}" action="index" />

                </h:form>

            </p>

        </div>

    </h:body>

</html>
```

### **PartyBean.java**

```
package javaeems.chapter4.widgets;

import javax.enterprise.context.*;

import javax.inject.Named;
```



```
import java.util.*;
import java.io.Serializable;
@Named("partyBean")
@SessionScoped
public class PartyBean implements Serializable {
    private String name;
    private boolean parentsAllowed;
    private List<Guest>items;
    private String imageUri;
    public PartyBean() {
        this.reset();
    }
    public String getImageUri() {
        return this.imageUri;
    }

    public void setImageUri(String imageUri) {
        this.imageUri = imageUri;
    }
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
}
```

```
public boolean getParentsAllowed() {  
    return parentsAllowed;  
}
```

```
public void setParentsAllowed(boolean on) {  
    this.parentsAllowed = on;  
}
```

```
public List<Guest> getItems() {  
    return this.items;  
}
```

```
public void reset() {  
    this.name = "(party title)";  
    this.parentsAllowed = false;  
    this.items = new ArrayList<>();  
    items.add(new Guest("Sally", 6));  
    items.add(new Guest("Carlos", 7));  
    items.add(new Guest("Nithan", 6));  
    this.imageUri = "party1.jpg";  
}
```

```
public String getSummary() {  
    StringBuilder sb = new StringBuilder();  
    sb.append("My party is called " + this.name);  
}
```

```
        sb.append(", ");
        sb.append(" there are " + items.size() + " guests");
        sb.append(", ");
        sb.append("and parents are " + (parentsAllowed ? "" : "not") + " allowed to stay.");
        return sb.toString();
    }
}
```

### **Guest.java:**

```
package javaeems.chapter4.widgets;

public class Guest {
    private String name;
    private int age;

    public Guest(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return this.age;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
public String getName() {  
    return this.name;  
}  
public String toString() {  
    return "Item: " + name;  
}  
}
```

## OUTPUT:

← → ↻ ⓘ localhost:8080/PartyPlanner/

---

### Party Pieces

Choose a title:


Choose a picture:  ▼

Are parents allowed ? ☐

### Party Summary

My party is called (party title), there are 3 guests, and parents are not allowed to stay.

Sally	6
Carlos	7
Nithan	6



[Follow me for a surprise...](#)



#### **4. Write a Program to demonstrate Asynchronous web sockets.**

**Aim:** To demonstrate a program on Asynchronous web sockets.

**PROCEDURE:**

1. Open NetBeans IDE.
2. Click on File, Choose New Project.
3. Select Web Application under Java Web and Click Next.
4. Choose a meaningful name for the project and specify the location for the project and Click Next.
5. Create a WebSocket Endpoint.
6. Right Click on project->New->Other->Web->WebSocket Endpoint.
7. Specify name, package and endpoint for the websocket endpoint file and then click Finish.
8. In index.html create a new websocket by using JavaScript.

**Index.html:**

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Chat Server</title>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <script type="text/javascript">
```

```
      var websocket=new
```

```
      WebSocket("ws://localhost:8080/ChatRoomServer/chatroomServerEndpoint");
```

```
      websocket.onmessage=function processMessage(message){
```

```
        if(message.data!=null)
```

```
          messageTextArea.value+=message.data+"\n";
```

```

    }

    function sendMessage(){

        websocket.send(messageText.value);

        messageText.value="";

    }

</script>

</head>

<body>

    <textarea id="messageTextArea" rows="10" cols="45"
readonly="readOnly"></textarea><br/>

    <input type="text" id="messageText" size="50"/>

    <input type="button" value="send" onclick="sendMessage()"/>

</body>

</html>

```

### **ChatRoomServerEndpoint.java:**

```

package aaa;

import java.io.IOException;

import javax.websocket.*;

import javax.websocket.server.ServerEndpoint;

import java.util.*;

@ServerEndpoint("/chatroomServerEndpoint")

public class ChatRoomServerEndpoint {

    static Set<Session> chatroomUsers=Collections.synchronizedSet(new HashSet<Session>());

    @OnOpen

```

```

public void handleOpen(Session userSession){
    chatroomUsers.add(userSession);
}

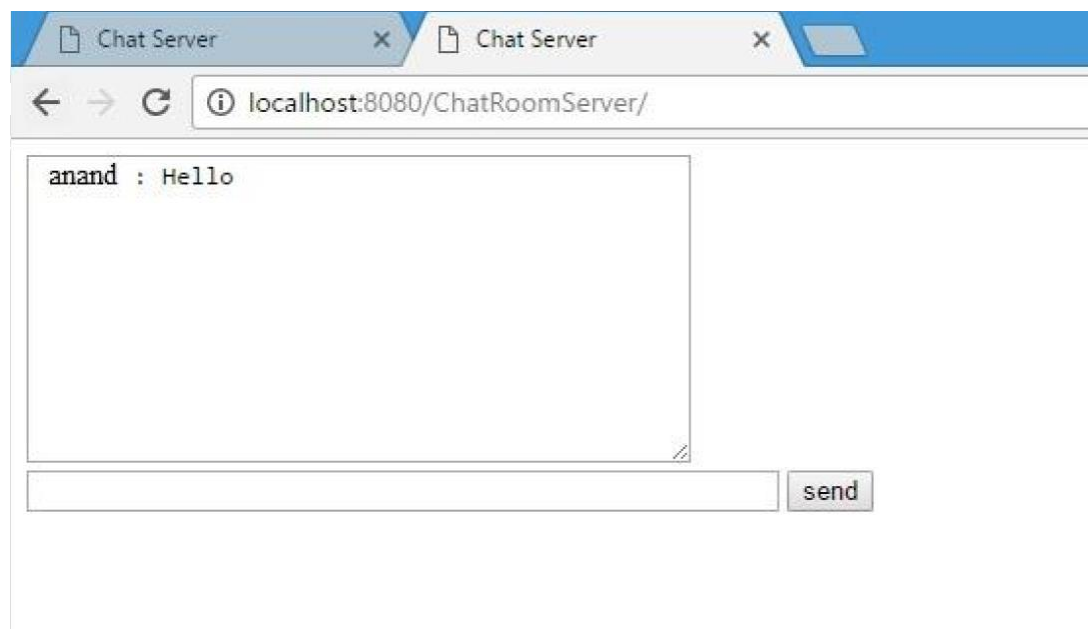
@OnMessage
public void handleMessage(String message,Session userSession) throws IOException{
    String username=(String)userSession.getUserProperties().get("username");
    if(username==null){
        userSession.getUserProperties().put("username",message);
        userSession.getBasicRemote().sendText("You are now connected as:"+message);
    }
    else{
        Iterator <Session> iterator=chatroomUsers.iterator();
        while(iterator.hasNext()){
            iterator.next().getBasicRemote().sendText(username+" : "+message);
        }
    }
}

@OnClose
public void handleClose(Session userSession){
    chatroomUsers.remove(userSession);
}
}

