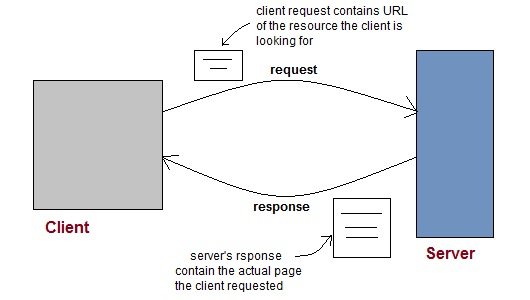
# Servlet: Introduction to Web

Web consists of billions of clients and server connected through wires and wireless networks. The web clients make requests to web server. The web server receives the request, finds the resources and return the response to the client. When a server answers a request, it usually sends some type of content to the client. The client uses web browser to send request to the server. The server often sends response to the browser with a set of instructions written in HTML(HyperText Markup Language). All browsers know how to display HTML page to the client.

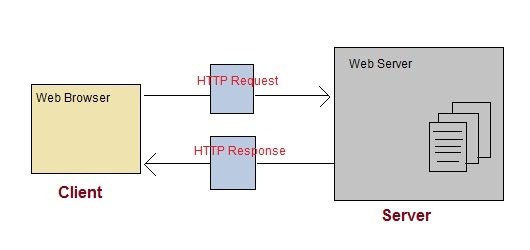


Web Application

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

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. |

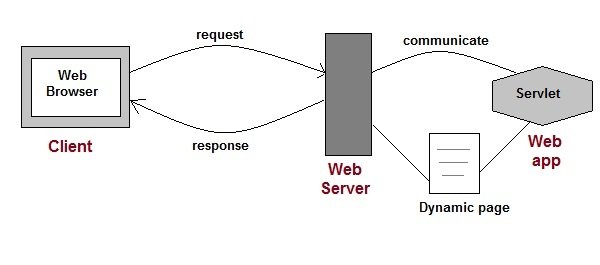
## Difference between GET and POST requests

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

# Introduction to Servlet

**Servlet** Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.

Web applications are helper applications that resides at web server and build dynamic web pages. A dynamic page could be anything like a page that randomly chooses picture to display or even a page that displays the current time.



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

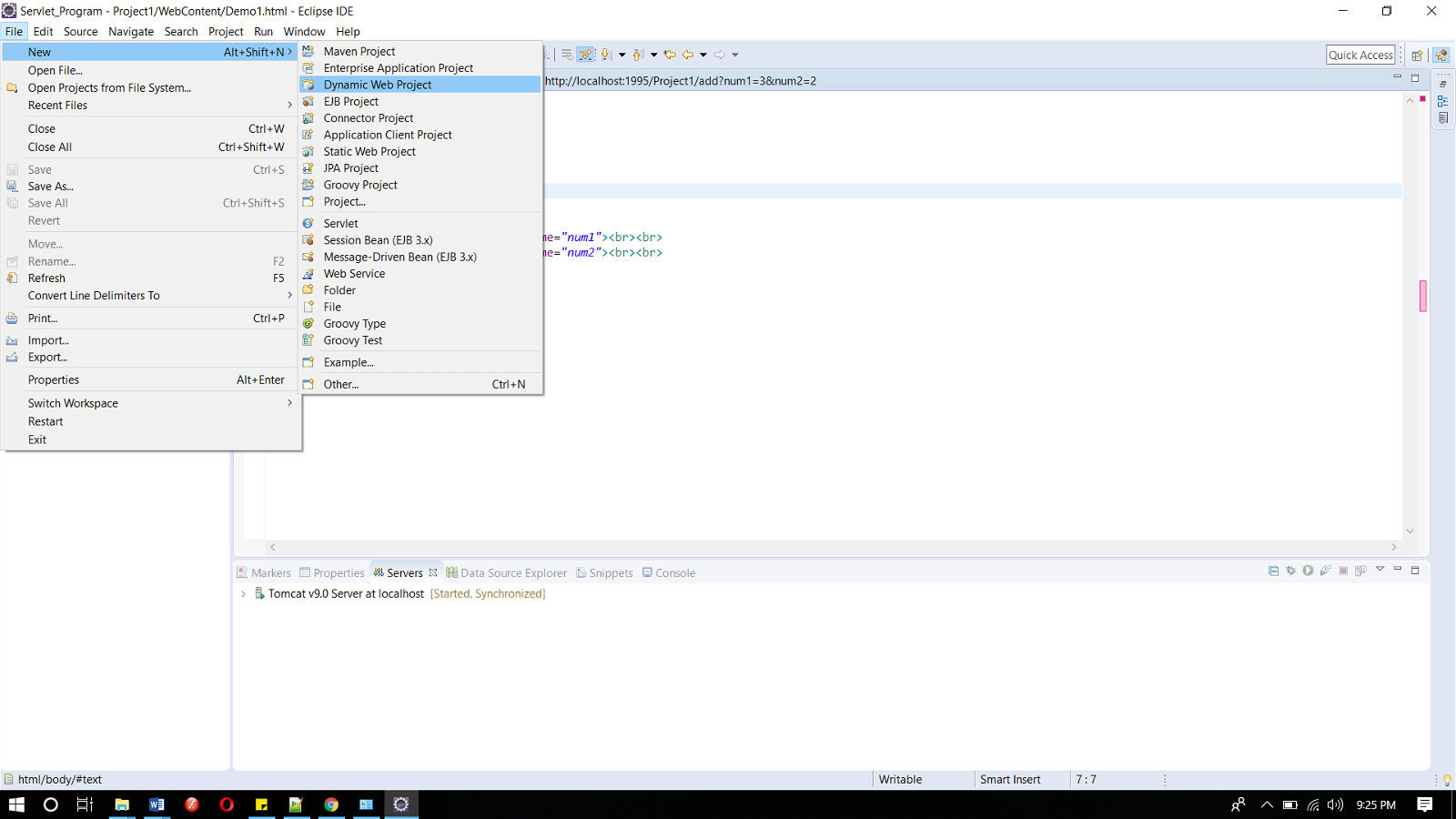
* **javax.servlet**
* **javax.servlet.http**

### Example of Servlet interface on Eclipse

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

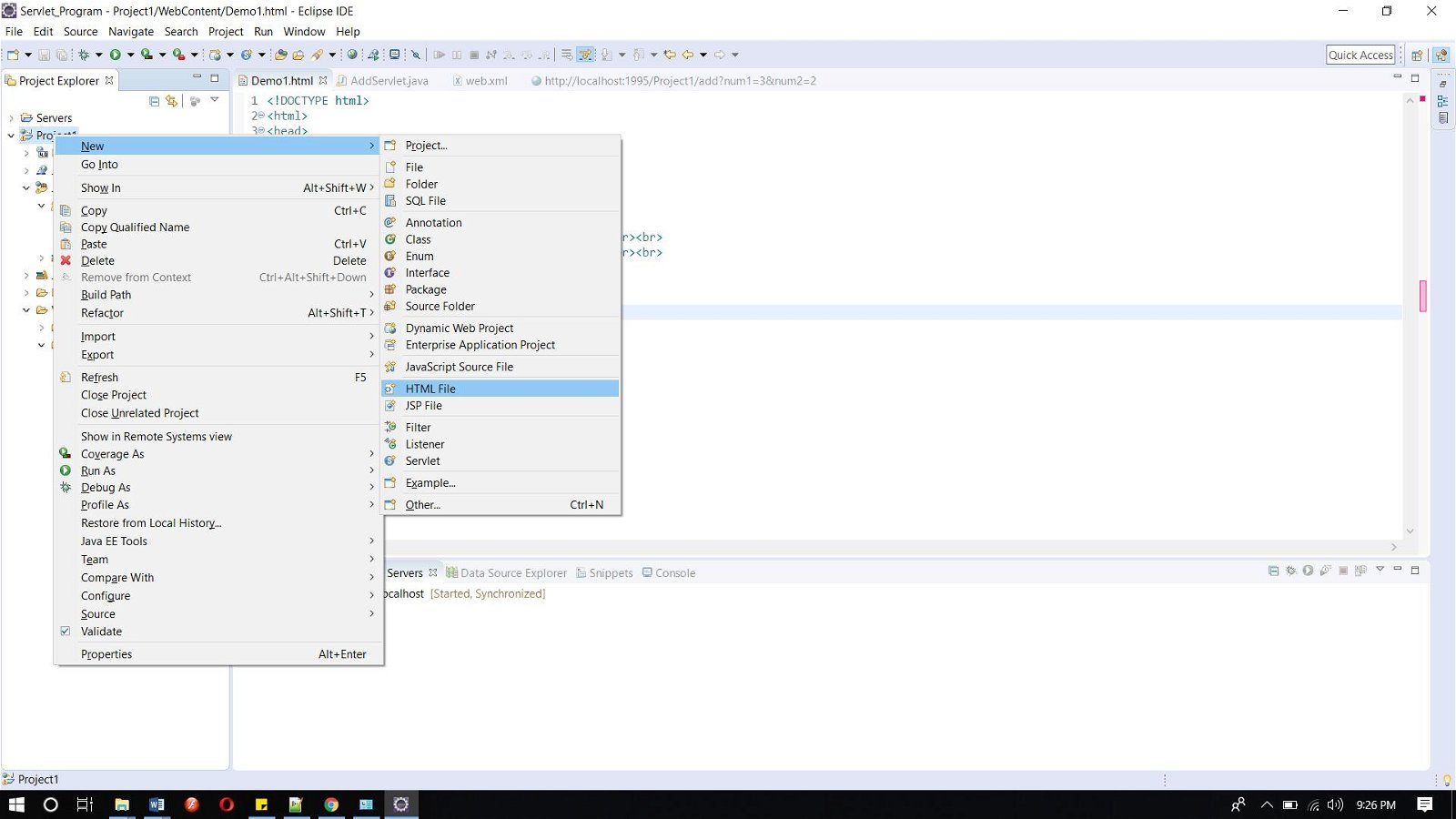
Following are the steps for creating the program.

