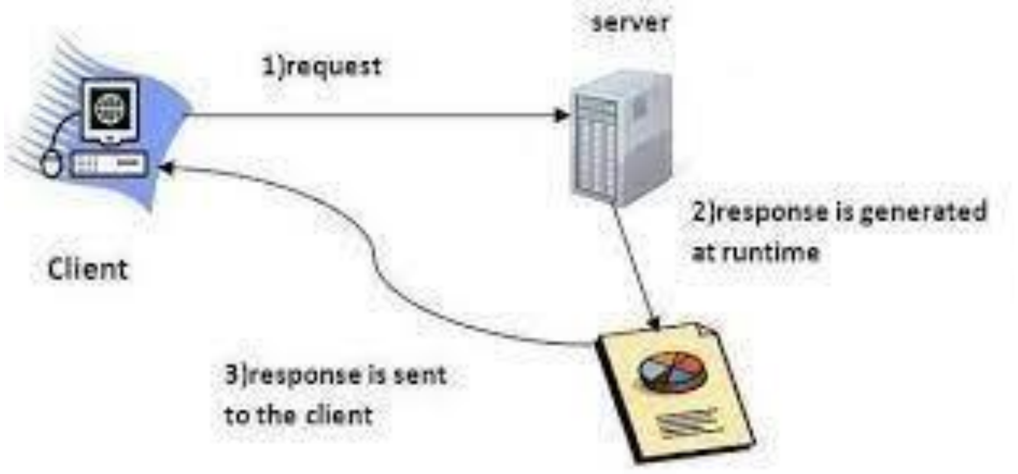
Servlets

1. Servlet is used to develop the web application.
2. To provide the web application to the user programmers going to use server.
3. Server is mediator between programmer & user which is responsible to accept the request from user and give the response to the user.

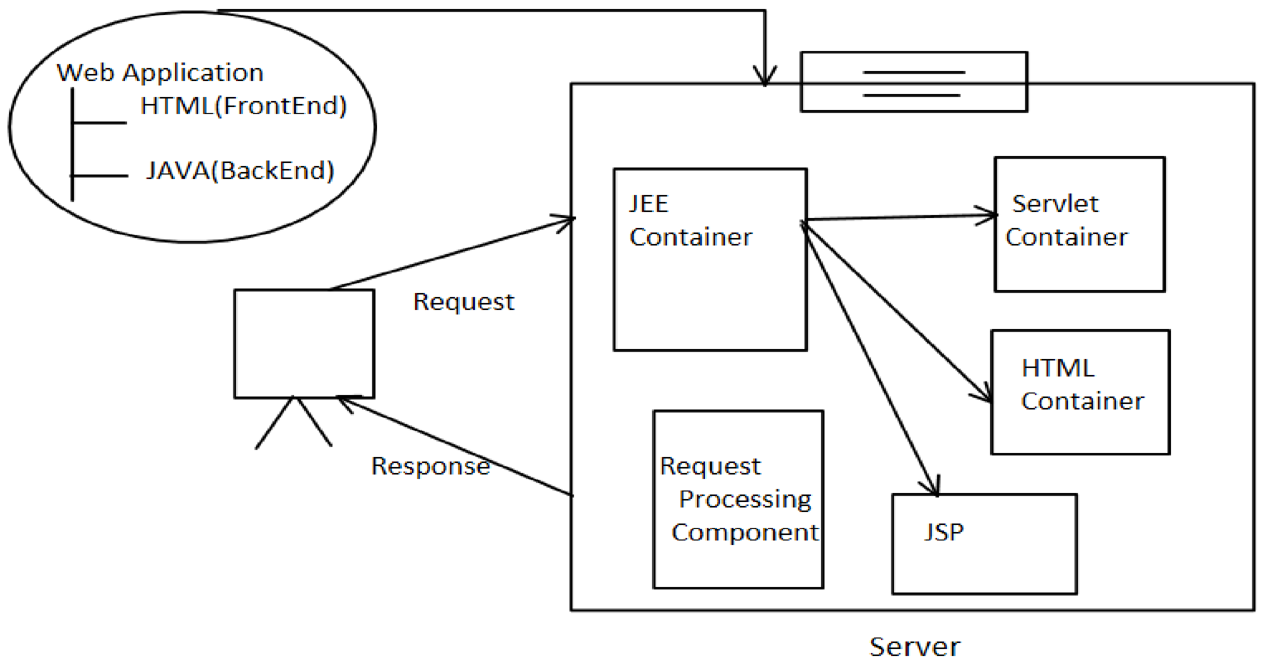

server
1)request
Client
2)response is generated at runtime
3)response is sent to the client

4. Programmer is giving an application to the server technically this process is referred as Deployment.
5. Deployment process is carry out once after development of application.
6. There are two types of Deployment
   a. Manual deployment.
   b. Automated Deployment.
7. In manual deployment programmers are responsible for writing the code as well as giving the application to the server.
8. In automated deployment programmers are responsible to writing the code whereas deployment is carried out by Maven Tool.

   Note: Developing the web application means writing the code for both frontend & backend of an application.

