Processing Input and Output

Form Processing in JSP

Reading Request Parameter Values

When we need to pass information from our browser to the web server, form processing the most common method for web processing. It is the common method that helps us to interact with the web pages which became very easy with the help of JSP. There are two ways through which a browser can send the data to the server:

1. **GET**: It is used to get a resource or data from the server.
2. **POST**: It is used to submit or post data to the server for processing.

GET Method

It is the default method used to pass information from the browser to the webserver. It sends the encoded user information to the page request separated by the "?" character. We can only send 1024 characters in the request as it has a size limitation. It is not recommended to use the GET method if you have the password or other sensitive information to pass to the server because it produces a long string that appears in your browser.

POST Method

It is a more reliable method used to pass information to a backend program. It sends information as a separate message instead of sending it as a text string after a "?" in the URL. It is recommended to use the GET method if you have the password or other sensitive information to pass to the server because it produces a long string that appears in your browser. The information comes as standard input to the backend program which we can use for processing.

Methods to Handle Form Data Processing in JSP

1. **getParameter()**: It is used to get the value of a form parameter.
2. **getParameterValues()**: It is used to return multiple values if the parameter appears more than once and return multiple values.
3. **getParameterNames()**: It is used to get the name of parameters if you want a complete list of all parameters in the current request.
4. **getInputStream()**: It is used to read binary data stream coming from the client.

Example of the GET Method using URL in JSP

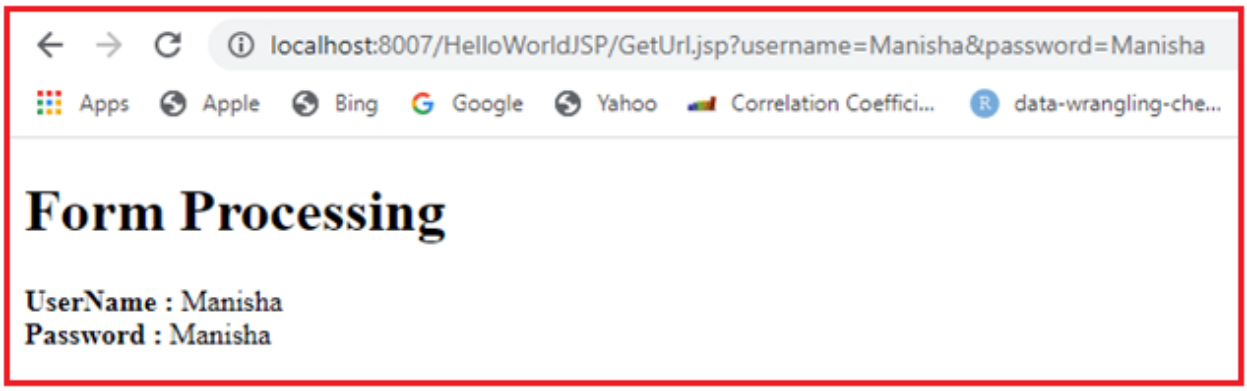
In this example, we have passed two values using the GET method. **GetUrl.jsp** is the file used to handle input given by the web browser. We have used the getParameter() method which is easy to access the passed information.

GetUrl.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<title>GET using URL</title>
</head>
<body>
<h1>Form Processing</h1>
<p>
<b>UserName :</b>
<%=request.getParameter("username")%>
<br /> <b>Password :</b>
<%=request.getParameter("password")%>
</p>
</body>
</html>
```

Output

Type **http://localhost:8007/HelloWorldJSP/GetUrl.jsp?username=Manisha&password=Manisha** to get the output:



Example of the GET Method using FORM in JSP

In this example, in **GetForm.jsp** we have used a form with two fields "username" and "password" with a submit button through which we have processed the action to another JSP page. We have fetched the input using the getParameter method. The submit button helps us to pass the field values into another GetFormProcess.jsp JSP page. In **GetFormProcess.jsp** pages, we get the values using the request object's getParemeter method.

Reading Form Data using JSP

JSP handles form data parsing automatically using the following methods depending on the situation:

- *getParameter()* − You call *request.getParameter()* method to get the value of a form parameter.
- *getParameterValues()* − Call this method if the parameter appears more than once and returns multiple values, for example - checkbox.
- *getParameterNames()* − Call this method if you want a complete list of all parameters in the current request.
- g*etInputStream()* − Call this method to read binary data stream coming from the client.

Reading HTML Form Data with JSP

We'll actually cover the following topics:

1. Read HTML forms input text field value in JSP
2. Read HTML forms select tag (drop-down list) value in JSP
3. Read HTML forms radio-button field value in JSP
4. Read HTML forms checkbox tag value in JSP

1. Read HTML Form Input Text field Value

In this example, we will build an HTML form to read student information. Let's first create a simple HTML form with input field elements. Later we will read HTML form data using JSP.

student-form.html

