# *servlet*

# What is the servlet?

Servlet is the technology that is used to create the dynamic web pages. Pre-requisites

Apache tomcat server.

Eclipse for web JDK install Note-

1. Download the tomcat and install it or
2. Download the tomcat and keep in into C drive directly.

Difference between web server and application server.

|  |  |
| --- | --- |
| Web Server | Application Server |
| Web server contain the web or servlet container | Application server contain web and EJB container. |
| It contain JSP, Servlet, HTML file | It contain the JSP, Servlet, HTML, EJB and JMS |
| Apache tomcat | JBoss, Weblogic, webSphere |
| It is the light weight components. | It is the heavy weight components |

Why Server is needed?

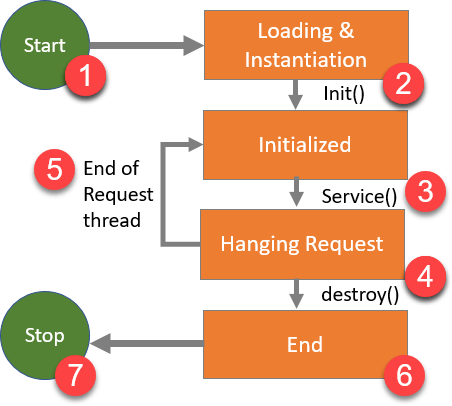
If you don’t use any server then whole project or web pages is stored into local hard disk. If someone wants to access the web site or project globally then how can access it? So every time he or she needs to use your laptop.

So to overcome this issue, server comes into the picture. Download and install the tomcat server.

1. Use of bin folder->to start and stop the server
2. Startup.bat file is used to start the server.

The web or servlet container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

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



As displayed in the above diagram, there are mainly three states of a servlet such as init, service and destroy.

1. Servlet class is loaded.

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

1. Servlet instance is created.

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

1. Init method is invoked.

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

public void init(ServletConfig config) throws ServletException

1. Service method is invoked.

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

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

1. Destroy method is invoked.

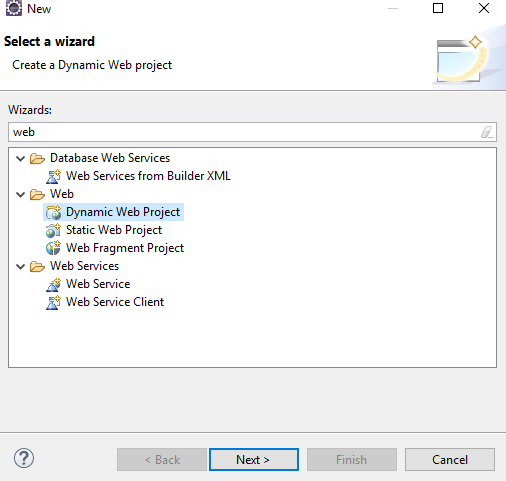
The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