Web Container

1. JEE container is an important component within the server which manages entire web application.
2. The functionalities perform by JEE container are-
   a. Once after deployment JEE container will segregate the files of an application into separate container the based on the extensions of the files, which results in faster response.
   b. Once after segregation of files user can make an request & JEE container provide an response. This process is technically referred as processing of users request.
   c. JEE container manages servlet life cycle.
   d. JEE container responsible for the creation of config object & context object.
   e. JEE container handles Servlet chaining.



Type of Project

1. In eclipse there are multiple types of projects can be created based on requirement
   a. Java project
   b. Dynamic web project
2. Programmers prefers java project if requirement consist of only backend files.
3. Programmers prefers dynamic web project if requirement consist of both frontend & backend file.
4. Dynamic web project consist of four important folders.
   a. Src folder
   b. Web content folder
   c. Build folder
   d. Lib folder
5. Build folder is basically used for testing an application.
6. Source folder is use for backend file whereas web content folder is use for frontend file.
7. Lib folder contents jar files which is required to run the program.

Types of Classes

1. We have many types of classes which can be created in the project.
   a. Normal Java class.
   b. Servlet class
2. A class is said to be a normal java class if the execution of class starts with main method.
3. A class is said to be a servlet class if execution of class starts with request from HTML file.
4. It is an API similar to javax.servlet package.
5. It is also an interface present in javax.servlet package, it has some methods any class

who implments servlet needs to provide implementation for this methods.

6. Servlet programs runs on the browser.
7. It is a special class which acquire properties of server, like reading request and creating a response.
8. Servlet contains following methods.
    a. public void init(ServletConfig config) – called to initialize the members of the class.
    b. public void service(ServletRequest req, ServletResponse resp) – take request and produce response.
    c. public void destroy() – called to unload servlet classes.
    d. public ServletConfig getServletConfig() – to get config object generated by web container.
    e. public String getServletInfo() - gives servlet information.
9. Servlet interface implement in a GenericServlet abstract class and HttpServlet extends GenericServlet.

Generic Servlet

1. It implements all other methods from Servlet except service()
2. It is an abstract class present in javax.servlet package, which contains one abstract method called it as service().
3. This method takes two arguments
    a. ServletRequest --- > it provides user request information to server.
    b. ServletResponse --- > it represents response send back to user.
4. Syntax.
```
class Sample extends GenericServlet
{
        @Override
        + void service(ServletRequest req, ServletResponse resp)
        {
                //Backend code
        }
}
```
5. All the servlet classes will be executed once after there is a request from HTML file.
6. The basic way of making request from html file is by creating a button.
    <input type = "submit" value = "checkDate">

To get servlet-api jar file :
c: ----> program files ----> apache tomcat foundation ----> tomcat 8.0 ----> lib---- > servlet-api.jar

Que. How to Link front-end file and back-end file in dynamic web project?
Ans- We can link front-end file and back-end file by providing two important information.

1. In html file we have to provide form action
    <form action= "Servlet Class Name"></form>
2. Above the servlet class, we have to provide @webservlet annotation with the slash.
    @WebServlet("/servletClassName")--- > relative path.

Fetching user information from HTML file.

- To fetch user information from HTML file to servlet class programmer have to provide two important steps.

  Step1 :
  Declare identifier for each input in html file by using 'name' attribute.

  ```
  <input type = "email" name = "email"></input>
  ```

  Step2:
  - In the servlet class programmer have to call getParameter().
  - As an argument for getParameter(), we have to pass identifiers which are declare in HTML file.
  - The return type of this methods is String type.
  - It called by using ServletRequest reference variable.

  ```
  String email = req.getParameter("email");
  ```

Note: For giving the user information from html page to Database through servlet class we have to use static method forName() which is present in class "Class" . It is used to register the driver with servlet class.

Syntax - Class.forName("com.mysql.jdbc.Driver");

PrintWriter

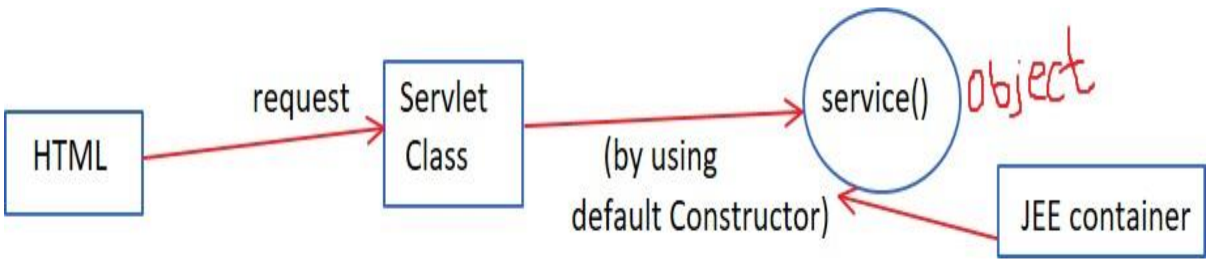
- Printing the output on to the browser, programmers using PrintWriter Class which is present in java.io package.
- The helper method for print writer class is getWriter(). To print the contents on to browser, we make use of println().
- Syntax : PrintWriter writer = resp.getWriter();

Servlet Life Cycle

1. Servlet life cycle shows all the phases from the time of request and till servlet class provide the response.
2. In servlet life five phases are there:
   a. Loading phase
   b. Instantiation phase
   c. Initialization phase
   d. Service phase
   e. Destruction phase

3. Loading phase
    a. In this phase, servlet class gets loads by class loader in servlet container.
    b. When first user sends the request, then above task gets perform, after that for further requests, class will not get load again & again.
    c. Only once servlet class loads in a servlet container.

