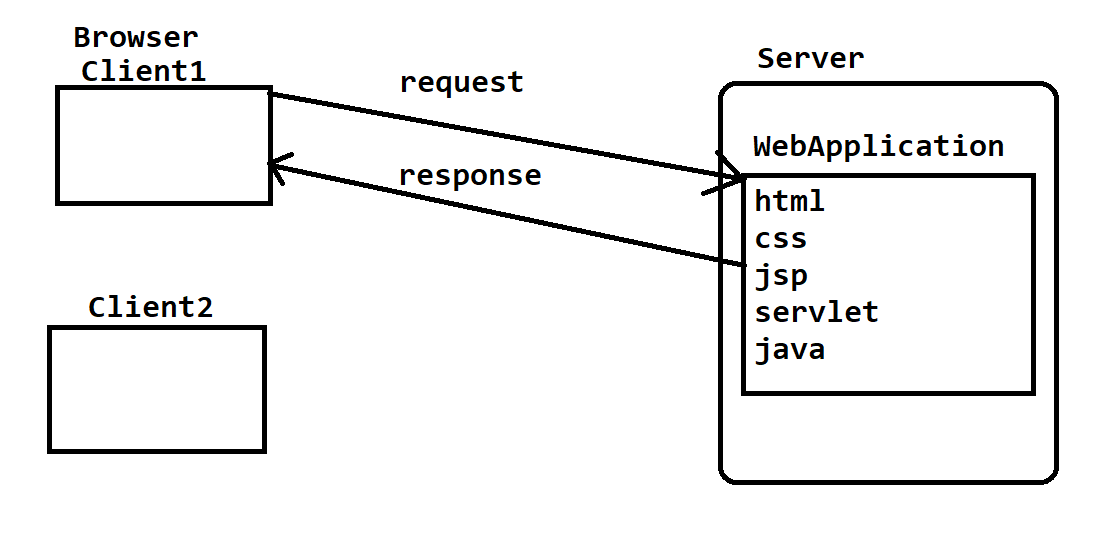
**Advance Java**

**Java Editions**

1. JSE (Java Standard Edition)
   1. Also known as Core Java
   2. It can be used to develop Console based application and Desktop application
2. JEE (Java Enterprise Edition)
   1. Also known as Advance Java. It is the combination of multiple technologies such as Servlet, JSP, JMS, JSR etc.
   2. It used to develop the Web application.
3. JME(Java Micro Edition)
   1. You can develop the embedded applications or mobile application.

