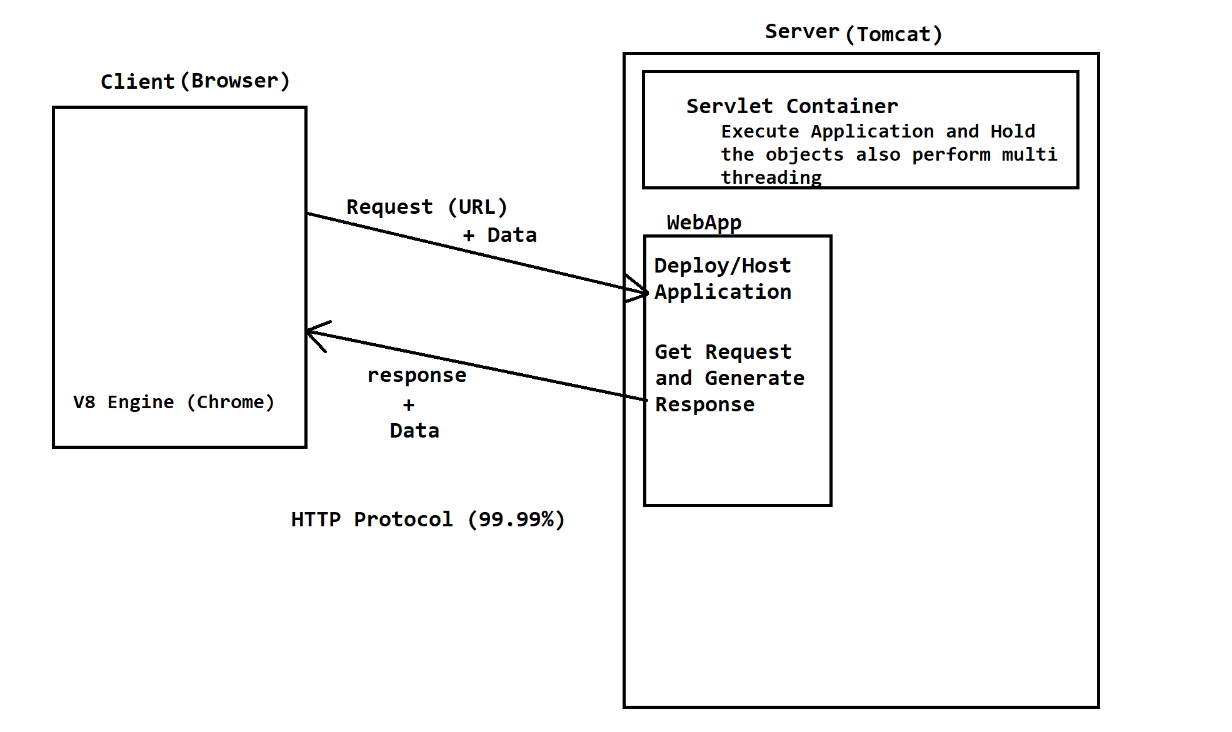
Advance Java

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.

