Contents

Context

This document contains assignments to be completed as part of the hands on for the subject **Presentation tier using JSP and JSF** (Course code: ED113A).

Note: In order to complete the course, assignments in this document must be completed in the sequence mentioned.

Day 2 Assignments

Assignment 1: Understanding declaration and scripting elements

Objective: To create a JSP page and understand the declaration, scripting element, expression and JSP life cycle methods.

Problem Description: Declare instance variables, local variable, methods in a JSP page and understand the lifetime of instance variable in a JSP.

Estimated time: 10 Min.

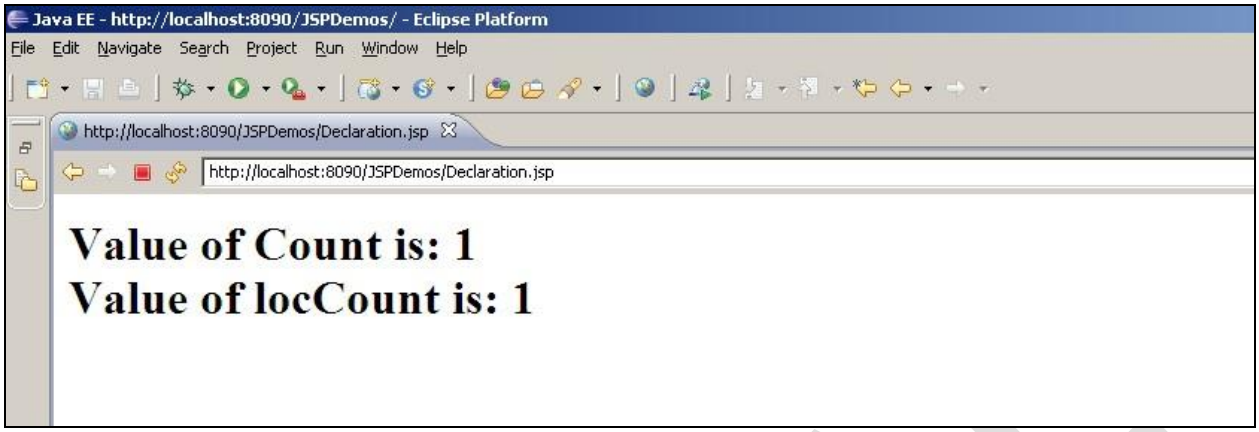
Step 1: Create a new **Dynamic Web Project** and name it **JSPDemos**. Right click on the Project select **New -> JSP** (if JSP option is not visible on the menu then select **Other**, it will open a new window. Locate the Web folder and expand it. Select JSP from the list). Name it **Declaration.jsp** and type the following code inside the **<body>** tag of created JSP.

```
// declaring a instance variable: This variable becomes class variable
<%!


/* Below method is one of the class methods, it is "private", hence it can be
accessed by the same JSP scriptlets or other methods in the same JSP*/
    int count=0;



    private int getCount(){
          count++;
     return count;
}
/* Following are LifeCycle methods of JSP init method will be called only once
in JSP life cycle*/        public void jspInit(){
      count=0;
      System.out.println("Inside init method" + count);
}
  //Destroy method will be called only once in JSP life cycle      public
void jspDestroy(){
      System.out.println("Inside Destroy() method"+ count);
}
%>
/*The following is scriptlet, and all the lines in scriptlet, //eventually get
the place in _jspService() //method. HENCE YOU CAN NOT DECLA     RE A METHOD
IN THE SCRIPTLET. All the variables declared in scriptlet become local
variables of

the _jspService() method.*/


<%

     // declaring local variable

     int locCount=0;
     locCount++;
     System.out.println("Inside Service method");
%>
        <h1>
```

```
                                        <B> Value of Count is: <%=
getCount() %><br>
              Value of locCount is: <%= locCount %><br> </B>
     </h1>
```

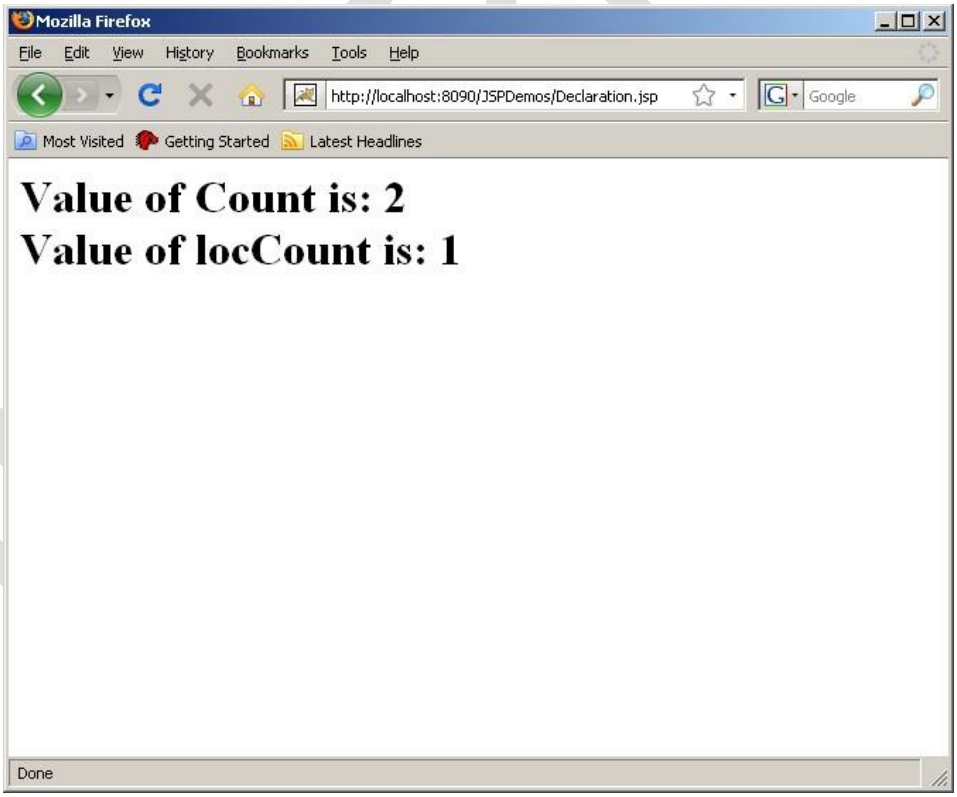

Step 2:

1. Run the JSP using JBoss Server (if it is not added, add it in the Server tab: **Refer Note** in **Step3 of Assignment No. 2**) 2. You will get following output:



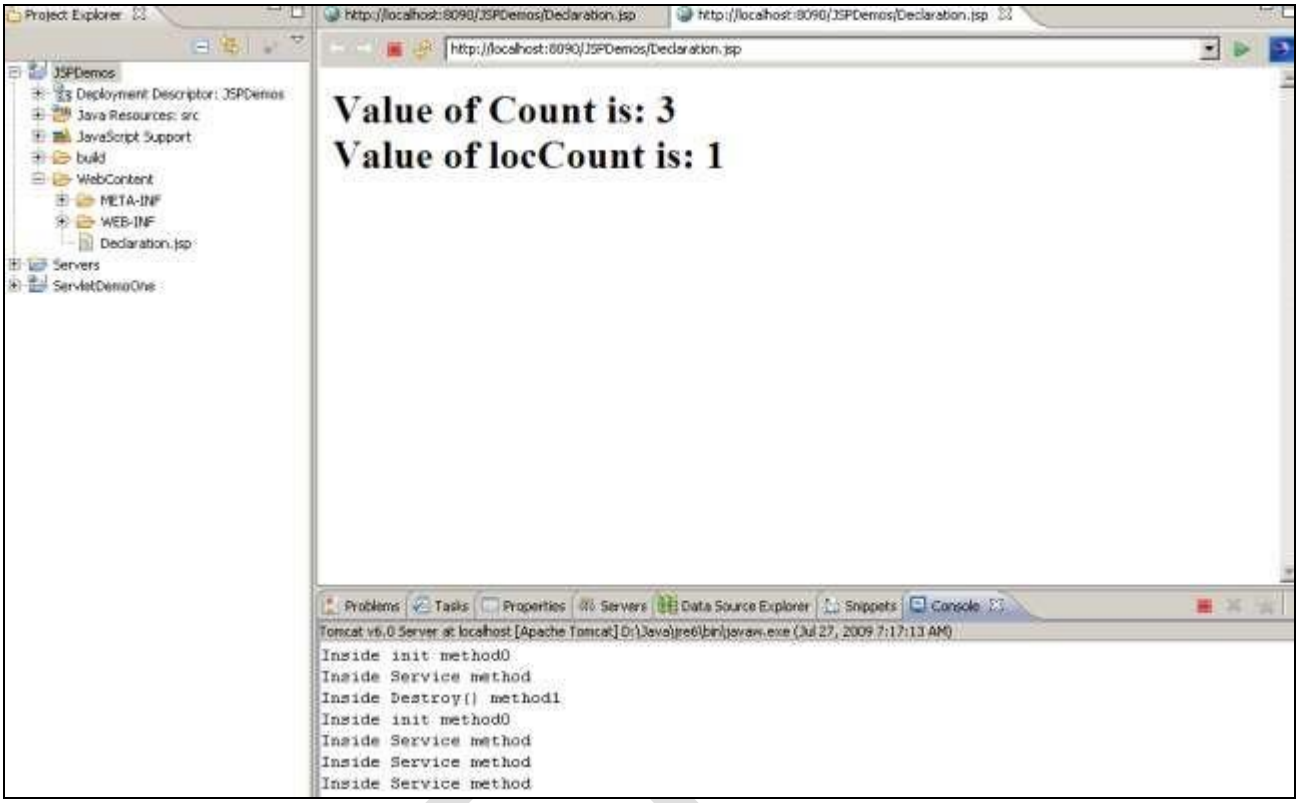
3. Analyze the output.
4. Check the console (Go to console view) for verifying the output.
   (System.out.println() prints on the console window, and not on the browser).

Step 4: Now close this browser window and open a new browser window. Type the same URL. What is the output now? Compare and reason the output?



If you want to execute again, No need to perform, **Run AS-> Run On Server** again, you can directly access the application using a web browser (**provided, you have not made any changes to the files,**

and your application is deployed and server is running)… Analyze the output as well as console window result.



Note: In the JSP life cycle the container creates only one instance of the generated Servlet. So there will be only one instance of the ser vlet and one copy of instance variable.
The code inside <% %> is placed in the _jspService() method of the generated servlet and this will be invoked for each request from the client.

If 10 users are accessing the JSP, all the 10 users are accessing (sharing) the same instance variable.

Summary of this assignment:

You have learnt,

- to declare instance variables
- to define methods
- to declare local variables
- Using expression element to display the data
- JSP life cycle methods

Assignment 2 : JSP compile time error and runtime exception

Objective: To understand the JSP compile time error and runtime exception.

Problem Description: Test and analyze a JSP page having compile time error and runtime exception.

Estimated time: 5 Min.

Step 1: Create a JSP page "**ErrorDemo.jsp**" and insert the below code

```
<html>
<body>
      <% String name; %>
      <h1> Name is <%= name %>
</body>
</html>
```

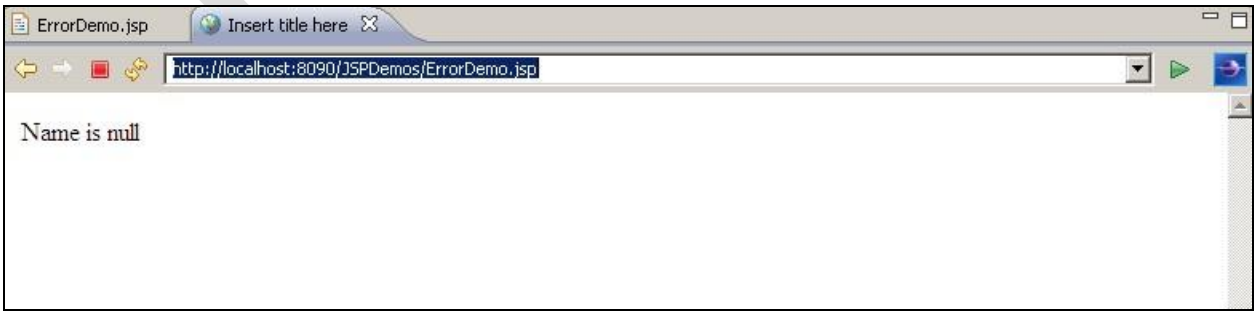
Step 2: In the Editor window of Eclipse, you would see a red mark on the line having the **compilation error**, mouse over on the red mark to read the message. (Message Displayed: **"The local variable name may not have been initialized.")**

Step 2: Don't correct the error and try to run the above JSP. You will get the following compilation error as you are forcing the Web container to compile the JAVA file(created form the JSP) into a JAVA class file.

Step 3: Verify the error, this is compile time. Now, modify the code to remove the compilation error as mentioned below.

```
<html>
<body>
     <% String name=null;%>
     <h1> Name is <%= name %>
</body>
</html>
```

Step 4: Run the JSP and verify the output.



Step 5: Verify the output. Now, modify the code as shown below and run the modified JSP.

```
<html>
<body>
      <% String name=null;%>
      <h1> Name is <%= name.toString() %>
</body>
</html>
```

What is output?

It results in a **Runtime Exception** as you are trying to invoke a method on a null reference variable (reference variable not referring to any object). So a **NullPointerException** is thrown by the container.

Summary of this assignment:

You have learnt,

- The difference between compile time error and runtime exception

Assignment 3: Accessing HTTP request parameters in JSP using request implicit object

Objective: To access HTTP request parameters using JSP request implicit object.

Problem Description: Create an html form to accept the login details (username and password) from the user and on submitting the details retrieve the user submitted details in **Login.jsp.** Based on the username (role) retrieved display appropriate links.

Theory: We saw in Assignment No. 1 that we had to write a lot of HTML code in a Java class, LoginServlet, which is difficult to read and maintain. Instead, JSP's can be used to do the job where bits of java code can be embedded with html markup using scriptlets.

Estimated time: 10 Min.

Step 1: Create a new dynamic web project, **LoginJSPDemo**. Copy **loginForm.html** created in **Assignment 1, Day 1** to the **WebContent** folder of this project.

Step 2: Create a JSP page **login.jsp** with the below code

login.jsp

```
<html>
<head><title>Login Page</title></head>
<body><br><br><br><h2>
<%
     String uname = request.getParameter("username");
if(uname.equalsIgnoreCase("admin")){

%>
          Hi <%=uname %>!!!<br><br>
          <a href="addAccount.html">Add Account</a><br>
```

```
<%
	}

	else if(uname.equalsIgnoreCase("staff")){
%>
		Hi <%=uname %>!!!<br><br>
		<a href="displayAccount.html">Display Account</a>
<%
	}
	else if(uname.equalsIgnoreCase("customer")){
%>
		Hi <%=uname %>!!!<br><br>
	<a href="modifyAccount.html">Modify Account</a><br>
```

Step 3:

```
<a href="displayAccount.html">Display Account</a>
```

In Day1 -Assignment No. 1, we had a loginForm.html, create a same file in this web application too.

```
<%
	} else{
%>
Invalid Login, Guest!!!
```

Step 3: You need to make a small change in the loginForm.html. Now the **action** attribute of the **form** element, has to invoke **login.jsp**, instead of the Servlet.

So, change it as below:

```
<form action= "login.jsp" >
</h2>
…
</body></html>
```

```
		<a href="displayAccount.html">Display Account</a> </form>
```

Step 4: Deploy the application. (*Refer Step 6 of Assignment No. 2 of Day1)* Check the URL and if you are not able to see any output, change the URL as, **http://localhost:8080/LoginJSPDemo/loginForm.html**

Step 5: Open another web browser window, say Mozilla Firefox, and type **http://localhost:8080/LoginJSPDemo/loginForm.html**
Check the output.