4. Instantiation phase
    a. In this phase servlet object is created by the container.
    b. When user makes a request to the servlet class, container converts servlet class into servlet object by using default constructor present in servlet class.
    c. If servlet class does not contain default constructor, then we getInstantiationException (unchecked exception)



5. Initialization phase
    a. The process of passing the value for varies is referred as Initialization of servlet class.
    b. In this phase, programmer will be passing the values which are varies by using init() method.
    c. Passing the values can be done in two different ways.
        i. By using **init parameter**
        ii. By using **context parameter**
    d. The values which are passed for one servlet class is referred as init parameter. It is having **local scope.**
    e. The values which are passed for all the servlet classes in the entire application referred as context parameters. It is having **global scope**.

    Config & Context Object

    6. Config object & context object are created by container during initialization phase.
    7. As a programmer we use config object to pass init parameter and context object to pass context parameter.
    8. Since, config object make changes for one servlet class, the scope of config objet is local scope.
    9. Context object make changes for all servlet classes of entire application the scope of context object is global scope.
    10. For web application, container creates one context object and n no. of config

objects where n is equals to no. of servlet classes.

6. Service Phase

   a. In this phase request and response object is created by service method with the help of container.
   b. then first user makes request , container converts servlet class into servlet object using default constructor present in servlet class.
   c. If there are 100 user request for 1 servlet class then 1 servlet class object and 100 request & response object will get created.

      100 servlet class present in application & 100 separate user send the request for each servlet class.
      Servlet object no --- > 100
      Request ----> 100(user) * 100(servlet) = 10000 request object
      Response ---> 100 * 100

   d. If there are 100 user request for 1 servlet class then only 1 servlet class object & 1 request & response object will get created then this kind of application we are calling it is single threaded application.

   Note:

   a. by default servlets will act as an multithreaded as a programmer we can make servlets to act as single threaded by using synchronized keyword.
   b. If container working with multithreaded applications then each user will be maintain separately by creating multiple request object & response object. This results in independent behaviour.
   c. If container is working with single threaded application then container creates one request object and one response object which can be use by only single user at time this results in dependent behaviour.

