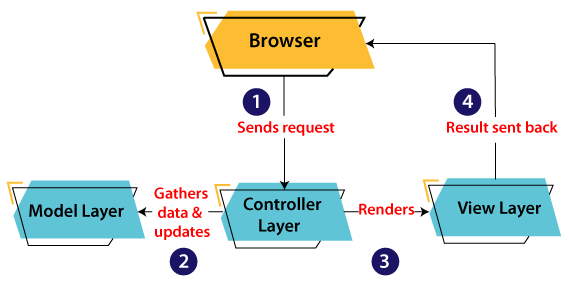
MVC Architecture in Java

* **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.

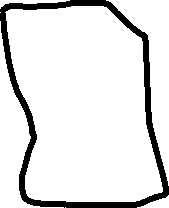
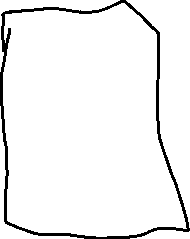
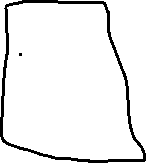
Let's understand the components of a web application:

### **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.



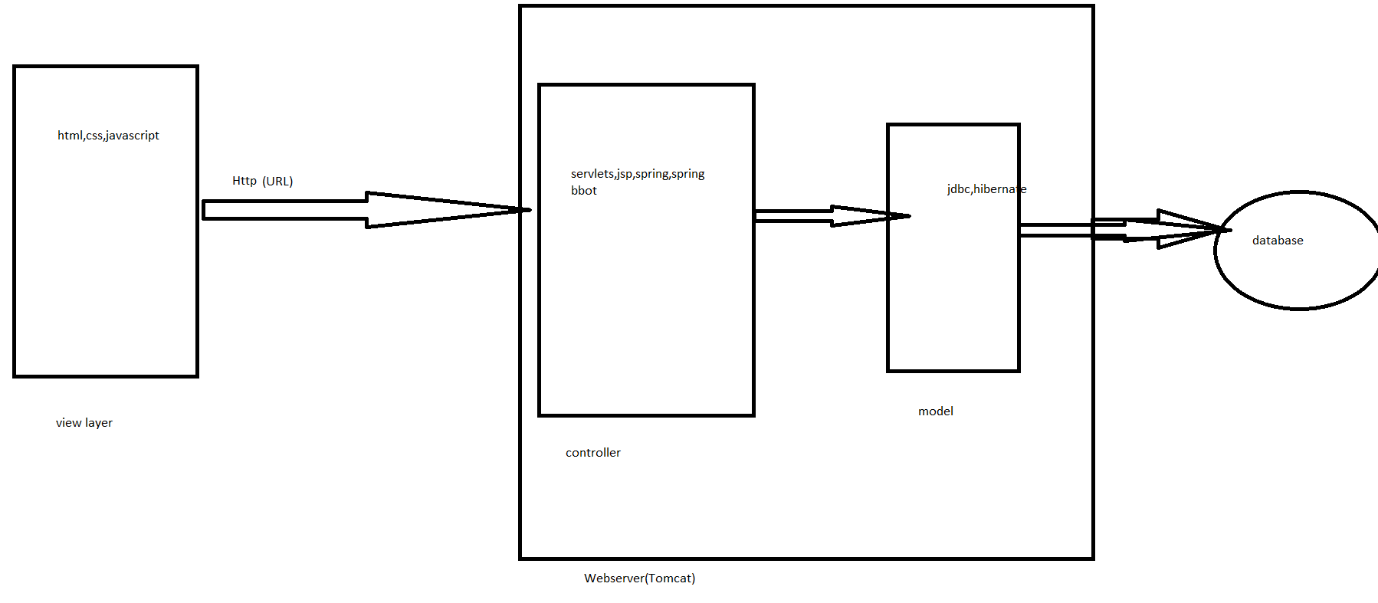
html, servlets,



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.

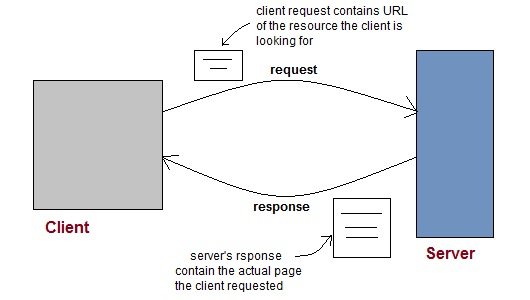
The **javax.servlet** and **javax.servlet.http** packages contain interfaces and classes for creating servlets. All servlets should implement the Servlet interface, which defines life-cycle methods. To implement a generic service, we can use the GenericServlet class by extending it. It provides **doGet**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

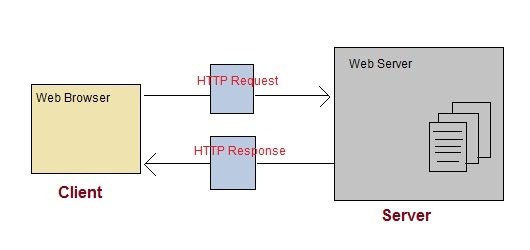


## **Web Application**

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.

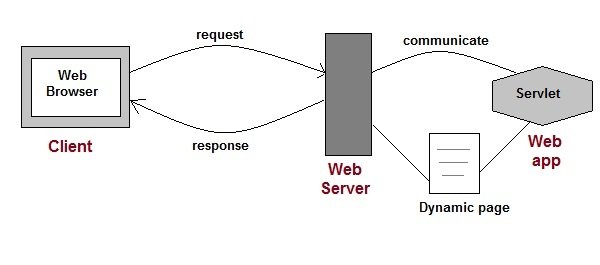


### **HTTP Methods**

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 |

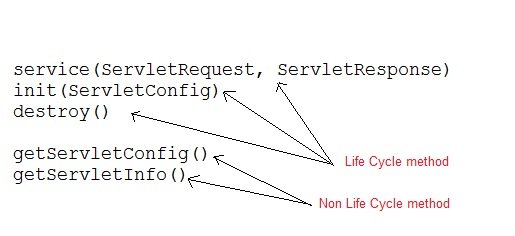
## **Servlet Interface**

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**

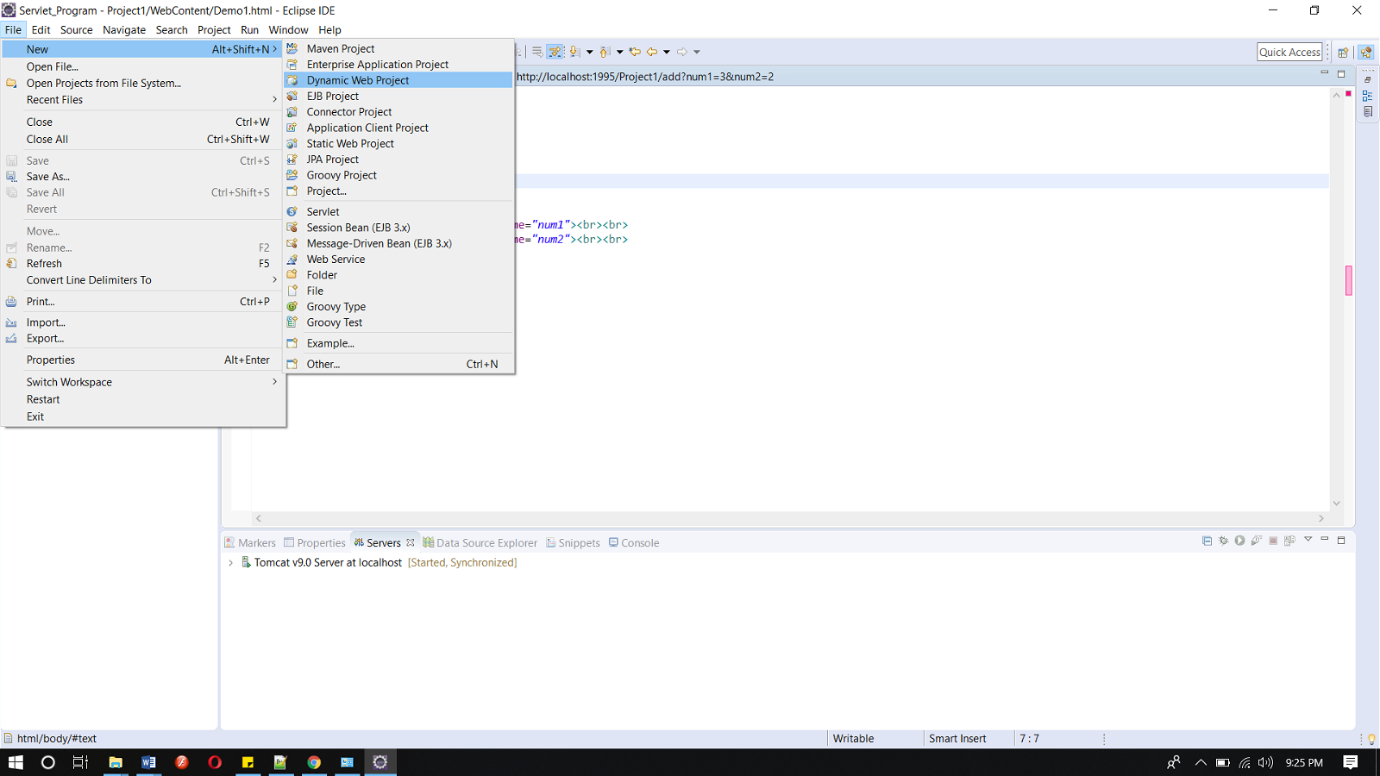
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Method** | **Description** |
| 1. | public void init(ServletConfigconfig) | It is used for initializing the servlet. It is invoked only once by the web container in a servlet life cycle. |
| 2. | public void service(ServletRequestreq, ServletResponse res) | 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:

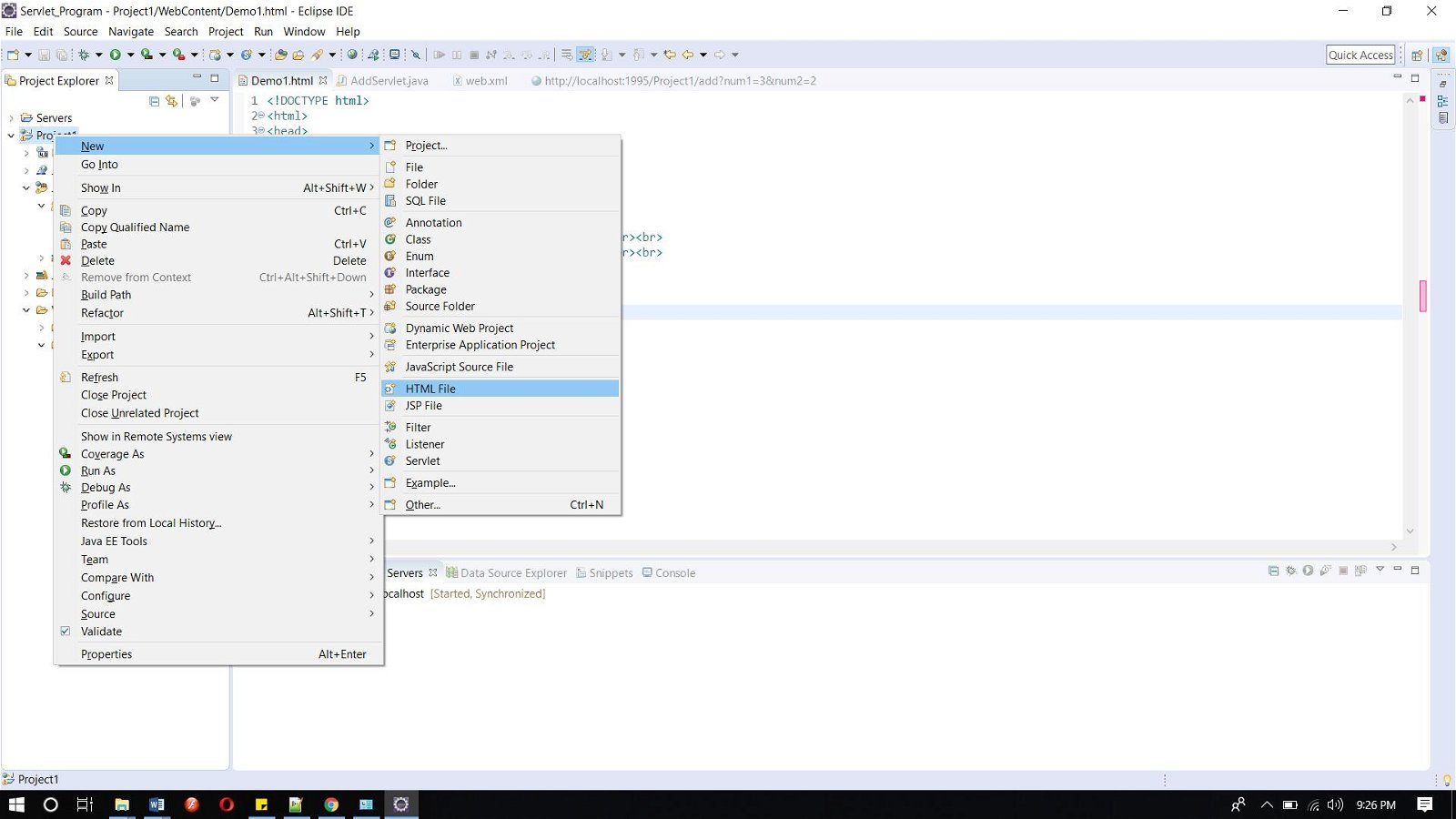
Following are the steps for creating the program.

**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>

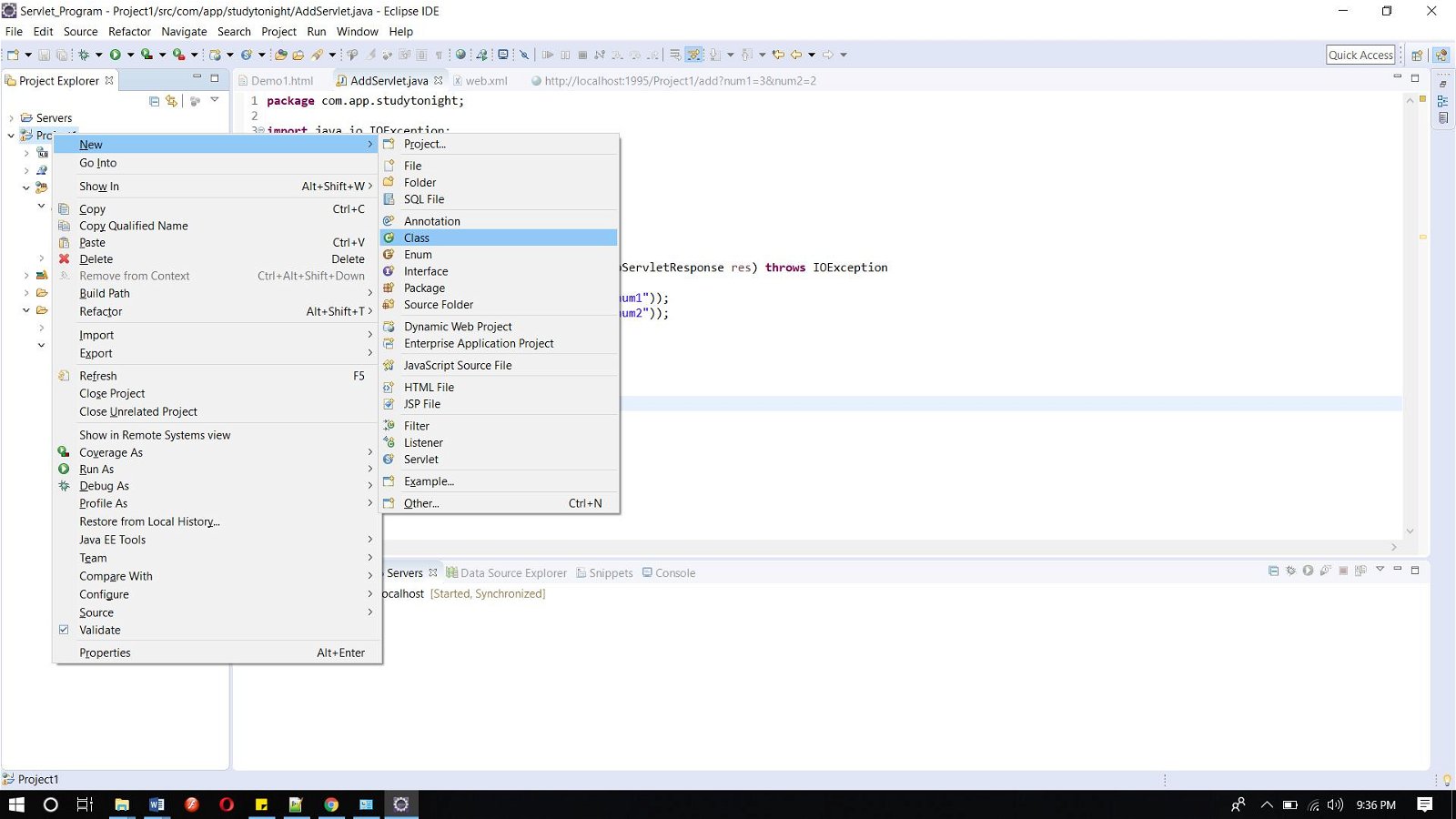
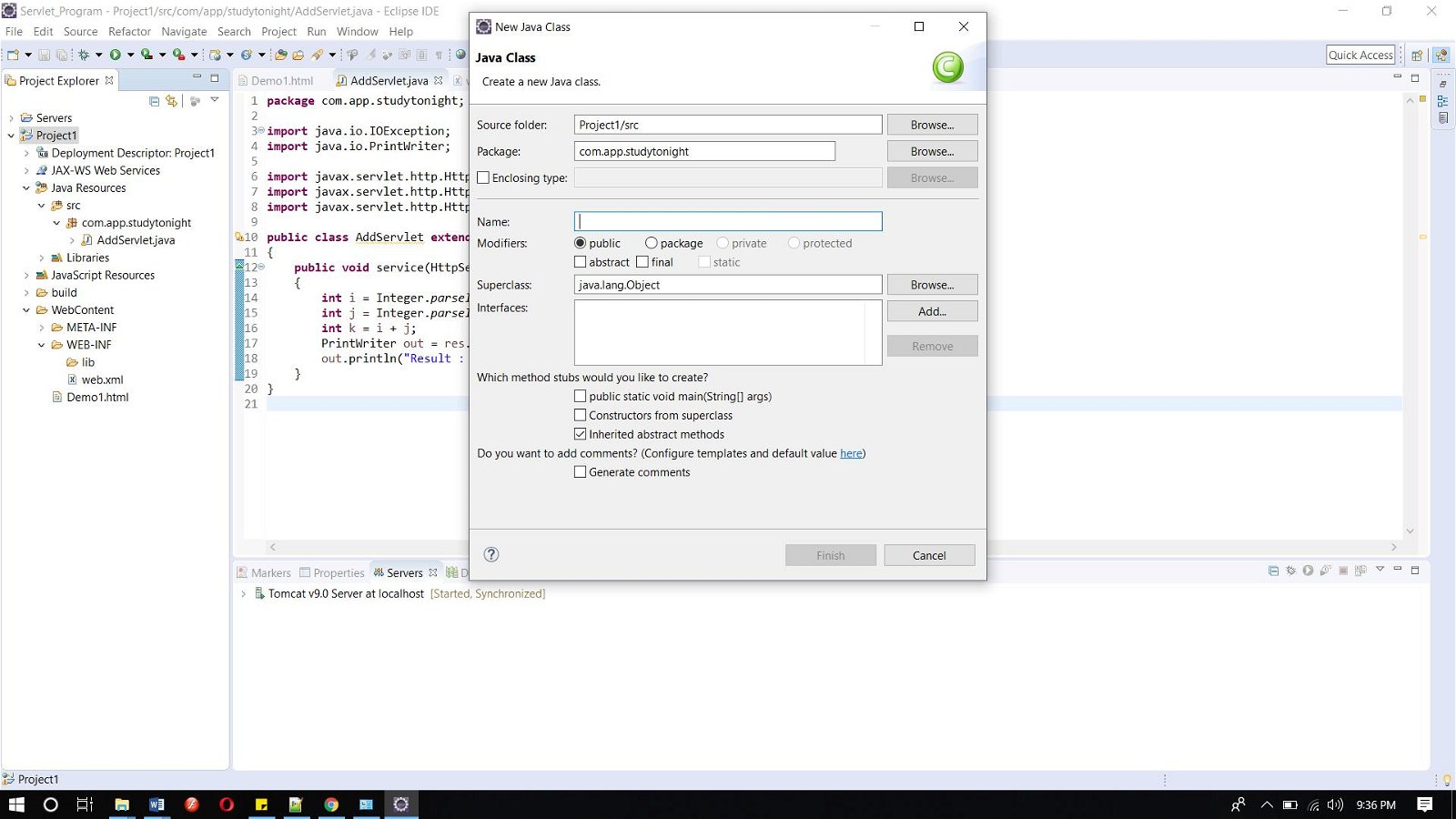
<url-pattern>/demo</url-pattern>