# How to create the first servlet program.

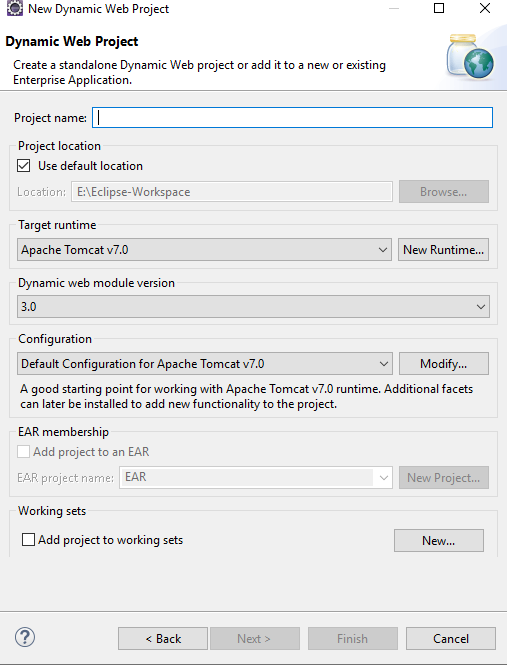
Open the Eclipse

Click on File icon-> New-> other-> Search Web

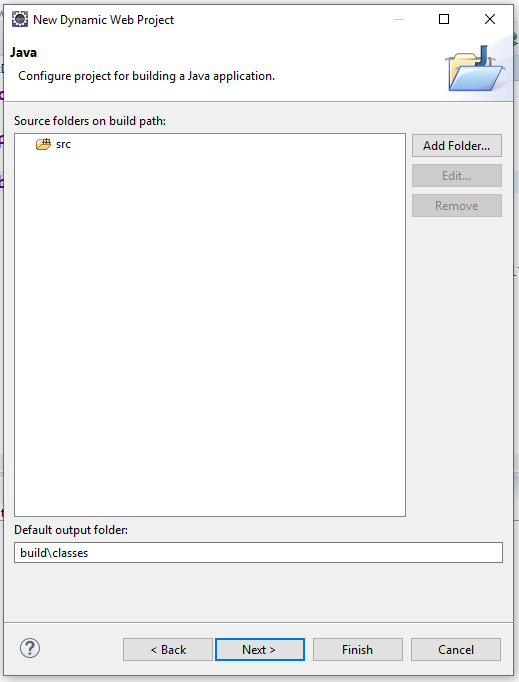


Select the Dynamic Web Project

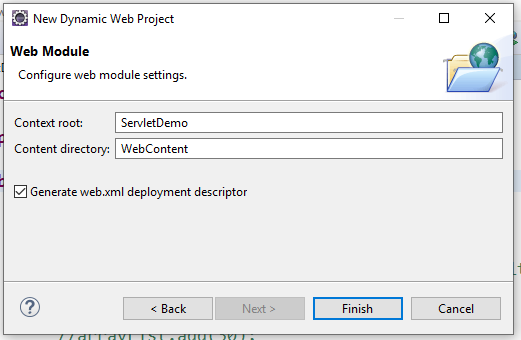
After new window will be displayed as below



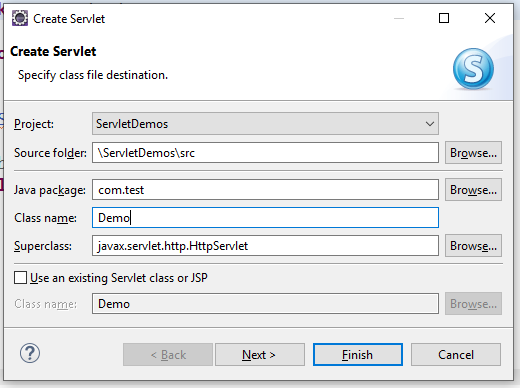
Write the Project name as FirstServlet and then click on Next button



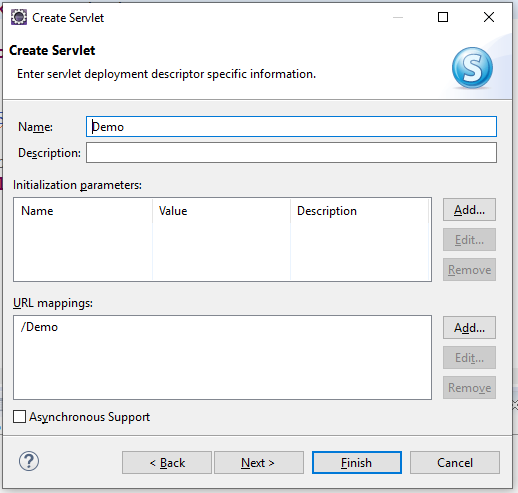
Click on Next button



Select the checkbox as clickable then click on finish button. Right click on Project->New->Select Servlet->



Click on Next Button



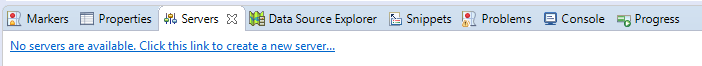
Click on Next Button

Uncheck the all and check only service option and then click on finish.

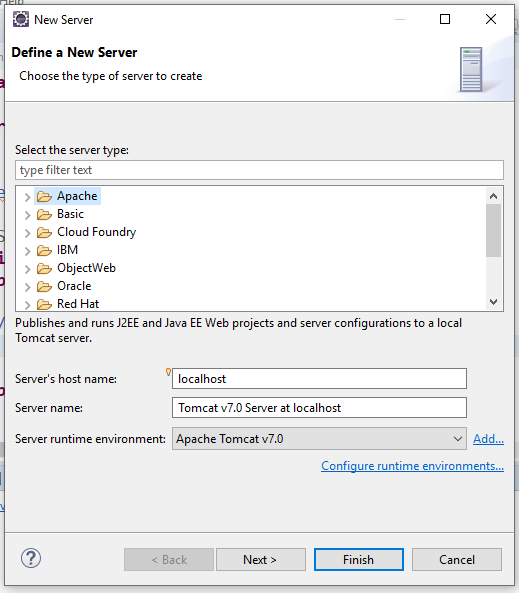
Right click on Project->New->HTML File-> enter the name for html file as simple.html

Add Server to run the web application.