```



## OUTPUT:



## 5. Write a Program to Demonstrate Stateful Session Bean (Shopping Cart).

**Aim:** To Demonstrate a program on Stateful Session Bean(Shopping Cart).

### Packages:

Java.util

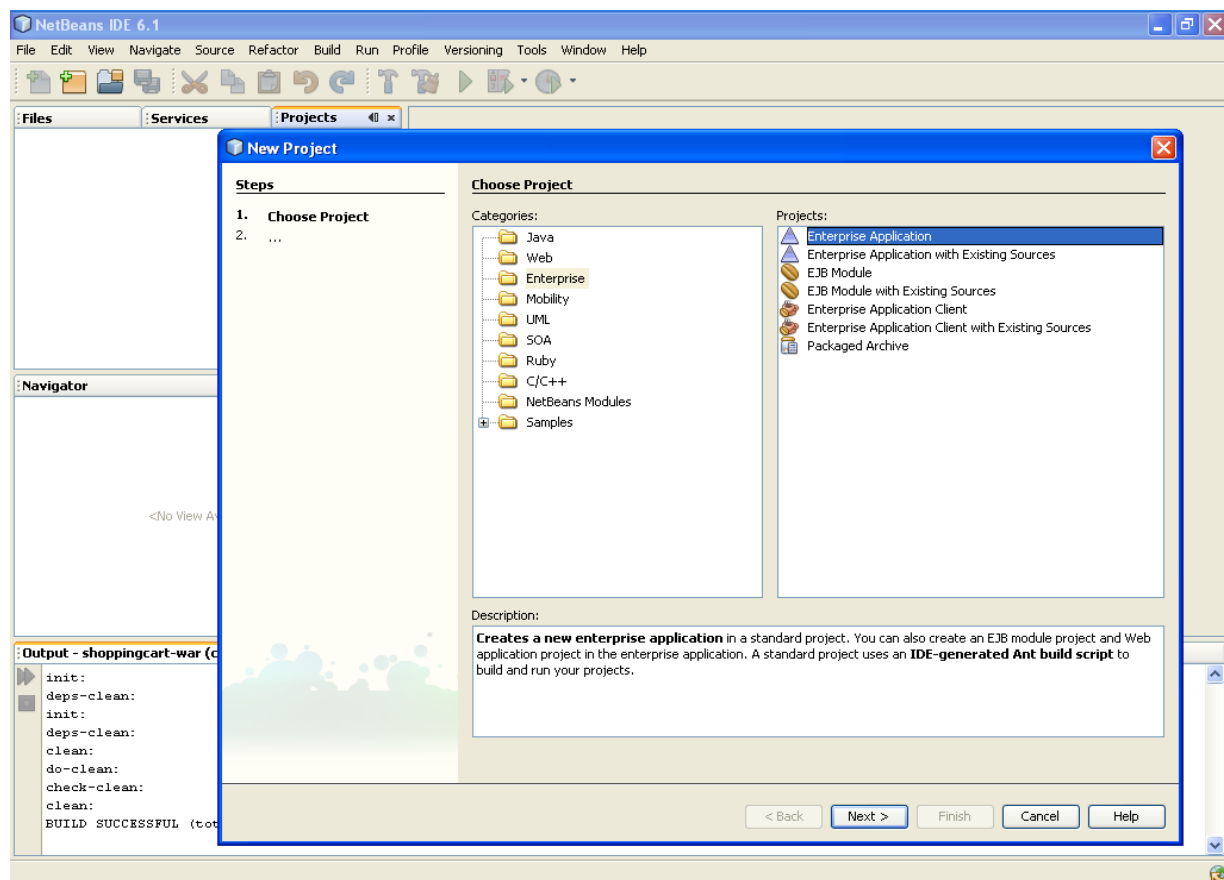
javax.annotation

javax.ejb

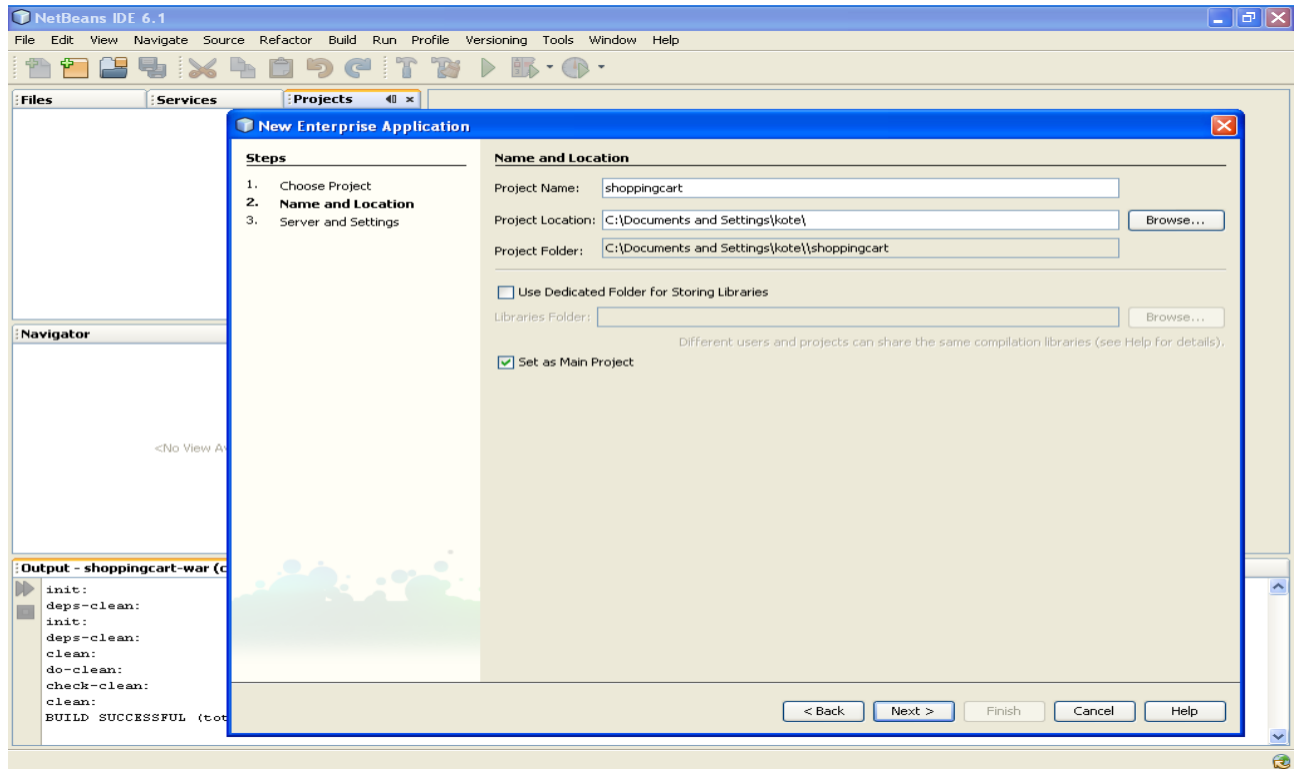
javax.naming

### Steps to Create Stateful Sessionbean:

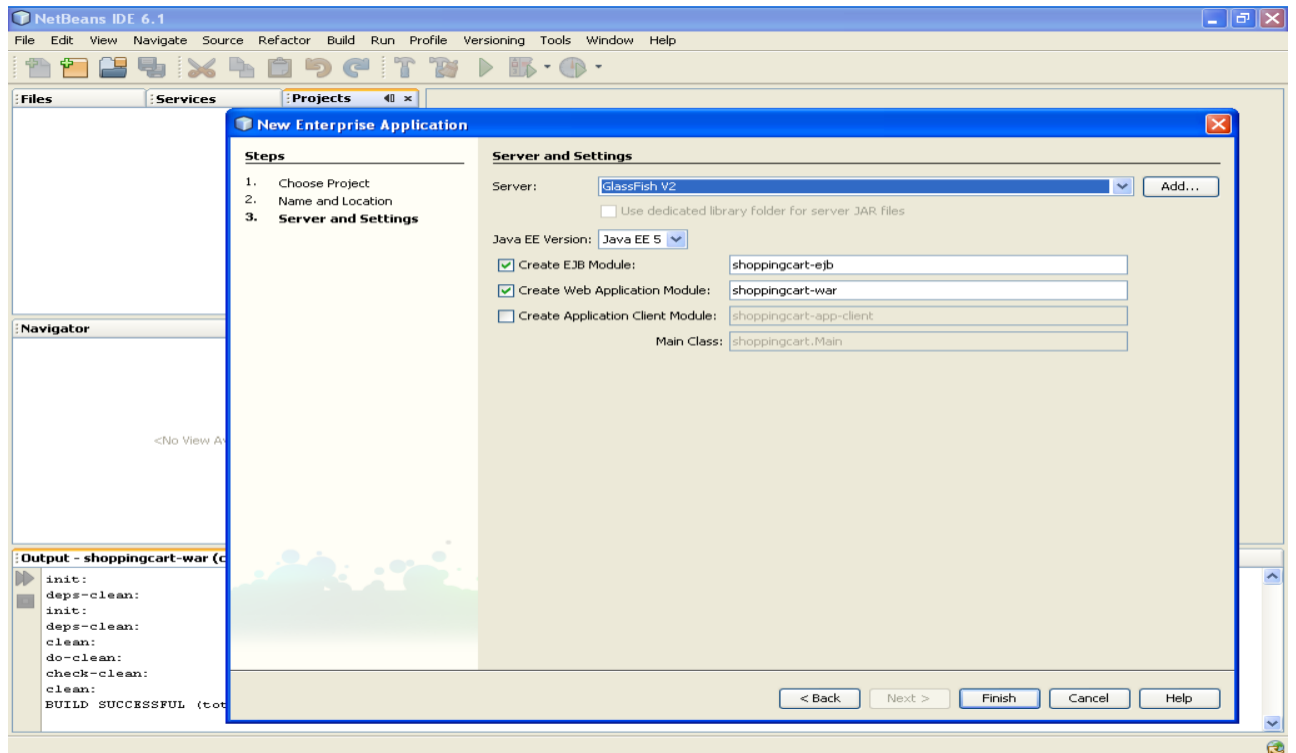
STEP1: Open the netbeans IDE 6.\* and open the file newproject and select enterprise application.



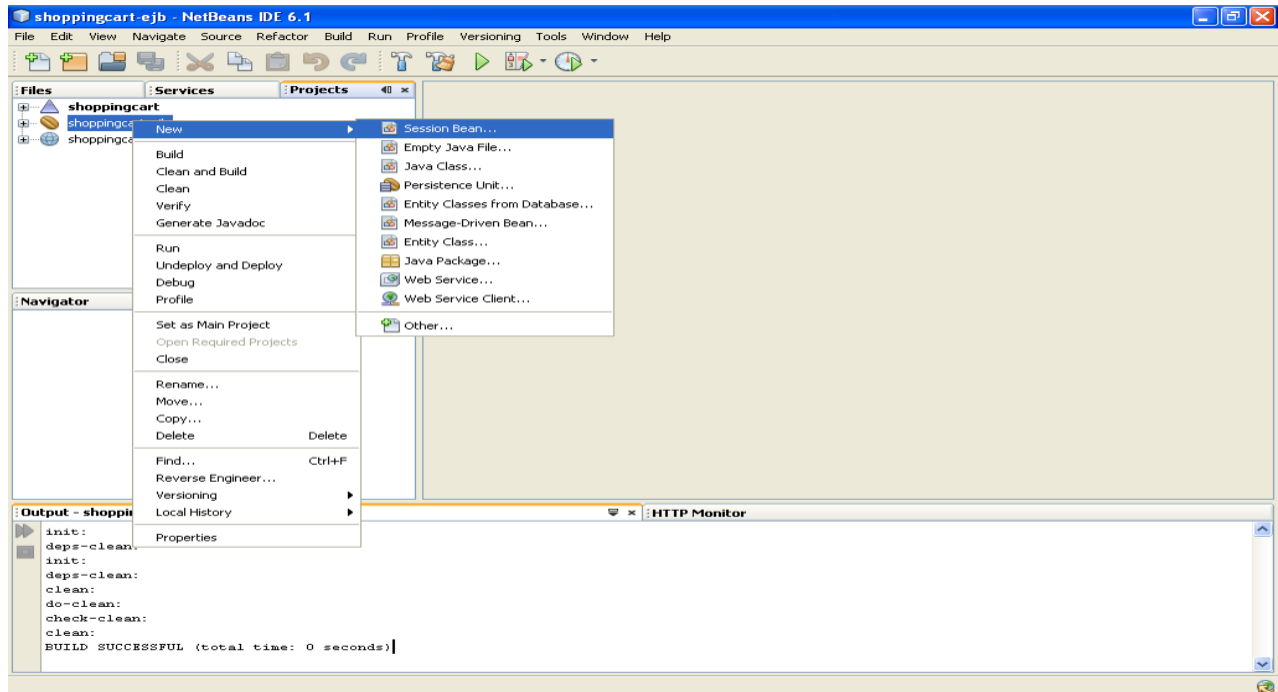
Step2: Name the application as shopping cart and set the project location