</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";

}

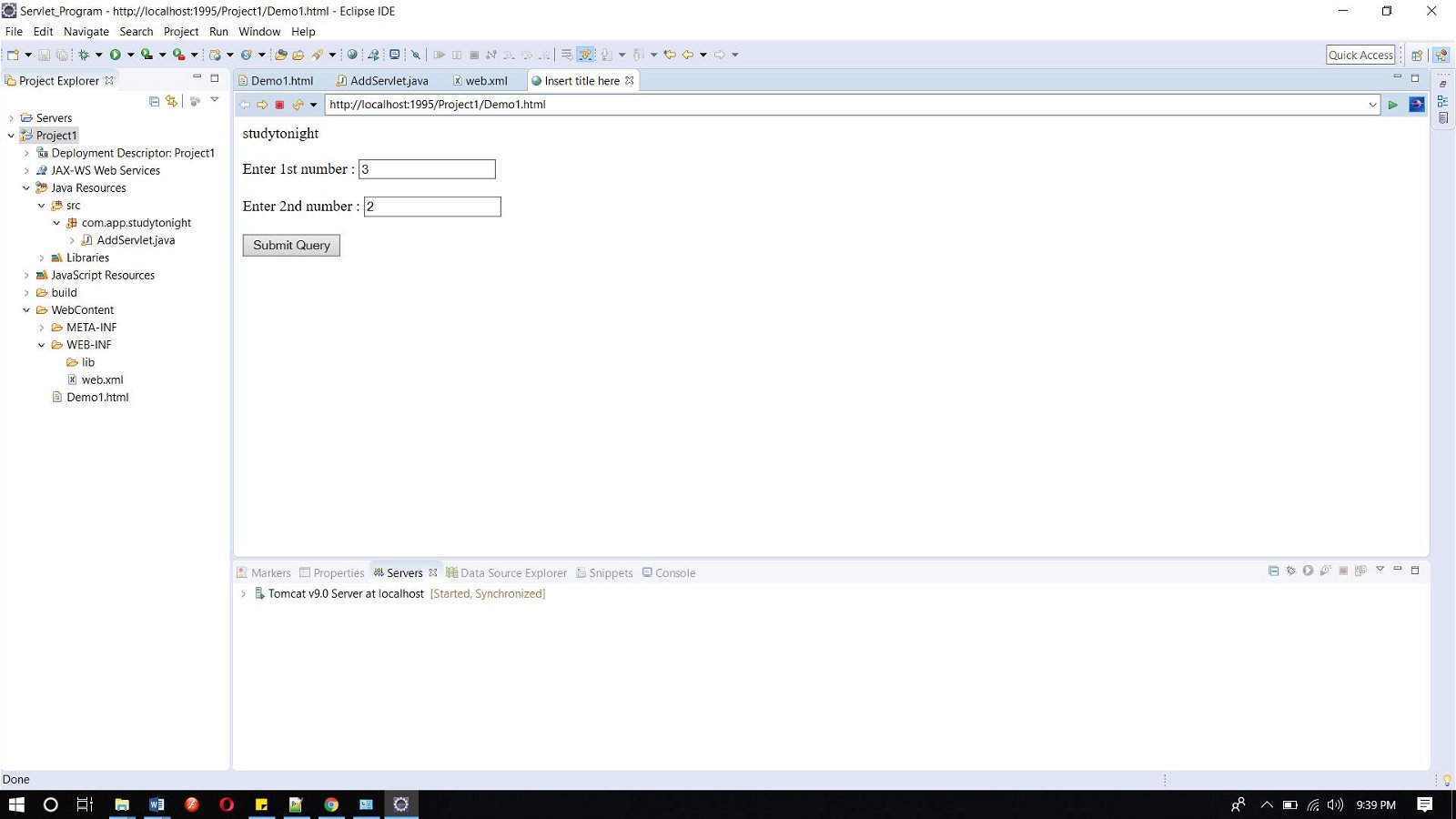
}

Copy

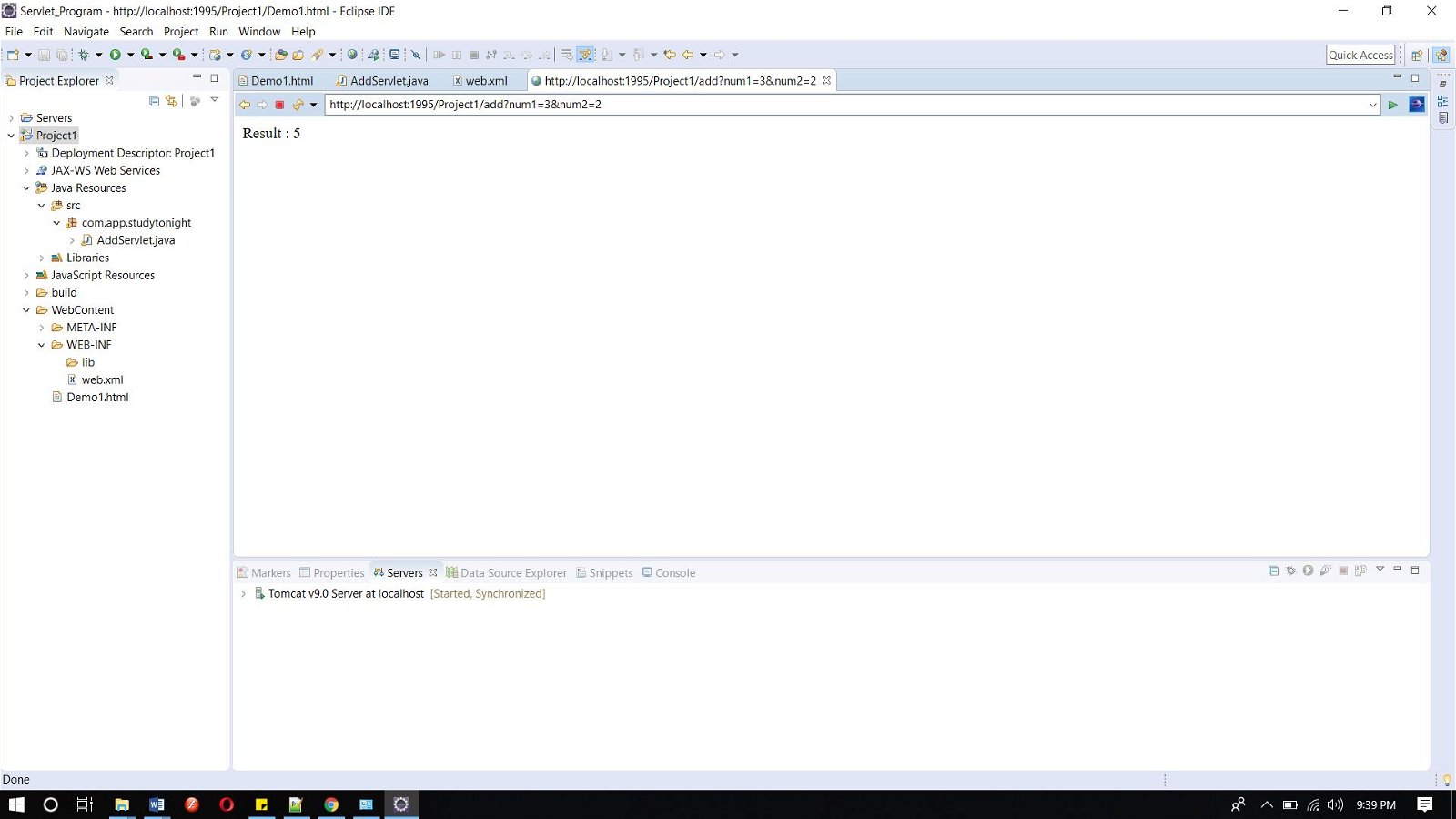
#### Now, Run the code.

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.

