unit- IV

Introducing servlets:- Background, the life cycle of a servlet, servlet development options, using Tomcat a simple servlet, create and compile the servlet source code, start Tomcat, start a web browser and request the servlet, the servlet API: The javax.servlet package, Reading servlet parameters, The javax.servlet.http package. Handling HTTP Requests and Responses. Handling HTTP GET Requests. Handling HTTP post Requests. Using Cookies. session tracking. Accessing Databases with JDBC using servlets.

Background:- we have to understand how the client is communicating with the server before going to the servlet we have to understand the co-ordination between client and server. If you consider a static web page a client send a request to the server in the server will process the request and send a response to the client. we are sending a request using web browser at client side simply it is following client server model.

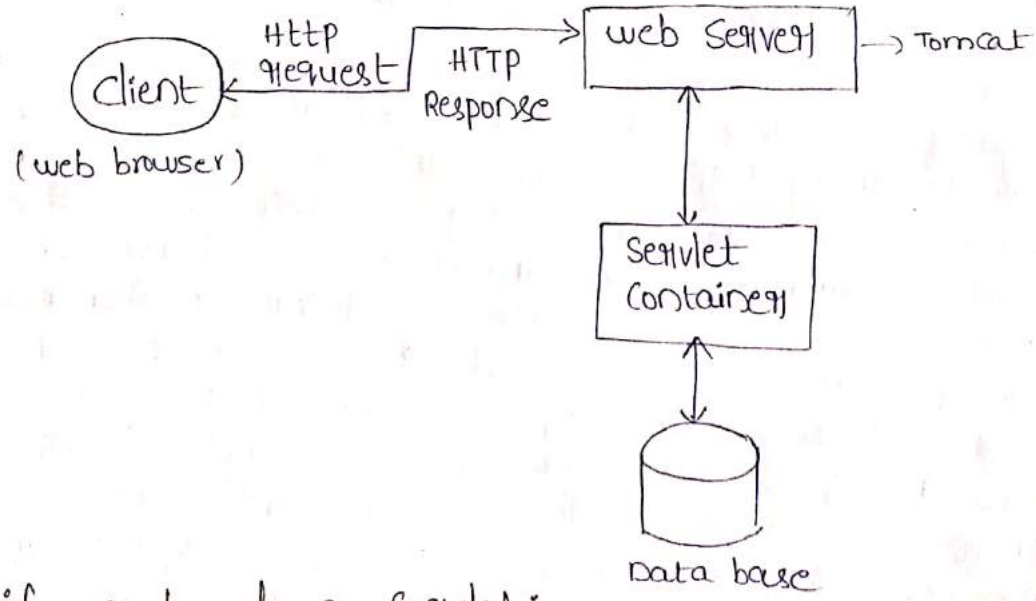
In dynamic web pages first a client sends a http request to the server. The server simply receives the request and perform the relevant activities (data base connections) and send a response to the client. To maintain communication we have an interface (CGI) common Gateway Interface. Generally the CGI programmes are built in C, C++. once the server is receives the request immediately the CGI will create a new process to handle the request and sending a response. These CGI applications are sufficient only limited number of client requests are there. If the no. of client requests are increased then the no. of process also increased by this. The performance may be degrade and memory utilization is more and it is more time consuming process. Another disadvantage of CGI is that it is not a platform independent.

To overcome all the limitations of CGI we have servlet concept. The servlet creates only one

instance in order to handle multiple client request. it is used only single address page and it is developed by using Java program. so that the servlets are platform independent.

A servlet is a java program which can be used to extend a capability of web server. Generally the servlets are used to create the dynamic web applications. Execution of servlets as follows:

1) Client send an http request to the server using web browsers.
2) The server receives the request and send same to the web container or servlet container. The servlet container is a part of web server which contains servlet classes.
3) Running servlet class will handle the request and send the response in the form of output.
4) The same response is transferred to the web server.
5) The web server transfer the same output to the client and display the output in a web browser.



Life cycle of a servlet:-

we have the following three method in life cycle of a servlet.

1) init()
2) service()
3) destroy()

1) init():- The init() will be call only once. it is used to initialize the parameters and loads the servlet execution. Generally, the init() is used to create a connection with database because while performing data base activities. we can connect the data base only once but not multiple times. it has the following signature.

```
public void init(ServletConfig con) throws ServletException
{
||--
}
```

where ServletConfig is an interface which provides the configuration details of servlet.
for example, web.XML
the above signature is a parameterised signature.
we have another signature of init() which is non-parameterised method.

```
public void init(      ) throws ServletException
{
||---
}
```

Every servlets may generate two types of exceptions those are ServletExceptions and unavailableException.