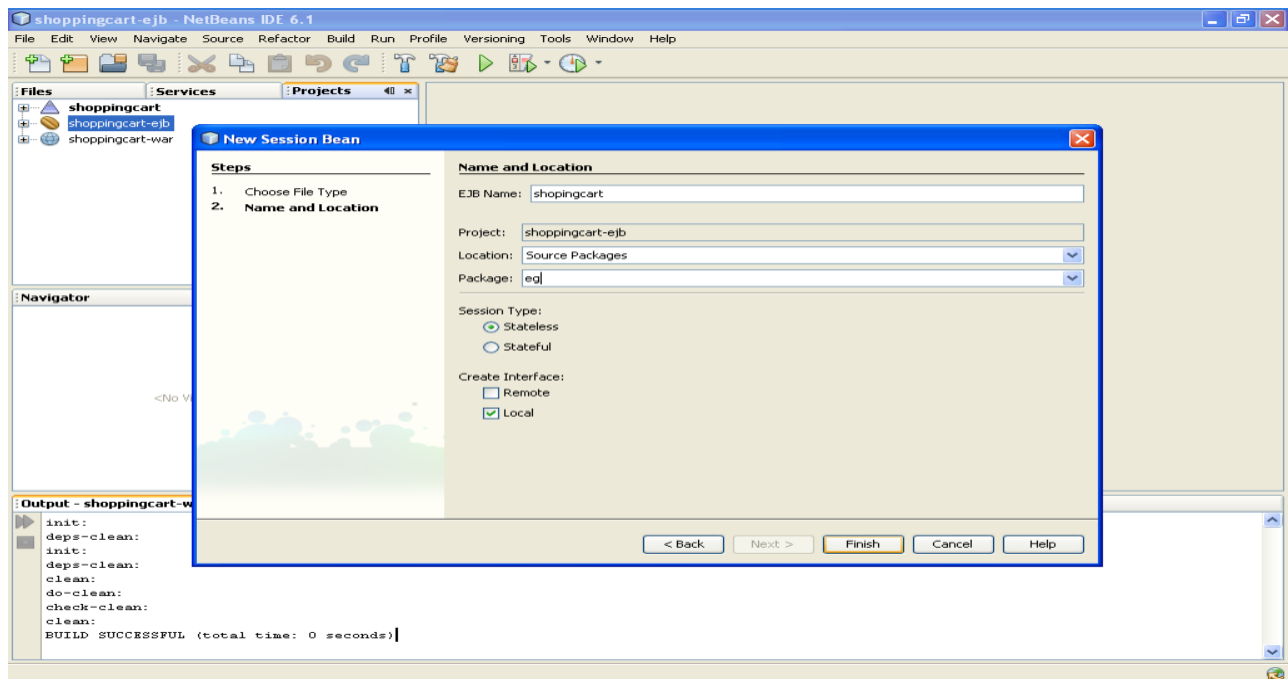
Step3: Mark either the web application module or client module



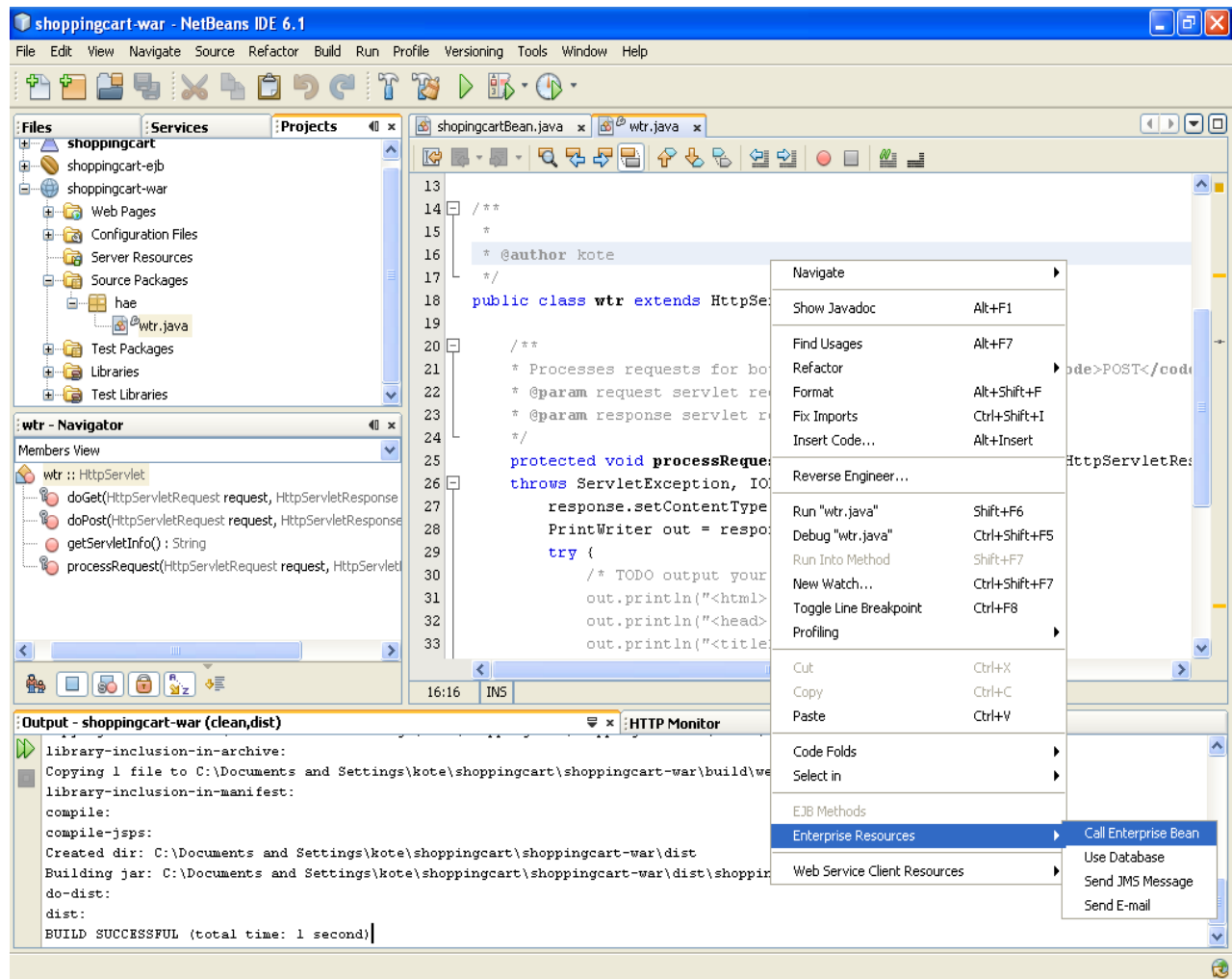
Step 4: Right click the bean and create new session bean



Step4: Name the bean with shopping cart and enter the package and mark either the local or remote. Local if bean is different system then mark it as remote otherwise Local



Step5: Create the servlet in war file by right clicking the shoppingcar-war.Right click in the servlet file and select call enterprise bean to create instance of remote bean



### **ShopingCartBean.java:**

```
package com.ejb;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.Stateful;

@Stateful

public class ShoppingCartBean implements ShoppingCartRemote {

    private Item item;

    private ArrayList order,items;

    @PostConstruct

    public void initialize() {

        item=null;

        items = new ArrayList();

        order=new ArrayList();

    }

    @PreDestroy

    public void pdestroy() {

        System.out.println("Inside PreDestroy()");

    }

    public void addItem(String pn,int q,double up) {

        item=new Item(pn,q,up);

        items.add(item);

    }

}
```

```

    }

    public void removeItem(String pn) {

        boolean flag=false;

        for (Iterator i = items.iterator(); i.hasNext();) {

            item=(Item)i.next();

            if(item.pname.equals(pn)) {

                items.remove(item);

                flag=true;

                break;

            }

        }

        if(flag)

            System.out.println(item.pname + " removed from Cart");

        else

            System.out.println(item.pname + " not found in Cart");

    }

    public Collection.getItems() {

        for (Iterator i = items.iterator(); i.hasNext();) {

            item=(Item)i.next();

            order.add(item.toString());

        }

        return order;

    }

}

class Item {

```

```
public String pname;

public int Qty;

public double uprice;

public Item(String pn,int q,double up) {

    pname=pn;

    Qty=q;

    uprice=up;

}

@Override

public String toString(){

    return "Product=" + pname + " Quantity=" + Qty + " UnitPrice=" + uprice;

}

}
```

### **ShopingCartRemote.java:**

```
package com.ejb;

import java.util.Collection;

import javax.ejb.Remote;

public interface ShoppingCartRemote {

    void addItem(String pn,int q,double up);

    void removeItem(String pn);

    Collection.getItems();

}
```