**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>

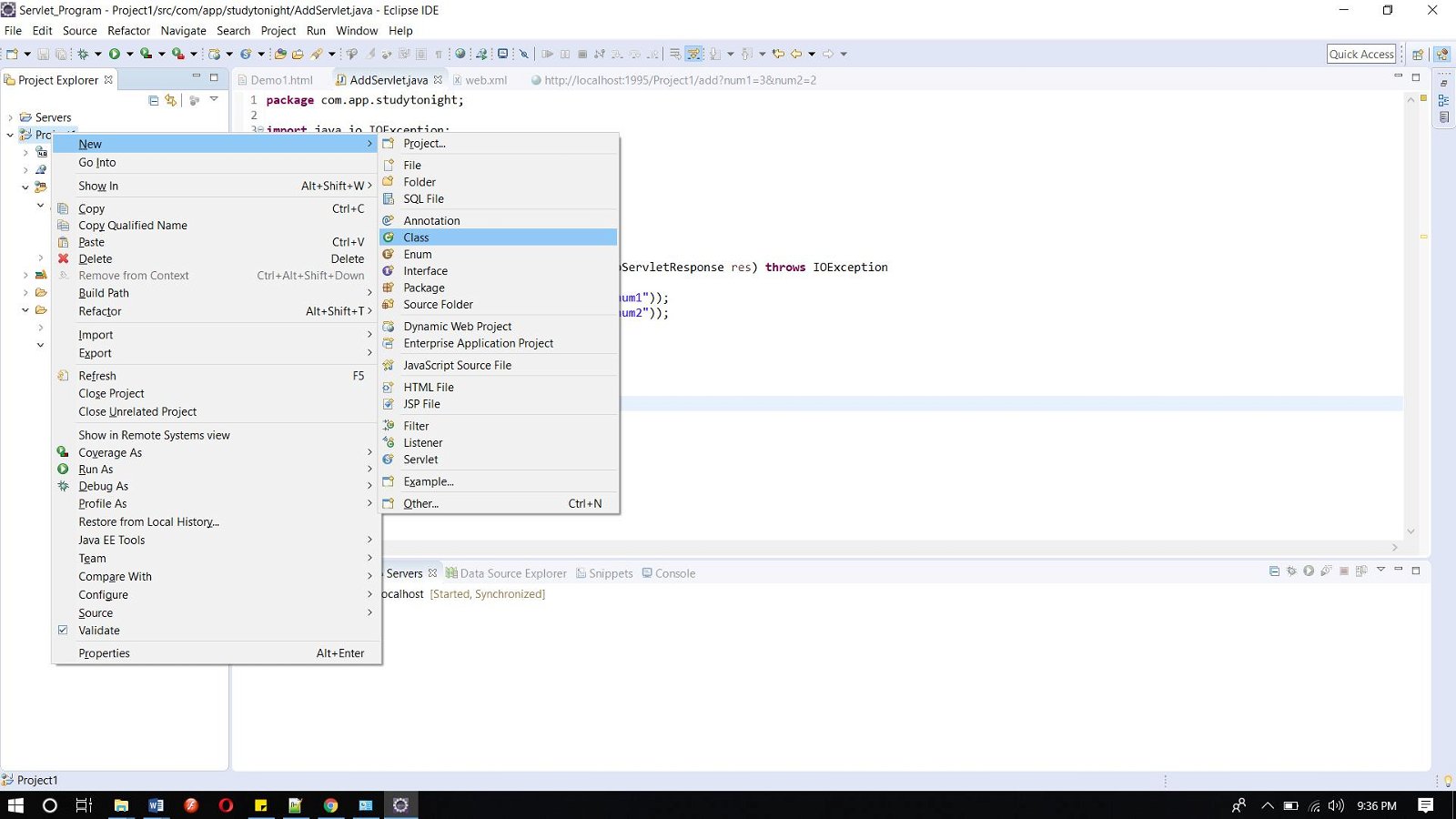
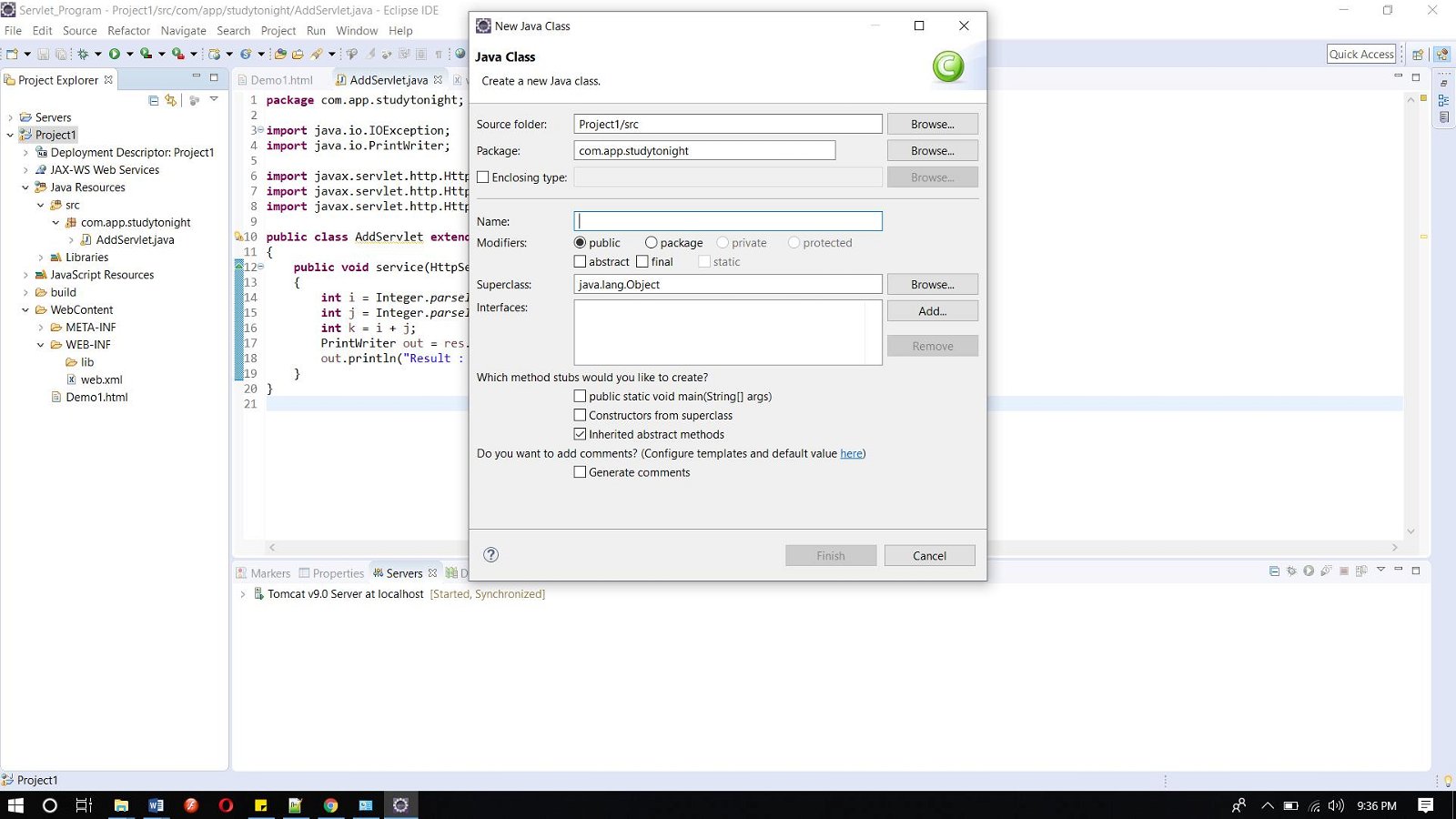
<url-pattern>/demo</url-pattern>

</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";

