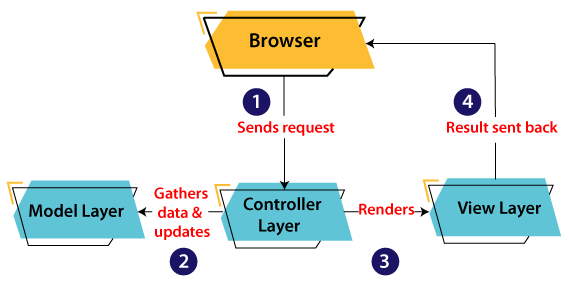
MVC Architecture in Java

* **Model:** It represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed.
* **View:** It represents the presentation layer of application. It is used to visualize the data that the model contains.
* **Controller:** It works on both the model and view. It is used to manage the flow of application, i.e. data flow in the model object and to update the view whenever data is changed.

In Java Programming, the Model contains the simple [Java classes](https://www.javatpoint.com/object-and-class-in-java)

, the View used to display the data and the Controller contains the [servlets](https://www.javatpoint.com/servlet-tutorial)

. Due to this separation the user requests are processed as follows:



1. A client (browser) sends a request to the controller on the server side, for a page.
2. The controller then calls the model. It gathers the requested data.
3. Then the controller transfers the data retrieved to the view layer.
4. Now the result is sent back to the browser (client) by the view.

# How to build a Web Application Using Java

 A web application is computer software that utilizes the web browser and technologies to perform tasks over the internet. A web application is deployed on a web server.

[Java](https://www.javatpoint.com/java-tutorial)

provides some technologies like [Servlet](https://www.javatpoint.com/servlet-tutorial)

and [JSP](https://www.javatpoint.com/jsp-tutorial)

that allow us to develop and deploy a web application on a server easily. It also provides some frameworks such as Spring, Spring Boot that simplify the work and provide an efficient way to develop a web application. They reduce the effort of the developer.

We can create a website using static [HTML](https://www.javatpoint.com/html-tutorial)

pages and style them using [CSS](https://www.javatpoint.com/css-tutorial)

, but we need server-side technology when we want to create a dynamic website.

In this section, we will see how to create a website using Java Servlets and HTML. Further, we will see how these technologies are useful for developing a web application.

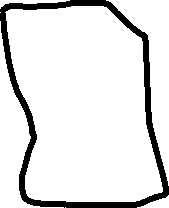
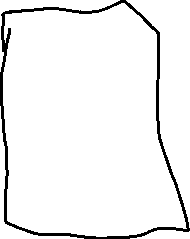
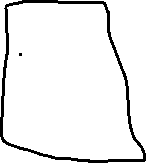
Let's understand the components of a web application:

### **What is a Web Application**

A web application is computer software that can be accessed using any web browser. Usually, the frontend of a web application is created using the scripting languages such as HTML, CSS, and JavaScript, supported by almost all web browsers. In contrast, the backend is created by any of the programming languages such as Java, Python, Php, etc., and databases. Unlike the mobile application, there is no specific tool for developing web applications; we can use any of the supported IDE for developing the web application.



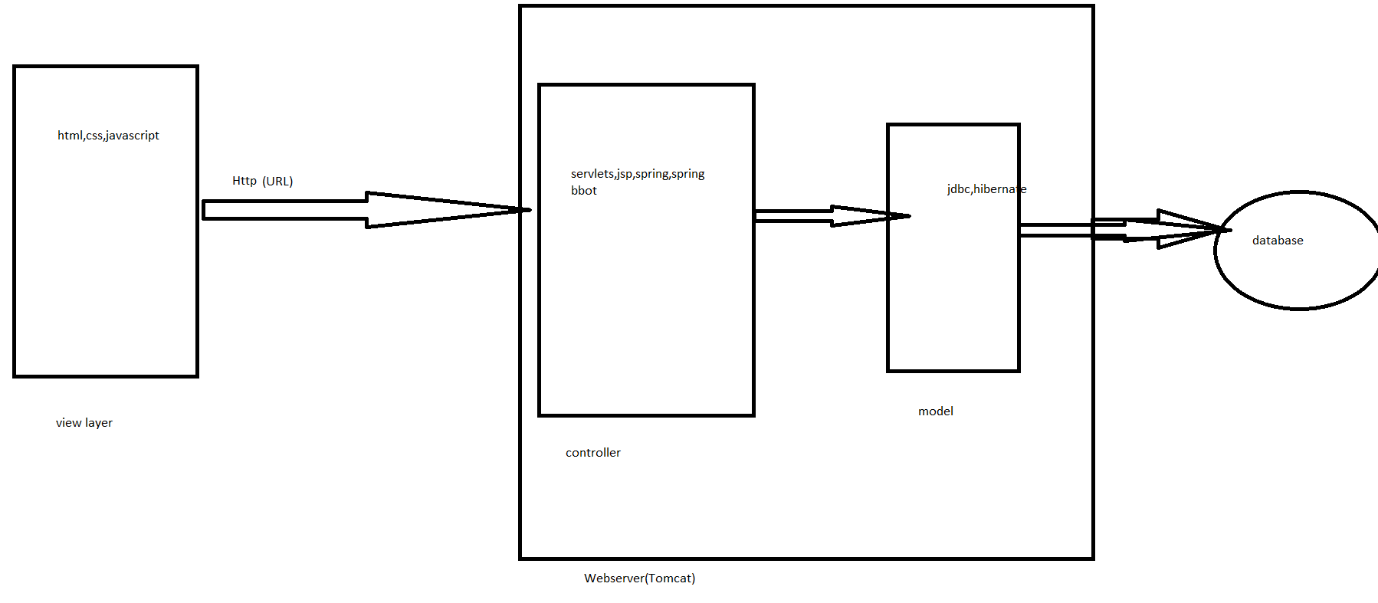
html, servlets,



javascript,

css





### **Web Server and Client**

The web server is a process that handles the client's request and responds. It processes the request made by the client by using the related protocols. The main function of the webserver is to store the request and respond to them with web pages. It is a medium between client and server. For example, Apache is a leading webserver.

A client is a software that allows users to request and assist them in communicating with the server. The web browsers are the clients in a web application; some leading clients are Google Chrome, Firefox, Safari, Internet Explorer, etc.

### **HTML and HTTP**

The HTML stands for HyperText Markup Language; it is a common language for Web Server and Web Client communication. Since both the web server and web client are two different software components of the web, we need a language that communicates between them.

The HTTP stands for HyperText Transfer Protocol; it is a communication protocol between the client and the server. It runs on top of the TCP/IP protocol.

Some of the integral components of an HTTP Request are as following:

**HTTP Method:** The HTTP method defines an action to be performed; usually, they are GET, POST, PUT, etc.

**URL:** URL is a web address that is defined while developing a web application. It is used to access a webpage.

**Form Parameters:** The form parameter is just like an argument in a Java method. It is passed to provide the details such as user, password details on a login page.

### **What is URL**

URL stands for Universal Resource Locator used to locate the server and resource. It is an address of a web page. Every web page on a project must have a unique name.

A URL looks like as follows:

1. http://localhost:8080/SimpleWebApplication/

Where,

**http or https:** It is the starting point of the URL that specifies the protocol to be used for communication.

**Localhost:** The localhost is the address of the server. When we run our application locally, it is called localhost; if we deployed our project over the web, then it is accessed by using the domain name like "javatpoint.com". The domain name maps the server to IP addresses.

**8080:** This is the port number for the local server; it is optional and may differ in different machines. If we do not manually type the port number in the URL, then by default, the request goes to the default port of the protocol. Usually, the port no between 0 to 1023 are reserved for some well-known services such as HTTP, HTTPS, FTP, etc.

We have discussed all the major components of a web application. Let's move towards our main motive How to build a web application in Java.

First, understand servlet:

### **What is Servlet**

A Servlet is a Java program that runs within a web server; it receives the requests and responds to them using related protocols (Usually HTTP). The Servlets are capable enough to respond to any type of request; they are commonly used to make the application functional.

We can create a static website using only HTML and CSS, but when it comes to dynamic, we need a server-side programming language. For these applications, Java provides Servlet technology, which contains HTTP-specific servlet classes.

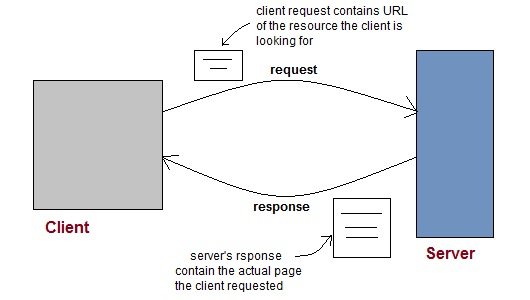
The **javax.servlet** and **javax.servlet.http** packages contain interfaces and classes for creating servlets. All servlets should implement the Servlet interface, which defines life-cycle methods. To implement a generic service, we can use the GenericServlet class by extending it. It provides **doGet** and **doPost** methods to handle HTTP-specific services.

### **Why are the Servlets Useful?**

Web servers are capable enough to serve static HTML requests, but they don't know how to deal with dynamic requests and databases. So, we need a language for dynamic content; these languages are PHP, Python, Java, Ruby on Rails, etc. In Java, there are two technologies Servlet and JSPs, that deals with dynamic content and database. Java also provides frameworks such as Spring, Spring Boot, Hibernate, and Struts to use the servlet and JSP easily.

The Servlets and JSPs are server-side technologies that extend the functionality of a web server. They support dynamic response and data persistence. We can easily create a web application using these technologies.

Web consists of billions of clients and server connected through wires and wireless networks. The web clients make requests to web server.

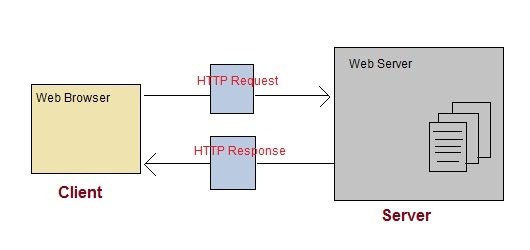


## **Web Application**

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.

### **HTTP (Hypertext Transfer Protocol)**

* 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.

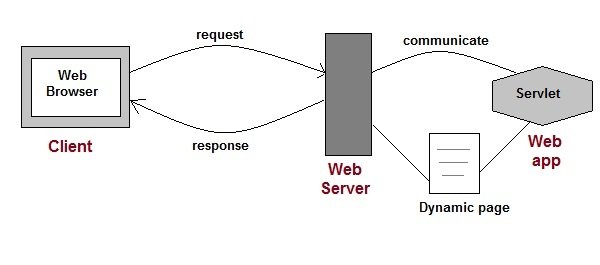


### **HTTP Methods**

HTTP request can be made using a variety of methods, but the ones you will use most often are **Get** and **Post**. The method name tells the server the kind of request that is being made, and how the rest of the message will be formated.

**HTTP Methods and Descriptions :**

| **Method Name** | **Description** |
| --- | --- |
| OPTIONS | Request for communication options that are available on the request/response chain. |
| GET | Request to retrieve information from server using a given URI. |
| HEAD | Identical to GET except that it does not return a message-body, only the headers and status line. |
| POST | Request for server to accept the entity enclosed in the body of HTTP method. |
| DELETE | Request for the Server to delete the resource. |
| CONNECT | Reserved for use with a proxy that can switch to being a tunnel. |
| PUT | This is same as POST, but POST is used to create, PUT can be used to create as well as update. It replaces all current representations of the target resource with the uploaded content. |



Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

* **javax.servlet**
* **javax.servlet.http**

### **Some Important Classes and Interfaces of javax.servlet**

|  |  |
| --- | --- |
| **INTERFACES** | **CLASSES** |
| Servlet | ServletInputStream |
| ServletContext | ServletOutputStream |
| [ServletConfig](https://www.studytonight.com/servlet/servlet-config.php) | ServletRequestWrapper |
| [ServletRequest](https://www.studytonight.com/servlet/servlet-request.php) | ServletResponseWrapper |
| [ServletResponse](https://www.studytonight.com/servlet/servlet-response.php) | ServletRequestEvent |
| ServletContextListener | ServletContextEvent |
| [RequestDispatcher](https://www.studytonight.com/servlet/request-dispatcher.php) | ServletRequestAttributeEvent |
| SingleThreadModel | ServletContextAttributeEvent |
| Filter | ServletException |
| FilterConfig | UnavailableException |
| FilterChain | GenericServlet |
| ServletRequestListener |  |

### **Some Important Classes and Interface of javax.servlet.http**

|  |  |
| --- | --- |
| **CLASSES and INTERFACES** | |
| HttpServlet | HttpServletRequest |
| HttpServletResponse | HttpSessionAttributeListener |
| HttpSession | HttpSessionListener |
| Cookie | HttpSessionEvent |

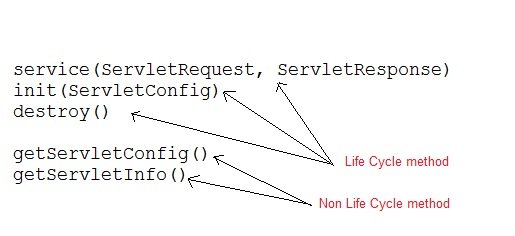
## **Servlet Interface**

In Java, An interface is used for the development of servlet. This interface is known as the servlet interface. This interface is implemented by all the interfaces. The servlet interface is used for the declaration of init(), service(), and destroy() method. These methods are called by the server during the life cycle of a servlet. The getServletConfig() method is called by the servlet to initialize the parameters. And the getServletInfo() method is used for providing important information.

Servlet Interface provides only five methods. Out of these five methods, three methods are of **Servlet life cycle** methods and rest two are **non-life cycle** methods.

### **Declaration :**

**public interface Servlet**



## **Methods of Servlet interface**

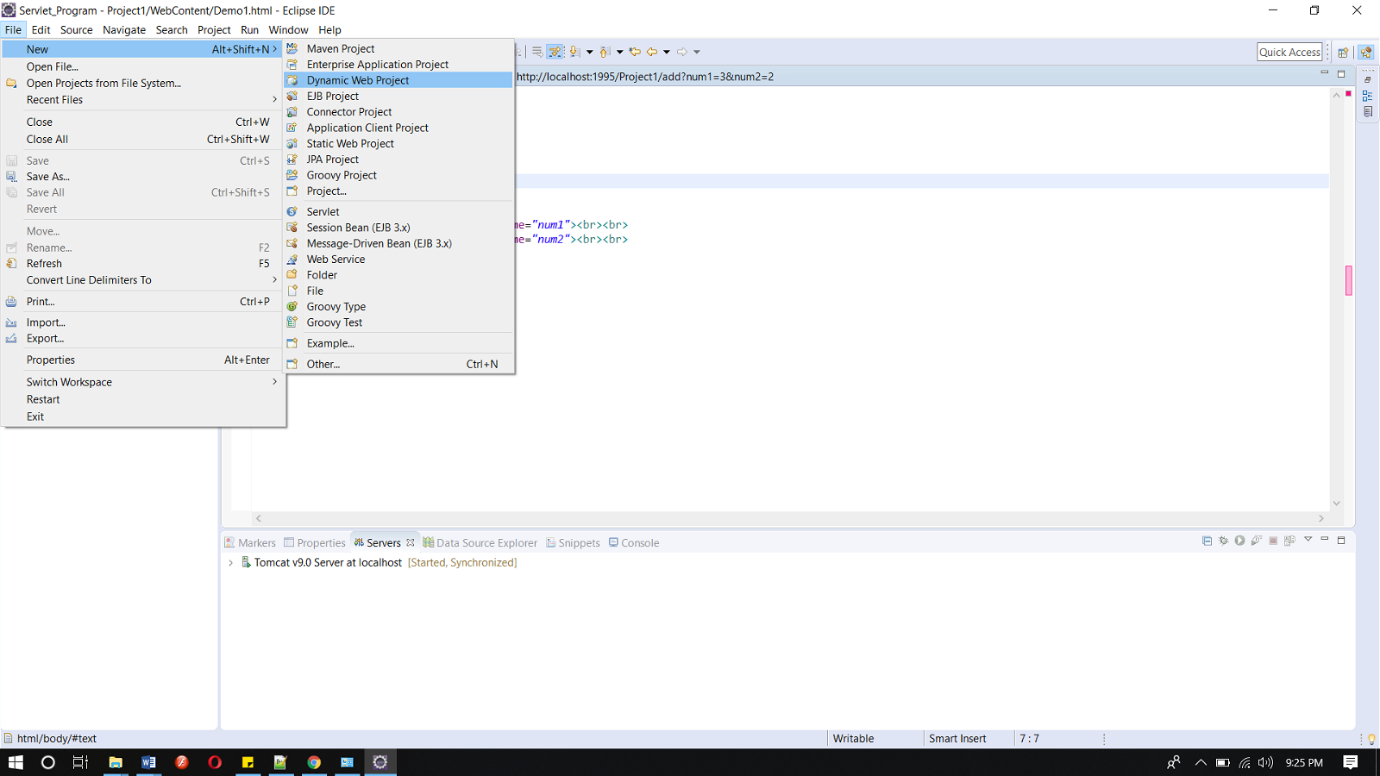
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Method** | **Description** |
| 1. | public void init(ServletConfigconfig) | It is used for initializing the servlet. It is invoked only once by the web container in a servlet life cycle. |
| 2. | public void service(ServletRequestreq, ServletResponse res) | 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. |
| 4. | public ServletConfiggetServletConfig() | It is used to get the object of ServletConfig. |
| 5. | Public String getServletInfo() | It is used to get information about writer, copyright etc of a servlet. |

### **Example of Servlet interface on Eclipse**

For creating a servlet interface below is the directory structure of the program:

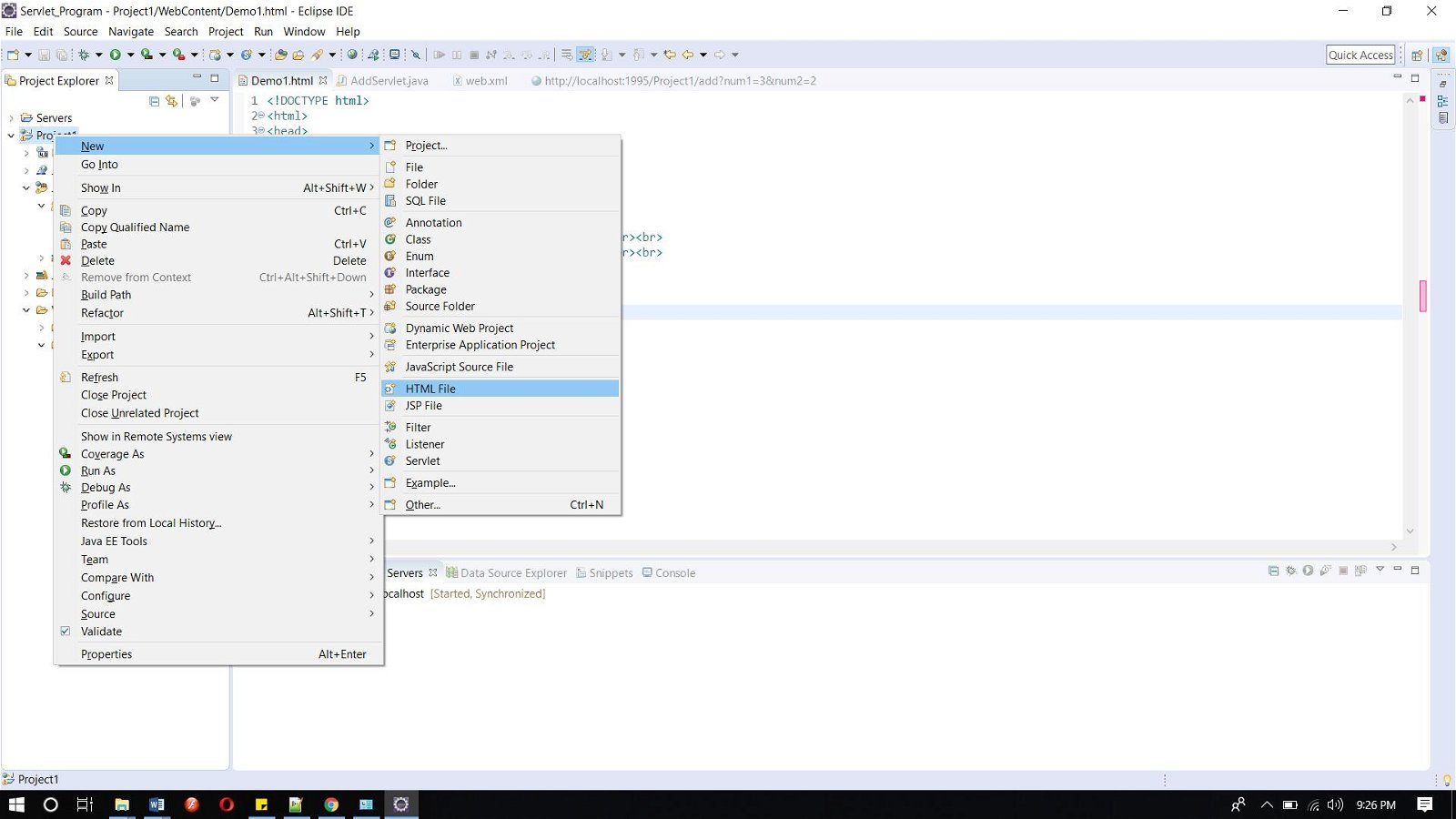
Following are the steps for creating the program.

