

Servlets

Web Application:

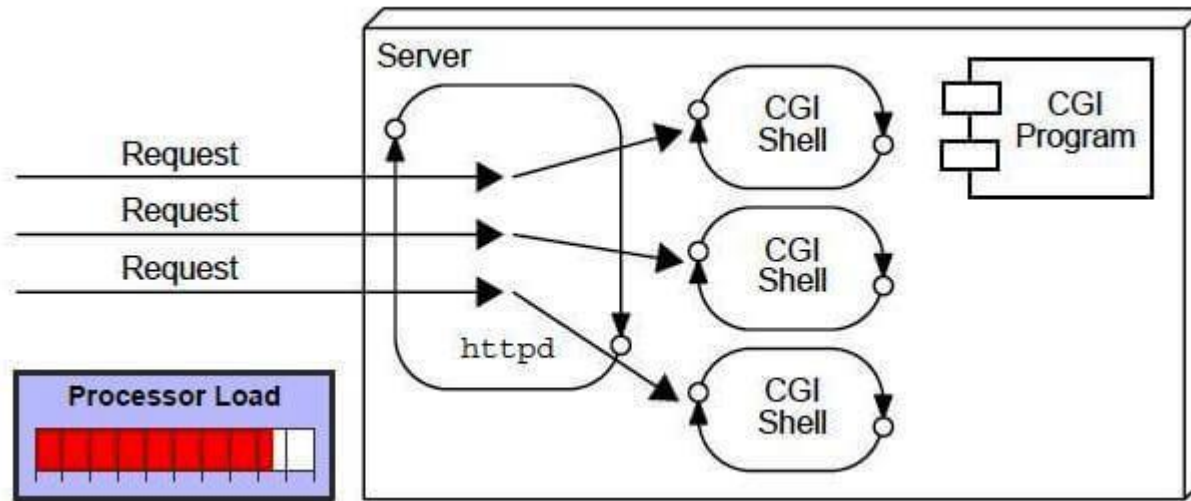
- A web application is an application accessible from the web.
- A web application is composed of web components like Servlet, JSP, etc. and other elements such as HTML, CSS, and JavaScript.
- The web components typically execute in Web Server and respond to the HTTP request.

Common Gateway Interface:

- CGI (Common Gateway Interface) scripting language is a basic server-side programming language. which is used to develop dynamic web pages.
- CGI technology enables the web server to call an external program and pass HTTP request information to process the request.

Common Gateway Interface (CGI)





In CGI, For each request, it starts a new process.

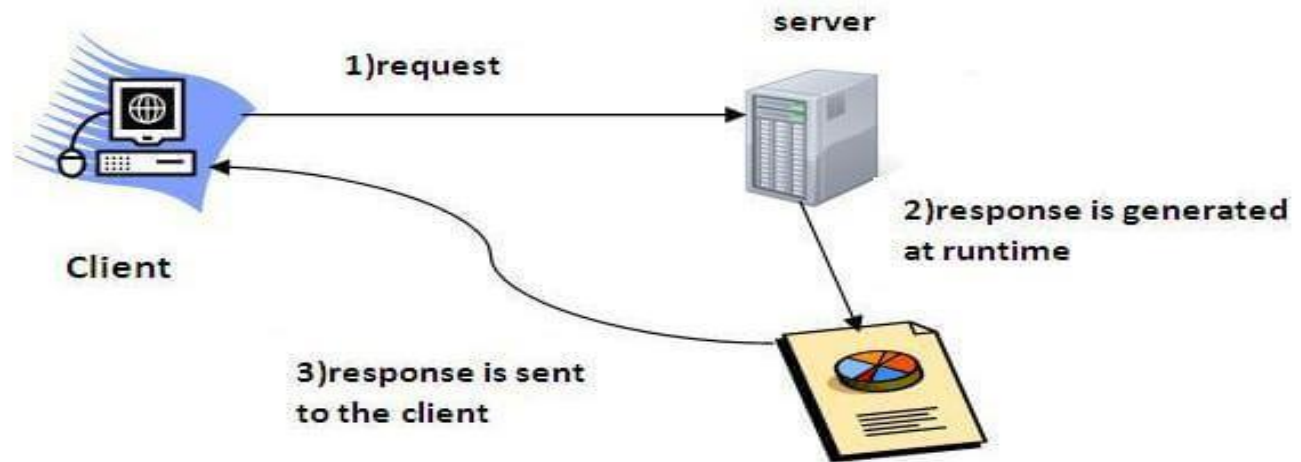
- Each new process requires separate memory, CPU time so processor load will increase.