}

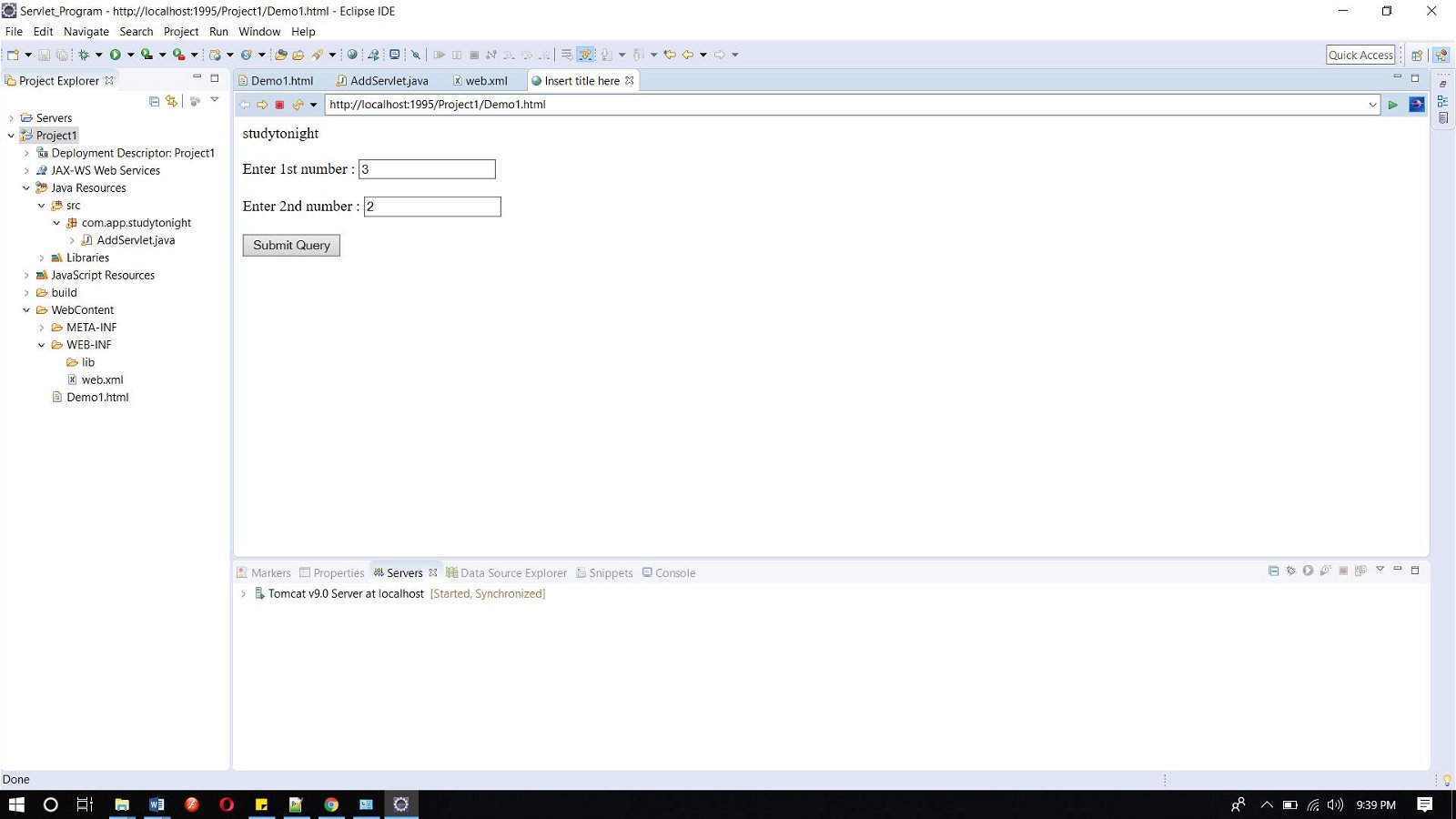
}

Copy

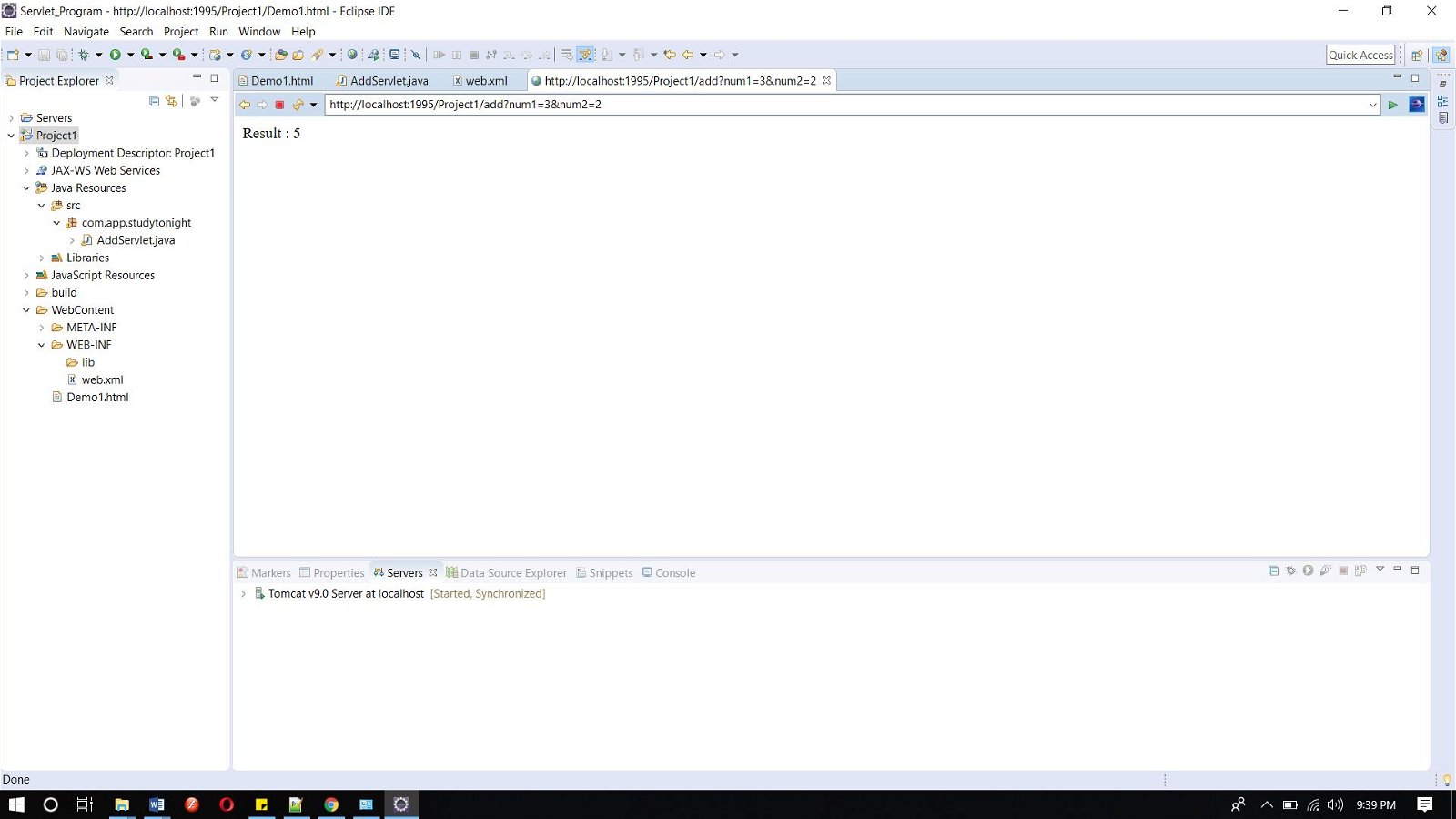
#### Now, Run the code.

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.

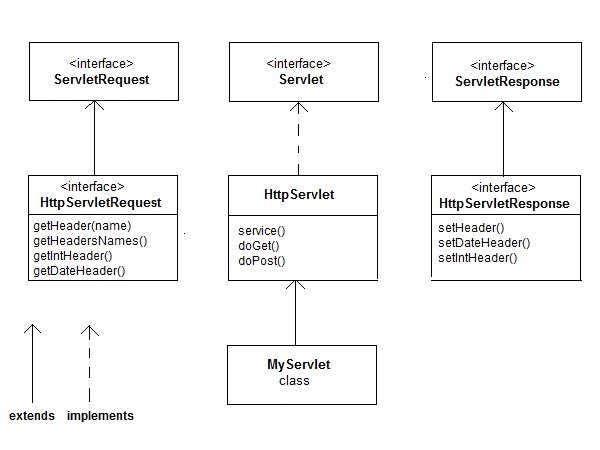


## HttpServlet class

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.

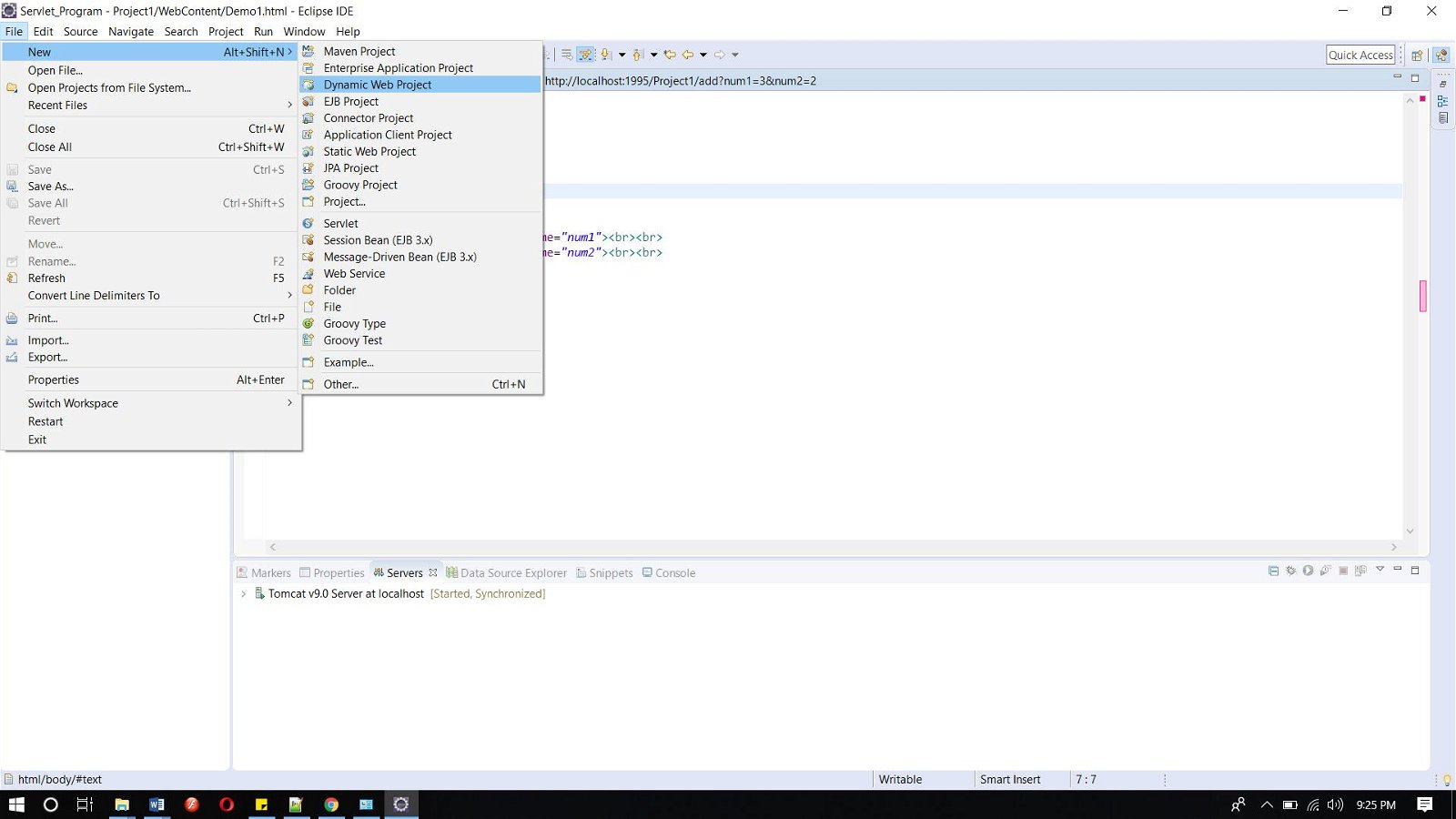


### Example of HttpServlet class on Eclipse

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

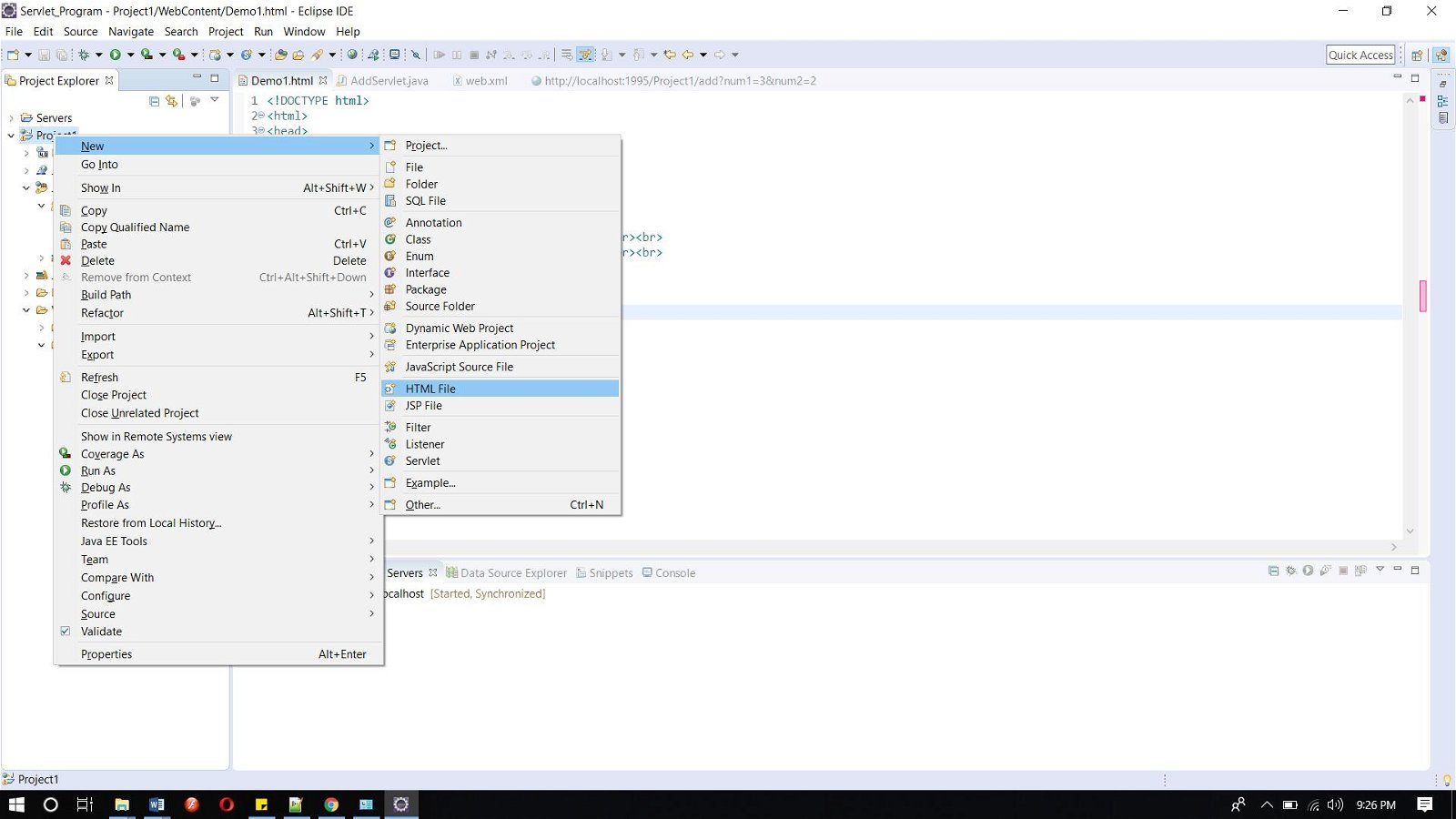
Following are the steps for creating the program.

