JSP Tutorial - Java Inside HTML

Contents

Time Required: 90 minutes

NOTE: This tutorial was tested with Eclipse Enterprise Java and Web Developers 2021-03 and Tomcat version 9 on Windows.

JSPs are Internally Compiled into Java Servlets

Anything that can be done using JSPs can also be accomplished using Java servlets. However, it is important to note that servlets and JSPs are complementary technologies, NOT replacement of each other. A servlet can be viewed as "HTML inside Java", which is better for implementing business logic - as it is Java dominant. JSP, on the other hand, is "Java inside HTML", which is superior for creating presentation - as it is HTML dominant. In a typical Model-View-Control (MVC) application, servlets are often used for the Controller (C), which involves complex programming logic. JSPs are often used for the View (V), which mainly deals with presentation. The Model (M) is usually implemented using JavaBean or EJB.

Tutorial 1.1: Do You Feel Lucky?

1. Start **Eclipse**. Open up your **HelloWorld** project.
2. In the Eclipse project: Go to **src** → **main** → **webapp**. Right Click **webapp** → **New** → Click **JSP File**.
3. File name: **DoYouFeelLucky.jsp**. Click **Finish**.
4. Insert the following code between the body tags.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Do You Feel Lucky?</title>
</head>
<body>
    <%
        // Inside the JSP code, use Java commenting
        // Generate a random double from 0 to <1
        double num = Math.random();
        // How lucky are you?
        if (num > 0.95) {
    %>
        <h2>You'll have a lucky day!</h2>
        <!-- Print random number variable -->
        <p>(<%=num%>)</p>
    <%
        } else {
    %>
        <h2>Well, life goes on ...</h2>
        <!-- Print random number variable -->
        <p>(<%=num%>)</p>
    <%
        }
    %>
    <!-- This is outside the JSP code, use html commenting -->
    <!-- Reload the page -->
    <h3><a href="<%=request.getRequestURI()%>">Try Again</a></h3>
</body>
</html>
```

5. Save the file.
6. Run the JSP project on Tomcat: **Right Click** on your JSP file → **Run as** → **Run on Server**.
7. **Run on Server** dialog box: You should see your Tomcat Server automatically selected. Click **Finish**.

8. Your jsp page should show up in a new tab.
9. Copy and paste the URL into a web browser. You should see the same web page.
10. From your browser, choose the "View Source" option to check the response message. It should be either of the following depending on the random number generated.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Do You Feel Lucky?</title>
</head>
<body>

        <h2>Well, life goes on ...</h2>
        <!-- Print random number variable -->
        <p>(0.5114929880774527)</p>

    <!-- This is outside the JSP code, we have to use html commenting -->
    <!-- Reload the page -->
    <a href="/HelloWorld/DoYouFeelLucky.jsp"><h3>Try Again</h3></a>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Do You Feel Lucky?</title>
</head>
<body>

        <h2>You'll have a luck day!</h2>
        <!-- Print random number variable -->
        <p>(0.9528847660930689)</p>

    <!-- This is outside the JSP code, we have to use html commenting -->
    <!-- Reload the page -->
    <a href="/HelloWorld/DoYouFeelLucky.jsp"><h3>Try Again</h3></a>
</body>
</html>
```

It is important to note that the client is not able to "view" the original JSP script (otherwise, you may have security exposure), but merely the result generated by the script.

Explanation

- A JSP script is a regular HTML page containing Java programs.

- The Java statements are enclosed by <% ... %> (called JSP scriptlet) or <%= ... %> (called JSP expression).
- JSP Scriptlet <% ... %> is used to include Java statements.
- JSP Expression <%= ... %> is used to evaluate a single Java expression and display its result.
- The method **request.getRequestURI()** is used to retrieve the URL of the current page. This is used in the anchor tag <a> for refreshing the page to obtain another random number.

Behind the Scenes with Servlets

When a JSP is first accessed, Tomcat converts the JSP into a servlet; compile the servlet, and execute the servlet.

