UNIT 3-SERVER SIDE PROGRAMMING

3.1 FORM GET AND POST ACTIONS

3.1.1 GET Method

- The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** (question mark) symbol as follows –

http://www.test.com/hello?key1 = value1&key2 = value2

- The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box. Never use the GET method if you have password or other sensitive information to pass to the server. The GET method has size limitation: only 1024 characters can be used in a request string.
- This information is passed using QUERY_STRING header and will be accessible through QUERY_STRING environment variable and Servlet handles this type of requests using **doGet()** method.

3.1.2 POST Method

- A generally more reliable method of passing information to a backend program is the POST method. This packages the information in exactly the same way as GET method, but instead of sending it as a text string after a ? (question mark) in the URL it sends it as a separate message. This message comes to the backend program in the form of the standard input which you can parse and use for your processing. Servlet handles this type of requests using **doPost()** method.

3.1.3 Reading Form Data using Servlet

Servlets 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.

3.1.4 GET Method Example using URL

Here is a simple URL which will pass two values to HelloForm program using GET method.

http://localhost:8080/HelloForm?first_name = ZARA&last_name = ALI

- Given below is the HelloForm.java servlet program to handle input given by web browser. We are going to use getParameter() method which makes it very easy to access passed information –

```
// Import required java

libraries import java.io.*;

import javax.servlet.*;

import

javax.servlet.http.*;

// Extend HttpServlet class

public class HelloForm extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse

   response) throws ServletException, IOException {

      // Set response content type

   response.setContentType("text/html

   "); PrintWriter out =

   response.getWriter();

   String title = "Using GET Method to Read Form Data";
```

```
String docType =

  "<!doctype html public \"-//w3c//dtd html 4.0 " + "transitional//en\">\n";
  out.println(docType + "<html>\n" +
    "<head><title>" + title + "</title></head>\n" +

    "<body bgcolor = \"#f0f0f0\">\n" +

      "<h1 align = \"center\">" + title + "</h1>\n" +

      "<ul>\n" +

        "  <li><b>First Name</b>: "

        +    request.getParameter("first_name")    +

        "\n" + "  <li><b>Last Name</b>: "

        + request.getParameter("last_name")  +  "\n"

      + "</ul>\n" +

    "</body>"           +

  "</html>"

  );

 }

}
```

- Assuming your environment is set up properly, compile HelloForm.java as follows –

$ javac HelloForm.java

- If everything goes fine, above compilation would produce HelloForm.class file. Next you would have to copy this class file in <Tomcat installationdirectory>/webapps/ROOT/WEB- INF/classes and create following entries in web.xml file located in <Tomcat-installation- directory>/webapps/ROOT/WEB-INF/

```
<servlet>

  <servlet-name>HelloForm</servlet-name>

  <servlet-class>HelloForm</servlet-class>

</servlet>

<servlet-mapping>

  <servlet-name>HelloForm</servlet-name>
```

<url-pattern>/HelloForm</url-pattern>

</servlet-mapping>

- Now type http://localhost:8080/HelloForm?first_name=ZARA&last_name=ALI in your browser's Location:box and make sure you already started tomcat server, before firing above command in the browser. This would generate following result –

Output

Using GET Method to Read Form Data

First Name: ZARA

Last Name: ALI

3.1.5 GET Method Example Using Form

Here is a simple example which passes two values using HTML FORM and submit button. We are going to use same Servlet HelloForm to handle this input.

```
<html>
  <body>
    <form action = "HelloForm" method = "GET">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>
  </body>
</html>
```

3.1.6 POST Method Example Using Form

- Let us do little modification in the above servlet, so that it can handle GET as well as POST methods. Below is HelloForm.java servlet program to handle input given by web browser using GET or POST methods.

```
// Import required java
libraries import java.io.*;
import javax.servlet.*;
```

```
import javax.servlet.http.*;
// Extend HttpServlet class
public class HelloForm extends HttpServlet {
  // Method to handle GET method request.
  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // Set response content type
    response.setContentType("text/html
    "); PrintWriter out =
    response.getWriter();
    String title = "Using GET Method to Read Form Data";
    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
          out.println(docTyp
      e + "<html>\n" +
        "<head><title>" + title + "</title></head>\n" +
        "<body bgcolor = \"#f0f0f0\">\n" +
          "<h1 align = \"center\">" + title + "</h1>\n" +
          "<ul>\n" +
            "  <li><b>First Name</b>: "
            +    request.getParameter("first_name")    +
            "\n" + "  <li><b>Last Name</b>: "
            + request.getParameter("last_name") + "\n"
          + "</ul>\n" +
        "</body>"
      "</html>"
    );
```

```
  }

  // Method to handle POST method request.

  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException {
    doGet(request, response);

  }

}
```

Now compile and deploy the above Servlet and test it using Hello.htm with the POST method as follows –

```
<html>

  <body>

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

      First Name: <input type = "text" name = "first_name">

      <br />

      Last Name: <input type = "text" name = "last_name" />

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

    </form>

  </body>

</html>
```

3.1.7 Get vs. Post

S.No	GET	POST
1	In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2	Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3	Get request can be bookmarked.	Post request cannot be bookmarked.
4	Get request is idempotent . It means second request will be ignored until response of first request is deliv-ered	Post request is non-idempotent.
5	Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.

Two common methods for the request-response between a server and client are:

- **GET**- It requests the data from a specified resource

- **POST**- It submits the processed data to a specified resource

Anatomy of Get Request

- The query string (name/value pairs) is sent inside the URL of a GET request:

GET/RegisterDao.jsp?name1=value1&name2=value2

- As we know that data is sent in request header in case of get request. It is the default request type. Let's see what information is sent to the server.

- Some other features of GET requests are:
  - ➢ It remains in the browser history
  - ➢ It can be bookmarked
  - ➢ It can be cached
  - ➢ It have length restrictions
  - ➢ It should never be used when dealing with sensitive data
  - ➢ It should only be used for retrieving the data

Anatomy of Post Request

The query string (name/value pairs) is sent in HTTP message body for a **POST request:**

POST/RegisterDao.jsp HTTP/1.1 Host:

www. javatpoint.com

name1=value1&name2=value2

As we know, in case of post request original data is sent in message body.

Let's see how information is passed to the server in case of post request.

- Some other features of POST requests are:

➢ This requests cannot be bookmarked

➢ This requests have no restrictions on length of data

➢ This requests are never cached

➢ This requests do not retain in the browser history