**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>

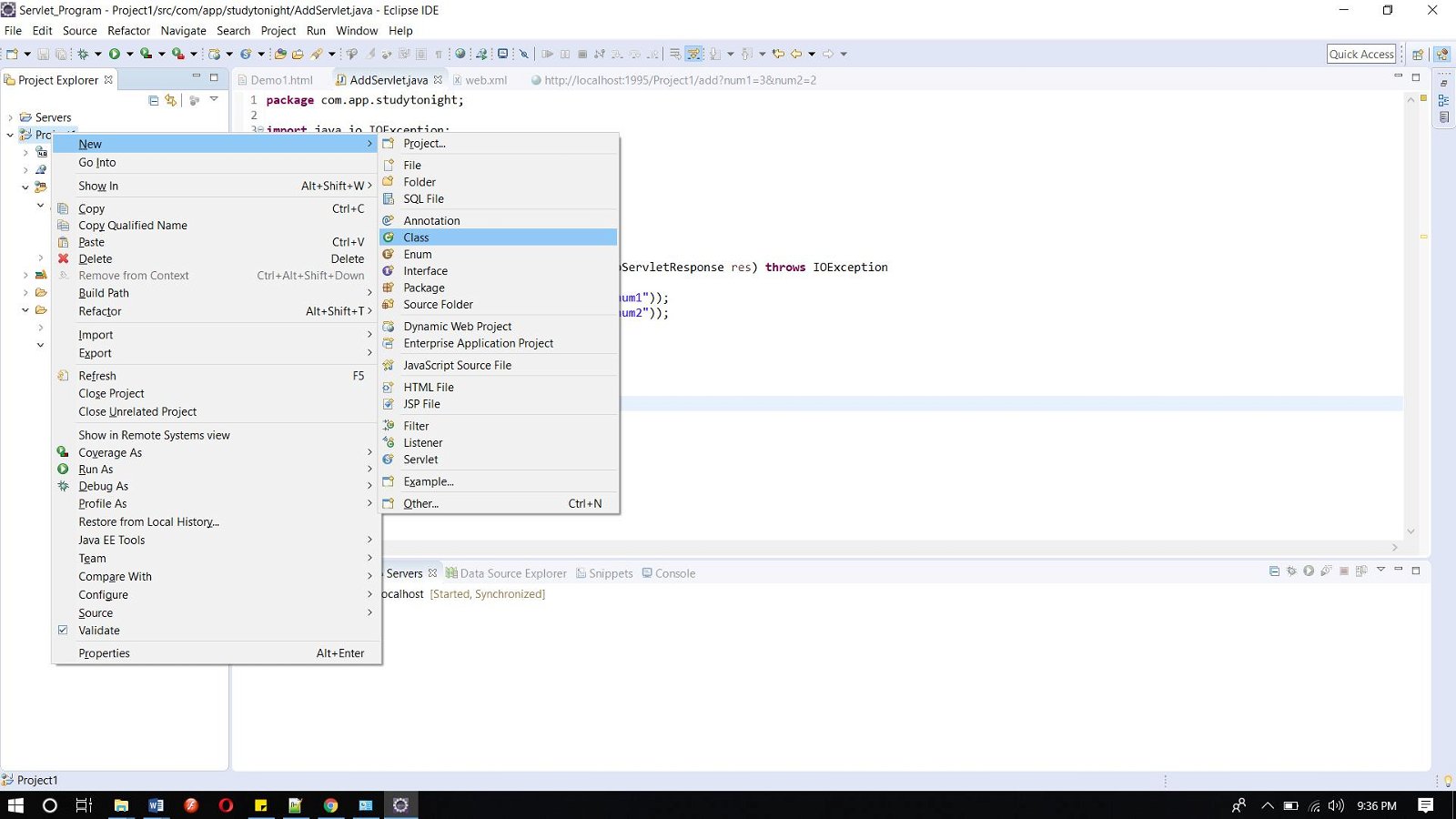
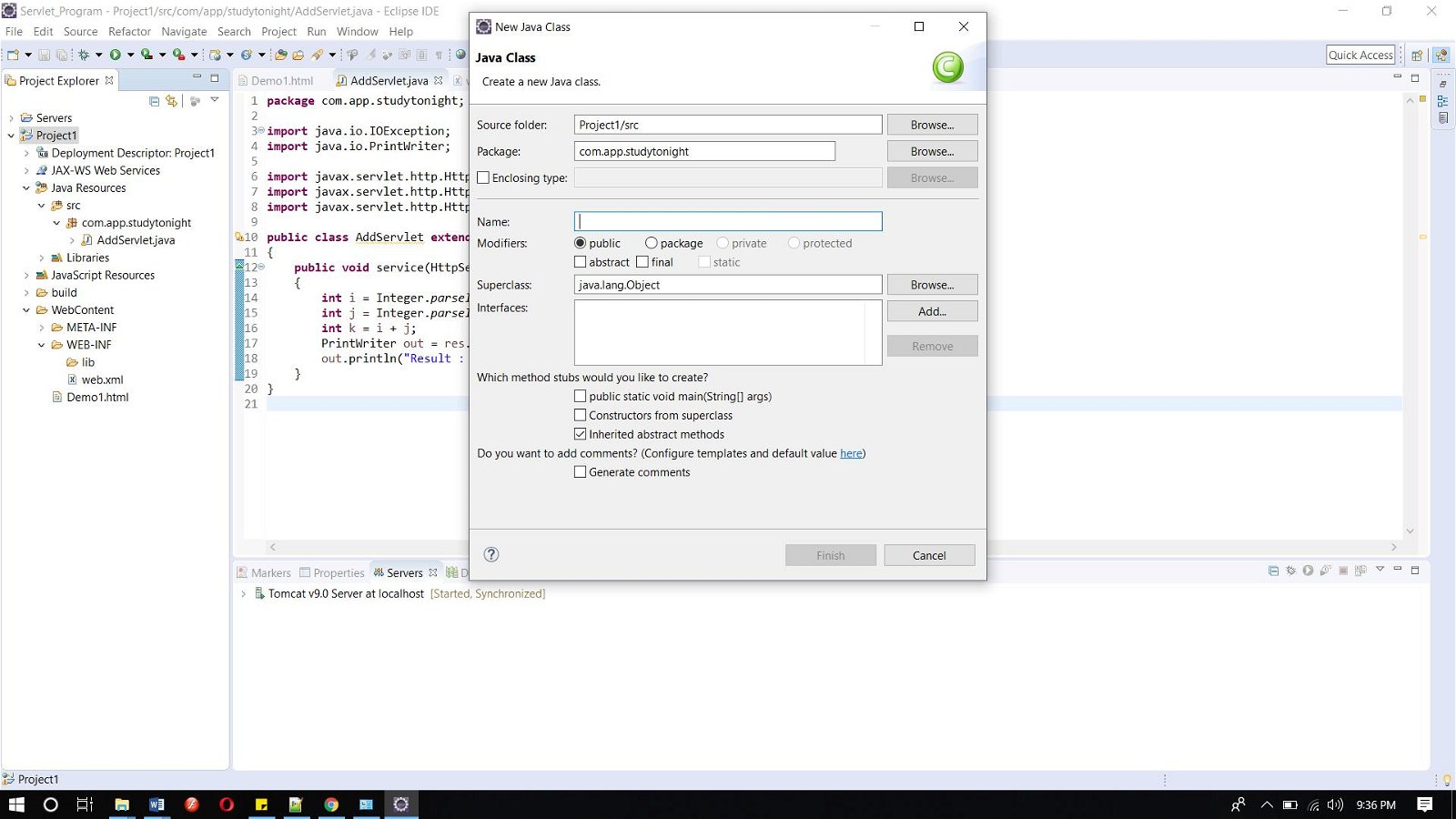
<url-pattern>/mar</url-pattern>

</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 dopost(HttpServletRequest req, HttpServletResponse 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);