Check out the generated servlet for **DoYouFeelLucky.jsp**, and study the JSP-to-servlet conversion. Look for the java files under your Eclipse workspace.

Eclipse-Projects\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\HelloWorld\org\apache\jsp

The relevant part of the generated servlet is extracted as follows:

```
out.write("\r\n");
out.write("<!DOCTYPE html>\r\n");
out.write("<html>\r\n");
out.write("<head>\r\n");
out.write("<meta charset=\"ISO-8859-1\">\r\n");
out.write("<title>Do You Feel Lucky?</title>\r\n");
out.write("</head>\r\n");
out.write("<body>\r\n");
out.write("\t");

// Inside the JSP code, we use Java commenting
// Generate a random double from 0 to <1
double num = Math.random();
// How lucky are you?
if (num > 0.95) {

out.write("\r\n");
out.write("\t\t<h2>You'll have a luck day!</h2>\r\n");
out.write("\t\t<!-- Print random number variable -->\r\n");
out.write("\t\t<p>(");
out.print(num);
out.write(")</p>\r\n");
out.write("\t");

} else {

out.write("\r\n");
out.write("\t\t<h2>Well, life goes on ...</h2>\r\n");
out.write("\t\t<!-- Print random number variable -->\r\n");
out.write("\t\t<p>(");
out.print(num);
out.write(")</p>\r\n");
out.write("\t");

}
```

Explanation

- The HTML statements are written out as part of the response via **out.write()**, as "it is".
- The JSP scriptlets <% ... %> are kept, as "it is", in the converted servlet as the program logic.
- The JSP expressions <%= ... %> are placed inside an **out.print()**. Hence, the expression will be evaluated, and the result of the evaluation written out as part of the response message.

Subsequent accesses to the same JSP will be much faster. They will be re-directed to the converted and compiled servlet directly (no JSP-to-servlet conversion and servlet compilation needed again), unless the JSP has been modified.

Tutorial 1.2: Getting with the Times!

You can insert any plain Java code inside a scriptlet. Everything that you have learned can be used.

The following illustrates the syntax of scriptlet:

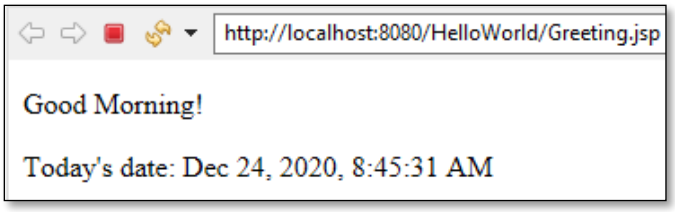
<% // any java source code here %>

1. Open your **HelloWorld** project.
2. In the **webapp** folder, create a new JSP file named **Greetings.jsp**
3. Enter the following code.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Greetings!</title>
</head>
<body>
    <%
    // using scriptlet
    java.util.Calendar now = new java.util.GregorianCalendar();
    String tod = "";

    if (now.get(now.HOUR_OF_DAY) < 12) {
        tod = "Morning!";
    } else if (now.get(now.HOUR_OF_DAY) < 18) {
        tod = "Afternoon!";
    } else {
        tod = "Evening!";
    }
    %>

    <p>Good <%=tod%></p>

    <p>Today's date: <%=(new java.util.Date()).toLocaleString()%></p>
</body>
</html>
```

4. Test and debug the application.

Example run:



Tutorial 1.3: HTML Form Processing

An HTML form is a graphic user interface (GUI) which you present to get the input data from users. The user submits the form from the client side. On the server side, the form data is captured for further processing such as business logic validation, saving the data into the database and so on.

1. Create a New **Dynamic Web Project** named **FormJSP**
2. In the **webapp** folder, create a new html file named **index.html**
3. Enter the following code in **index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
  <title>JSP Form Demo</title>
        <style type="text/css">
            label{ margin-right:20px;}
            input{ margin-top:5px;}
        </style>
</head>
    <body>
        <form action="/Form/HandleUserInfo.jsp" method="post">
            <fieldset>
                <legend>User Information</legend>
                <label for="fistName">First Name</label>
                <input type="text" name="firstName" /> <br/>
                <label for="lastName">Last Name</label>
                <input type="text" name="lastName" /> <br/>
                <label for="email">Email</label>
                <input type="text" name="email" /> <br/>
                <input type="submit" value="submit">
            </fieldset>
        </form>
    </body>
</html>
```

The form is very simple. It contains three fields: first name, last name, and email. It also contains a submit button which users can submit when he/she finish entering the data.