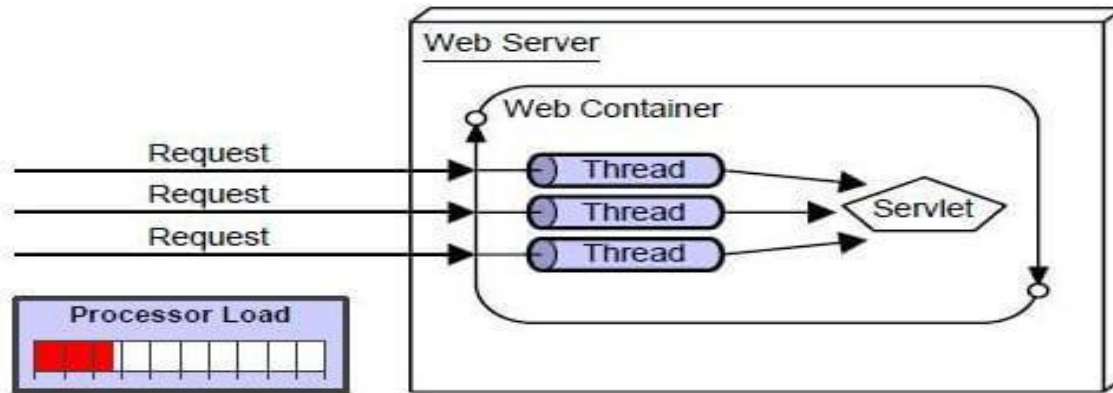
There are many problems in CGI technology:

- If the number of clients increases, it takes more time for sending the response.
- For each request, it starts a process, and the web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.

Servlets:

- Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).
- Servlet technology is robust and scalable because of java language.
- Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests.





- The web container creates threads for handling the multiple requests to the Servlet.
- Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.

The advantages of Servlet are as follows:

- **Better performance:** because it creates a thread for each request, not process.
- **Portability:** because it uses Java language.
- **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
- **Secure:** because it uses java language.

