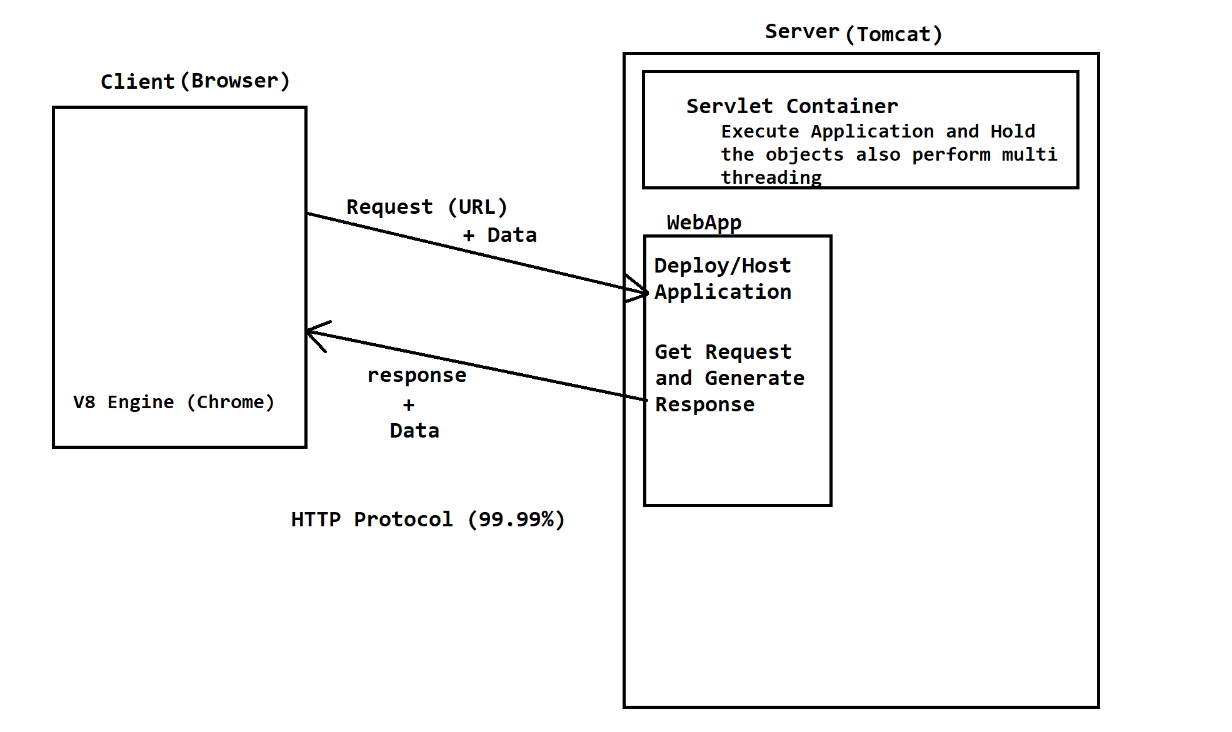
Advance Java

1. JEE also known as advance Java
2. You can start developing a Dynamic Web Applications.
3. **Servlet and JSP** are the 2 main technologies to work on web application.
4. Web applications are also known as client server application.

**Web Application**

1. Web Applications are also known as web sites in layman term.
2. There are 2 types of web applications.
   1. **Static web application**
      1. The content of the page will be same for every user.
      2. This application can be develop mostly using the Client-side technologies.
      3. Technical Stack: HTML, CSS, JS, and any front end framework
      4. Applications like Informative web sites.
   2. **Dynamic Web application**
      1. The content of the page will be different by user to user.
      2. These applications are developed using client and server-side technologies.
      3. Technical Stack: HTML, CSS, JS, Servlet, JSP
      4. Applications like Social Networking sites, Shopping sites, Banking Sites are the examples of dynamic web applications.

