3.3 SERVLET GET AND POST ACTIONS

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver. Properties of Servlets are as follows:

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Handling the client data

We must have come across many situations where we need to fill a form with our data to access some application. In this situation, as a developer, you need to handle that data or information to pass from the browser to the web/application server and finally to your backend program to process that request. In this chapter, we will learn how to handle this form of data using Servlets.

HttpServlet Class

Java provides HttpServlet class which extends the GenericServlet class to process HTTP-specific request/response for a website. It provides HTTP specific methods such as doGet(), doPost(), doPut(), doDelete() etc. If we are extending the HttpServlet in our class, we must override at least one of these methods to process the request and send the response. We need to use either doGet() or doPost() method in the servlet class to get the information from the browser based on the method specified in the form.

GET method type and doGet() method

The doGet() method in servlets is used to process the HTTP GET requests. So, basically, the HTTP GET method should be used to get the data from the server to the browser. Although in some requests, the GET method is used to send data from the browser to the server also. In this case, you need to understand the below points on how the GET method will work.

- The data that is being submitted to the server will be visible in the URL using query parameters like this "*http://localhost:8080/HelloServlet/hello?myParam=myValue*".

- So, if you are sending any sensitive information like passwords, you should not use the GET method as the data entered can be clearly visible in the browser URL.

POST method type and doPost() method

The doPost() method in servlets is used to process the HTTP POST requests. It is used to submit the data from the browser to the server for processing. The data submitted with POST method type is sent in the message body so it is secure and cannot be seen in the URL. And there is no limit on the data that can be sent through the POST method. Ideally, we need to use the POST method, to send the form data to the webserver. So, we will be using the doPost() method in this example. And we will learn how to handle some of the common HTML fields data such as *Text field, Checkbox, Radio button, Dropdown*, etc., values in the servlet.
To achieve this, Java provides **ServletRequest** interface.

HttpServletRequest(I):

- HttpServletRequest interface extends the ServletRequest interface to provide the Http-specific request information for Servlets.
- The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods – doPost(), doGet(), etc.,
- The object provides data like parameter name and values, attributes, and an input stream. And it has various methods to work with the client data.
- In this example, we will use *getParameter*() and *getParameterValues*() to read values of the form fields.

getParameter() method:

- The method returns the value of a field/parameter from the request which is specified by the given name, as a String. If the specified parameter name does not exist, it returns null.
- This method should be used on the parameter that has only one value. If multiple values are there, then it returns only the first value.

Syntax:

Java

```
// String type specifying the parameter

// name of which the value to be returned as String

java.lang.String getParameter(java.lang.String name)
```

getParameterValues() method:

The method returns an array of String objects containing all of the values of the given field/parameter from the request. If the specified parameter name does not exist, it returns null.

Syntax:

Java

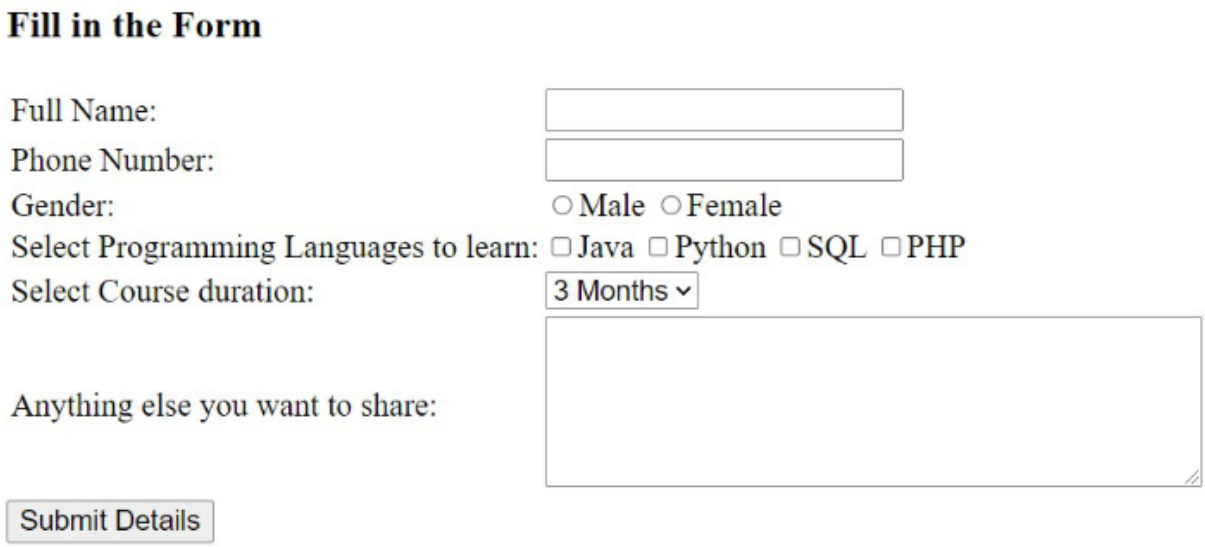
```
// String type specifying the parameter name

// of which the values to be returned in array

java.lang.String[] getParameterValues(java.lang.String name)
```

Example

We will create a simple form to get the details from client like below,



Steps to create the application:

- Create an HTML to take the input data from the client.
- Create a Servlet to handle the data, process the request, and generate the response.
- Run the application.