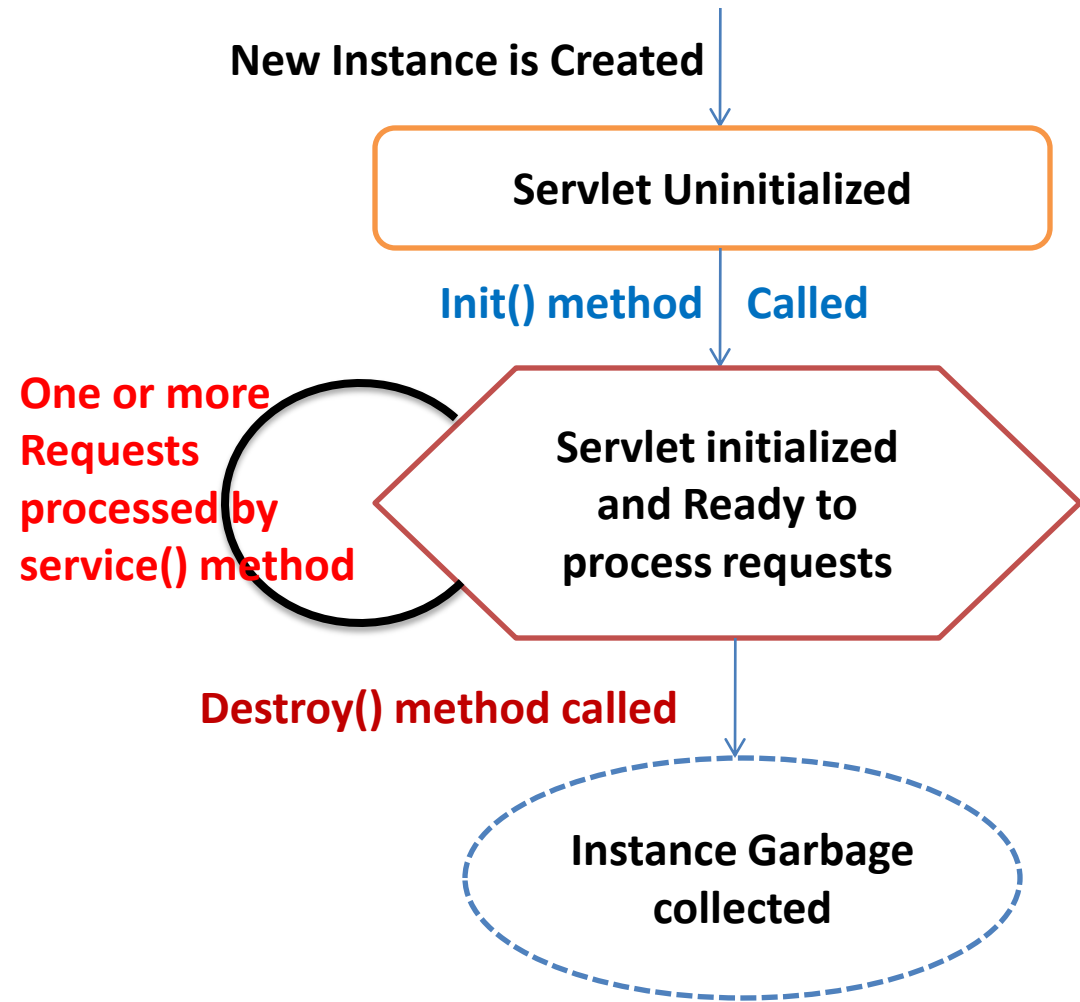
Servlet Vs CGI

Servlet	CGI (Common Gateway Interface)
Servlets are portable	CGI is not portable.
In Servlets each request is handled by lightweight Java Thread	IN CGI each request is handled by heavy weight OS process
In Servlets, Data sharing is possible	In CGI, data sharing is not available.
Servlets can link directly to the Web server	CGI cannot directly link to Web server.
Session tracking and caching of previous computations can be performed	Session tracking and caching of previous computations cannot be performed
Automatic parsing and decoding of HTML form data can be performed.	Automatic parsing and decoding of HTML form data cannot be performed.
Servlets can read and Set HTTP Headers	CGI cannot read and Set HTTP Headers
Servlets can handle cookies	CGI cannot handle cookies
Servlets can track sessions	CGI cannot track sessions
Servlets is inexpensive than CGI	CGI is more expensive than Servlets

Lifecycle of Servlet:

Lifecycle of a servlet describes how and when a servlet is loaded, initialized, able to handle requests and unloaded.

- There are 3 methods in lifecycle of servlet, those are
 - **init()** : This method initializes the servlet.
 - **service()** : This method process the http requests.
 - **destroy()**: This method is used to destroy the servlet.



Create a servlet example

The servlet example can be created by three ways:

- By implementing Servlet interface,
- By inheriting GenericServlet class, (or)
- By inheriting HttpServlet class

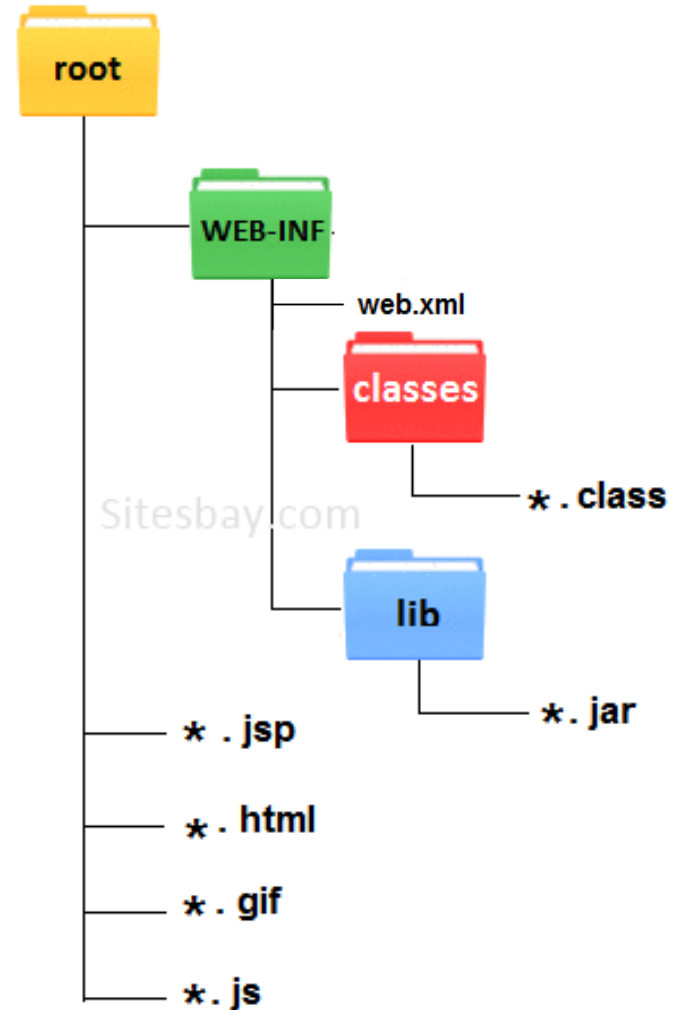
Here, we are going to use **tomcat server** in this example.