}

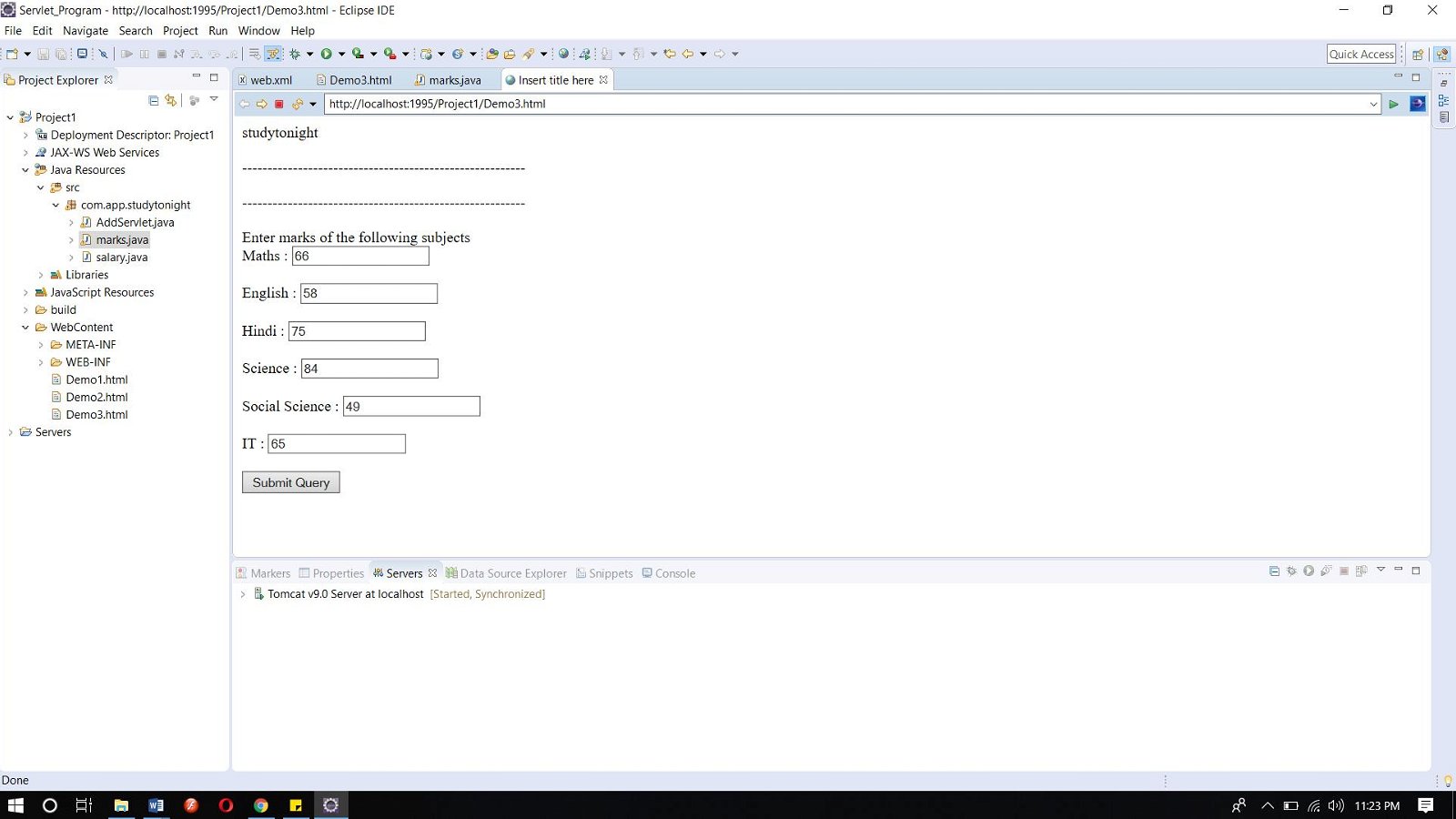
}

Copy

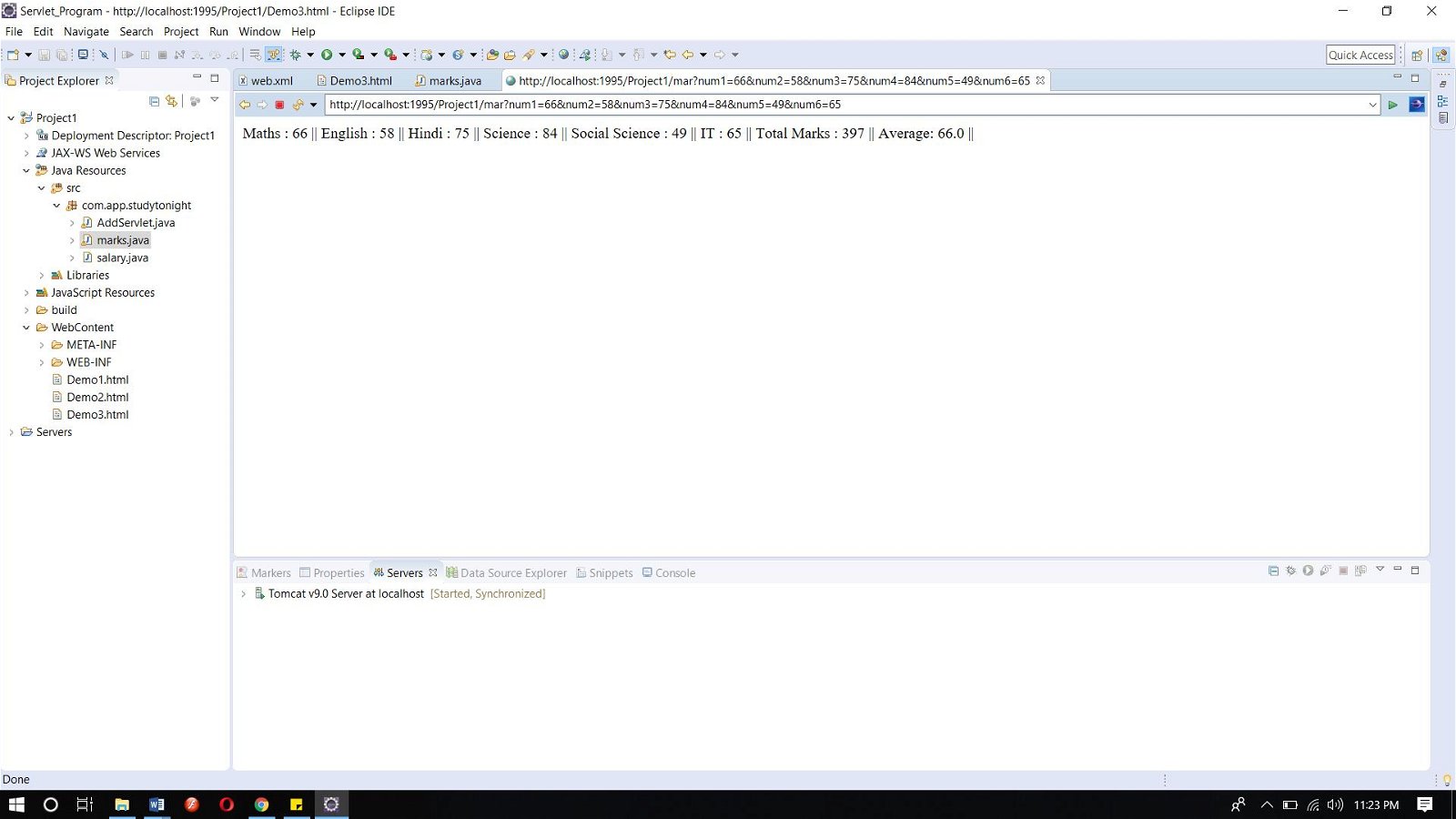
#### Now, Run the code.

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.



# How a Servlet Application works

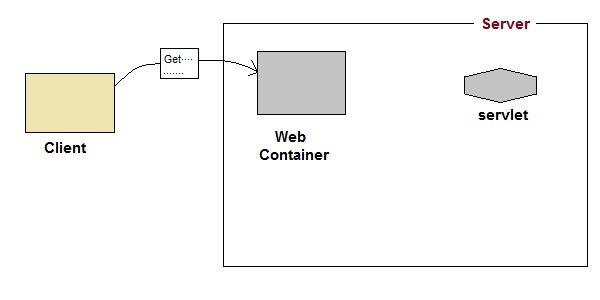
**Web container** is responsible for managing execution of servlets and [JSP pages](https://www.studytonight.com/jsp/introduction-to-jsp.php) for Java EE application.

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

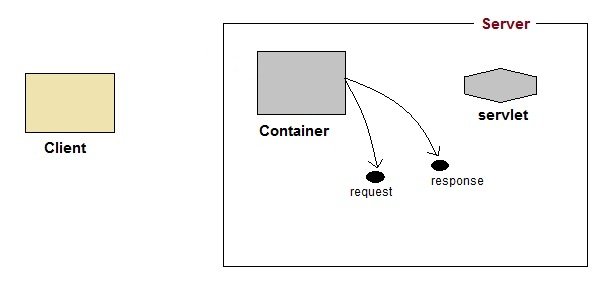
**Servlets don't have a main() method**. Web Container manages the life cycle of a Servlet instance.

## Quick Revision on How a Servlet works

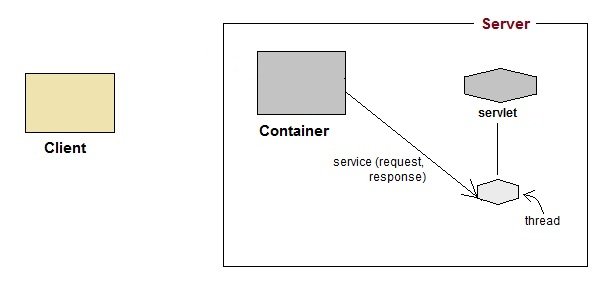
1. User sends request for a servlet by clicking a link that has URL to a servlet.



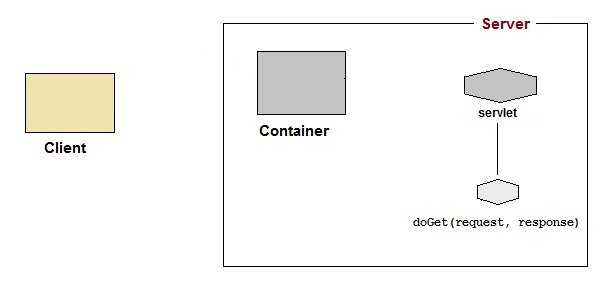
1. The container finds the servlet using **deployment descriptor** and creates two objects :
   1. [**HttpServletRequest**](https://www.studytonight.com/servlet/servlet-request.php)
   2. [**HttpServletResponse**](https://www.studytonight.com/servlet/servlet-response.php)



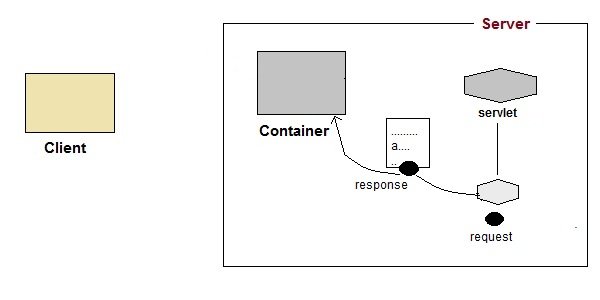
1. Then the container creates or allocates a thread for that request and calls the Servlet's service() method and passes the **request, response** objects as arguments.



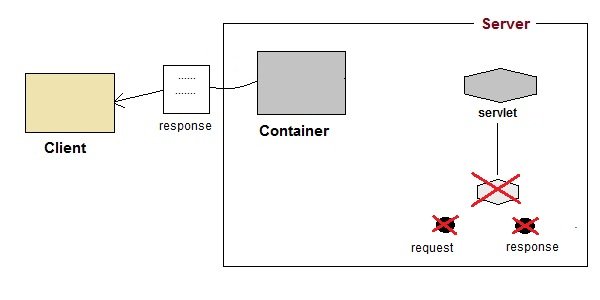
1. The service() method, then decides which servlet method, doGet() or doPost() to call, based on **HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the service() will call Servlet's doGet() method.