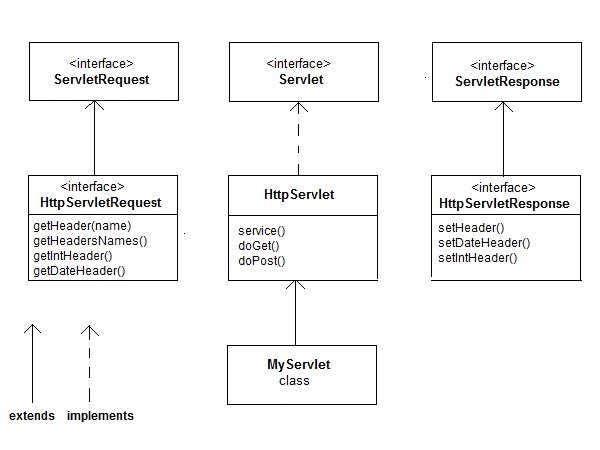


## **HttpServlet class**

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**

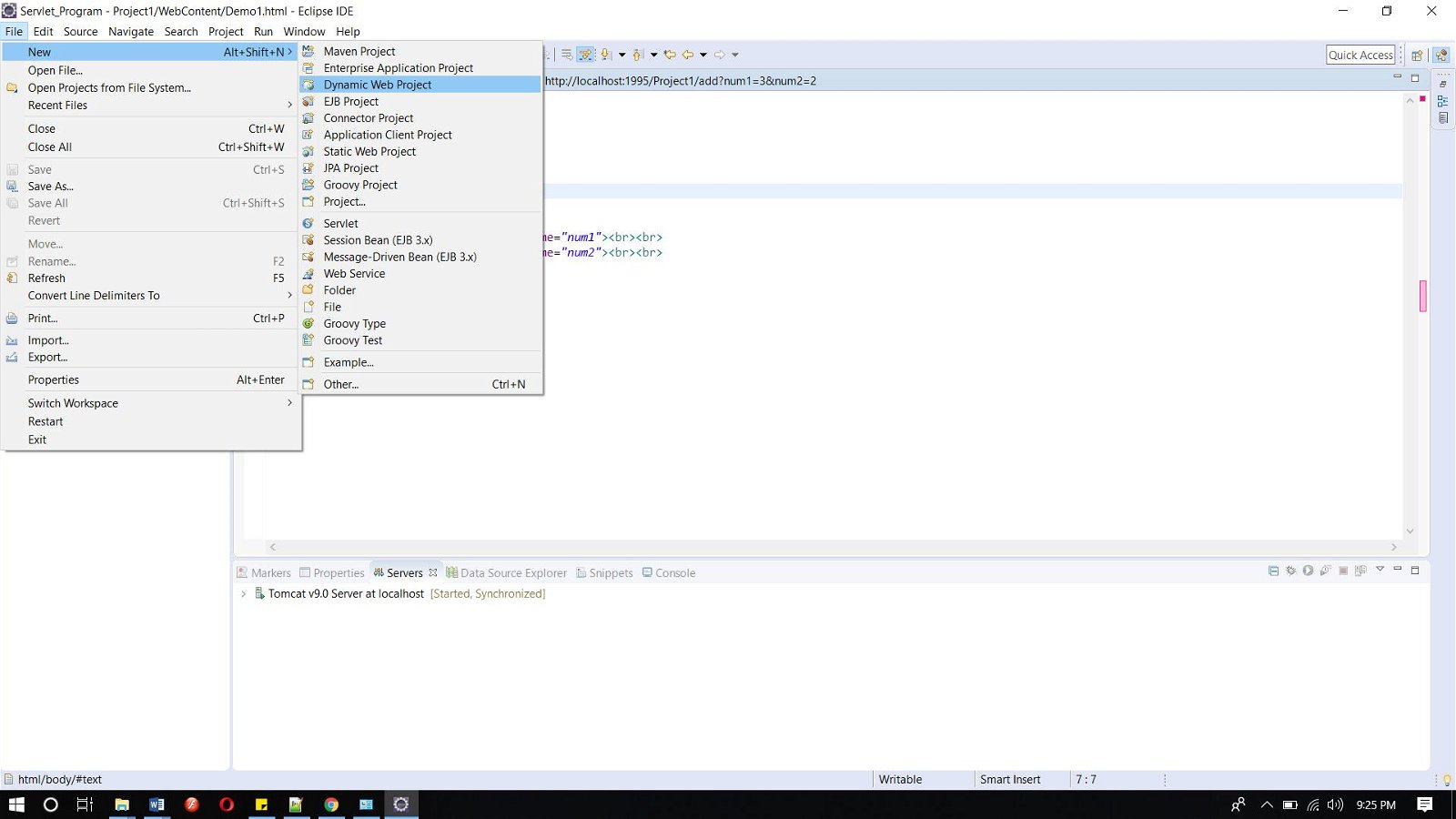
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Method** | **Description** |
| 1 | public void service(ServletRequest req,ServletResponse res) | It is used for securing the service method by creating objects of request and response. |
| 2 | protected void service(HttpServletRequest req, HttpServletResponse 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:

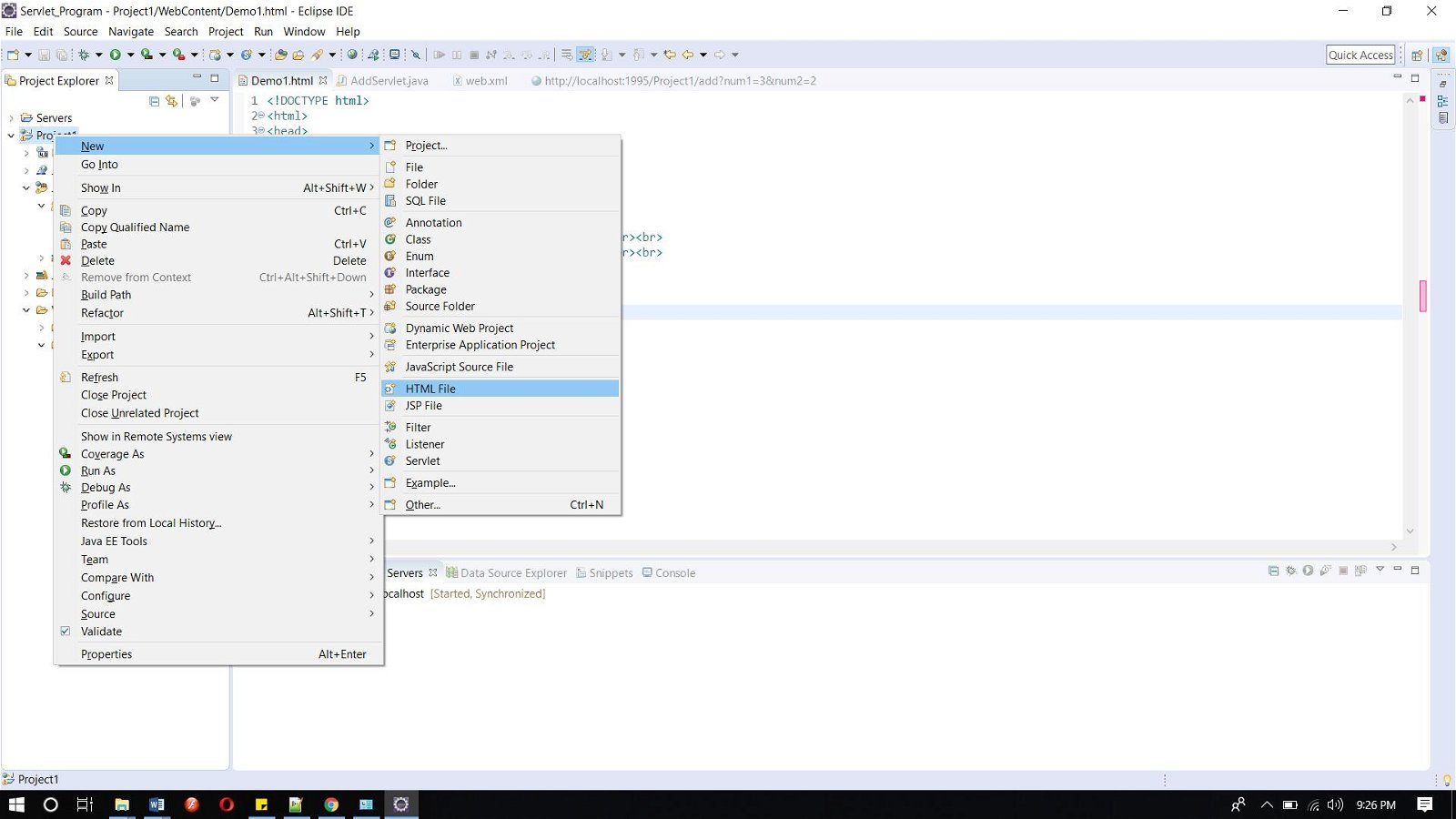
Following are the steps for creating the program.