The steps are as follows:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server
6. Access the servlet

1. Create a directory structure:

- The **directory structure** defines that where to put the different types of files so that web container may get the information and respond to the client.
- Let's see the directory structure that must be followed to create the servlet.



2.Create a servlet:

```
import java.io.*;
import javax.servlet.*;
public class First extends GenericServlet
{
    public void service(ServletRequest req,ServletResponse res)
    throws IOException,ServletException{
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>Welcome to servlet</b>");
        out.print("</body></html>");
    }
}
```

First.java

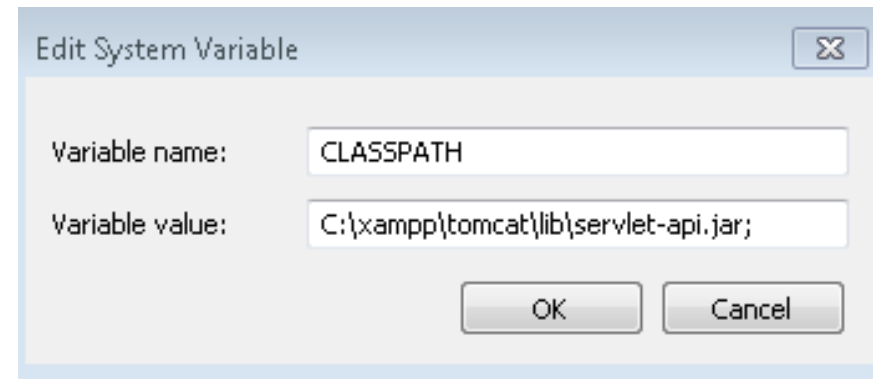
3.Compile the servlet:

- Servlet can be compiled by using jar file
- Jar file are required to be loaded.

Servlet-api.jar

Two ways to load the jar file

1. set classpath
2. paste the jar file in JRE/lib/ext folder



- ✓ Put the java file in any folder. After compiling the java file, paste the class file of servlet in **WEB-INF/classes** directory.

4.Create a deployment description:

- Deployment description is an XML file with a name web.xml
- From this xml file web container gets the information about the servlet to be invoked

