CSC584 Enterprise Programming

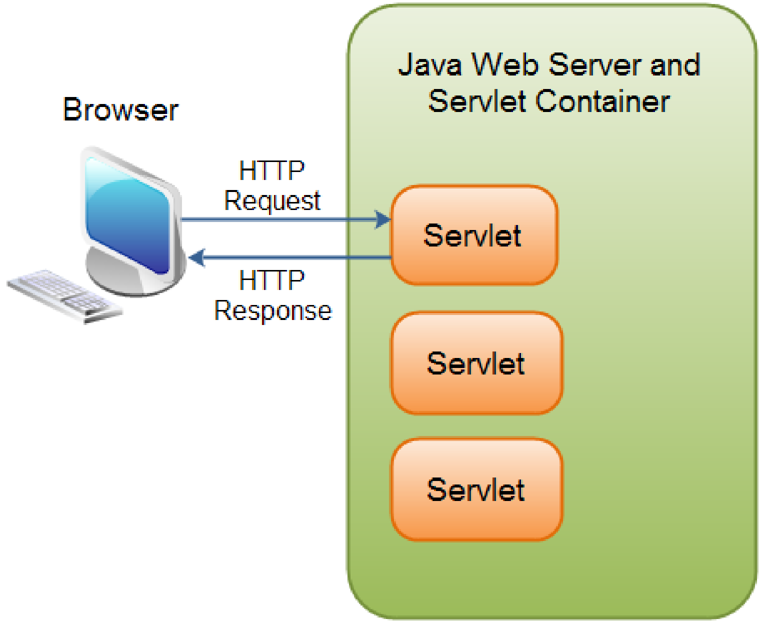
CHAPTER 3 – SERVLET

Chapter 3 Outline

- Creating & Running Servlets
- The Servlet API
- HTML forms
- Session tracking
- Database programming in servlets

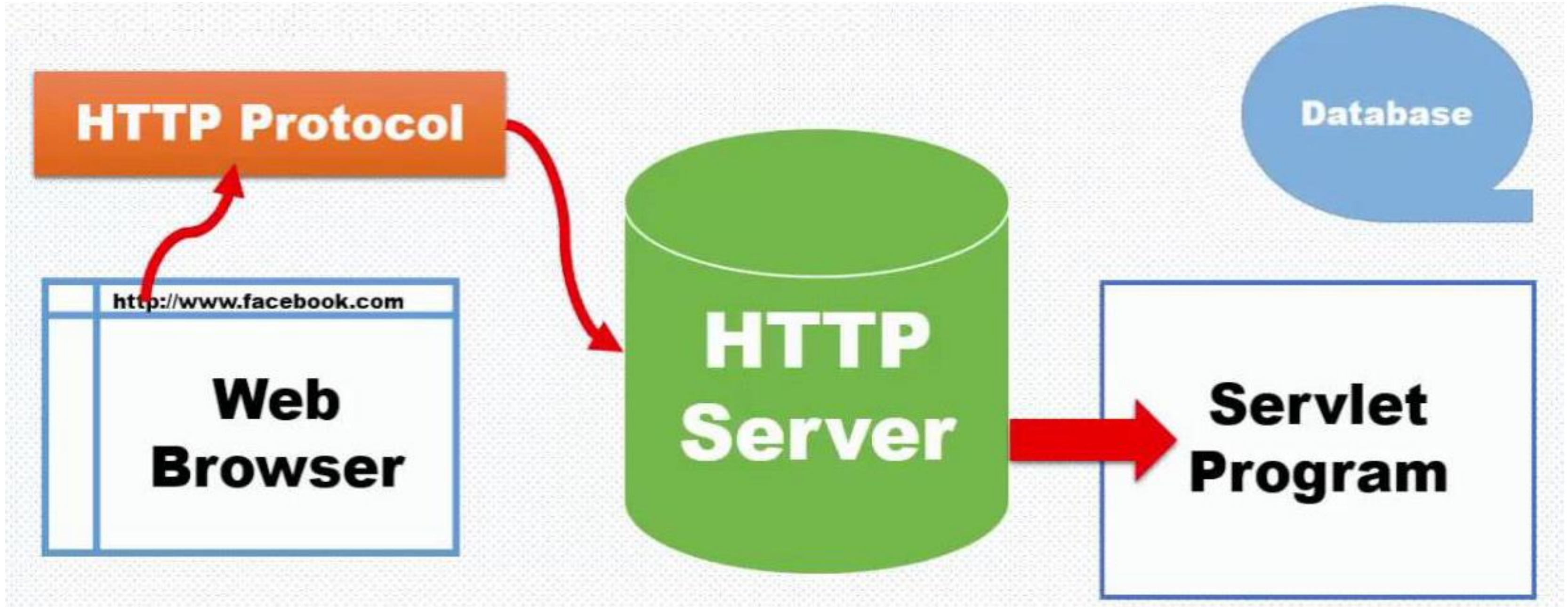
Understand the concept of servlets

- Servlet technology is primarily designed for use with the HTTP protocol of the Web.
- Java Servlets are programs that **run on a Web server.**
- Java servlets can be used to process client requests or produce dynamic Web pages.



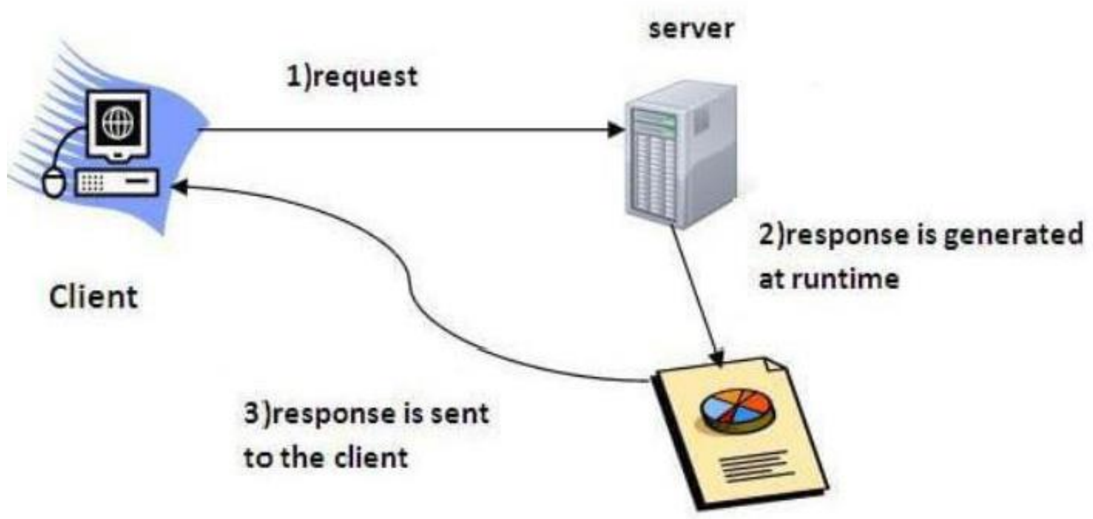
What are Servlets?

- Servlets run on server and act as a middle layer between requests coming from a **Web browser** and **databases**.



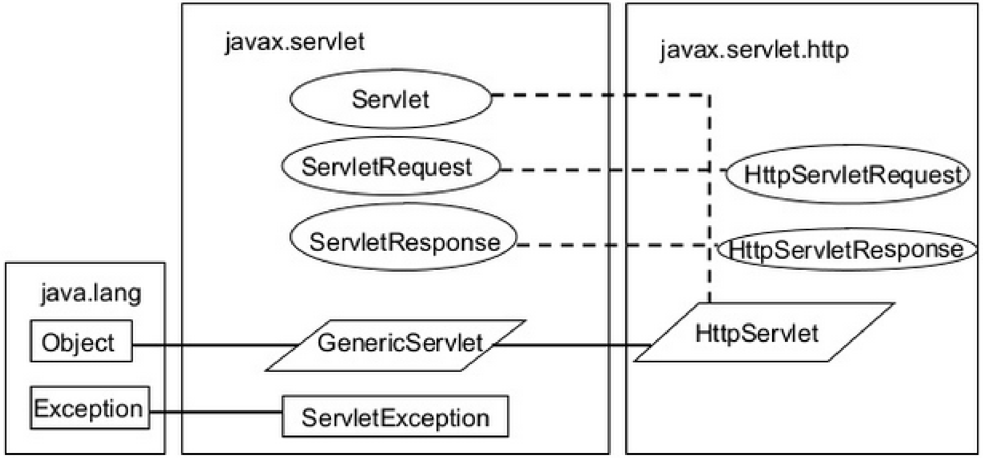
What are Servlets?