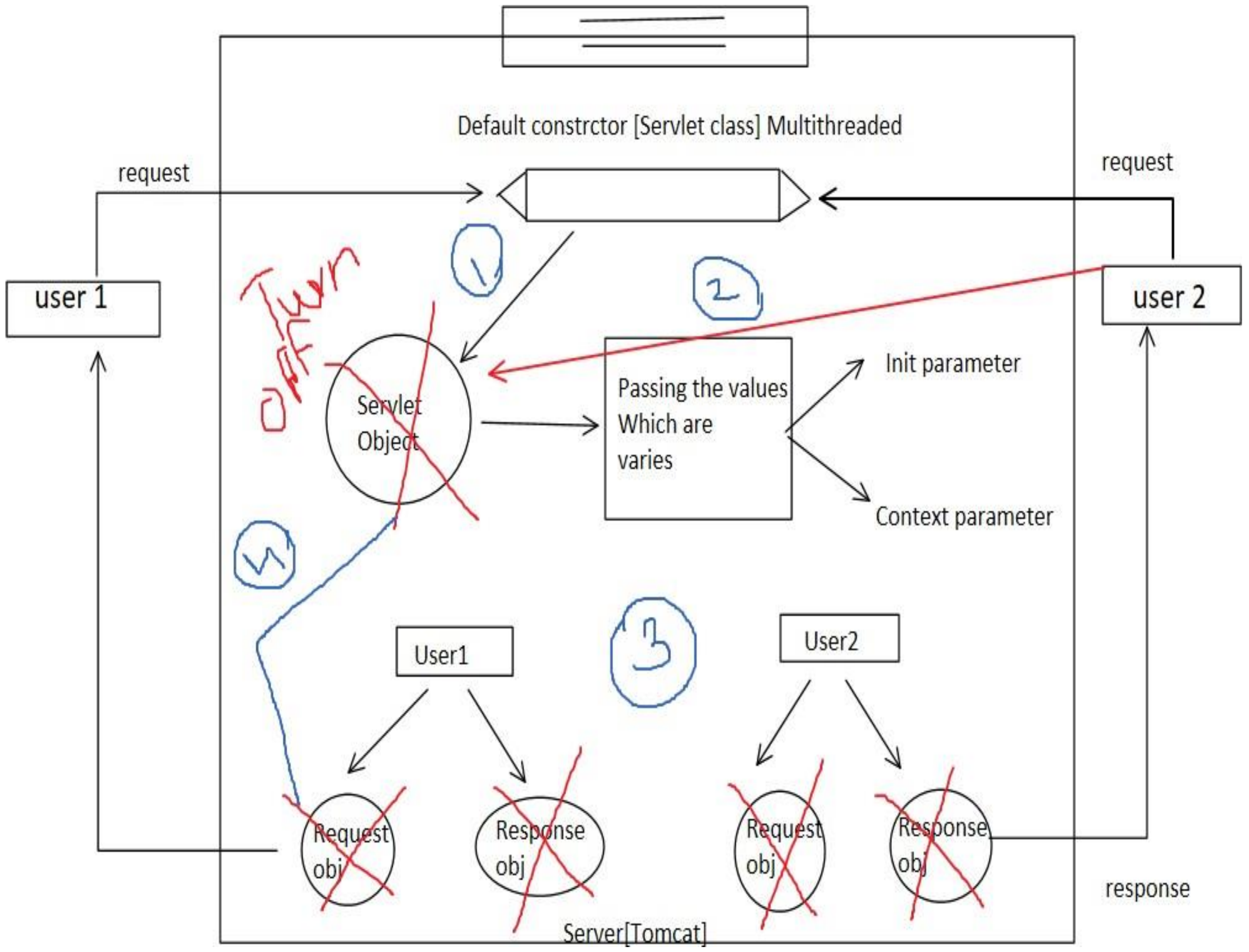
Conclusion ---->

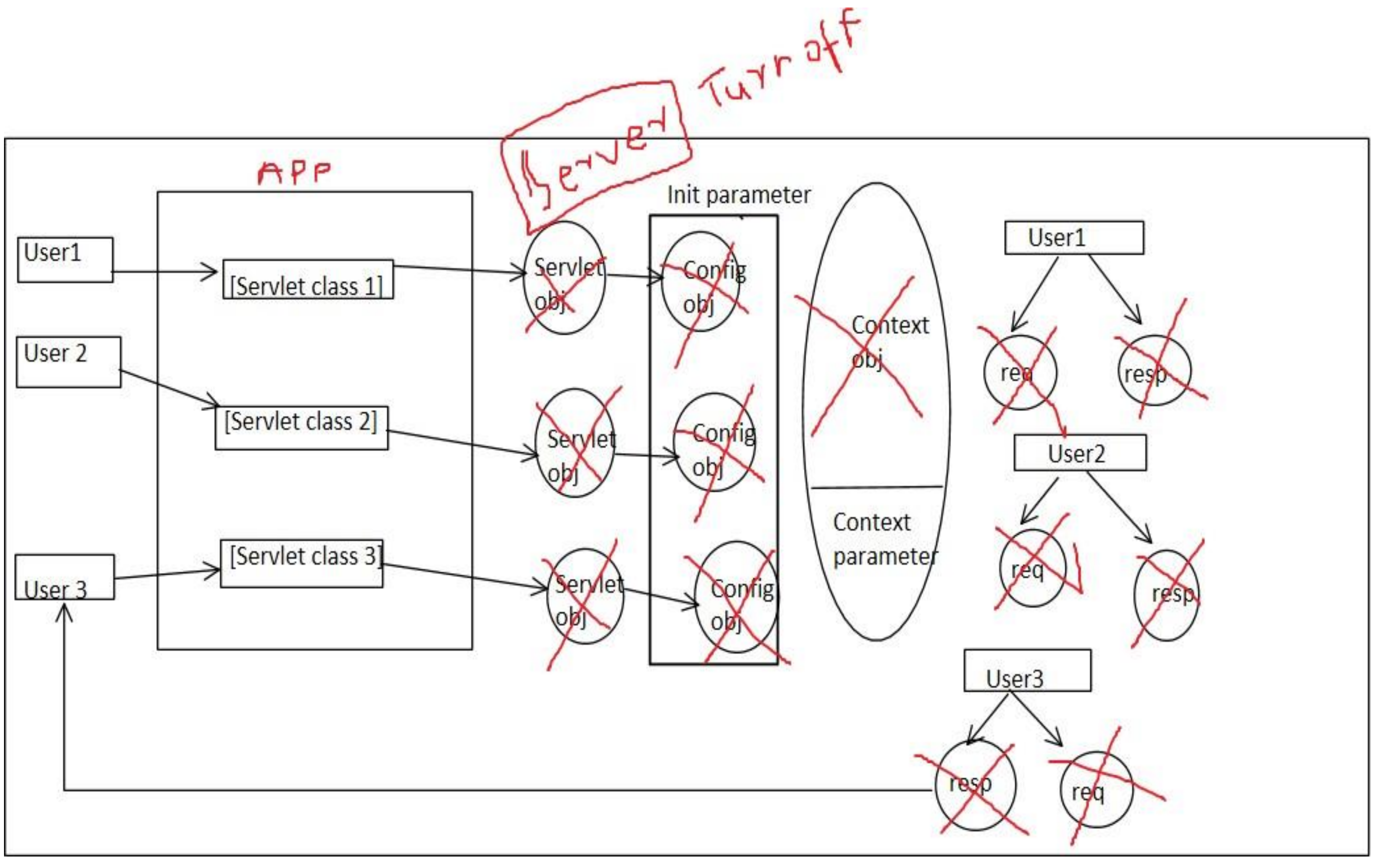
a. No. of servlet class = no. of servlet object ---> 1st phase
b. No. of servlet class = no. of Config object & only one Context object for application----- > 2nd phase
c. No. of user request = no. of request & response object for multithreaded application
   No. of servlet class = no. of request & response object for single threaded application---- > 3rd phase

g. Destruction phase

   a. In this phase the objects which have created in the above phases will be destroyed by the container.
   b. Once after objects are destructed servlet life cycle ends.