**Step 1: Create a dynamic project on eclipse by clicking on File => New => Dynamic Web Project**



#### Step 2: Now create an HTML file.

Right-click on the project and then click on HTML file. Give the name of the file and then click on the finish button.



#### And write the below code.

**Index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>studytonight => servlet interface example</title>

</head>

<body>

<h1>studytonight.com</h1><br><br>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*<br><br>

<h3><a href="demo">Click here to proceed...</a></h3><br><br>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*<br><br>

</body>

</html>

Copy

#### Step 3: Now add the below code in web.xml file.

web.xml file is a deployment descripter. Here we have all the configurations.

<?xmlversion="1.0"encoding="UTF-8"?>

<web-appxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns="http://xmlns.jcp.org/xml/ns/javaee"xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"id="WebApp\_ID"version="4.0">

<servlet>

<servlet-name>abc</servlet-name>

<servlet-class>DemoServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>abc</servlet-name>

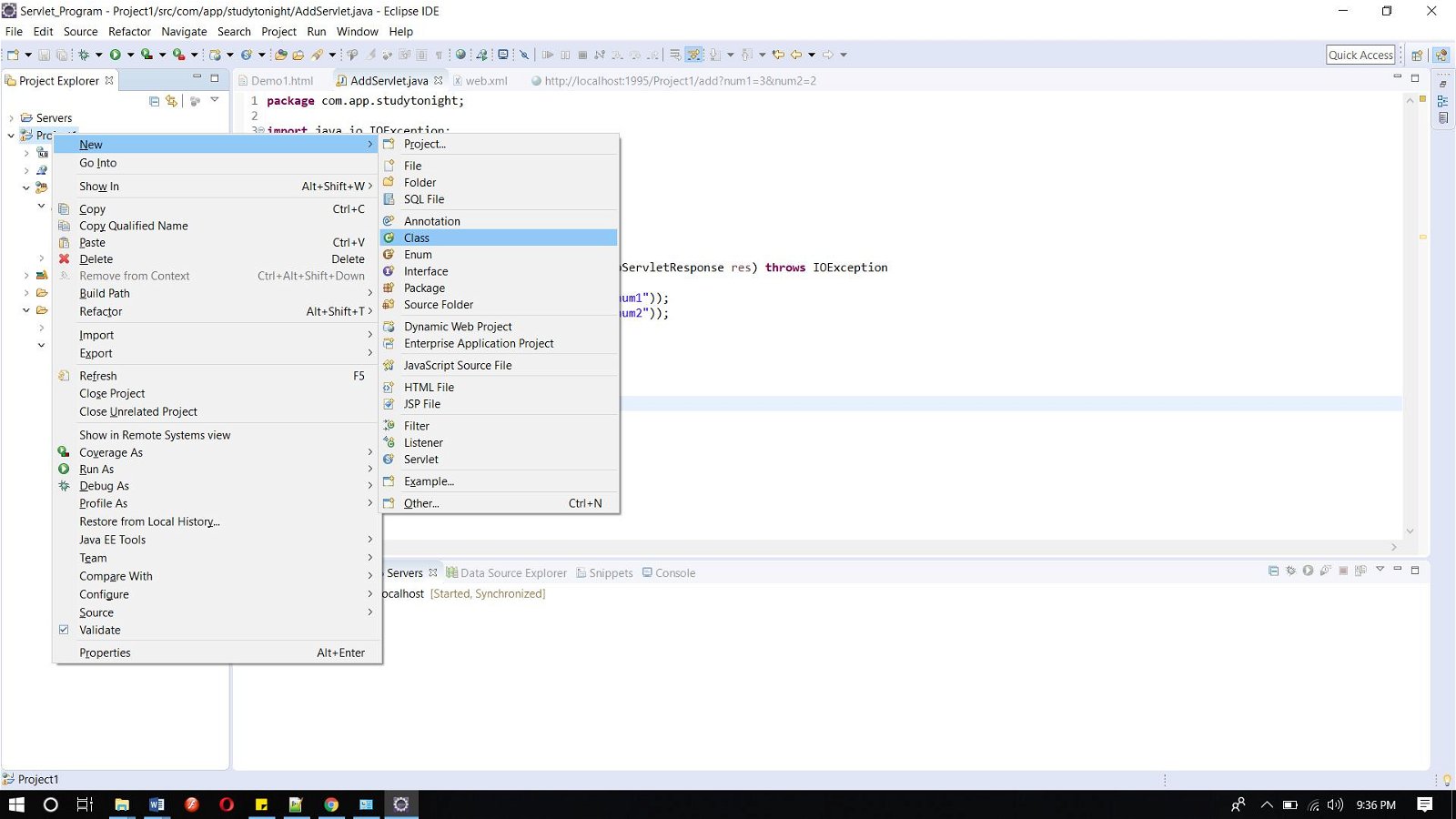
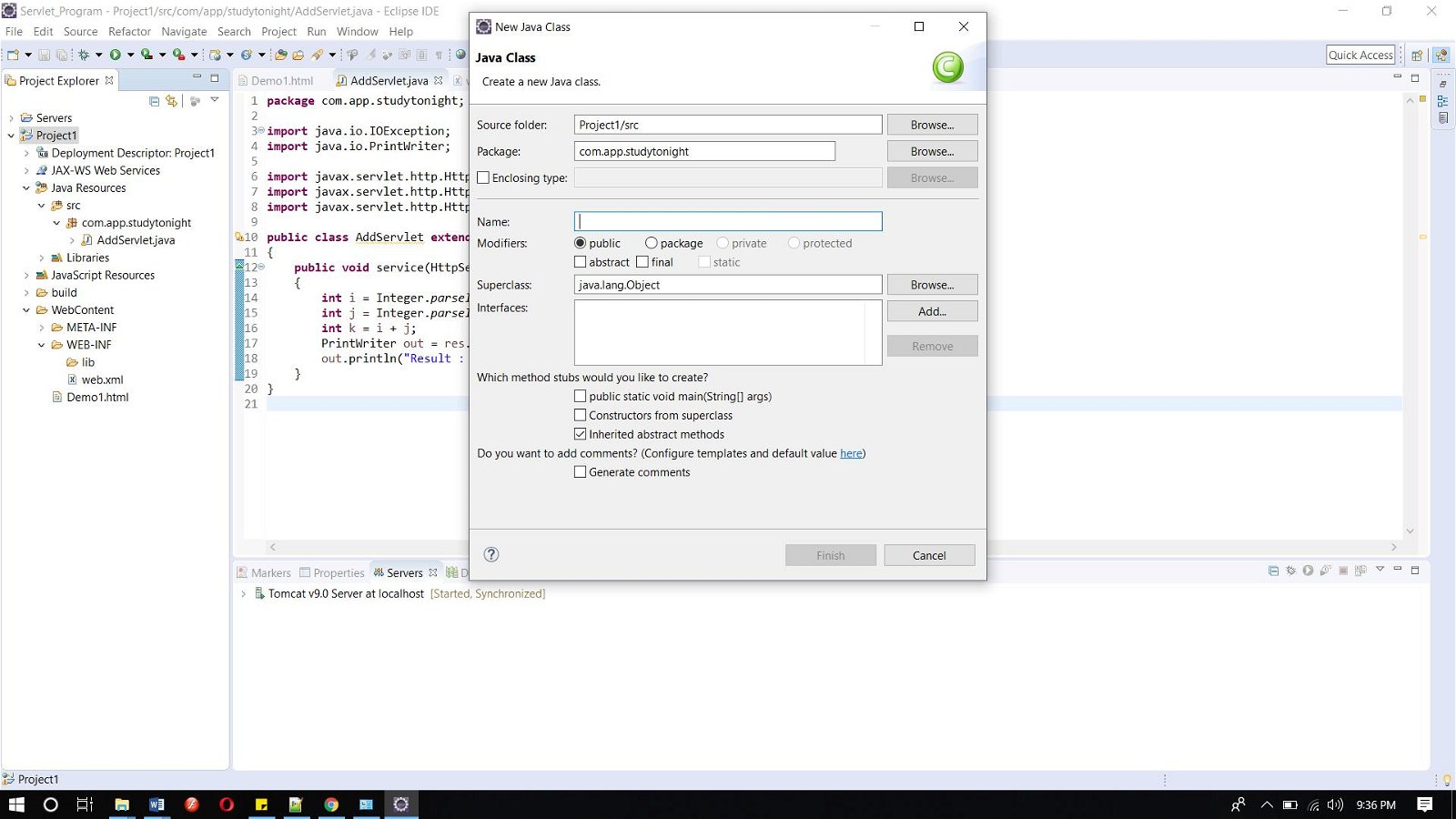
<url-pattern>/demo</url-pattern>

</servlet-mapping>

</web-app>

Copy

#### Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.

#### Now add the below code in the class file.

DemoServlet.java

import java.io.\*;

import javax.servlet.\*;

public class DemoServlet implements Servlet{

ServletConfig config=null;

public void init(ServletConfig config){

this.config=config;

}

public void service(ServletRequest req,ServletResponse res)

throws IOException,ServletException{

res.setContentType("text/html");

PrintWriter pwriter=res.getWriter();

pwriter.print("<html>");

pwriter.print("<body>");

pwriter.print("<h1>Hello Welcome to studytonight. This example is of servlet interface. </h1>");

pwriter.print("</body>");

pwriter.print("</html>");

}

public void destroy(){

System.out.println("servlet destroy");

}

public ServletConfig getServletConfig(){

return config;

}

public String getServletInfo(){

return "studytonight.com";

}

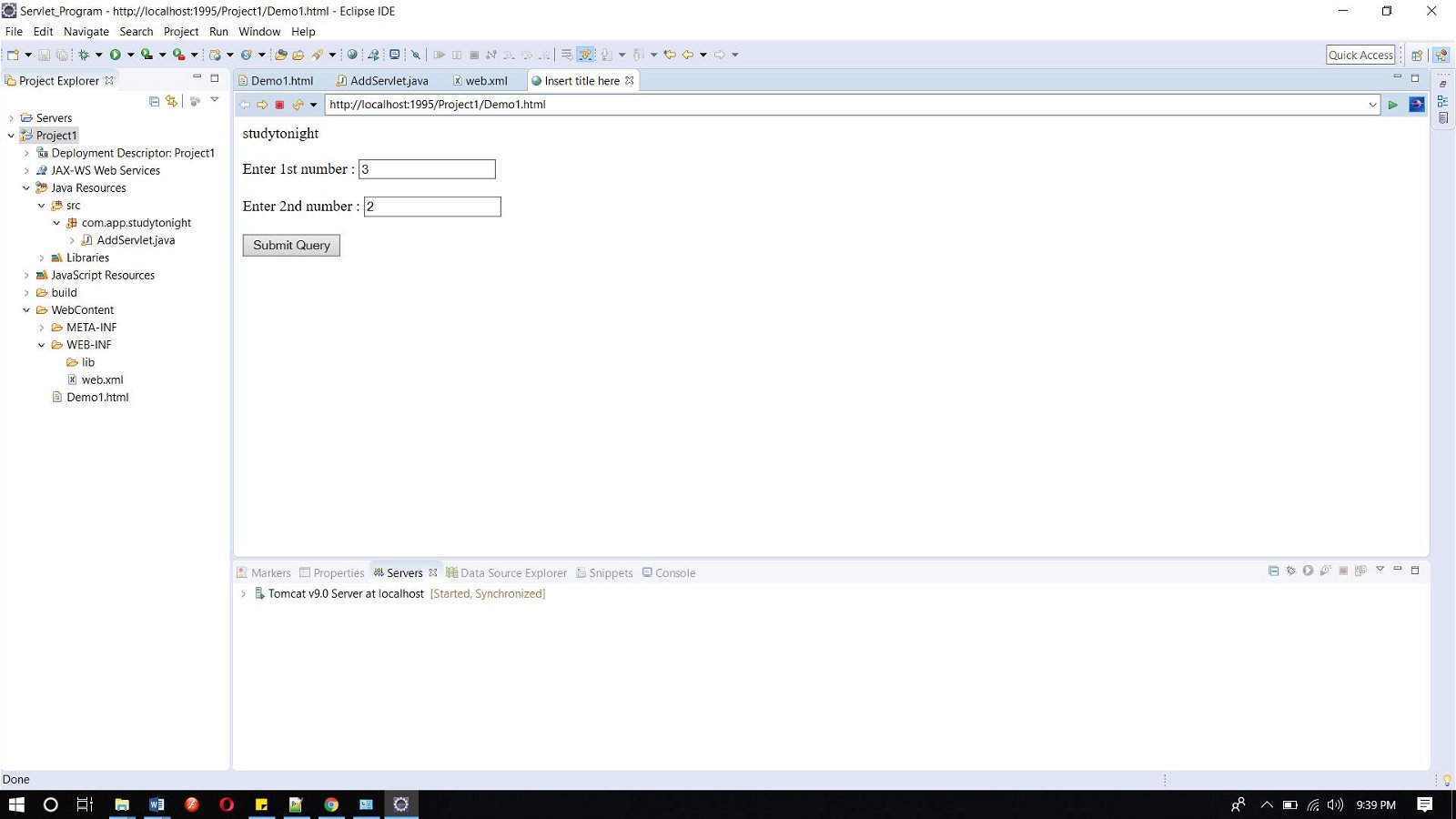
}

Copy

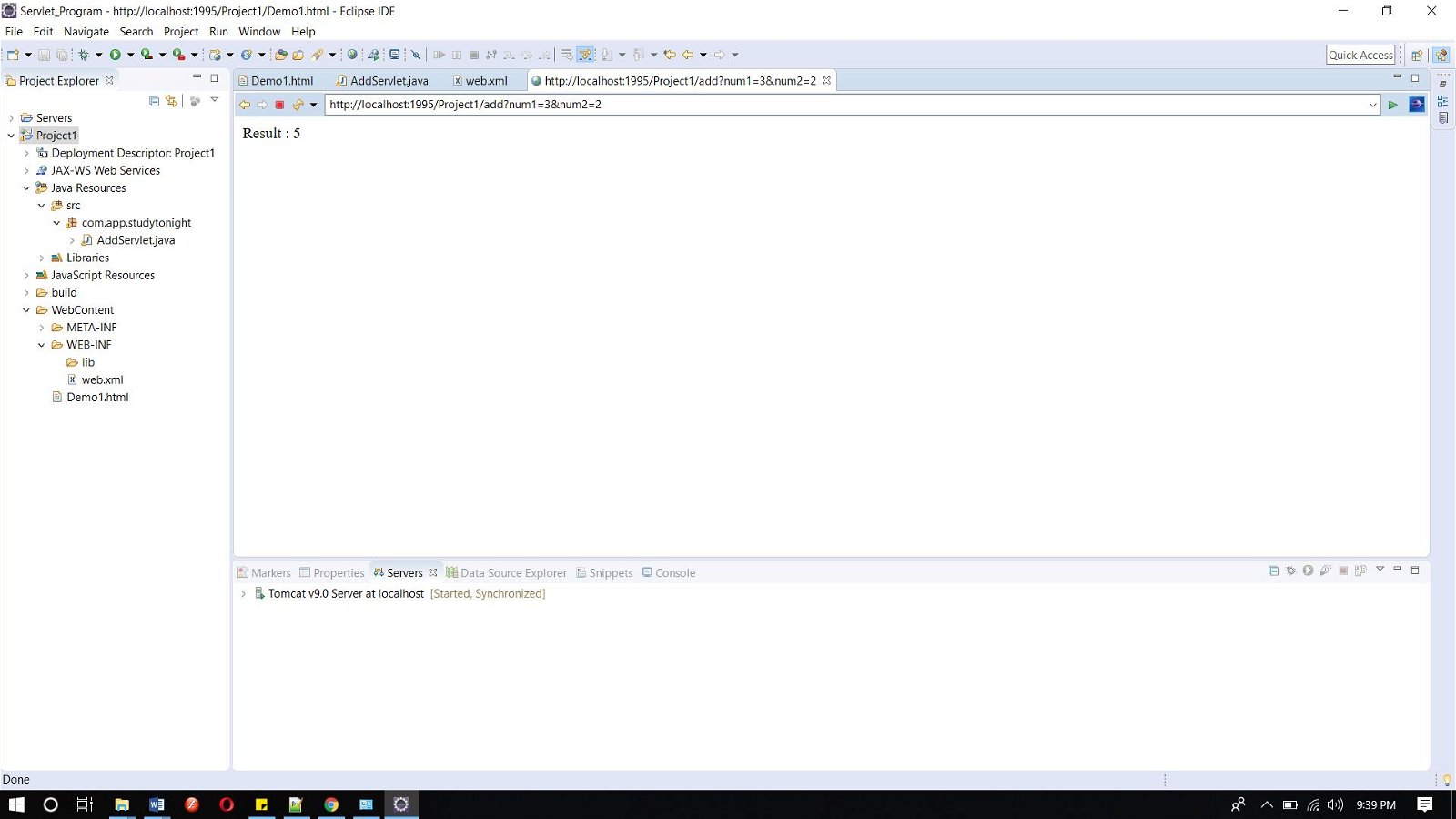
#### Now, Run the code.

To run the code, right-click on the project and select Run As => Run on Server.

Below is the index.html page. Click on the link for landing in the servlet page.



This is the servlet page.

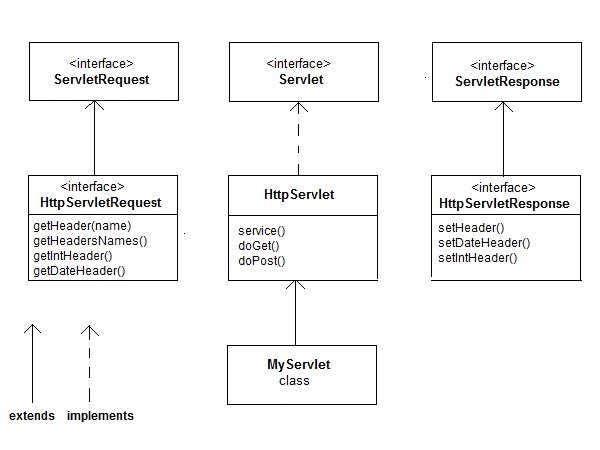


## **HttpServlet class**

HttpServlet is also an abstract class. This class gives implementation of various service() methods of **Servlet** interface.