### **NewServlet.java:**

```
package c;

import e.NewSessionBeanLocal;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(name="NewServlet", urlPatterns={"/NewServlet"})

public class NewServlet extends HttpServlet {

    NewSessionBeanLocal n = lookupNewSessionBeanLocal1();

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();

        String name=request.getParameter("t1");

        int qty=Integer.parseInt(request.getParameter("t2"));

        double price=Double.parseDouble(request.getParameter("t3"));
```

```

n.addItem(name,qty,price);

out.println( "item descript"+n.getItems());

out.println(" <a href=index.jsp>add some more items</a>");
}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    processRequest(request, response);
}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    processRequest(request, response);
}

@Override

public String getServletInfo() {

    return "Short description";
}

private NewSessionBeanLocal lookupNewSessionBeanLocal() {

    try {

        Context c = new InitialContext();

        return (NewSessionBeanLocal) c.lookup("java:global/stateful/stateful-
ejb/NewSessionBean!be.NewSessionBeanLocal");

    } catch (NamingException ne) {

        Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);
    }
}

```

```

        throw new RuntimeException(ne);
    }
}

private NewSessionBeanLocal lookupNewSessionBeanLocal1() {
    try {
        Context c = new InitialContext();

        return (NewSessionBeanLocal) c.lookup("java:global/carr/carr-
ejb/NewSessionBean!e.NewSessionBeanLocal");
    } catch (NamingException ne) {
        Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);
        throw new RuntimeException(ne);
    }
}
}

```

### **Index.jsp:**

```

<html>

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <title>JSP Page</title>

</head>

<body>

    <h1>Hello World!</h1>

    <form action="NewServlet">

        name <input type="text" name="t1"><br>

        quanti<input type="text" name="t2"><br>

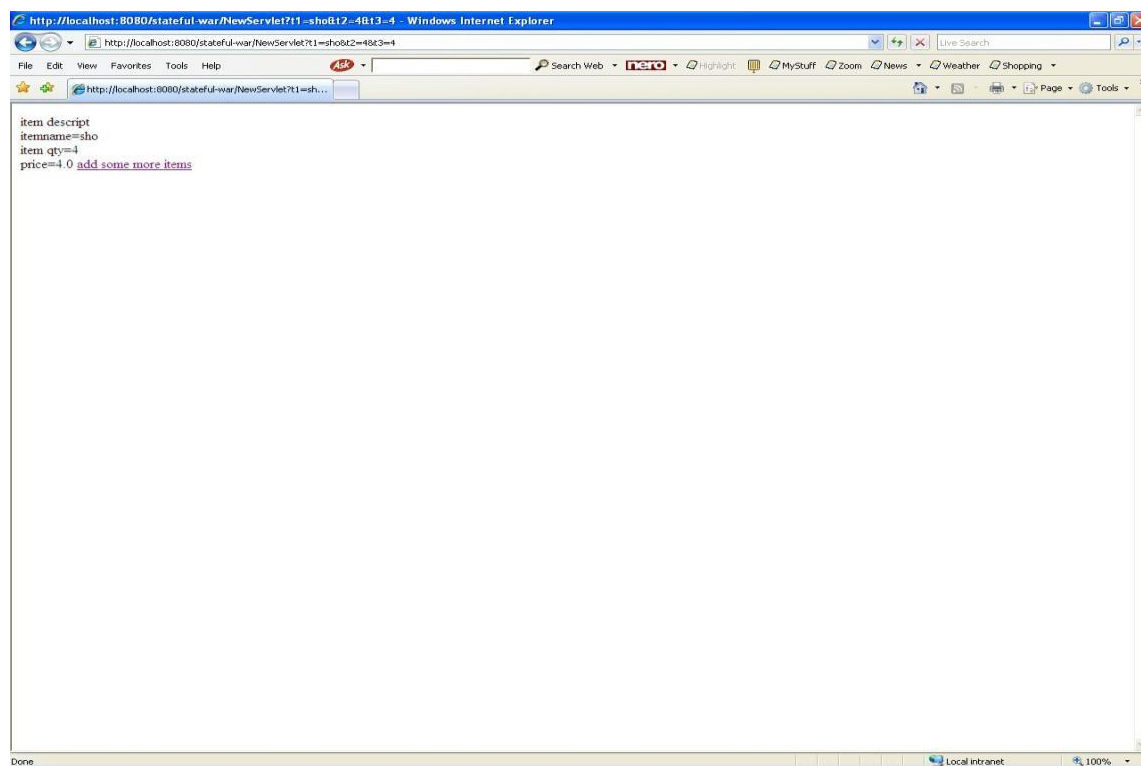
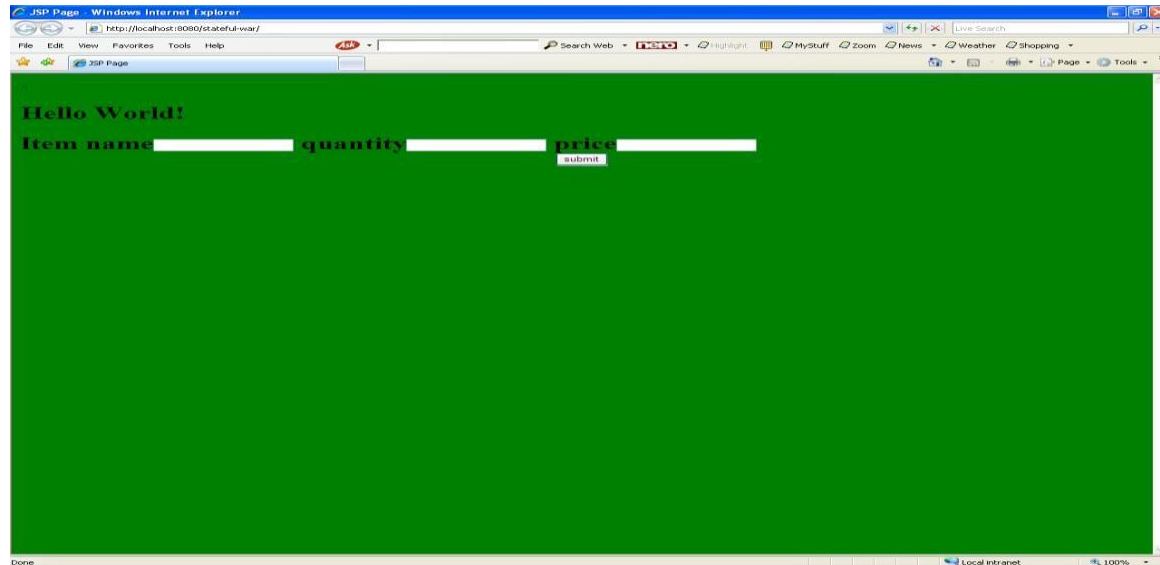
```

```
valu<input type="text" name="t3"><br>
<input type="submit">
</form>
</body>
</html>
```