2) Service():- once the servlet is initialize then the service() will be called inorder to handle multiple request and sending response to the server.
It has few more methods in order to handle a getType request and postType request (doGet(), doPost())
It has the following signature

```
public void service(ServletRequest Req, ServletResponse Res)
throws ServletException, IoException
{
||---
}
```

It has two parameters those are request and respon se. if the servlet is extending a GenericServletType then we have a ServletRequest and a ServletResponse as a parameters. if the Servlet is extending an

HTTPServlet type then it has HTTPServletRequest and HTTPServlet Response. Then the another Signature.

```
Public void service(HttpServletRequest Req, HttpServletResponse Res)
throws ServletException, IOException)
{
// - - -
}
```

3) destroy(): The destroy() will be called only once. it will call at the end of lifeCycle generally the destroy() is used to perform the cleanup activities. For Example, Closing database Connection, Releasing memory space occupied by the Servlet and releasing resources which are utilized by the Servlet. After destroy() is called, the garbageCollector take into an action. It has the following signature

```
Public void destroy()
{
// - -
}
```

Servlet development options:-

We can develop the servlet by Connecting to the servlet Container on server generally we have the following two servers.

a) Glassfish
b) Tomcat

The Glassfish is from oracle and it was embedded in the net beans IDE the Tomcat server is maintained by Apache software foundation it is also embedded in netbeans IDE but the ~~Glassfish~~ tomcat is open source Software.

These two servers ~~are~~ are also embedded in eclipse IDE at a Configuration of developing a servlet is differ from one IDE to another IDE.

we can also develop a servlet without using any IDE by using a Command prompt we can Compile our servlet and by setting the relevant path we can Execute Servlet.

using Netbeans IDE:-

1) select file Menu then you can find a new project
2) under new project we have to select java web by this it will create a web application.
3) Next we have to select the server that may be either tomcat or glassfish
4) under web application we have a source package under source package we have a default package by right click on default package we can create a servlet.
5) Now we have to add the deployment discription (web XML). Generally the deployment description contains the servlet name, servlet class and the servlet url pattern
6) we can also add Html pages by right clicking on web pages directly as node.
7) By default we have an Html page that is Index.Html while running your servlet by default this Html page will execute.
8) if you want to run a particular Html or servlet we have to right click on those files then select run file.
9) Before executing servlet first of all we have to run our project once because it will create the respective class file then we can go for execution.

Note: In net beans IDE the default port no. of tomcat server is 8084.

using Tomcat:

1) we have to download the Tomcat latest version software from the tomcat.apache.org
2) After downloaded we have to go for installation which installing tomcat server by default it has 8080 http Connection port. if this port no. is already using in our system then we can assign a different port no. from the tomcat server.
3) After installation the respective directory will create in following path C:\>Programming files\
4) In the above url we can find the apache software foundation directory in these directory we have all related files to run our tomcat server.

5) To run a tomcat server manually we have to open a tomcat manager panel then we can start our tomcat service we can find the tomcat manager in the following path c:|> programming files| Asf| Tomcat 6.0| bin

But in latest version of tomcat server we have start batch file in bin directory to start our tomcat service we can also shut down our service by selecting shutdown batch file.

6) A servlet program has a different servlet APIs classes and interfaces. So that we have to set the servlet API. Jar has a classpath environment variable. we can find the servlet-API. Jar file in the following path.

c:|>programming files| Asf| Tomcat 6.0| lib;

7) we have to create our own directory web apps directory it can find as follows-

c:|program files| Apache software foundation

↳Tomcat 6.0

↳ web apps

↳ create new directory (CSE0)

8) Create a servlet program in our directory and compile it using javac it will generate the class file.

9) Now we have to copy the webInf0 directory from root directory to our newly created directory.

c: |program files| Asf

↳ Tomcat 6.0

↳ web apps

↳ create New directory(CSE0)

- program
- class
- web Inf0
  - classes
  - web.XML

↳ Root

↳ web :Inf0 (copy)

Create a new directory in CSE| web Inf0 and name it as "classes"

Now copy or cut our class file into CSE0| web-INF| classes directory.