To create a servlet, we should create a class that extends **HttpServlet** abstract class. The Servlet class that we will create, must not override service() method. Our servlet class will override only the doGet() and/or doPost() methods.

The 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 type.



### **Methods of HttpServlet interface**

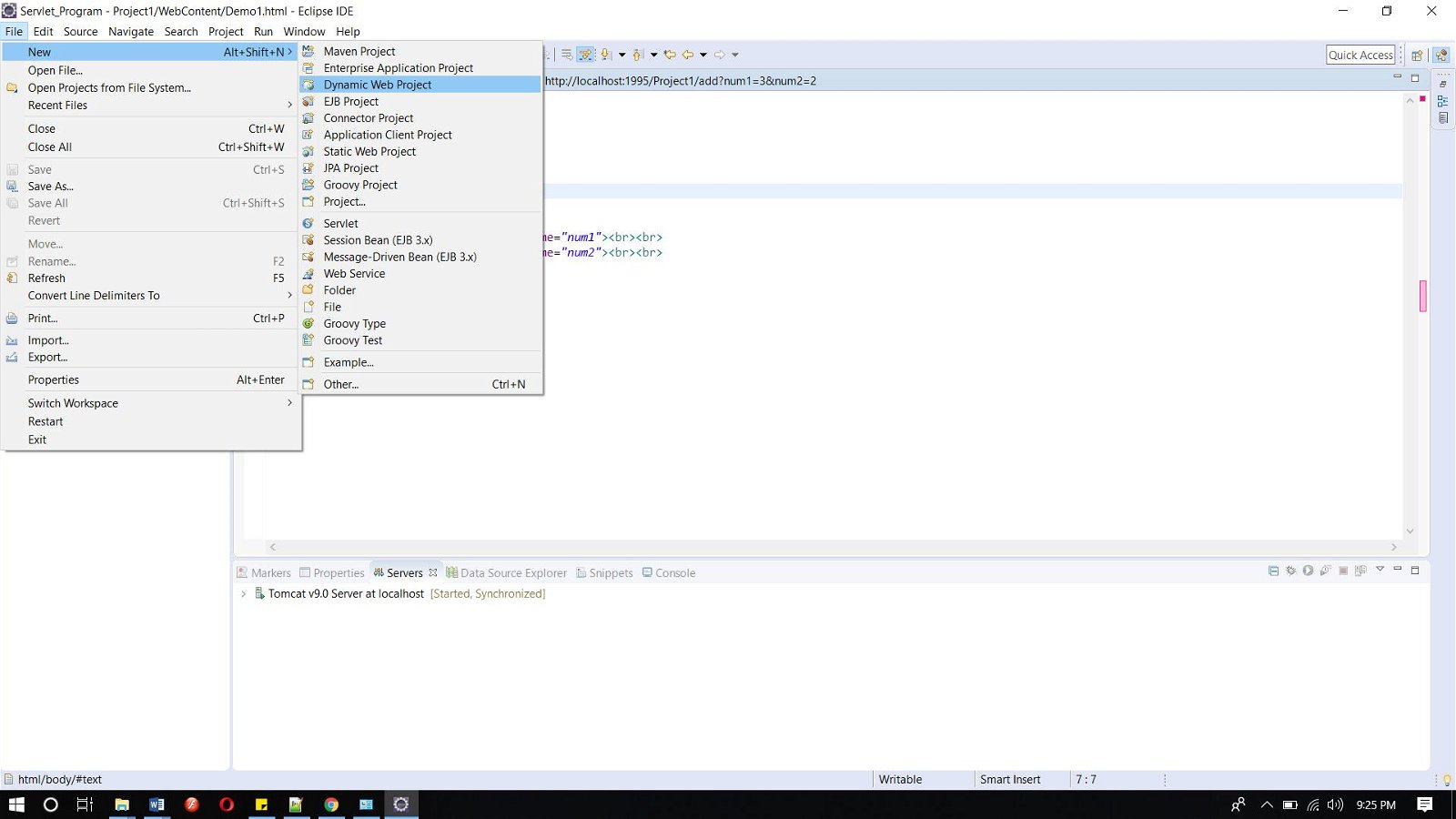
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Method** | **Description** |
| 1 | public void service(ServletRequest req,ServletResponse res) | It is used for securing the service method by creating objects of request and response. |
| 2 | protected void service(HttpServletRequest req, HttpServletResponse 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. |
| 6 | protected void doOptions(HttpServletRequest req, HttpServletResponse res) | It is invoked by the web container and it handles the OPTIONS request. |
| 7 | protected void doPut(HttpServletRequest req, HttpServletResponse res) | It is invoked by the web container and it handles the OPTIONS request. |
| 8 | protected void doTrace(HttpServletRequest req, HttpServletResponse res) | It is invoked by the web container and it handles the TRACE request |
| 9 | protected void doDelete(HttpServletRequest req, HttpServletResponse res) | It is invoked by the web container and it handles the DELETE request. |
| 10 | protected long getLastModified(HttpServletRequest req) | It is used for getting the time of last modified HttpServletRequest. |

### **Example of HttpServlet class on Eclipse**

For creating a HttpServlet class below is the directory structure of the program:

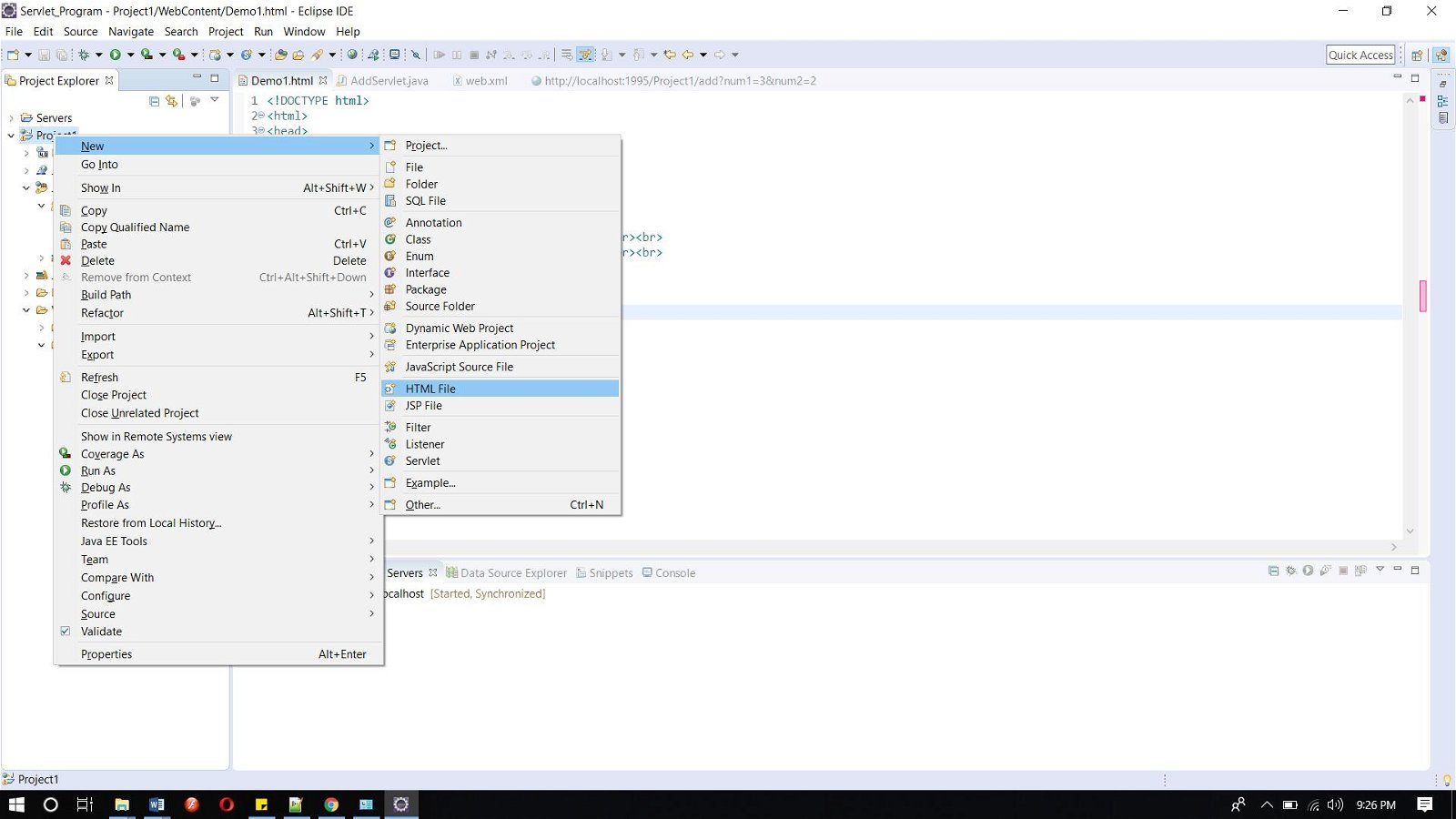
Following are the steps for creating the program.

**Step 1: Create a dynamic project on eclipse by clicking on File => New => Dynamic Web Project**



**Step 2: Now create an HTML file.**

Right-click on the project and then click on HTML file. Give the name of the file and then click on the finish button.



#### **And write the below code.**

**Demo.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<form action="mar" align="center">

<h3 align="center">studytonight.com</h3>

<h3 align="center">--------------------------------------------------------</h3>

Enter marks of the following subjects<br><br><br>

Maths : <input type="text" name="num1"><br><br>

English : <input type="text" name="num2"><br><br>

Hindi : <input type="text" name="num3"><br><br>

Science : <input type="text" name="num4"><br><br>

Social Science : <input type="text" name="num5"><br><br>

IT : <input type="text" name="num6"><br><br>

<input type="submit">

</form>

</body>

</html>

Copy

**Step 3: now add the below code in web.xml file.**

**web.xml file is a deployment descripter. Here we have all the configurations.**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

< <servlet>

<servlet-name>abc2</servlet-name>

<servlet-class>marks</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>abc2</servlet-name>

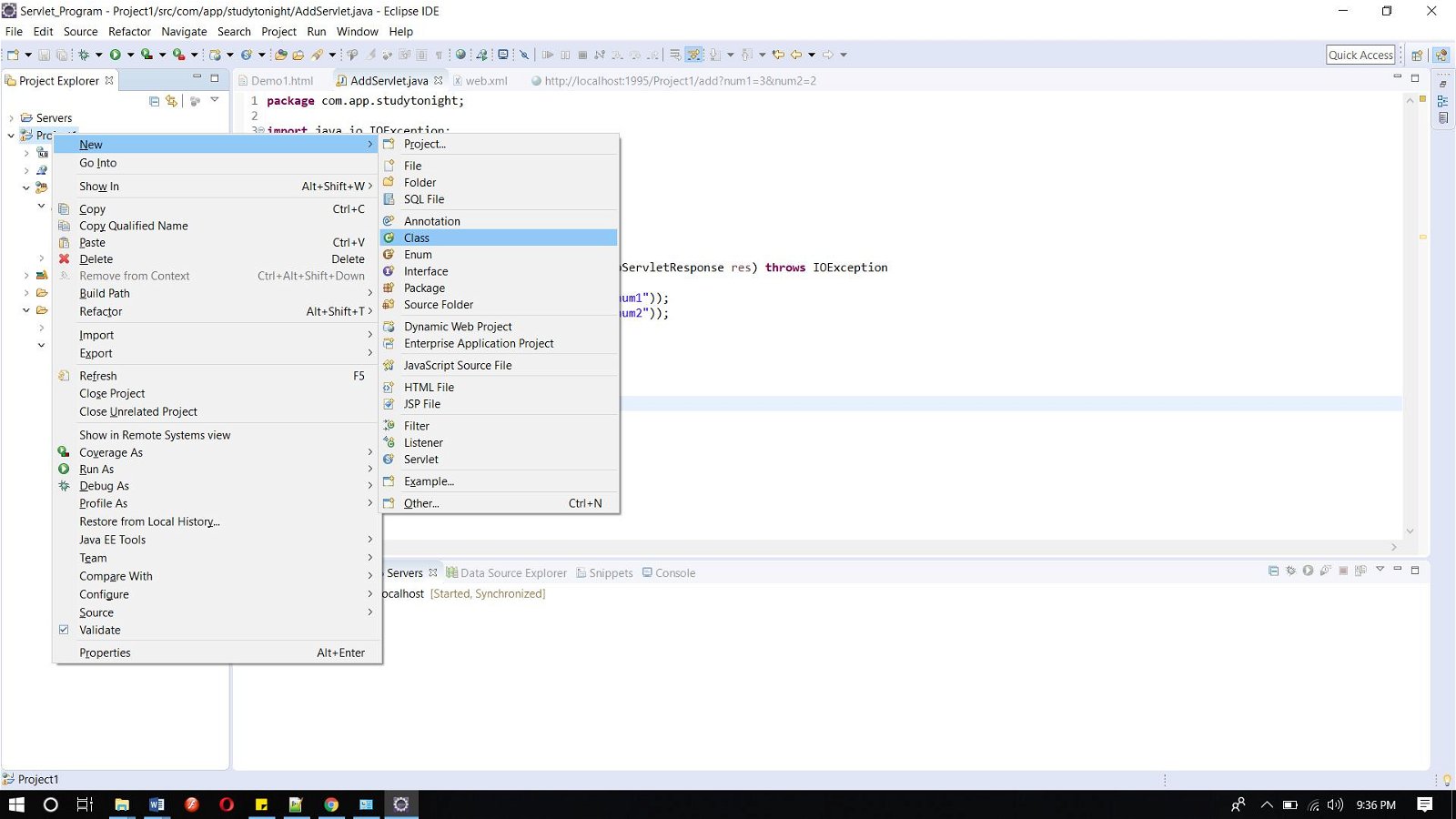
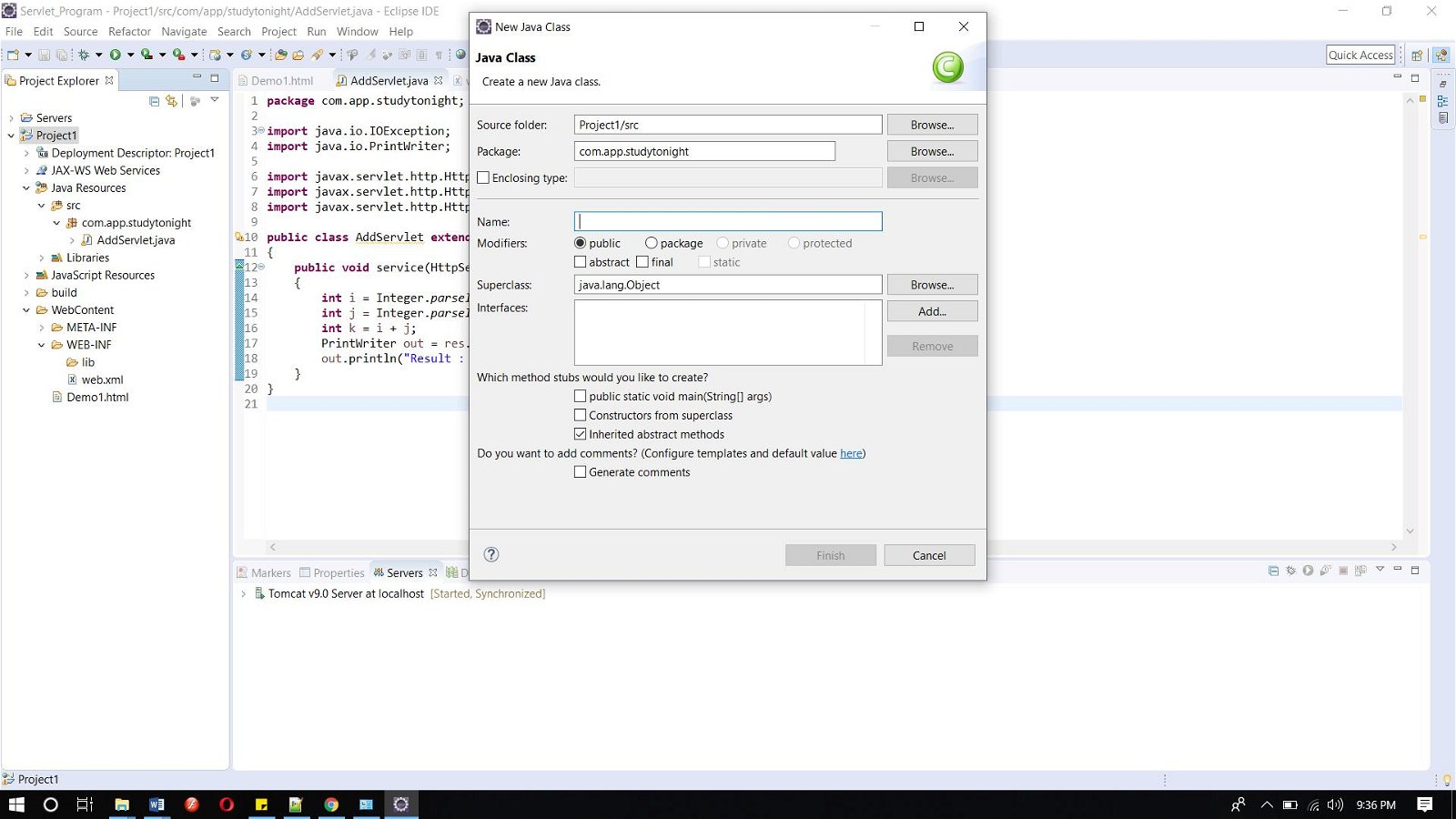
<url-pattern>/mar</url-pattern>

</servlet-mapping>

</web-app>

Copy

#### Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.