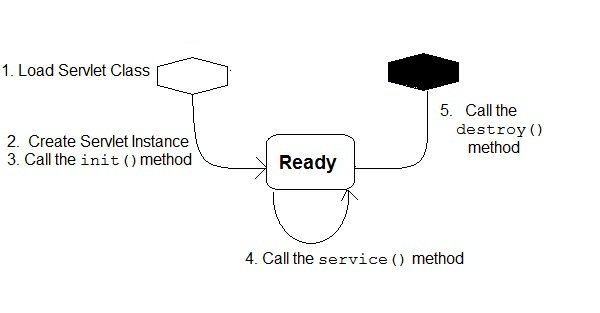
1. Then the Servlet uses response object to write the response back to the client.



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



Servlet Life Cycle



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.

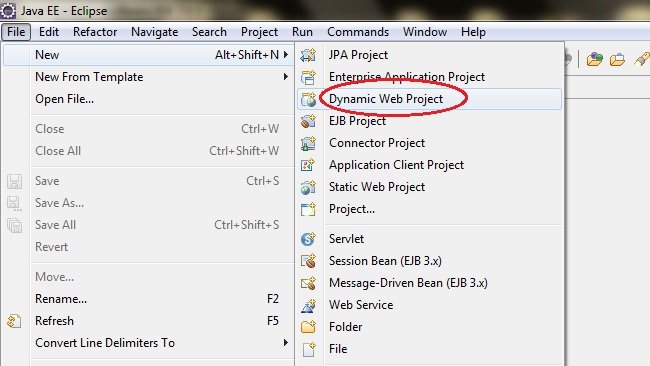
Creating First Servlet Application using Eclipse IDE

**Eclipse** IDE is the most popular Java IDE used in the Industry. It is developed by an open source community and can be downloaded for free from [Eclipse.org](http://www.eclipse.org/)

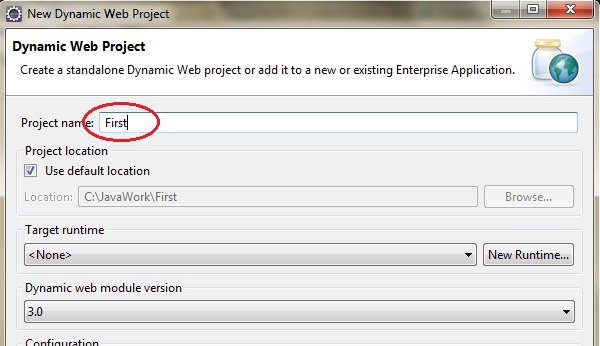
Steps to create Servlet using Eclipse IDE

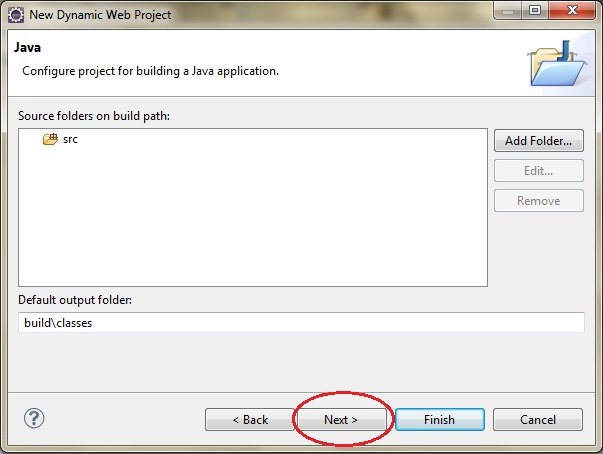
To create a Servlet application in Eclipse IDE you will need to follow the following steps:

1. Goto **File** -> **New** -> **Dynamic Web Project**

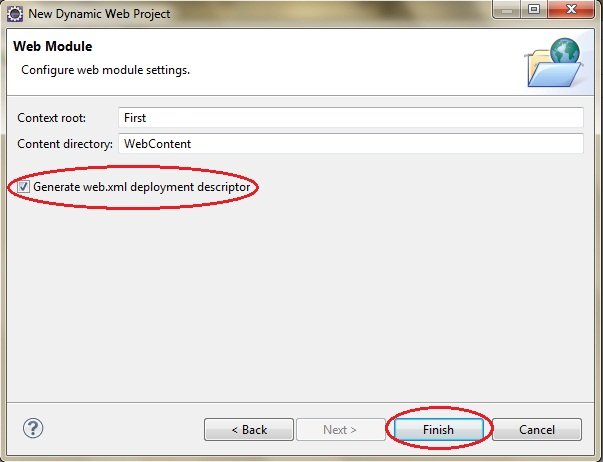


1. Give a Name to your Project and click **Next**

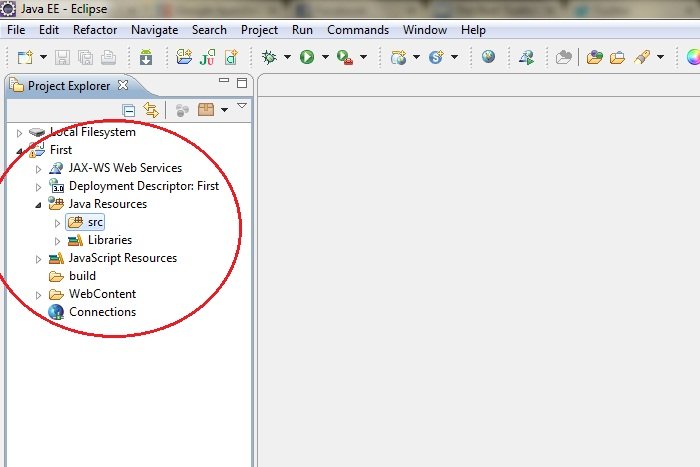




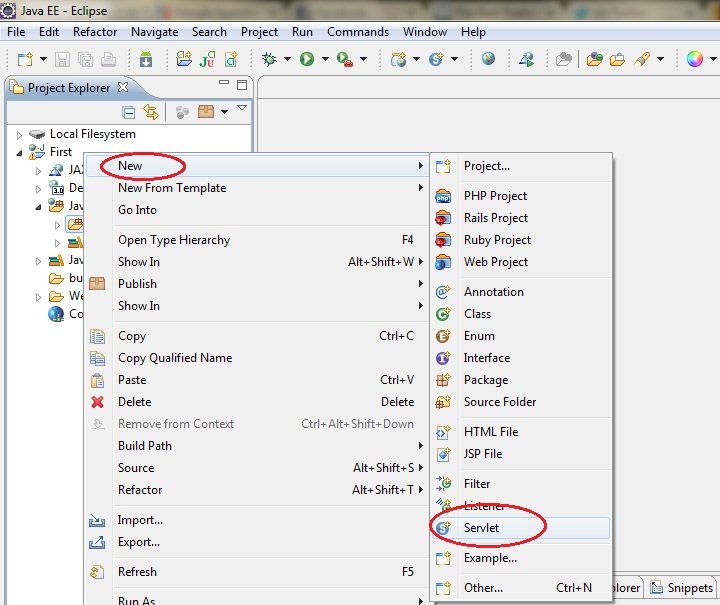
1. Check **Generate web.xml Deployment Descriptor** and click **Finish**



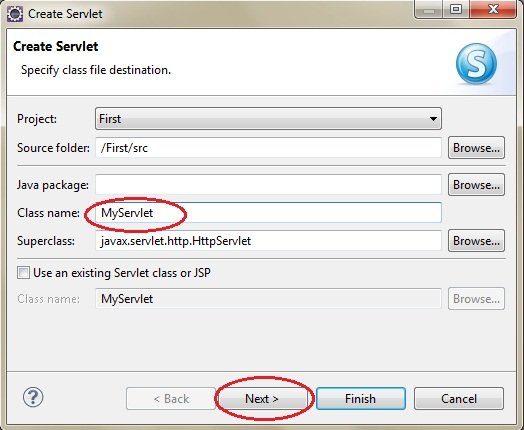
1. Now, the complete directory structure of your Project will be automatically created by Eclipse IDE.



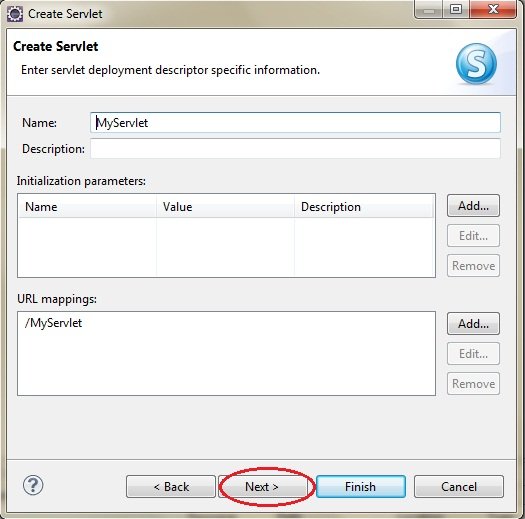
1. Click on **First** project, go to **Java Resources** -> **src**. Right click on **src** select **New** -> **Servlet**



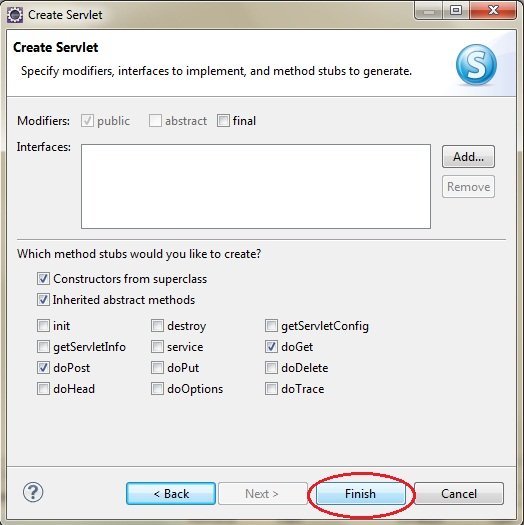
1. Give Servlet class name and click **Next**



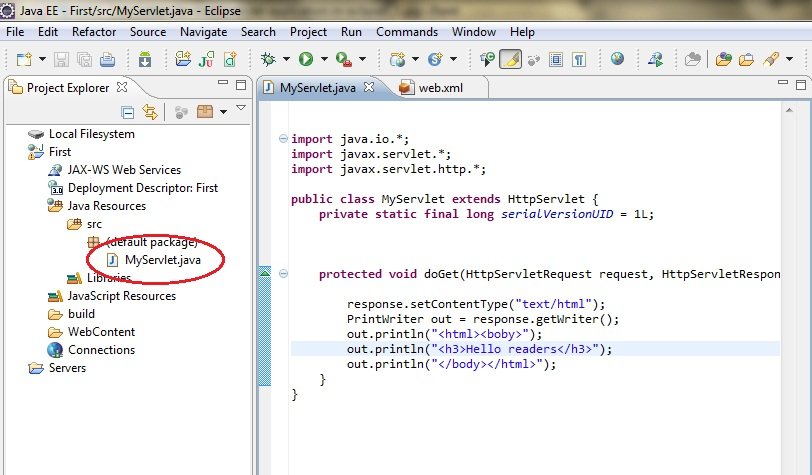
1. Give your Servlet class a Nmae of your choice.