**Web Application**

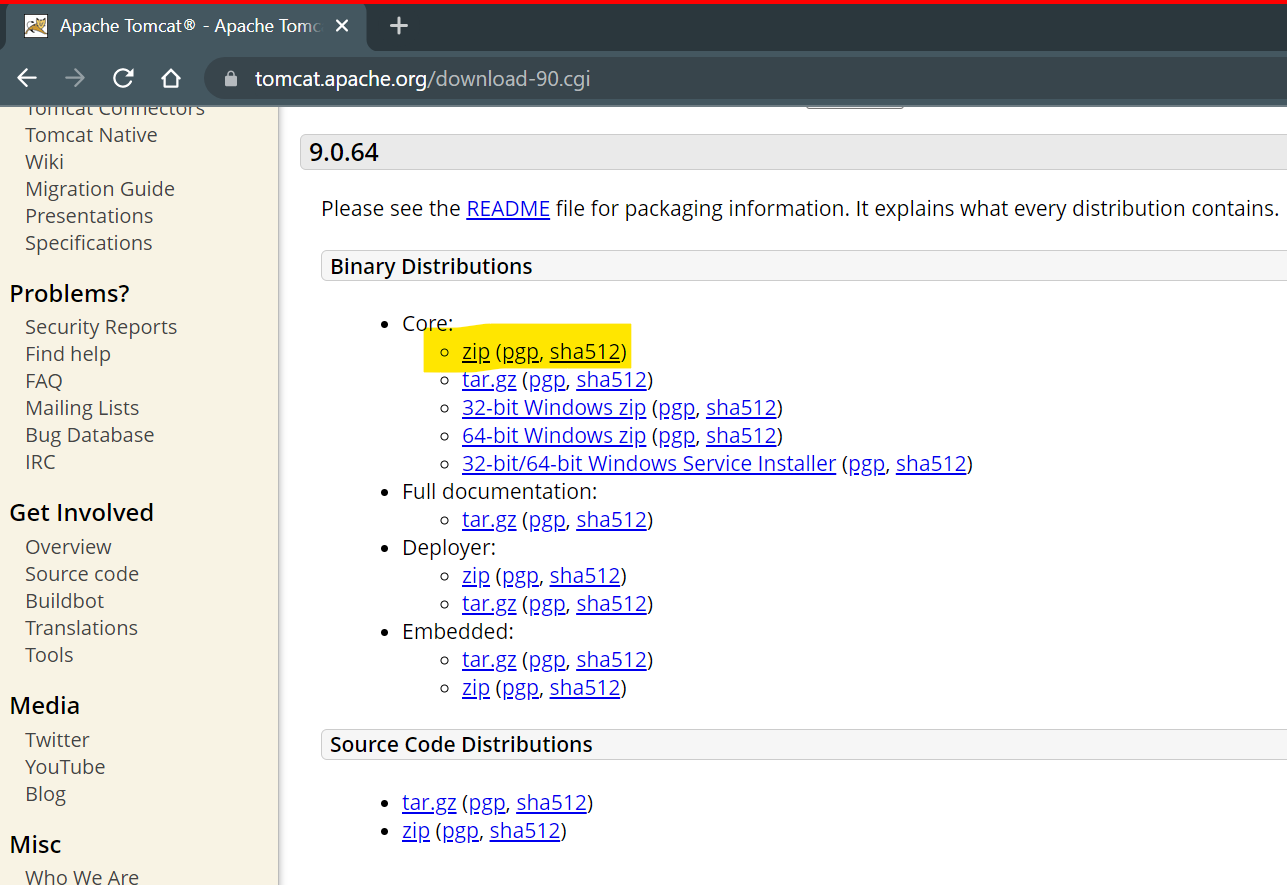
1. This application can be used by the end user from the network.
2. These applications are access from the browser and not need to do any setup on the client machine.
3. Are also known as client server application.
4. Where client is a browser which access the application using request and from application client will received the response.
5. There are 2 types of web application
   1. Static Web application.
      1. These web sites are the informative web sites where all the users will get the same page content.
      2. Technologies used are Html, CSS, JS, JQuery
   2. **Dynamic Web application.**
      1. In this type of application the content of the page will be changed for every user.
      2. Technologies used are Html, CSS, JS, JQuery, server-side technologies like JSP, Servlet, php etc.