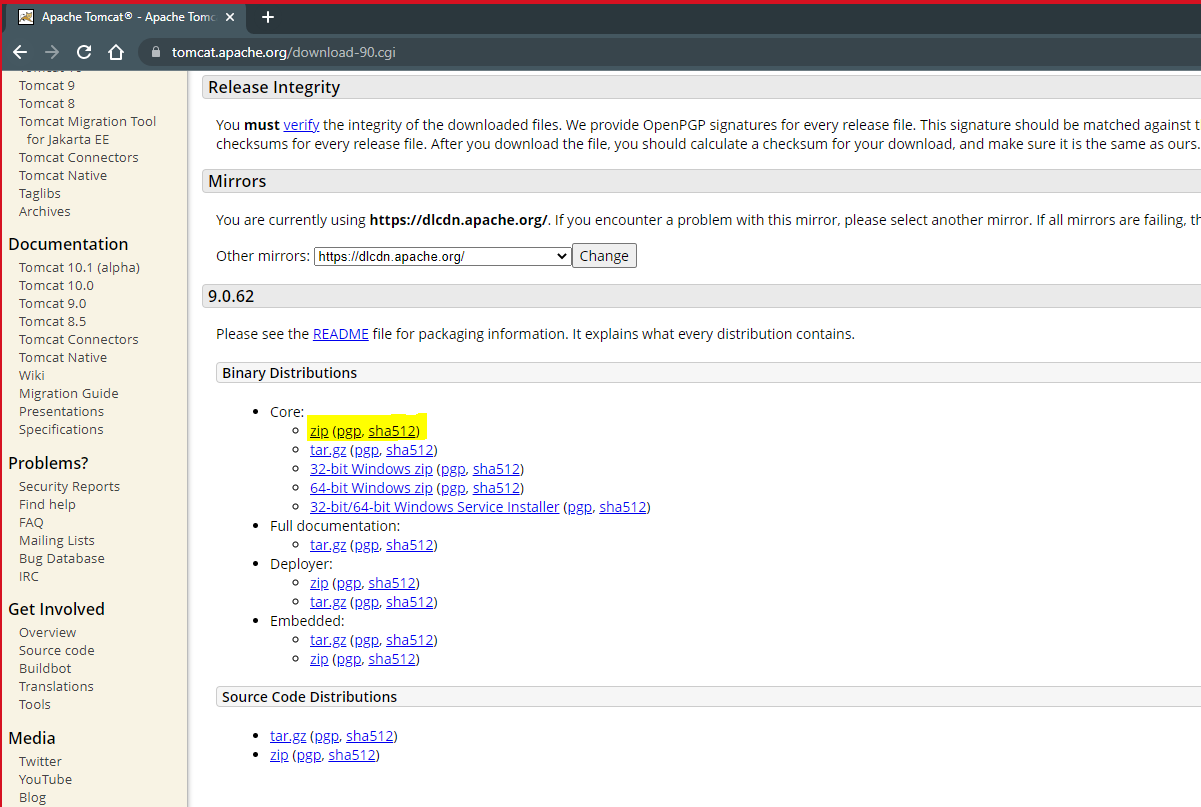
**Client Server Application**



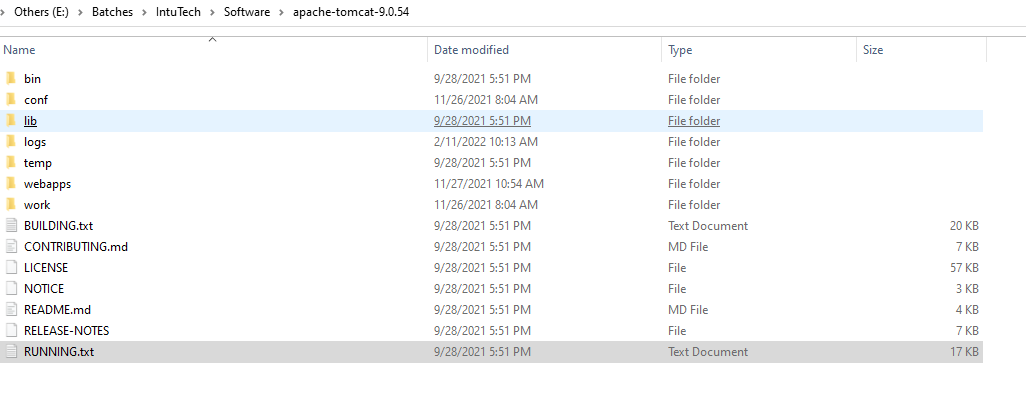
**Server Setup**

1. Download a Tomcat Server (Zip file)