**Add the below code in the class file.**

**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 i = Integer.parseInt(req.getParameter("num1"));

int j = Integer.parseInt(req.getParameter("num2"));

int k = Integer.parseInt(req.getParameter("num3"));

int l = Integer.parseInt(req.getParameter("num4"));

int m = Integer.parseInt(req.getParameter("num5"));

int n = Integer.parseInt(req.getParameter("num6"));

int total = i + j + k + l + m + n;

float avg = total / 6;

PrintWriter out = res.getWriter();

out.println("Maths : " + i );

out.println("English : " + j );

out.println("Hindi : " + k);

out.println("Science : " + l);

out.println("Social Science : " + m);

out.println("IT : " + n);

out.println("Total Marks : "+ total);

out.println("Average: "+avg);

}

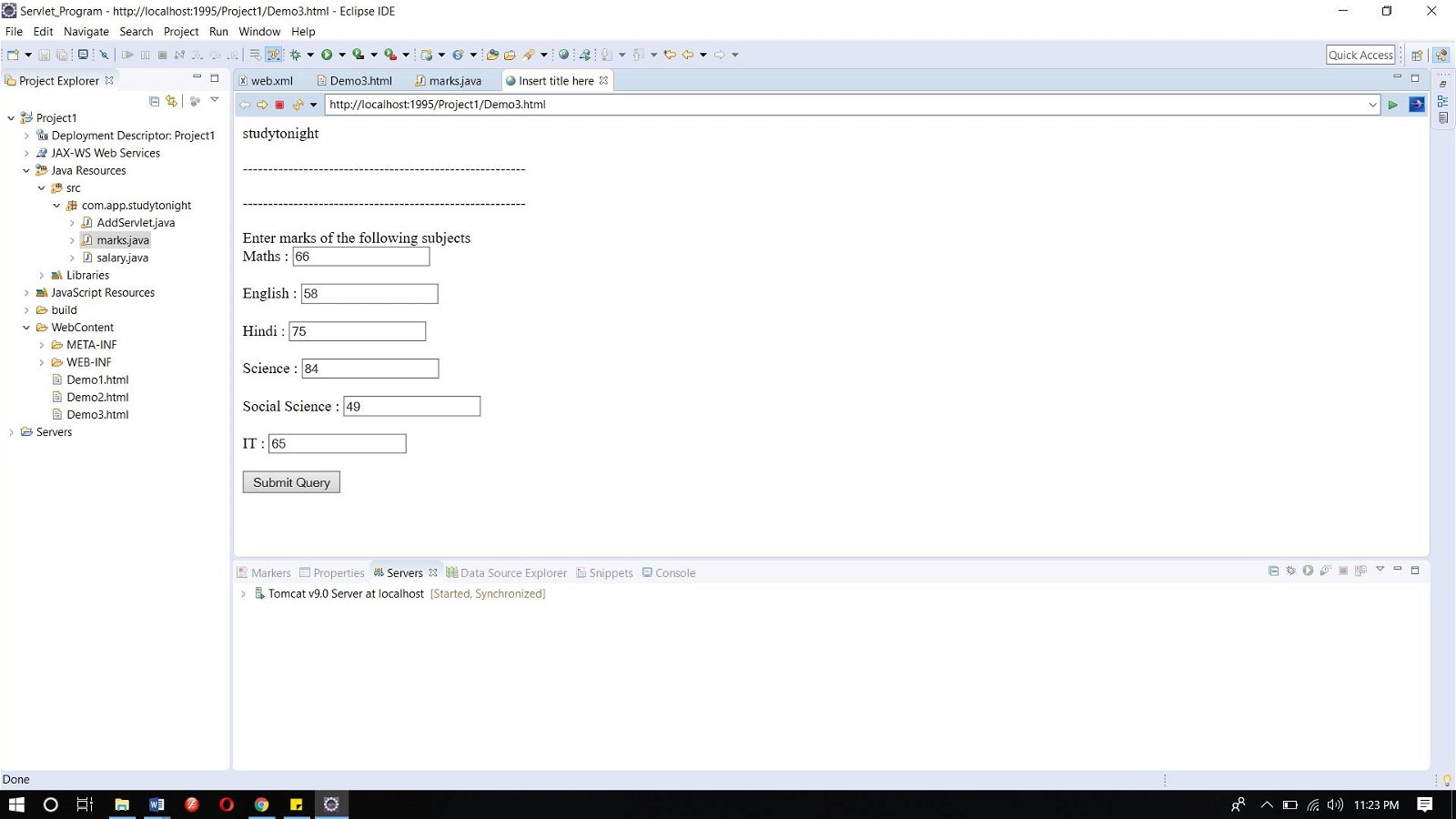
}

Copy

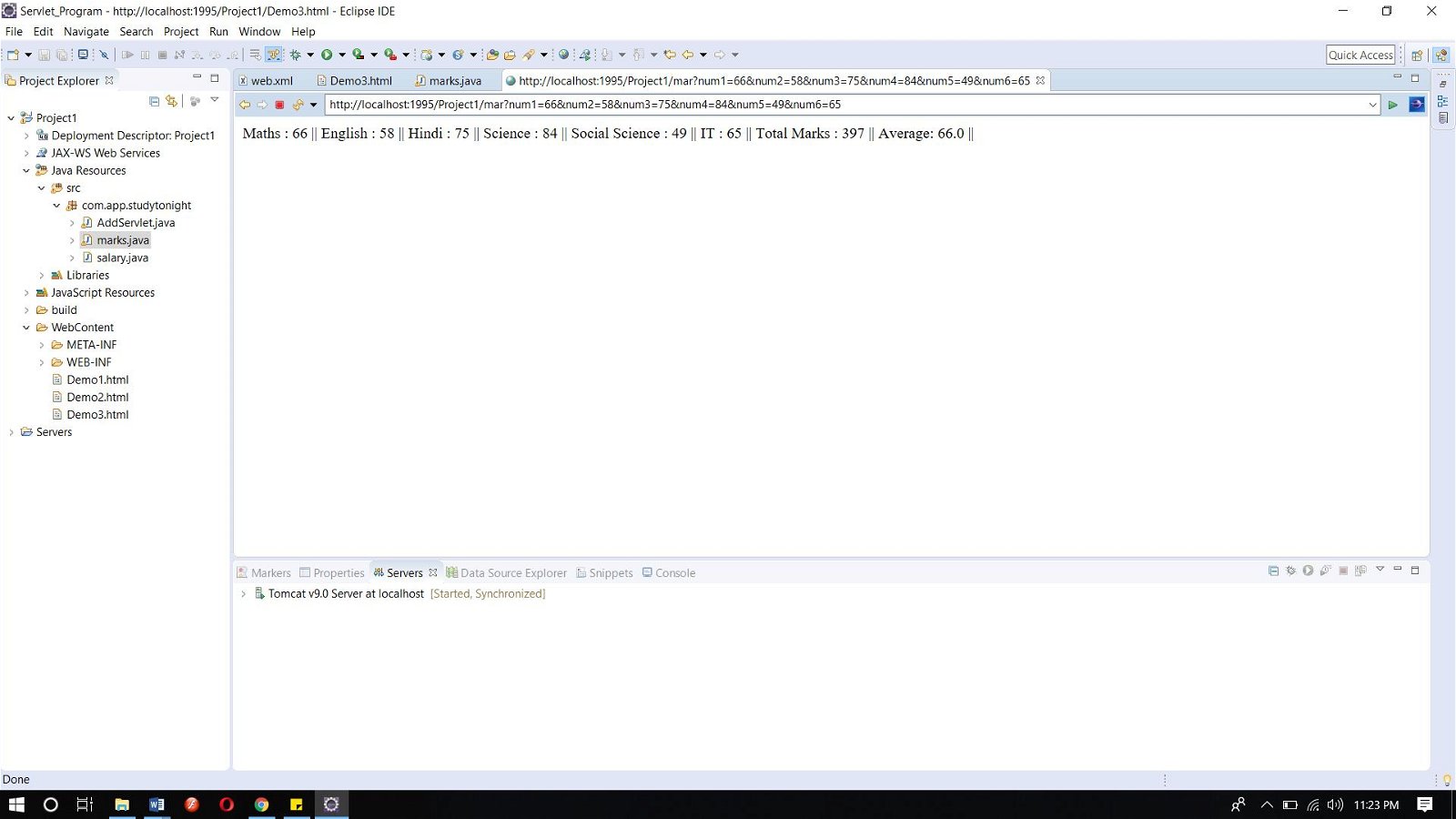
#### Now, Run the code.

To run the code, right-click on the project and select Run As => Run on Server.

Below is the index.html page. Click on the link for landing in the servlet page.



This is the servlet page.



## **GenericServlet class**

In Servlet, GenericServlet is an abstract class. This class implements the servlet, ServletConfig and Serializable interface. This class provides the implementation of most of the basic servlet methods. The protocol of this class is independent as it can handle any type of request.

#### Class:

### **Methods of GenericServlet interface**

#### Implemented Interfaces:

java.io.Serializable, Servlet, ServletConfig

#### Constructor:

GenericServlet() : this constructor does nothing. Everything is initialized by the init method.

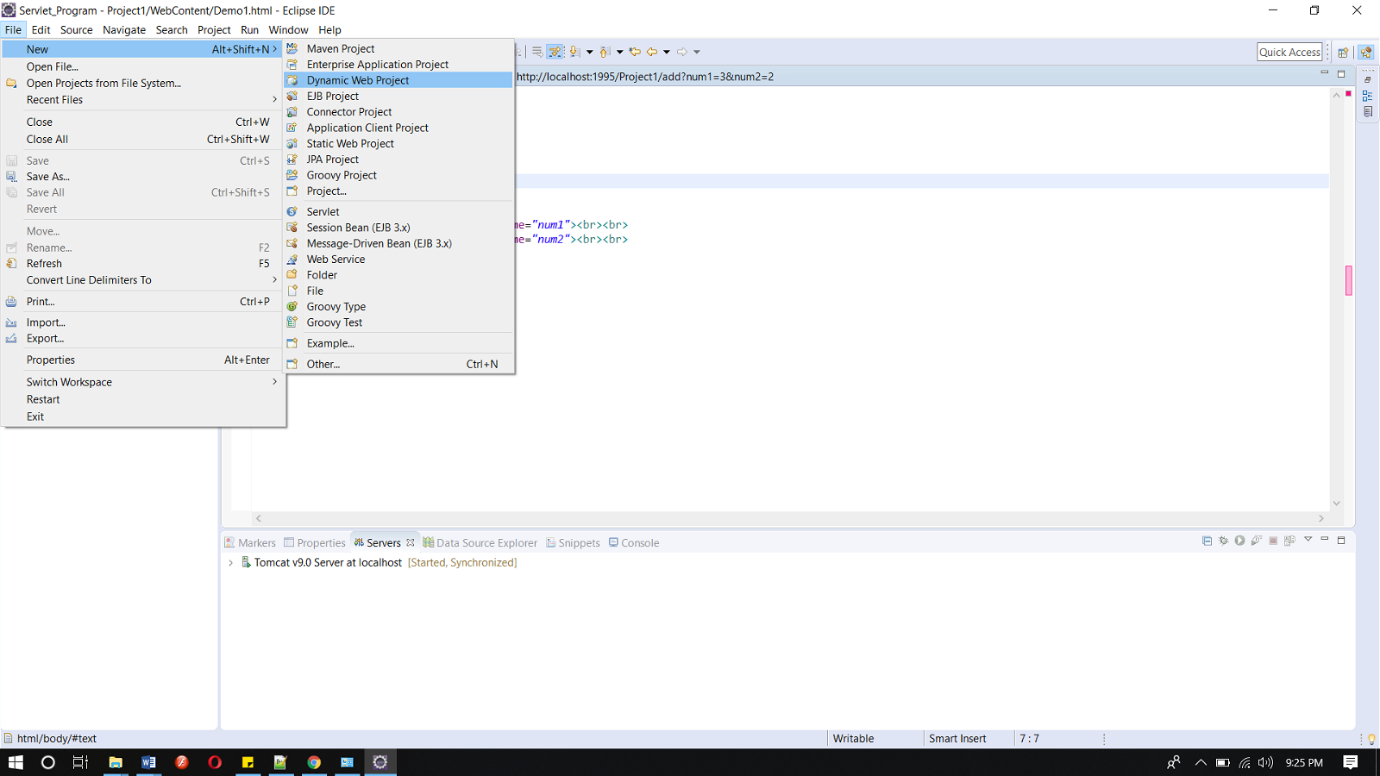
|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Method** | **Desciption** |
| 1 | public void init(ServletConfig config) | It is used for initialization of a servlet. |
| 2 | public abstract void service(ServletRequest request, ServletResponse response) | It is used for providing all the services for the incoming request. When a user request then only it invokes. |
| 3 | public void destroy() | It is used for destroying the servlet. It is invoked only once in a life cycle of a servlet. |
| 4 | public ServletConfig getServletConfig() | It is used to get the object of ServletConfig |
| 5 | public String getServletInfo() | It is used to get information about writer, copyright etc of a servlet. |
| 6 | public void init() | It is a very easy and convenient method for programmers. |
| 7 | public ServletContext getServletContext() | It is used for getting object of a servlet |
| 8 | public String getInitParameter(String name) | It is used for getting all the parameter values from the given parameter names. |
| 9 | public Enumeration getInitParameterNames() | It is used for getting parameters which are defined in web.xml files |
| 10 | public String getServletName() | It is used for getting the name of a servlet object. |
| 11 | public void log(String msg) | It is used for writing a message in a servlet log file. |
| 12 | public void log(String msg, Throwable t) | It is used for writing a message in a servlet log file and stack trace. |

### **Example of GenericServlet interface on Eclipse**

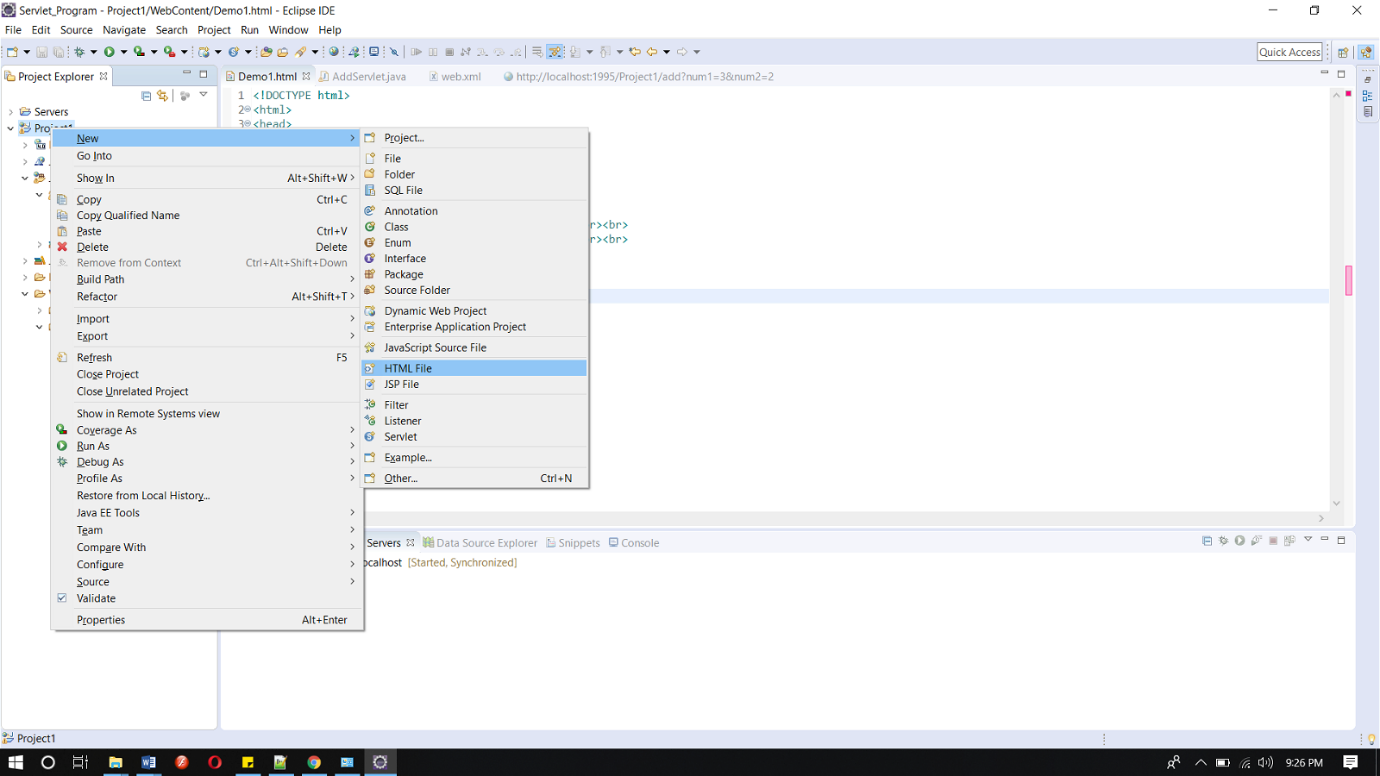
For creating a GenericServlet class below is the directory structure of the program:

Following are the steps for creating the program

**Step 1: Create a dynamic project on eclipse by clicking on File => New => Dynamic Web Project**



**Step 2: Now create an HTML file.**



**And write the below code.**

**Demo2.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Studytonight.com</title>

</head>

<body>

<form action="sal" align="center">

<h3 align="center">studytonight.com</h3>

<h3 align="center">--------------------------------------------------------</h3><br><br>

Enter Basic Salary <input type="text" name="num1"><br><br>

Enter Basic DA <input type="text" name="num2"><br><br>

Enter Basic HRA <input type="text" name="num3"><br><br>

<input type="submit">

</form>

</body>

</html>

Copy

#### Step 3: now add the below code in web.xml file.

**web.xml file is a deployment descripter. Here we have all the configurations.**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

<display-name>Generic\_Servlet</display-name>

<servlet>

<servlet-name>abc1</servlet-name>

<servlet-class>salary</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>abc1</servlet-name>

