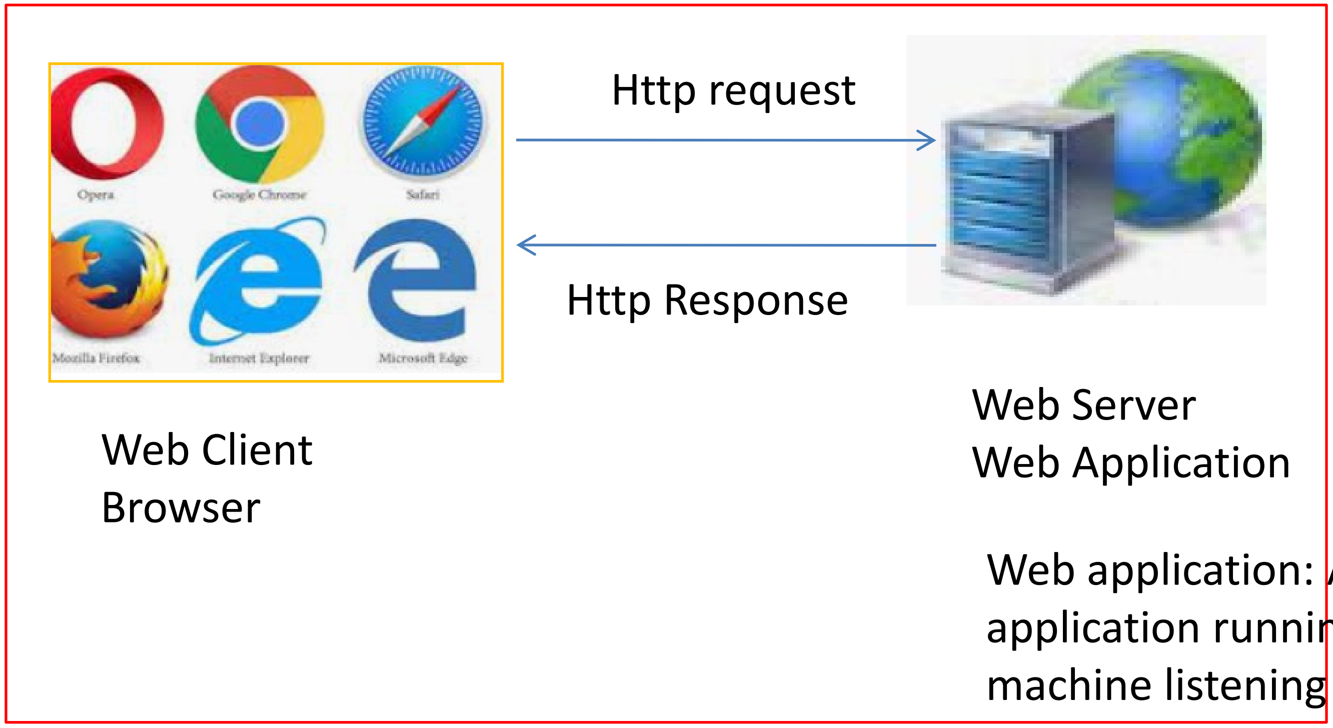
CS6308- Java Programming

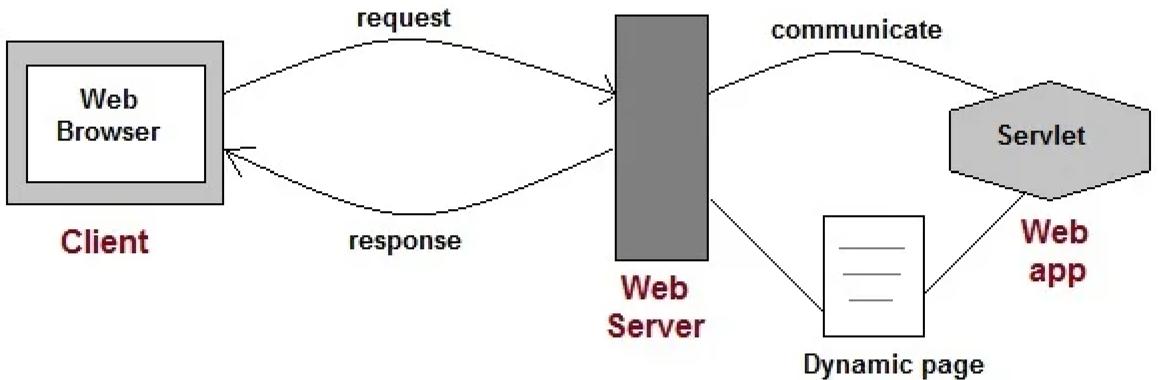
V P Jayachitra
Assistant Professor
Department of Computer Technology
MIT Campus
Anna University



Web application: A network application running on a machine listening to a port

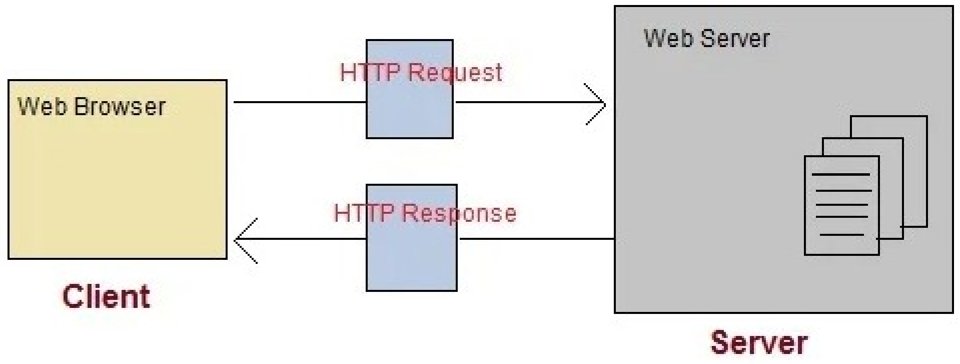
Web application

- A website is a collection of static files(webpages) such as HTML pages, images, graphics etc. A **Web application** is a web site with dynamic functionality on the server.
- **Google**, **Facebook**, **Twitter** are examples of web applications.
- **Servlet** Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.