Create Details.html page:

HTML

```
<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>User Details</title>

</head>
```

```
<body >
   <h3>Fill in the Form</h3>
     <form action="FormData" method="post">
       <table>
          <tr>
             <td>Full Name:</td>
             <td><input type="text" name="name" /></td>
          </tr>
          <tr>
             <td>Phone Number:</td>
             <td><input type="text" name="phone" /></td>
          </tr>
          <tr>
             <td>Gender:</td>
             <td><input type="radio" name="gender" value="male" />Male
                 <input type="radio" name="gender" value="female" />Female</td>
          </tr>
          <tr>
             <td>Select Programming Languages to learn:</td>
```

```
    <td><input type="checkbox" name="language" value="java" />Java
        <input type="checkbox" name="language" value="python" />Python
        <input type="checkbox" name="language" value="sql" />SQL
        <input type="checkbox" name="language" value="php" />PHP</td>
  </tr>
  <tr>
    <td>Select Course duration:</td>
    <td><select name="duration">
        <option value="3months">3 Months</option>
        <option value="6months">6 Months</option>
        <option value="9months">9 Months</option></select></td>
  </tr>
  <tr>
    <td>Anything else you want to share:</td>
    <td><textarea rows="5" cols="40" name="comment"></textarea></td>
  </tr>
</table>
<input type="submit" value="Submit Details">
```

```
    </form>
  </body>
</html>
```

- Create a form page using common HTML tags.
- To display the form fields in a structured manner, we are using the <table> tag.
- We have '*Full Name'* and *'Phone Number'* as input text fields, '*Gender'* as Radio button, '*Programming Languages'* as Checkbox, '*Duration of course'* in Dropdown values, and Text area for '*Comments*'.
- We mentioned the form action to '*FormData*' and the method to '*post*'. So, when the form is submitted, the page will be mapped to the URL – '/*FormData*' specific Servlet and execute the '*doPost()*' method.

Create FormDataHandle.java Servlet:

Java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
        // Servlet implementation class FormDataHandle
```

```
	// Annotation to map the Servlet URL
@WebServlet("/FormData")
public class FormDataHandle extends HttpServlet {
	private static final long serialVersionUID = 1L;
	// Auto-generated constructor stub
	public FormDataHandle() {
		super();
	}
	// HttpServlet doPost(HttpServletRequest request, HttpServletResponse response) method
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// Get the values from the request using 'getParameter'
		String name = request.getParameter("name");
		String phNum = request.getParameter("phone");
		String gender = request.getParameter("gender");
		// To get all the values selected for
		// programming language, use 'getParameterValues'
		String progLang[] = request.getParameterValues("language");
		// Iterate through the String array to
		// store the selected values in form of String
		String langSelect = "";
		if(progLang!=null){
			for(int i=0;i<progLang.length;i++){
```

```
                    langSelect= langSelect + progLang[i]+ ", ";
                }
            }
            String courseDur = request.getParameter("duration");
            String comment = request.getParameter("comment");
            // set the content type of response to 'text/html'
            response.setContentType("text/html");
            // Get the PrintWriter object to write
            // the response to the text-output stream
            PrintWriter out = response.getWriter();
            // Print the data
            out.print("<html><body>");
            out.print("<h3>Details Entered</h3><br/>");
            out.print("Full Name: "+ name + "<br/>");
            out.print("Phone Number: "+ phNum +"<br/>");
            out.print("Gender: "+ gender +"<br/>");
            out.print("Programming languages selected: "+ langSelect +"<br/>");
            out.print("Duration of course: "+ courseDur+"<br/>");
            out.print("Comments: "+ comment);

            out.print("</body></html>");