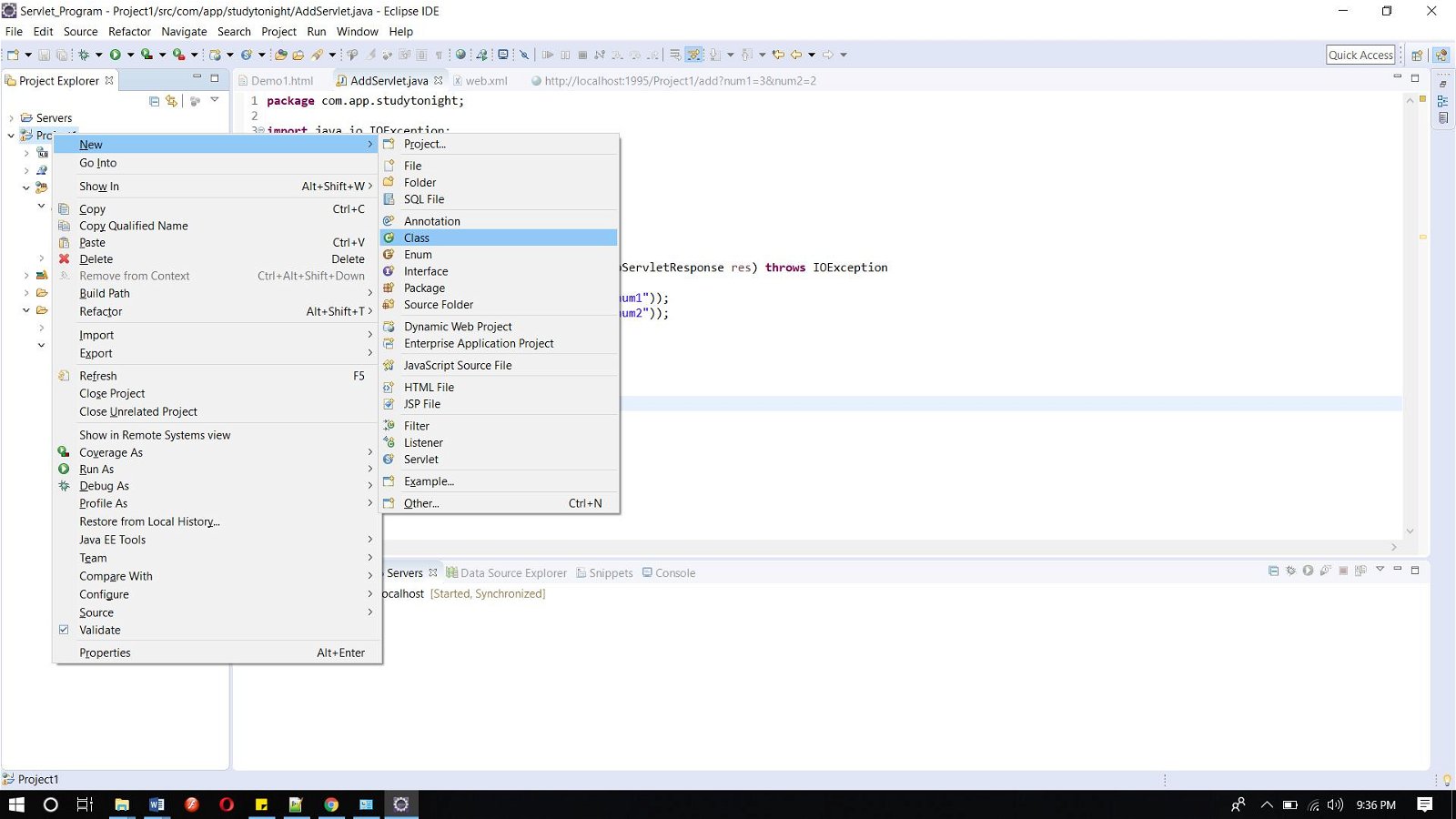
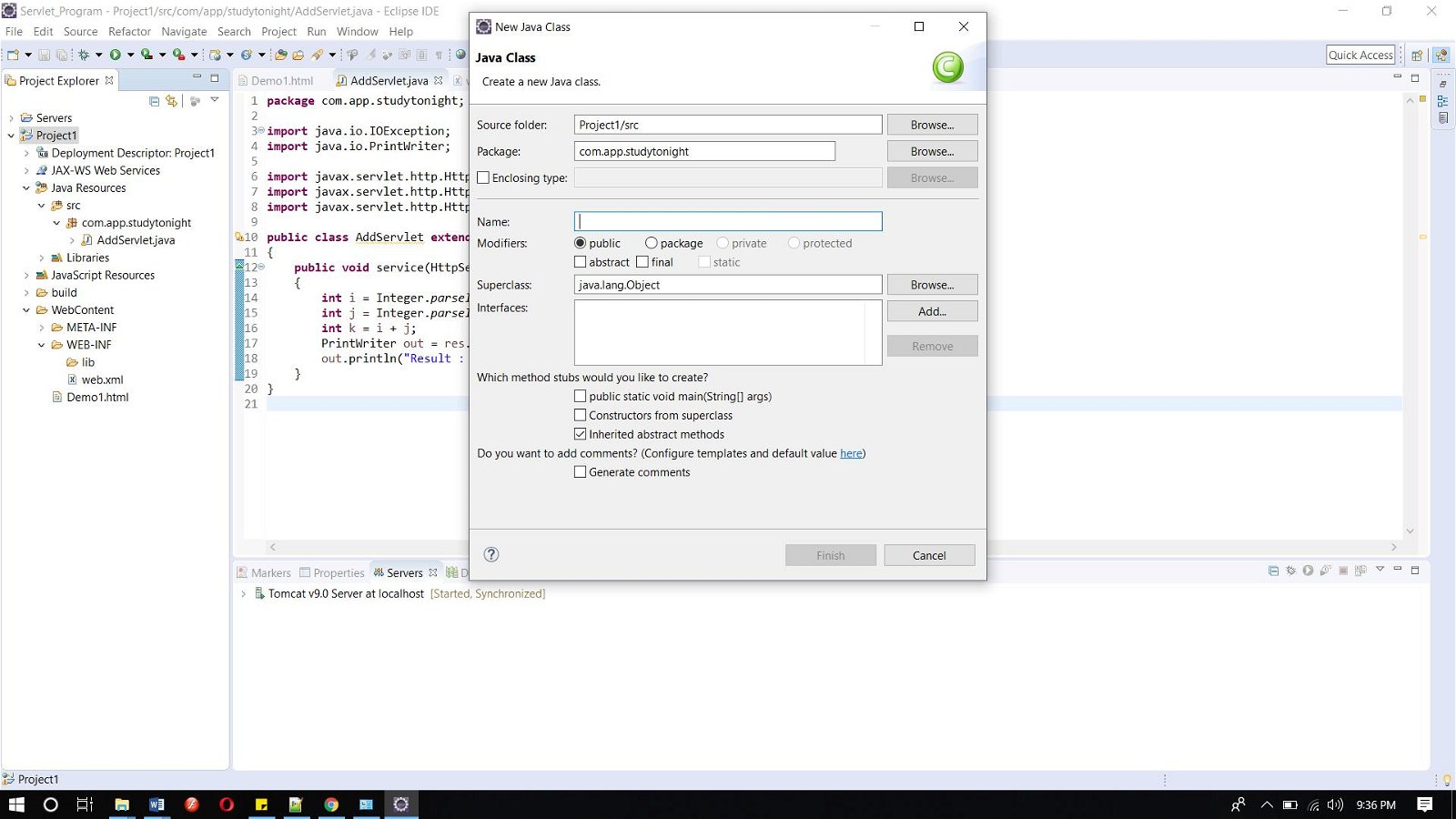
<url-pattern>/sal</url-pattern>

</servlet-mapping>

</web-app>

Copy

#### Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.

#### Add the below code in the class file.

**salary.java**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

public class salary extends GenericServlet

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public void service(ServletRequest req, ServletResponse res) throws IOException, ServletException

{

int i = Integer.parseInt(req.getParameter("num1"));

int j = Integer.parseInt(req.getParameter("num2"));

int k = Integer.parseInt(req.getParameter("num3"));

int da = (j \* i) / 100;

int hra = (k \* i) / 100;

int g = i + da + hra;

PrintWriter out = res.getWriter();

out.println("studytonight.com");

out.println("DA : "+da);

out.println("HRA : "+hra);

out.println("Gross Salary : "+g);

}

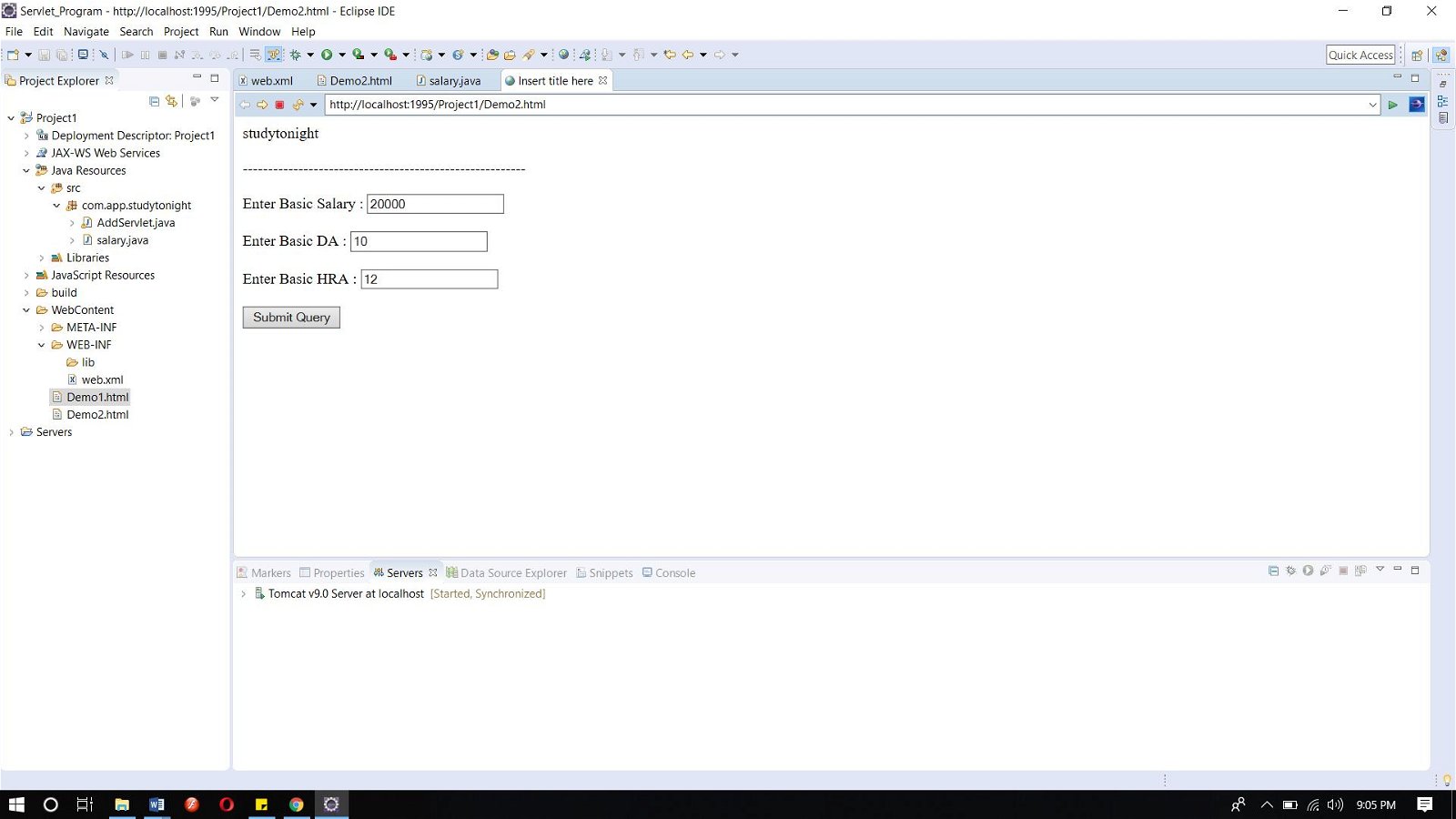
}

Copy

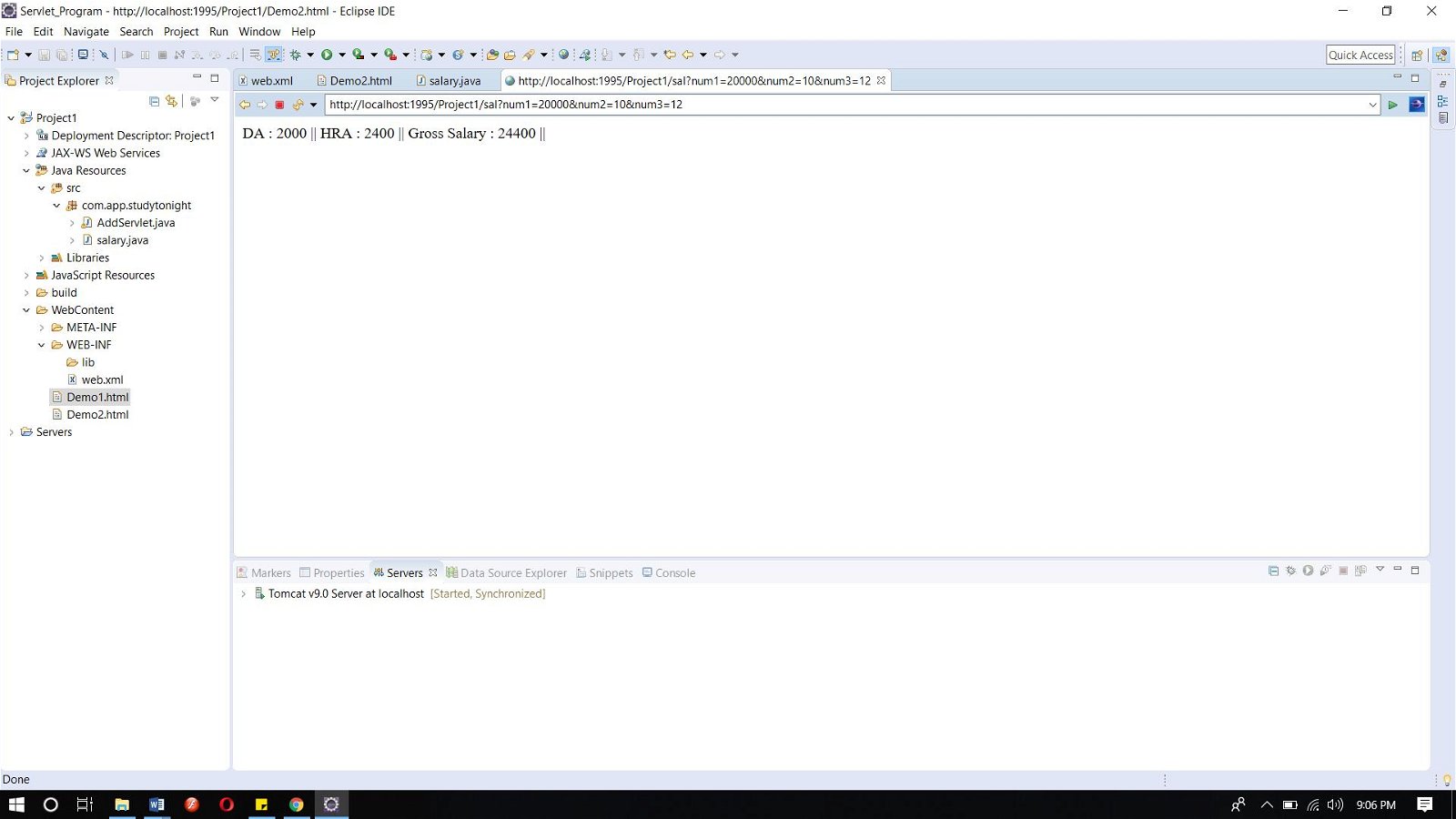
#### Now, Run the code.

To run the code, right-click on the project and select Run As => Run on Server.

Below is the demo2.html page. Click on the link for landing in the servlet page.



This is the servlet page.



# How a Servlet Application works

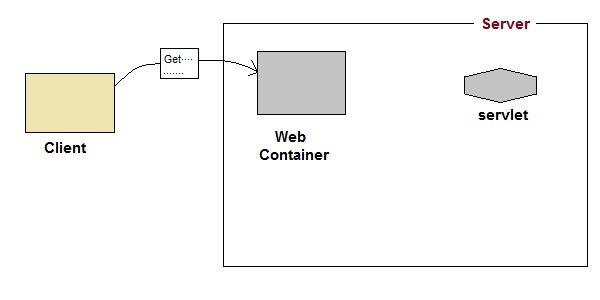
**Web container** is responsible for managing execution of servlets and [JSP pages](https://www.studytonight.com/jsp/introduction-to-jsp.php) for Java EE application.

When a request comes in for a servlet, the server hands the request to the Web Container. **Web Container** is responsible for instantiating the servlet or creating a new thread to handle the request. Its the job of Web Container to get the request and response to the servlet. The container creates multiple threads to process multiple requests to a single servlet.

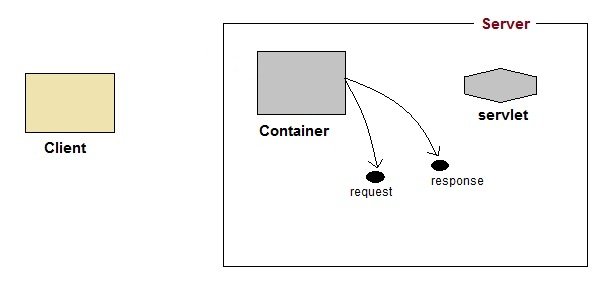
**Servlets don't have a main() method**. Web Container manages the life cycle of a Servlet instance.

## **Quick Revision on How a Servlet works**

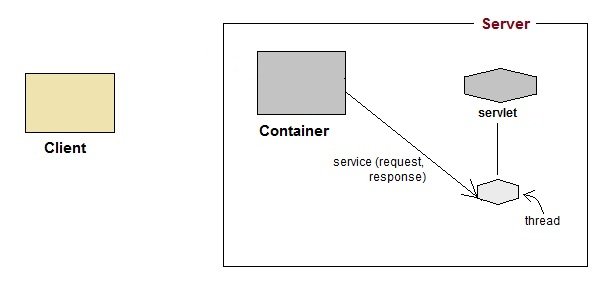
1. User sends request for a servlet by clicking a link that has URL to a servlet.



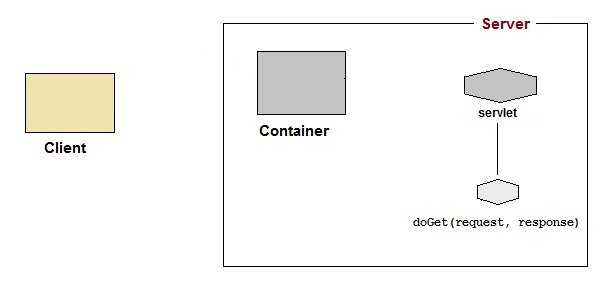
1. The container finds the servlet using **deployment descriptor** and creates two objects :
   1. [**HttpServletRequest**](https://www.studytonight.com/servlet/servlet-request.php)
   2. [**HttpServletResponse**](https://www.studytonight.com/servlet/servlet-response.php)



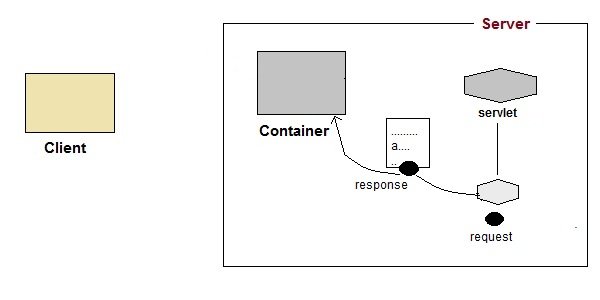
1. Then the container creates or allocates a thread for that request and calls the Servlet's service() method and passes the **request, response** objects as arguments.



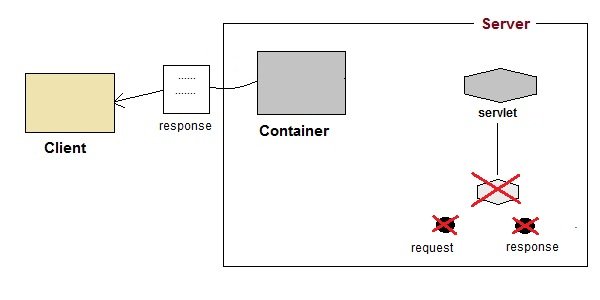
1. The service() method, then decides which servlet method, doGet() or doPost() to call, based on **HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the service() will call Servlet's doGet() method.



1. Then the Servlet uses response object to write the response back to the client.



1. After the service() method is completed the **thread** dies. And the request and response objects are ready for **garbage collection**.



1. **Loading Servlet Class :** A Servlet class is loaded when first request for the servlet is received by the Web Container.
2. **Servlet instance creation :**After the Servlet class is loaded, Web Container creates the instance of it. Servlet instance is created only once in the life cycle.
3. **Call to the init() method :** init() method is called by the Web Container on servlet instance to initialize the servlet.

**Signature of init() method :**

public void init(ServletConfig config) throws ServletException

Copy

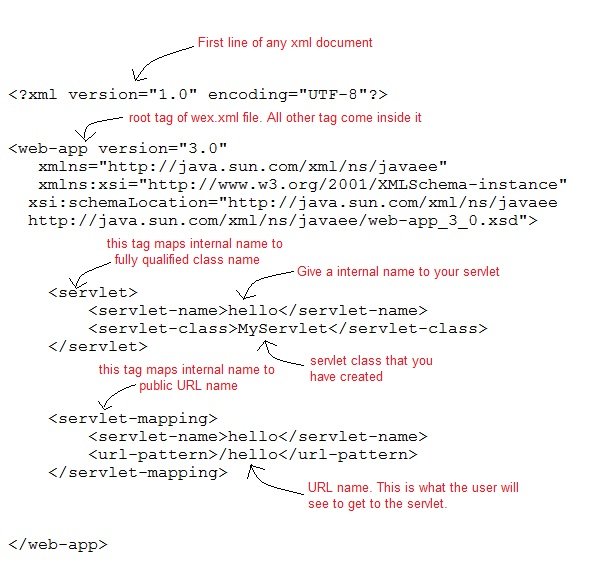
1. **Call to the service() method :** The containers call the service() method each time the request for servlet is received. The service() method will then call the doGet() or doPost() methos based ont eh type of the HTTP request, as explained in previous lessons.

**Signature of service() method :**

public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException

Copy

1. **Call to destroy() method:** The Web Container call the destroy() method before removing servlet instance, giving it a chance for cleanup activity.