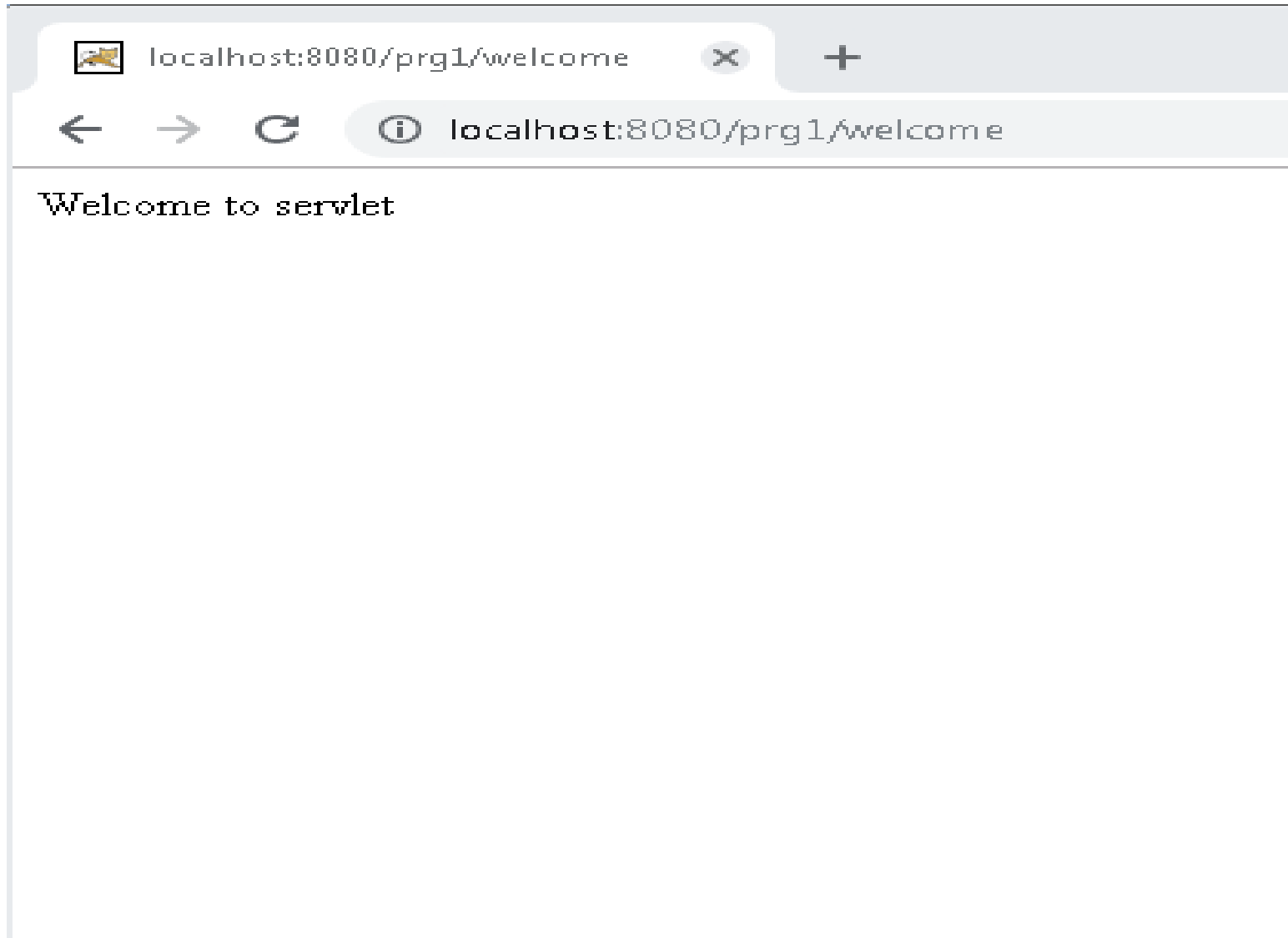
Web.xml

```
<web-app>
  <servlet>
    <servlet-name>hello</servlet-name>
    <servlet-class>First</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>hello</servlet-name>
    <url-pattern>/welcome</url-pattern>
  </servlet-mapping>
</web-app>
```

5.Start the server and deploy the project:

- There are two ways to deploy the project
 - Deployment
 - Copy project into `c://xampp/tomcat/web-apps`
 - Start Tomcat server
 - By using Xampp control panel
 - By running **catalina_start** batch file.
(`c://xampp/tomcat/`)

6. Access the servlet:



Reading Servlet Parameters:

- The Servlet Request class includes methods that allow you to read the names and values of parameters that are included in a client request.