HTTP

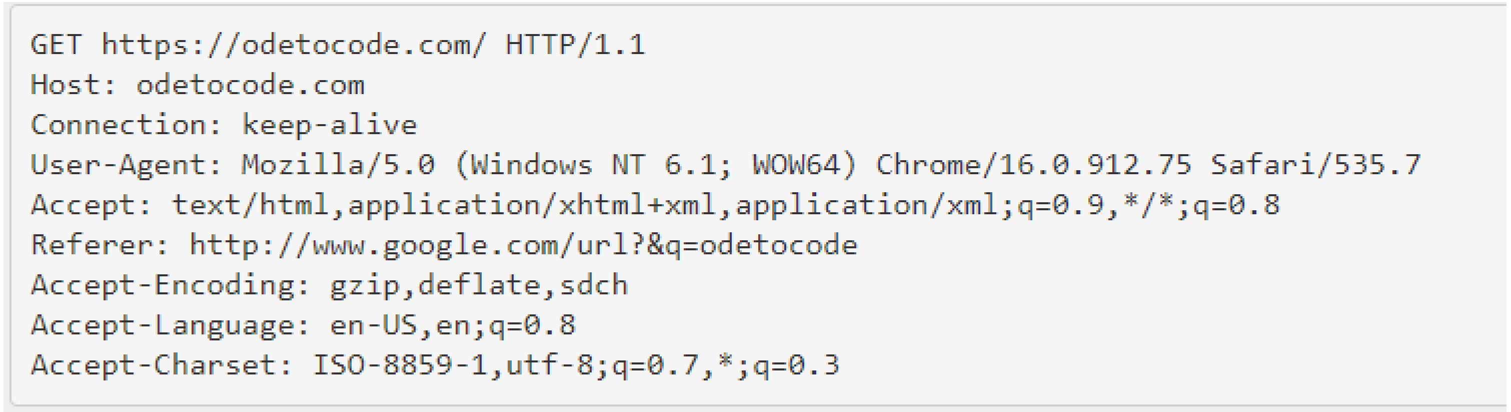
- HTTP is a protocol that clients and servers use on the web to communicate.
- It is similar to other internet protocols such as SMTP(Simple Mail Transfer Protocol) and FTP(File Transfer Protocol) but there is one fundamental difference.
- HTTP is a **stateless protocol** i.e HTTP supports only one request per connection. This means that with HTTP the clients connect to the server to send one request and then disconnects. This mechanism allows more users to connect to a given server over a period of time.
- The client sends an HTTP request and the server answers with an HTML page to the client, using HTTP.



HTTP Request
[method] [URL] [version]
[headers]
[body]

HTTP REQUEST METHODS

Method	**Description**
GET	Retrieve a resource
PUT	Store a resource
DELETE	Remove a resource
POST	Update a resource
HEAD	Retrieve just the headers for a resource



```
GET https://odetocode.com/ HTTP/1.1
Host: odetocode.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/16.0.912.75 Safari/535.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://www.google.com/url?&q=odetocode
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Difference between GET and POST requests

GET Request	POST Request
Data is sent in header to the server	Data is sent in the request body
Get request can send only limited amount of data	Large amount of data can be sent.
Get request is not secured because data is exposed in URL	Post request is secured because data is not exposed in URL.
Get request can be bookmarked and is more efficient.	Post request cannot be bookmarked.

Forms and GET Requests

```
<form action="/search" method="GET">
        <label for="term">Search:label>
        <input id="term" name="term" type="text" />
        <input type="submit" value="Sign up!"/>
form>
```

```
GET http://localhost:1060/search?term=love HTTP/1.1
Host: localhost:1060
```

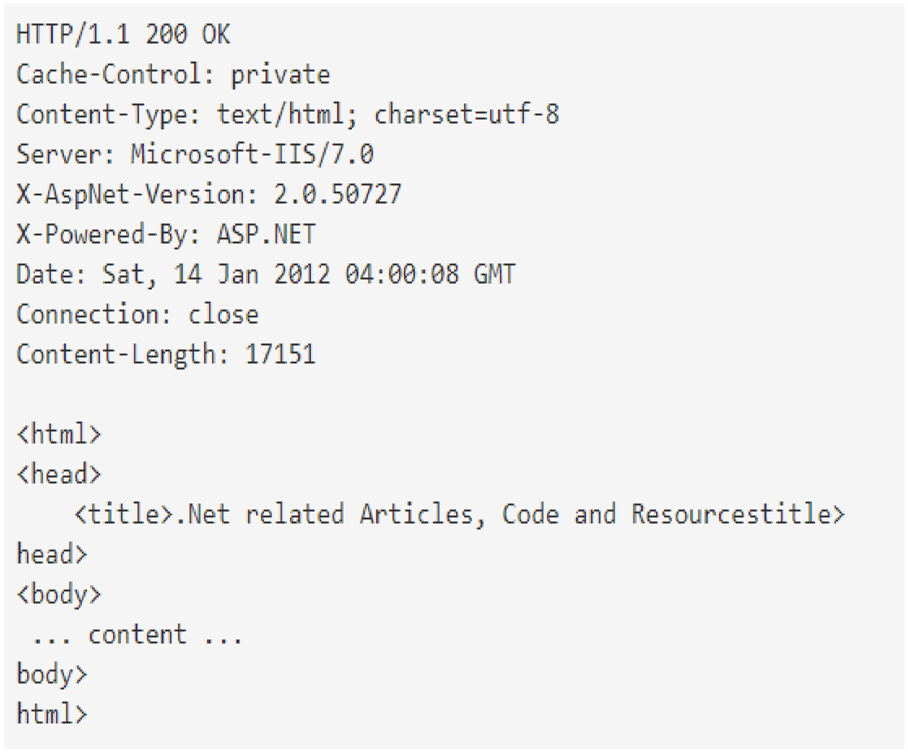
```
<form action="/account/create" method="POST">
    <label for="firstName">First namelabel>
    <input id="firstName" name="firstName" type="text" />

    <label for="lastName">Last namelabel>
    <input id="lastName" name="lastName" type="text" />