Note: servlet object will get destroy when server gets turn off.


Default constrctor [Servlet class] Multithreaded
request
request
user 1
user 2
Turn off
1
2
3
Servlet Object
Passing the values Which are varies
Init parameter
Context parameter
User1
User2
Request obj
Response obj
Request obj
Response obj
response
Server[Tomcat]


Server Turn off
APP
User1
User 2
User 3
[Servlet class 1]
[Servlet class 2]
[Servlet class 3]
Servlet obj
Servlet obj
Servlet obj
Init parameter
Config obj
Config obj
Config obj
Context obj
Context parameter
User1
req
resp
User2
req
resp
User3
resp
req

HTTP SERVLET

1. There are two types of method request in HTML File
    a. Get
    b. Post
2. If method = "post" then user information is hidden without displaying on url. Hence post request is secure.
3. If method = "get" then user information is displaying on url. Hence get request is unsecure.
   Note: by default, if "method " attribute is not declared in html then it works as get request.
4. HTTP servlet is an abstract class present in javax.servlet package.
5. Being abstract class it does not contains any abstract method instead it contains complete methods. Those complete methods are doGet(HttpServletRequest req, HttpServletResponse resp) & doPost(HttpServletRequest req, HttpServletResponse resp)
6. If method = "post" in html file then override doPost() in HttpServlet class.
7. If method = "get" in html file then override doGet() in HttpServlet class.

   Note:
   1. If there is mismatch between method in HTML file & method overridden in HttpServlet then its results in 405 error.
    1. doPost() & doGet() are declared as protected which makes inheritance mandatory.
    2. If class contains super keyword then its results in 500 error/405 error.

Servlet Chaining

1. As a programmer we use servlet chaining to link one functionality to another functionality sequential flow of execution.
2. When a user make a request, servlet class can execute the logic either by using service methods, doPost() or doGet() and then communicate to other resources like HTML files, JSP files and another servlet class.
3. This type of servlet communication to other resource is referred as servlet chaining.
4. Servlet chaining can be performed in two ways.
    a. By using RequestDispatcher Interface
    b. By using sendRedirect()
5. RequestDispatcher Interface
    a. It is present in javax.servlet package
    b. It is used to chain from one servlet class to other resources like HTML file, Jsp file or another servlet class.
    c. Syntax -

```
RequestDispatcher dipatcher =
request.getRequestDispatcher("filename/relative path");
```

6. In the RequestDispatcher two methods are present –
    - Include(req, resp) – It include response of current servlet as well as further servlets.
    - Forward(req, resp) – It forward complete response to requested servlets.
7. To transfer the data during servlet chaining, make use of "setAttribute(identifier, value)"method present in ServletRequest interface, which upcast give data to object class.
8. To get transferred data into requested servlet, make use of "getAttribute(identifier)", present in ServletRequest interface, which returns upcasted object.

9. sendRedirect()
    a. It is present in HttpServletResponse interface.
    b. It is used to chain one servlet class to other resources which is present in same project or in external project.
    c. As an argument programmer can pass relative path / absolute path/ url.
    d. Syntax:

```
resp.sendRedirect("relative path / absolute path")
```

Session

1. It is time interval for user between login & logout.
2. Sessions are used to store user information in temporary format.
   Note: Database is store user information permanently.

3. Whenever application needs user information to perform any task, so that time programmer can access it from session instead of DB and this results in faster execution.
4. When user logged in into application then information is stored in session object from database and when user logged out from application then information is deleted from session.
5. To store session object programmer using HttpSession interface rv & to create object we have to use getSession() which present in HttpServletRequest.

   Syntax. - `HttpSession session = req.getSession();`

6. To store user information into session object programmer using setAttribute() present in HttpSession interface. This method takes two argument 1) identifier for user information.
    1. User information which we want to store.

   For example - `session.setAttribute("stuName", name);`

7. By using Session programmers can reduced the execution and save the memory.

8. To give the time interval for session used setMaxInactiveInterval(int sec) which accepts arguments in seconds.
9. To retrieve values from session object in program used following method.

   session.getAttribute("identifier of user info");

   When programmers wants to retrieve value from session that values are already upcasted into the object class. Hence firstly downcast the value and then use it.

   For example --->   String name = (String)session.getAttribute("identifier");

10. After the session time interval all the object which stored in session that gets null. So that time programmers can close the session by using invalidate() which present in HttpSession interface.