Summary:
Now we have written the business logic (java code) in the JSP using **<% %>, scriptlets.** We used **request.getParameter() to get the user entered details in the login form. request** is

an implicit object **in JSP** and **getParameter()** is a method on it, which accepts a String argument (name of the html request parameter).
Note: request.getParameter() always returns the String value of the parameter.

We used **expression element, <%= %>** to output data to the client (browser)**.**
The contents of all the scriptlets will be going in the _jspService() method of the converted Servlet code for this JSP.
The variable "**uname**" declared in the scriptlet will be different for every request, as it is a local variable of the **_jspService()** method, which will be invoked separately for each request.

You have just learnt

- To fetch request parameters in the JSP using implicit object **request**
- To use scriptlets and expression elements in JSP

Note: In the above example the parameters are passed to the server using POST method. Change it to GET and observe the URL. The extra path information in the URL is called the **query string**. Whenever, you need to pass sensitive information to server, you use POST method.

Note: Each time, you make change to the JSP, it is required to be deployed in the App Server, otherwise, the older version of equivalent Servlet class will be executed. Re-Deploy the application and check it again.

Assignment 4: Working with HTML form and request parameters

Objective: To create an HTML form and send the data to the server for processing

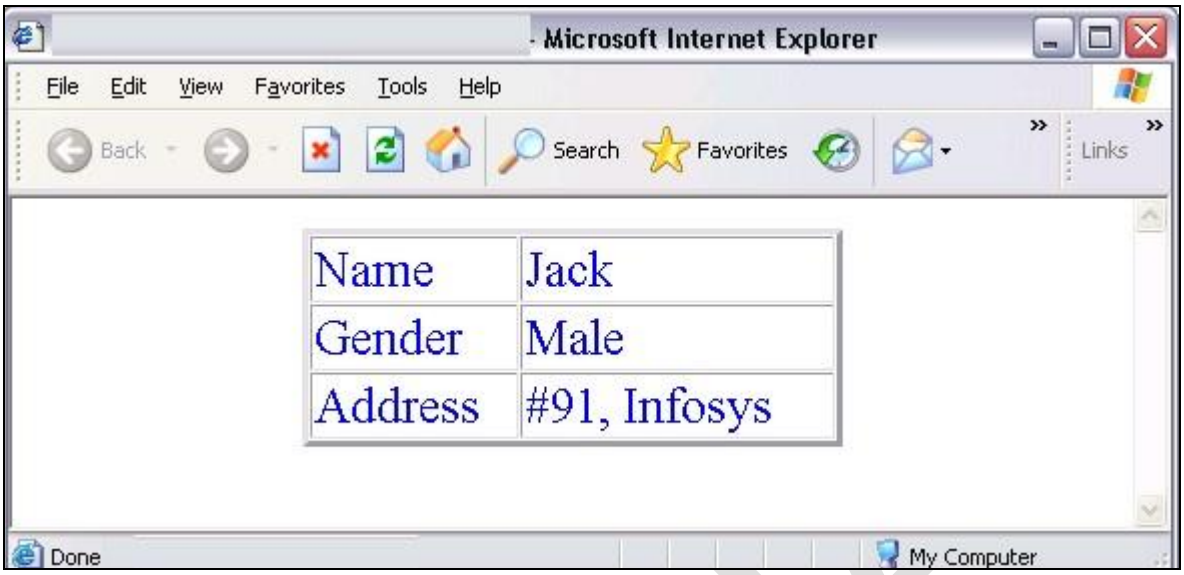
Problem Description:

(Create a new project, "JSPDemos" for this assignment)

1) Create HTML page to accept the following fields (**input.html**)

Name : (Text box)
Gender : (Radio Buttons for Male and Female)
Address : (Textarea)

1) When the user clicks on the **input.html** it should invoke a JSP called **FormDisplay.jsp**. This **FormDisplay.jsp** should display the input in a HTML table format. The expected output is displayed below.



Estimated time: 15 Min.

Assignment 5: Understanding session object and session variable

Objective: To understand session variable and its lifetime.

Problem Description: Use implicit object **session** to set one integer value as session attribute, named "**oneAttribute**". Display its contents.

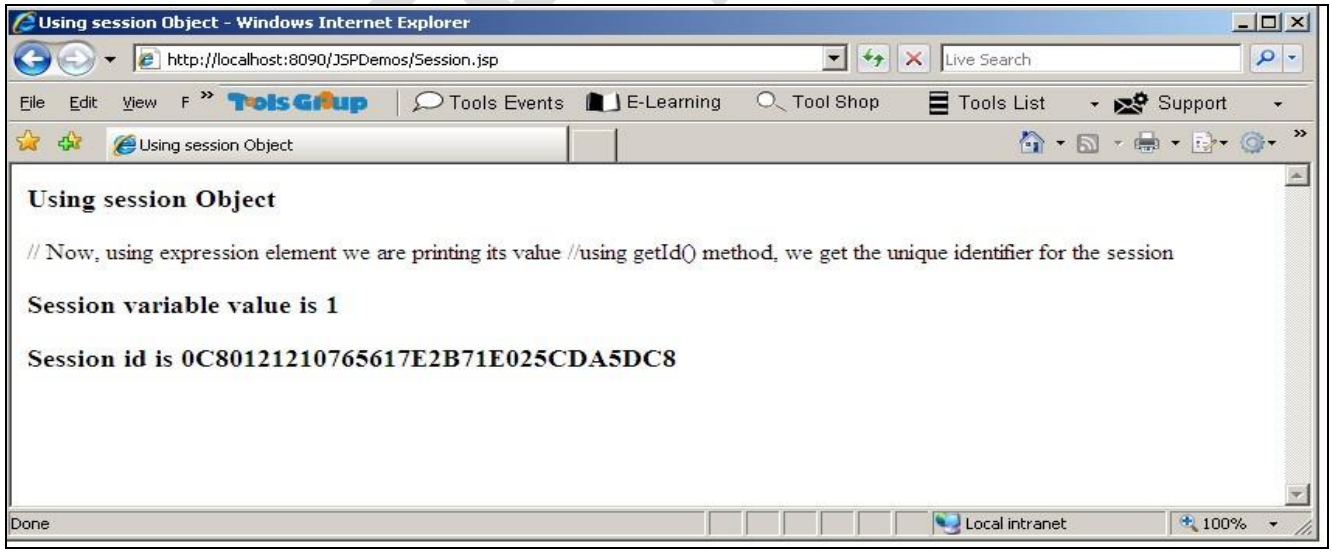
Estimated time: 20 Min.

Step 1: Create a JSP page, **Session.jsp** (use "**JSPDemos**" project created for the previous assignment) and type below mentioned code:

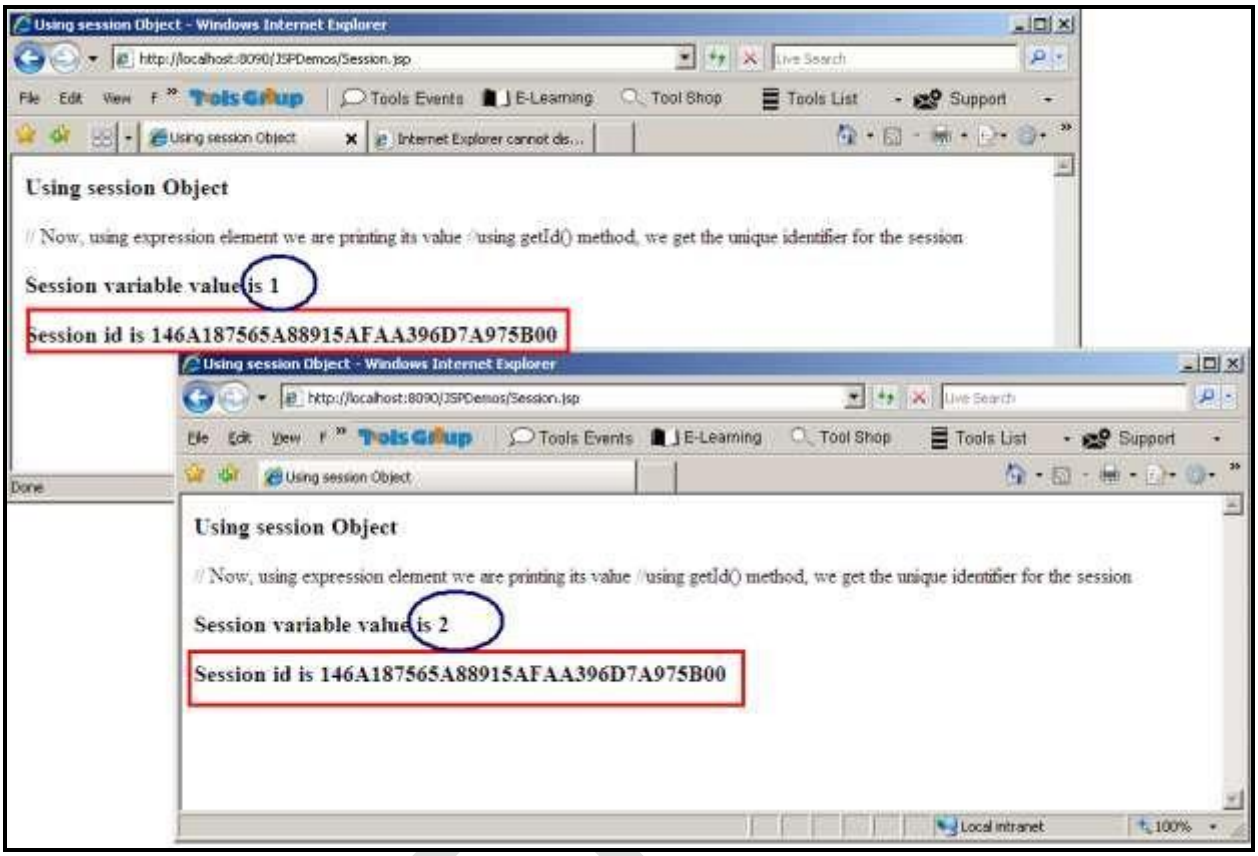
```
<HTML>
    <HEAD>
        <TITLE>Using session Object</TITLE>
    </HEAD>

    <BODY>
        <H3>Using session Object</H3>
<%
        Integer data = (Integer) session.getAttribute("oneAttribute");
     // Have we set, any value in "oneAttribute"?
      //No, then what will we get for the first time we are trying to
      //retrieve value from this object?
      //Well, we will get null…
      // so, let us check,
```

```
      if (null == data){
            data = new Integer(1);
         }
else{
      //otherwise if, it is not null (means, this JSP is accessed
      //again in the same session… we will assign it with the (previous
      //value + 1)              data = new
Integer(data.intValue() + 1);
         }

      // Now, for later use in the same session, let us set the
      //session attribute, oneAttribute
session.setAttribute("oneAttribute", data);
%>
      // Now, using expression element we are printing its value
      //using getId() method, we get the unique identifier for the session
      <h3>Session variable value is <%= data.intValue() %></h3>
      <h3>Session id is <%= session.getId() %></h3>
</BODY>
</HTML>
```

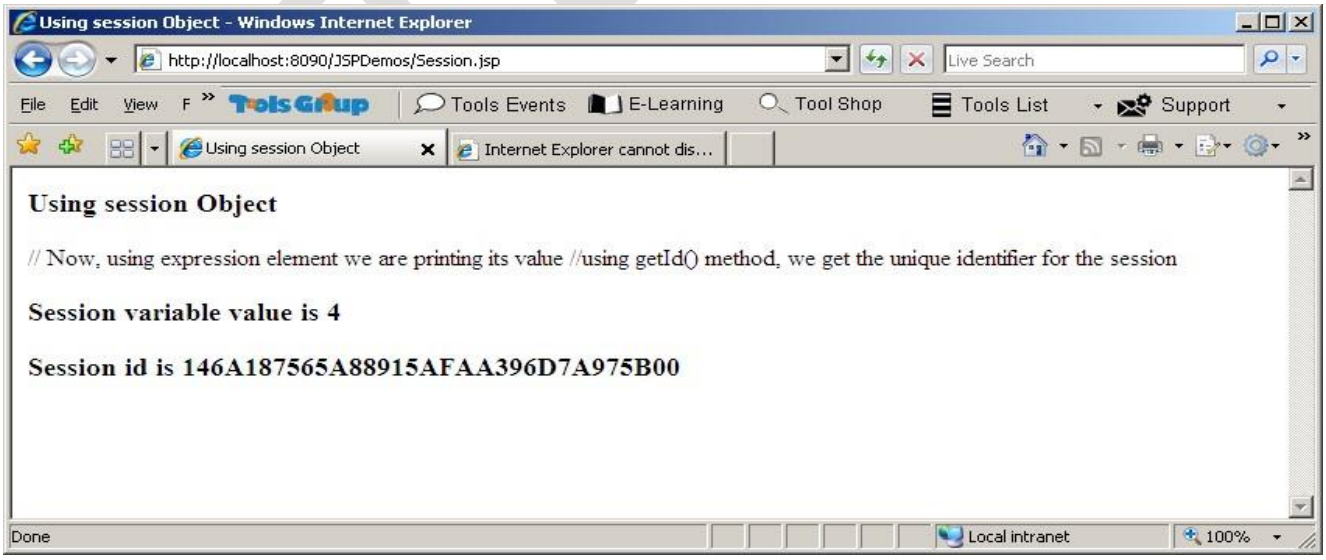
Step 2: Redeploy the application. Open a new browser window and access this page by typing proper URL in the browser address bar. You should see output as shown below (sessionId will be different for your session)



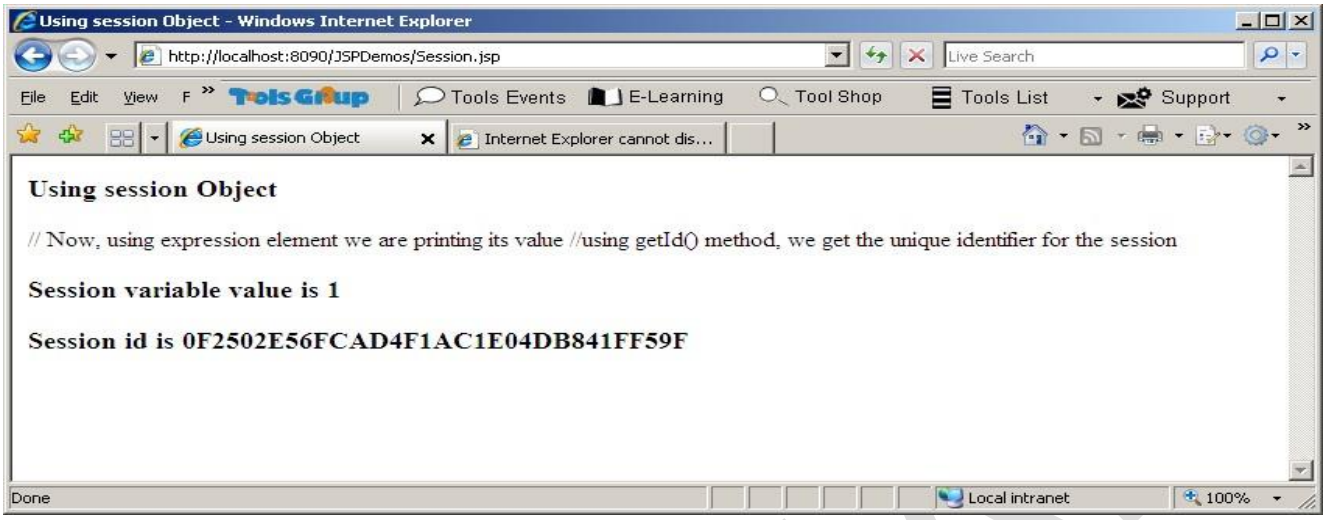
Step 3: Open a new browser window from the existing window. (From the browser select **file menu -> new -> window).** Verify the output and note that the session id is same for both the browsers.



Step 4: Close the new window (opened using step 3) and refresh (2 times) the first window (opened using step 2). Check the output.



Step 5: Open a new browser not from the existing browser. (From start-> IE or from the IE shortcut). Verify the output and note that the session id is different for the new browser.