Now we have to open web.xml (c:\tomcat\webapps\web-inf) and add the following two tags in order web.xml to deploy our servlet.

```
<web apps>
 <servlet>
 <servlet-class>servlet class name</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name> csed </servlet-name>
 <url-pattern>/hello </url-pattern>
 </servlet-mapping>
 </web apps>
```

Note: we can also copied about two tags from the any web.xml which can find in different directory (Manager, Example directory etc)

simple servlet:

1) we have to create a simple servlet as follows

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class simple extends GenericServlet
{
public void service(ServletRequest req, ServletResponse
                    res, throws ServletException, IoException)
 {
 res.setContentType("text/html");
 PrintWriter out = res.getWriter();
 out.println("welcome to simple servlet");
 }
}
```

2) we have to import the servlet API packages those are javax.servlet
javax.servlet.http

3) Create a class which extends either genericServlet or http servlet

4) Now we have to implement the lifecycle methods (init() service and destroy) but generally we use service method in servlet programs the service() will simple gets a request from the client using servlet request interfaces and sends a response using

servlet response.

5) Now we have to use setContentType method to set the mime type of response (multipurpose internet mail extension) usually we have the text/html mime type to set an html response to the client.

6) A getWriter() will written to print writer generally it will send response to the stream.

7) using println() we can display something on client browser.

Create and compile the servlet source code:-

Every servlet execution has following steps:

1) Create and Compile the servlet

we can create a servlet by using above sample code then we have to compile it. and move our class file into appropriate directory.

Ex:- javac Simple.java

2) start the Tomcat.

we have to start the Tomcat service using either Tomcat manager control panel or by executing start.bat file. If you want to check whether the tomcat service is running or not. Then we have to type the following url in our web browser.

localhost:8080

Note: The above url the tomcat port no is 8080 if it is already running by any other server. Then we cannot start our tomcat service. we have to use the following procedure to change the tomcat port number.

C:\> Program Files →

Apache Software foundation

↓

Tomcat 6.0

↓

Config

↓

Server.xml

↓

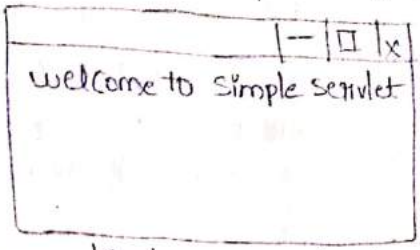
< Connetor port = "9090"

3) start a web browser and request the servlet

In this step we have to open any web browser in our system. and type the following url to

request the servlet and display the response in our web browser.

http://localhost:9090/csed/hello



we can also display a text using html tags as follows:

```
out.println("<html>");  ---> out.println("<head>");
out.println("<title> Simple servlet </title>");
out.println("</head>");
out.println("<body>");
out.println("<h1> welcome to simple servlet </h1>");
out.println("</body>")
out.println("</html>");
```

Note: we can align the text in middle as follows