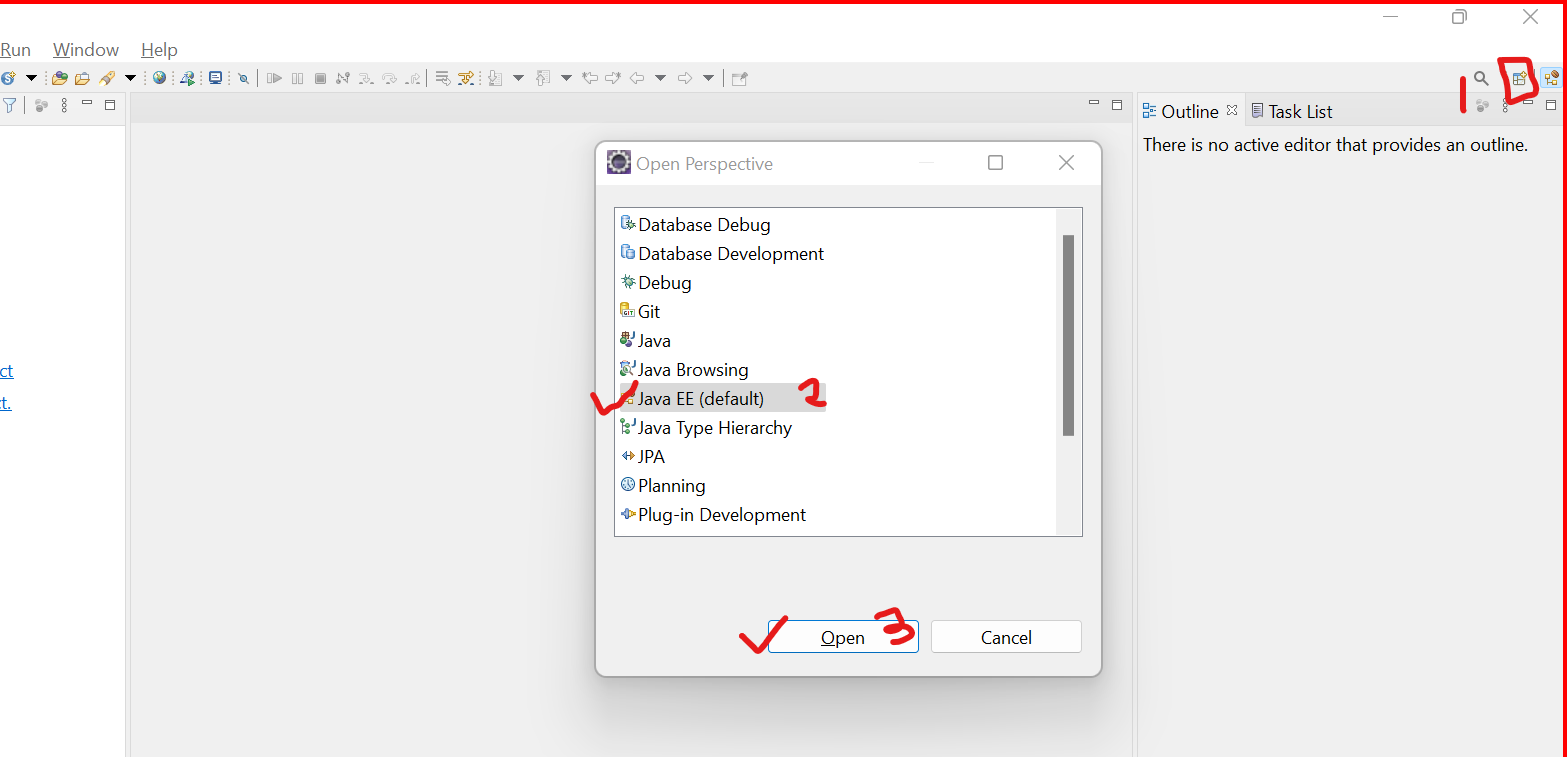
**Server Setup**

1. Download the Tomcat Server

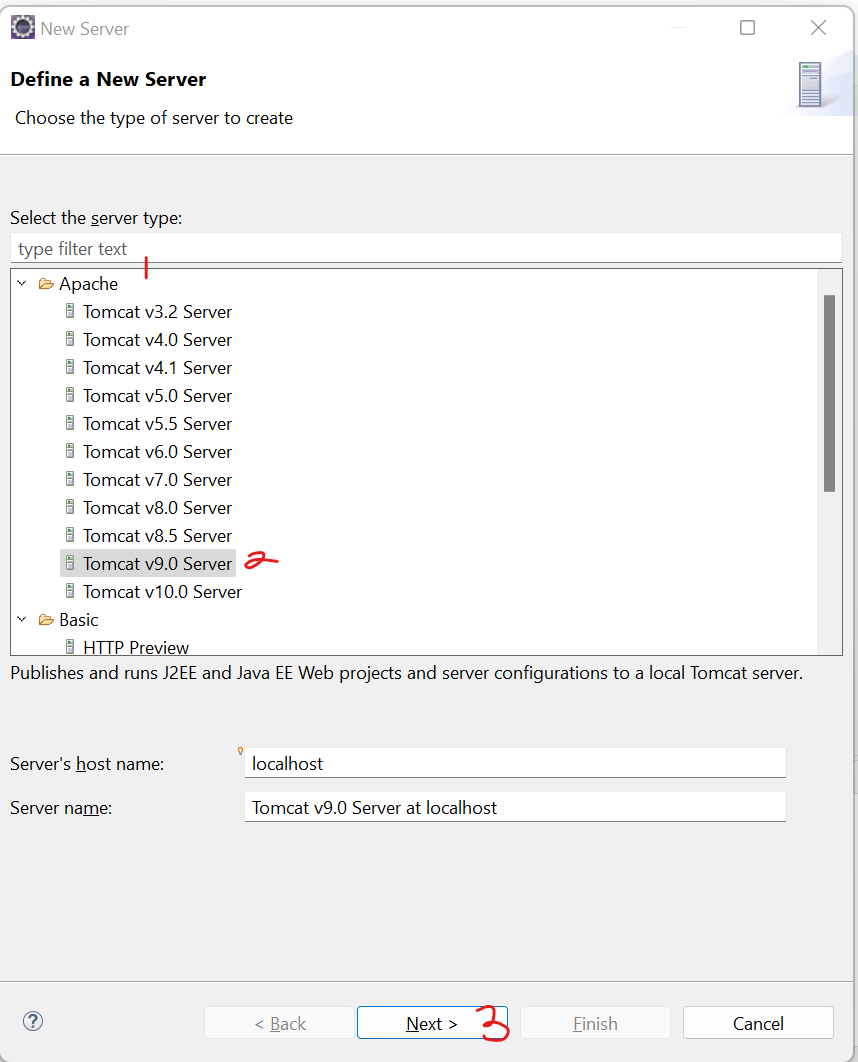
<https://tomcat.apache.org/download-90.cgi>