- Units of Java code that run server-side.
  - Run in *containers* (provide context)
  - Helps with client-server communications
  - Not necessarily over HTTP
  - But usually over HTTP (we'll focus here)


server
1)request
2)response is generated at runtime
Client
3)response is sent to the client

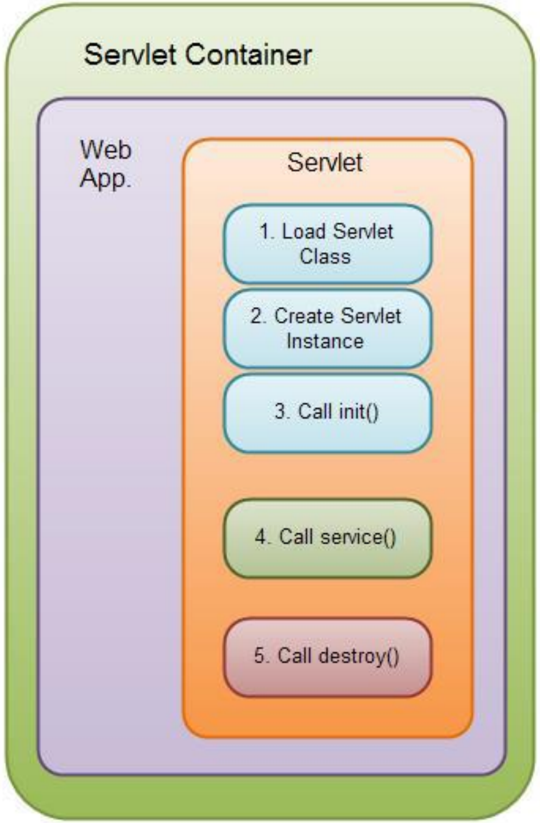
What are Servlets?

- A servlet is any class that implements the javax.servlet.Servlet interface
  - In practice, most servlets extend the javax.servlet.http.HttpServlet class
  - Some servlets extend javax.servlet.GenericServlet instead
- Servlets usually lack a main method, but must implement or override certain other methods



What is the life-cycle of a servlet?

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



Servlets

- A servlet is run on the server side

Client sends a request to server

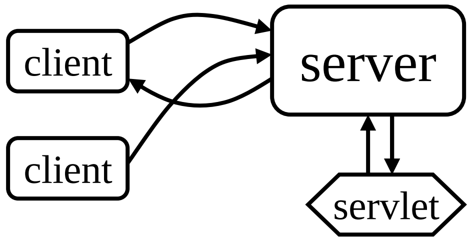
Server starts a servlet

Servlet computes a result for server and *does not quit*

Server returns response to client

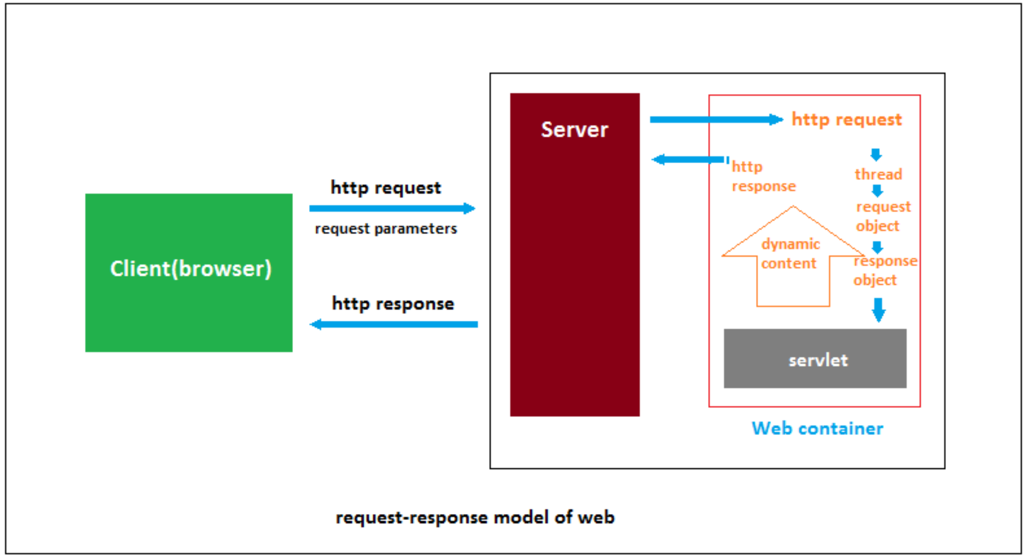
Another client sends a request

Server calls the servlet again



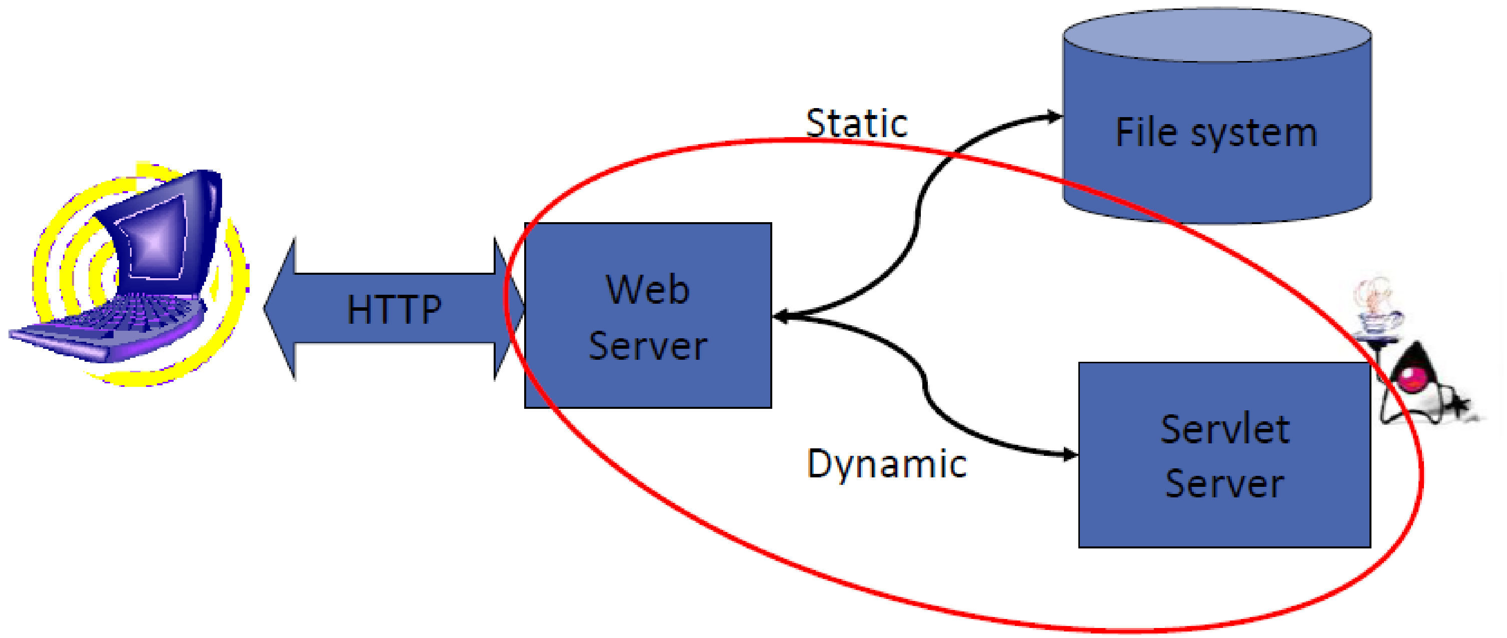
Why are Servlets?

- Web pages with <u>dynamic content</u>
- Easy coordination between Servlets to make <u>Web applications</u>
- Containers support many features
  - Sessions, persistence, resource management (e.g., database connections), security, etc.



request-response model of web

Where are Servlets?