```
out.println("<h1 align='center'> welcome to simple servlet </h1>");
```

Servlet API:-

The servlet API contains different packages generally we have the following two packages to build a servlet

a) javax.servlet.*
b) javax.servlet.http.*

The above two packages are various servlet classes and interfaces to build a servlet. The JavaSE is not providing these packages. The Tomcat server and the JavaEE is providing these packages.

a) javax.servlet:-

The javax.servlet package has the following interfaces and classes. These interfaces are implemented by servlet class. A user class is always extended by a servlet class. The following are the servlet interfaces.

Interface Name		Description
1) Servlet	→	It describes all lifecycle methods of a servlet
2) ServletConfig	→	It describes the configuration parameters of a servlet (Initial parameters)
3) ServletContext	→	It describes the log events of a servlet and servlet information.
4) ServletRequest	→	It describes the request from client data (reading request)
5) ServletResponse	→	It describes the responses from the servlet (writing response to the client).

The following are the Servlet classes

Classname		Description
1) GenericServlet	→	This class is extended by a user servlet class. This class is extended if there is no any http request and http response.
2) ServletInputStream	→	It is used to read the data
3) ServletOutputStream	→	It is used to write the data
4) ServletException	→	It is used to generate an exception when an error is occur.
5) UnavailableException	→	It is used to generate an exception when a servlet is unavailable.

The following are the methods in Servlet interface.

method name		description
1) void destroy()	→	It is used to destroy the loaded servlet (terminate the servlet)
2) void init(ServletConfig)	→	It is used to initialize the servlet
3) void service(ServletRequest, ServletResponse, throws ServletException)	→	It is used to start the services (handling request and responses)

4) ServletConfig getServletConfig() → It is used to get the servlet configuration.

5) String getServletInfo() → It is used to return a servlet information (version, Copyright etc)

The following are the methods in ServletConfig Interface.

1) ServletContext getServletContext() → it is used to get a Context of servlet (log events, etc.)

2) String getInitParameter(String param) → it is used to return an initial parameter based on param name.

3) Enumeration<String> getInitParameterNames → It is used to return the initial parameter names which are in web.xml

4) String getServletName() → It is used to get a servlet name.

ServletContext:

1) object getAttribute(String attr) → Returns an attribute name based on attr

2) String getMimeType(String s) → It will returns Mime type of a servlet

3) String getRealPath(String path) → It returns an absolute path based on relative path.

4) void log(String s) → writes String s to the log file.

5) void log(String s, Throwable e) → writes the String s and the description of exception to the log file.

6) String getServerInfo() → returns the information of server

7) void setAttribute(String attr, object val) → set the attribute attr on current working object val.

The following methods are in servlet request interface.

1) object getAttribute(String attr) → returns an attribute attr on Client request.

2) String getCharacterEncoding() → returns a character encoding of a request.

3) int getContentLength() → returns the length of the Content in a request. if there is no Content it will returns -1.

4) String getContentType() → returns the type of content in a request.

5) String getparameter(String param) → returns a parameter value of a specified param.

6) Enumeration<String> getparameterNames() → returns Enumeration of parameter names

using hasMoreElements() we can check the elements in a Enumeration object.

using nextElement() we can move to the next element in a Enumeration object.

7) String getparametervalues(String param) → returns an array of parameter values.

8) ServletInputStream getInputStream() throws IoException → returns an inputStream in the form of binary data in a request.

It will throws an illegal stateException if we use getReader() before it.

9) BufferReader getReader() throws IoException → returns a bufferReader in the form of text in a request.

It will throw an illegal stateException if we use getInputStream() before it.

10) String getprotocol() → returns the protocol of a server.

11) String getRemoteAddress() → returns the String value which contains the remote machine Ip address.

12) String getRemoteHost() → returns a string value which contains the remote machine Host name.

13) String getScheme() → returns the transfer scheme of a servlet (http scheme (or) ftp scheme)

14) String getServerName() → returns name of the server

15) String getServerport() → returns port no of a server.

The following methods are in ServletResponse Interface

1) String getCharacterEncoding() → returns a character encoding of a response.

2) ServletOutputStream getoutputStream() throws IoException → returns an output stream in the form binary data it will generate an illegal StateException. if we use getwriter() before it.

3) PrintWriter getwrite() → throws IoException — returns printwrite in the form of text. it generates an illegal stateException if

we use getOutputStream() before it.

4) void setContentLength(int size) → Set the content of a length based on specified parameter size.

5) void setContentType(String type) → It will set the type of content based on specified type.

Generic Servlet class:-

It is used to implement the life cycle methods of a servlet by implementing servlet and servlet config interfaces. it will also append a log events to the log file by using following methods

void log(String s)
void log(String s, Throwable e)

ServletInputStream class:-

It is used to read the data from the client request. it is always extend and input stream class. it has the following method

int readLine(byte[] buff, int offset, int size);

ServletOutputStream class:-

It is used to write the data from the response. it always extends an output stream class. we have print() and println() methods in order to point response in a browser

Servlet Exception class:-

The servlet generates two types of exceptions

a) Servlet Exception:- The servlet generates whenever an error is occur in a servlet.

b) unavailable Exception: It is generated whenever the servlet is unavailable.

Reading Servlet parameters:-

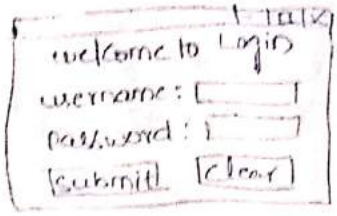
we can read servlet parameters using following methods

Enumeration getParameterNames()
String getParameters(String name);

using getParameterNames() we can retrieve names of parameters and using getParameter() we can retrieve parameter value based on specified name. With the

In the following example we have two pages, those are week6b.html and week6b.java. In week6b.html has a form which contains user name password and two buttons.
* using week6b.java we can create a servlet which, can be used to read whatever the values we have submitted in form.

week6b.html:-