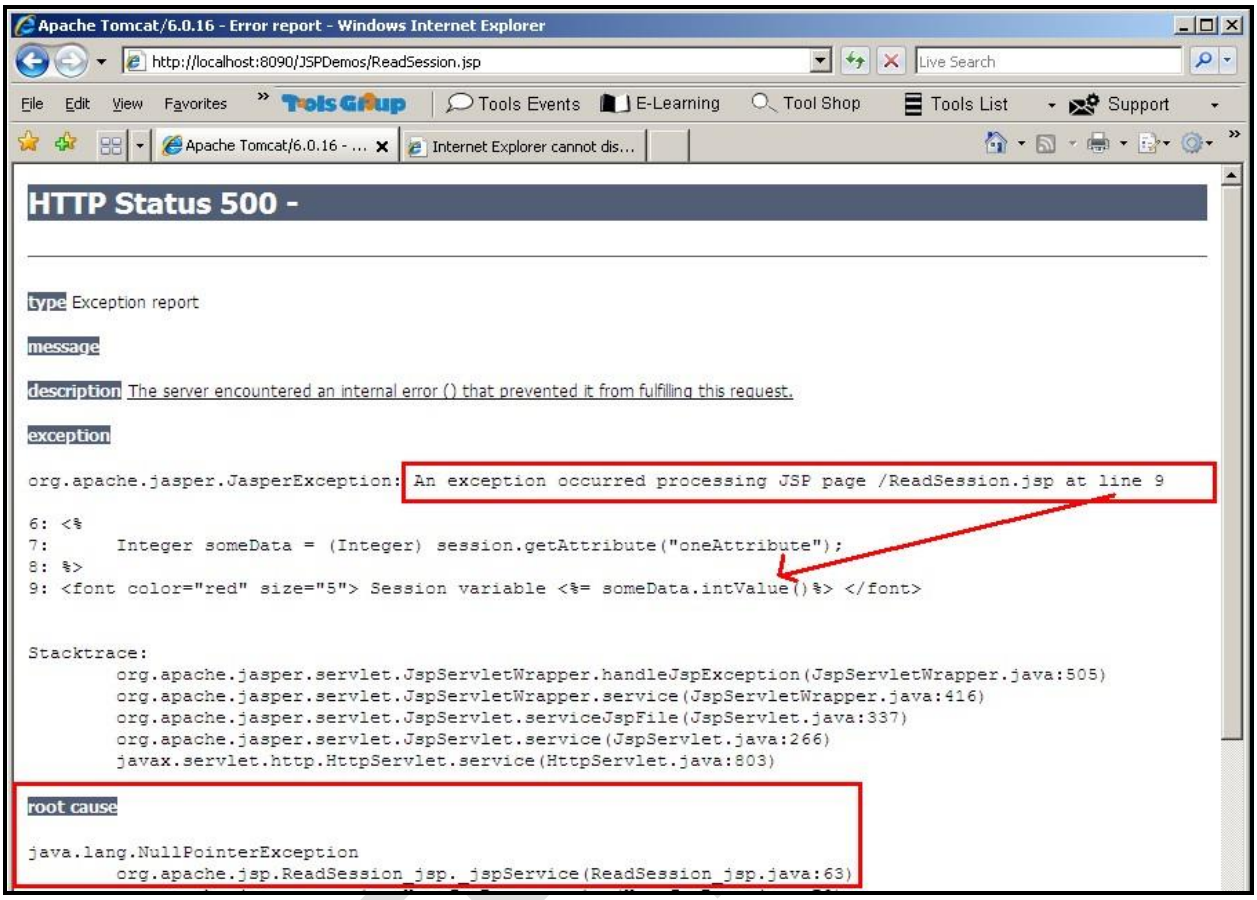


Note: Session object is created for each client and stored in the server when the client accesses the application for the first time. Session variables are accessible from any JSP page for that particular session (client or browser). Each user has his/her copy of session object and session variables/attributes on the server memory, till that session is active.
Once you close the browser window, or you click on logout button, or session is timed out, the session object is made eligible for garbage collection.
When you are opening a browser from existing browser (using file -> new ->window or window.open() method) the session object is shared between multiple browser windows(however, it depends on the Browsers and versions, Internet Explorer treats it as same session). If you are opening a new browser window then the sessionId is not shared with the new browser instance and hence when a resource is requested from that browser window, the server creates a new session object (as the client request will not have the sessionId information).

Step 6: Create a JSP, **ReadSession.jsp** and type below mentioned code:

```
<HTML>
    <HEAD>
        <TITLE>Read Session</TITLE>
    </HEAD>
<BODY>
<%
    Integer someData = (Integer) session.getAttribute("oneAttribute"); %>
<font color="red" size="5"> Session variable <%= someData.intValue()%>
</font>
```

Step 7: Close all the browsers and open **ReadSession.jsp** in a new window and verify the output.



Step 8: Read the error details. The error is generated because of **NullPointerException**.

Step 9: Verify why the session variable is null.

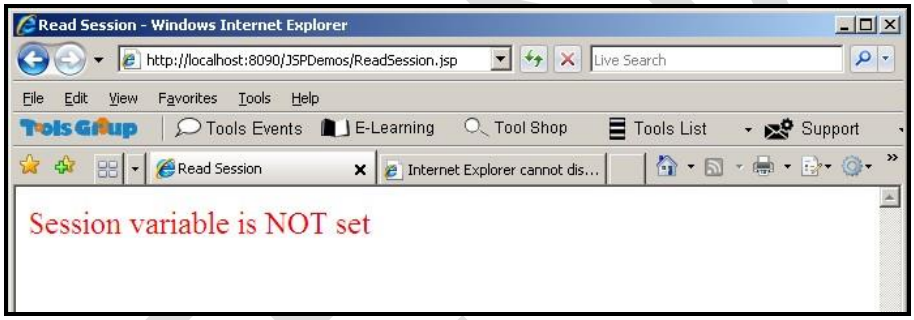
Note: We opened **ReadSession.jsp** directly without setting the session variable. Because of this, the session variable is null**. It is always better to first check whether the object is created/exists; by checking it against null, before using it. This is applicable to all JSP implicit objects (request, session, application)**

Step 10: Modify the code as mentioned below.

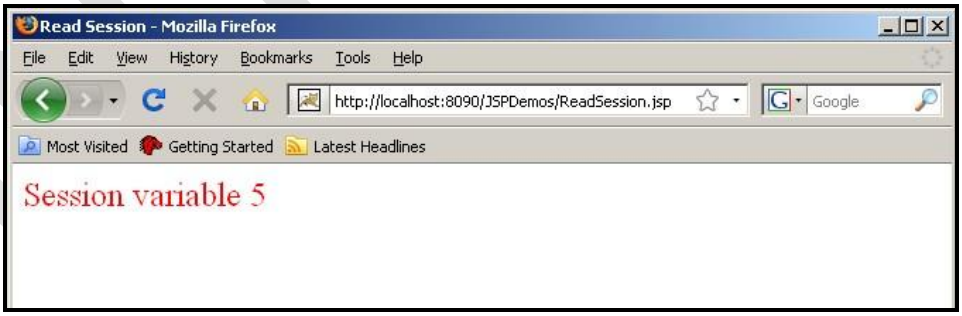
```
<HTML>
    <HEAD>
        <TITLE>Read Session</TITLE>
    </HEAD>
<BODY>
<%
```

```
    Integer someData = (Integer) session.getAttribute("oneAttribute");
if (null ==someData){
%>
   <font color="red" size="5">Session variable is NOT set</font>
<%
        } else {
%>
<font color="red" size="5">
     Session variable <%= someData.intValue()%>
</font>
<%
        }
%>
```

Step11: Close all the browsers and open the **ReadSession.jsp** in a new browser. Check the output.



Step 12: Open **Session.jsp in a new** browser window and refresh the page for 5 times. Without closing the browser open **ReadSession.jsp.** Check the output and analyze it.