```
<html>

<head>
   <title>Student Registration Form</title>
</head>

<body>

   <form action="student-response.jsp" method="post">

      First name: <input type="text" name="firstName" />

      <br/><br/> Last name: <input type="text" name="lastName" />

      <br/><br/>

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

   </form>

</body>

</html>
```

student-response.jsp

Let's create *student-response.jsp* file and add the following code. The key points in this file:

1. We use *request.getParameter()* method to get the value of a form parameter.
2. We can also use *${param.firstName}* expression language to read the input value.

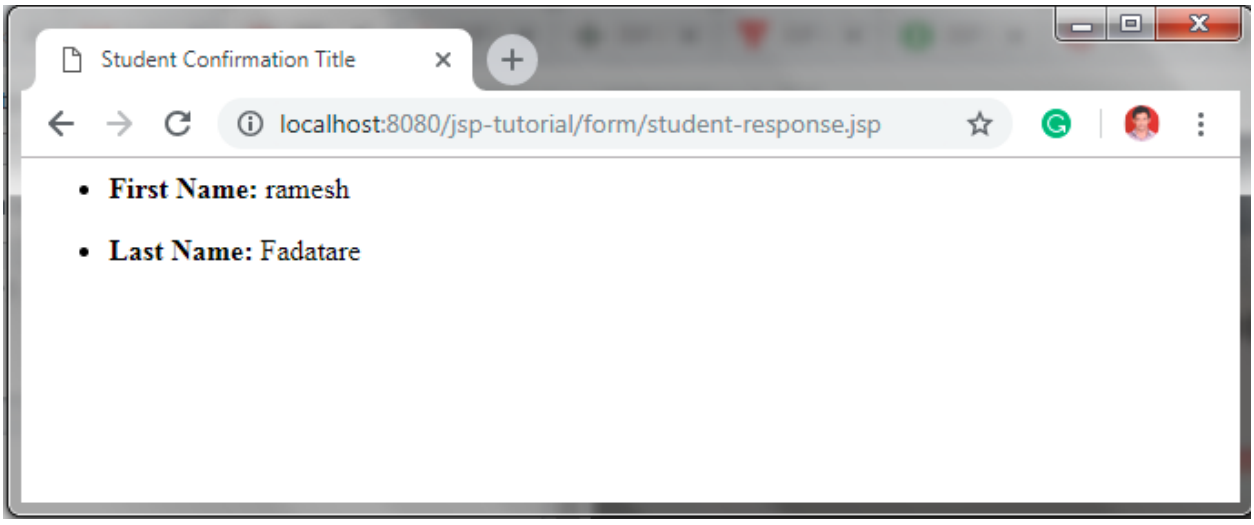
3. We are using an expression tag <%= %> to display a result.

```
<html>

<head><title>Student Confirmation Title</title></head>
<body>
      <ul>
        <li><p><b>First Name:</b>
          <%= request.getParameter("firstName") %>
        </p></li>
        <li><p><b>Last  Name:</b>
          <%= request.getParameter("lastName") %>
        </p></li>
      </ul>
</body>
</html>
```

Fill above Student registration HTML form and hit submit button will result in below page.

Output



the JSP Action tags are used to achieve the above purpose. JSP tags are specifically used during request processing. The tags used here will be as follows :

jsp:useBean: It will be used to create the java bean and instantiate it.

jsp:setProperty: It will be used to set the property of the created bean, using the form data.

jsp:getProperty: It will be used to display the details entered.

It may be noted here that, all actions tags use the id attribute to uniquely identify an action element and refer to it inside the JSP page.

The program has been tested on Netbeans IDE 8.1 using Apache Tomcat as the application server.

Steps to Validate a User:

1. We click the link on index.html page to deploy the application.
2. We are then presented with a form, where we enter username and password and click submit.
3. The JSP gets automatically called and it returns the data entered in the form and the result of Validation.

Form to accept username and password : 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>Login Page</title>
    </head>
    <body>
       <h1>User Details</h1>
       <%-- The form data will be passed to acceptuser.jsp
            for validation on clicking submit
       --%>
       <form method ="get" action="acceptuser.jsp">
          Enter Username : <input type="text" name="user"><br/><br/>
          Enter Password : <input type="password" name ="pass"><br/>
             <input type ="submit" value="SUBMIT">
       </form>
    </body>
</html>
```