Tomcat = Web Server + Servlet Server

Do you know?

1. What is the web application?

   A Web application is **an application program that is stored on a remote server and delivered over the Internet through a browser interface**

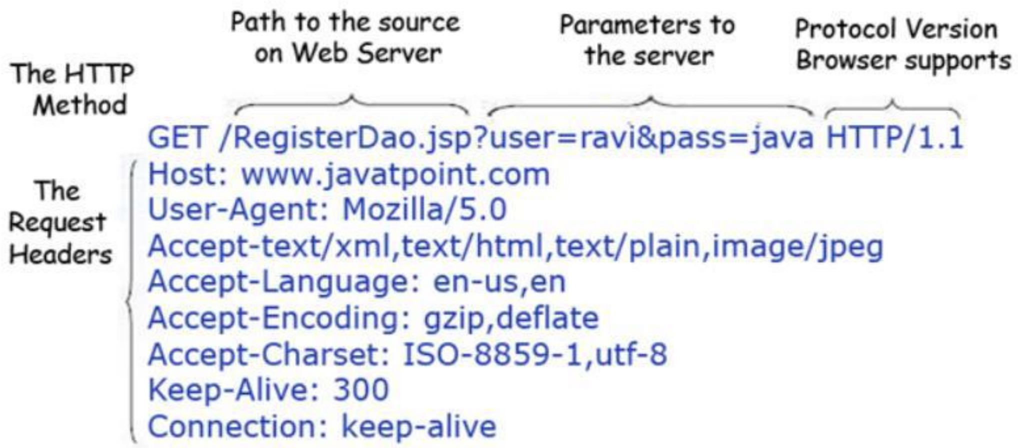
2. What is the difference between Get and Post request?
   - GET – it requests the data from a specified resource
   - POST – It submits the processed data to specified resource

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
```

- 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:


The HTTP Method
Path to the source on Web Server
Parameters to the server
Protocol Version Browser supports
The Request Headers

```
GET /RegisterDao.jsp?user=ravi&pass=java HTTP/1.1
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept-text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive
```

Anatomy of Post Request

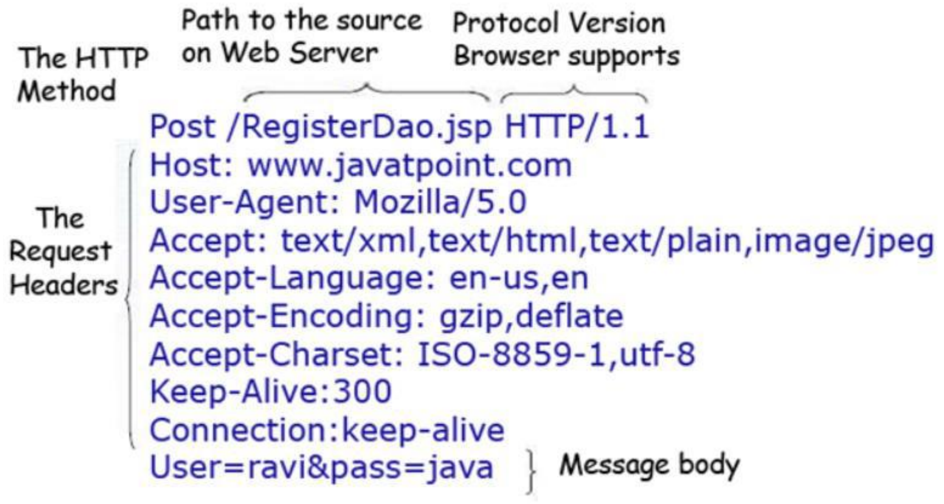
- The query string (name/value pairs) is sent in HTTP message body for a POST request:

```
POST/RegisterDao.jsp HTTP/1.1

Host: www. javatpoint.com

name1=value1&name2=value2
```

- 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.