Summary of this assignment:

You have learnt,

- to use session object
- to create session variables
- to read session variables
- lifetime of session object

Session is an implicit object of type HttpSession. This class has many other methods, which you can refer it from the documentation. We discussed, getId(), which returns the unique session id, allocated for each session. Some more useful methods are discussed on **Day2 slide no 22, and notes page.** The setAttribute() and getAttribute() methods are discussed on Slide no. 19 of Day2.

Assignment 6: JSP Standard Actions

Objective: To understand the use of <jsp:forward> and <jsp:include> standard action elements/tags.

Problem Description: In the same web application (JSPDemos), we are going to create inputForm.html, One.jsp and Two.jsp.
inputForm.html accepts **username** and submits the form to **One.jsp**
One.jsp verifies whether data is entered for request parameter, **username.** For displaying Welcome message, it forwards the work to Two.jsp

Theory: When a JSP wants to delegate / share its work with another JSP, it forwards the request to another JSP instead of generating and sending back the response itself. While doing so, end user (client) is not aware of this forward; **as the URL is not changed**.

Estimated time: 20 Min

Step 1: Create an HTML file, "**inputForm.html**". Type the following code in it and save it.

```
<html>
      <head><title> Input Page </title></head>
      <body>
            <form name="inputForm" action="One.jsp">
            <table width="50%" align="center" border="1">
             <tr bgcolor="lightgrey">
                  <td align="center" colspan="2"> <b>Login Form</b></td>
             </tr>
             <tr>
                  <td> User Name </td>
                  <td> <input type="text" name="username"> </td>
             </tr>
            <tr>
            <td colspan="2">
                  <input type="Submit" value="Submit Data"> </td>
```

```
            </tr>
            </table>
            </form>
      </body>
</html>
```

Step 2: Create a JSP page, as "**One.jsp**". Type the following code in it and save it.

```
<html>
<head>
      <title>One JSP</title>
</head>
<body>
<%
      String uname = request.getParameter("username");
      if(uname== null || uname.length()==0){
%>
            <h2 align="center" style="background-color:red">
            Mandatory field UserName missing...
            </h2><br><br><hr>
            <jsp:include page="inputForm.html" />
            <a href="inputForm.html"> Click to go back to input page</a> <%
      }
      else{
%>
            <jsp:forward page="Two.jsp">
            <jsp:param value="This string is set in the One.jsp"
            name="extraParameter"/>
            </jsp:forward>
<%
      }
%>
</body>
</html>
```

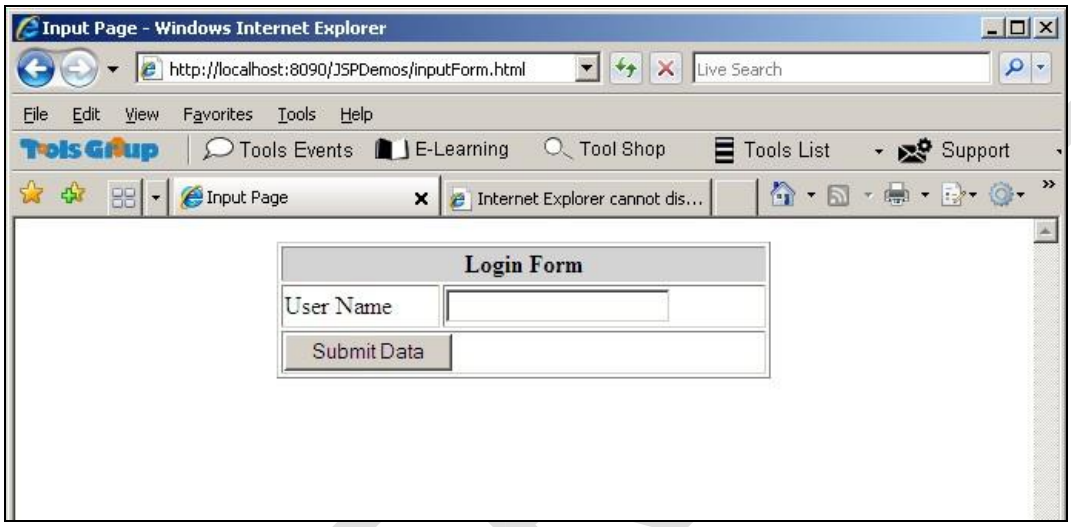
Step 3: Create a JSP file, as "**Two.jsp**". Type the following code in it and save it.

```
<html>
<head><title>Two JSP</title></head>
<body>
      <h1>This output is displayed by Two.jsp, but check the URL.<br>
      This is because of forwarding request.<br><hr>
      Hello <%= request.getParameter("username") %><br><br>
      <u><%= request.getParameter("extraParameter") %></u>
```

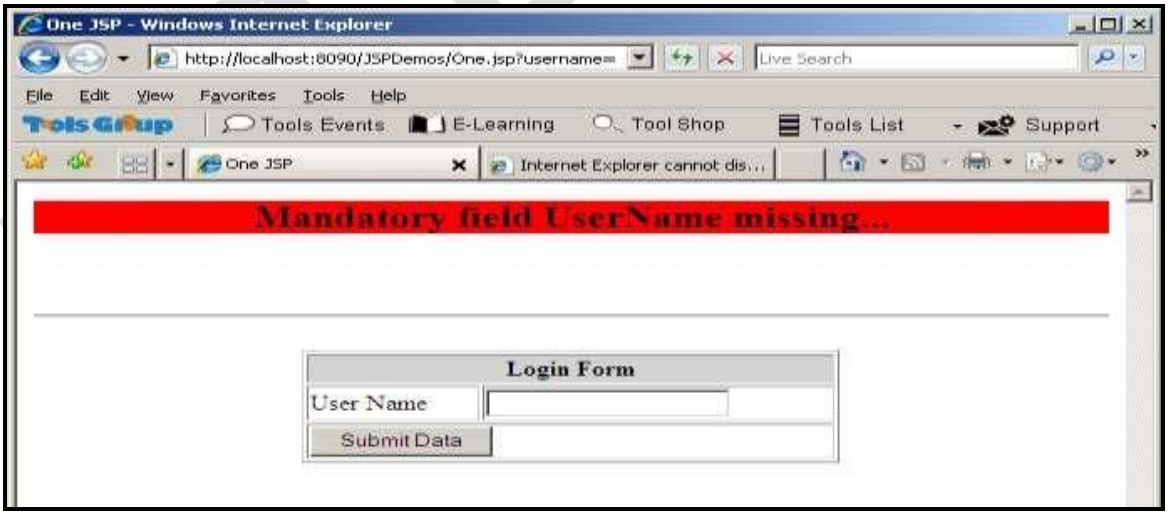
```
</h1>
</body>
</html>
```

Step 4: Deploy the application on the server.