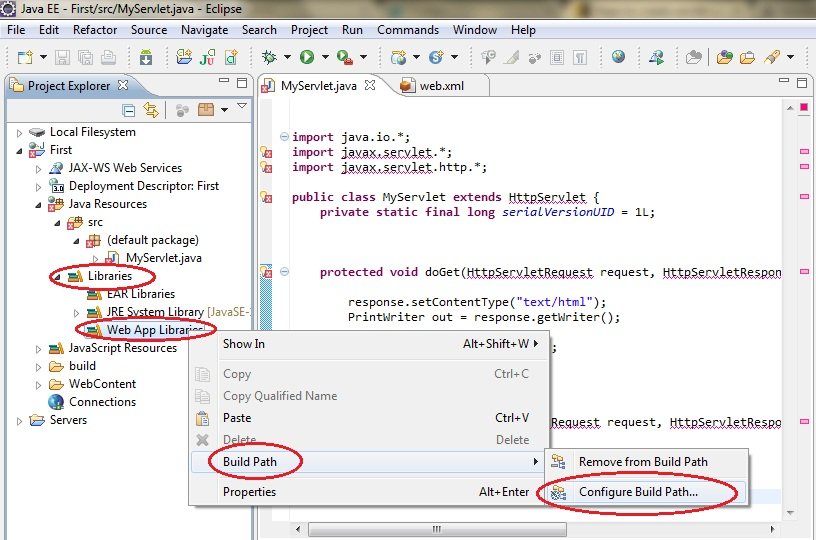
1. Leave everything else to default and click **Finish**



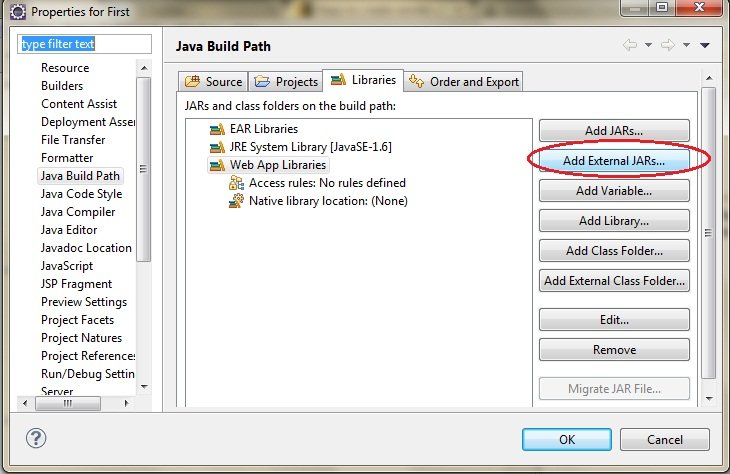
1. Now your Servlet is created, write some code inside it. You can take reference from the code in the picture below.



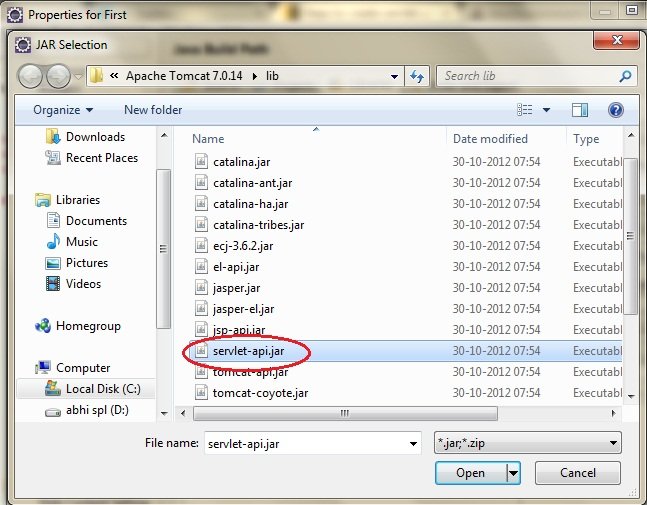
1. Add **servlet-api.jar** JAR file to your project. Click on **Libraries**, right click on **Web App Libraries** select **Build Path** -> **Configure Build Path**

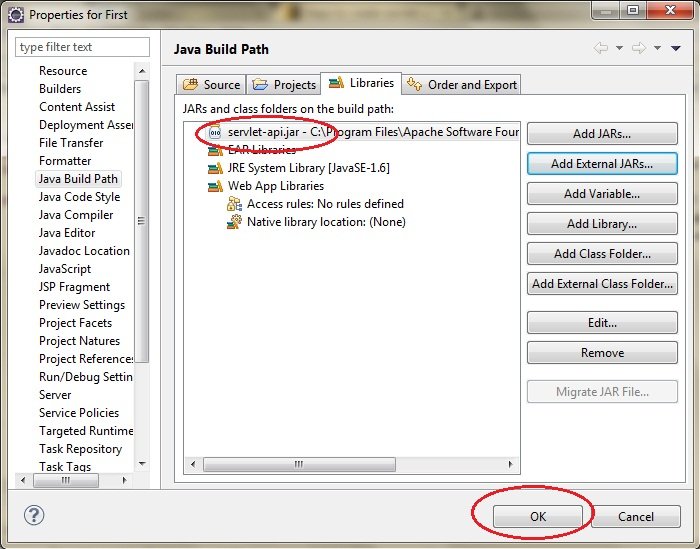


1. Click on **Add External JARs**

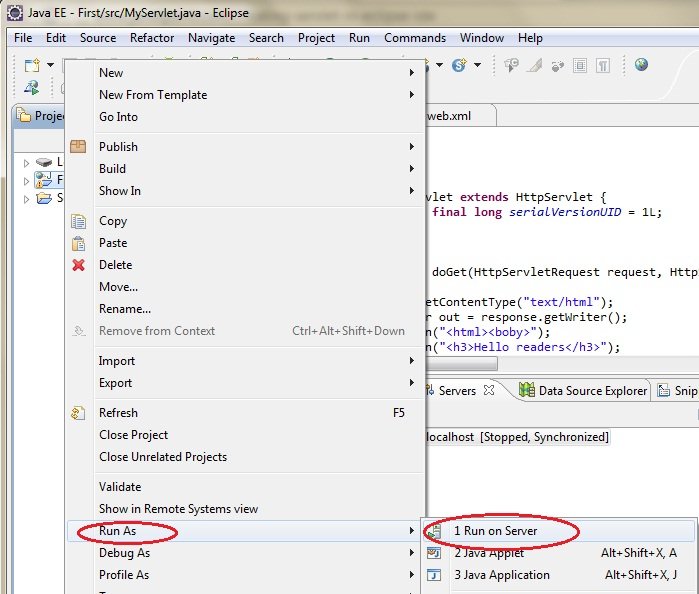


1. This JAR is now added to your project's build path.
2. Select **servlet-api.jar** from **Apache Tomcat Directory**





1. Now all you have to do is Start the server and run the application.



Example of Servlet Response on Eclipse

Step 1: Create a dynamic project on eclipse.

Step 2: Now create an HTML file.

**And write the below code.**

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<form action="display" method="get">

Enter User name: <input type="text" name="val1">

Enter Password: <input type="text" name="val2">

<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">

<servlet>

<servlet-name>abc3</servlet-name>

<servlet-class>com.app.studytonight.demo4</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>abc3</servlet-name>

<url-pattern>/display</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.

**Demo4.java**

package com.app.studytonight;

import javax.servlet.http.\*;

import javax.servlet.\*;

import java.io.\*;

public class demo4 extends HttpServlet{

public void doGet(HttpServletRequest req,HttpServletResponse res)

throws ServletException,IOException

{

res.setContentType("text/html");

PrintWriter pwriter=res.getWriter();

String uname=req.getParameter("val1");

String pw=req.getParameter("val2");

pwriter.println("User Details Page:");

pwriter.println("Hello "+uname);

pwriter.println("Your Password is \*\*"+pw+"\*\*");

pwriter.close();

}

}

Copy

**Now, Run the code.**

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:

|  |  |
| --- | --- |
| **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. |

Servlet: Methods of RequestDispatcher

**RequestDispatcher** interface provides two important methods

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>

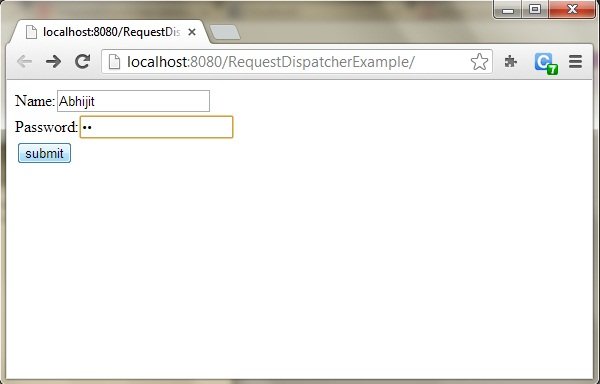
<welcome-file>index.html</welcome-file>

</welcome-file-list>

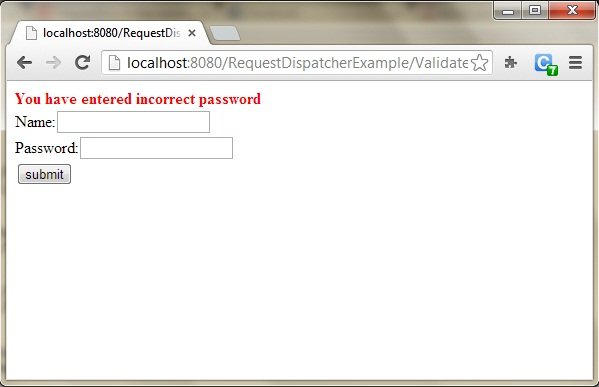
</web-app>

Copy

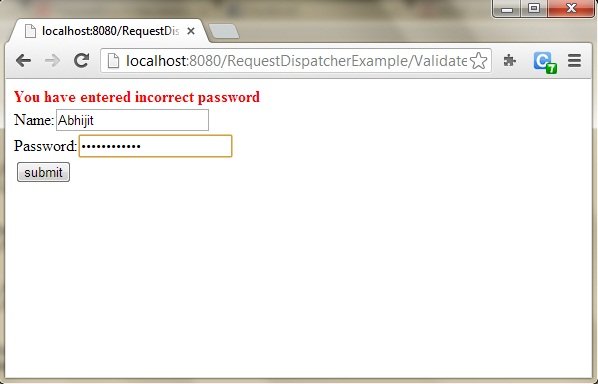
This will be the first screen. You can enter your Username and Password here.



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() Method in Servlet

sendRedirect() method redirects the response to another resource. This method actually makes the client(browser) to create a new request to get to the resource. The client can see the new url in the browser.