# Request Dispatcher in Servlet

In Java, the RequestDispatcher Interface is used for dispatching the request to a resource i.e Html, servlet or JSP. The Contents of another resource can be included in this interface. There are two methods of RequestDispatcher. They are as following:

## **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. |

### **forward() method:**



### **include() method:**



### **Example of forward() and include() method on Eclipse**

For creating a program using forward() and include() method below is the directory structure of the program:

### **Following are the steps for creating the program.**

Step 1: Create a dynamic project on eclipse by clicking on **File => New => Dynamic Web Project**

Step 2: Now create an HTML file.

Right-click on the project and then click on HTML file. Give the name of the file and then click on the finish button.

And write the below code.

**Index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>studytonight</title>

</head>

<body>

<form align="center" action="display" method="post">

<h3>studytonight.com</h3>

<hr>

Name: <input type="text" name="val1"><br> <br>

User Id: <input type="text" name="val2"><br> <br>

Password: <input type="password" name="val3"><br> <br>

<input type="submit" value="login">

</form>

</body>

</html>

Copy

Step 3: Now add the below code in web.xml file.

web.xml file is a deployment descripter. Here we have all the configurations.

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

<display-name>forward</display-name>

<servlet>

<servlet-name>demo5</servlet-name>

<servlet-class>forward.demo5</servlet-class>

</servlet>

<servlet>

<servlet-name>demo5i</servlet-name>

<servlet-class>forward.demo5i</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>demo5</servlet-name>

<url-pattern>/display</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>demo5i</servlet-name>

<url-pattern>/display1</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

Copy

Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.

For this example we need two servlet classes.

Now add the below code in the class file.

**demo5.java**

package forward;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class demo5 extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String n=request.getParameter("val1");

String u=request.getParameter("val2");

String p=request.getParameter("val3");

if(p.equals("studytonight"))

{

RequestDispatcher rd=request.getRequestDispatcher("display1");

rd.forward(request, response);

}

else{

out.print("Incorrect UserId or Password");

RequestDispatcher rd=request.getRequestDispatcher("/index.html");

rd.include(request, response);

}

}

}

Copy

**demo5i.java**

package forward;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class demo5i extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String n=request.getParameter("val2");

out.print("Welcome "+n);

}

}

Copy

Now, Run the code.

To run the code, right-click on the project and select Run As => Run on Server.

Below is the index.html page. Fill all the fields and click on the login button for landing in the servlet page.

If your password is correct then it will land on the servlet page demo5.java

If your password is incorrect then it will land on demo5i.java page.

### **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);

Copy



**OR**

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

rs.include(request,response);

Copy



### **Example demonstrating usage of RequestDispatcher**

In this example, we will show you how 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.

**index.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>

Copy

**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();

}

}

}

Copy

**Welcome.java**

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();

}

}

}

Copy

**web.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>

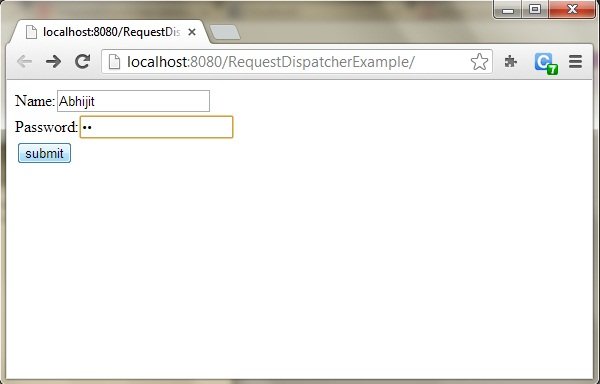
<welcome-file>index.html</welcome-file>

</welcome-file-list>

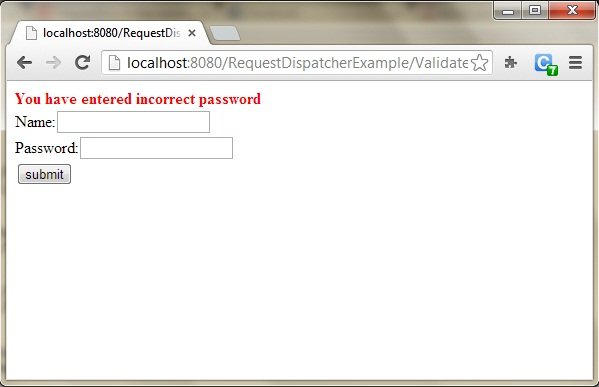
</web-app>

Copy

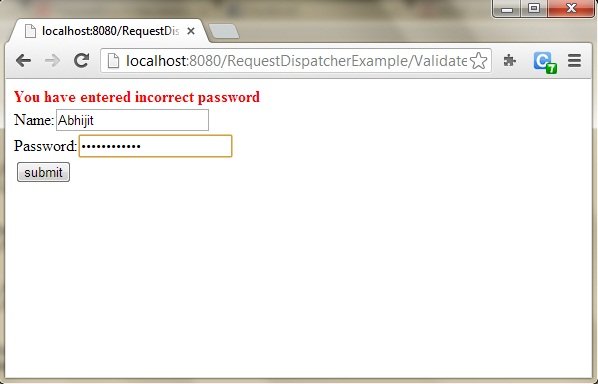
This will be the first screen. You can enter your Username and Password here.



When you click on Submit, Password will be validated, if it is not 'studytonight' , error message will be displayed.



Enter any Username, but enter 'studytonight' as password.



Password will be successfully validated and you will be directed to the Welcome Servlet.



**sendRedirect()** accepts relative **URL**,

### **Syntax of sendRedirect() method**

1. **public** **void** sendRedirect(String URL)**throws** IOException;

### **Example of sendRedirect() method**

1. response.sendRedirect("http://www.javatpoint.com");

### **Full example of sendRedirect method in servlet**

|  |
| --- |
| In this example, we are redirecting the request to the google server. Notice that sendRedirect method works at client side, that is why we can our request to anywhere. We can send our request within and outside the server. |

*DemoServlet.java*

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
5. **public** **class** DemoServlet **extends** HttpServlet{
6. **public** **void** doGet(HttpServletRequest req,HttpServletResponse res)
7. **throws** ServletException,IOException
8. {
9. res.setContentType("text/html");
10. PrintWriter pw=res.getWriter();
12. response.sendRedirect("http://www.google.com");
14. pw.close();
15. }}

### **Creating custom google search using sendRedirect**

In this example, we are using sendRedirect method to send request to google server with the request data.

*index.html*

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="ISO-8859-1"**>**
5. **<title>**sendRedirect example**</title>**
6. **</head>**
7. **<body>**

10. **<form** action="MySearcher"**>**
11. **<input** type="text" name="name"**>**
12. **<input** type="submit" value="Google Search"**>**
13. **</form>**
15. **</body>**
16. **</html>**

*MySearcher.java*

1. **import** java.io.IOException;
2. **import** javax.servlet.ServletException;
3. **import** javax.servlet.http.HttpServlet;
4. **import** javax.servlet.http.HttpServletRequest;
5. **import** javax.servlet.http.HttpServletResponse;
7. **public** **class** MySearcher **extends** HttpServlet {
8. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
9. **throws** ServletException, IOException {
11. String name=request.getParameter("name");
12. response.sendRedirect("https://www.google.co.in/#q="+name);
13. }
14. }

An **attribute** is an object that is used to share information in a web app. Attribute allows Servlets to share information among themselves. Attributes can be SET and GET from one of the following scopes :

1. request
2. session
3. application



## **Servlet: How to SET an Attribute**

public void setAttribute(String name, Object obj) method is used to SET an Attribute.

**Example demonstrating Setting Attribute**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class First extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

ServletContext sc = getServletContext();

sc.setAttribute("user","Abhijit"); //setting attribute on context scope

}

}

Copy

## **Servlet: How to GET an Attribute**

Object getAttribute(String name) method is used to GET an attribute.

**Example demonstrating getting a value of set Attribute**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Second extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

ServletContext sc = getServletContext();

String str = sc.getAttribute("user"); //getting attribute from context scope

out.println("Welcome"+str); // Prints : Welcome Abhijit

}

}

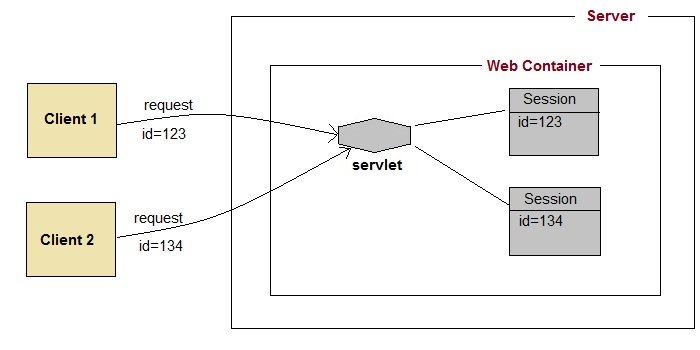
# Managing Session in Servlets

**Session Management** is a mechanism used by the **Web container** to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows:

1. **Cookies**
2. **Hidden form field**
3. **URL Rewriting**
4. **HttpSession**

Session is used to store everything that we can get from the client from all the requests the client makes.

## **How Session Works**

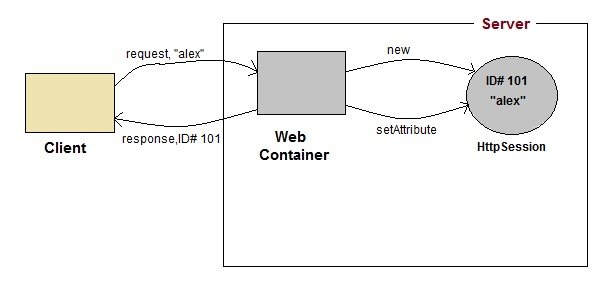


The basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.

# Servlet: What is HttpSession?

**HttpSession** object is used to store entire session with a specific client. We can store, retrieve and remove attribute from **HttpSession** object. Any servlet can have access to **HttpSession** object throughout the getSession() method of the **HttpServletRequest** object.

## **Servlet: How HttpSession works**



1. On client's first request, the **Web Container** generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The **Web Container** uses this ID, finds the matching session with the ID and associates the session with the request.

## **Servlet: HttpSession Interface**



### **Some Important Methods of Servlet HttpSession**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| long getCreationTime() | returns the time when the session was created, measured in milliseconds since midnight January 1, 1970 GMT. |
| String getId() | returns a string containing the unique identifier assigned to the session. |
| long getLastAccessedTime() | returns the last time the client sent a request associated with the session |
| int getMaxInactiveInterval() | returns the maximum time interval, in seconds. |
| void invalidate() | destroy the session |
| boolean isNew() | returns true if the session is new else false |
| void setMaxInactiveInterval(int interval) | Specifies the time, in seconds,after servlet container will invalidate the session. |

### **Complete Example demonstrating usage of HttpSession**

All the files mentioned below are required for the example.

**index.html**

<form method="post" action="Validate">

User: <input type="text" name="user" /><br/>

Password: <input type="text" name="pass" ><br/>

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

</form>

Copy

**web.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>

Copy

**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");

String name = request.getParameter("user");

String pass = request.getParameter("pass");

if(pass.equals("1234"))

{

//creating a session

HttpSession session = request.getSession();

session.setAttribute("user", name);

response.sendRedirect("Welcome");

}

}

}

Copy

**Welcome.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Welcome extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

String user = (String)session.getAttribute("user");

out.println("Hello "+user);

}

}

# Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

### **How Cookie works**

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



### **Types of Cookie**

There are 2 types of cookies in servlets.

26.4M

516

History of Java

1. Non-persistent cookie
2. Persistent cookie

### **Non-persistent cookie**

It is **valid for single session** only. It is removed each time when user closes the browser.

### **Persistent cookie**

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

### **Advantage of Cookies**

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

### **Disadvantage of Cookies**

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

#### **Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.**

### **Cookie class**

**javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

### **Constructor of Cookie class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| Cookie() | constructs a cookie. |
| Cookie(String name, String value) | constructs a cookie with a specified name and value. |

### **Useful Methods of Cookie class**

There are given some commonly used methods of the Cookie class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void setMaxAge(int expiry) | Sets the maximum age of the cookie in seconds. |
| public String getName() | Returns the name of the cookie. The name cannot be changed after creation. |
| public String getValue() | Returns the value of the cookie. |
| public void setName(String name) | changes the name of the cookie. |
| public void setValue(String value) | changes the value of the cookie. |

### **Other methods required for using Cookies**

|  |
| --- |
| For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:   1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object. 2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser. |

### **How to create Cookie?**

Let's see the simple code to create cookie.

1. Cookie ck=**new** Cookie("user","sonoo jaiswal");//creating cookie object
2. response.addCookie(ck);//adding cookie in the response

### **How to delete Cookie?**

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

1. Cookie ck=**new** Cookie("user","");//deleting value of cookie
2. ck.setMaxAge(0);//changing the maximum age to 0 seconds
3. response.addCookie(ck);//adding cookie in the response

### **How to get Cookies?**

Let's see the simple code to get all the cookies.

1. Cookie ck[]=request.getCookies();
2. **for**(**int** i=0;i<ck.length;i++){
3. out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
4. }

### **Simple example of Servlet Cookies**

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.



### **index.html**

1. <form action="servlet1" method="post">
2. Name:<input type="text" name="userName"/><br/>
3. <input type="submit" value="go"/>
4. </form>

### **FirstServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;

6. **public** **class** FirstServlet **extends** HttpServlet {
8. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response){
9. **try**{
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. String n=request.getParameter("userName");
15. out.print("Welcome "+n);
17. Cookie ck=**new** Cookie("uname",n);//creating cookie object
18. response.addCookie(ck);//adding cookie in the response
20. //creating submit button
21. out.print("<form action='servlet2'>");
22. out.print("<input type='submit' value='go'>");
23. out.print("</form>");
25. out.close();
27. }**catch**(Exception e){System.out.println(e);}
28. }
29. }

### **SecondServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
5. **public** **class** SecondServlet **extends** HttpServlet {
7. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response){
8. **try**{
10. response.setContentType("text/html");
11. PrintWriter out = response.getWriter();
13. Cookie ck[]=request.getCookies();
14. out.print("Hello "+ck[0].getValue());
16. out.close();
18. }**catch**(Exception e){System.out.println(e);}
19. }

22. }

### **web.xml**

1. <web-app>
3. <servlet>
4. <servlet-name>s1</servlet-name>
5. <servlet-**class**>FirstServlet</servlet-**class**>
6. </servlet>
8. <servlet-mapping>
9. <servlet-name>s1</servlet-name>
10. <url-pattern>/servlet1</url-pattern>
11. </servlet-mapping>
13. <servlet>
14. <servlet-name>s2</servlet-name>
15. <servlet-**class**>SecondServlet</servlet-**class**>
16. </servlet>
18. <servlet-mapping>
19. <servlet-name>s2</servlet-name>
20. <url-pattern>/servlet2</url-pattern>
21. </servlet-mapping>
23. </web-app>

# Servlet Login and Logout Example using Cookies

A **cookie** is a kind of information that is stored at client side.

In the previous page, we learned a lot about cookie e.g. how to create cookie, how to delete cookie, how to get cookie etc.

Here, we are going to create a login and logout example using servlet cookies.

In this example, we are creating 3 links: login, logout and profile. User can't go to profile page until he/she is logged in. If user is logged out, he need to login again to visit profile.

In this application, we have created following files.

1. index.html
2. link.html
3. login.html
4. LoginServlet.java
5. LogoutServlet.java
6. ProfileServlet.java
7. web.xml

*File: index.html*

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="ISO-8859-1"**>**
5. **<title>**Servlet Login Example**</title>**
6. **</head>**
7. **<body>**
9. **<h1>**Welcome to Login App by Cookie**</h1>**
10. **<a** href="login.html"**>**Login**</a>**|
11. **<a** href="LogoutServlet"**>**Logout**</a>**|
12. **<a** href="ProfileServlet"**>**Profile**</a>**
14. **</body>**
15. **</html>**

*File: link.html*

1. **<a** href="login.html"**>**Login**</a>** |
2. **<a** href="LogoutServlet"**>**Logout**</a>** |
3. **<a** href="ProfileServlet"**>**Profile**</a>**
4. **<hr>**

*File: login.html*

1. **<form** action="LoginServlet" method="post"**>**
2. Name:**<input** type="text" name="name"**><br>**
3. Password:**<input** type="password" name="password"**><br>**
4. **<input** type="submit" value="login"**>**
5. **</form>**

*File: LoginServlet.java*

1. **package** com.javatpoint;
3. **import** java.io.IOException;
4. **import** java.io.PrintWriter;
5. **import** javax.servlet.ServletException;
6. **import** javax.servlet.http.Cookie;
7. **import** javax.servlet.http.HttpServlet;
8. **import** javax.servlet.http.HttpServletRequest;
9. **import** javax.servlet.http.HttpServletResponse;
10. **public** **class** LoginServlet **extends** HttpServlet {
11. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)
12. **throws** ServletException, IOException {
13. response.setContentType("text/html");
14. PrintWriter out=response.getWriter();
16. request.getRequestDispatcher("link.html").include(request, response);
18. String name=request.getParameter("name");
19. String password=request.getParameter("password");
21. **if**(password.equals("admin123")){
22. out.print("You are successfully logged in!");
23. out.print("<br>Welcome, "+name);
25. Cookie ck=**new** Cookie("name",name);
26. response.addCookie(ck);
27. }**else**{
28. out.print("sorry, username or password error!");
29. request.getRequestDispatcher("login.html").include(request, response);
30. }
32. out.close();
33. }
35. }

*File: LogoutServlet.java*

1. **package** com.javatpoint;
3. **import** java.io.IOException;
4. **import** java.io.PrintWriter;
5. **import** javax.servlet.ServletException;
6. **import** javax.servlet.http.Cookie;
7. **import** javax.servlet.http.HttpServlet;
8. **import** javax.servlet.http.HttpServletRequest;
9. **import** javax.servlet.http.HttpServletResponse;
10. **public** **class** LogoutServlet **extends** HttpServlet {
11. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
12. **throws** ServletException, IOException {
13. response.setContentType("text/html");
14. PrintWriter out=response.getWriter();

17. request.getRequestDispatcher("link.html").include(request, response);
19. Cookie ck=**new** Cookie("name","");
20. ck.setMaxAge(0);
21. response.addCookie(ck);
23. out.print("you are successfully logged out!");
24. }
25. }