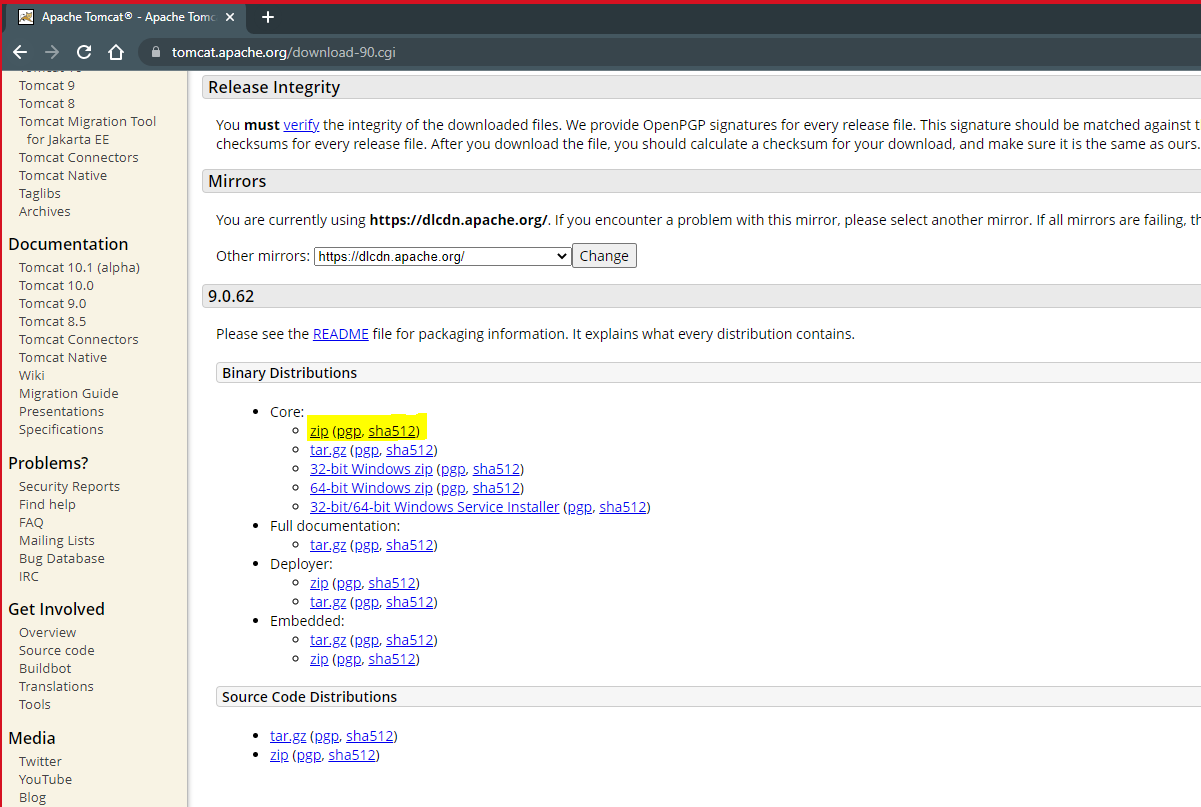
**Client Server Application**



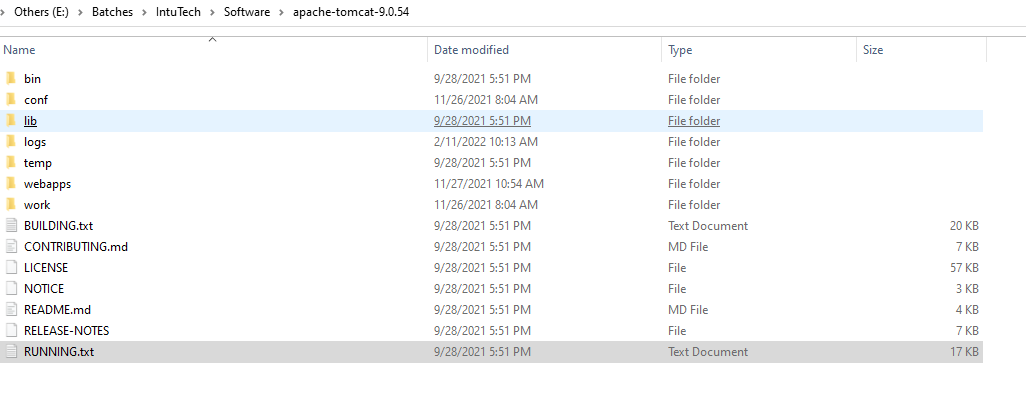
**Server Setup**

1. Download a Tomcat Server (Zip file)

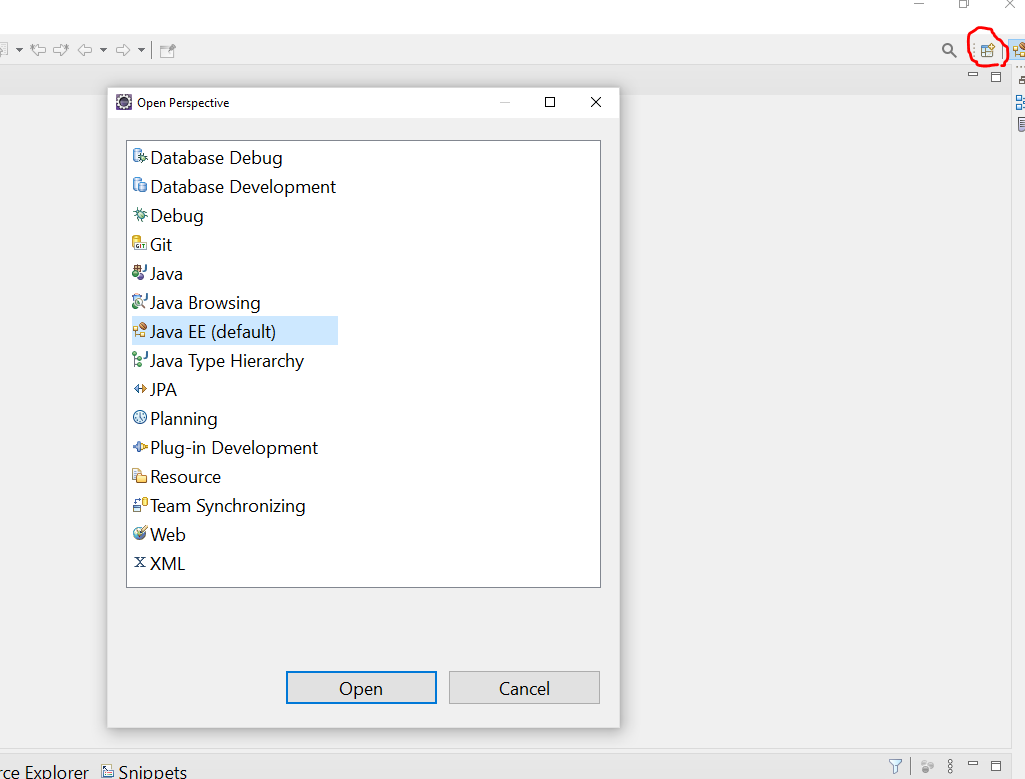
<https://tomcat.apache.org/download-90.cgi>