    <input type="submit" value="Sign up!"/>
form>
```

```
POST http://localhost:1060/account/create HTTP/1.1
Host: localhost:1060

firstName=Scott&lastName=Allen
```

POST parameters go into the body of the HTTP message.
GET parameters go into the query string.

The Response

An HTTP response has a similar structure to an HTTP request. The sections of a response are:
[version] [status] [reason]
[headers]
[body]

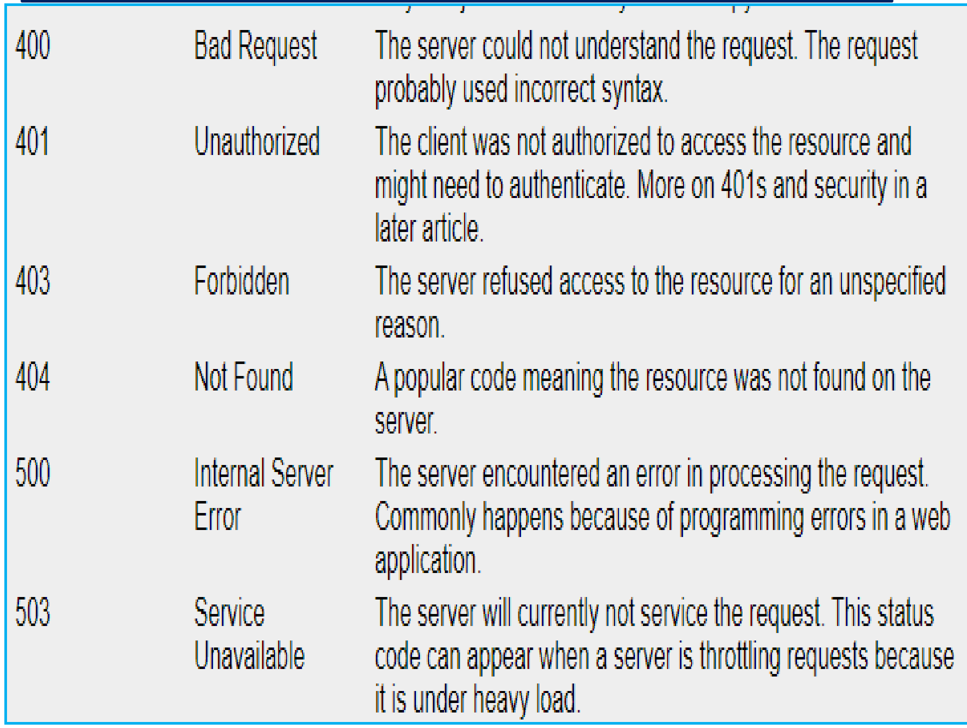


```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.0
X-AspNet-Version: 2.0.50727
X-Powered-By: ASP.NET
Date: Sat, 14 Jan 2012 04:00:08 GMT
Connection: close
Content-Length: 17151

<html>
<head>
    <title>.Net related Articles, Code and Resourcestitle>
head>
<body>
 ... content ...
body>
html>
```

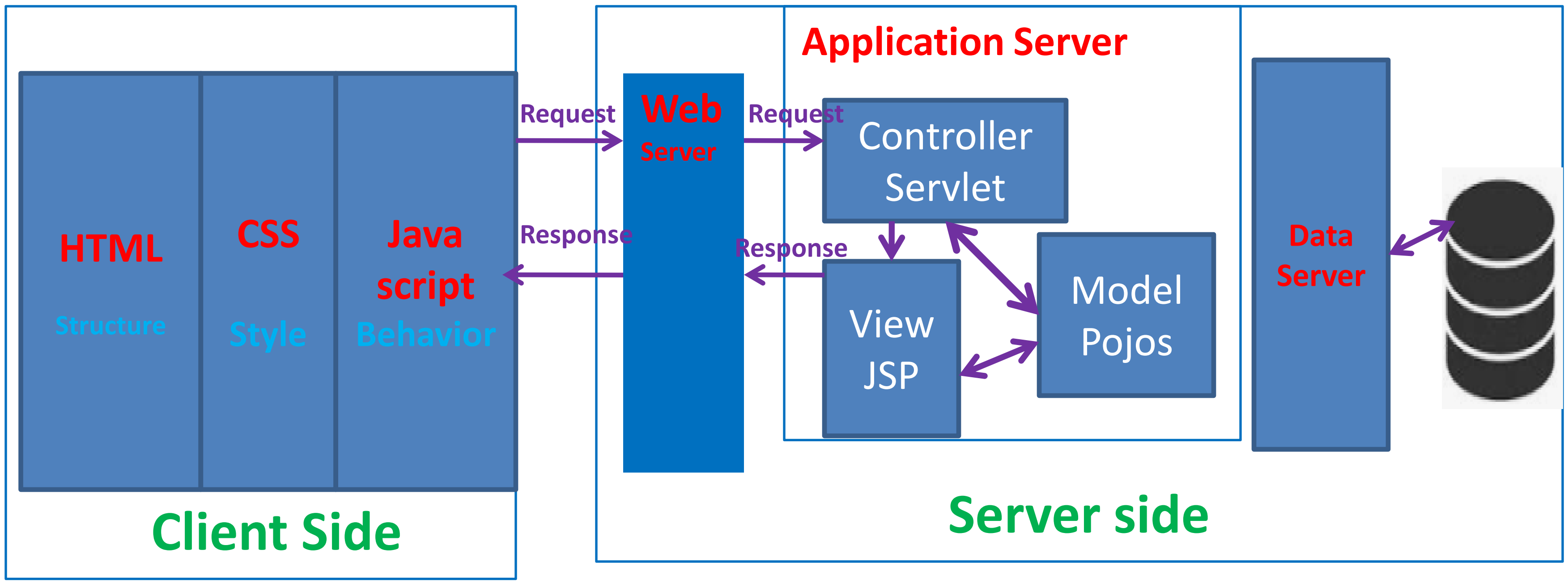
Response Status Codes

Code   Reason
200   OK

Description: A 200 code in the response means everything worked!



400	Bad Request	The server could not understand the request. The request probably used incorrect syntax.
401	Unauthorized	The client was not authorized to access the resource and might need to authenticate. More on 401s and security in a later article.
403	Forbidden	The server refused access to the resource for an unspecified reason.
404	Not Found	A popular code meaning the resource was not found on the server.
500	Internal Server Error	The server encountered an error in processing the request. Commonly happens because of programming errors in a web application.
503	Service Unavailable	The server will currently not service the request. This status code can appear when a server is throttling requests because it is under heavy load.