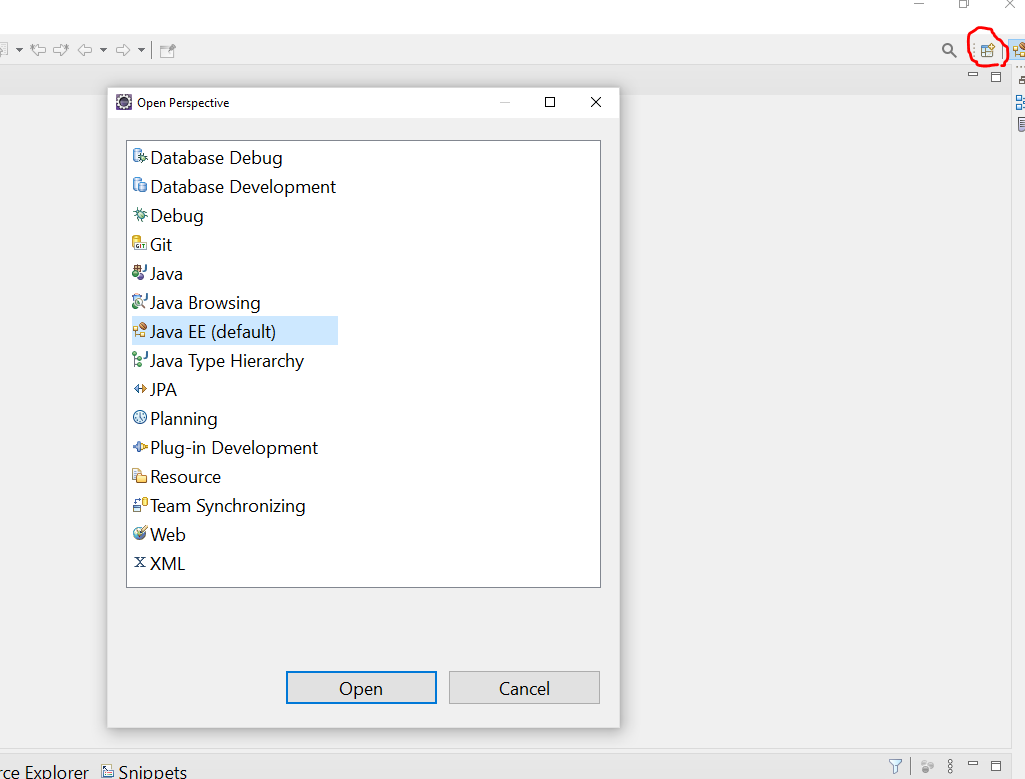
<https://tomcat.apache.org/download-90.cgi>



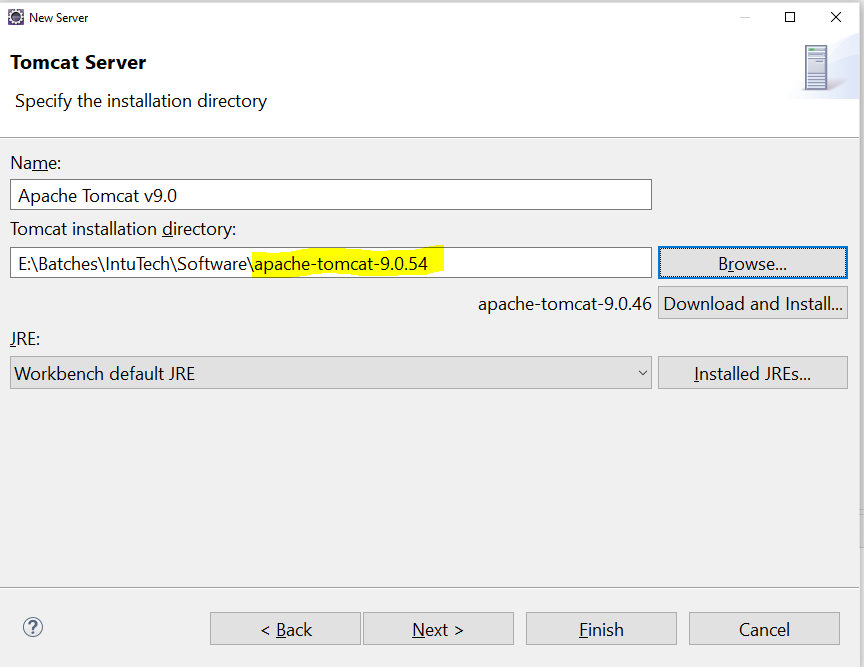
1. Extract The ZIP file into a specific folder.