**sendRedirect()** accepts relative **URL**, so it can go for resources inside or outside the server.

## Servlet: sendRedirect() and Request Dispatcher

The main difference between a **redirection** and a **request dispatching** is that, redirection makes the client(browser) create a new request to get to the resource, the user can see the new URL while request dispatch get the resource in same request and URL does not changes.

Also, another very important difference is that, sendRedirect() works on **response** object while request dispatch work on **request** object.

### Example demonstrating usage of sendRedirect()

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

try {

response.sendRedirect("https://www.studytonight.com");

}

finally {

out.close();

}

}

}

Copy

Managing Session in Servlets

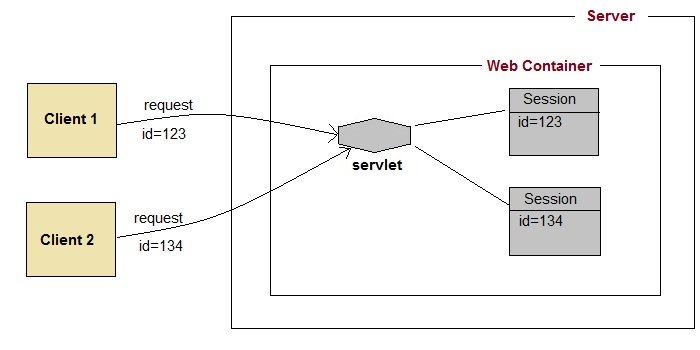
We all know that **HTTP** is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session.

**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

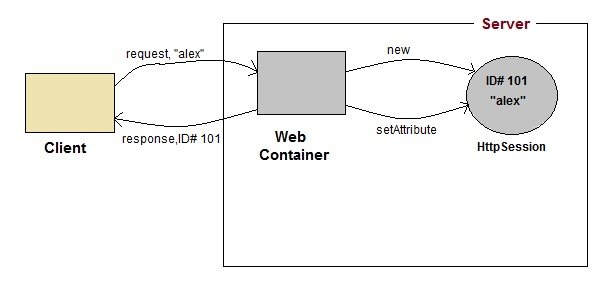


The basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always 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);

}

}

# Using Cookies for Session Management in Servlet

**Cookies** are small pieces of information that are sent in response from the web server to the client. **Cookies** are the simplest technique used for storing client state.

**Cookies** are stored on client's computer. They have a lifespan and are destroyed by the client browser at the end of that lifespan.

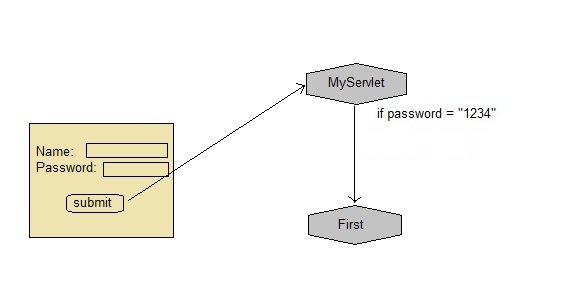
Using Cookies for storing client state has one shortcoming though, if the client has turned of COokie saving settings in his browser then, client state can never be saved because the browser will not allow the application to store cookies.

## Servlet: Cookies API

Cookies are created using **Cookie** class present in Servlet API. Cookies are added to **response** object using the addCookie() method. This method sends cookie information over the HTTP response stream. getCookies() method is used to access the cookies that are added to response object.



### Example demonstrating usage of Cookies



Below mentioned files are required for the example:

**index.html**

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

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...>

<servlet>

<servlet-name>validate</servlet-name>

<servlet-class>MyServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>validate</servlet-name>

<url-pattern>/validate</url-pattern>

</servlet-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

**MyServlet.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet 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"))

{

Cookie ck = new Cookie("username", name);

response.addCookie(ck);

response.sendRedirect("First");

}

}

}

Copy

**First.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class First extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

Cookie[] cks = request.getCookies();

out.println("Welcome "+ cks[0].getValue());

}

}

Copy

### Types of Cookies

There are two types of cookies. They are as following:

* Session
* Persistent

#### 1) Session cookies:

The session cookies do not have any expiration time. It is present in the browser memory. When the web browser is closed then the cookies are destroyed automatically.

#### 2) Persistent Cookies:

The Persistent cookies have an expiration time. It is stored in the hard drive of the user and it is destroyed based on the expiry time.

### How cookies works?

When a user Start a web and request information from the website. The website server replies and it sends a cookie. This cookie is put on the hard drive. Next time when you return to the same website your computer will send the cookies back. Now the website server identifies the data and sale your information to other sellers.

#### demo5.html

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

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

Enter User name: <input type="text" name="val1"><br>

Enter Password: <input type="text" name="val2"><br>

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

</form>

</body>

</html>

Copy

#### cookie1.html

package com.app.studytonight;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class cookie1 extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response){

try{

response.setContentType("text/html");

PrintWriter out = response.getWriter();

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

out.print("Welcome "+n);

Cookie ck=new Cookie("uname",n);

response.addCookie(ck);

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

out.print("<input type='submit' value='go'>");

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

out.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

Copy

#### cookie2.html

package com.app.studytonight;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class cookie2 extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response){

try{

response.setContentType("text/html");

PrintWriter out = response.getWriter();

Cookie ck[]=request.getCookies();

out.print("Hello "+ck[0].getValue());

out.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

Copy

#### web.xml

<servlet>

<servlet-name>s1</servlet-name>

<servlet-class>com.app.studytonight.cookie1</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/abc</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>s2</servlet-name>

<servlet-class>com.app.studytonight.cookie1</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>s2</servlet-name>

<url-pattern>/pqr</url-pattern>

</servlet-mapping>

</web-app>

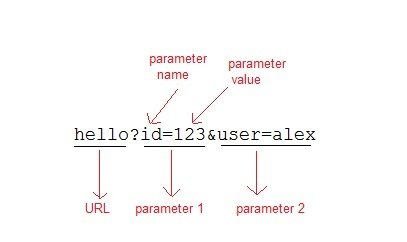
Copy

# Using URL Rewriting for Session Management in Servlet

If the client has disabled cookies in the browser then session management using cookie wont work. In that case **URL Rewriting** can be used as a backup. **URL rewriting** will always work.

In URL rewriting, a token(parameter) is added at the end of the URL. The token consist of name/value pair seperated by an equal(=) sign.

**For Example:**



When the User clicks on the URL having parameters, the request goes to the **Web Container** with extra bit of information at the end of URL. The **Web Container** will fetch the extra part of the requested URL and use it for session management.

The getParameter() method is used to get the parameter value at the server side.

#### Example demonstrating usage of URL rewriting

Below mentioned files are required for the example:

**index.html**

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

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...>

<servlet>

<servlet-name>validate</servlet-name>

<servlet-class>MyServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>validate</servlet-name>

<url-pattern>/validate</url-pattern>

</servlet-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

**MyServlet.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet 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"))

{

response.sendRedirect("First?user\_name="+ name);

}

}

}

Copy

**First.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class First extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

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

out.println("Welcome "+user);

}

}

Copy

Using Hidden Form Field for Session Management in Servlet

Hidden form field can also be used to store session information for a particular client. In case of hidden form field a hidden field is used to store client state. In this case user information is stored in hidden field value and retrieved from another servlet.

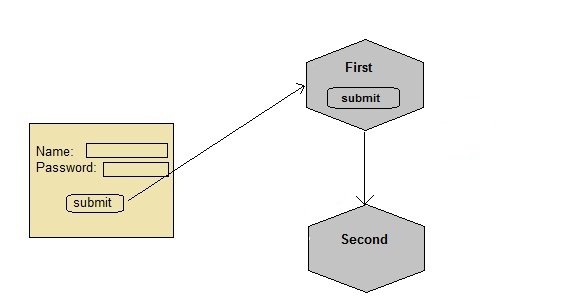
Advantages of Using Hidden Form Field for Session Management

* Does not have to depend on browser whether the cookie is disabled or not.
* Inserting a simple HTML Input field of type hidden is required. Hence, its easier to implement.

Disadvantage of Using Hidden Form Field for Session Management

* Extra form submission is required on every page. This is a big overhead.

Example demonstrating usage of Hidden Form Field for Session



Below mentioned files are required for the example:

**index.html**

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

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...>

<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>

<servlet>

<servlet-name>Second</servlet-name>

<servlet-class>Second</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Second</servlet-name>

<url-pattern>/Second</url-pattern>

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

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();

//getting value submitted in form from HTML file

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

//creating a new hidden form field

out.println("<form action='Second'>");

out.println("<input type='hidden' name='user' value='"+user+"'>");

out.println("<input type='submit' value='submit' >");

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

}

}

Copy

**Second.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class Second extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

//getting parameter from the hidden field

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

out.println("Welcome "+user);

}

}

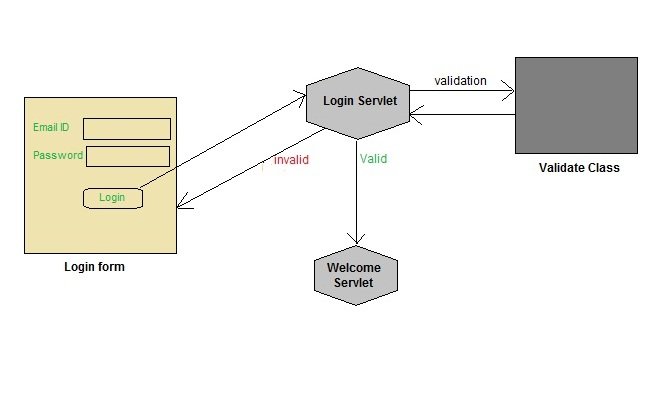
Copy

Like we created a hidden field in **First** Servlet, populated the value of user, and sent it to the **Second** Servlet, now Second servlet also has the user information. Similarly we will have to keep sending this information, wherever we need this, using hidden fields.

# Login System in Servlet

In 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>

# 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>");

List<stu> list=stuDao.getAllStudent();

out.print("<table border='1' width='100%'");

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

for(stu e1:list){

out.print("<tr><td>"+e1.getId1()+"</td><td>"+e1.getName1()+"</td><td>"+e1.getAge1()+"</td><td>"+e1.getCourse1()+"</td><td>"+e1.getCity1()+"</td><td><a href='EditServlet?id="+e1.getId1()+"'>edit</a></td><td><a href='DeleteServlet?id="+e1.getId1()+"'>delete</a></td></tr>");

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

out.close();

}

}

Copy

Run the Program.

To add a record of fill the fields and click on the save button.

Record save successfully. You can add more records.

To view the records click on the link View Student. After clicking on the link, below are the records.

To view the records click on the link View Student. After clicking on the link, below are the records.

To delete any record click on the delete link of that record. The record will be deleted from the database.

Data of Kiran has been deleted.