```
int  var=request.getParameter("textName");
```

What Is a Servlet?

- A **servlet** is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.
- Servlet is a java program that runs inside JVM on the web server.
- Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.
- It is used for developing dynamic web applications.
- For such applications, Java Servlet technology defines HTTP-specific servlet classes.

What Is a Servlet?

- The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets.
- All servlets must implement the Servlet interface, which defines life-cycle methods.
- When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API.
- The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

Features of Servlet

- **Portable:**
  - For example, you can create a servlet on Windows operating system that users GlassFish as web server and later run it on any other operating system like Unix, Linux with Apache tomcat web server, this feature makes servlet portable.
- **Efficient and scalable:**
  - The web server invokes servlet using a lightweight thread so multiple client requests can be fulling by servlet at the same time using the multithreading feature of Java.

Features of Servlet

- **Robust:**
  - the servlet is less prone to memory management issues and memory leaks (by inheriting the top features of Java (such as Garbage collection, Exception handling, Java Security Manager etc.))

Servlet API

- Every Servlet must implement the java.servlet.Servlet interface, you can do it by extending one of the following two classes: javax.servlet.GenericServlet or javax.servlet.http.HttpServlet.
- The first one is for protocol independent Servlet and the second one for http Servlet.

javax.servlet.GenericServlet
javax.servlet.http.HttpServlet

Generic Servlet

javax.servlet.GenericServlet class.

- GenericServlet class has an abstract service() method. Which means the subclass of GenericServlet should always override the service() method.
  Signature of service() method:

```
public abstract void service(ServletRequest request, ServletResponse response)
      throws ServletException, java.io.IOException
```

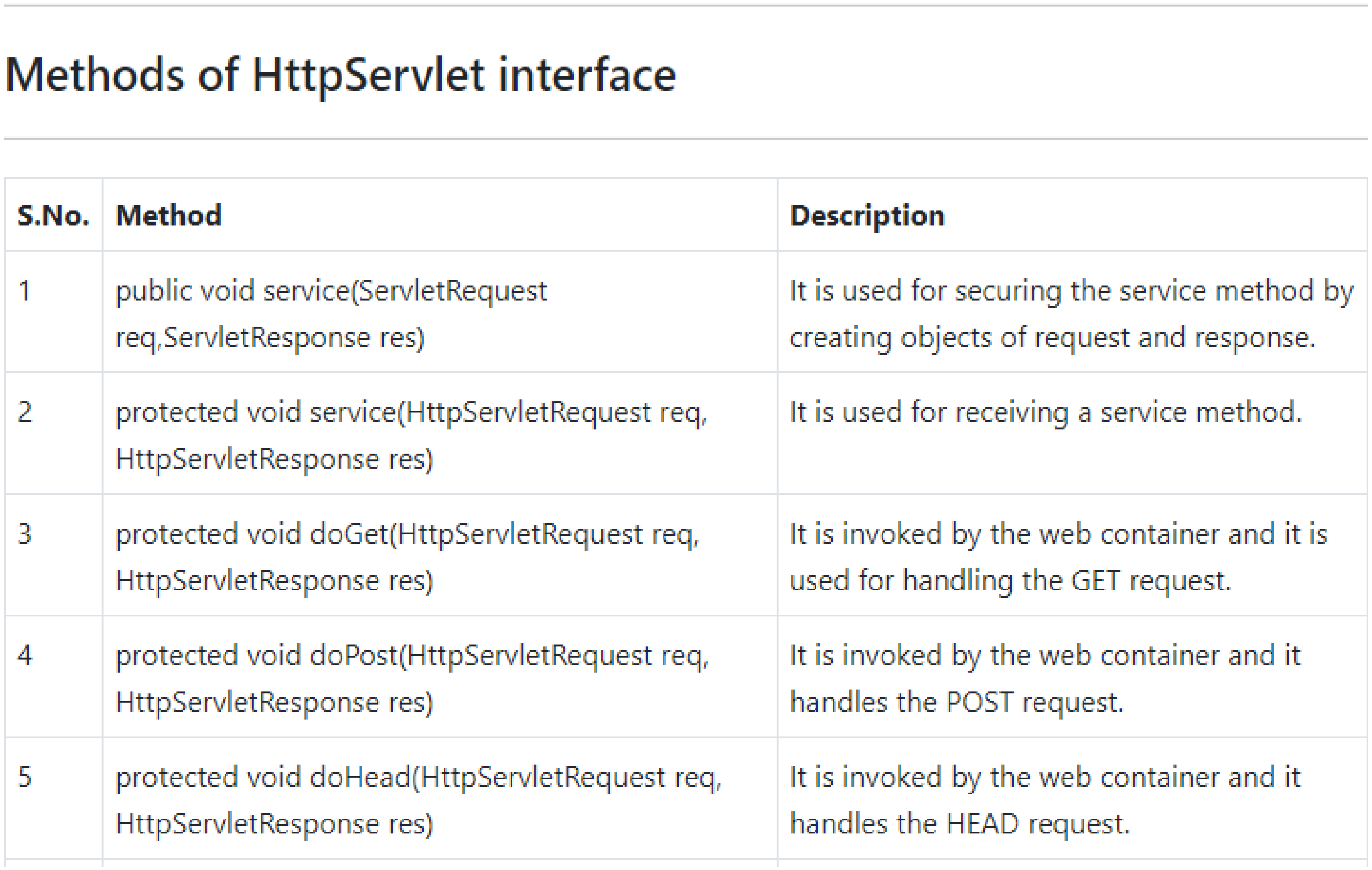
- The service() method accepts two arguments ServletRequest object and ServletResponse object.
- The request object tells the servlet about the request made by client while the response object is used to return a response back to the client.

Methods of Servlet interface

S.No.	Method	Description
1.	public void init(ServletConfigconfig)	It is used for initializing the servlet. It is invoked only once by the web container in a servlet life cycle.
2.	public void service(ServletRequestreq, ServletResponse res)	It is used for providing a response to all the incoming request. It is invoked every time by the web container for each request.
3.	public void destroy()	It is used for destroying the servlet. It is invoked only once in a life cycle of a servlet.

HTTP Servlet

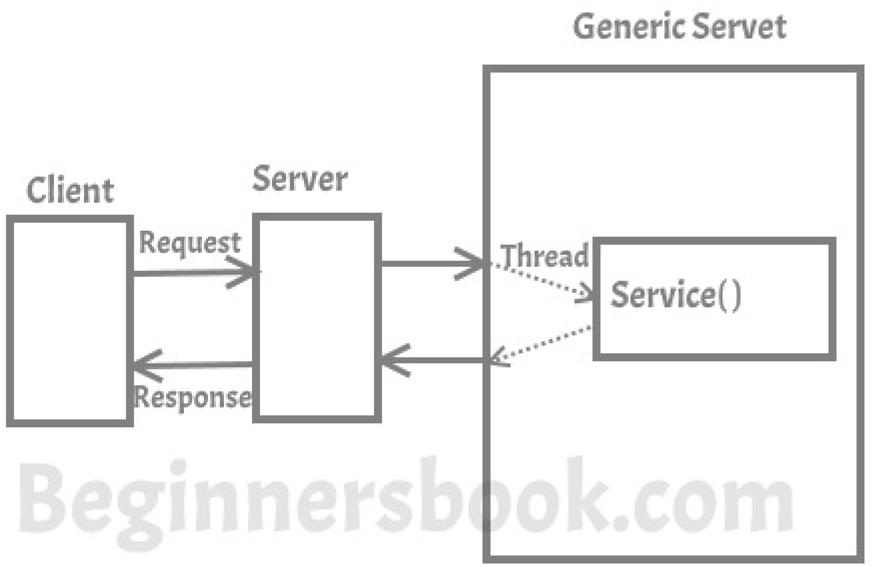
- **doGet()** – This method is called by servlet service method to handle the HTTP GET request from client. **The Get method is used for getting information from the server**
- **doPost()** – **Used for posting information to the Server**
- **doPut()** – This method is **similar to doPost** method but unlike doPost method where we send information to the server, this method sends file to the server, this is **similar to the FTP operation from client to server**
- **doDelete()** – allows **a client to delete a document, webpage or information from the server**
- **init() and destroy()** – Used for managing resources that are held for the life of the servlet
- **getServletInfo()** – Returns information about the servlet, such as author, version, and copyright.

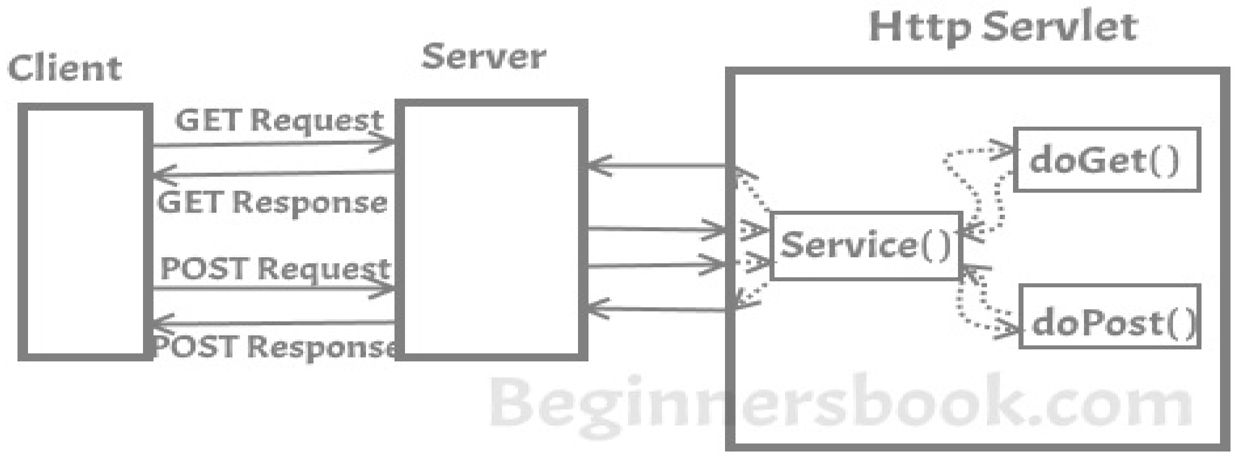


Methods of HttpServlet interface

S.No.	Method	Description
1	public void service(ServletRequest req,ServletResponse res)	It is used for securing the service method by creating objects of request and response.
2	protected void service(HttpServletRequest req, HttpServletResponse res)	It is used for receiving a service method.
3	protected void doGet(HttpServletRequest req, HttpServletResponse res)	It is invoked by the web container and it is used for handling the GET request.
4	protected void doPost(HttpServletRequest req, HttpServletResponse res)	It is invoked by the web container and it handles the POST request.
5	protected void doHead(HttpServletRequest req, HttpServletResponse res)	It is invoked by the web container and it handles the HEAD request.

service() method of HttpServlet class listens to the Http methods (GET, POST etc) from request stream and invokes doGet() or doPost() methods based on Http Method


Generic Servet
Client
Server
Request
Response
Thread
Service()
Beginnersbook.com


Http Servlet
Client
Server
GET Request
GET Response
POST Request
POST Response
Service()
doGet()
doPost()
Beginnersbook.com

Generic Servlet

- **Pros of using Generic Servlet:**
  1. Generic Servlet is easier to write
  2. Has simple lifecycle methods
  3. To write Generic Servlet you just need to extend javax.servlet.GenericServlet and override the service() method
- **Cons of using Generic Servlet:**
  Working with Generic Servlet is not that easy because methods such as doGet(), doPost(), doHead() etc does not exist in Generic Servlet that we can use in Http Servlet.
  - In Http Servlet we need to override particular convenience method for particular request,
  - for example to get information then override doGet(),
  - Similarly, to send information to server override doPost().
  - However in Generic Servlet only service() method need to be override for each type of request which is cumbersome.

INDEX.HTML