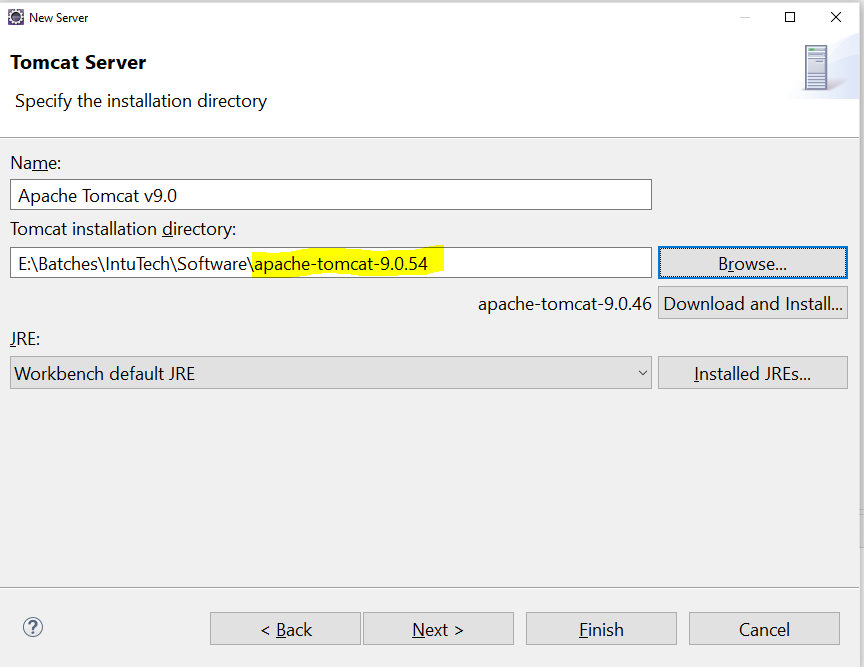
1. Extract The ZIP file into a specific folder.



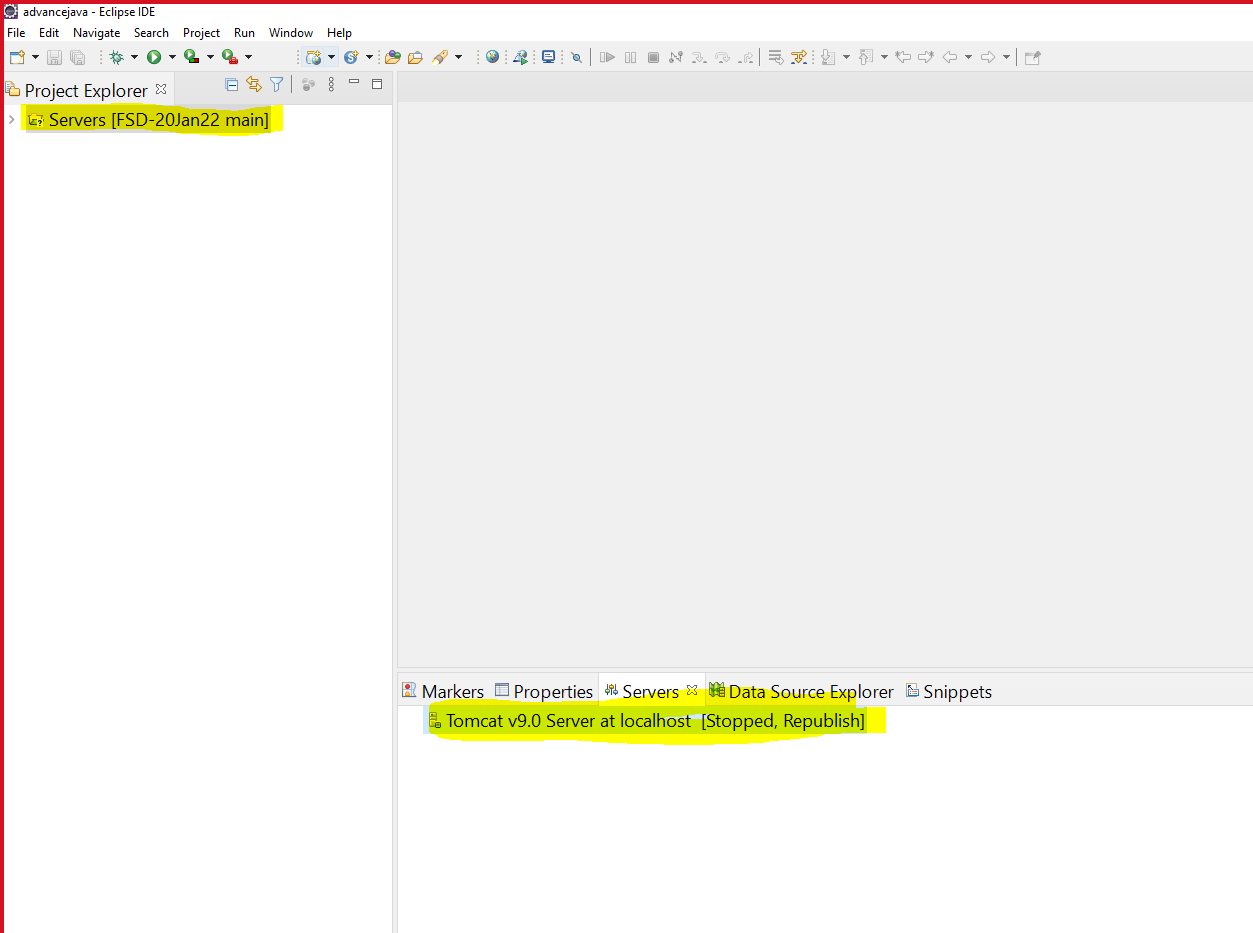
1. Configure server into eclipse.
   1. Click On “Open Perspective” option in the right side top of the Eclipse