```
Post /RegisterDao.jsp HTTP/1.1
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive:300
Connection:keep-alive
User=ravi&pass=java
```

The HTTP Method
Path to the source on Web Server
Protocol Version Browser supports
The Request Headers
Message body

Differences between the Get and Post request

HTTP GET

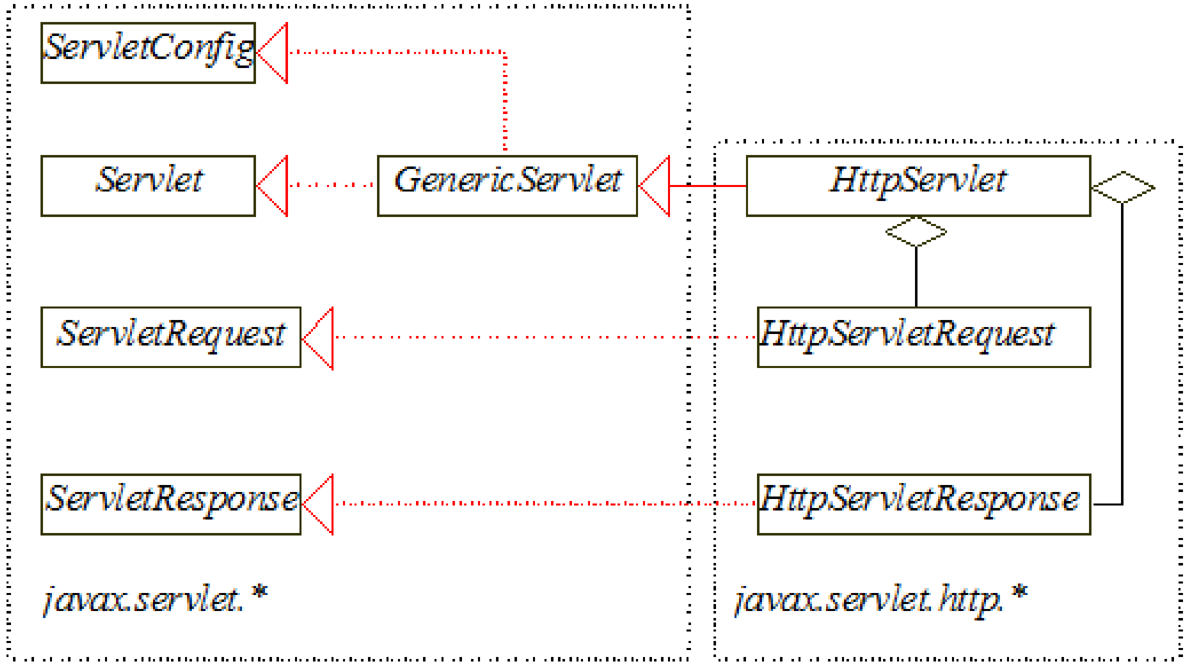
- Data is sent in header
- Limited amount data can be sent
- Not secured (data is exposed in URL bar)
- Can be bookmarked

HTTP POST

- Data is sent in body
- Large amount data can be sent
- Secured (data is not exposed in URL bar)
- Cannot be bookmarked

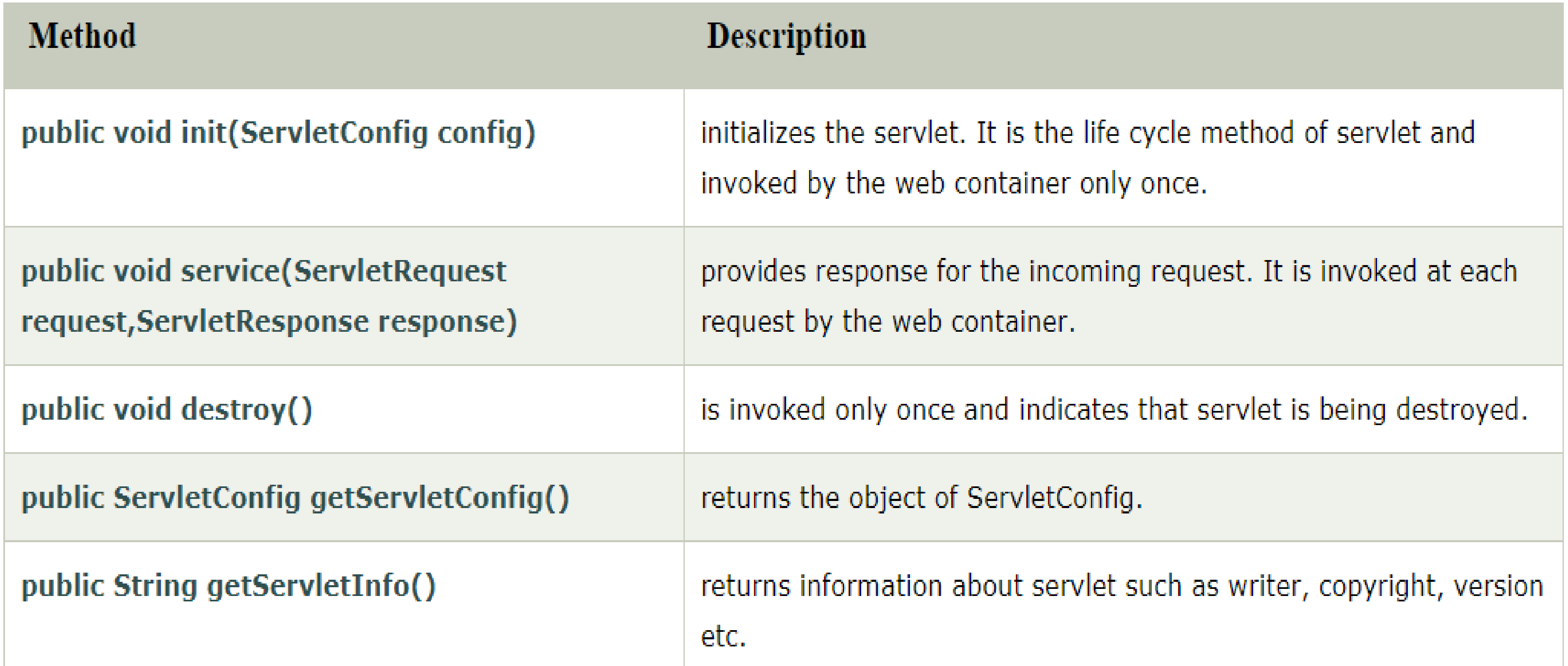
The Servlet API

The servlet API provides the interfaces and classes that support servlets. These interfaces and classes are grouped into two packages: javax.servlet, and javax.servlet.http.



The Servlet Interface

Servlet interface provides common behavior to all servlets

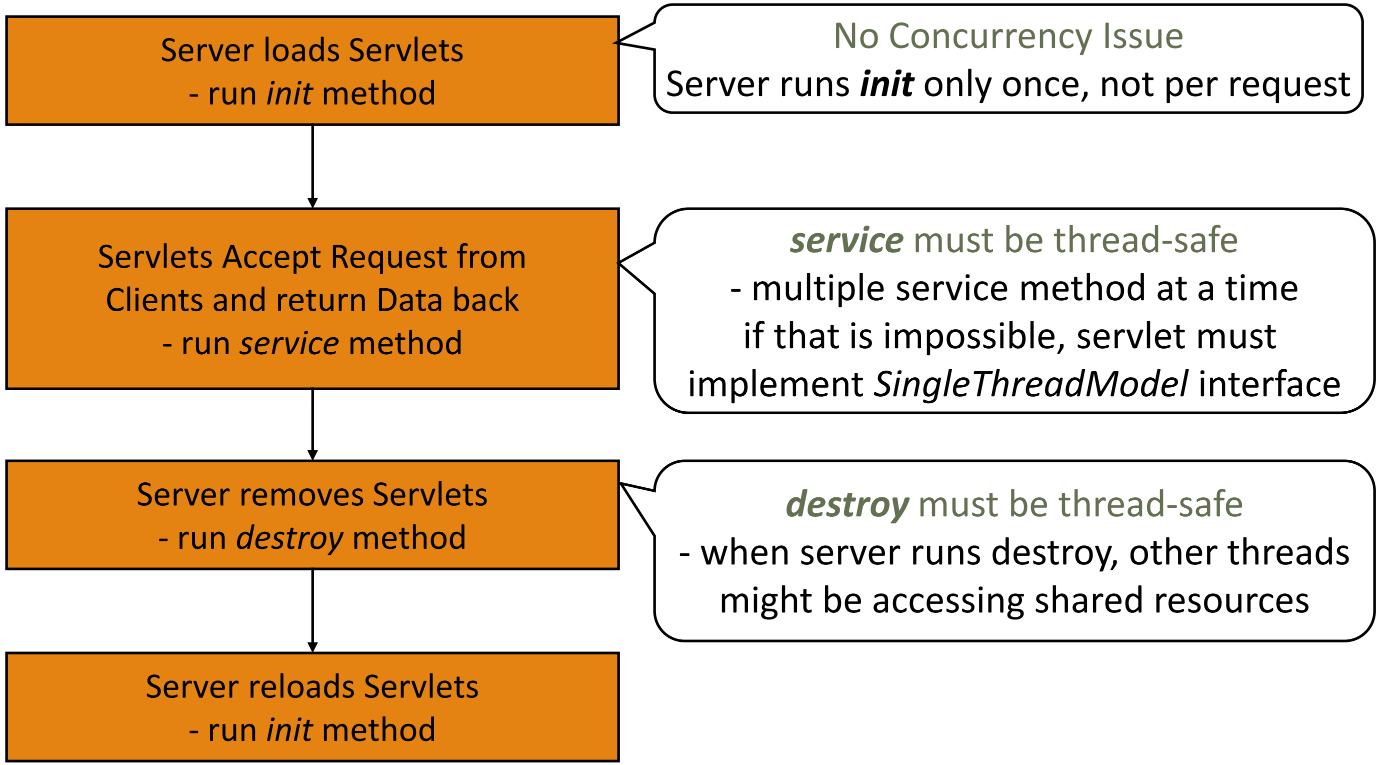


Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Lifecycle

The important methods of the Servlet interface are as follows:

- **init**
- **service**
- **destroy**



Server loads Servlets
- run *init* method

No Concurrency Issue
Server runs ***init*** only once, not per request

Servlets Accept Request from Clients and return Data back
- run *service* method

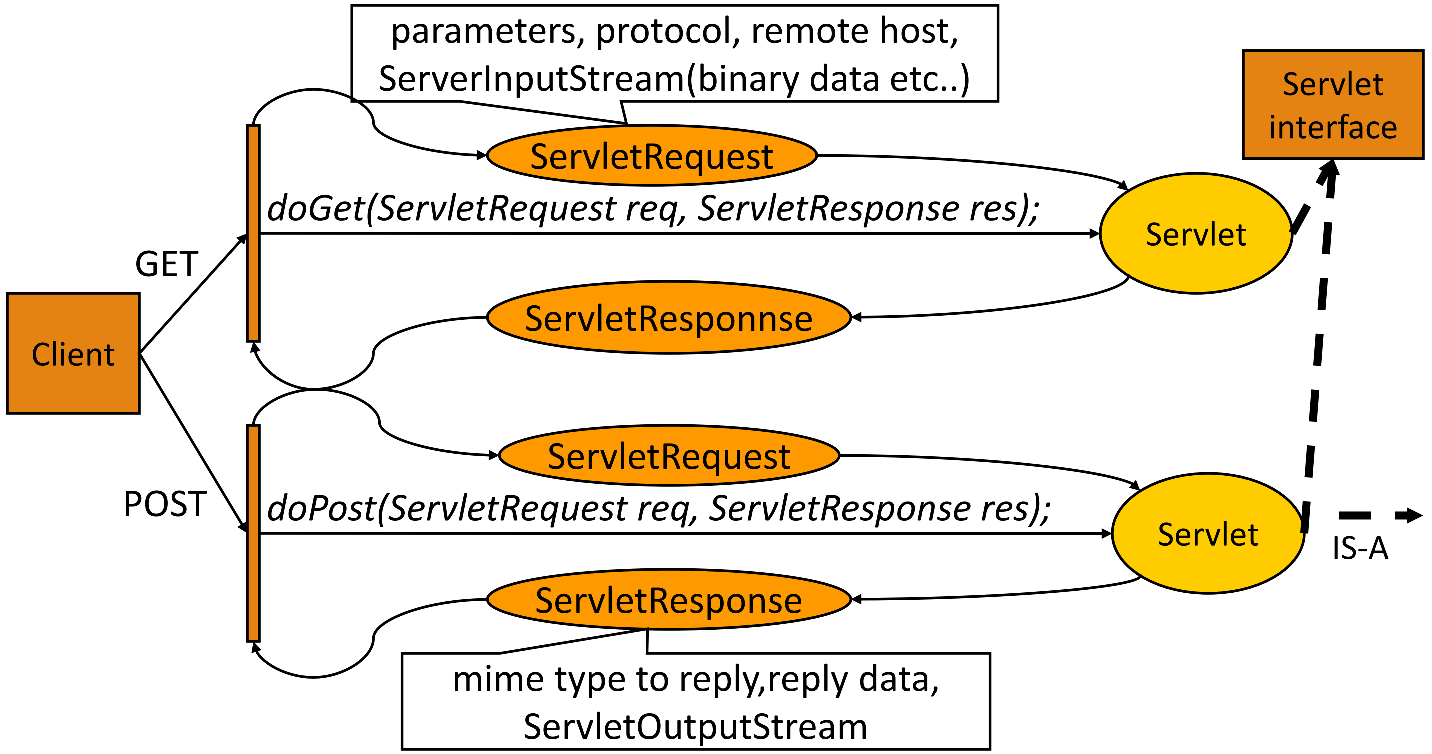
service must be thread-safe
- multiple service method at a time
if that is impossible, servlet must implement *SingleThreadModel* interface

Server removes Servlets
- run *destroy* method

destroy must be thread-safe
- when server runs destroy, other threads might be accessing shared resources

Server reloads Servlets
- run *init* method

Servlet Architecture Overview
- HTTP servlets


parameters, protocol, remote host, ServerInputStream(binary data etc..)
Servlet interface
ServletRequest
doGet(ServletRequest req, ServletResponse res);
Servlet
GET
Client
ServletResponnse
ServletRequest
POST
doPost(ServletRequest req, ServletResponse res);
Servlet
IS-A
ServletResponse
mime type to reply,reply data, ServletOutputStream

The HTTPServlet Class

1. The HttpServlet class defines a servlet for the HTTP protocol. It extends GenericServlet and implements the service method.

2. The service method is implemented as a dispatcher of HTTP requests. The HTTP requests are processed in the following methods: doGet, doPost, doDelete, doPut, doOptions, and doTrace. All these methods have the same signature as follows:

```
protected void doXxx(HttpServletRequest req,
HttpServletResponse resp) throws
ServletException, java.io.IOException
```

The HttpServletRequest Interface

1. Every doXxx method in the HttpServlet class has an argument of the HttpServletRequest type, which is an object that contains HTTP request information including **parameter name and values, attributes, and an input stream.**

2. HttpServletRequest is a sub interface of ServletRequest.ServletRequest defines a more general interface to provide information for all kinds of clients.

The HttpServletResponse Interface

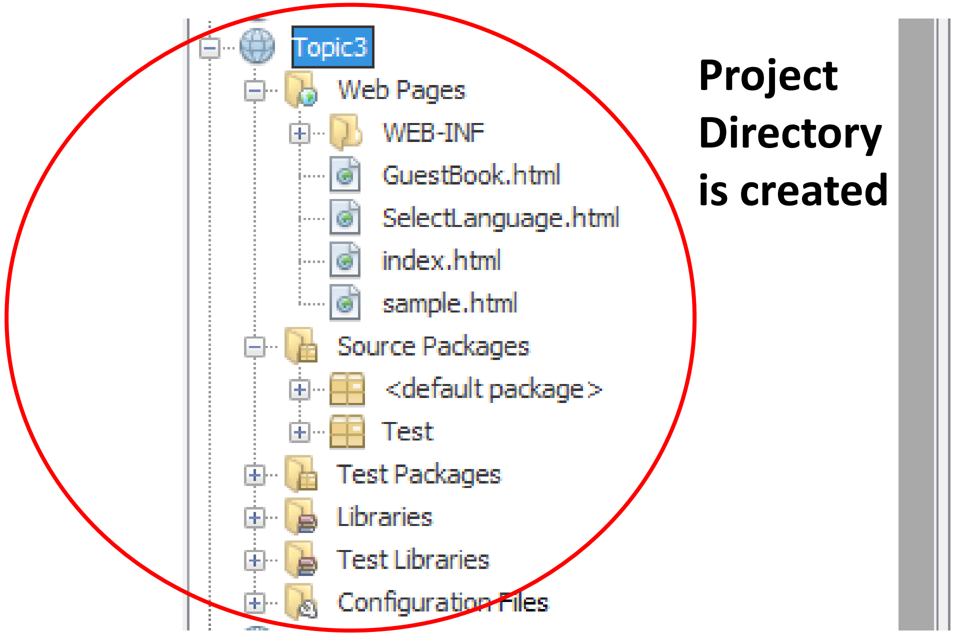
1. Every doXxx method in the HttpServlet class has an argument of the HttpServletResponse type, which is an object that assists a servlet in sending a response to the client.
2. HttpServletResponse is a subinterface of ServletResponse. ServletResponse defines a more general interface for sending output to the client.

Steps to create a servlet

1. Create a **Java Web** project
2. Create a **Servlet**
3. Check the deployment descriptor (web.xml)
4. Start the server and deploy the project
5. Access the servlet

1. Create a project

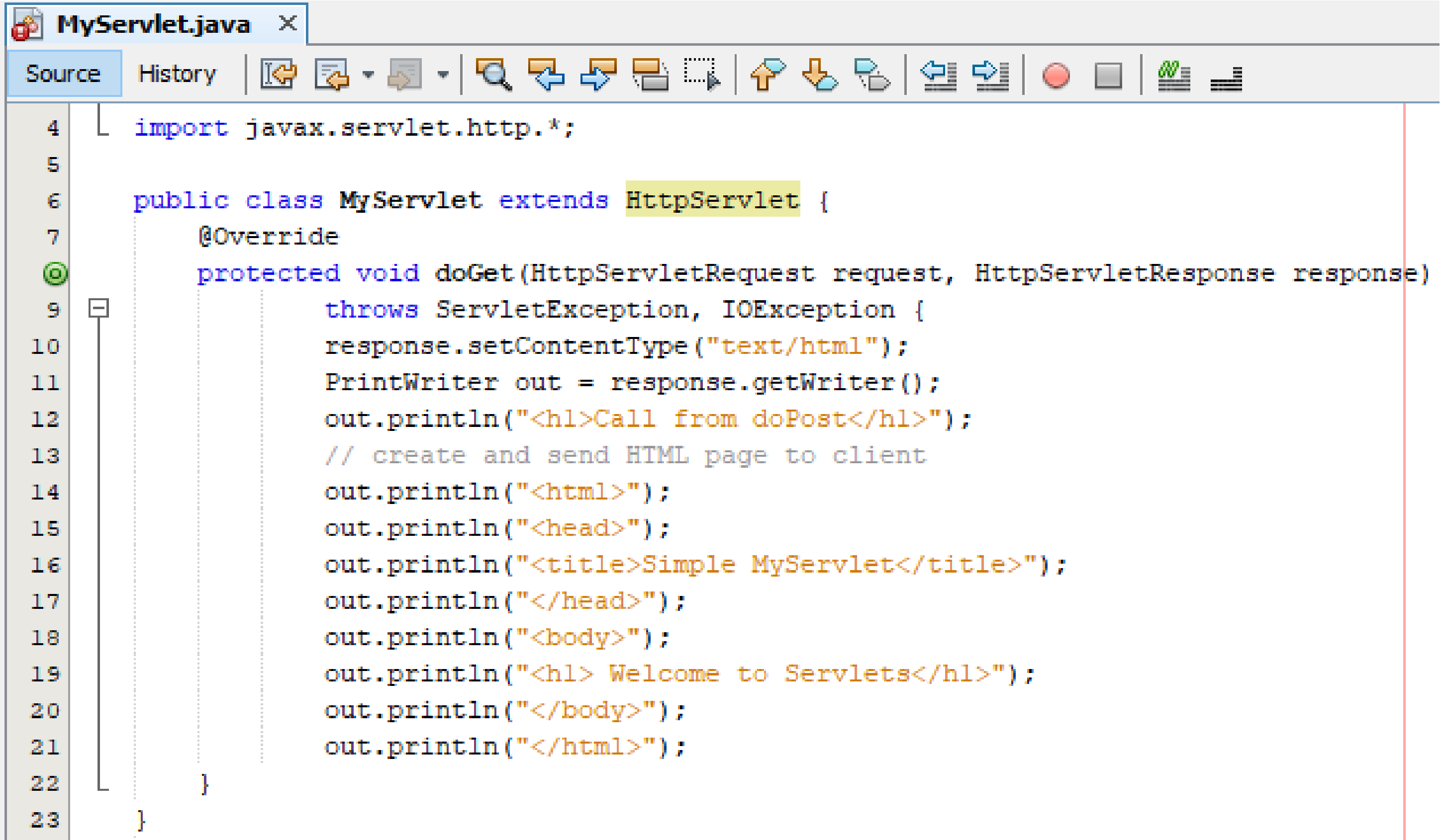
- The **directory structure in the project** defines that where to put the different types of files so that web container may get the information and respond to the client



Project Directory is created

2. Create a Servlet

- Create a servlet that extends the HttpServlet class. In this example, we are inheriting the HttpServlet class and providing the implementation of the doGet() method.



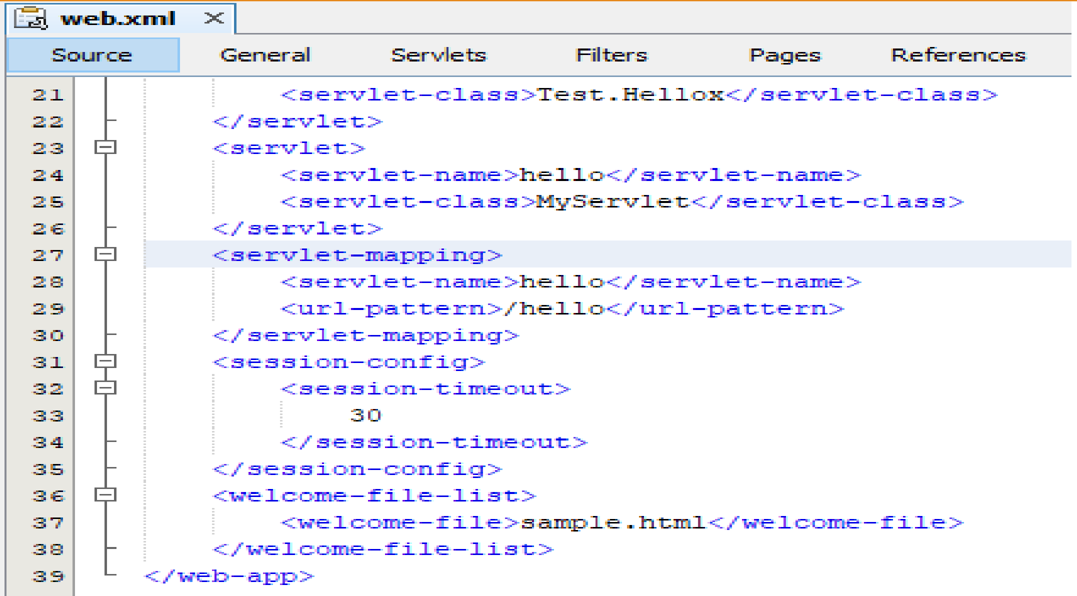
```java
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<h1>Call from doPost</h1>");
            // create and send HTML page to client
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Simple MyServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1> Welcome to Servlets</h1>");
            out.println("</body>");
            out.println("</html>");
    }
}
```

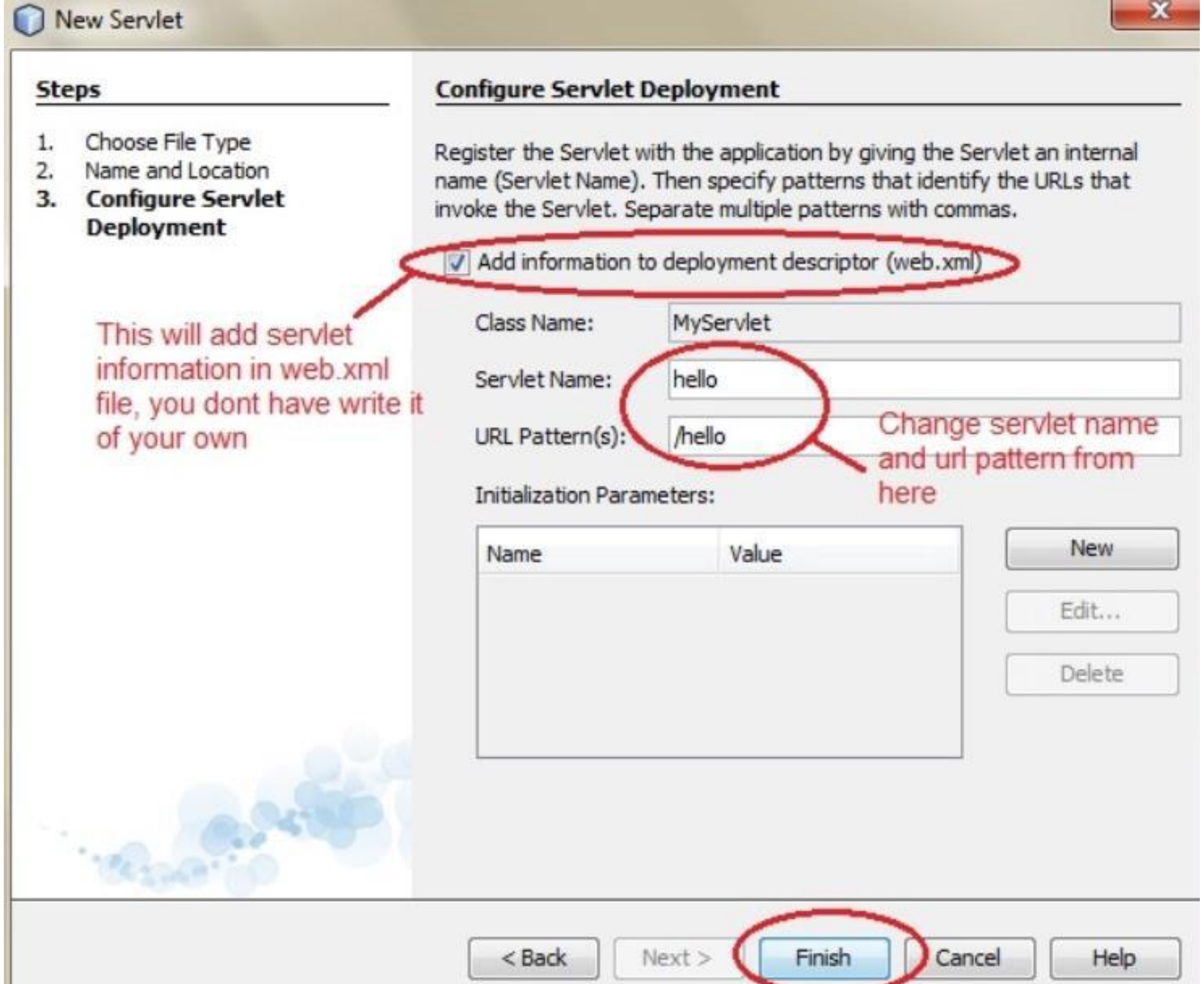
3. Configure the deployment descriptor

- The **deployment descriptor** is an xml file, from which Web Container gets the information about the servlet to be invoked.

<web-app> represents the whole application.
<servlet> is sub element of <web-app> and represents the servlet.
<servlet-name> is sub element of <servlet> represents the name of the servlet.
<servlet-class> is sub element of <servlet> represents the class of the servlet.
<servlet-mapping> is sub element of <web-app>. It is used to map the servlet.
<url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.