1. Extract the file into appropriate location.
2. Setup the server into eclipse
   1. Open an eclipse workspace
   2. Switch to “**Java EE”** perspective



* 1. Select a “servers” tab from the bottom of the screen.
  2. Click on the hyperlink to add new server.
  3. On the new screen expand Apache option. Select an Apache downloaded version. Click Next button

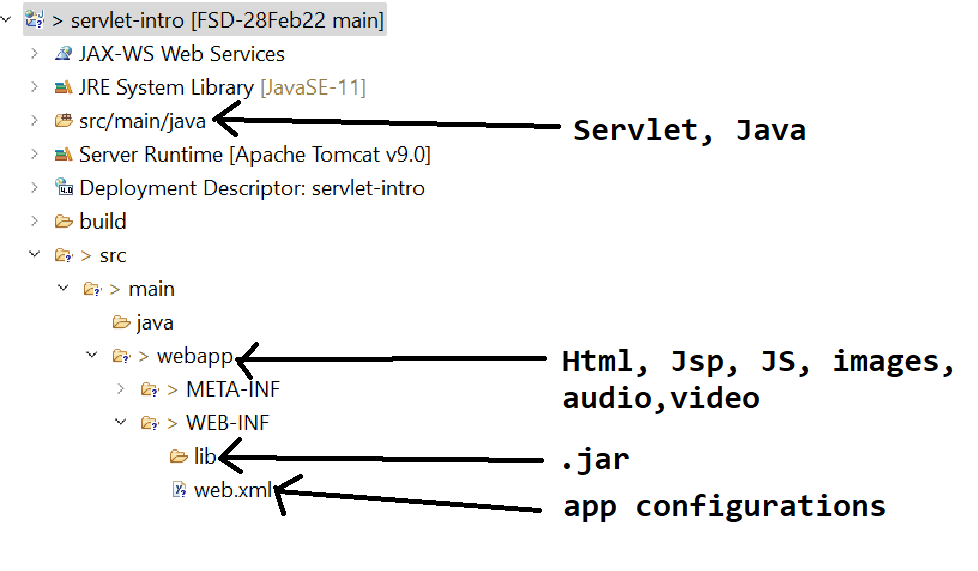


* 1. Browse the extracted tomcat folder and select a path till the parent folder of bin, lib, config.
  2. Click on Finish after setting the path.

**Create Dynamic Web Application**

1. Go to “File” menu -> “New” option -> Select “Dynamic Web Project.”
2. Provide a project name.
   1. Make sure that your target runtime is not **<NONE>.** It must be a tomcat server which is configured inside eclipse.
3. Click on “Next” -> Click on “Next” again.
4. Make sure that “Generate web.xml Deployment Descriptor ” check box is selected.
5. Click on “Finish” button.