Example:

“index.html”

```
<html>
```

```
<body>
```

```
<form action="home" method="get">
```

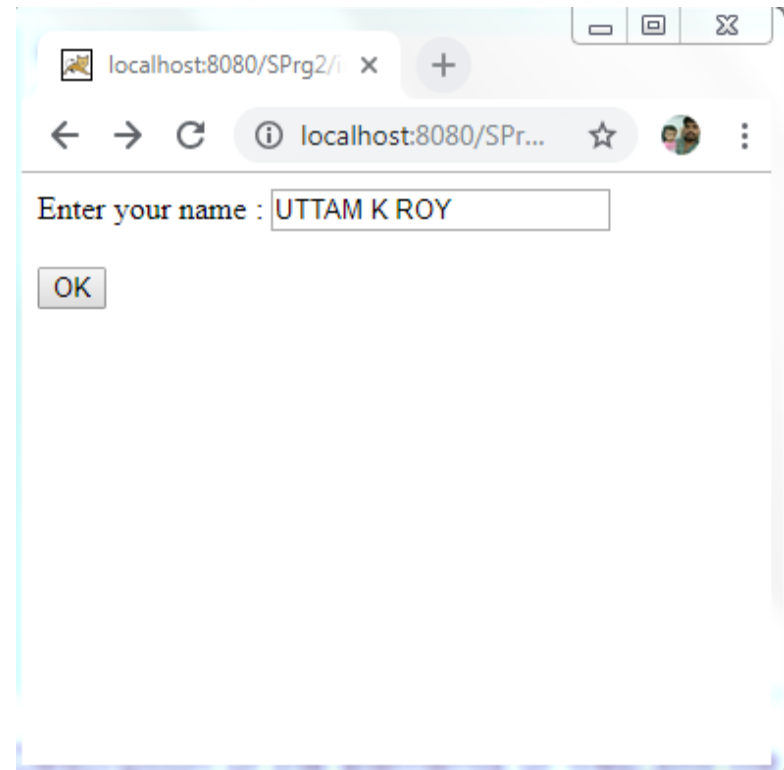
Enter your name :

```
<input type="text" name="name"><br>
```

```
<input type="submit" value="OK">
```

```
</body>
```

```
</html>
```



“ReadingPar.java”

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class ReadingPar extends HttpServlet{
    public void doGet(HttpServletRequest req,HttpServletResponse
        res)throws ServletException,IOException
    {
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        String name=req.getParameter("name");
        pw.println("Welcome "+name);
    }
}
```

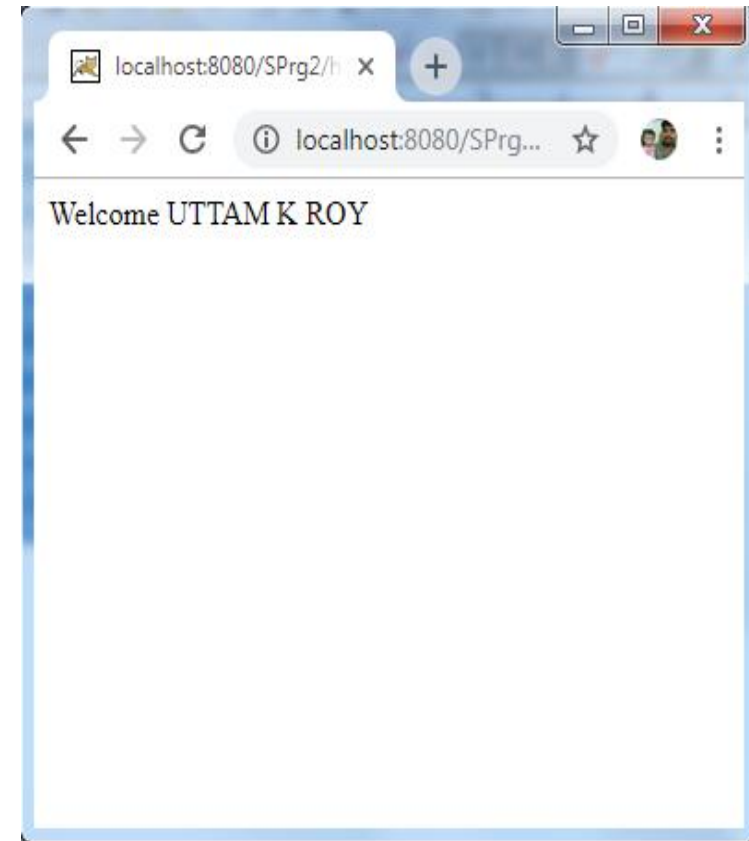
Compilation:

Javac ReadingPar.java

After compiling the java file, paste the class file of servlet in **WEB-INF/classes** directory.

“web.xml”

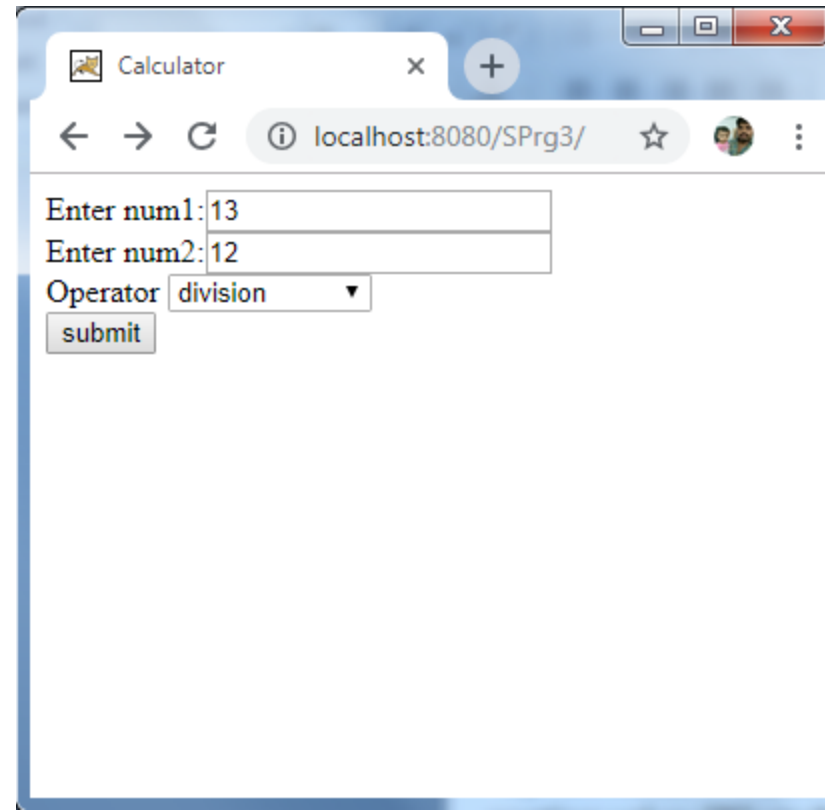
```
<web-app>
<servlet>
<servlet-name>SPrg2</servlet-name>
<servlet-class>ReadingPar</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>SPrg2</servlet-name>
<url-pattern>/home</url-pattern>
</servlet-mapping>
</web-app>
```



Example:

"index.html"