### **Execution Steps:**

1. Build Project
2. Run Project

## Output:



## 6. Write a program to demonstrate Message Driven Bean.

**Aim:** To Demonstrate a program on Message Driven Bean.

### **Packages:**

Javax.annotation,

Javax.ejb,

Javax.jms,

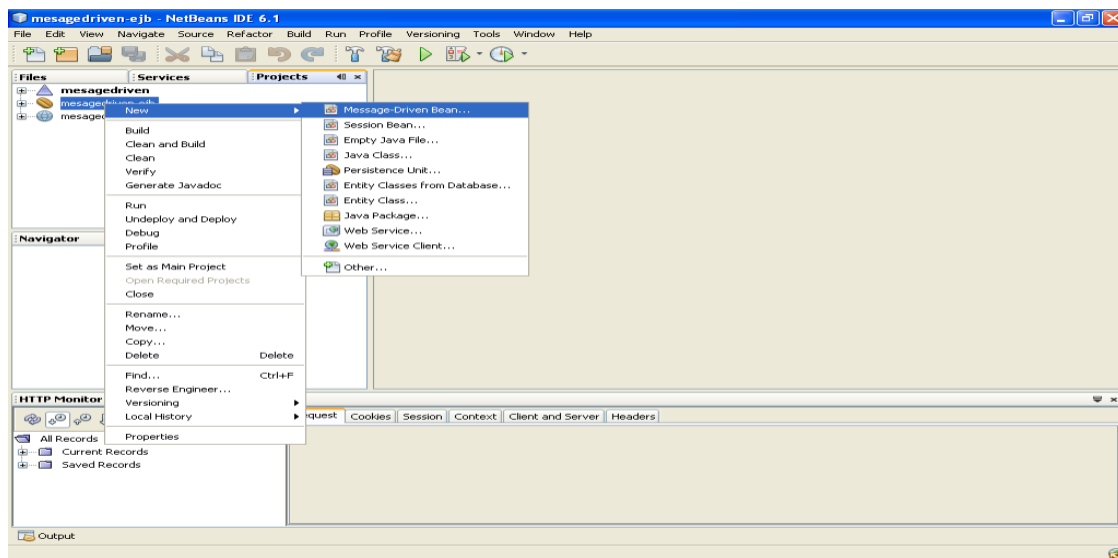
Java.util,

Javax.io,

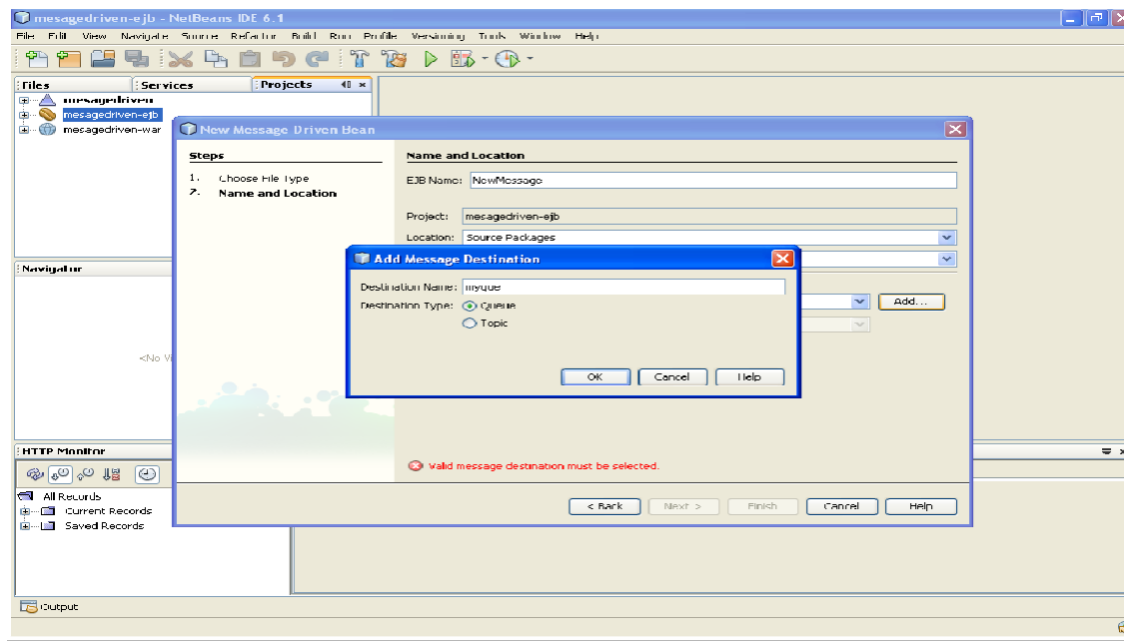
Javax.servlet.http

### **Steps to Create Message Driven Bean:**

Step1: Create the application project type as enterprise application and right click the message ejb and select new message driven bean.



Step2: Add the message destination as queue and name it as myque



### msgbean.java:

package ab;

import javax.annotation.PreDestroy;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.MessageDriven;

import javax.jms.JMSEException;

import javax.jms.Message;

import javax.jms.MessageListener;

import javax.jms.TextMessage;

@MessageDriven(mappedName = "jms/newMessage", activationConfig = {

    @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =  
    "Auto-acknowledge"),

    @ActivationConfigProperty(propertyName = "destinationType", propertyValue =  
    "javax.jms.Queue")

})

public class msgbean implements MessageListener {

```

public mesgbean() {
}

@Override

public void onMessage(Message message) {
    System.out.println("bean");
    if(message instanceof TextMessage){
        TextMessage tm=(TextMessage)message;
        try{
            String te=tm.getText();
            System.out.println("received message is"+te);
        }catch(JMSEException e){
        }
    }
}

@PreDestroy

public void remove(){
}

```

### **NewServlet1.java:**

```

package asd;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.annotation.Resource;

import javax.jms.Connection;

```



```

import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name="NewServlet1", urlPatterns={"/NewServlet1"})
public class NewServlet1 extends HttpServlet {
    @Resource(name = "jms/newMessage")
    private Queue newMessage;
    @Resource(name = "jms/newMessageFactory")
    private ConnectionFactory newMessageFactory;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try{
            try {
                String note=request.getParameter("body");

```

```

        if(note!=null)
        { Connection conn=newMessageFactory.createConnection();

        Session ses=conn.createSession(false,Session.AUTO_ACKNOWLEDGE);

        MessageProducer mp=ses.createProducer(newMessage);

        TextMessage tm=ses.createTextMessage();

        tm.setText(note);

        mp.send(tm);

        out.println("<h1>Your Message is transferred successfully</h1>");

        mp.close();

        }

    } catch (JMSEException e) {

        }}catch(Exception e)

    {

    }

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

    processRequest(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

    processRequest(request, response);

}

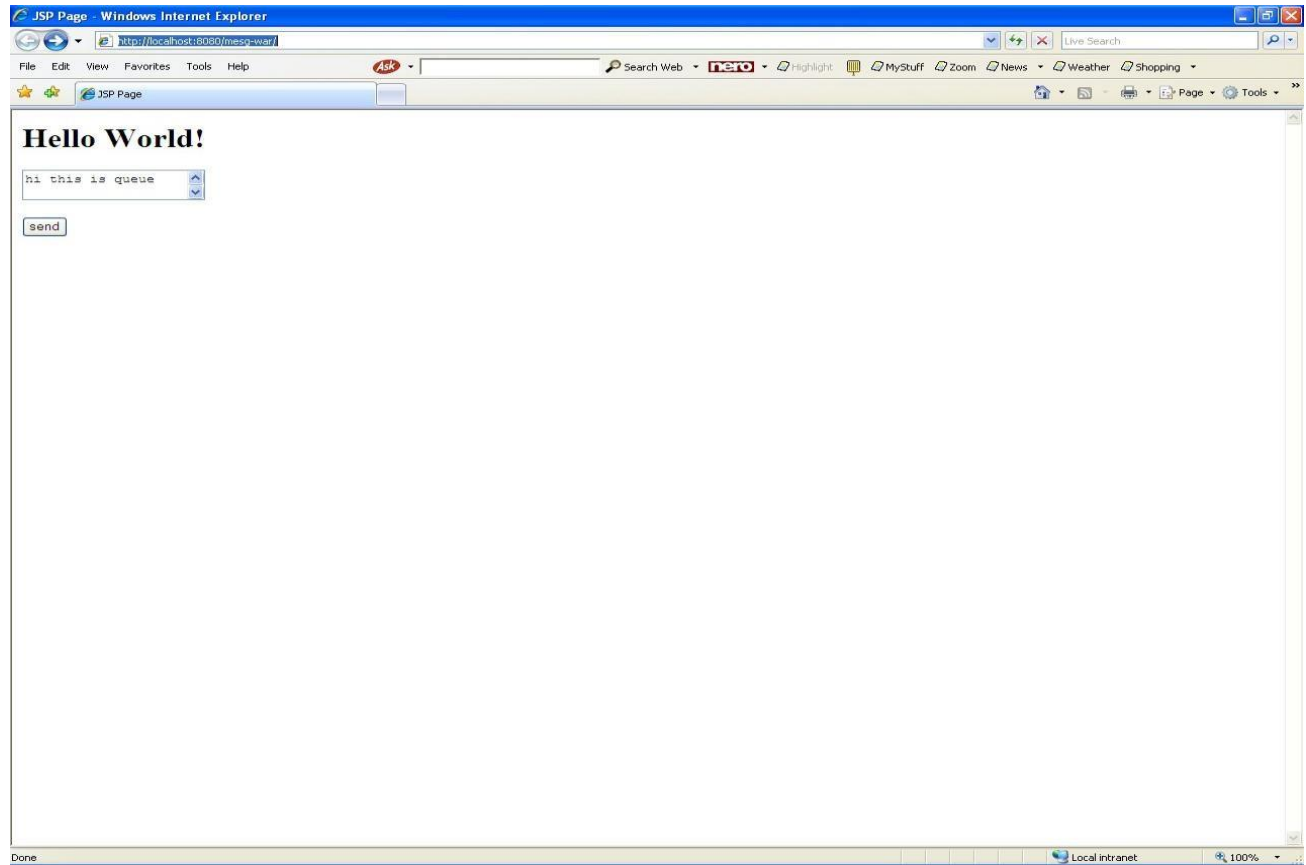
```

```
@Override  
  
public String getServletInfo() {  
    return "Short description";  
}  
}
```

### **Index.jsp:**

```
<html>  
  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>JSP Page</title>  
  </head>  
  
  <body>  
    <h1>Hello World!</h1>  
    <form action="NewServlet1">  
      <textarea name="body">  
        </textarea><br><br>  
      <input type="submit" value="send">  
    </form>  
  </body>  
</html>
```