Dynamic Web Application Structure



**Add external jar file into web Project**

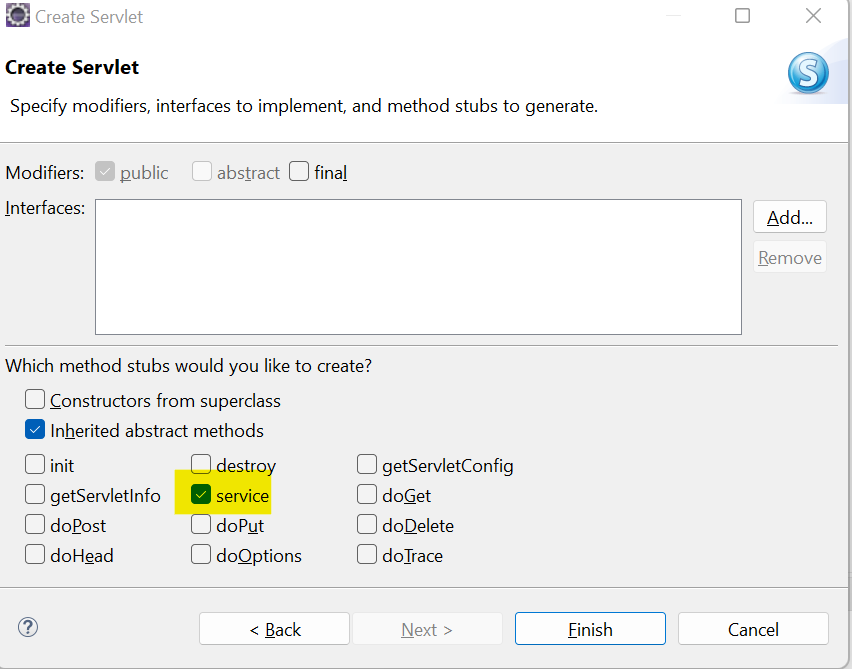
1. Copy a jar file from a location (Disk)
2. Paste a jar file into web project at location “**src/main/webapp/WEB-INF/lib**”

**Servlet**

1. Servlets are the java classes which are used to create a dynamic web pages.
2. In the servlet you can used HTML, CSS and Java code.
3. Servlets are also known as Html in Java.
4. Servlets are mainly used for getting request from the user, process a request and generate response. That is, servlets are used to manage user request and response.
5. Servlets do not have main method.
6. Servlets are managed by the servlet container which is a component form the server. Servlet container is responsible to create object of servlet, calls the methods from the servlet and destroy the object of servlet which is also known as servlet life cycle.
7. Every servlet will have a unique URL. Using which you can access the servlet.

**Create servlet using Eclipse**

1. Right click on “src/main/java”
2. Go to “New” option -> Select “Servlet” Option
3. Provide a servlet class name.
4. Click on “Next” button
5. Set the URL Pattern. Default URL is the class name you can edit the URL by click on “Edit” button.
6. Make sure that URL pattern must be start with ‘/’
7. Click on “Next” button.
8. Select the service option from the check box.



1. Click on “Finish” button

**Response In the Servlet**

1. Set the content type of the response. (Specify which type of response you are going to return)

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

response.setContentType("text/html");