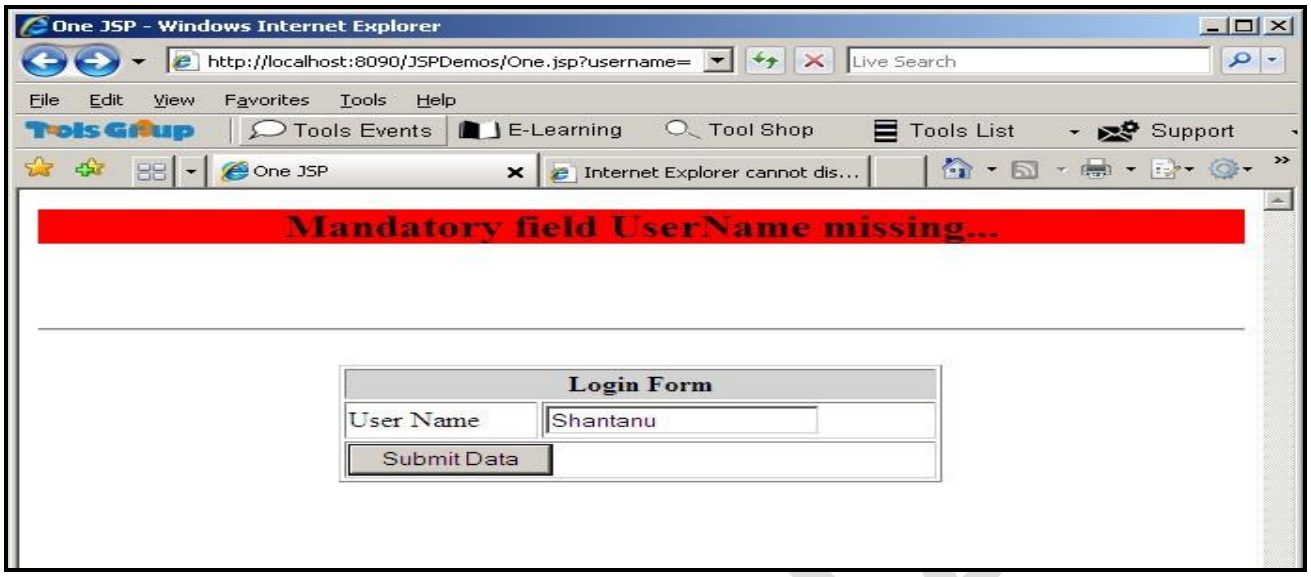
Step 5: Open the inputForm.html

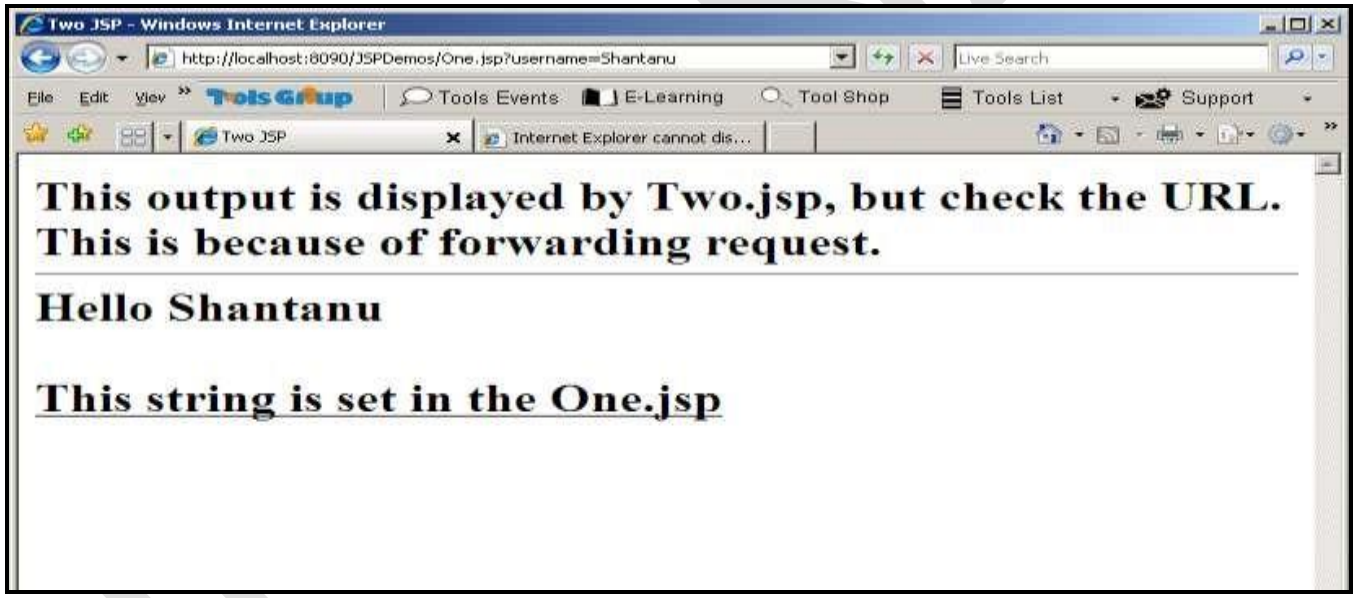


Step 6: Do not enter anything in User Name field and Click on the "Submit Data" button to see the following output.



Step 7: Now enter the User Name as "Shantanu" and click on "Submit Data" button to see the following output.





Summary of this assignment:

You have learnt

- Using <jsp:forward> and <jsp:include> tags

Note:

- With <jsp:forward>, the buffer is cleared before the forward.

Assignment 7: JSP Standard Actions <jsp:useBean>

Objective: To understand <jsp:useBean>, <jsp:setProperty> and <jsp:getProperty> action elements.

Problem Description: Create a simple web application to understand the jsp:useBean tag

Estimated time: 20 Min.

Step 1: Create a Java class "**CustomerBean.java**" in a package named "com..enr.beans".

```
package com..enr.beans;
 public class CustomerBean {
      private String name;
      private String address;
      private String city;

      //add corresponding getters and setters
}
```

Step 2: Create a html page, "**customer.html**". The output of the html page should be:

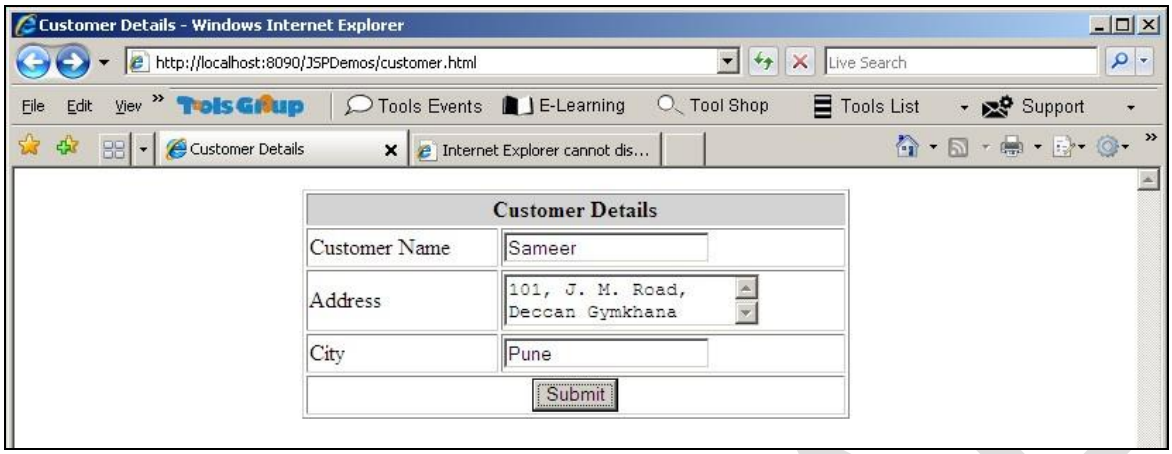
Names of the input fields should be same as that of the bean properties.

Note: Having same name allows setting the property without explicitly mentioning **param or value** attribute.
Also, for using property="*" the input field names should be same as that of the bean properties. Even, the data type of the bean properties should be primitive/wrapper classes only.
If there is any mismatch in name/data type of the bean properties then we cannot set all those properties using property= "*".