## Output:

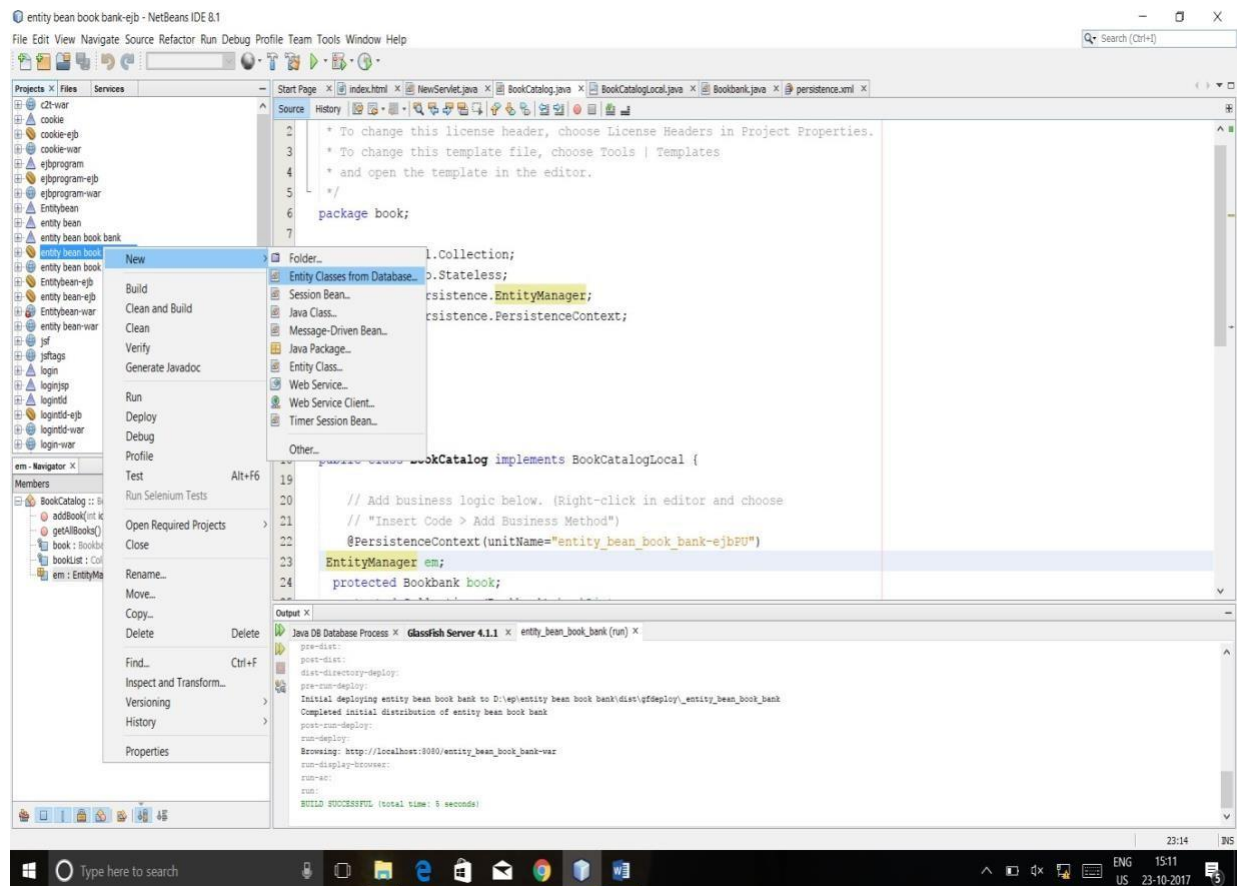


## 7. Write a program to demonstrate Entity Bean

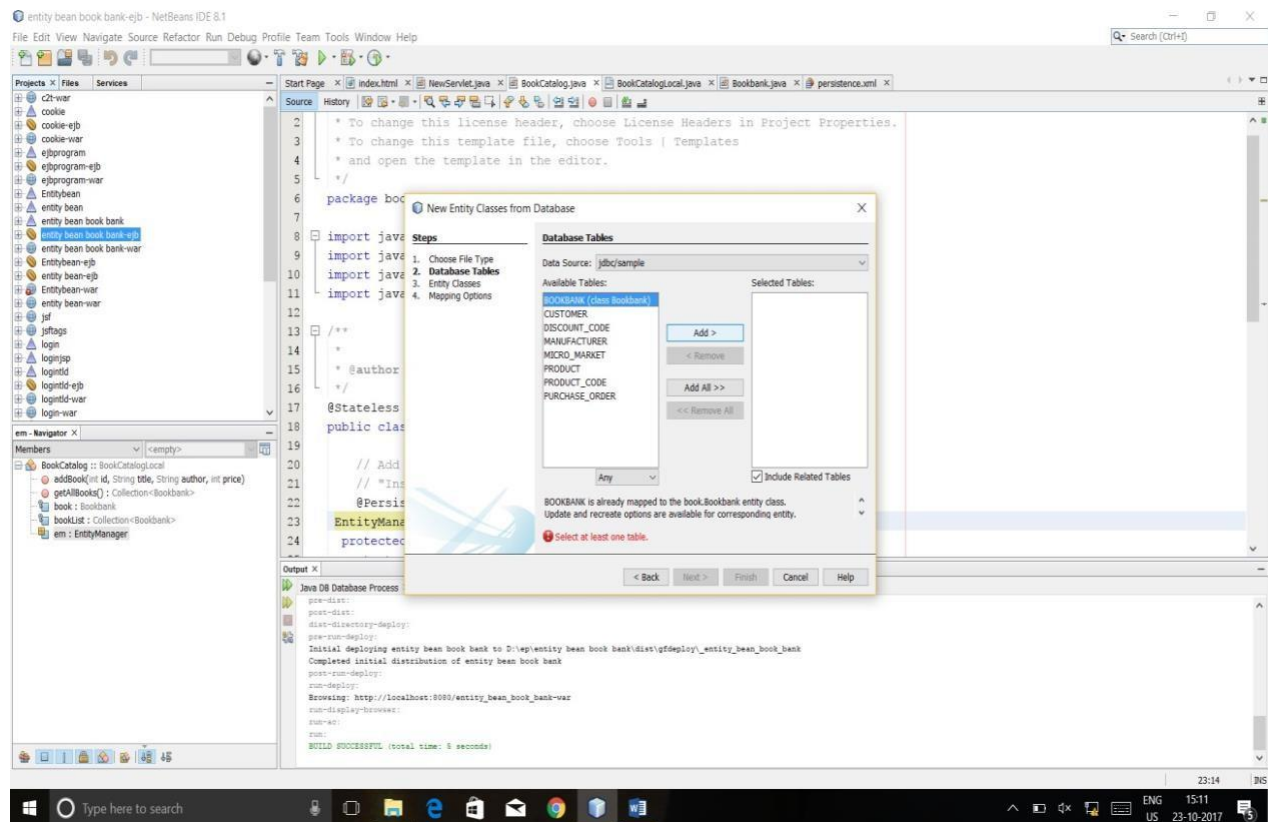
**Aim:** To demonstrate a Program on Entity Bean

### **Procedure:**

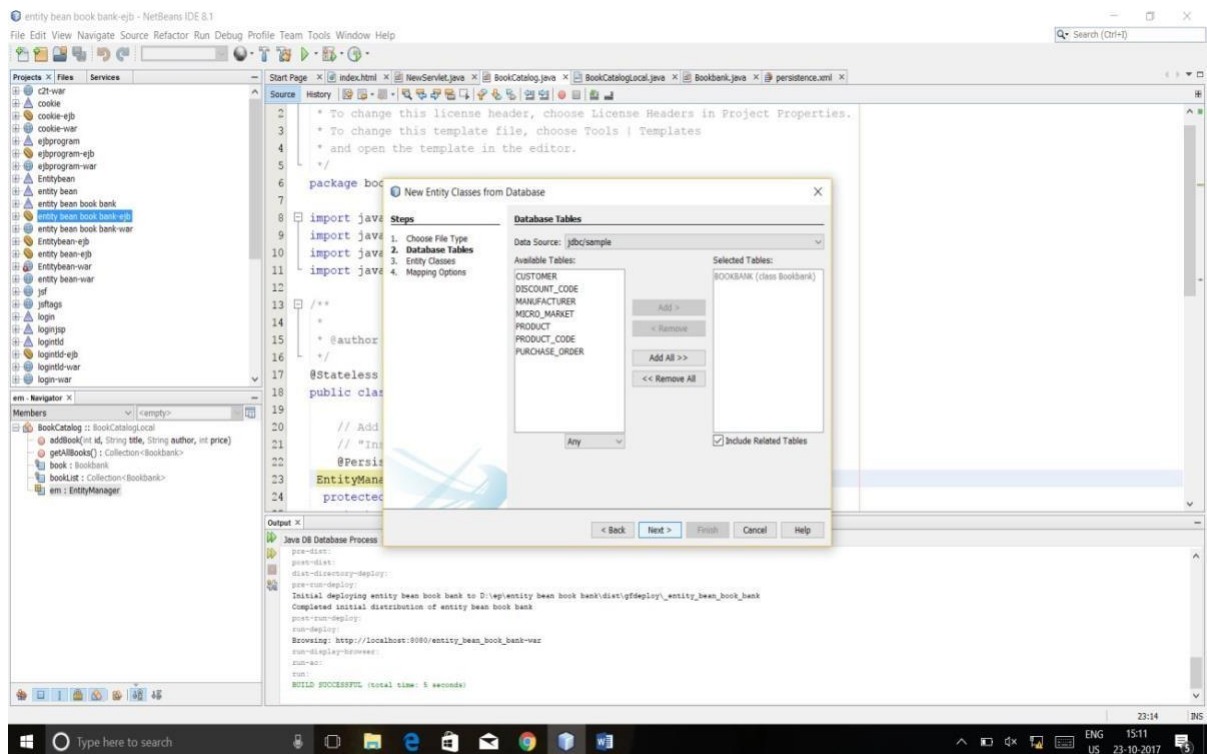
1. Create a JavaEE application.
2. Create a table to store the values of books in sample database.
3. Right click on ejb and choose New-> Entity classes from Database.



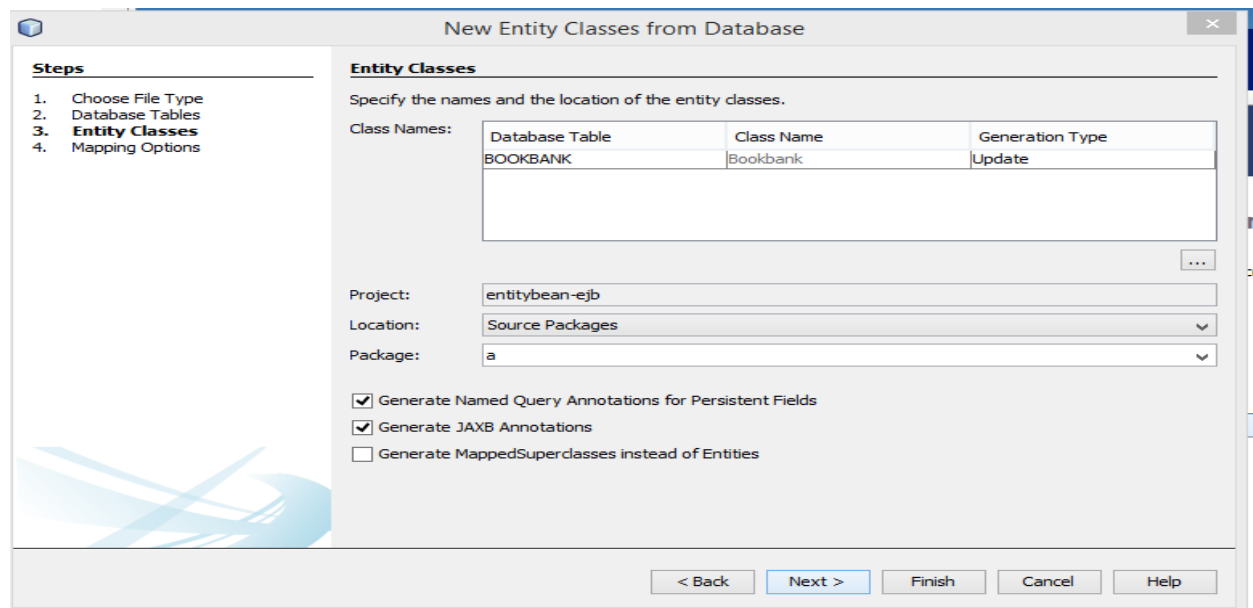
4. Select data source as jdbc/sample.