}
```

}

- Instead of using web.xml to map the URL requests to the servlets, we are using '*@WebServlet()*' annotation to map the URL to the servlet.
- Once the form page is submitted, the '*doPost()*' method of the Servlet class will be invoked.
- The field values are submitted to the servlet in form of parameters in the HTTP servlet request. So, we can get those parameter values using '*getParameter()*' and '*getParameterValues()*' methods on the request object.

Explanation:

As we learned, to get the field value having only one value, '*getParameter()*' is used. So for the fields,

- First Name
- Phone Number
- Gender
- Course Duration
- Comments

'*getParameter()*' returns the respective parameter value as a String. But the 'Programming Languages' field is the type Checkbox, we can have multiple values selected for this checkbox. So '*getParameterValues()*' method returns all the selected values as an array. This way all the input data entered in the browser will pass through the request body to the respective servlets. After getting the parameter values, we can perform our own logic to process the information such as,

- Can perform validations on the data
- Can store these values to a database using JDBC
- Can pass the data to another resource like Servlet or JSP etc.,

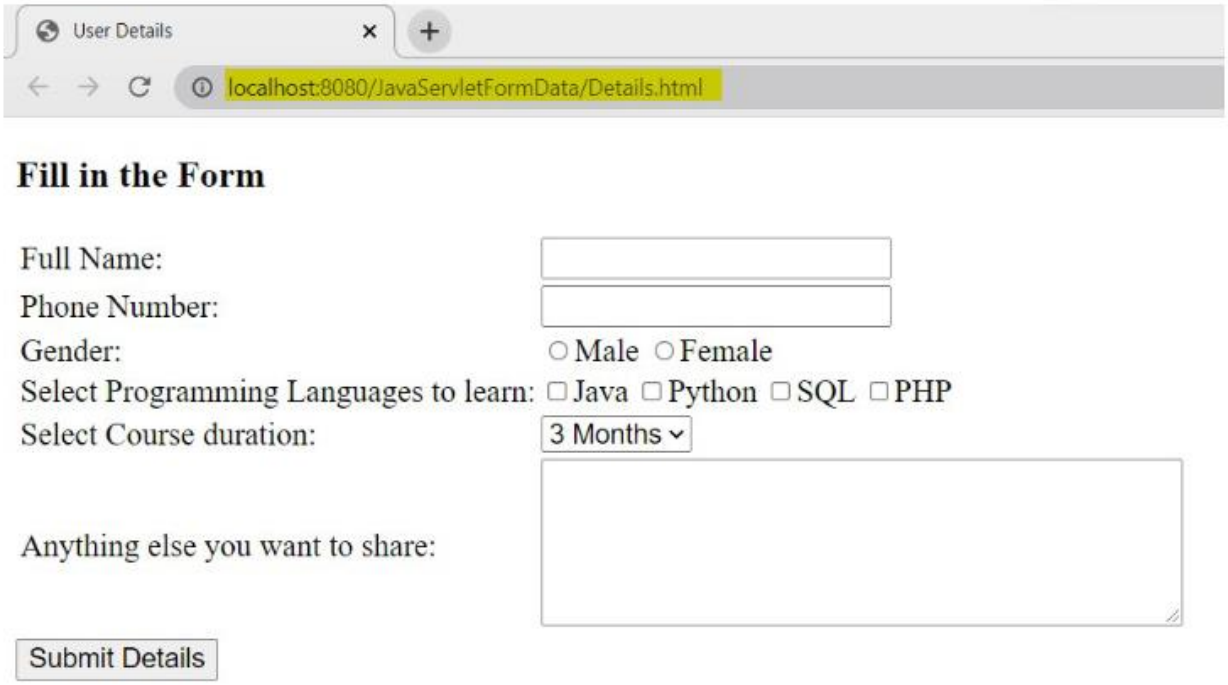
In this example, we are simply writing the input data we got from the request object to the browser through the response object. To achieve this, we are using the PrintWriter class object from java.io package.

- Set the content type of the response to '*text/html*', so that the response process the html tags provided through the PrintWriter object.

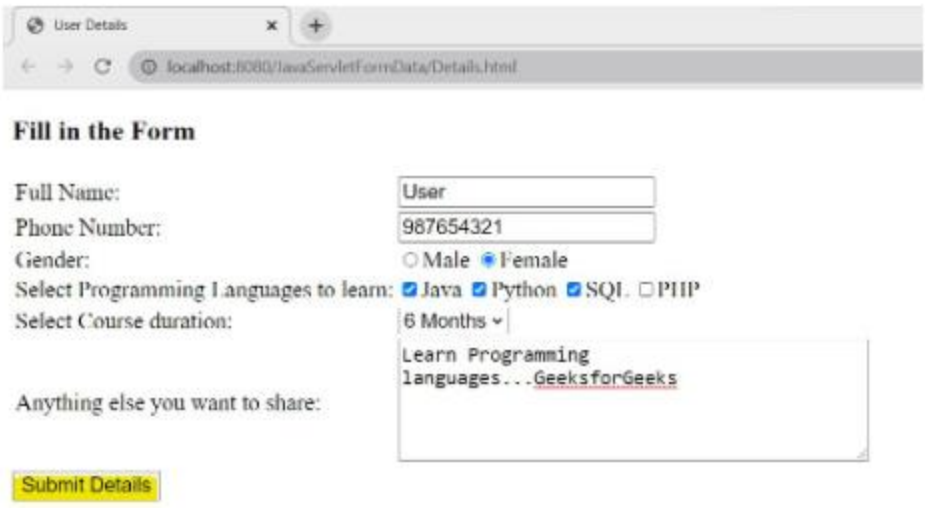
- To Print the data, use '*print*' statements and pass the respective variables.

Output:

- To Run the application, right-click on the project, **Run As -> Run on Server**
- Run the URL: ***http://localhost:8080/JavaServletFormData/Details.html*** in the browser to get the Initial form page.



The HTML form page will be displayed with all the fields.



- Enter all the values in respective fields and click on '*Submit Details*'.
- The details will be submitted to the servlet and the data is written to the response that displays on the browser.

- You can check changes in the URL that is being mapped to the servlet.

This way, we can handle the HTML form data in the Servlet classes using methods in HttpServletRequest Interface.