```
<html>
<head>
<title> login page </title>
</head>
<body>
<center>
<h1> welcome to Login</h1>
<form name = " login" method = "Get" action = " week6b">
<table>
<tr>
<td> user Name : </td>
<td><input type ="text" name = "user" > </td>
</tr>
<tr>
<td> password : </td>
<td><input type ="password" name = "pass"> </td>
</tr>
</table>
<input type =" submit" value ="login">   
<input type = "reset" value = "clear"> </form>
</body>
</html>
```

week6b.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class week6b extends Httpservlet
{
public void doGet(HttpservletRequest req, HttpservletResponse
                                                    res)
{
res.SetContentType ("text/html");
```

```
PrintWriter out = res.getWriter();
String UserName = req.getParameter("user");
String password = req.getParameter("pass");
out.println("<html>");
out.println("<head>");
out.println("</head>");
out.println("<body>");
out.println("<center>");
out.println("<h1> welcome "+ userName +"</h1>");
if(userName.equalsIgnoreCase(password))
{
out.println("<h2> Congraluation! your details are matched
                                                    </h2>");
}
else
{
out.println("<h2> Sorry! your details are not Incorrect
                                                    matched</h2>");
}
out.println("</center></body></html>");
}
}
```

week6c.html:-

```
<html>
<head>
<title> Registration page</title>
</head>
<body>
<center>
<h1> welcome to Registration </h1>
<form name =" Registration" method = "post" action="week6c">
<table>
<tr>
<td> user Name : </td>
<td><input type ="text" name="user"></td>
</tr>
<tr>
<td> password : </td>
<td><input type="password" name = "pass"></td>
</tr>
<tr>
```

```
<td> Email : </td>
<td><input type = "text" name = "email" </td>
</tr>
<tr>
<td> mobile Number : </td>
<td> <input type = "mob no" name = "mobno" </td>
</tr>
</table>
<input type = "Submit" value = "Registration" >    
<input type = "reset" value = "clear" > </form>
</body>
</html>
```

week 6C. java:-

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class week6C extends HttpServlet
{
public void doPost(HttpServletRequest req, HttpServletResponse
                                                    res)
{
res.SetContentType("text/html");
PrintWriter out = res.getWriter();
String userName = req.getParameter("user");
String password = req.getParameter("pass");
String Email = req.getParameter("email");
String mobileNumber = req.getParameter("mob no");
 out.println("<html>");
 out.println("<head>");
 out.println("</head>");
 out.println("<body>");
 out.println("<Center>");
 out.println("<h1> welcome" + userName + "</h1>");
 out.println("<h2> Registration success! your details are
                                    as follows </h2>");
 out.println("<table border = '5'>");
 out.println("<th> username </th> <th> password </th>
                    <th> Email ID </th> <th> mobileNumber </th>
out.println("<tr> <td>" + username + "</td>");
```

```
out.println("<td>"+password + "</td>");
out.println("<td>" + email +"</td>");
out.println("<td>" +mobilenumber +"</td></tr>");
out.println("</table>");
out.println("</center></body></html>");
}
}
```

we can get a parameter value using getparameter() we can also get a parameter names using getparameterNames(). The following program demonstrate. How to get the parameter names. These program displays a parameter names as well as parameter values.

*In week6c.html we have 4 parameters. Those are username, password, email and mobileNumber. Now, we can retrieve those values using following program.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Readparam extends HttpServlet
{
public void dopost(HttpServletRequest req, HttpServletResponse
throws IOException, ServletException                    res)
{
res.SetContentType("text/html");
PrintWriter out = res.getWriter();
Enumeration e = req.getparameterNames();
while(e.hasMoreElements())
{
String pname = (String)e.nextElement();
out.print(pname + " :");
String pvalue = req.getparameter(pname);
out.println(pvalue);
}
}
}
```

hasMoreElements() returns true wherever there are more elements in Enumeration. otherwise, it returns false. The nextElement() is used to get

the nextElement from the Enumeration.

Javax.servlet.http package:

This package is used to handling request and responses of type http we have the following interfaces and classes.

1) HttpServletRequest: It is an interface to enable servlets to read data from http request. It has the following methods.