```
            <servlet-class>Test.Hellox</servlet-class>
        </servlet>
        <servlet>
            <servlet-name>hello</servlet-name>
            <servlet-class>MyServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>hello</servlet-name>
            <url-pattern>/hello</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>sample.html</welcome-file>
        </welcome-file-list>
    </web-app>
```



New Servlet

Steps

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

[✓] Add information to deployment descriptor (web.xml)

This will add servlet information in web.xml file, you dont have write it of your own

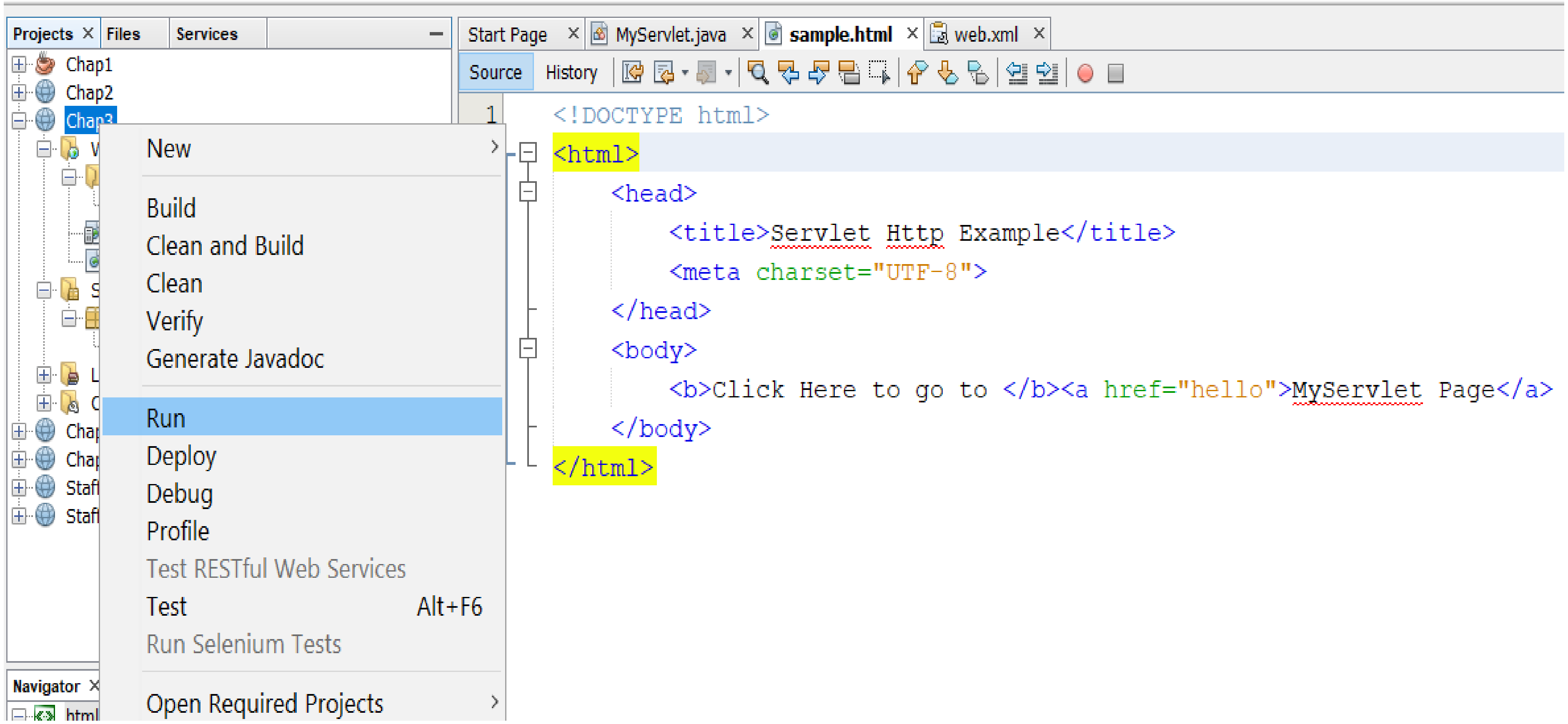
Class Name:	MyServlet
Servlet Name:	hello
URL Pattern(s):	/hello

Change servlet name and url pattern from here

Initialization Parameters:

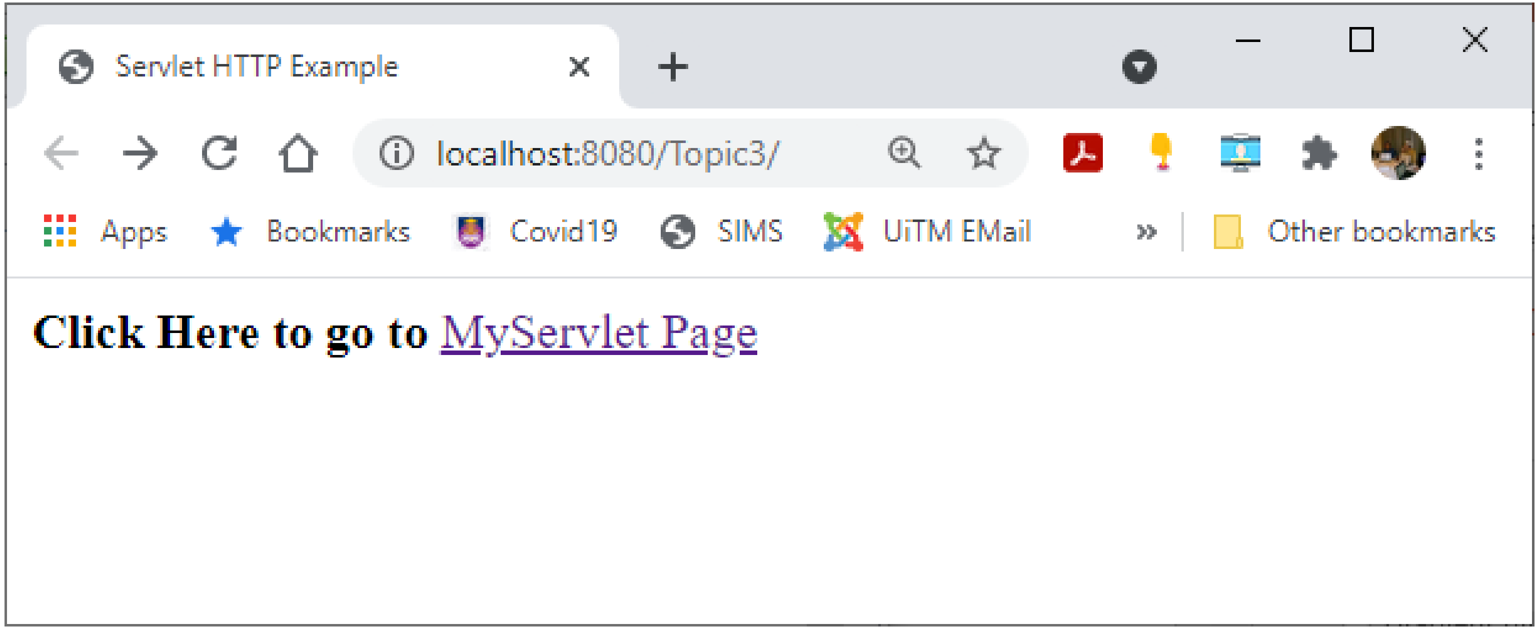
Name	Value

4. Run the project



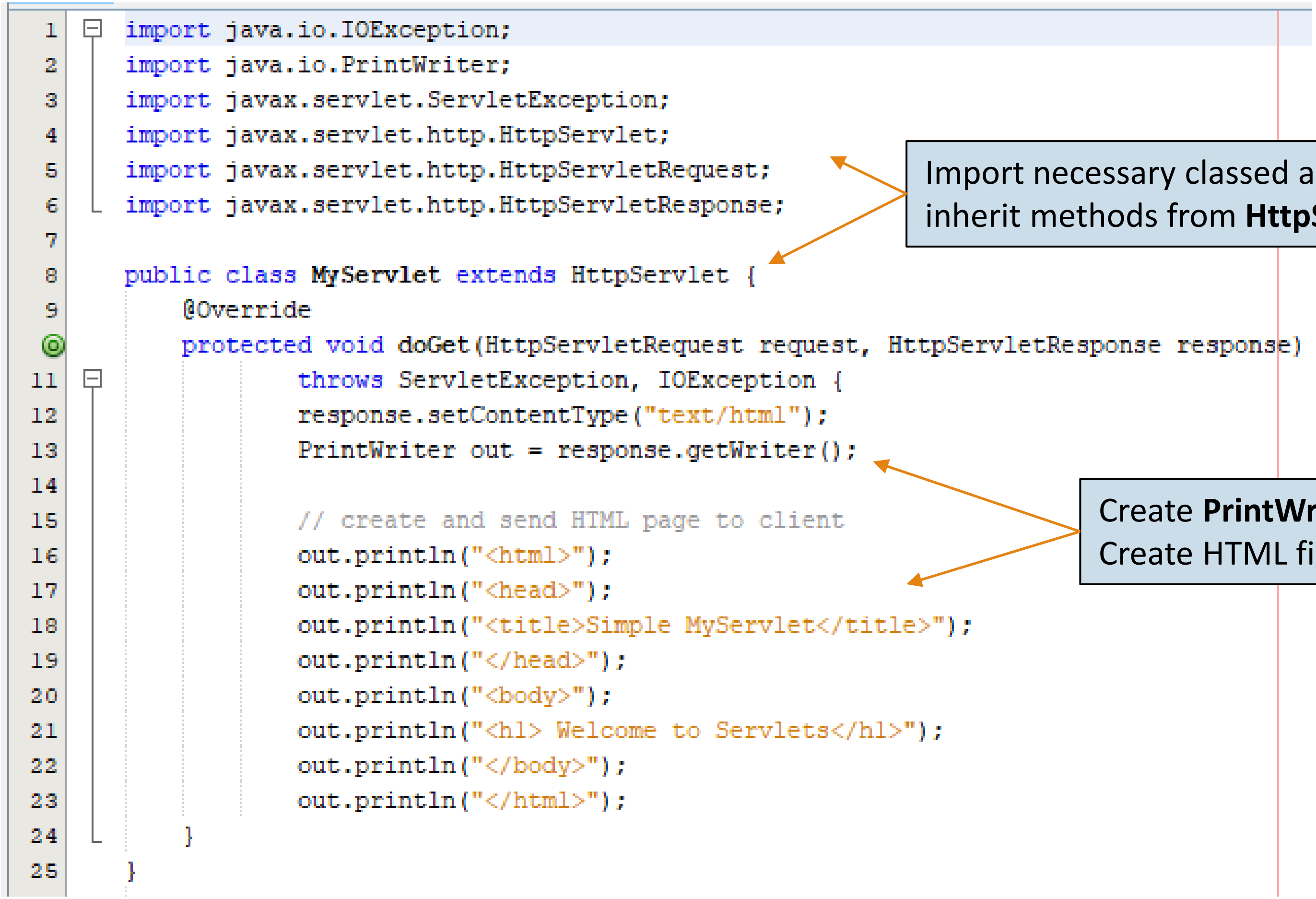
```
<!DOCTYPE html>
<html>
    <head>
        <title>Servlet Http Example</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <b>Click Here to go to </b><a href="hello">MyServlet Page</a>
    </body>
</html>
```

4. Access the servlet



Click Here to go to MyServlet Page

Examples-Servlet