*File: ProfileServlet.java*

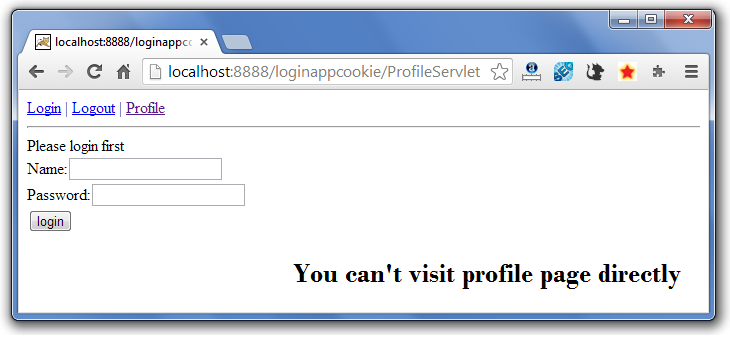
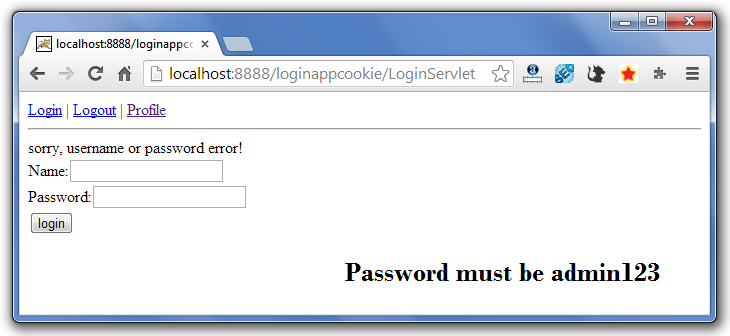
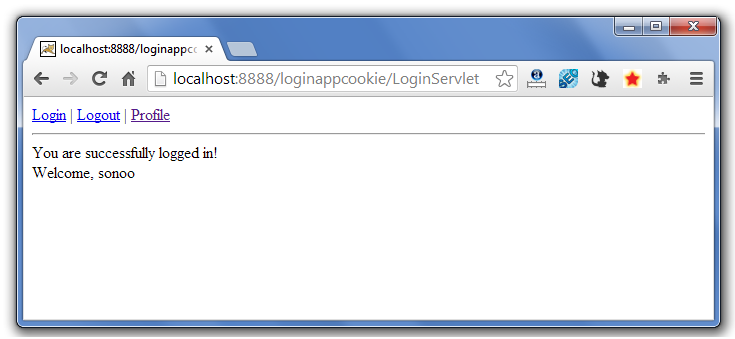
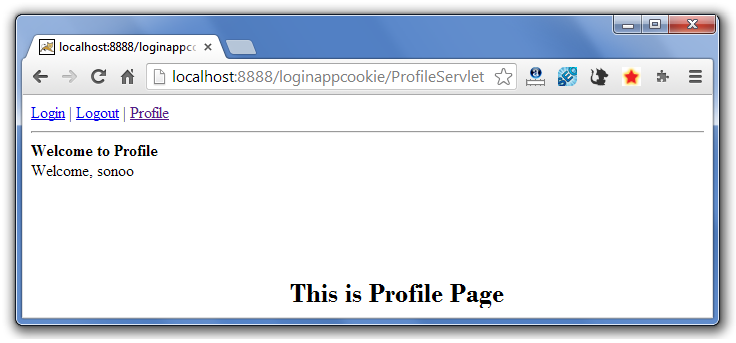
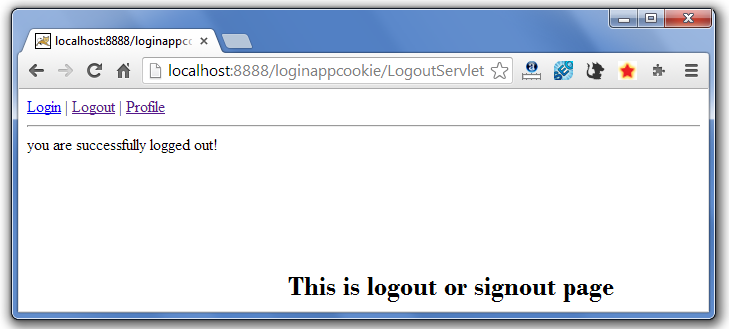
1. **package** com.javatpoint;
3. **import** java.io.IOException;
4. **import** java.io.PrintWriter;
5. **import** javax.servlet.ServletException;
6. **import** javax.servlet.http.Cookie;
7. **import** javax.servlet.http.HttpServlet;
8. **import** javax.servlet.http.HttpServletRequest;
9. **import** javax.servlet.http.HttpServletResponse;
10. **public** **class** ProfileServlet **extends** HttpServlet {
11. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
12. **throws** ServletException, IOException {
13. response.setContentType("text/html");
14. PrintWriter out=response.getWriter();
16. request.getRequestDispatcher("link.html").include(request, response);
18. Cookie ck[]=request.getCookies();
19. **if**(ck!=**null**){
20. String name=ck[0].getValue();
21. **if**(!name.equals("")||name!=**null**){
22. out.print("<b>Welcome to Profile</b>");
23. out.print("<br>Welcome, "+name);
24. }
25. }**else**{
26. out.print("Please login first");
27. request.getRequestDispatcher("login.html").include(request, response);
28. }
29. out.close();
30. }
31. }

*File: web.xml*

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
4. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5"**>**
6. **<servlet>**
7. **<description></description>**
8. **<display-name>**LoginServlet**</display-name>**
9. **<servlet-name>**LoginServlet**</servlet-name>**
10. **<servlet-class>**com.javatpoint.LoginServlet**</servlet-class>**
11. **</servlet>**
12. **<servlet-mapping>**
13. **<servlet-name>**LoginServlet**</servlet-name>**
14. **<url-pattern>**/LoginServlet**</url-pattern>**
15. **</servlet-mapping>**
16. **<servlet>**
17. **<description></description>**
18. **<display-name>**ProfileServlet**</display-name>**
19. **<servlet-name>**ProfileServlet**</servlet-name>**
20. **<servlet-class>**com.javatpoint.ProfileServlet**</servlet-class>**
21. **</servlet>**
22. **<servlet-mapping>**
23. **<servlet-name>**ProfileServlet**</servlet-name>**
24. **<url-pattern>**/ProfileServlet**</url-pattern>**
25. **</servlet-mapping>**
26. **<servlet>**
27. **<description></description>**
28. **<display-name>**LogoutServlet**</display-name>**
29. **<servlet-name>**LogoutServlet**</servlet-name>**
30. **<servlet-class>**com.javatpoint.LogoutServlet**</servlet-class>**
31. **</servlet>**
32. **<servlet-mapping>**
33. **<servlet-name>**LogoutServlet**</servlet-name>**
34. **<url-pattern>**/LogoutServlet**</url-pattern>**
35. **</servlet-mapping>**
36. **</web-app>**

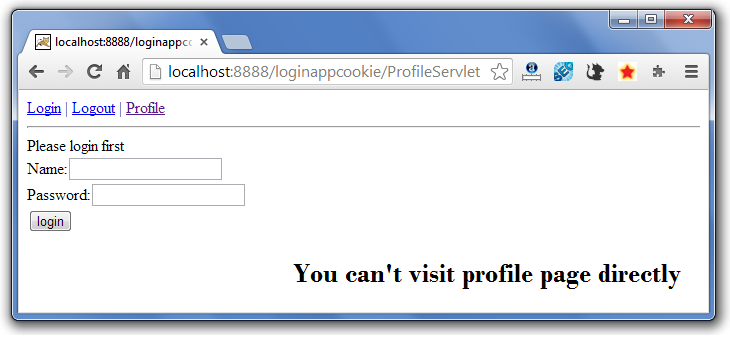
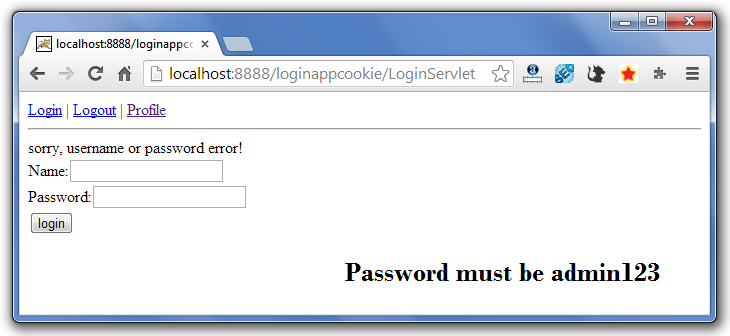
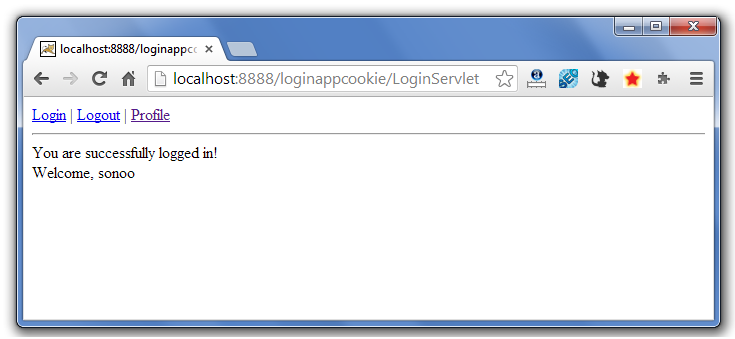
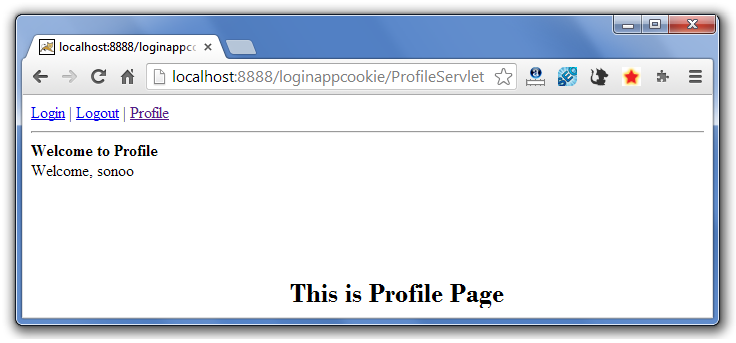
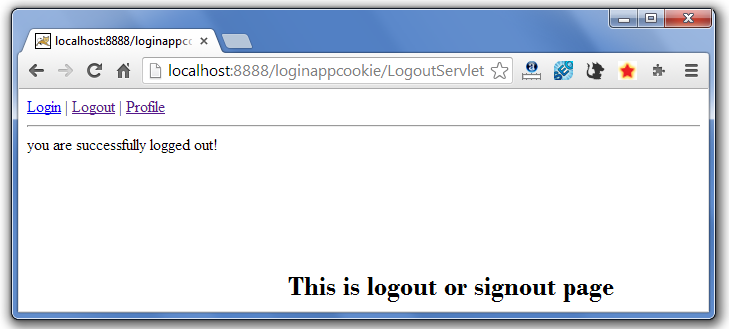
[download this example (developed using Eclipse IDE)](https://static.javatpoint.com/src/servlet/eclipse/loginappcookie.zip)

#### **Output**

#### **Output**



# 3)URL Rewriting

1. [URL Rewriting](https://www.javatpoint.com/url-rewriting-in-session-tracking)
2. [Advantage of URL Rewriting](https://www.javatpoint.com/url-rewriting-in-session-tracking#urladv)
3. [Disadvantage of URL Rewriting](https://www.javatpoint.com/url-rewriting-in-session-tracking#urldisadv)
4. [Example of URL Rewriting](https://www.javatpoint.com/url-rewriting-in-session-tracking#urlex)

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.



### **Advantage of URL Rewriting**

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

### **Disadvantage of URL Rewriting**

1. It will work only with links.
2. It can send Only textual information.

### **Example of using URL Rewriting**

In this example, we are maintaning the state of the user using link. For this purpose, we are appending the name of the user in the query string and getting the value from the query string in another page.

### **index.html**

1. <form action="servlet1">
2. Name:<input type="text" name="userName"/><br/>
3. <input type="submit" value="go"/>
4. </form>

### **FirstServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;

6. **public** **class** FirstServlet **extends** HttpServlet {
8. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response){
9. **try**{
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. String n=request.getParameter("userName");
15. out.print("Welcome "+n);
17. //appending the username in the query string
18. out.print("<a href='servlet2?uname="+n+"'>visit</a>");
20. out.close();
22. }**catch**(Exception e){System.out.println(e);}
23. }
25. }

### **SecondServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
5. **public** **class** SecondServlet **extends** HttpServlet {
7. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
8. **try**{
10. response.setContentType("text/html");
11. PrintWriter out = response.getWriter();
13. //getting value from the query string
14. String n=request.getParameter("uname");
15. out.print("Hello "+n);
17. out.close();
19. }**catch**(Exception e){System.out.println(e);}
20. }

23. }

### **web.xml**

1. <web-app>
3. <servlet>
4. <servlet-name>s1</servlet-name>
5. <servlet-**class**>FirstServlet</servlet-**class**>
6. </servlet>
8. <servlet-mapping>
9. <servlet-name>s1</servlet-name>
10. <url-pattern>/servlet1</url-pattern>
11. </servlet-mapping>
13. <servlet>
14. <servlet-name>s2</servlet-name>
15. <servlet-**class**>SecondServlet</servlet-**class**>
16. </servlet>
18. <servlet-mapping>
19. <servlet-name>s2</servlet-name>
20. <url-pattern>/servlet2</url-pattern>
21. </servlet-mapping>
23. </web-app>

Events and Listeners:

# Event and Listener in Servlet

1. [Event and Listener in Servlet](https://www.javatpoint.com/Event-and-Listener-in-Servlet)
2. [Event classes](https://www.javatpoint.com/Event-and-Listener-in-Servlet#eventclasses)
3. [Event interfaces](https://www.javatpoint.com/Event-and-Listener-in-Servlet#eventinterfaces)

Events are basically occurrence of something. Changing the state of an object is known as an event.

We can perform some important tasks at the occurrence of these exceptions, such as counting total and current logged-in users, creating tables of the database at time of deploying the project, creating database connection object etc.

There are many Event classes and Listener interfaces in the javax.servlet and javax.servlet.http packages.

## **Event classes**

The event classes are as follows:

38.6M

812

Exception Handling in Java - Javatpoint

1. ServletRequestEvent
2. ServletContextEvent
3. ServletRequestAttributeEvent
4. ServletContextAttributeEvent
5. HttpSessionEvent
6. HttpSessionBindingEvent

## **Event interfaces**

The event interfaces are as follows:

1. ServletRequestListener
2. ServletRequestAttributeListener
3. ServletContextListener
4. ServletContextAttributeListener
5. HttpSessionListener
6. HttpSessionAttributeListener
7. HttpSessionBindingListener
8. HttpSessionActivationListener

# ServletContextEvent and ServletContextListener

1. [ServletContextEvent and ServletContextListener](https://www.javatpoint.com/ServletContextEvent)
2. [Constructor of ServletContextEvent class](https://www.javatpoint.com/ServletContextEvent#constructor)
3. [Method of ServletContextEvent class](https://www.javatpoint.com/ServletContextEvent#method)
4. [Methods of ServletContextListener interface](https://www.javatpoint.com/ServletContextEvent#methods)
5. [Example of ServletContextEvent and ServletContextListener](https://www.javatpoint.com/ServletContextEvent#ex1)
6. [Example of ServletContextListener to create table of a project](https://www.javatpoint.com/ServletContextEvent#ex2)

The ServletContextEvent is notified when web application is deployed on the server.

If you want to perform some action at the time of deploying the web application such as creating database connection, creating all the tables of the project etc, you need to implement ServletContextListener interface and provide the implementation of its methods.

### **Constructor of ServletContextEvent class**

There is only one constructor defined in the ServletContextEvent class. The web container creates the instance of ServletContextEvent after the ServletContext instance.

1. ServletContextEvent(ServletContext e)

### **Method of ServletContextEvent class**

There is only one method defined in the ServletContextEvent class:

1. **public ServletContext getServletContext()**: returns the instance of ServletContext.

### **Methods of ServletContextListener interface**

There are two methods declared in the ServletContextListener interface which must be implemented by the servlet programmer to perform some action such as creating database connection etc.

1. **public void contextInitialized(ServletContextEvent e)**: is invoked when application is deployed on the server.
2. **public void contextDestroyed(ServletContextEvent e)**: is invoked when application is undeployed from the server.

### **Example of ServletContextEvent and ServletContextListener**

In this example, we are retrieving the data from the emp32 table. To serve this, we have created the connection object in the listener class and used the connection object in the servlet.

**index.html**

1. <a href="servlet1">fetch records</a>

**MyListener.java**

1. **import** javax.servlet.\*;
2. **import** java.sql.\*;
4. **public** **class** MyListener **implements** ServletContextListener{
5. **public** **void** contextInitialized(ServletContextEvent event) {
6. **try**{
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection(
9. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
11. //storing connection object as an attribute in ServletContext
12. ServletContext ctx=event.getServletContext();
13. ctx.setAttribute("mycon", con);
15. }**catch**(Exception e){e.printStackTrace();}
16. }
18. **public** **void** contextDestroyed(ServletContextEvent arg0) {}
19. }

**MyListener.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
4. **import** java.sql.\*;
6. **public** **class** FetchData **extends** HttpServlet {
8. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
9. **throws** ServletException, IOException {
11. response.setContentType("text/html");
12. PrintWriter out = response.getWriter();
14. **try**{
15. //Retrieving connection object from ServletContext object
16. ServletContext ctx=getServletContext();
17. Connection con=(Connection)ctx.getAttribute("mycon");
19. //retieving data  from emp32 table
20. PreparedStatement ps=con.prepareStatement("select \* from emp32",
21. ResultSet.TYPE\_SCROLL\_SENSITIVE,ResultSet.CONCUR\_UPDATABLE);
23. ResultSet rs=ps.executeQuery();
24. **while**(rs.next()){
25. out.print("<br>"+rs.getString(1)+" "+rs.getString(2));
26. }
28. con.close();
29. }**catch**(Exception e){e.printStackTrace();}
31. out.close();
32. }
33. }

[download this example (developed using Myeclipse IDE)](https://static.javatpoint.com/src/servlet/fetchservletcontextevent.zip)

### **Example of ServletContextListener to create table of a project**

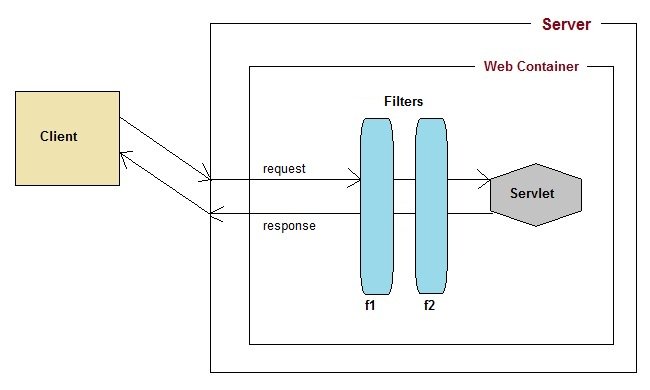
In this example, we are creating table of the project. So we don't need to create all the tables manually in the database.

**MyListener.java**

1. **import** javax.servlet.\*;
2. **import** java.sql.\*;
4. **public** **class** MyListener **implements** ServletContextListener{
6. **public** **void** contextInitialized(ServletContextEvent arg0) {
7. **try**{
8. Class.forName("oracle.jdbc.driver.OracleDriver");
9. Connection con=DriverManager.getConnection("
10. jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
12. String query="create table emp32(id number(10),name varchar2(40))";
13. PreparedStatement ps=con.prepareStatement(query);
14. ps.executeUpdate();
16. System.out.println(query);
18. }**catch**(Exception e){e.printStackTrace();}
19. }
21. **public** **void** contextDestroyed(ServletContextEvent arg0) {
22. System.out.println("project undeployed");
24. }
25. }

# Servlet: Introduction to Filter API

**Filters** are compontents that you can use and configure to perform some filtering tasks. Filter is used for pre-processing of requests and post-processing of responses. You can have any number of filters for pre-processing of a request and post-processing of a response. Filters are configured in the deployment descriptor of a web application.



## **Servlet: How Filters Works?**

* When a request reaches the **Web Container**, it checks if any filter has URL patterns that matches the requested URL.
* The **Web Container** locates the first filter with a matching URL pattern and filter's code is executed.
* If another filter has a matching URL pattern, its code is then executed. This continues until there are no filters with matching URL patterns left.
* If no error occurs, the request passes to the target servlet. Hence we know, that the request will be passed to the target servlet only when all the related Filters are successfully executed.
* The servlet returns the response back to its caller. The last filter that was applied to the request is the first filter applied to the response.
* At last the response will be passed to the **Web Container** which passes it to the client.

