Assignment 5

→ **Aim :**

Understand servlet life cycle, create login page & apply proper validations with appropriate messages using doGet() & doPost() methods

→ **Objectives :**

1. It gives an introduction into the basic concepts of Java Servlets.
2. It gives understanding of how you can use Servlets
3. It gives an understanding of various web servers and server-side programming

→ **Theory :**

Before Servlets :

- CGI Scripts were used. (Common Gateway Interface)
- Client sends a request to server.
- Server starts the CGI script
- Script computes a result for server and quits.
- Server returns response to client.
- Another client sends a request
- Server starts the CGI script again

Etc....

Servlets

- A servlet is like an applet, but on the server side.
- Client sends a request to server.
- Server starts a servlet
- Servlet computes a result for server & does not quit. Server returns response to client.
- Another client sends a request.
- Server calls the server again
  Etc...

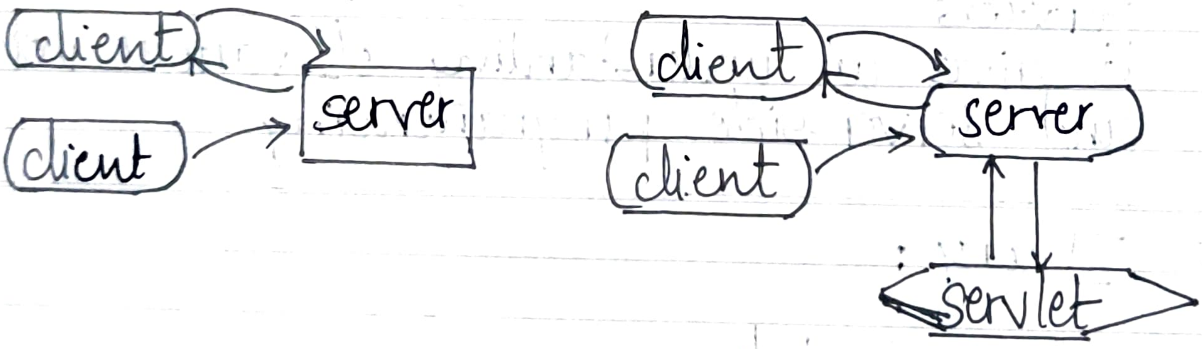

Fig. CGI vs Servlet

- The service () method is called by the container and service method invokes doGet(), doPost, doPut, doDelete, etc methods as appropriate
- A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified & it should be

handled by doGet() method.

Ex.
```
public void doGet (HttpServlet Request request,
                   HttpServletResponse response)
        throws: ServletException, IOException
{
    //Servlet code
}
```

- The post request results from an HTML form that specifically lists POST as the Method and it should be handled by doPost() method.

Ex.
```
public void doPost (HttpServletRequest request,
     HttpServletResponse response)
        throws ServletException, IOException
{
 // Servlet Code
}
```

- The destroy() method is used only once at the end of life cycle of the servlet. This method gives your servlet a chance to to close all database connections, halt background activities/threads, write cookie lists and perform cleanup activities.

Ex.
```
public void destroy () {
    // finalization code
}
```

Ex. Sample Hello World code.

```
//Import required java libraries

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
//Extend HttpServlet class.

public class HelloWorld extends HttpServlet {
    private String message;
  public void init() throws ServletException {
      message = "Hello World";
  }

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

    response.setContentType("text/html");
   PrintWriter out = response.getWriter();
    out.println("<h1>"+message+"</h1>");
  }
  public void destroy() {
  } // do nothing
}
```

→ Conclusion :

The servlet features have been studied & used for creating dynamic & interactive websites.