**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>

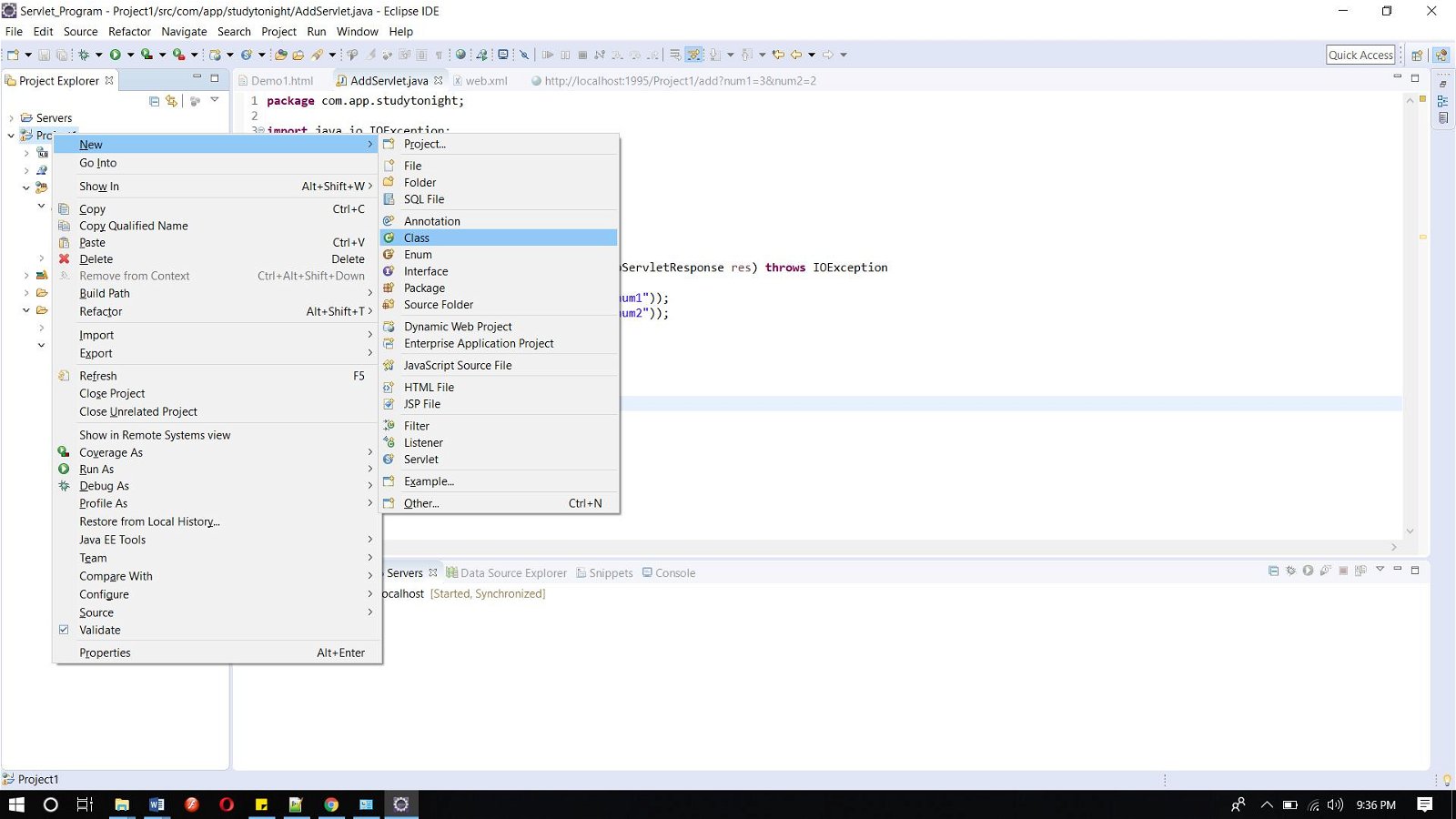
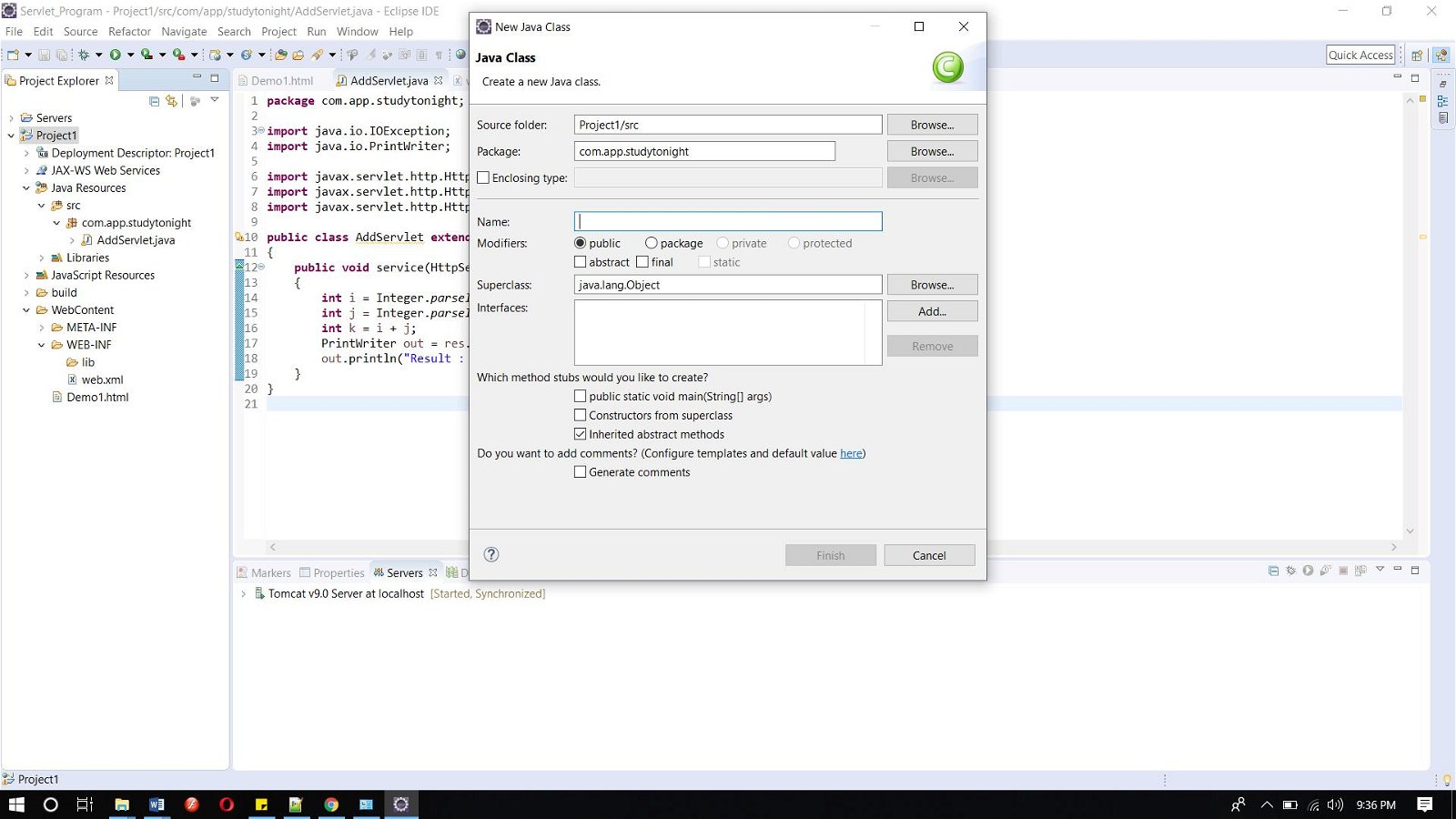
<url-pattern>/mar</url-pattern>

</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);