* 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).



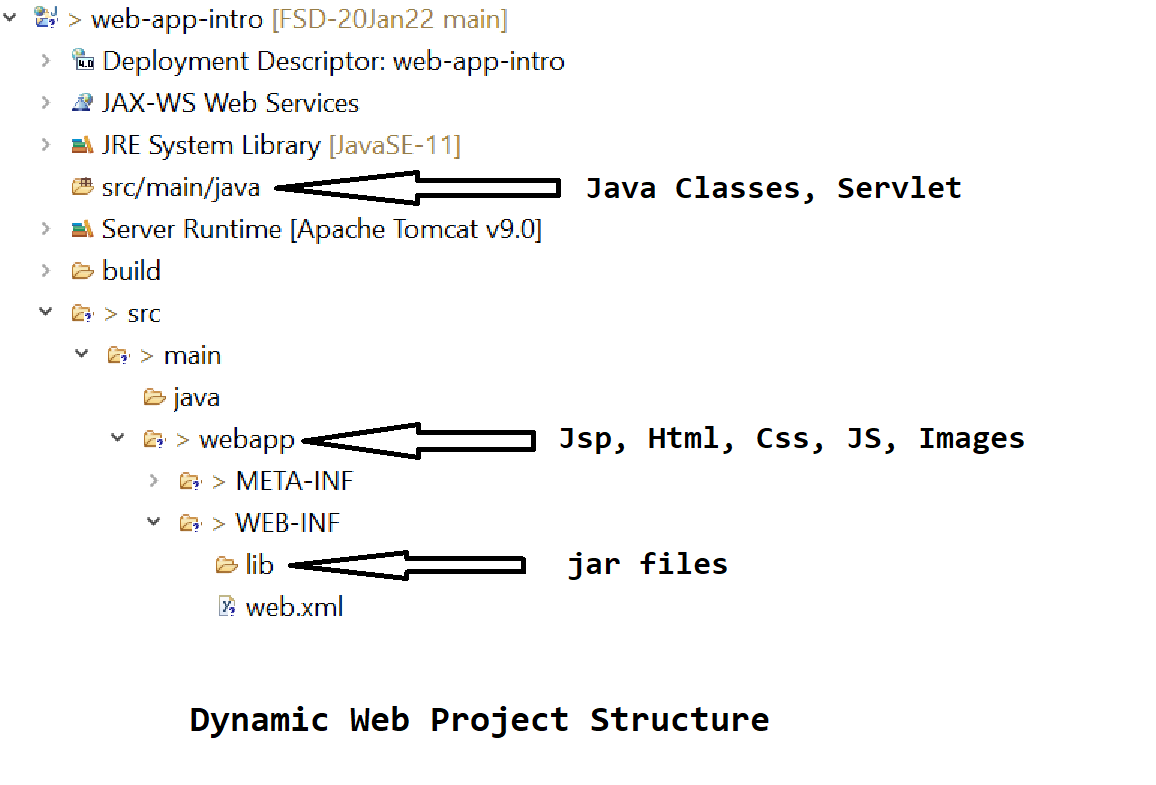
* 1. Click on “Finish” button



* 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

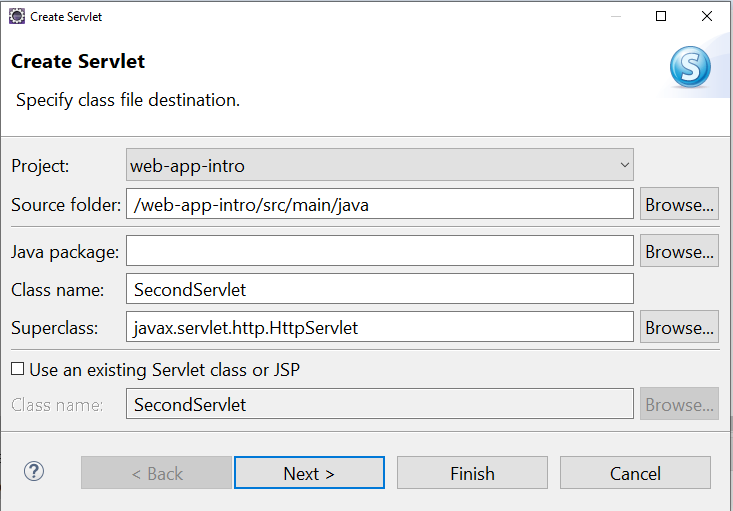
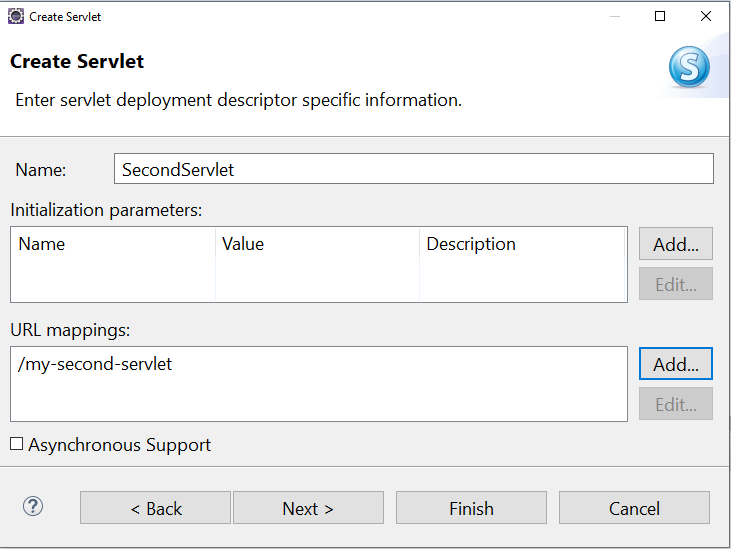