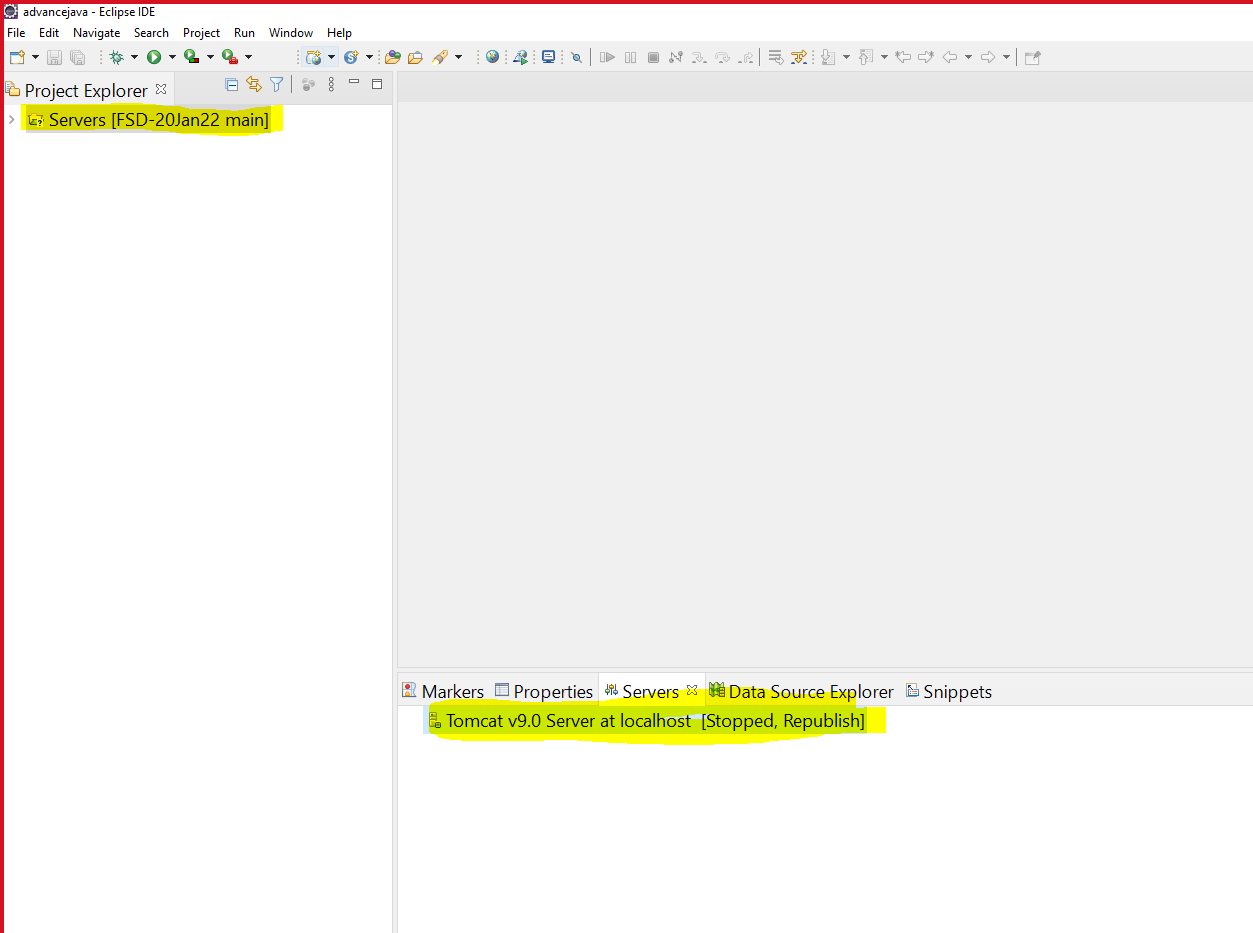
1. Configure server into eclipse.
   1. Click On “Open Perspective” option in the right side top of the Eclipse