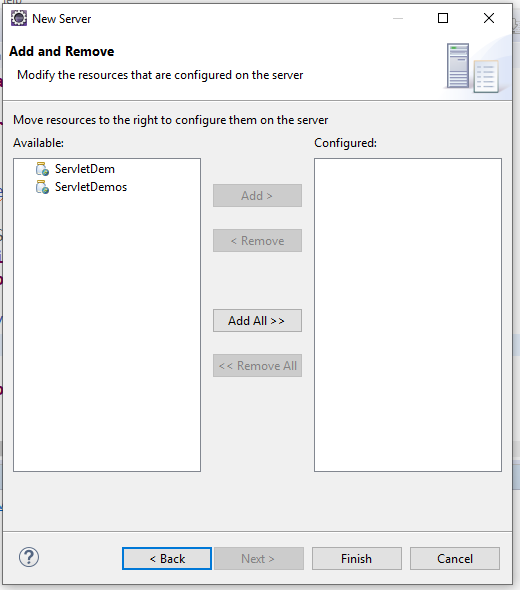
Go to Server tab->



Click this link to create the new server.



Select Tomcat v7.0 Server from Apache list and click on Next button.

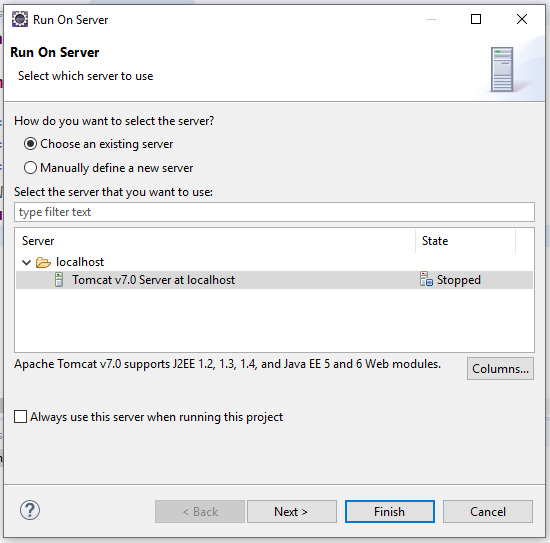


Select the project from the list and click on Add button then will see Project into configured

Then click on finish button.

How to run the Project.

Right click on Project-> Run as-> Run on Server. See the screen

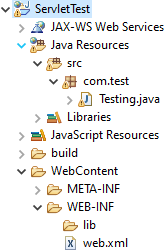


Click on Next Button-> Finish button.

Then will displayed the message on console.

Feb 28, 2022 11:47:44 AM org.apache.catalina.startup.Catalina start INFO: Server startup in 11081 ms

It means server is started. Project Structure



**package** com.test;

**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;

/\*\*

\* Servlet implementation class Testing

\*/

**public class** Testing **extends** HttpServlet {

**private static final long *serialVersionUID*** = 1L;

**public** Testing() {

System.***out***.println("in testing constructor");

}

@Override

**public void** destroy() {

System.***out***.println("in destroy method");

}

@Override

**public void** init() {

System.***out***.println("in init method");

}

**protected void** service(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException { System.***out***.println("in service method"); response.setContentType("text/html");

PrintWriter out=response.getWriter();

out.print("<h1>This is the first

servlet </h1>");

}

}

Web.xml

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

<web-app xmlns:xsi=*"*[*http://www.w3.org/2001/XMLSchema-*](http://www.w3.org/2001/XMLSchema-) *instance"*

xmlns=*"*[*http://java.sun.com/xml/ns/javaee"*](http://java.sun.com/xml/ns/javaee)xsi:schemaLocation=*"*[*http://java.sun.com/xml/ns/ja*](http://java.sun.com/xml/ns/ja)

*vaee* [*http://java.sun.com/xml/ns/javaee/web-*](http://java.sun.com/xml/ns/javaee/web-) *app\_3\_0.xsd"*

id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>ServletTest</display-name>

<servlet>

<servlet-name>test</servlet-name>

<servlet-class>com.test.Testing</servlet-

class>

</servlet>

<servlet-mapping>

<servlet-name>test</servlet-name>

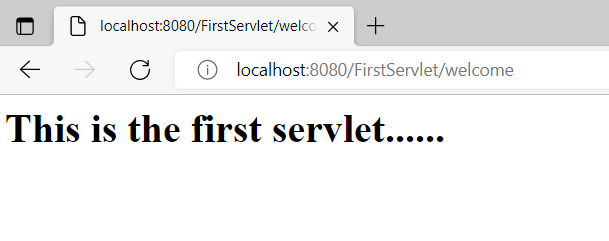
<url-pattern>/welcome</url-pattern>

</servlet-mapping>

</web-app>

URL- http://localhost:8080/ServletTest/welcome

Output-



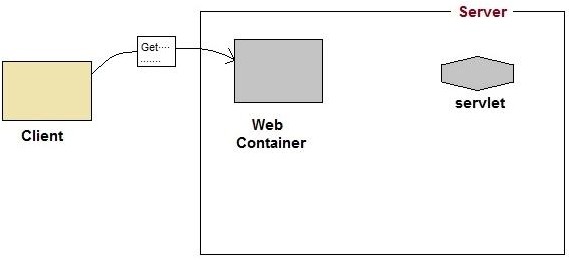
**This is the first servlet......**

***How Servlet internally works?***

The web container is responsible to handle the request. Let's see how it handles the request.