```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 MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            // create and send HTML page to client
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Simple MyServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1> Welcome to Servlets</h1>");
            out.println("</body>");
            out.println("</html>");
    }
}
```

Import necessary classed and inherit methods from **HttpServlet**

Create **PrintWriter** object
Create HTML file and send to client

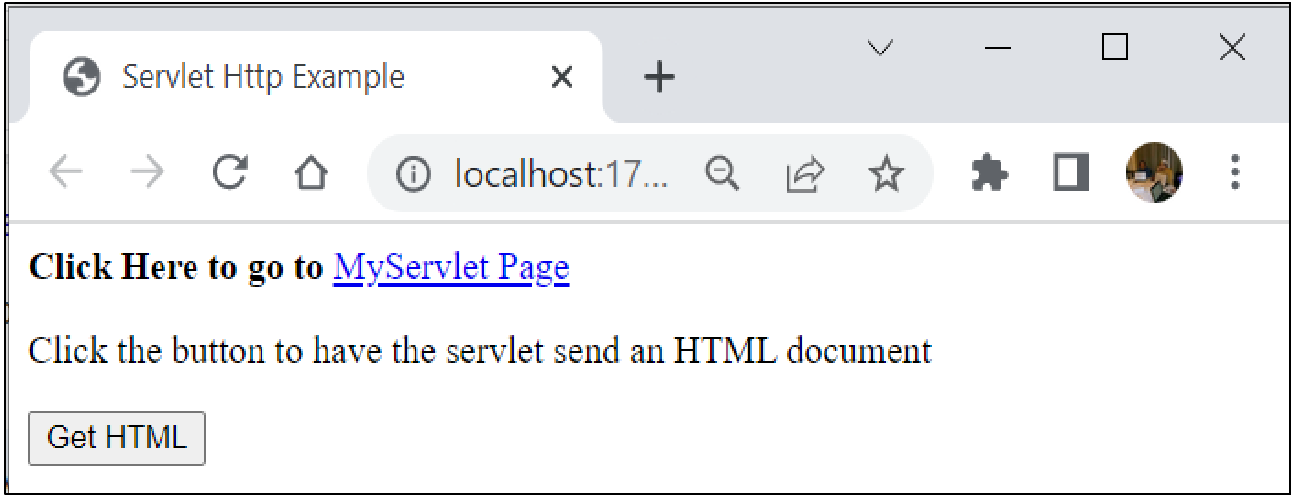
Handling HTTP GET Requests

Sample.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Servlet HTTP Example</title>
   </head>
   <body>
     <b>Click Here to go to</b> <a href="hello">MyServlet Page</a>
     <br><br>
     <form action="HTTPGetServlet"  method="get">
       <P>Click the button to have the servlet send an HTML document</P>
       <input type="submit" value="Get HTML">
      </form>
   </body>
</html>
```