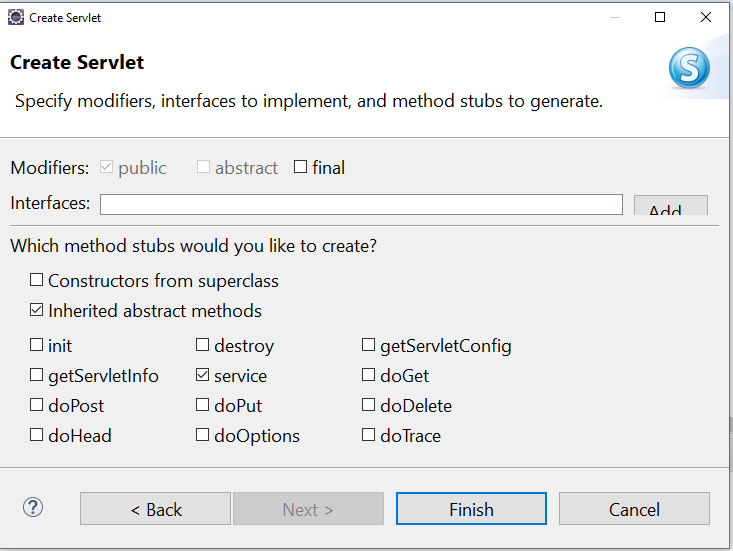
* + 1. Override the service method from the Parent class.

@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>”)**

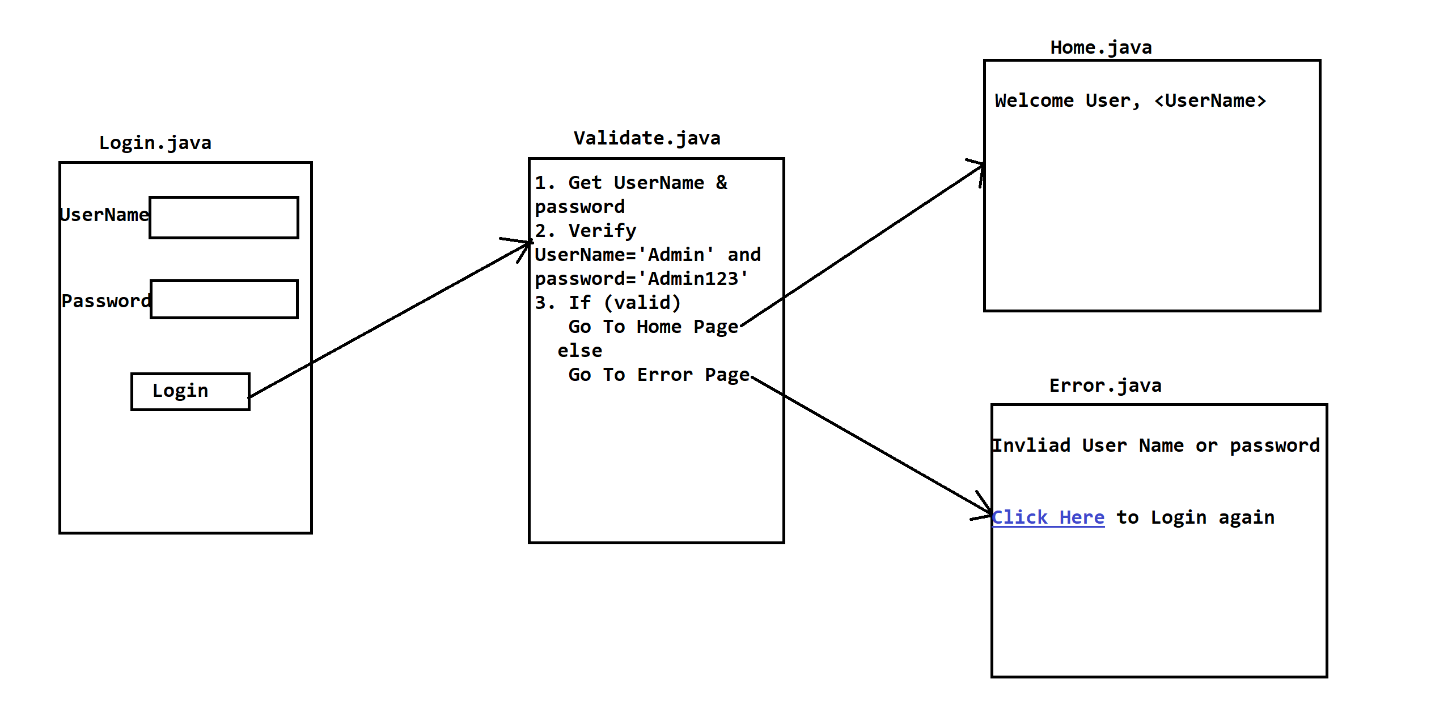
Object.forward(request,response);