* 1. Select a “Java EE” perspective and click on “Open” button.
  2. Click On the “Servers” tab at the bottom section of the eclipse
  3. Click on the link shown inside server tab.
  4. Expand “Apache” option from the list on the new window
  5. Select a Tomcat version you downloaded and click on “Next” Button.
  6. Provide the Installation directory (Browse a Location where you extracted a tomcat server and select a root/parent folder path of bin, conf, lib, webapp folder).



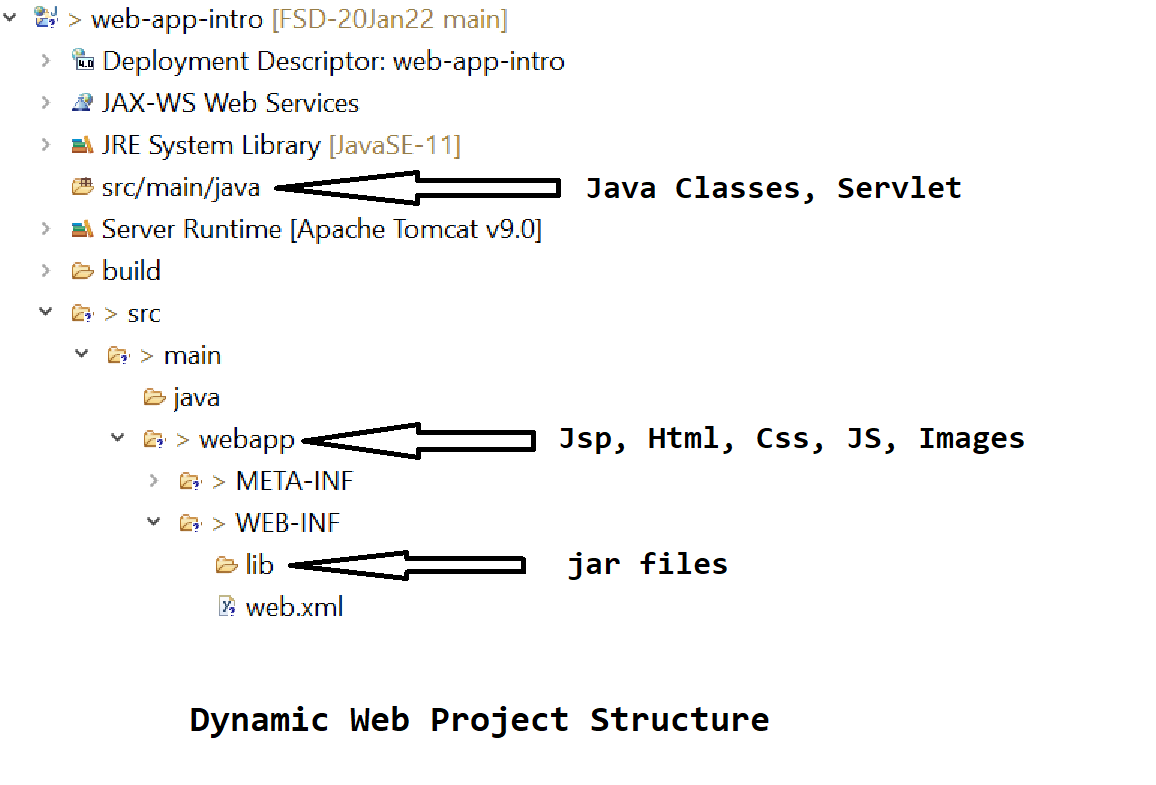
* 1. Click on “Finish” button



* 1. Right click on Apache tomcat server and click on “Start” Button

**Steps to create Dynamic Web Application**

1. Go to “File” Menu -> “New” Option -> Select “Dynamic Web Project”
2. Provide Project Name on the new window. Make sure that the target runtime is selected and it must not be “NONE”.
3. Click On “Next” Button -> “Next” Button
4. Make Sure that check box for Deployment Descriptor is selected.
5. Click on “Finish” button



**Servlet**

1. Servlets is a technology; servlet pages are use to create a dynamic web page.