```html
<html><body>
<form method="post" action="check">
   Name :<input type="text" name="user" >
   <input type="submit">
</form></body></html>
```

web.xml

```xml
<servlet>
   <servlet-name>check</servlet-name>
   <servlet-class>Servlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>check</servlet-name>
   <url-pattern>/check</url-pattern>
</servlet-mapping>
```

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

public class Servlet extends HttpServlet {

 protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     try {
        String user = request.getParameter("user");
        out.println("<h2> Welcome "+user+"</h2>");
     } finally {
        out.close();
     }
   }
}
```

Program 1

ExampleHttpServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
// Creating Http Servlet by Extending HttpServlet class
public class ExampleHttpServlet extends HttpServlet
{
   private String mymsg;
   public void init() throws ServletException
   {
     msg = "Http Servlet Demo";
   }
   public void doGet(HttpServletRequest request,
       HttpServletResponse response) throws ServletException,
       IOException
   {
       // Setting up the content type of web page
       response.setContentType("text/html");
       // Writing the message on the web page
       PrintWriter out = response.getWriter();
       out.println("<h1>" + msg + "</h1>");
       out.println("<p>" + "Hello Students!" + "</p>");
   }
```

index.html

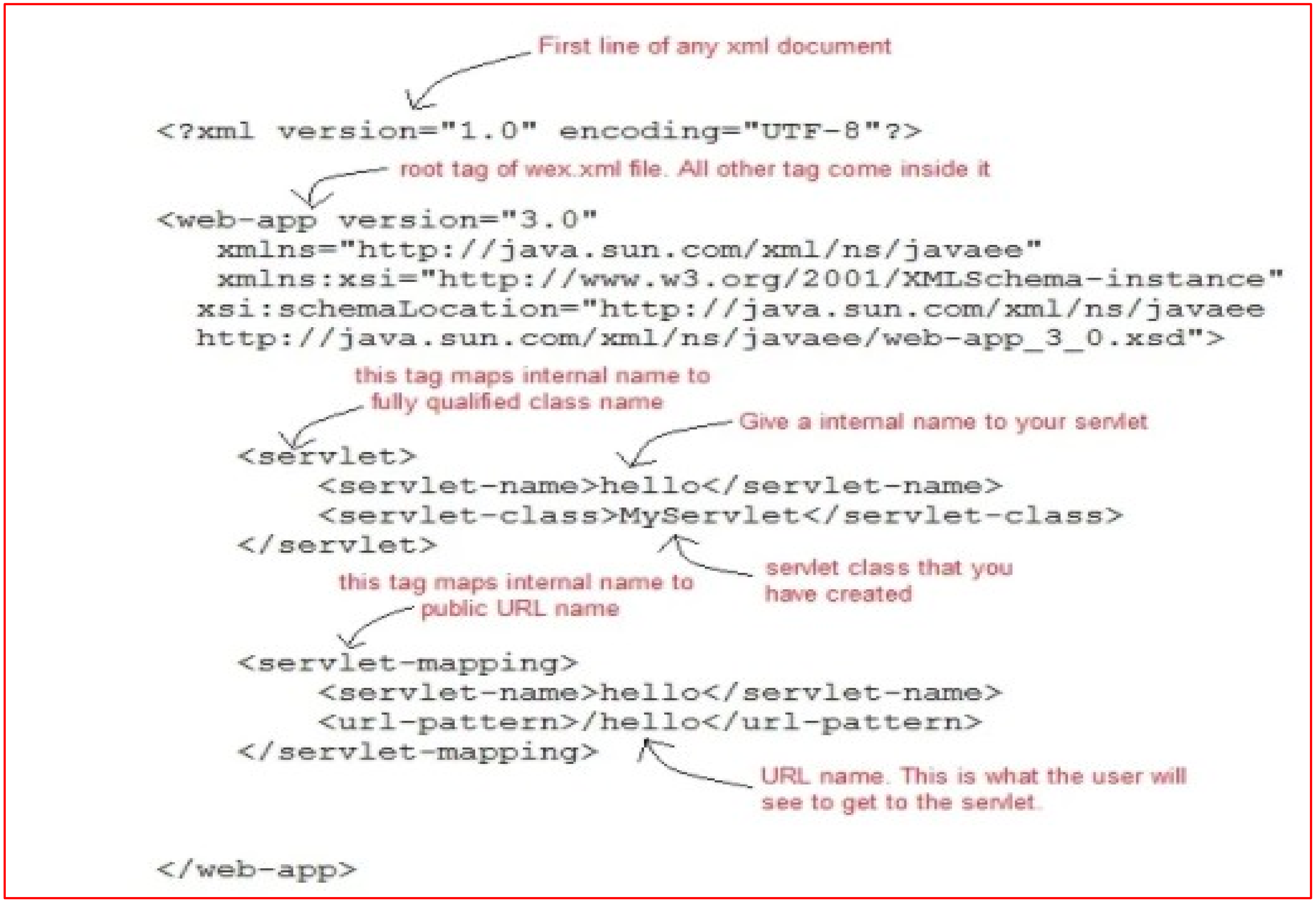
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Http Servlet Demo</title>
</head>
<body>
<a href="Demo">Click to call Servlet</a>
</body>
</html>
```

web.xml web.xml file is a deployment descripter.

```
<web-app>
<servlet>
<servlet-name>HttpServletDemo</servlet-name>
<servlet-class>ExampleHttpServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HttpServletDemo</servlet-name>
<url-pattern>/Demo</url-pattern>
</servlet-mapping>
</web-app>
```

Program 2

•A set of servlet elements that identify all the servlet instances of the application.
•A set of servlet-mapping elements that map the servlets to URL patterns. More than one URL pattern can be defined for a particular servlet.



```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

    <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>


</web-app>
```

First line of any xml document

root tag of wex.xml file. All other tag come inside it

this tag maps internal name to fully qualified class name

Give a internal name to your servlet

servlet class that you have created

this tag maps internal name to public URL name

URL name. This is what the user will see to get to the servlet.