1. Create an Object of PrintWriter

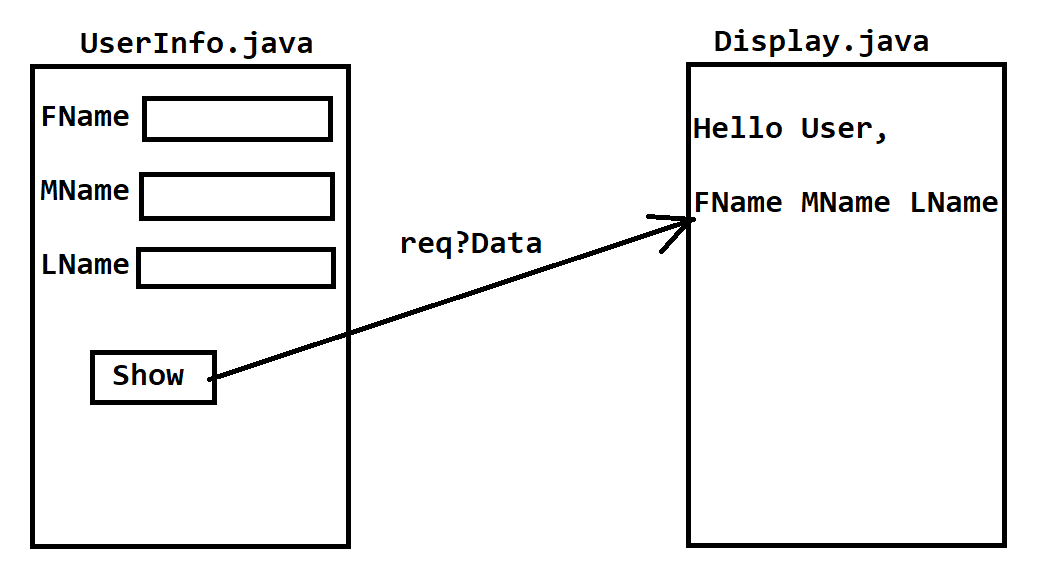
PrintWriter out = response.getWriter();

1. Write a response using Object of PrintWriter

**Parameter**

1. Parameter is the user details which send from the client to server inside URL.
2. Parameters are passed from the URL or internally from the request body.
3. Parameters are always added inside the URL after ‘?’.
4. There can be multiple parameters inside URL which must be separated by ‘&’
5. Every parameter has a key and value pair.
6. Parameters are always String type.

**Task 5-7-2022**



**Redirection Techniques**

1. In this you can redirect from one page to another without any user action.
2. There are two redirection techniques
   1. RequestDispatcher
      1. You can go from one page to another without any user action.
      2. This is an interface which has to ne instantiated using request object.
      3. In this option we have 2 methods one to FORWARD the request to next page and another is INCLUDE the next page into existing page.
      4. In this to redirect from one page to another same request will be used and no new request is generated, hence the data from the old request will be available on the new page also.
      5. Syntax:

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

**dis.forward(request, response);**

**dis.include(request, response)**

* 1. sendRedirect
     1. You can go from one page to another without any user action.
     2. This is a method which can be called by using response object.
     3. In this technique new request will be generated to go from one page to another and hence the data from old request will be deleted and it will redirected to new page.
     4. Syntax

**response.sendRedirect(“<URL>”);**

**Java Server Pages (JSP)**

1. JSP is mostly used for a dynamic pages design.
2. On JSP page you can use HTML, CSS, JS, Java.
3. In JSP HTML is use as primary language. 1.
4. The file extension of the jsp page is .jsp
5. There is no need of providing a URL for the JSP pages. By default the URL of the JSP page is the /filename.jsp
6. Jsp gets converted into servlet internally at the time of execution.

**Scripting element/Tags**

1. Scripting tags/elements are use to write a java code on JSP page.
2. Nesting Tags are not allowed, but you can achieve it by using break and continue rule.
3. There are 3 types of scripting tags
   1. Scriptlet tag
      1. Is used to write a java code on JSP page.
      2. Using this tag, you can write a java code which is local to a service method.
      3. The variables created using this tag will be a local variables.
      4. You cannot create method or static or instance variable using this tag.
      5. Syntax:

**<% Java Code %>**

* 1. Expression tag
     1. Is used to write a java expression which executes and display result on browser.
     2. The code written inside expression tag will be added inside out.print(<Expression>) method.
     3. This tag is an alternative for out.print().
     4. The Java Expression which added inside this tag will be included inside service method.
     5. Syntax:

**<%= Java Expression %>**

* 1. Declaration tag
     1. This tag is use to write a java code inside JSP.
     2. The java code will be added outside the service method and inside java class.
     3. This code is a instance level code, using this tag you can create a methods or static or instance variable inside JSP.
     4. The java code will be added outside service method and inside class.
     5. Syntax:

**<%! Java Code %>**

**Implicit Object**

1. The objects which are readily provided on every jsp page are called implicit objects.
2. All the implicit Objects are only accessible inside a service method.
3. There is total 9 implicit objects

|  |  |
| --- | --- |
| **Object** | **Class/Interface** |
| request | HttpServletRequest |
| response | HttpServletResponse |
| session | HttpSession |
| application | ServletContext |
| config | ServletConfig |
| pageContext | PageContext |
| out | PrintWriter/JSPWriter |
| exception | Throwable |
| page | this |

**Note: You can access these implicit objects only inside Scriptlet and Expression tag**

**Servlet Life Cycle**

1. Servlet life cycle is managed by Servlet Container.
2. In the servlet life cycle there are 3 stages.
   1. Init stage
      1. This stage gets executed after the instantiation of the servlet.
      2. This method gets called after the constructor, In this you can perform the initialization for the servlet.
      3. As a part of this stage init(ServletConfig) gets invoked.
      4. This step executes only once.
   2. Service Stage
      1. This stage gets executed after the request comes from end user.
      2. As a part of this method service method gets called.
      3. This method gets called multiple time in a life cycle.
   3. Destroy Stage
      1. This stage gets executed when the servlet object gets destroyed.
      2. As a part of this method destroy() method will be invoked.
      3. This method also called only once in a life cycle.

**JSP Life Cycle**

1. JSP life cycle managed by Servlet Container.
2. There are total 5 stage in the JSP life cycle.
   1. Translation
      1. The Jsp page gets converted into servlet that is in java
      2. Here, .jsp gets converted into .java
   2. Compilation
      1. The translated page gets compiled to .class file
      2. Here, .java gets converted into .class5
   3. Init
      1. In this stage the servlet (converted servlet) gets initialized.
      2. As a part of this stage jspInit() method invoked.
      3. This method gets executed only once in a life cycle.
   4. Service
      1. This method gets invoked when the user send a request.
      2. As a part of this stage the \_jspService(Request,Response) gets invoked.
      3. This methods gets executes multiple times in a life cycle.
   5. Destroy
      1. This method gets called when the servlet object gets destroyed
      2. As a part of this stage jspDestroy() method gets invoked.
      3. This method gets executes only once in a life cycle.

**Session Tracking Techniques**

1. These techniques are used to maintain the user information into multiple request.
2. There are 4 techniques of Session Tracking
   1. Hidden Form Field
      1. Used in a case where you are sending new request from one page to another using form tag and submit button.
      2. Hidden field techniques can be used by creating a hidden field and set the old request data into hidden field and send to the next page.
   2. URL Rewriting
      1. Used in a case where new request getting generated by anchor tag or sendRedirect technique.
      2. In this technique we can add the parameter manually inside a URL.
   3. Cookies
      1. Used to store the user information at client side.
      2. Cookies are create at server side and then sends to a client side to store into a client location.
      3. Cookies are always forward from every request to the server.
      4. Cookies can be store for a specific time (persistent cookie) or for a browser session (non-persistent cookie).
      5. Syntax to Create Cookie :

Cookie object = new Cookie(“Name”, “Value”); // to create cookie

response.addCookie(object); // send cookie to client side.

* 1. HttpSession
     1. Is used to store use information at server side.
     2. HttpSession internally used Cookies to store a JSessionId (token)
     3. HttpSession can be created or get existing session using following line of code

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

* + 1. To set the values inside session you can use following line of code

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

* + 1. To get the values from session you can use following line of code

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

* + 1. To Destroy session you can use two options
       1. Invalid the session

**session.invalidate();**

* + - 1. Timeout the session

Configure following tag in web.xml.

**<session-config>**

**<session-timeout>1</session-timeout>**

**</session-config>**

Here the time is in minutes.

**Add Jar Files into Dynamic Web Application**

1. Copy a ja file from the directory.
2. Paste The jar file into ecplise project at location src/main/webapp/WEB-INF/lib

Http Session Code

[**https://developer.mozilla.org/en-US/docs/Web/HTTP/Status**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)

**Set Error Pages in JSP**

1. Create and Design a Jsp page which has to display on error.
2. Set the JSP page as an error page

<%@ page isErrorPage=*"true"* %>

1. Configure the error page into web.xml.

<error-page>

<error-code>404</error-code>

<location>/not-found.jsp</location>

</error-page>