2. Servlet is a java class without main method.
3. Servlets are executes at server side.
4. Servlet are use to get the request, process the request data and generate the response.
5. The file extension of the servlet is .java
6. The Object of servlet and the servlet life cycle will be maintained by servlet container.
7. On Servlet you can use the HTML code along with Java code. This HTML code has to write inside Java code (HTML in Java)
8. Every thing in the Web Application has to execute using a URL. That’s why every servlet must have a URL.

**How to create Servlet**

1. You can create a servlet by 2 ways
   1. Manually by creating a class and converting it into servlet.
      1. First create a Java Class. Do not include Main method.
      2. Convert java class into Servlet any one of the option as follows

Implement Servlet Interface

Extends GenericServlet abstract class

Extends HttpServlet Abstract Class

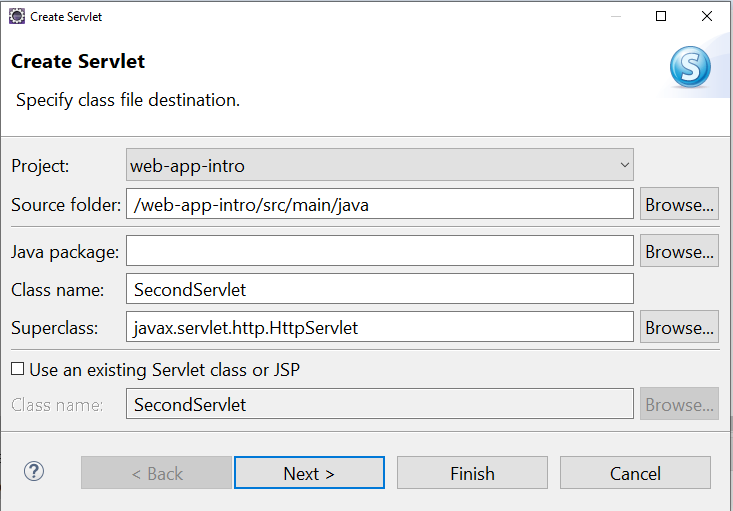
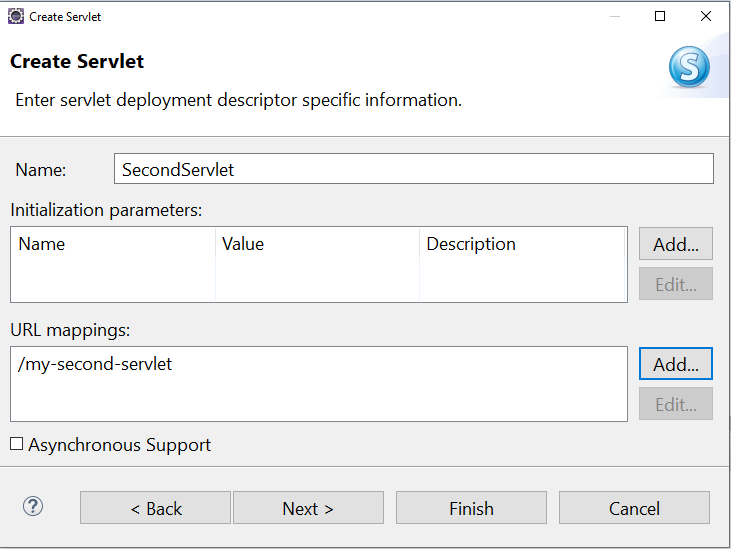
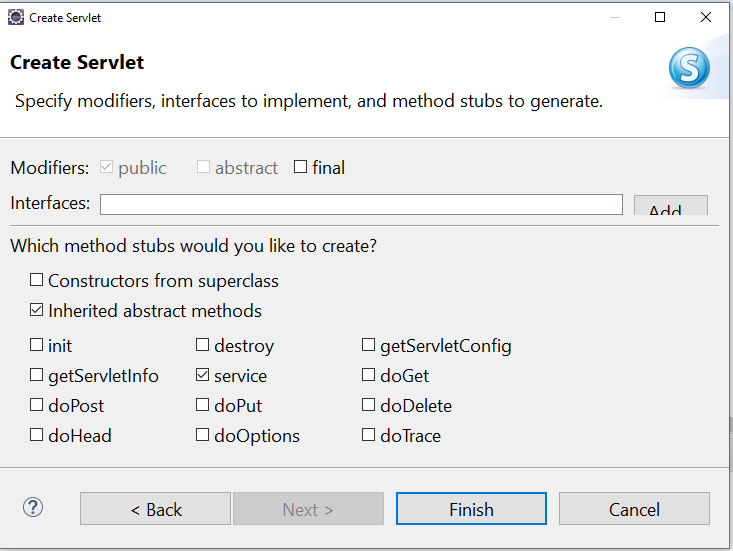
* + 1. Override the service method from the Parent class.