}

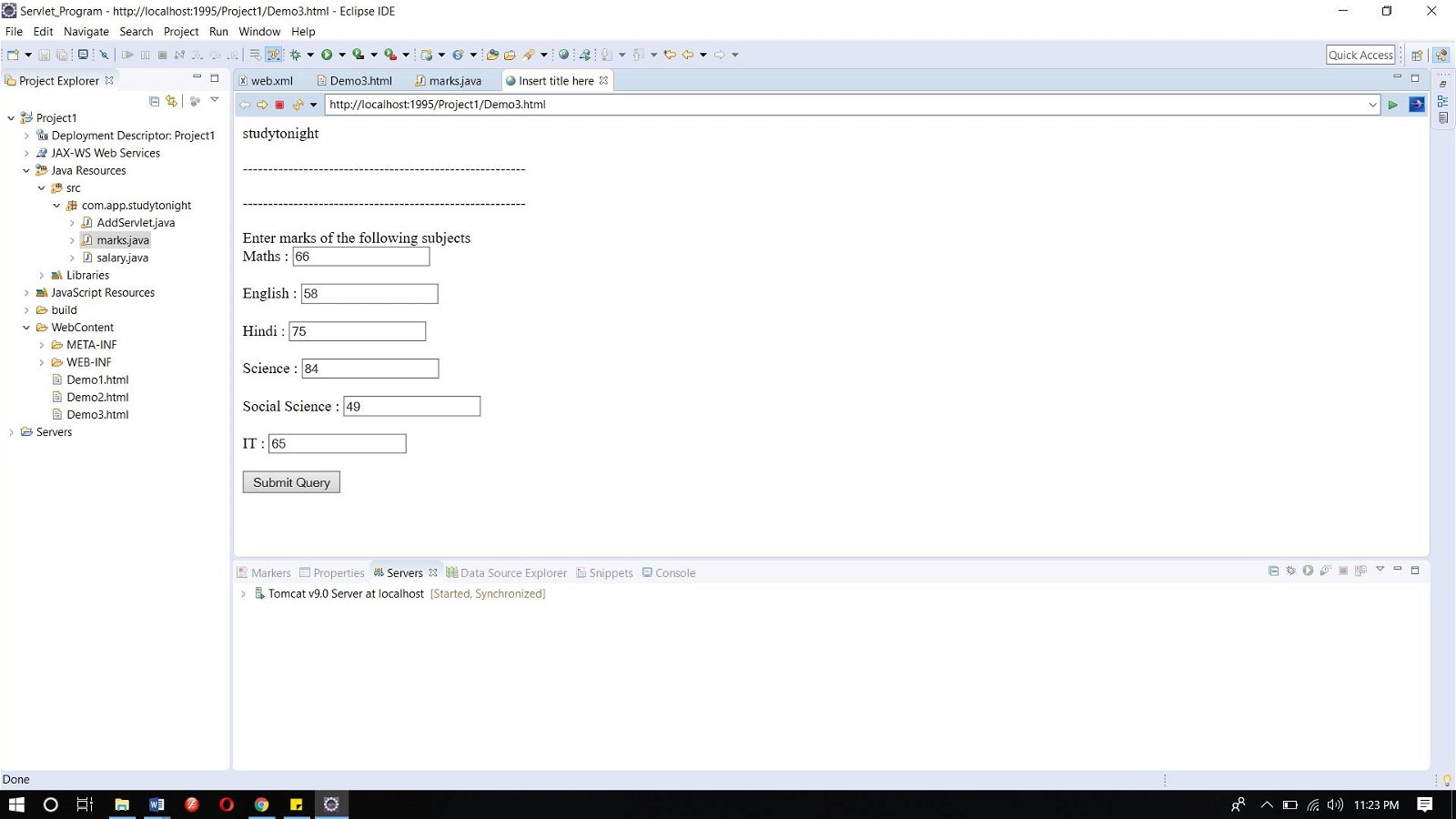
}

Copy

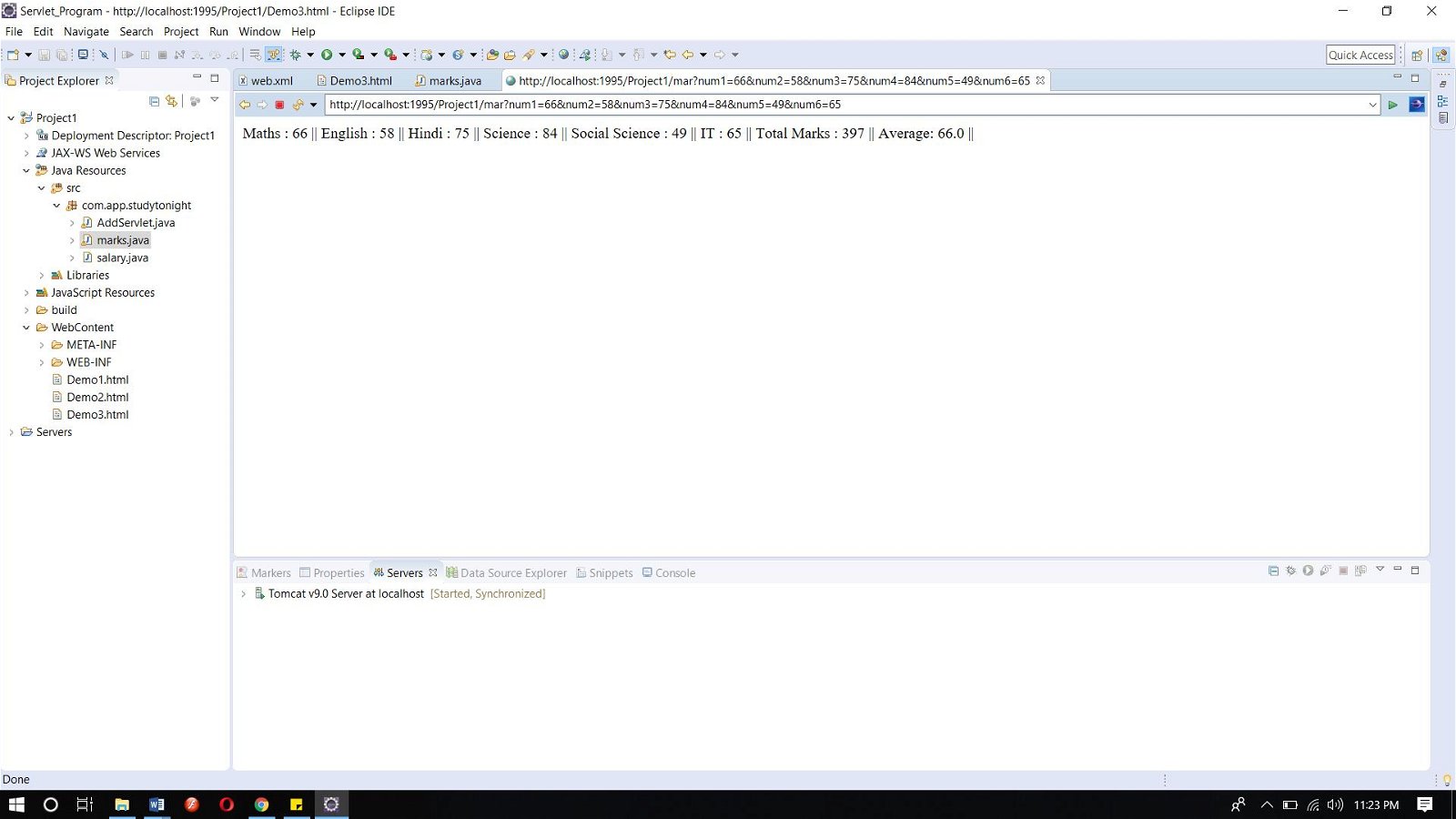
#### Now, Run the code.

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.

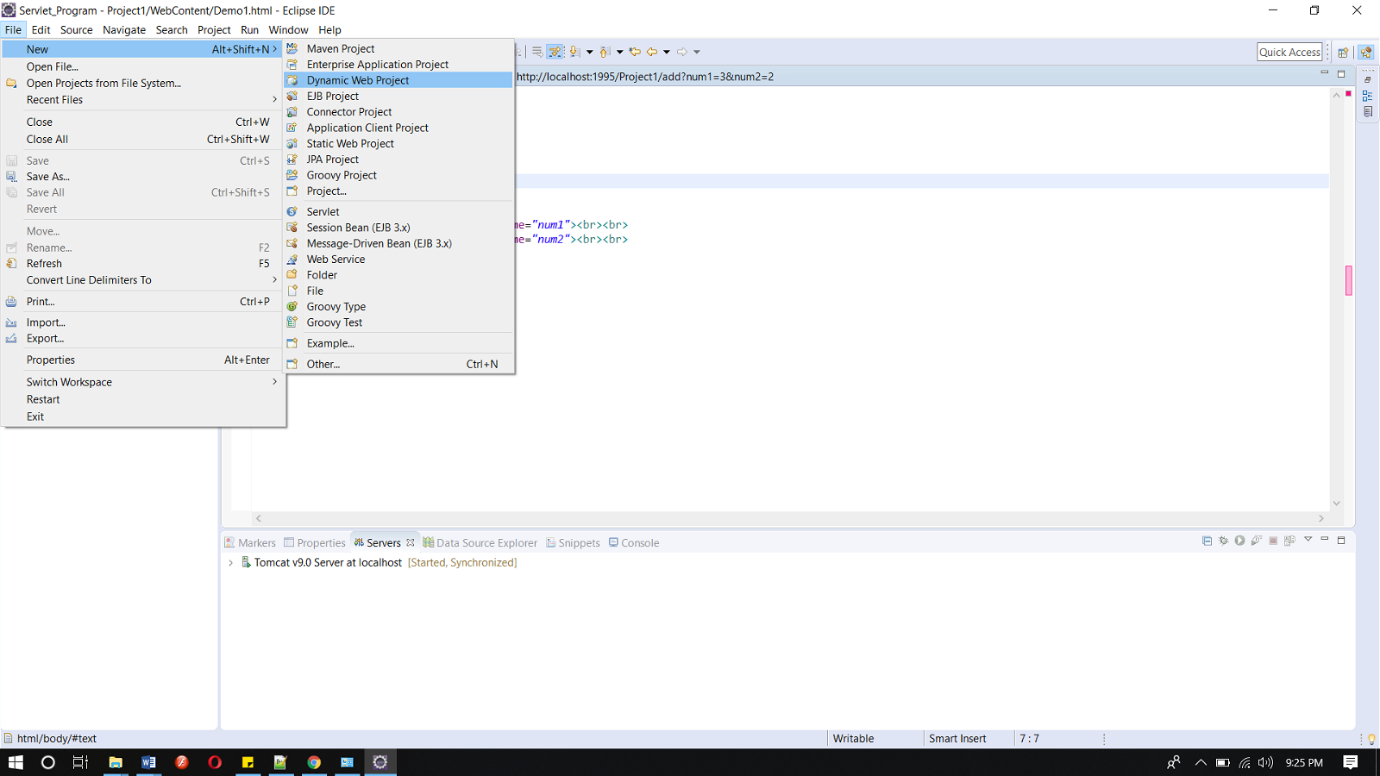
|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Method** | **Desciption** |
| 1 | public void init(ServletConfig config) | It is used for initialization of a servlet. |
| 2 | public abstract void service(ServletRequest request, ServletResponse response) | It is used for providing all the services for 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**

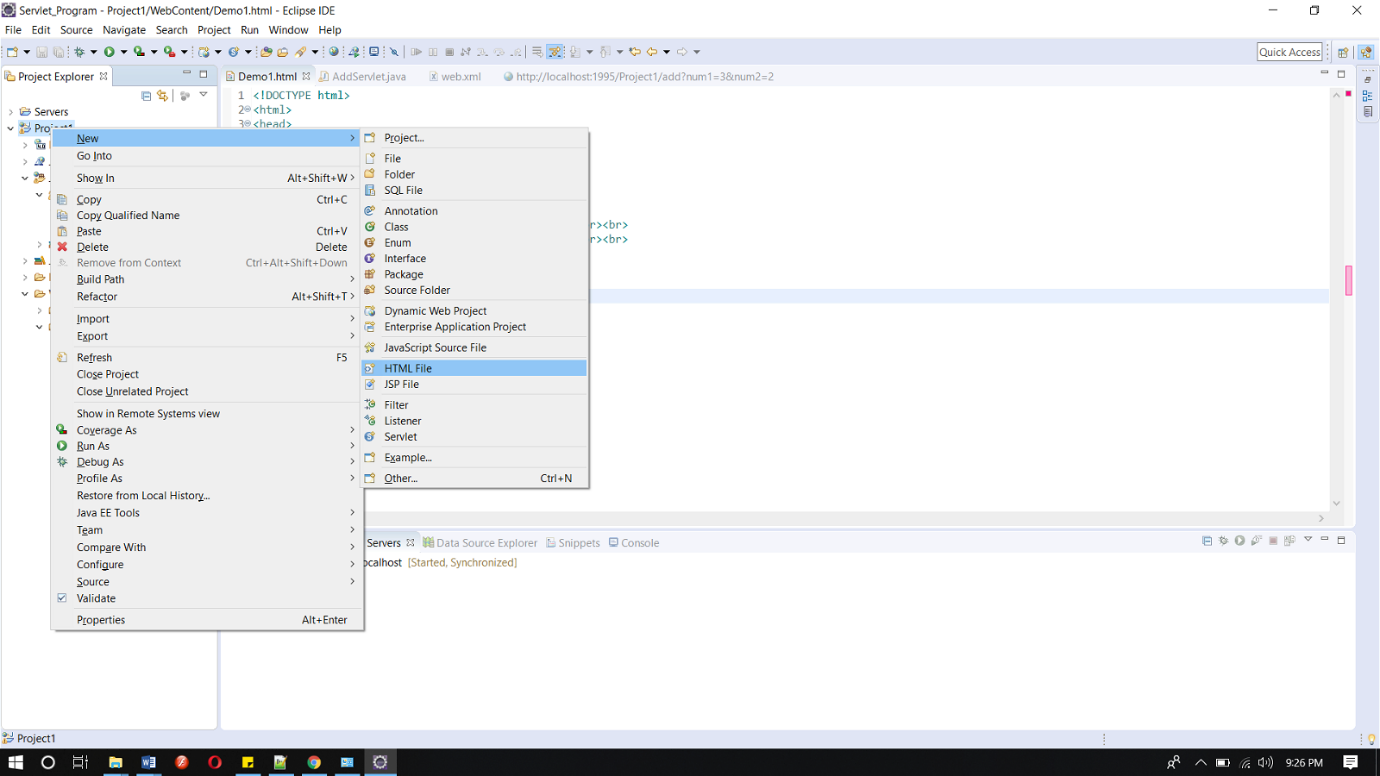
For creating a GenericServlet class below is the directory structure of the program:

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>

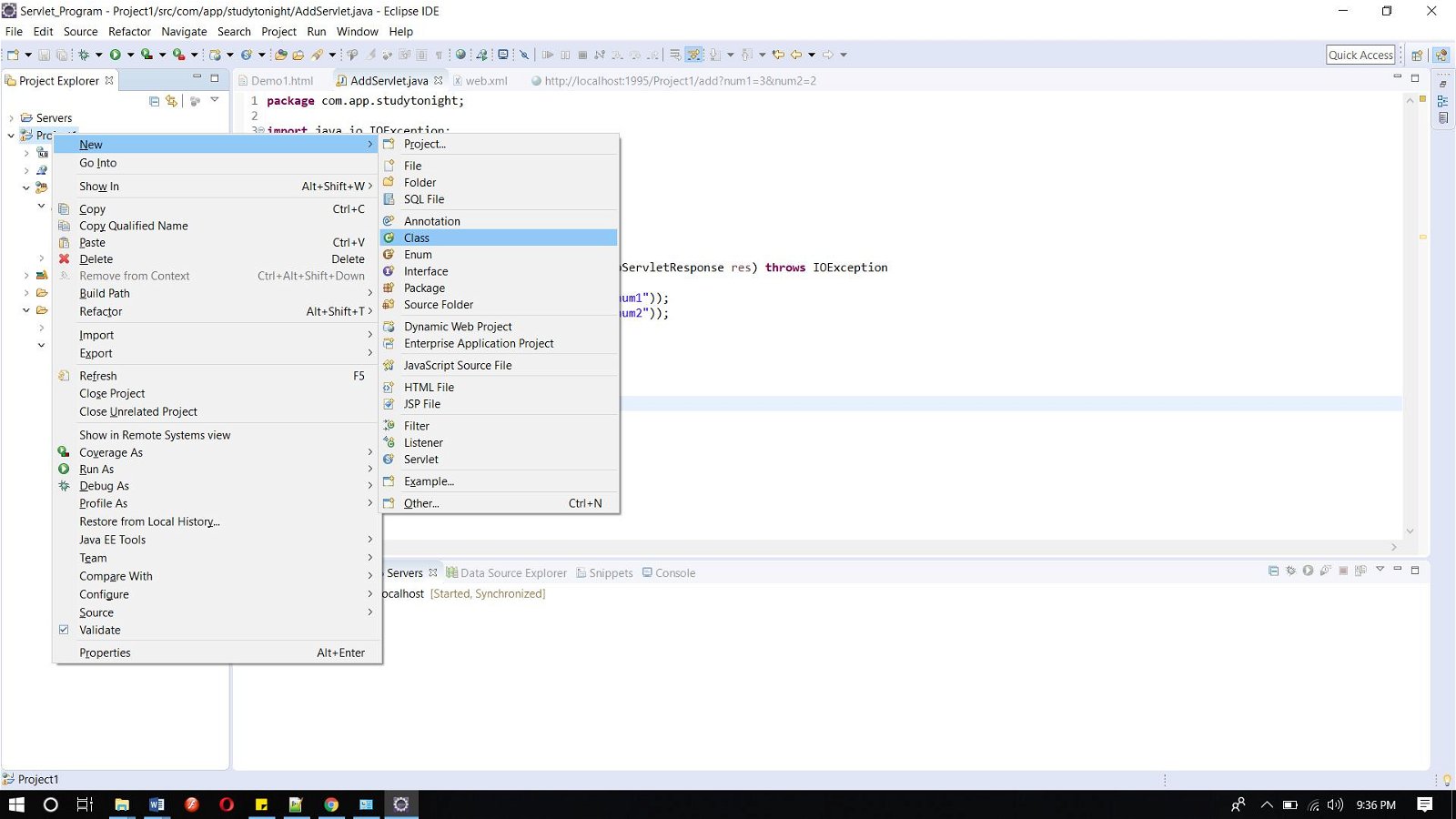
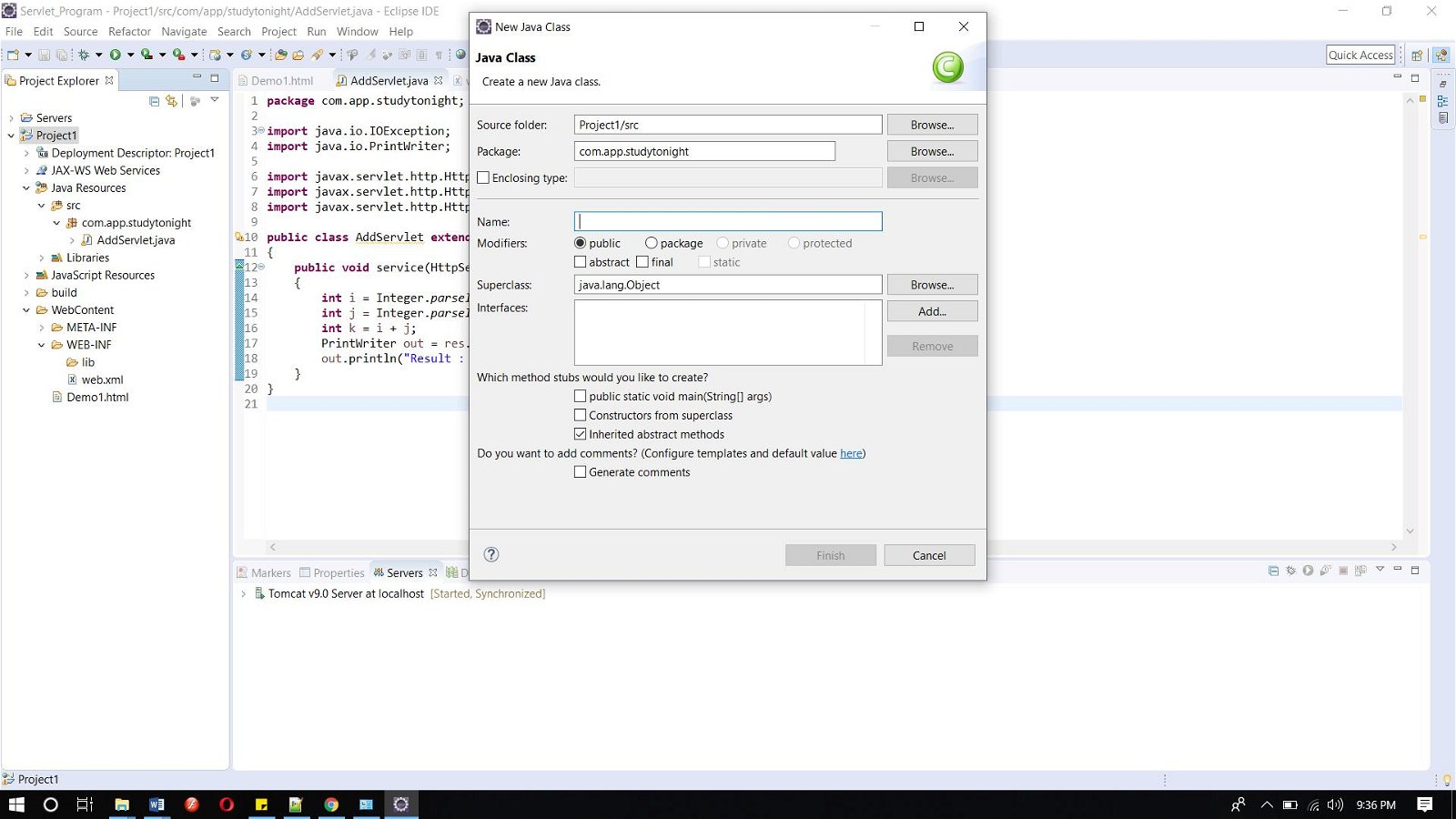
<url-pattern>/sal</url-pattern>

</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);