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**

****

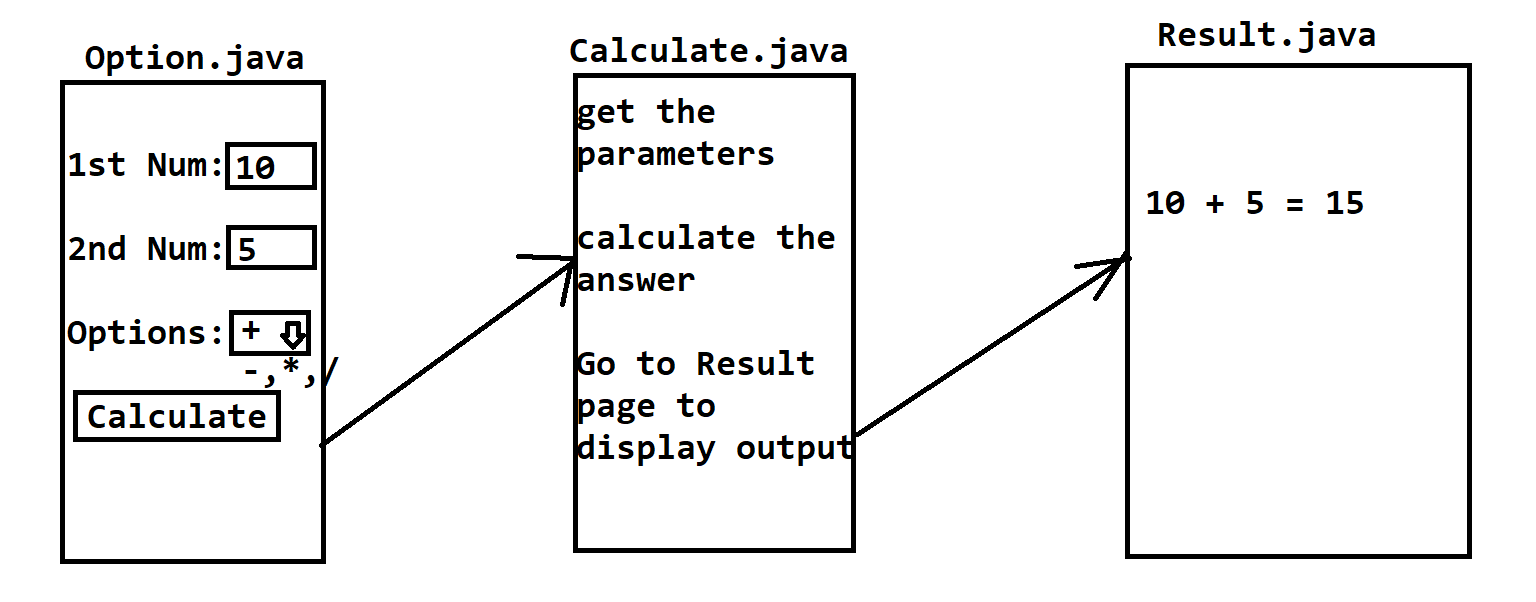
**Attribute:**

1. Attributes are used to pass the user information from one page to another.
2. This user details can be in the object format.
3. You can set and get the attribute manually.
4. It can be set and get inside request, session and application scope.
5. Syntax.

setAttribute(“Key/Name”, Object Value)

getAttribute(“Key/Name”) : Object;

Task-1



**Java Server Pages (JSP)**

1. JSP is the extended technology of the Servlet.
2. JSP is used to create the dynamic web pages.
3. JSP is mainly used for a designing of the web pages.
4. On the JSP page you can use the HTML, CSS, JavaScript and also java language.
5. The primary language is HTML here. Java can be written inside HTML (JAVA in HTML)
6. The file extension of the jsp page is .jsp.
7. No need to provide the URL for the JSP, by default the URL of the JSP is /PageName.jsp.
8. JSP internally gets converted into servlet before execution.