In the form tag, you can see it uses the post method to post the form data to the server. Let's create a JSP file that will get and process the form's data.

4. Create a JSP file named: **HandleUserInfo.jsp**

5. Enter the following code.

```
<%@ page language="java" contentType="text/html; charset=ISO-8
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Form Handler</title>
</head>
<body>
    <%
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String email = request.getParameter("email");
    %>
    <p>Hi <%=firstName%> <%=lastName%>!,
        your submitted email is <%=email%>.</p>

</body>
</html>
```

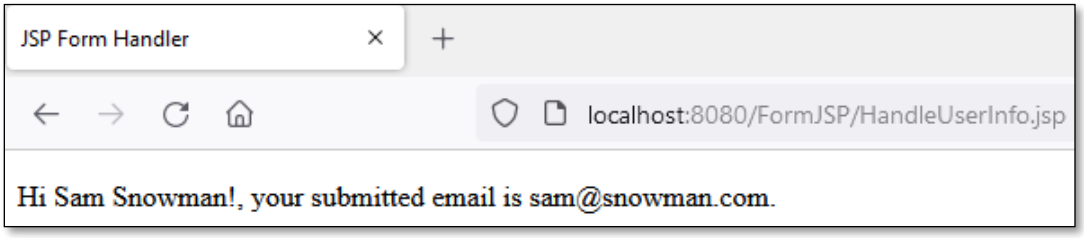
We use the request object to get the form data. The **getParameter()** method of request object accepts the name of the form field and return field value. The returned value of the **getParameter()** method is always string type. If you have a form field that accepts a numerical value, you have to convert it.

If no form field is found, the **getParameter()** method returns null. After getting the values from the form, the **handleUserInfo.jsp** uses those values to display a message.

6. Go to the project name: Right Click on the project **FormJSP**

7. Click **Run as → Run on Server**.

8. Enter your information in the form. Click **Submit**. You should see something like the following.



Tutorial 1.4: Single Page Form Processing

When dealing with HTML, you have multiple ways to separate the code to handle the form data and HTML form itself. These are the most two common ways to do so:

- Page which contains HTML Form and JSP page which handles HTML Form are separated. (as you see in the previous tutorial)
- Single JSP page to display form and handle form.

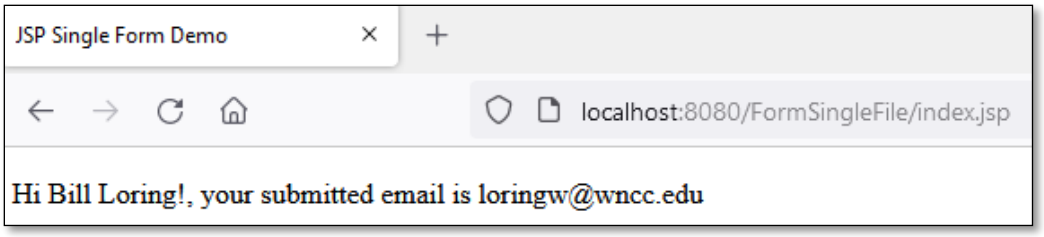
In order to do it in the second way, we need to add an additional hidden field in the HTML form. When user submits the form data, we check the value of this hidden field to see whether the form is submitted or not.

index.jsp is the default file for a web application in JSP. We can run the application, rather than a specific page.