    For example ---> session.invalidate()

Java Server Pages

1. The main idea of JSP files is to avoid the disadvantage of Printwriter class.
2. While developing applications, programmers will be having a requirement of giving the output which is the combination of Java code and html code.
3. But PrintWriter class is having the ability of giving the output of java coding only.
4. To overcome the above drawback the programmer will make use of JSP files.
5. To differentiate java coding & html coding in the jsp file, programmer using jsp elements.
6. There are five types of JSP elements.
   a. Expression
   b. Declaration
   c. Action
   d. Directive
   e. Scriplets.

7. Scriplets :

   a. Programmers will make use of this element to provide local contents in the jsp file.
   b. It is indicated by <% -JC- %>

h. Expression:

   a. Programmers will make use of this element to print java contents on the browser.
   b. It is indicated by <%= -JC- %>

Note: All expression jsp elements contents will not end with semicolon(;).

JSP Life Cycle:

a. In Jsp life cycle we have five phases.
   a. Translation
   b. Instantiation
   c. Initialization   Servlet life cycle
   d. Service
   e. Destruction
b. Translation phase
   a. In this phase all the jsp files will be converted into servlet class, converted servlet classes is technically referred as "Translated Servlets"(T-servlets).
   b. All the jsp files are converted into servlet class at runtime that means only after execution of JSP file.
   c. Translated servlet file names will similar to jsp file name.
      Example - If jsp file name is Demo.jsp then Translated servlet would be Demo_jsp.java.

   d. During runtime all JSP files are firstly converted into .java file and later compiled for .class file of T-servlets.
   e. After compilation of T-servlets, the servlet lifecycle process starts.

Note. Instantiation, Initilization, service and destruction phase of JSP life cycle is similar to servlet life cycle.

Path of Translated servlet :
D:\WorkSpaceFolder\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\ProjectName\org\apache\package-name

3. Declaration:
   a. Programmers will make use of this element to provide global contents within JSP file.
   b. It is indicated by <%! -JC- %>

4. Action JSP element

   a. As a programmer action jsp elements allows programmers to perform Java task with the appearance of HTML tags.
   b. Syntax:
      <jsp:action-name attribute = "value"></jsp:action-name>
   c. There are many JSP action tags, some of them are

      i. useBean
         - It is basically use by the programmers in order to create an object of a class.
         - It takes three arguments.
           1. Id :- represents reference variable of an object.
           2. Class :- represents the restriction on the object it takes four values.
         - If scope = "page" then the object created is accessible only in the current JSP files.

- If scope = "application" then the object created is accessible only in the entire application.
- If scope = "request" then the object created is accessible only in the current JSP files as well as immediate JSP file.
- If scope = "session" then the object created is accessible only till particular time interval.

Note: If a scope is not represented for <jsp:useBean> tag then by default scope is considered as "page".

For example : <jsp:useBean id = "c1" class = "org.jsp.app.car" scope = "page"></jsp:useBean>

i. setProperty
   - It is basically use for initializing the object using action tag. It takes three important attribute.
     1. Property :- It represents variable name of an object which has to be initialize.
     2. Name :- It represents reference variable name of an object.
     3. Value : represents actual information which has to be passed for the variables.

ii. getProperty
   - It is basically use to retrieve the information from the object. It takes two attribute.
     1. Property
     2. Name

- Forward
  - It is basically used to link one jsp file to another jsp file. It takes one important attribute.
    1. Page :- represents jsp file name to which chaining has to be perform.
- Include
  - It is used to include one jsp page to another one for achieving code reusability and to link one jsp file with another. It takes one important attribute.
    1. Page :- represent jsp file name which code has to be include.

Note: for include action tag, we have to pass only the file names which are existing. If in case we pass file name which is not existing then it throws compilation error.
for forward action tag, if we pass file name which are not existing then it throws runtime exception.

- Directive JSP element
  1. JSP directives basically takes the help of other packages, other files to overcome from errors appearing in JSP file.
  2. It is represented as <%@ -JC- %>
  3. We have three types of directives

    i. Page
    ii. Include
    iii. Taglib

4. Page Directive
    i. It mainly used to make changes in current jsp file.
    ii. Represented as given below -->

        <%@ page %>

- Few attribute which can be used along with page directive are --->
    1. Import - used to import the files from differentiate package into current jsp file.
    2. Extends - used to extends super class members in current jsp file. We have to pass fully qualified name for this attribute. Once After extending the class programmer can override methods of super class by using declaration.

    Note: Programmer can extend only user-defined servlet class.

    - errorPage -
        1. It is used to maintain JSP file name which consists of recovery code.
        2. If given jsp file consist of an exception then control will be automatically shifted to the jsp file name which is consist of recovery code.
    - isErrorPage -
        1. It is used to mention the current JSP file consist of only recovery code.
        2. It is always writes to indicate the jsp file which only consist of recovery code.

- Include Directive
    1. It is basically mentioned within body tag.

```
<body>
        <%@ include file = "filename.jsp" %>
</body>
```

- This helps the programmer use multiple file together, where all the file output will be displayed at once on the browser.

f. Taglib directive
    i. It is used to create custom jsp elements which can be used only if given jsp elements is not working in the application.
    ii. Taglib is advance one for that we need to go through sprinMVC.(Model View Controller)
    iii. We use TLD(Tag library Discriptor) file to define the tags.