JSP to accept form data and verify a user : acceptuser.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Accept User Page</title>
    </head>
    <body>
        <h1>Verifying Details</h1>
        <%-- Include the ValidateUser.java class whose method
            boolean validate(String, String) we will be using
        --%>
        <%-- Create and instantiate a bean and assign an id to
            uniquely identify the action element throughout the jsp
        --%>
        <jsp:useBean id="snr" class="saagnik.ValidateUser"/>

        <%-- Set the value of the created bean using form data --%>
        <jsp:setProperty name="snr" property="user"/>
        <jsp:setProperty name="snr" property="pass"/>

        <%-- Display the form data --%>
        The Details Entered Are as Under<br/>
        <p>Username : <jsp:getProperty name="snr" property="user"/></p>
        <p>Password : <jsp:getProperty name="snr" property="pass"/></p>

        <%-- Validate the user using the validate() of
            ValidateUser.java class
        --%>
        <%if(snr.validate("GeeksforGeeks", "GfG")){%>
            Welcome! You are a VALID USER<br/>
        <%}else{%>
            Error! You are an INVALID USER<br/>
        <%}%>
    </body>
```

```
</html>
```

The ValidateUser.java class

```
package saagnik;
import java.io.Serializable;

// To persist the data for future use,
// implement serializable
public class ValidateUser implements Serializable {
   private String user, pass;

   // Methods to set username and password
   // according to form data
   public void setUser(String u1) { this.user = u1; }
   public void setPass(String p1) { this.pass = p1; }

   // Methods to obtain back the values set
   // by setter methods
   public String getUser() { return user; }
   public String getPass() { return pass; }

   // Method to validate a user
   public boolean validate(String u1, String p1)
   {
      if (u1.equals(user) && p1.equals(pass))
         return true;
      else
         return false;
   }
}
```

Outputs:

login.jsp

Format tags in JSP are imported with the help of (The JavaServer Pages Standard Tag Library) JSTL library. These are used to edit the look of text or numbers used on the page. For example, date, time, timestamp, etc., are formats which can be obtained in a standardized form with the help of format functions in JSP. These standards are maintained to avoid the confusion on formats used throughout the world. There can be several ways to define a format as per the business requirements; some of the ways are via JSP functions, using format tags from the JSTL library in JSP, using XML data source for maintaining the format, and then extracting it using JSTL and JSP. In this topic, we would see examples and explanations for formatting the data using some of these methods.

Syntax

Some dependent libraries should be included in the project's lib folder as the format tag has dependencies in the JSTL library. You can either link a full set of libraries in jSTL or can target specific files as per your purpose. Two statements to be written before start writing the code to use the format tag are:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
```

The prefix denotes the format one would like to use. One can get a complete list of "prefixes" with uri the same as above as per the requirements.

The syntaxes used by JSP Format are:

1. Number: <fmt:formatNumber properties> Content to be formatted </fmt:formatNumber>

2. Parsing a customised TimeZone: <fmt:timeZone properties> Content to be formatted </fmt:timeZone>
3. Parsing a new number: <fmt:parseNumber properties> Content to be formatted </fmt:parseNumber>
4. Parsing a date: </fmt:parseDate val="" var="" pattern="">
5. Setting a timezone from existing Time Zones in the system: <fmt:setTimeZone value="" var="" scope=""/>
6. Setting a date format from existing date format in the system: <fmt:formatDate var="" value="" type="" dateStyle="" timeStyle=""/>