@Override

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

System.***out***.println("This is the service method...");

}

* + 1. Provide the URL for the servlet.
  1. Can use an eclipse option to create servlet
     1. Right Click on “src/main/java” -> “New” Option -> Select “Servlet”
     2. 
     3. 
     4. 

**Provide Servlet URL**

* + - 1. There are 2 option to provide servlet URL
         1. XML

The URL will be provided inside the web.xml file.

You have to use tags inside XML file.

<servlet>

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

<servlet-class>FirstServlet</servlet-class> <!-- Java Class name without file extension -->

</servlet>

<servlet-mapping>

<servlet-name>first</servlet-name> <!-- It must be same as a Servlet tag name -->

<url-pattern>/my-first-servlet</url-pattern> <!-- It Must be start with '/' -->

</servlet-mapping>

* + - * 1. Annotation

To Provide URL by annotation, you have to use following annotation on the servlet class

@WebServlet("/<URL>")

**To Run Servlet:**

1. Right Click On Servlet -> “Run As” option -> click on “Run On Server” option

**Generate Response from Servlet**

1. Set the content type of the response
   1. The type of response which you wanted to return to the user.
   2. This is also known as MIME type (MIME type of use for both request and response)

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types>

* 1. To Set the response type you can use the following method

response.setContentType("text/html");

1. Get the Object to write response
   1. To write a response and return to the user you have to get the object of PrintWriter

PrintWriter write = response.getWriter();

* 1. The Text or HTML tag write inside the print metho will return as a response to the user.

write.print("<h1>Welcome To Servlet</h1>");

Parameter:

1. Parameter is a user details which is added inside URL after the ‘?’
2. Parameters are always pass from client to server.
3. Each parameter has 2 parts Name=Value
4. There can be multiple parameter in a request. Each parameter separated by ‘&’
5. Parameters are always in String format no other Data type will be used for parameter.
6. There are 2 ways to send parameter to the server side
7. From the URL after ‘?’, here the values will be visible in URL
8. From the Form Data, here the values will not be visible inside URL

Task-1

Page-1



Page-2 (Get the value entered b user into the page-1 test box and display it on the page-2)



**Request Redirection**

1. If you want to redirect user from one page to another without any user action then you can use this redirection techniques.
2. There are two redirection techniques
   1. **Request Dispatcher**
      1. Is use to redirect user from one page to another without any user action.
      2. To go from one page to another no new request will ne generated here, it will use the same request to go from one page to another.
      3. Because the same request is forwarding on the next page the data from the request is also available on the next page.
      4. To use this technique, you have to user RequestDispatcher interface
      5. In this interface you will get 2 methods
         1. Forward: is use to send request to a next page.
         2. Include: To get the content of next page into current page.
      6. Syntax:

**RequestDispatcher object = request.getRequestDispatcher(“<URL>”)**

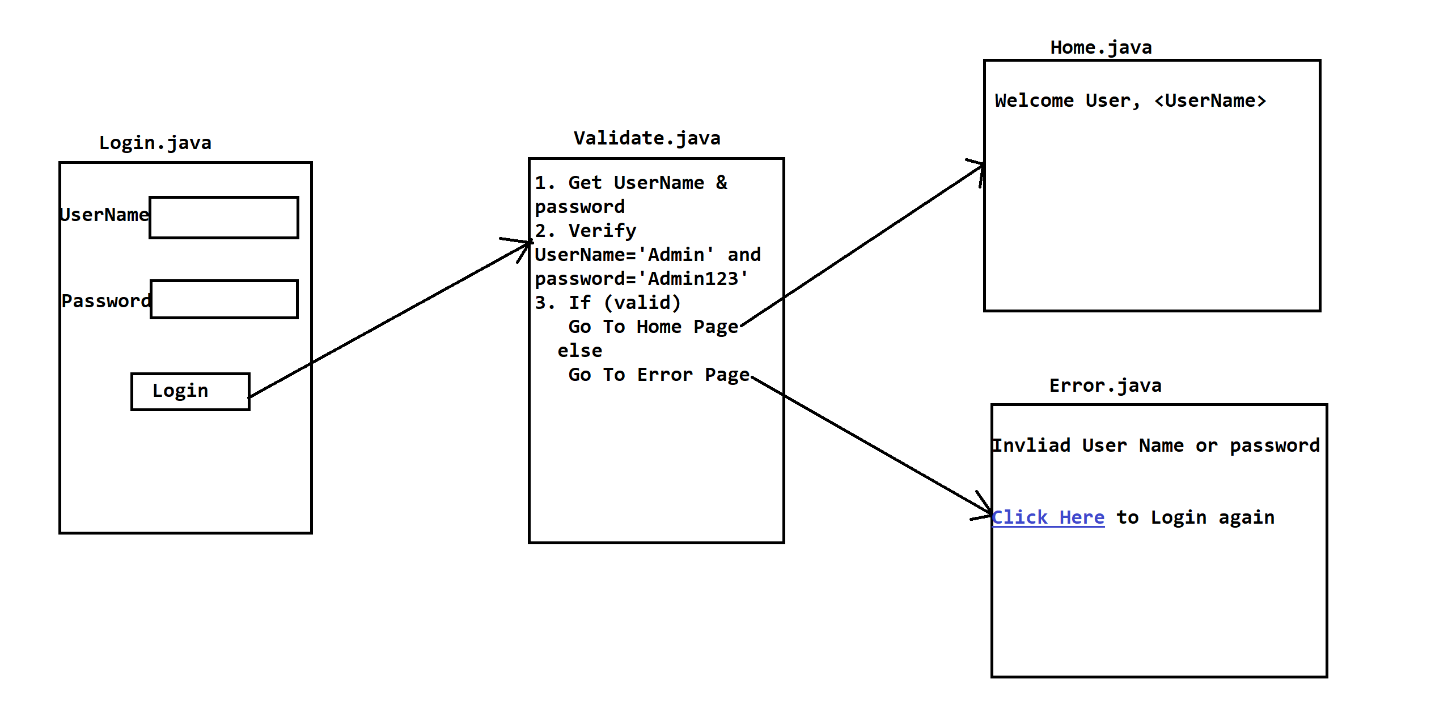
Object.forward(request,response);

Object.include(request,response);

* 1. **Send Redirect** 
     1. Is use to redirect user from one page to another without any user action.
     2. TO go from one page to another new request will be generated and the old request will get destroyed.
     3. So, the data from the old request will not be available inside the new request.
     4. sendRedirect is a method. Which can be access from the response object.
     5. Syntax:

**response.sendRedirect(“<URL>”);**

**Task-1**

****