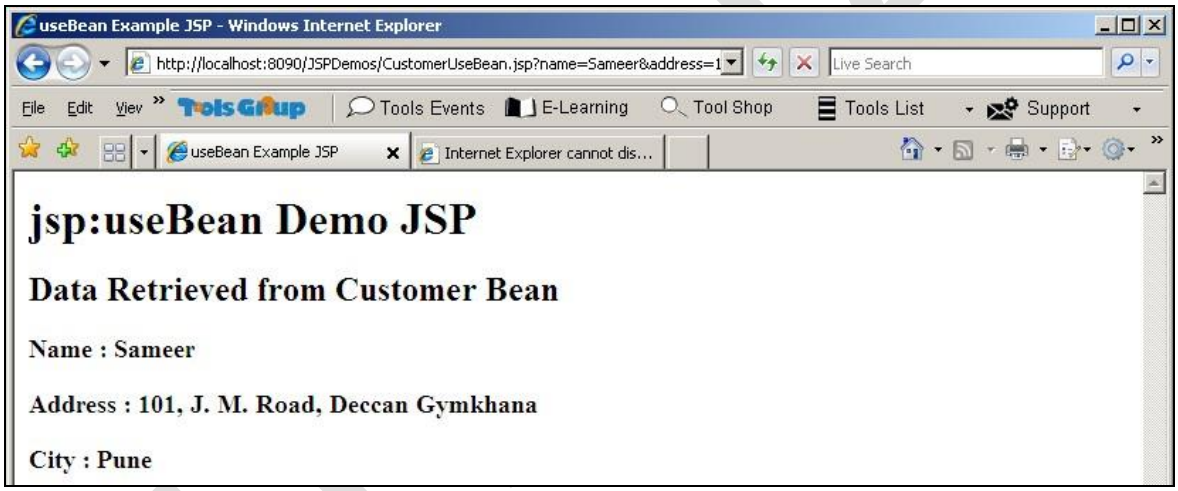
Step 3: Create a JSP, and name it as "**CustomerUseBean.jsp**". Type the following code in it and save it.

```
<%@ page import="com..enr.beans.CustomerBean" %>
<html>
<head><title> Cutsomer Use Bean JSP </title></head>
<body>
      <%-- Creation of bean --%>
      <jsp:useBean id="customerBean"
      class="com..enr.beans.CustomerBean">

            <%-- Setting the properties of the Bean --%>
            <jsp:setProperty name="customerBean" property="name"
                  value= "<%= request.getParameter("name")%>" />

            <%-- The param attribute lets you set the value of a bean
            property to the value of request parameter --%>
            <jsp:setProperty name="customerBean" property="address"
                        param="address" />

            <%-- Another way to set the property if the property name
            matches the input field name --%>
            <jsp:setProperty name="customerBean" property="city" />
      </jsp:useBean>
      <h1> jsp:useBean Demo JSP </h1>
      <h2> Data Retrieved from Customer Bean </h2>
      <h3> Name :
            <jsp:getProperty name="customerBean" property="name"/>
      </h3>
      <h3> Address :
          <jsp:getProperty name="customerBean" property="address"/>
</h3>
      <h3> City :
          <jsp:getProperty name="customerBean" property="city"/>
       </h3>
</body>
</html>
```

Step 4: Deploy all the files on web server and run customer.html



Step 6: Enter the details as in the web page and click on **"Submit"** to view the result.



Summary of this assignment:

In this assignment you have learnt

1. the use of <jsp:useBean> tag
2. Ways to set and get the property of the Bean, i.e. use of <jsp:setProperty> and <jsp:getProperty>

Note: With <jsp:useBean> you can have scope attribute which can take four different values to decide the scope of a bean.

a. **page (default)**
b. **request**
c. **application**
d. **session**

Exercises for Self Review:

1. Consider the following code snippet written in JSP and analyze, what will be the output, if we try to compile and execute this JSP?

```
<%

      int someValue = 0; try{
            someValue = 100/someValue;
      } finally{       out.println("Some Value = "
      +someValue);
      }

%>
```

2. Consider the following code snippet written in JSP and analyze, what will be the output, if we try to compile and execute this JSP?

```
<%!
      String str;
%>
The value in String is <%= str %>
```

3. Consider the following code snippet written in JSP and analyze, what will be the output, if we try to compile and execute this JSP?

```
<%!
      String str;
%>
The value in String is <%= str.toString() %>
```

4. Consider the following JSPs and analyze, what will be the output, if we try to compile and execute this JSP?

```
One.jsp
      <%@ page errorPage="Two.jsp" %>
      <% int i = 10/0; %>
```

Will One.jsp be compiled?

```
Two.jsp
      <%@ page isErrorPage="false" %>
      The exception occurred is <%= exception %>
```

Will Two.jsp be compiled?
If we try to invoke One.jsp with following URL:
http://localhost:8080/JSPDemos/One.jsp

5. What is difference between include directive and jsp:include action?

6. Consider the below code snippet written in Test.jsp present in **JSPDemos** web application, what will be the output if we try to compile it?

```
Hello <%= request.getParameter("UName") %>
```

Try to execute this JSP, through URL : http://localhost:8080/JSPDemos/Test.jsp **Note- This JSP is not invoked from any HTML Form**

7. Now the same JSP is called from the HTML page, testing.html, if we type **"Java"** in the User Name text field, what will be the output?

```
<form action="Test.jsp">
      User Name <input type = "TEXT" name="UName">
      <input type = "SUBMIT" name="OK">
</form>
```

Extra Assignment #1: Use of different control constructs in the JSP

Objective: To understand the usage of various control constructs, like for loop, while loop and do…while loop as well as if and switch statements, in JSP.

Problem Description: Use different control constructs and display data in tabular format by creating HTML table dynamically.

Estimated time: 15 Min.

Step 1: Create a web application. Create a file **index.html** and add the following code.

```
<body>
      <h1>
      <form action="Login.jsp">
            <input type="text" name="userid" / >
            <input type="Submit" name="Ok" />
      </form>
      </h1>
</body>
```

Step 2: Create a file **Login.jsp**. Check the value of userid submitted from **"index.html".** If the value submitted is **"Admin"** then display a link to **admin.jsp** else display the link to **trainee.jsp** <body>

```
<h1>
<%
    String userId = request.getParameter("userid");
    if(userId.equals("Admin")){
%>
          Welcome to Admin Page!!!
          <a href="admin.jsp">Admin Link</a>
<%
     }
     else{
%>
          Welcome to Trainee Page!!!
          <a href="trainee.jsp">Trainee Link</a>
<%
     }
%>
</h1>
</body>
```

Step 3: Create a file **trainee.jsp** to accept the name of the trainee. Depending upon its value, display the marks and grade for the trainee.

```
<body> <%!
      String [] names = {"Shantanu", "Sameer", "Saanika"};
      int[] marks = {100, 98, 99};     int x = 0;
%>
<%
      int mark = 0;    char grade='\0';
      if(request.getParameter("name") == null)
      {
%>
            Welcome!!! Currently we do not have your data, please enter your
            name!!!<br>
            <jsp:include page="trainee.jsp" />
<%
      }
      else
      {
%>
            Information of <%=request.getParameter("name") %>
```

```
            <table border="1">
            <tr>
                  <td>Name</td>
                  <td>Marks</td>
                  <td>Grade</td>
            </tr>
            <tr>
                  <td><%=request.getParameter("name")%></td>
<%
                  x=0;
            do
                  {
                        if(names[x].equals(request.getParameter("name")))
                        {
                        mark = marks[x];
            break;
                        }
                        x++;
                  }while(x<3);
%>
                  <td><%= mark%></td>
<%
                  switch(mark){
      case 100:grade = 'A';break;
      case 99:grade = 'B';break;
      case 98:grade = 'C';break;
                  }
%>
                  <td><%= grade%></td>
            </tr>
            </table>
<%
      }
%>
</body>
```

Step 4: Create a file **admin.jsp,** in this JSP, display the names, marks and grades of 3 trainees.

```
<h1>
      All Trainee Report
</h1><br><br><br> <%!
      String [] names = {"Shantanu", "Sameer", "Saanika"};
      int[] marks = {100, 98, 99};     int x = 0;
%>


<table border="1">
      <tr>
         <td>First</td>
         <td>Second</td>
         <td>Third</td>
      </tr>
      <tr>
<%    for(x=0;x<3;x++)
      {
%>
         <td><%=names[x] %></td>
<%
      }
%>
      </tr>
      <tr>
<%
           x=0;
          do{
%>
                <td><%=marks[x++] %></td>
<%
           }while(x<3);
%>
      </tr>
</table>
</body>
```

Summary of this assignment: You have just learnt

- Using selectional control constructs like “if” in the JSP
- Using iterational control constructs in JSP

Extra Assignment# 2: Passing parameters through URL

Objective: Passing parameters through URL

Problem Description: When the user clicks on different links on the same page, display is changing based on the parameters passed by the URL.
Estimated time: 10 Min.

1) Create a html page called **linkDemo.html** with the following links.
   a) User1
b) User2
c) User3

When the user clicks on the links it should invoke user.jsp and display the **name** based on the links.

Hint: The name is available with the request object as a parameter. Using request object you can retrieve the name value.

Note: POST method can be used only with HTML <form> tag. All request methods to a web server is GET by default. So when we click on the above URL the data is passing to the server through the URL. Modifying the URL with parameters is known as URL rewriting.

Summary of this assignment:

In this assignment you have learnt

- URL rewriting technique, one way to perform session tracking