Cookies

1. Cookie is a piece of information which is used to store user information within the browser.

2. There are two types of Cookies-
   a. Persistent
   b. Non-Persistent
3. The cookies which must be removed explicitly is referred as Persistent Cookie whereas The cookie which is active particular time spam is referred as Non-Persistent Cookie.

Note: Cookies are created within server and it is stored within the browser.

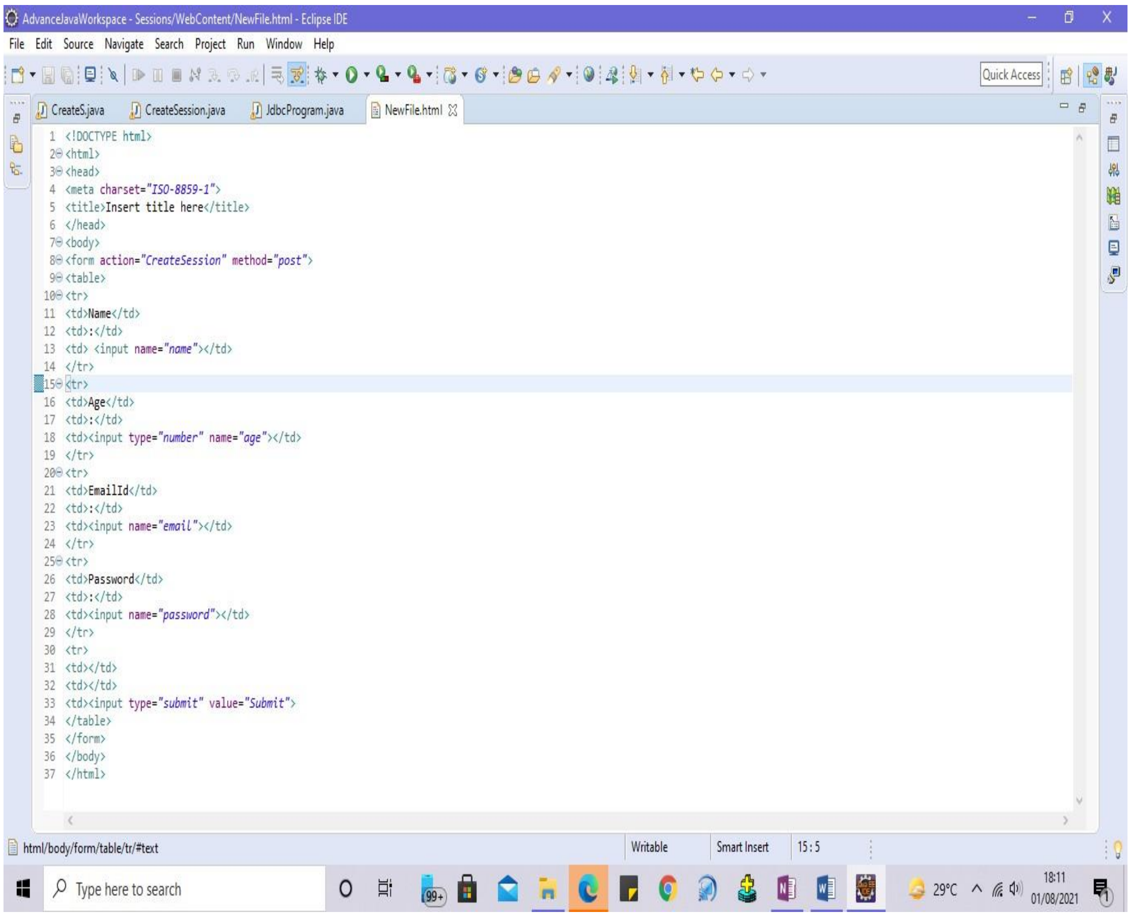
4. As a programmer if we have to create cookies then we have to make use of Cookie class, which is present in javax.servlet.http package.
5. The syntax of creating cookies is –

Cookie refvar = new Cookie(String identifier, String userInfo);