Program 3

```
//marks.java
 import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;
public class marks extends HttpServlet{
public void service(ServletRequest req, ServletResponse res) throws IOException,
ServletException
        {       int Sub1 = Integer.parseInt(req.getParameter(“sub1"));
                int Sub2 = Integer.parseInt(req.getParameter("sub2"));
                int Sub3 = Integer.parseInt(req.getParameter("sub3"));
                int Sub4= Integer.parseInt(req.getParameter("sub4"));
                int Sub5 = Integer.parseInt(req.getParameter("sub5"));
                int Sub6 = Integer.parseInt(req.getParameter("sub6"));
                int total = Sub1+Sub2+Sub3+Sub4+Sub5+Sub6;
                float avgerage = total / 6;
                PrintWriter out = res.getWriter();
                out.println(“Subject1 : " + Sub1 );     out.println((“Subject2 : " + Sub2 );
                out.println((“Subject3 : " + Sub3);     out.println((“Lab1 : " + Sub4);
                 out.println(“Lab2: " + Sub5);      out.println(“Project : " + Sub6);
                out.println("Total Marks : "+ total); out.println("Average: "+average);
        } }
```

web.xml

Program 3

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
        xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <servlet>
                <servlet-name>abcd</servlet-name>
                <servlet-class>marks</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>abcd</servlet-name>
  <url-pattern>/ Average </url-pattern>
  </servlet-mapping>
</web-app>
```

Demo.html

Program 3

```
<!DOCTYPE html>
<html>
<body>
<form action="Average" align="center">
<h3 align="center">----------------------------------------------------</h3>
Enter marks of the following subjects<br><br><br>
Subject1 : <input type="text" name="sub1"><br><br>
Subject2 : <input type="text" name="sub2"><br><br>
Subject3 : <input type="text" name="sub3"><br><br>
Lab1 : <input type="text" name="sub4"><br><br>
Lab2: <input type="text" name="sub5"><br><br>
Project: <input type="text" name="sub6"><br><br>
<input type="submit">
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<form action="display" method="get">
<hr>
User name: <input type="text"
name="val1"> <br><br>
Password: &nbsp;&nbsp;<input
type="password" name="val2" ><br><br>
<input type="submit" value="login">
</body>
</html>
```

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class demo4 extends HttpServlet{
  public void doGet(HttpServletRequest
req,HttpServletResponse res)
  throws ServletException,IOException
  {
    res.setContentType("text/html");
    PrintWriter pwriter=res.getWriter();
    String uname=req.getParameter("val1");
    String pw=req.getParameter("val2");
    pwriter.println("User Details Page:");
    pwriter.println("Hello "+uname);
    pwriter.println("Your Password is **"+pw+"**");
    pwriter.close();   } }
```

Web.xml

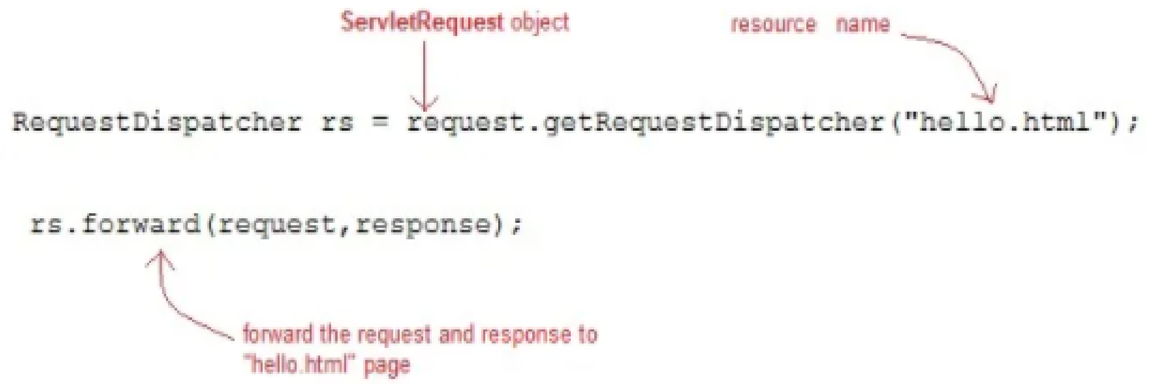
```
 <servlet>
  <servlet-name>abc3</servlet-name>
  <servlet-class>demo4</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>abc3</servlet-name>
  <url-pattern>/display</url-pattern>
  </servlet-mapping>
```

Program 4

How to get an Object of RequestDispatcher

`getRequestDispatcher()` method of **ServletRequest** returns the object of **RequestDispatcher**.

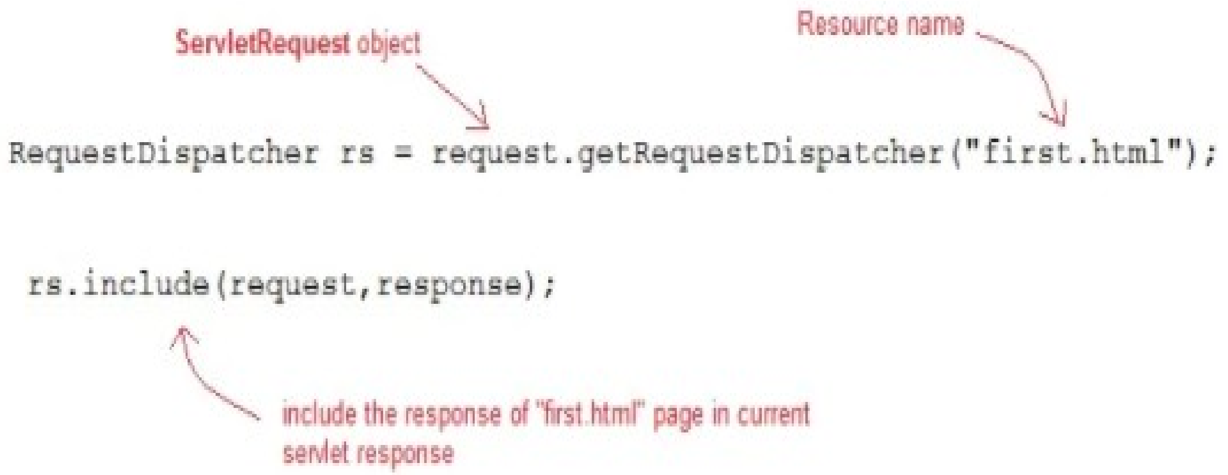
```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");
rs.forward(request,response);
```



```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");

rs.forward(request,response);
```