```
<html>
<head>
<title>Calculator</title>
</head>
<body>
<form action="/SPrg3/evaluate" method="post" name="frm">
Enter num1:<input name="txt1" type="text" /><br>
Enter num2:<input name="txt2" type="text" /><br>
Operator
<select name="op">
<option value="+">Addition</option>
<option value="-">Subtraction</option>
<option value="*">multiplication</option>
<option value="/">division</option>
<option value="%">remainder</option>
</select><br>
<input type="submit" value="submit" />
</form>
</body>
</html>
```



“Calculator.java”

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Calculator extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        String n1 = request.getParameter("txt1");
        String n2 = request.getParameter("txt2");
        String op = request.getParameter("op");
```

```
if(op.equals("+")){
    out.println("SUM IS :" +(Integer.parseInt(n1) + Integer.parseInt(n2)));
}
else if(op.equals("-")){
    out.println("SUB IS :" +(Integer.parseInt(n1) - Integer.parseInt(n2)));
}
else if(op.equals("*")){
    out.println("MUL IS :" +(Integer.parseInt(n1) * Integer.parseInt(n2)));
}
else if(op.equals("%")){
    out.println("MOD IS :" +(Integer.parseInt(n1) % Integer.parseInt(n2)));
}
else{
    out.println("DIV IS :" +(Integer.parseInt(n1) / Integer.parseInt(n2)));
}
} }
```

“web.xml”

```
<web-app>
```

```
<servlet>
```

```
<servlet-name>SPrg3</servlet-name>
```

```
<servlet-class>Calculator</servlet-class>
```

```
</servlet>
```

```
<servlet-mapping>
```

```
<servlet-name>SPrg3</servlet-name>
```

```
<url-pattern>/evaluate</url-pattern>
```

```
</servlet-mapping>
```

```
</web-app>
```