5. Select the table and click Add.



6. Click Next.

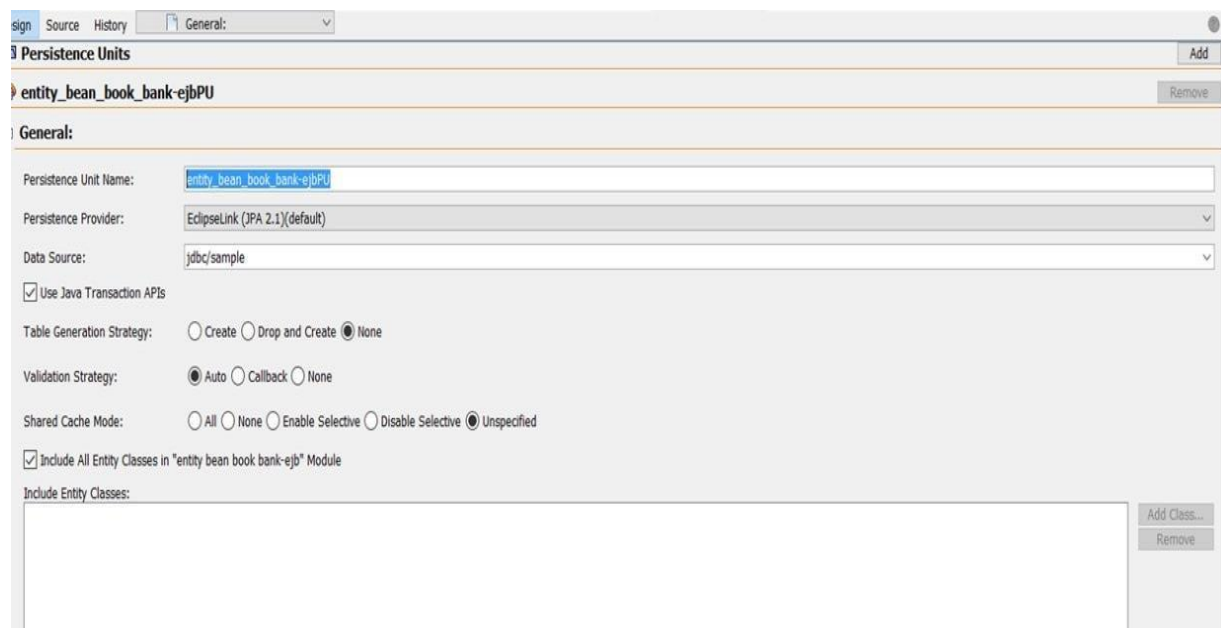


7. Click Finish.

8. This automatically creates a entity bean file for the selected database to store the values.

To get the value of persistence unit:

Expand your ejb application and select web services and expand it select persistence.xml file and open it.



## **Index.html**

```
<html>

  <head>

    <title>TODO supply a title</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

  </head>

  <body>

    <form action="NewServlet" method="POST">

      <p> Enter the id: <input type="text" name="t0"/></p><br/>

      <p> Enter the book title: <input type="text" name="t1"/></p><br/>

      <p> Enter the book author: <input type="text" name="t2"/></p><br/>

      <p> Enter the price: <input type="text" name="t3"/></p><br/>

      <input type="submit" value="submit"/>

      <input type="button" value="reset"/>

    </form>

  </body>

</html>
```

## **NewServlet.java**

```
package book;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Collection;

import java.util.Iterator;

import javax.ejb.EJB;
```



```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class NewServlet extends HttpServlet {

    @EJB

    private BookCatalogLocal bci;

    /**
     * Processes requests for both HTTP GET and POST
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        int id=Integer.parseInt(request.getParameter("t0"));
        String title=request.getParameter("t1");
        String author=request.getParameter("t2");
        int price=Integer.parseInt(request.getParameter("t3"));

        try (PrintWriter out = response.getWriter()) {

```

```

    /* TODO output your page here. You may use following sample code. */

    out.println("<!DOCTYPE html>");

    out.println("<html>");

    out.println("<head>");

    out.println("<title>Servlet NewServlet</title>");

    out.println("</head>");

    out.println("<body>");

        if ( title != null && author != null) {

bci.addBook(id,title,author,price);

        }

        out.println("<p><b>Record added</b></p>");

        Collection list=(Collection)bci.getAllBooks();

for (Iterator iter = list.iterator(); iter.hasNext();){

    Bookbank element = (Bookbank) iter.next();

    out.println(" <p>Book ID: <b>" + element.getId() + "</b></p>");

    out.println(" <p>Title: <b>" + element.getTitle() + "</b></p>");

    out.println(" <p>Author: <b>" + element.getAuthor() + "</b></p>");

    out.println(" <p>Price: <b>" + element.getPrice() + "</b></p>");

}

        //out.println("<h1>Servlet NewServlet at " + request.getContextPath() + "</h1>");

        out.println("</body>");

        out.println("</html>");

    }

```

```
}
```

```
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
```

```
/**
```

```
 * Handles the HTTP <code>GET</code> method.
```

```
 *
```

```
 * @param request servlet request
```

```
 * @param response servlet response
```

```
 * @throws ServletException if a servlet-specific error occurs
```

```
 * @throws IOException if an I/O error occurs
```

```
 */
```

```
@Override
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
    throws ServletException, IOException {
```

```
    processRequest(request, response);
```

```
}
```

```
/**
```

```
 * Handles the HTTP <code>POST</code> method.
```

```
 *
```

```
 * @param request servlet request
```

```
 * @param response servlet response
```

```
 * @throws ServletException if a servlet-specific error occurs
```

```
 * @throws IOException if an I/O error occurs
```

```

    */

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

        processRequest(request, response);

    }

    /**

    * Returns a short description of the servlet.

    *

    * @return a String containing servlet description

    */

    @Override

    public String getServletInfo() {

        return "Short description";

    } // </editor-fold>

}

```

### **BookCatalogLocal.java**

```

package book;

import java.util.Collection;

import javax.ejb.Local;

@Local

public interface BookCatalogLocal {

    public Collection <Bookbank> getAllBooks();

}

```

```
    public void addBook(int id, String title, String author, int price);  
}
```

### **BookCatalog.java**

```
package book;  
  
import java.util.Collection;  
  
import javax.ejb.Stateless;  
  
import javax.persistence.EntityManager;  
  
import javax.persistence.PersistenceContext;  
  
@Stateless  
  
public class BookCatalog implements BookCatalogLocal  
{  
    // Add business logic below. (Right-click in editor and choose  
    // "Insert Code > Add Business Method")  
  
    @PersistenceContext(unitName="entity_bean_book_bank-ejbPU")  
    EntityManager em;  
  
    protected Bookbank book;  
  
    protected Collection <Bookbank> bookList;  
  
    @Override  
  
    public void addBook(int id,String title, String author, int price) {  
        // Initialize the form  
  
        if (book == null)  
  
            book = new Bookbank(id,title, author, price);  
  
        em.persist(book);  
    }  
}
```

```

    }

    @Override

    public Collection <Bookbank>getAllBooks() {

        //System. out.println("u");

        bookList=em.createQuery("select b from Bookbank b").getResultList();

        return bookList;

    }

}

```

### **Bookbank.java**

```

package book;

import java.io.Serializable;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Size;

import javax.xml.bind.annotation.XmlRootElement;

@Entity

@Table(name = "BOOKBANK")

@XmlRootElement

@NamedQueries({

```

```

    @NamedQuery(name = "Bookbank.findAll", query = "SELECT b FROM Bookbank b"),

    @NamedQuery(name = "Bookbank.findById", query = "SELECT b FROM Bookbank b
WHERE b.id = :id"),

    @NamedQuery(name = "Bookbank.findByTitle", query = "SELECT b FROM Bookbank b
WHERE b.title = :title"),

    @NamedQuery(name = "Bookbank.findByAuthor", query = "SELECT b FROM Bookbank b
WHERE b.author = :author"),

    @NamedQuery(name = "Bookbank.findByPrice", query = "SELECT b FROM Bookbank b
WHERE b.price = :price"))))

public class Bookbank implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id

    @Basic(optional = false)

    @NotNull

    @Column(name = "ID")

    private Integer id;

    @Basic(optional = false)

    @NotNull

    @Size(min = 1, max = 30)

    @Column(name = "TITLE")

    private String title;

    @Basic(optional = false)

    @NotNull

    @Size(min = 1, max = 30)

    @Column(name = "AUTHOR")

    private String author;

    @Basic(optional = false)