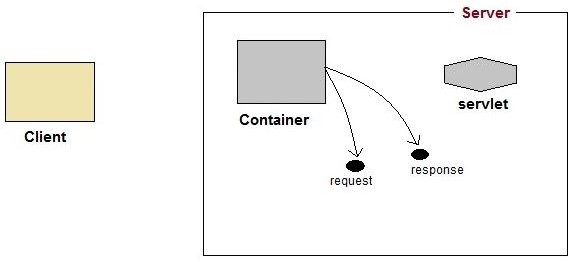
* maps the request with the servlet in the web.xml file.
* creates request and response objects for this request
* calls the service method on the thread
* The public service method internally calls the protected service method
* The protected service method calls the doGet method depending on the type of request.
* The doGet method generates the response and it is passed to the client.
* After sending the response, the web container deletes the request and response objects. The thread is contained in the thread pool or deleted depends on the server implementation.

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

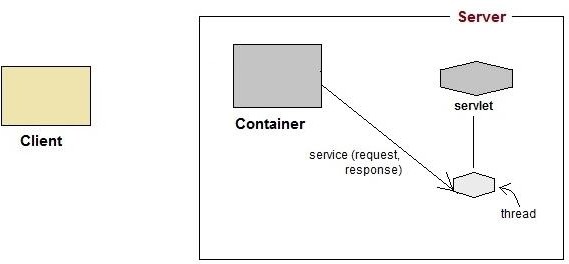


The container finds the servlet using deployment descriptor and creates two objects:

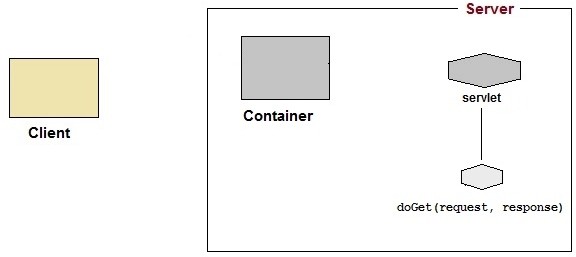
* 1. HttpServletRequest
  2. HttpServletResponse



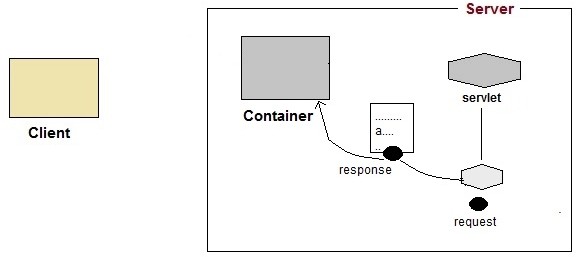
1. Then the container creates or allocates a thread for that request and calls the Servlet's service() method and passes the request, response objects as arguments.



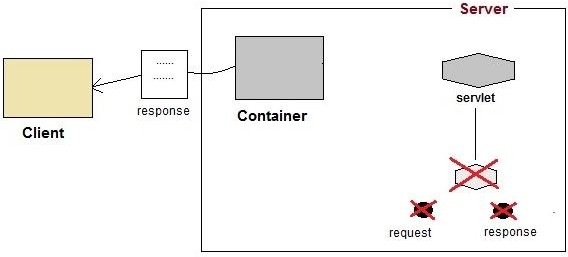
1. The service() method, then decides which servlet method, doGet() or doPost() to call, based on HTTP Request Method(Get, Post etc) sent by the client. Suppose the client sent an HTTP GET request, so the service() will call Servlet's doGet() method.



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



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



HTML + Servlet Example-

How to pass the data from HTML page to Servlet Page. Register.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

# <!DOCTYPE html>

<html>

# <head>

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

# <title>Insert title here</title>

</head>

# <body>

<form action=*"test"* method=*"post"*> Username<input type=*"text"* name=*"username"*/> Password<input type=*"password"*name=*"password"*/>

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

</form>

# </body>

</html>

# RegistrationServlet.java

**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;

/\*\*

\* Servlet implementation class Testing

\*/

@WebServlet("/Testing")

**public** **class** Testing **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#service(HttpServletRequest request, HttpServletResponse response)

\*/

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

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

System.***out***.println("username >> " + username);

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

System.***out***.println("password >> " + password);

}

}

Note- Form action name and WebServlet(“”) name should be same. http://localhost:8080/ServletExample/Register.jsp

then username and password displayed on screen.