OR

```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");
rs.include(request,response);
```



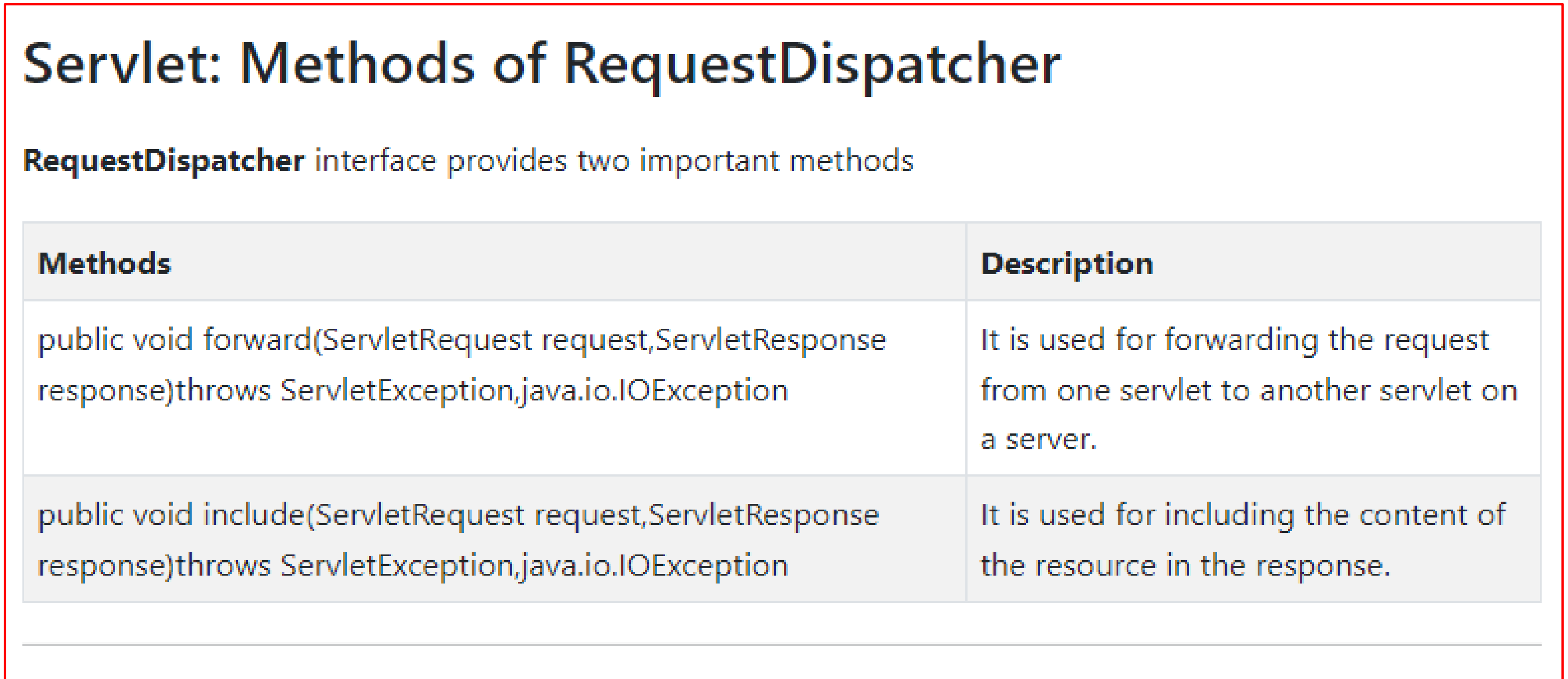
```
RequestDispatcher rs = request.getRequestDispatcher("first.html");

rs.include(request,response);
```

ServletRequest object
Resource name
include the response of "first.html" page in current servlet response

RequestDispatcher is used to **forward** or **include** response of a resource in a Servlet. Here we are using **index.html** to get username and password from the user, **Validate** Servlet will validate the password entered by the user, if the user has entered "studytonight" as password, then he will be forwarded to **Welcome** Servlet else the user will stay on the index.html page and an error message will be displayed.

Files to be created :

•**index.html** will have form fields to get user information.
•**Validate.java** will validate the data entered by the user.
•**Welcome.java** will be the welcome page.
•**web.xml** , the deployment descriptor.



Servlet: Methods of RequestDispatcher

RequestDispatcher interface provides two important methods

Methods	Description
public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException	It is used for forwarding the request from one servlet to another servlet on a server.
public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException	It is used for including the content of the resource in the response.

```html
<form method="post" action="Validate">
Name:<input type="text" name="user"
/><br/>
Password:<input type="password"
name="pass" ><br/>
<input type="submit" value="submit">
</form>
```

```xml
<web-app>
  <servlet>
    <servlet-name>Validate</servlet-name>
    <servlet-class>Validate</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Welcome</servlet-name>
    <servlet-class>Welcome</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Validate</servlet-name>
    <url-pattern>/Validate</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Welcome</servlet-name>
    <url-pattern>/Welcome</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Validate.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Validate extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
      String name = request.getParameter("user");
      String password = request.getParameter("pass");
      if(password.equals("studytonight"))
      {
        RequestDispatcher rd = request.getRequestDispatcher("Welcome");
        rd.forward(request, response);
      }
      else
      {
        out.println("<font color='red'><b>You have entered incorrect password</b></font>");
        RequestDispatcher rd = request.getRequestDispatcher("index.html");
        rd.include(request, response);
      }
    }
    finally {
      out.close();     }  }}
```

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

public class Welcome extends HttpServlet {

   protected void doPost(HttpServletRequest request, HttpServletResponse
response)
         throws ServletException, IOException {

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
         out.println("<h2>Welcome user</h2>");
      }
      finally {
         out.close();
      }
   }
}
```

Welcome.java