## **Servlet: More about Filter API**

**Filter API** is part of **Servlet API**. Filter interface is found in the **javax.servlet** package.

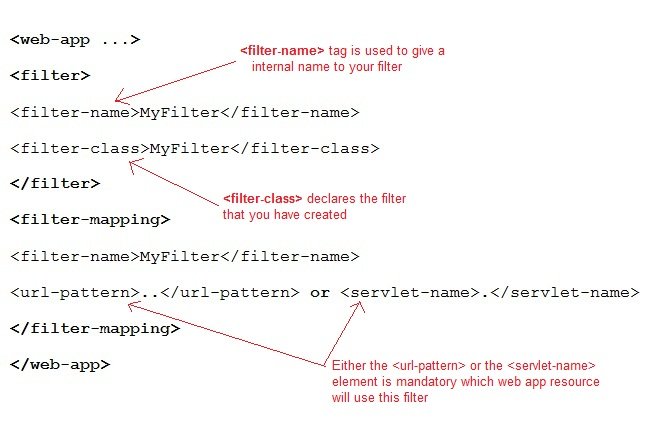
For creating a filter, we must implement Filter interface. Filter interface gives the following life cycle methods for a filter:

1. void init(FilterConfig filterConfig): invoked by the web container to indicate to a filter that it is being placed into service.
2. void **doFilter(ServletRequest request, ServletResponse response, FilterChain chain)**: invoked by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
3. void destroy(): invoked by the web container to indicate to a filter that it is being taken out of service.

### **Servlet: What is FilterChain Interface?**

**FilterChain** object is used to invoke the next filter in the chain, or if the calling filter is the last filter in the chain then the rosource at the end of the chain invoked. The resources at the end of Filter chain can either be a target Servlet(in case of request flow) or the Client(in case of response flow) as described in the diagram above.

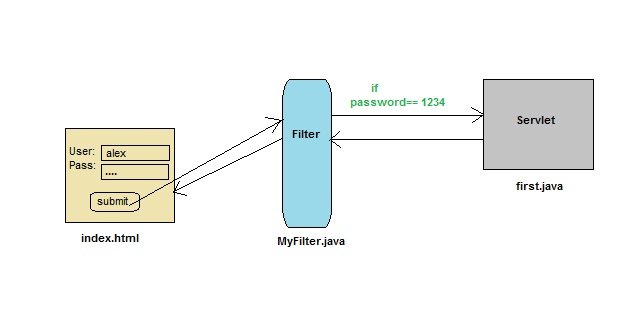
### **Declaring a Servlet Filter inside Deployment Descriptor**



### **Example demonstrating Servlet Filter usage**

In this example we are using Filter to authenticate(check correct username and password). Here **index.html** will ask username and password from the user, **MyFilter** will validate the password entered by the user, if the user has entered "1234" as password, then he will be forwarded to **first** servlet else the index.html will be shown again to the user.

This is exactly what we used to do earlier using two servlet classes earlier, one for validation and the other to Welcome the user. Now we will insert a Filter for validating the user.



**index.html**

<form method="post" action="first">

Name:<input type="text" name="user" /><br/>

Password:<input type="text" name="pass" /><br/>

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

</form>

Copy

**web.xml**

<web-app..>

<filter>

<filter-name>MyFilter</filter-name>

<filter-class>MyFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>MyFilter</filter-name>

<servlet-name>first</servlet-name>

</filter-mapping>

<servlet>

<servlet-name>first</servlet-name>

<servlet-class>first</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>first</servlet-name>

<url-pattern>/first</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

Copy

**MyFilter.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyFilter implements Filter {

public void init(FilterConfig fc) throws ServletException {}

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

PrintWriter out = response.getWriter();

String pass = request.getParameter("pass");

if(pass.equals("1234"))

{

chain.doFilter(request, response);

}

else

{

out.println("You have enter a wrong password");

RequestDispatcher rs = request.getRequestDispatcher("index.html");

rs.include(request, response);

}

}

public void destroy() { }

}

Copy

**first.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class first extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String user = request.getParameter("user");

out.println("Wellcome " + user);

}

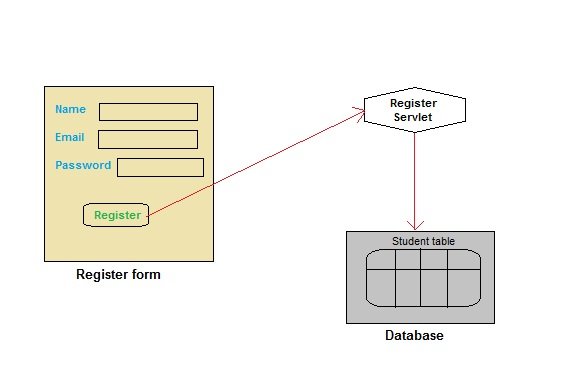
}

Examples on Servlets:

Registration Form:

# Registration form in Servlet

In this example you will see how to develop a registration form in Servlet. To develop a registration form you will need to connect your servlet application with database. Here we are using **MySQL** database.



### **STEP 1: Create a Table in your Database**

create table Student

(

name varchar(60),

email varchar(60),

pass varchar(100)

)

#### **index.html**

<html>

<head>

<title>Register form</title>

</head>

<body>

<form method="post" action="register">

Name:<input type="text" name="name" /><br/>

Email ID:<input type="text" name="email" /><br/>

Password:<input type="text" name="pass" /><br/>

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

</form>

</body>

</html>

Copy

#### **Register.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.sql.\*;

public class Register extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String name = request.getParameter("name");

String email = request.getParameter("email");

String pass = request.getParameter("pass");

try {

// loading drivers for mysql

Class.forName("com.mysql.jdbc.Driver");

//creating connection with the database

Connection con = DriverManager.getConnection

("jdbc:mysql:/ /localhost:3306/test","username","password");

PreparedStatement ps = con.prepareStatement

("insert into Student values(?,?,?)");

ps.setString(1, name);

ps.setString(2, email);

ps.setString(3, pass);

int i = ps.executeUpdate();

if(i > 0) {

out.println("You are sucessfully registered");

}

}

catch(Exception se) {

se.printStackTrace();

}

}

}

Copy

#### **web.xml**

<?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>register</servlet-name>

<servlet-class>Register</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>register</servlet-name>

<url-pattern>/register</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

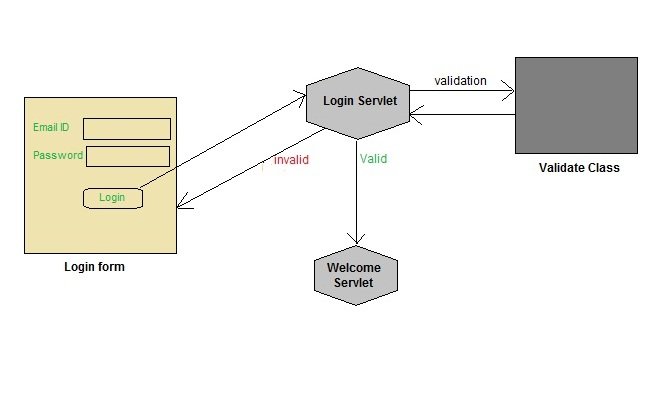
</welcome-file-list>

</web-app>

# Login System in Servlet

n this example we will show you how to develop a login form using servlet. Here we are using **MySql** database. List of file to be created are:

* **index.html**
* **Login.java**
* **Validate.java**
* **Welcome.java**
* **web.xml**



To try this application you will need to create a table in your database and enter some record into it. Refer the previos Lesson for creating table.

#### **index.html**

<html>

<head>

<title>login form</title>

</head>

<body>

<form method="post" action="login">

Email ID:<input type="text" name="email" /><br/>

Password:<input type="text" name="pass" /><br/>

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

</form>

</body>

</html>

Copy

#### **Login.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.sql.\*;

public class Login extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String email = request.getParameter("email");

String pass = request.getParameter("pass");

if(Validate.checkUser(email, pass))

{

RequestDispatcher rs = request.getRequestDispatcher("Welcome");

rs.forward(request, response);

}

else

{

out.println("Username or Password incorrect");

RequestDispatcher rs = request.getRequestDispatcher("index.html");

rs.include(request, response);

}

}

}

Copy

#### **Validate.java**

import java.sql.\*;

public class Validate {

public static boolean checkUser(String email,String pass)

{

boolean st =false;

try {

//loading drivers for mysql

Class.forName("com.mysql.jdbc.Driver");

//creating connection with the database

Connection con = DriverManager.getConnection("jdbc:mysql:/ /localhost:3306/test","root","studytonight");

PreparedStatement ps = con.prepareStatement("select \* from register where email=? and pass=?");

ps.setString(1, email);

ps.setString(2, pass);

ResultSet rs =ps.executeQuery();

st = rs.next();

}

catch(Exception e) {

e.printStackTrace();

}

return st;

}

}

Copy

#### **Welcome.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.sql.\*;

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();

out.println("Welcome user");

}

}

Copy

#### **web.xml**

<?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>login</servlet-name>

<servlet-class>Login</servlet-class>

</servlet>

<servlet>

<servlet-name>Welcome</servlet-name>

<servlet-class>Welcome</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>login</servlet-name>

<url-pattern>/login</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Welcome</servlet-name>

<url-pattern>/Welcome</url-pattern>

</servlet-mapping>

Crud Example:

# CURD Example

CURD stand for Create, Read, Update, Delete. Below is an example of how to perform add, update, delete and view operation in servlet.

**Table:Student**

For creating a this example below is the directory structure of the program:

**student.html**

<form action="SaveServlet1" method="post">

<table>

<tr><td>Id:</td><td><input type="text" name="id1"/></td></tr>

<tr><td>Name:</td><td><input type="text" name="name1"/></td></tr>

<tr><td>Age:</td><td><input type="text" name="age1"/></td></tr>

<tr><td>Course:</td><td><input type="text" name="course1"/></td></tr>

<tr><td>City:</td><td>

<select name="city1" style="width:150px">

<option>Delhi</option>

<option>Noida</option>

<option>Raipur</option>

<option>Bhopal</option>

</select>

</td></tr>

<tr><td colspan="2"><input type="submit" value="Save"/></td></tr>

</table>

</form>

<br/>

<a href="ViewServlet1">View Student</a>

</body>

</html>

Copy

**stu.java**

package com.app.studytonight;

public class stu {

private String id1, name1, age1, course1, city1;

public String getId1() {

return id1;

}

public void setId1(String id1) {

this.id1 = id1;

}

public String getName1() {

return name1;

}

public void setName1(String name1) {

this.name1 = name1;

}

public String getAge1() {

return age1;

}

public void setAge1(String age1) {

this.age1 = age1;

}

public String getCourse1() {

return course1;

}

public void setCourse1(String course1) {

this.course1 = course1;

}

public String getCity1() {

return city1;

}

public void setCity1(String city1) {

this.city1 = city1;

}

}

Copy

**stuDao.java**

package com.app.studytonight;

import java.util.\*;

import java.sql.\*;

public class stuDao {

public static Connection getConnection(){

Connection con1=null;

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

con1=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

}

catch(Exception e1)

{

System.out.println(e1);

}

return con1;

}

public static int save(stu e1){

int status=0;

try{

Connection con1=stuDao.getConnection();

PreparedStatement ps1=con1.prepareStatement(

"insert into student(id,name,age,course,city) values (?,?,?,?,?)");

ps1.setString(1,e1.getId1());

ps1.setString(2,e1.getName1());

ps1.setString(3,e1.getAge1());

ps1.setString(4,e1.getCourse1());

ps1.setString(5,e1.getCity1());

status=ps1.executeUpdate();

con1.close();

}

catch(Exception ex1)

{

ex1.printStackTrace();

}

return status;

}

public static int update1(stu e1){

int status=0;

try{

Connection con1=stuDao.getConnection();

PreparedStatement ps1=con1.prepareStatement(

"update student set id=?, name=?,age=?,course=?,city=? where id=?");

ps1.setString(1,e1.getId1());

ps1.setString(2,e1.getName1());

ps1.setString(3,e1.getAge1());

ps1.setString(4,e1.getCourse1());

ps1.setString(5,e1.getCity1());

status=ps1.executeUpdate();

con1.close();

}

catch(Exception ex1)

{ex1.printStackTrace();}

return status;

}

public static int delete1(int id1){

int status=0;

try{

Connection con1=stuDao.getConnection();

PreparedStatement ps1=con1.prepareStatement("delete from student where id=?");

ps1.setInt(1,id1);

status=ps1.executeUpdate();

con1.close();

}catch(Exception e1){e1.printStackTrace();}

return status;

}

public static stu getStudentById(int id1){

stu e1=new stu();

try{

Connection con1=stuDao.getConnection();

PreparedStatement ps1=con1.prepareStatement("select \* from student where id=?");

ps1.setInt(1,id1);

ResultSet rs1=ps1.executeQuery();

if(rs1.next()){

e1.setId1(rs1.getString(1));

e1.setName1(rs1.getString(2));

e1.setAge1(rs1.getString(3));

e1.setCourse1(rs1.getString(4));

e1.setCity1(rs1.getString(5));

}

con1.close();

}catch(Exception ex1){ex1.printStackTrace();}

return e1;

}

public static List<stu> getAllStudent(){

List<stu> list=new ArrayList<stu>();

try{

Connection con1=stuDao.getConnection();

PreparedStatement ps1=con1.prepareStatement("select \* from student");

ResultSet rs1=ps1.executeQuery();

while(rs1.next()){

stu e1=new stu();

e1.setId1(rs1.getString(1));

e1.setName1(rs1.getString(2));

e1.setAge1(rs1.getString(3));

e1.setCourse1(rs1.getString(4));

e1.setCity1(rs1.getString(5));

list.add(e1);

}

con1.close();

}catch(Exception e1){e1.printStackTrace();}

return list;

}

}

Copy

**SaveServlet.java**

package com.app.studytonight;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/SaveServlet1")

public class SaveServlet1 extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String id1=request.getParameter("id");

String name1=request.getParameter("name");

String age1=request.getParameter("age");

String course1=request.getParameter("course");

String city1=request.getParameter("city");

stu e1=new stu();

e1.setId1(id1);

e1.setName1(name1);

e1.setAge1(age1);

e1.setCourse1(course1);

e1.setCity1(city1);

int status=stuDao.save(e1);

if(status>0){

out.print("<p>Record saved successfully!</p>");

request.getRequestDispatcher("student.html").include(request, response);

}else{

out.println("Sorry! unable to save record");

}

out.close();

}

}

Copy

**EditServlet.java**

package com.app.studytonight;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/EditServlet")

public class EditServlet1 extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

out.println("<h1>Update Student</h1>");

String sid=request.getParameter("id");

int id1=Integer.parseInt(sid);

stu e1=stuDao.getStudentById(id1);

out.print("<form action='EditServlet2' method='post'>");

out.print("<table>");

out.print("<tr><td></td><td><input type='hidden' name='id' value='"+e1.getId1()+"'/></td></tr>");

out.print("<tr><td>Name:</td><td><input type='text' name='name' value='"+e1.getName1()+"'/></td></tr>");

out.print("<tr><td>Age:</td><td><input type='age' name='age' value='"+e1.getAge1()+"'/></td></tr>");

out.print("<tr><td>Course:</td><td><input type='course' name='course' value='"+e1.getCourse1()+"'/></td></tr>");

out.print("<tr><td>City:</td><td>");

out.print("<select name='city' style='width:150px'>");

out.print("<option>Delhi</option>");

out.print("<option>Noida</option>");

out.print("<option>Raipur</option>");

out.print("<option>Bhopal</option>");

out.print("</select>");

out.print("</td></tr>");

out.print("<tr><td colspan='2'><input type='submit' value='Edit & Save '/></td></tr>");

out.print("</table>");

out.print("</form>");

out.close();

}

}

Copy

**EditServlet2.java**

package com.app.studytonight;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/EditServlet2")

public class EditServlet2 extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String sid=request.getParameter("id");

int id1=Integer.parseInt(sid);

String name1=request.getParameter("name");

String age1=request.getParameter("age");

String course1=request.getParameter("course");

String city1=request.getParameter("city");

stu e1=new stu();

e1.setId1(id1);

e1.setName1(name1);

e1.setAge1(age1);

e1.setCourse1(course1);

e1.setCity1(city1);

int status=stuDao.update(e1);

if(status>0){

response.sendRedirect("ViewServlet");

}else{

out.println("Sorry! unable to update record");

}

out.close();

}

}

Copy

**DeleteServlet.java**

package com.app.studytonight;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/DeleteServlet")

public class DeleteServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String sid=request.getParameter("id");

int id1=Integer.parseInt(sid);

stuDao.delete(id1);

response.sendRedirect("ViewServlet");

}

}

Copy

**ViewServlet.java**

package com.app.studytonight;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/ViewServlet")

public class ViewServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

out.println("<a href='student.html'>Add New Student</a>");

out.println("<h1>Student List</h1>");

# Pagination Example

When we have large numbers of records which can note be displayed on a single page. Then we use the concept of Pagination. It helps use to display limited number of records in a single page. Below is an example of pagination.

### **For creating this example below is the directory structure of the program:**

**pStud.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<a href="PageViewServlet?page=1">View Student</a>

</body>

</html>

Copy

**PageViewServlet.java**

package com.app.studytonight;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/PageViewServlet")

public class PageViewServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request1, HttpServletResponse response1) throws ServletException, IOException

{

response1.setContentType("text/html");

PrintWriter out=response1.getWriter();

String spageid1=request1.getParameter("page");

int pageid1=Integer.parseInt(spageid1);

int total1=5;

if(pageid1==1){}

else{

pageid1=pageid1-1;

pageid1=pageid1\*total1+1;

}

List<pStudGetSet> list=StudDao.getRecords(pageid1,total1);

out.print("<h1>studytonight Page No: "+spageid1+"</h1>");

out.print("<table border='1' cellpadding='4' width='60%'>");

out.print("<tr><th>Id</th><th>Name</th><th>Age</th><th>City</th></tr>");

for(pStudGetSet e1:list){

out.print("<tr><td>"+e1.getId1()+"</td><td>"+e1.getName1()+"</td><td>"+e1.getAge1()+"</td></td>"+e1.getCity1()+"</td></tr>");

}

out.print("</table>");

out.print("<a href='PageViewServlet?page=1'>1</a> ");

out.print("<a href='PageViewServlet?page=2'>2</a> ");

out.print("<a href='PageViewServlet?page=3'>3</a> ");

out.close();

}

}

Copy

**pStudGetSet.java**

package com.app.studytonight;

public class pStudGetSet {

private int id1;

private String name1, age1, city1;

public int getId1() {

return id1;

}

public void setId1(int id1) {

this.id1 = id1;

}

public String getName1() {

return name1;

}

public void setName1(String name1) {

this.name1 = name1;

}

public String getAge1() {

return age1;

}

public void setAge1(String age1) {

this.age1 = age1;

}

public String getCity1() {

return city1;

}

public void setCity1(String city1) {

this.city1 = city1;

}

Copy

**StudDao.java**

package com.app.studytonight;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class StudDao {

public static Connection getConnection(){

Connection con1=null;

try{

Class.forName("com.mysql.jdbc.Driver");

con1=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","System","oracle");

}

catch(Exception e)

{

System.out.println(e);

}

return con1;

}

public static List<pStudGetSet> getRecords(int start1,int total1){

List<pStudGetSet> list=new ArrayList<pStudGetSet>();

try{

Connection con1=getConnection();

PreparedStatement ps1=con1.prepareStatement("select \* from student limit "+(start1-1)+","+total1);

ResultSet rs1=ps1.executeQuery();

while(rs1.next()){

pStudGetSet e1=new pStudGetSet();

e1.setId1(rs1.getInt(1));

e1.setName1(rs1.getString(2));

e1.setAge1(rs1.getString(3));

e1.setCity1(rs1.getString(4));

list.add(e1);

}

con1.close();

}catch(Exception e){System.out.println(e);}

return list;

}

}