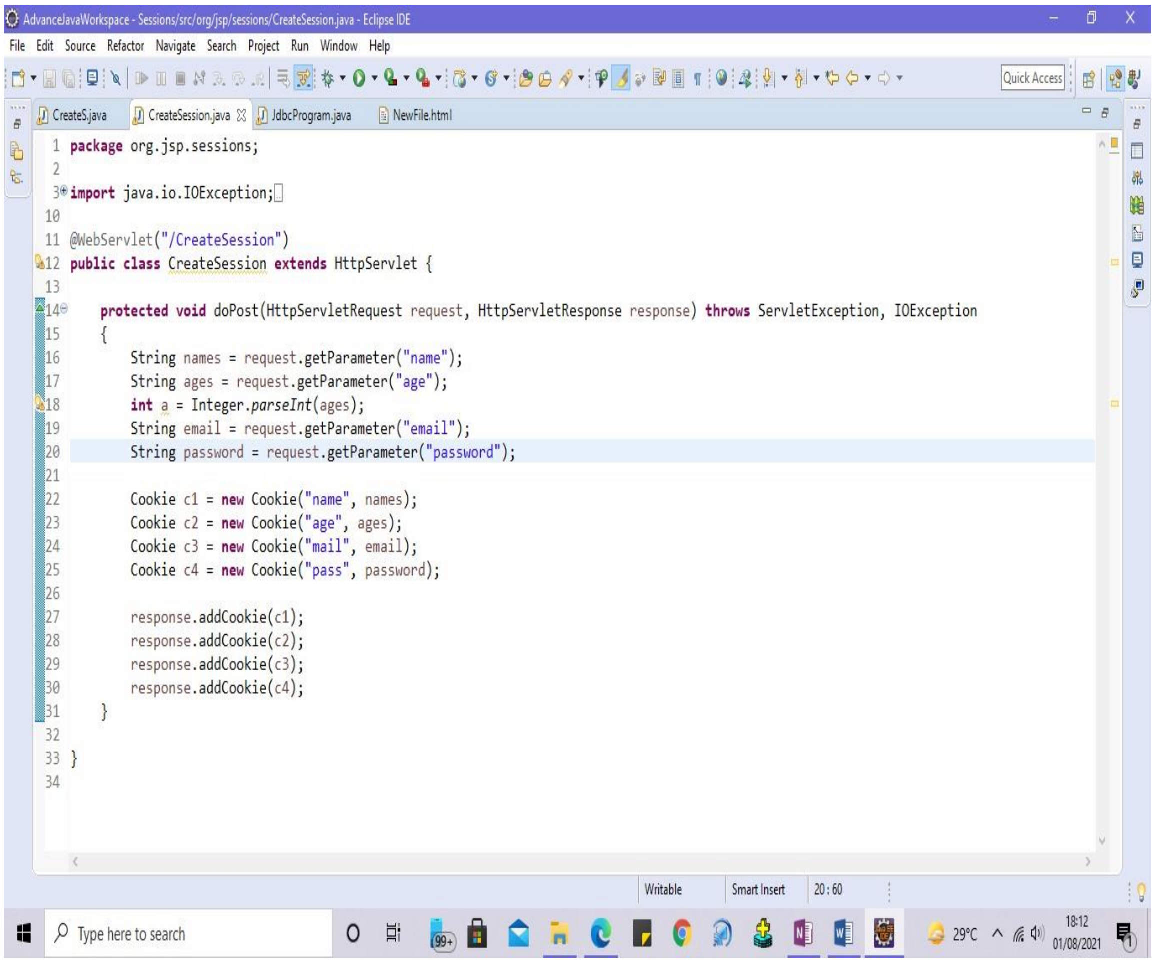
Where, identifier reference to the Cookie which is stored within the browser.

6. To send that cookies from server to browser, we have to call addCookie(), which must be called by using respose ref var.

Example: resp.addCookie(cookie1);


AdvanceJavaWorkspace - Sessions/WebContent/NewFile.html - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
CreateS.java
CreateSession.java
JdbcProgram.java
NewFile.html
Quick Access
html/body/form/table/tr/#text
Writable
Smart Insert
15 : 5

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="CreateSession" method="post">
<table>
<tr>
<td>Name</td>
<td>:</td>
<td> <input name="name"></td>
</tr>
<tr>
<td>Age</td>
<td>:</td>
<td><input type="number" name="age"></td>
</tr>
<tr>
<td>EmailId</td>
<td>:</td>
<td><input name="email"></td>
</tr>
<tr>
<td>Password</td>
<td>:</td>
<td><input name="password"></td>
</tr>
<tr>
<td></td>
<td></td>
<td><input type="submit" value="Submit">
</table>
</form>
</body>
</html>
```



```
package org.jsp.sessions;

import java.io.IOException;

@WebServlet("/CreateSession")
public class CreateSession extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        String names = request.getParameter("name");
        String ages = request.getParameter("age");
        int a = Integer.parseInt(ages);
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        Cookie c1 = new Cookie("name", names);
        Cookie c2 = new Cookie("age", ages);
        Cookie c3 = new Cookie("mail", email);
        Cookie c4 = new Cookie("pass", password);

        response.addCookie(c1);
        response.addCookie(c2);
        response.addCookie(c3);
        response.addCookie(c4);
    }

}
```

Note: We can store userinformation inside String array and then pass that rv to Cookie.

7. getCookies() is used to retrieve cookies from browser which must be called by using request ref var.
8. The return of getCookies() is Cookies[], which represents multiple cookies.
9. To retrieve the values from Cookie we have to make use getValue().

Note: We cannot use one jsp element into another jsp element.

10. As a programmer we can decide the time span of an cookie by using setmaxAge() which accepts integer values in seconds and called by using cookies ref var.

For example:

```
Cookie [] carr = request.getCookies();
For(Cookie cks : carr)
{
        System.out.println(cks.getValue());
}
```

Note: To view stored cookie in client machine follow below steps:

1. Go to chrome settings
2. Cookies and other site data
3. See all cookies
4. Search cookies -→ localhost.