```

```
@NotNull

@Column(name = "PRICE")

private int price;

public Bookbank() {

}

public Bookbank(Integer id) {

    this.id = id;

}

public Bookbank(Integer id, String title, String author, int price) {

    this.id = id;

    this.title = title;

    this.author = author;

    this.price = price;

}

public Integer getId() {

    return id;

}

public void setId(Integer id) {

    this.id = id;

}

public String getTitle() {

    return title;

}

public void setTitle(String title) {

    this.title = title;
```



```

    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    @Override
    public int hashCode() {
        int hash = 0;

        hash += (id != null ? id.hashCode() : 0);

        return hash;
    }

    @Override
    public boolean equals(Object object) {

        // TODO: Warning - this method won't work in the case the id fields are not set

        if (!(object instanceof Bookbank)) {
            return false;
        }

```

```

Bookbank other = (Bookbank) object;

if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {

    return false;

}

return true;

}

@Override

public String toString() {

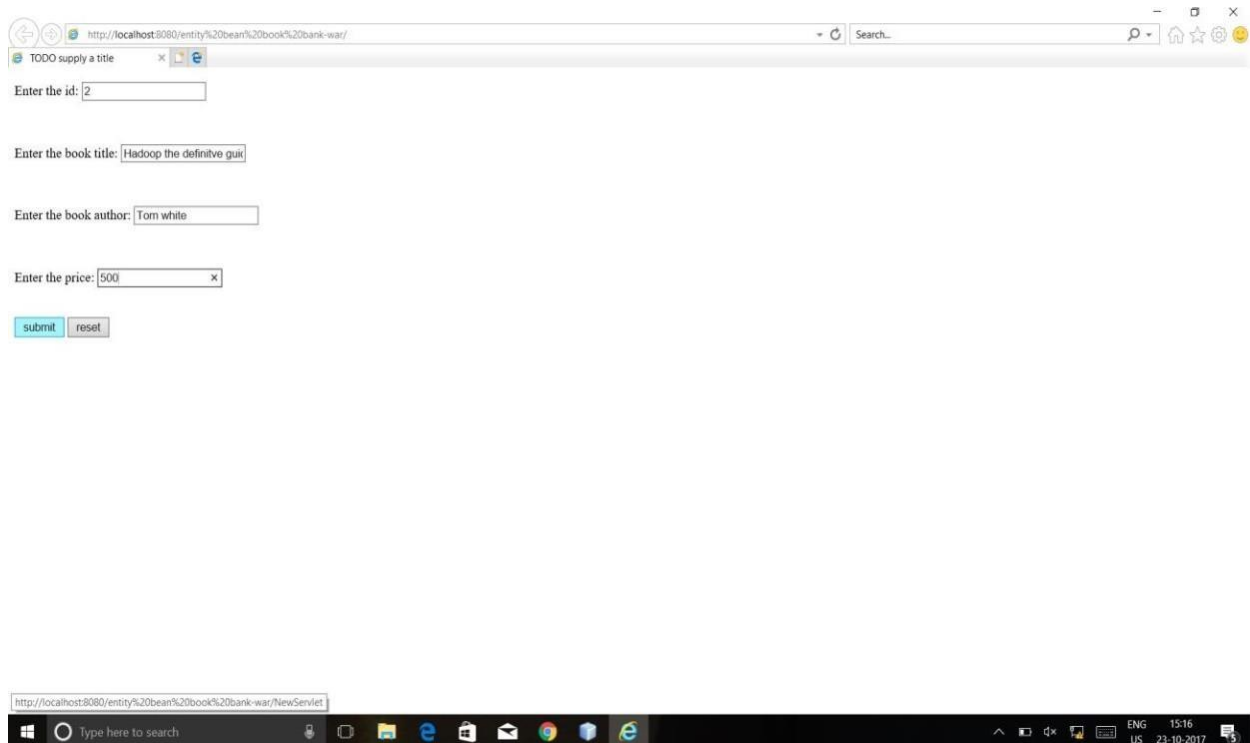
    return "book.Bookbank[ id=" + id + " ]";

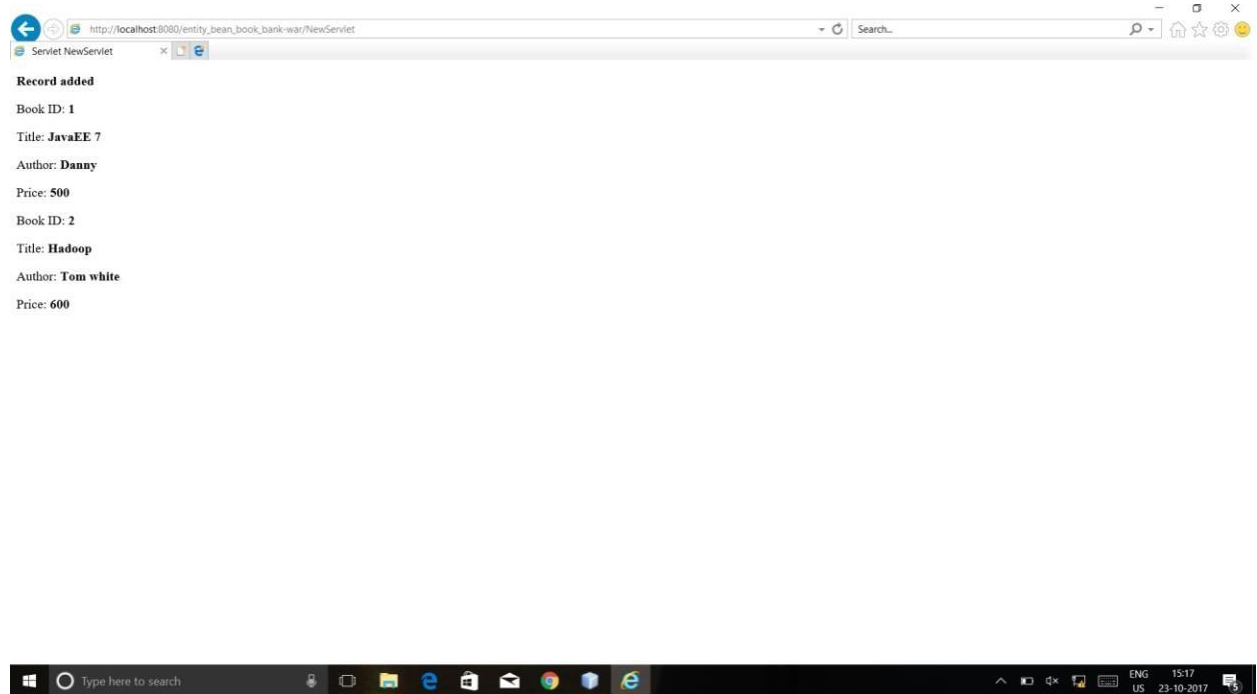
}

}

```

## **OUTPUT:**





## 8. Write a program to demonstrate Custom Tag

### Index.html:

```
<html>
  <head>
    <title>Date</title>
  </head>
  <body>
    <form action="Date.jsp" method="get">
      To Display current date and time<br>
      <input type="submit" value="on click"/>
    </form>
  </body>
</html>
```

### Date.jsp

```
<%--
  Document   : newjsp
  Created on : 24 Jun, 2023, 10:43:57 AM
  Author      : leemo
--%>
<% @ taglib prefix="m" uri="/WEB-INF/tlds/newtag_library" %>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<html><head>
<title>JSP Page</title>
</head><body>
Current date and time is:<m:today/>
</body></html>
```

### NewTagHandler.java

```
package bec;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;
import java.util.Calendar;
```

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

```

```

/**
 *
 * @author leemo
 */

```

```

public class NewTagHandler extends TagSupport{

```

```

    /**
     *
     * @throws JspException
     */
    @Override
    public int doStartTag() throws JspException
    {
        JspWriter out=pageContext.getOut();
        try{
            out.print(Calendar.getInstance().getTime());
        }
        catch(Exception e){
            System.out.println(e);
        }
        return SKIP_BODY;
    }
}

```

### **Tld file:**

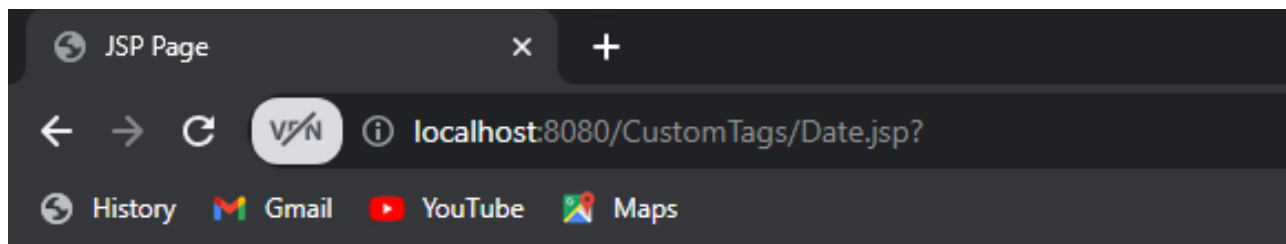
```

<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd">
    <tlib-version>1.0</tlib-version>
    <short-name>newtag_library</short-name>
    <uri>/WEB-INF/tlds/newtag_library</uri>
    <tag>
        <name>today</name>
    </tag>
</taglib>

```

```
<tag-class>bec.NewTagHandler</tag-class>
<body-content>scriptless</body-content>
</tag>
<tag>
  <name>NewTagHandler</name>
  <tag-class>bec.NewTagHandler</tag-class>
  <body-content>scriptless</body-content>
</tag>
</taglib>
```

## OUTPUT:



Current date and time is: Fri Aug 04 15:08:18 IST 2023

---