Scripting Element/Tag

1. Scripting are use to write java code on JSP page
2. The Different type of scripting tags are as follows
   1. Scriptlet Tag
      1. This tag is use to write a java code, this code is a local code which gets included inside the service method.
      2. The variable created inside this tag will be a local variable.
      3. You cannot create method or static variables using this tag.
      4. Syntax:

**<%**

**Java Code**

**%>**

* 1. Expression tag
     1. Using this tag, you can write a java expression which will execute and result will be display on the browser.
     2. This tag is use to display the values on browser.
     3. This is exactly same as the out.print
     4. The content written inside Expression tag will be added inside the out.print(<Expression>)
     5. Syntax:

**<%= Java Expression %>**

* 1. Declaration Tag
     1. This tag is use to create an instance method and variable.
     2. This is use to write a java code just like a scriptlet tag, but the java code written inside this tag will be added outside service method and inside class.
     3. Using this tag you can create your own methods inside generated servlet and also create static variables and method.
     4. Syntax:

**<%!**

**Java Code**

**%>**

1. You cannot use one tag into another tag. To achieve this, you use a break and continue option.

**Comments in JSP**

1. There are 2 types of comment tag in JSP
2. To Comment HTML code
   1. Syntax:

<!-- HTML code -->

1. To Comment Scritpting tag (Java Code)
   1. Syntax:

<%-- Scripting Tags --%>

**JSP implicit Object**

1. Implicit object is the object which are by default provided on every JSP page.
2. There are total 9 implicit objects.

|  |  |
| --- | --- |
| **Object Name** | **Class/Interface** |
| request | HttpServletRequest |
| response | HttpServletResponse |
| session | HttpSession |
| application | ServletContext |
| config | ServletConfig |
| out | JspWriter |
| exception | Throwable |
| pageContext | PageContext |
| page | this (current class object) |

1. These implicit objects are accessible only inside Scriptlet tag and Expression tag.

**Servlet Life cycle**

Servlet life cycle will be manage by the servlet container.

1. Init stage
   1. init(ServletConfig) method will be executed as a part of this stage.
   2. In side this method you can perform the initialization to carryout servlet functionality.
   3. This method gets invoked after the Object creation of servlet.
   4. There is only one object of servlet in an application, and hence this method invoked only once in a life cycle.
2. Service stage
   1. service(HttpServletRequest, HttpServletResponse) method will be executed as a part of this stage.
   2. This method is use to get the request, process the request and generate the response.
   3. The service will gets executed for every user request, hence this stage executes multiple time in a servlet life cycle.
3. Destroy stage
   1. destroy() method will be executes in this stage.
   2. This method is use to perform any functionality before closing/destroying the servlet object.
   3. This stage executes only one in a lifecycle.

**JSP Life Cycle**

JSP life cycle will be manage by the servlet container.

1. Translation Stage
   1. The Jsp page will gets converted into servlet.
   2. That is .jsp gets converted into .java
2. Compilation Stage
   1. The Generated servlet gets compiled in this stage.
   2. After compilation .class file will be created for the generated servlet java class.
3. Init stage
   1. \_jspInit() method will be executed as a part of this stage.
   2. In side this method you can perform the initialization to carryout servlet functionality.
   3. This method gets invoked after the Object creation of servlet.
   4. There is only one object of servlet in an application, and hence this method invoked only once in a life cycle.
4. Service stage
   1. \_jspService(HttpServletRequest, HttpServletResponse) method will be executed as a part of this stage.
   2. This method is use to get the request, process the request and generate the response.
   3. The service will gets executed for every user request, hence this stage executes multiple time in a servlet life cycle.
5. Destroy stage
   1. \_jspDestroy() method will be executes in this stage.
   2. This method is use to perform any functionality before closing/destroying the servlet object.
   3. This stage executes only one in a lifecycle.

**Session Management/ Session Tracking**

1. Session tracking/management is a process where user information will be store/maintain into a multiple request.
2. There are 4 types of session management technique
   1. Hidden Form field
      1. Hidden form field is a way to carry an old request information in to a new request which is generated by Form and submit button.
      2. In this type you can create a hidden fields and set the information which is to be include into a new request.
   2. URL rewriting
      1. This type is use when the new request generated using anchor tag or sendRedirect option.
      2. Using this technique, you can manually provide a parameter into a URL so that will be set inside a new request.
   3. Cookies
      1. Cookie is use to set the user information at client side (browser).
      2. Cookies will be created at server side and save at client side.
      3. Cookies will be included inside every request made for the server.
      4. Disadvantages of Cookies
         1. There is limit to store a cookie
         2. Can store only data in the string format.
         3. Cookies can be disabled at client.
   4. HttpSession
      1. HttpSession is use to save the information at server side.
      2. This is the temporary location which is provided to every user and it is always unique for every user.
      3. HttpSession internally used cookies to carry out the functionality
      4. Syntax to create session