}

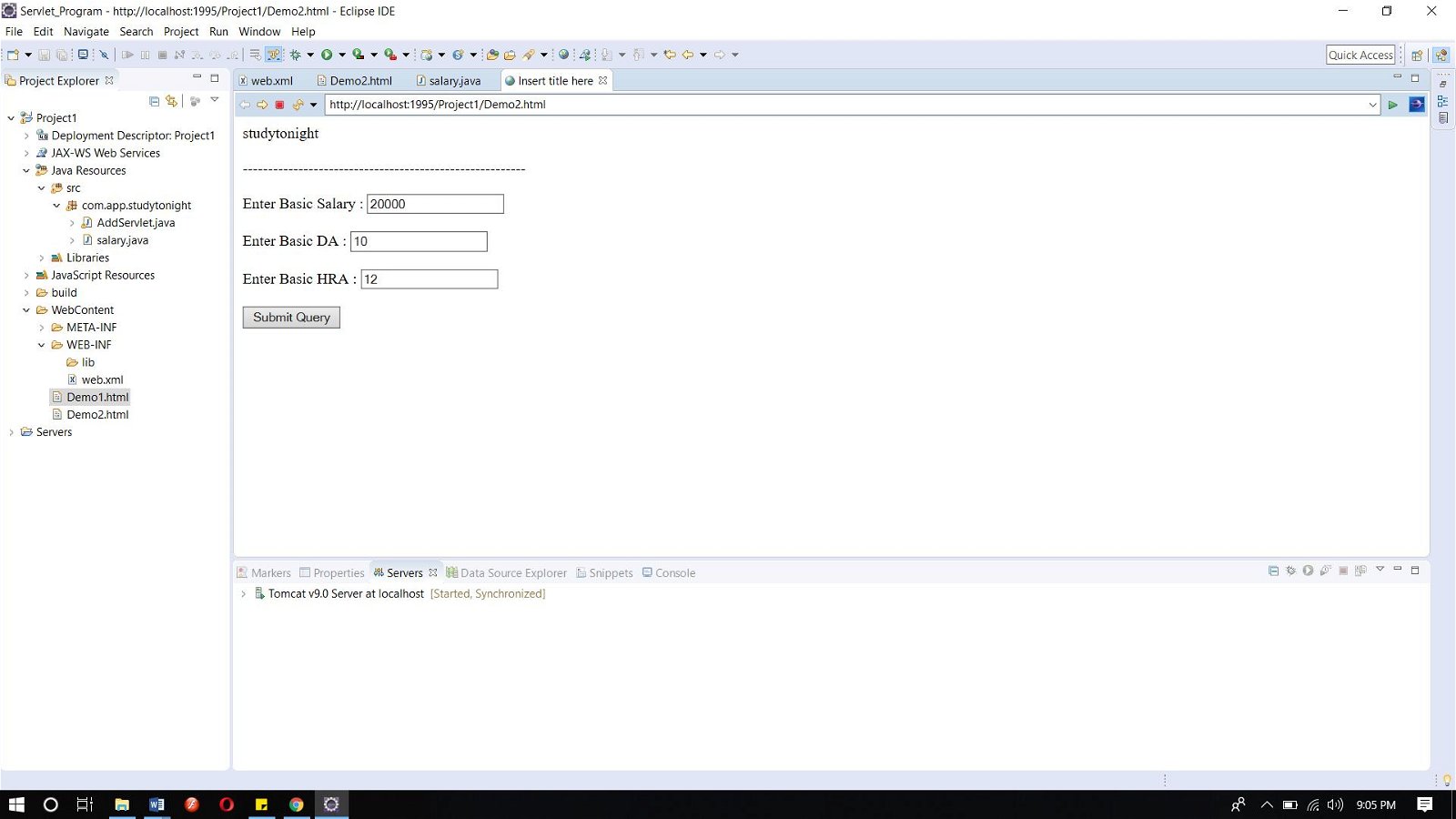
}

Copy

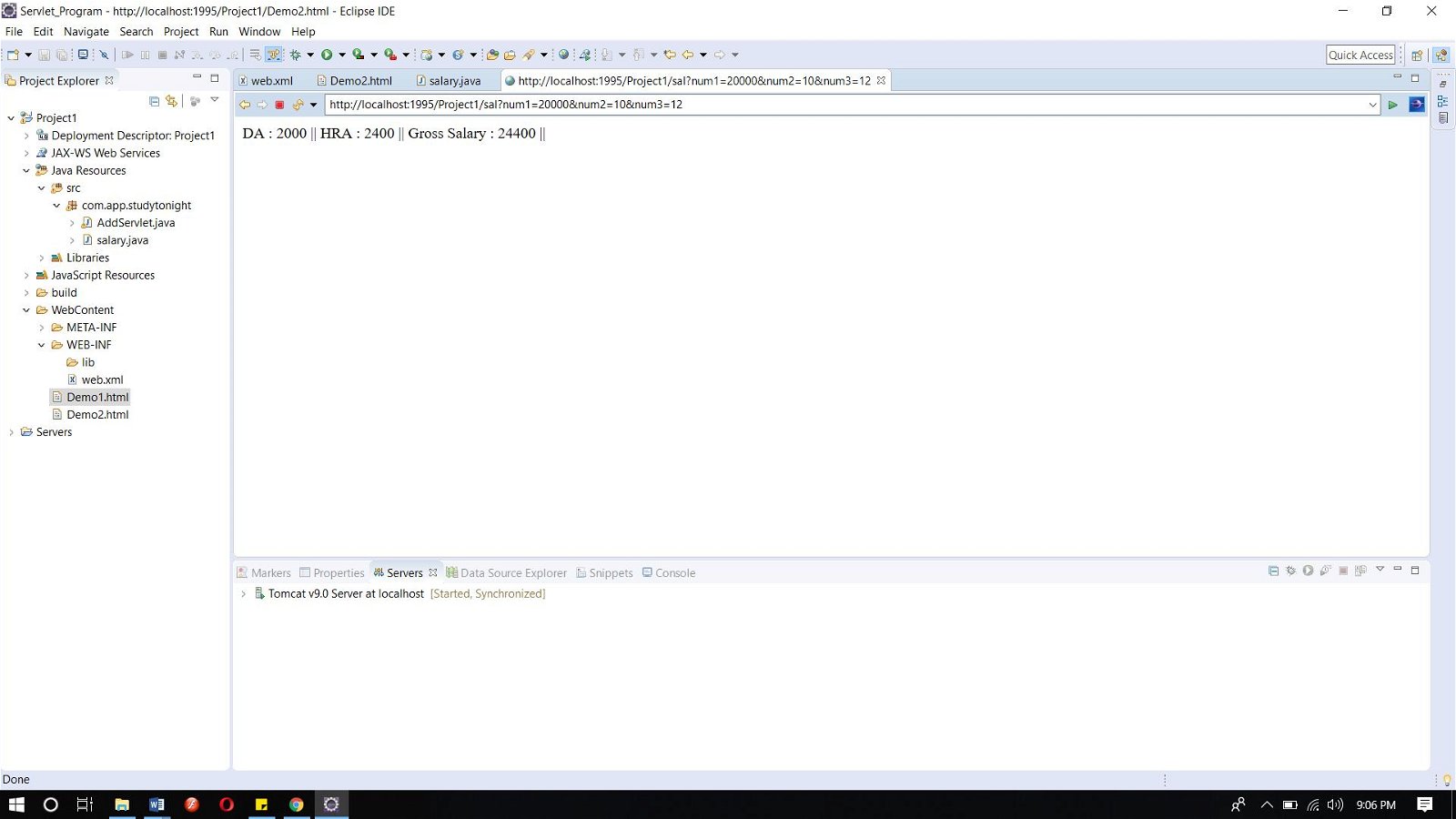
#### Now, Run the code.

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

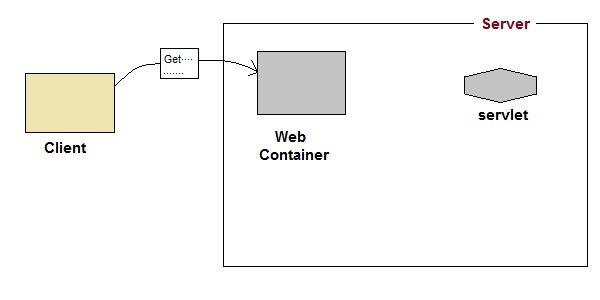
**Web container** is responsible for managing execution of servlets and [JSP pages](https://www.studytonight.com/jsp/introduction-to-jsp.php) for Java EE application.

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.

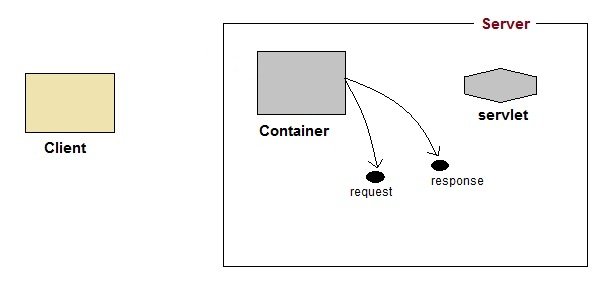
**Servlets don't have a main() method**. Web Container manages the life cycle of a Servlet instance.

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

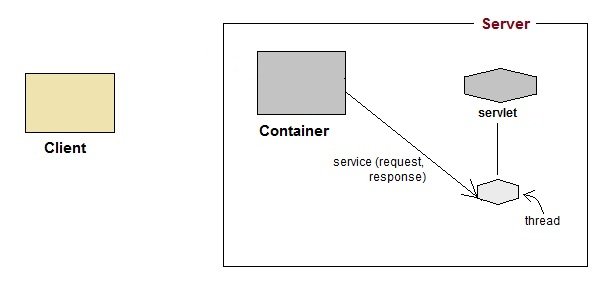
1. User sends request for a servlet by clicking a link that has URL to a servlet.



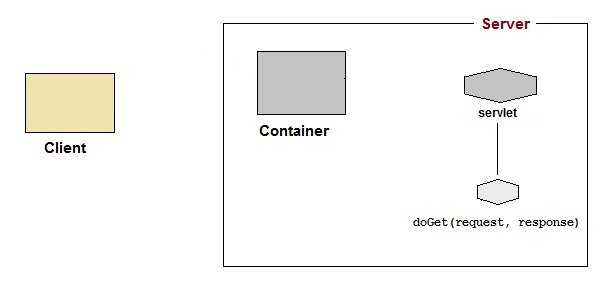
1. The container finds the servlet using **deployment descriptor** and creates two objects :
   1. [**HttpServletRequest**](https://www.studytonight.com/servlet/servlet-request.php)
   2. [**HttpServletResponse**](https://www.studytonight.com/servlet/servlet-response.php)



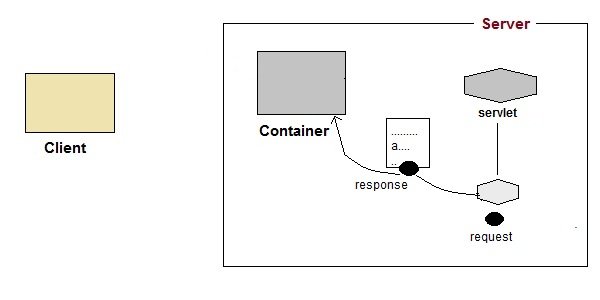
1. Then the container creates or allocates a thread for that request and calls the Servlet's service() method and passes the **request, response** objects as arguments.



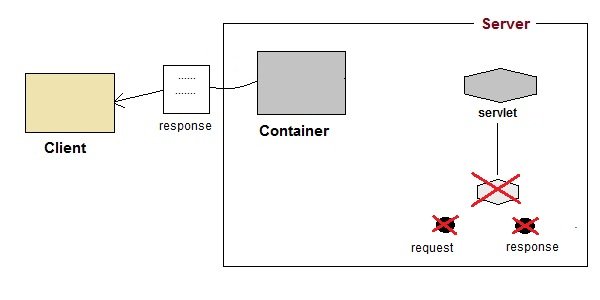
1. The service() method, then decides which servlet method, doGet() or doPost() to call, based on **HTTP Request Method**(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the service() will call Servlet's doGet() method.



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.