1. Create a New **Dynamic Web Project** named **Form**
2. In the **webapp** folder, create a new html file named **index.jsp**
3. Enter the following code in **index.jsp**

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
  <title>JSP Single Form Demo</title>
        <style type="text/css">
            label{ margin-right:20px;}
            input{ margin-top:5px;}
        </style>
</head>
<body>
    <%
    // Get the value of the isSubmitted variable
    String val = request.getParameter("isSubmitted");

    // isSubmitted = 0 page has not been submited
    int isSubmitted = 0;
    if (val != null) {
        isSubmitted = Integer.parseInt(val);
        if (isSubmitted == 1) {
            String firstName = request.getParameter("firstName");
            String lastName = request.getParameter("lastName");
            String email = request.getParameter("email");
            out.println("<p>Hi " + firstName + " " + lastName +
                "!, your submitted email is " + email + "</p>");
        }
    }
    %>
    <%

    // Has the form been submitted? 0 = form not submitted
    if (isSubmitted == 0) {%>
    <form action="index.jsp" method="post">
        <fieldset>
            <legend>User Information</legend>
            <label for="fistName">First Name</label> <input type="text"
                name="firstName" /> <br /> <label for="lastName">Last Name</label>
            <input type="text" name="lastName" /> <br /> <label for="email">Email</label>
            <input type="text" name="email" /> <br />

            <!-- value = "1" page has been submited -->
            <input type="hidden" name="isSubmitted" value="1" />
                <input type="submit" value="submit">
        </fieldset>
    </form>
    <%}%>
</body>
</html>
```

Example run:



4. Right Click on the **Form** project folder.
5. Click **Run As → Run on Server**.

NOTE: When you look at the initial url, notice it doesn't show the page name. The **index.jsp** file is automatically run as it is the default document for the web application. The name **index.jsp** only shows up after the form has posted.

http://localhost:8080/FormSingleFile/

Explanation

The action of the Form now is pointed to the page which form is embedded, the **index.jsp** page. The logic of processing the form data and displaying the form are in a single JSP page. The sequence of events occurs as follows.

1. The first time when the page is loaded, the code checks for the form field name **isSubmitted**. Because **getParameter** will return null the **isSubmitted** variable is set to 0 and the code displays the Form.
2. When user submits the form, a hidden field called **isSubmitted** with the value 1 is available on the form. The parameter **isSubmitted** is available. The code checks for the parameter **isSubmitted** and finds it. The process of collecting data and printing them out occurs as normal but no Form is output after the post back.

One of the most important thing to keep in mind when you deal with any form: Never trust data which is submitted from client side. You can do client validation for simple validation such as required field, email is in the correct format in by using JavaScript. In the server side, you can do simple validation also after collecting the data, plus business validation to validate the constraints among data you collected.

Tutorial 1.5: Session

The HTTP protocol is stateless. There is no permanent connection between the client (web browser) and Web Server. When a client requests a page from a web server, it opens a connection, retrieves the page and closes the connection. Web servers don't know what

happens next on the client side. If another request from a client is made, the web server doesn't associate the new connection with the connection has been made.

To overcome the statelessness of the HTTP protocol, JSP provides you the implicit session object which is an **HttpSession** object. The session object resides on the server side so you can keep arbitrary data about the client and other data as well in session and later on in different requests you can retrieve the saved data for processing. JSP stores data on the server side in the session object by using a single key that client remembers.

Let's take a look at an example how to use session object. In this example, we have three pages: In the first page, we collect data from the user, after that user submits the form to a second page which is used to store data in a session. In the last page, we get data back from the session and display it.

1. Create a New **Dynamic Web Project** named **SessionJSP**
2. In the **webapp** folder, create an html file named **index.html**
3. Enter the following code in **index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSP Form</title>
</head>
<body>
    <form method="post" action="/SessionJSP/SavetoSession.jsp">
        <table>
            <tr>
                <td>First Name</td>
                <td><input type="text" name="firstName" /></td>
            </tr>
            <tr>
                <td>Last Name</td>
                <td><input type="text" name="lastName" /></td>
            </tr>
            <tr>
                <td>Comments</td>
                <td><textarea name="comments" cols="30" rows="5"></textarea></td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="Submit" /></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

In the form above, when the user enters information, clicks submit button, the data goes through the page **SavetoSession.jsp**. In the **SavetoSession.jsp** page, we save all the submitted data into the **session** object and forward the request to another page called **Display.jsp**.

4. In the **webapp** folder, create a JSP file named **SavetoSession.jsp**
5. Enter the following code in **SavetoSession.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>

<%
    // Get the data from the form, store in String variables
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String comments = request.getParameter("comments");

    // Save the data into the session
    session.setAttribute("firstName", firstName);
    session.setAttribute("lastName", lastName);
    session.setAttribute("comments", comments);
%>

<html>
<head>
<meta charset="ISO-8859-1">
<title>Save to Session</title>
</head>
<body>
    <%-- forward to the display.jsp page--%>
    <jsp:forward page="Display.jsp" />
</body>
</html>
```