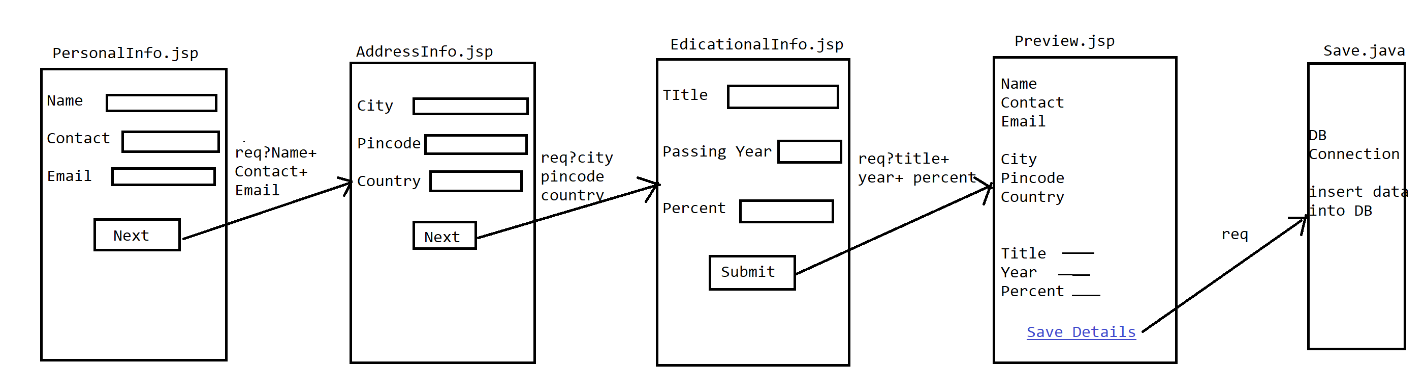
**HttpSession session = request.getSession();**

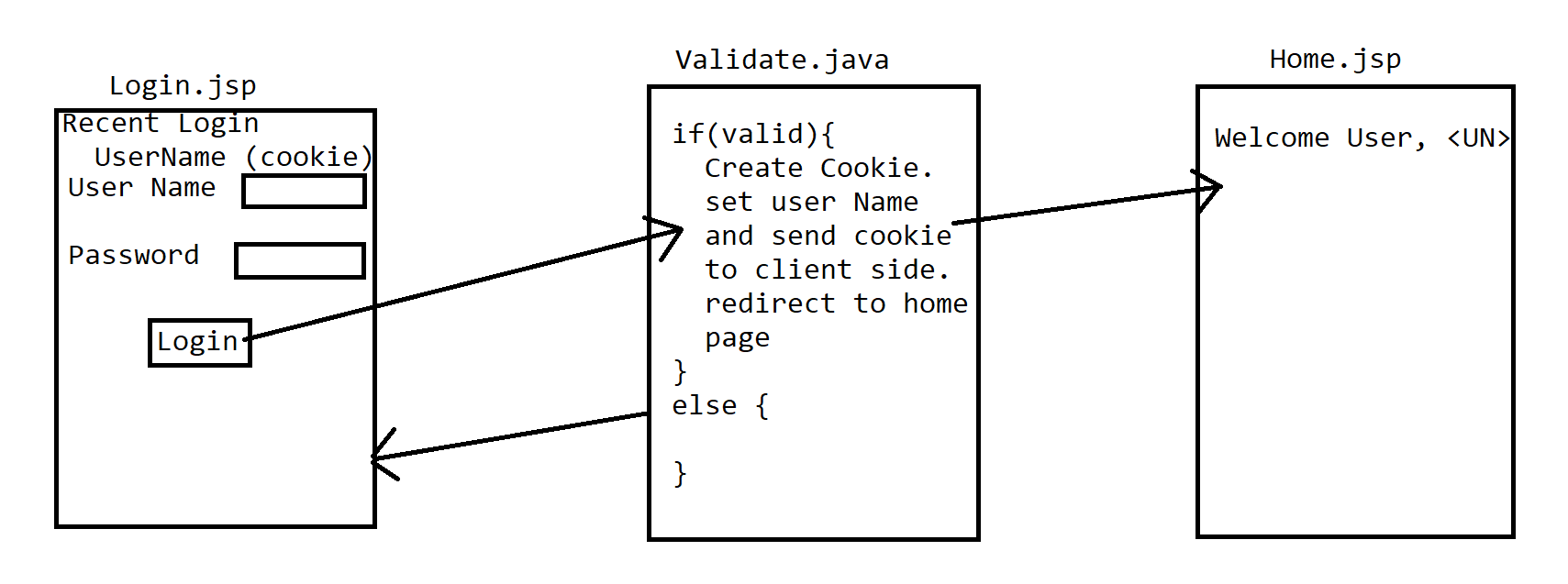
* + 1. Using above line you will get a session, this will not create a new session always.
    2. Internally it will first check if the session is present for the user or not, if session is not created then create a new session else return the existing session.
    3. Syntax to set the data into session

**session.setAttribute(“Key”, Object value);**

* + 1. Syntax to get the data from session

**session.getAttribute(“Key”) : Object;**

****

**Http Code :** [**https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client\_error\_responses**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client_error_responses) **Example For Cookie  
  
**

**Add External Jar file into Web Application**

1. Copy a jar file from the file system.
2. Paste jar file into a src/main/webapp/WEB-INF/lib into this folder

**Session Internal Working**