ACTION specifies form handler,
METHOD specifies request type.

Creates submit button,
performs **ACTION** when clicked..

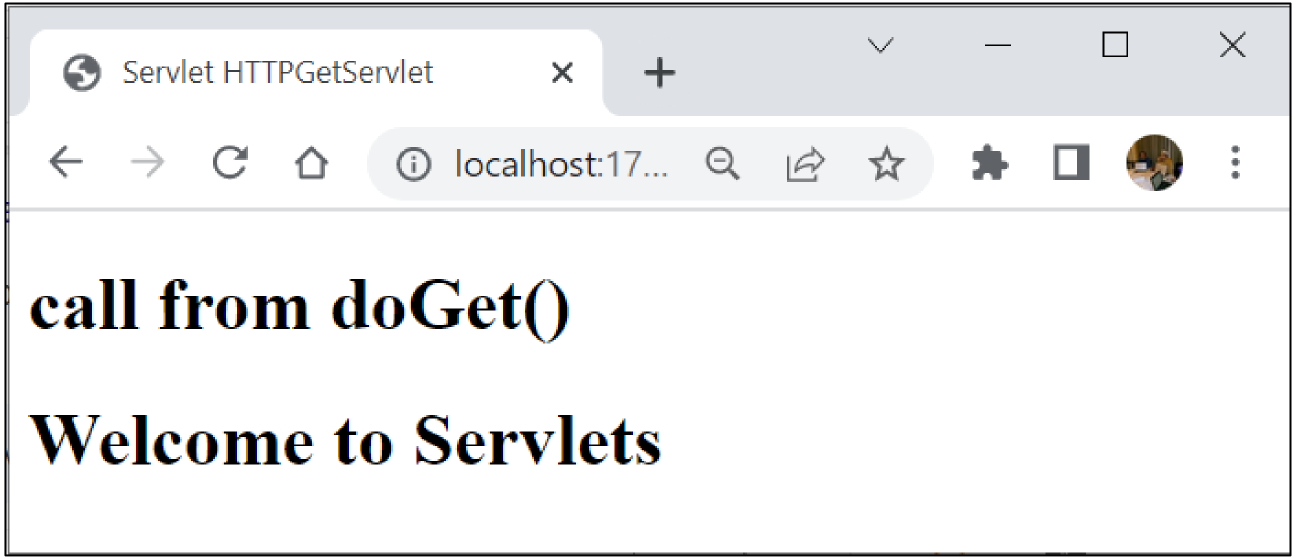

Servlet Http Example

localhost:17...

Click Here to go to **MyServlet Page**

Click the button to have the servlet send an HTML document

Get HTML


Servlet HTTPGetServlet

localhost:17...

call from doGet()

Welcome to Servlets