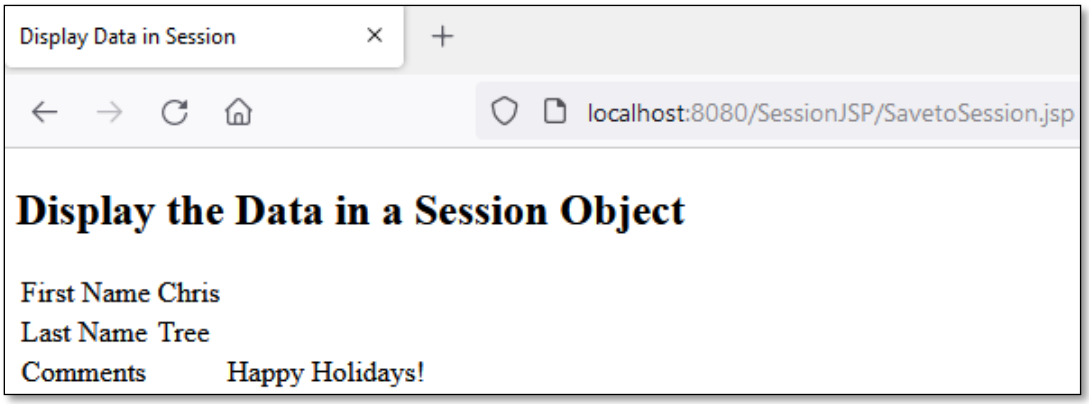
In the code above, we use the **setAttribute()** method to save data into the **session** object.

1. In the **webapp** folder, create a JSP file named **Display.jsp**
2. Enter the following code in **Display.jsp**

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Display Data in Session</title>
</head>
<body>
    <h2>Display the Data in a Session Object</h2>
    <table>
        <tr>
            <td>First Name</td>
            <td><%=session.getAttribute("firstName")%></td>
        </tr>
        <tr>
            <td>Last Name</td>
            <td><%=session.getAttribute("lastName")%></td>
        </tr>
        <tr>
            <td>Comments</td>
            <td>
            <td><%=session.getAttribute("comments")%></td>
        </tr>
    </table>
</body>
</html>
```

We used the **getAttribute()** method of the **session** object to retrieve data which was entered in the form and displayed it on the page by using the expression.

Run the **Form.html** page and try it out. You should end up with something like this.



How Session Works

When the server creates a new session, it always adds a session identifier in the form of a cookie. When the web browser asks for a page or makes a request, the web browser always sends a cookie which is created by the web server in the request. Therefore in the server

side, web server checks for that cookie and find the corresponding session that is matched to the received cookie.

The session is normally short-lived so the session cookie is not saved into the disk. Session also has a timeout. When the time is out, the session is no longer exist on the server side. You can set time out of session in the configuration file in the server.

Assignment Submission

- Insert screenshots showing your jsp pages running in Eclipse and in a web browser.
- Zip up the project folders.
- Attach and submit in Blackboard.