1) String getAuthType() → returns an Authentication scheme.
2) Cookie[] getCookies() → returns an array of cookie names and values.
3) long getDateHeader(String field) → returns a date header of the specified header name. The return value is in terms of milliseconds
4) String getHeader(String field) → returns the value of specified header name.
5) int getINTHeader(String field) → returns an equivalent integer value of specified header.
6) Enumeration getHeaderNames() → returns an enumerated strings.
7) String getMethod() → returns requested Method type (get (or) post etc..)
8) String getpathInfo() → returns the information of servlet path.
9) String getpathTranslated() → returns a translated servlet path.
10) String getServletpath() → returns servlet path
11) String getQueryString() → returns a query string from the URL.
12) String getRemoteUser() → returns the name of the remote user.
13) String getRequestedURI() → returns the requested URI Information.
14) StringBuffer getRequestedURL() → returns the requested URL
15) HttpSession getSession() → Create a session if there is no Existed session
16) HttpSession getSession(boolean value) → if value is true it will Create a new session for every new user.
17) Boolean isRequestedSessionIdFromCookie() → it will returns true if the session id from cookie.
18) Boolean isRequestedSessionIdFromURL() → it returns true

session id from URL

19) boolean isRequestedSessionIdValid() → it returns true if the sessionId is valid.

20) int getRequestedSessionId() → it returns an integer which contains a session id

2) HttpServletResponse: It is an interface to enable servlet to write data to the http response. It has the following methods.

1) void addCookie(Cookie c) → It will add cookies to the browser.

2) boolean containsHeader(String field) → It is used to check the specified header whether it is contains in url or not.

3) String encodeURL(String url) → It determines whether the url is encoded or not. if it is encoded it will returns a modified url. otherwise, it returns url.

4) String erodeRedirectURL(String url) → It determines whether redirected URL is encoded or not. if it is encoded. it returns modified url. otherwise, it returns url. All url's are passed to send redirect().

5) void sendRedirect(String url) → It will send to the redirected url based on specified client

6) void sendError(int c) → It is used to send an integer Error Codes.

7) void sendError(int c, String s) → it is used to send an error codes and as well as messages to the client.

8) void setDateHeader(String field, long msec) → it is used to set Date header.

9) void setHeader(String field, String value) → it is used to set Header name and value.

10) void setIntHeader(String field, int code) → It is used to set an integer codes to the header

11) void setStatus(int code) → It is used to set Status of header.

Http Session:

It is an interface to allow a servlet can read and write session data we have the following methods

1) boolean isNew() → it return true whenever the session is new one otherwise false.
2) void invalidate() → it is used to close the session or invalidate the session.
3) object getAttribute(String Attr) → it is used to get the attributes of sessions (sessionId, creation time, last modified time and no. of digit counts etc.)
4) void setAttribute(String attr, String value) → it is used to set session attribute.
5) void removeAttribute(String Attr) → it is used to remove a session attribute
6) Enumeration getAttributeNames() → It is used to return the list of attribute names.
7) Long getCreationTime() → It is used to return the creation time of a session.
8) long getLastAccessTime() → It is used to return the last accessing time.
9) int getId() → it is used to return a session id.

Cookie:-

A Cookie is a state of user (user data) these are stored in client web browser. Generally the Cookies are used to track the user activities for Example, In online store the user enter his name, address, phone number etc. So, whenever he has entered those data will be stored as Cookies in his web browser. If the same person) entered into the same online store no need to entered his details again. The Cookies are always stored in web browser. These Expires time is based on their ages. we can set the maximum age of cookie using ~~set~~ setMaxAge() in terms of seconds. The following information is generally maintained in Cookies.

1) The name of the Cookie
2) The value of the Cookie
3) Domain and Path of the Cookie
4) Age of the Cookie

we can create a Cookie using Cookie class in servlets.

syn:- Cookie(String name, String value);

* we can add the created cookies to the browser using addCookie().

syn:- void addCookie(Cookie)

for Example, Cookie c = new Cookie("username", "ss");
response.addCookie(c);

we have the following methods in Cookie class.

1) Object clone() → it returns a copy of object
2) String getComment() → it returns a comment of a cookie.
3) void setComment(String s) → used to set a comment to the cookie
4) String getDomain() → returns a domain
5) void setDomain() → it is used to set domain.
6) int getMaxAge() → it returns the maximum age of cookie
7) void setMaxAge(int sec) → used to set the maximum age of cookie (in terms of seconds)
** 8) String getName() → it is used to return the cookie name.
9) ~~void String~~ setName
9)** String getValue() → it is used to returns the Cookie value.
10) int getVersion() → returns the version of cookie
11) void setVersion(int) → it is used to set version of cookie
12) String getPath() → returns the path.
13) void setPath(String s) → it is used to set the path
14) boolean getSecure() → returns true if the cookie is secure. other wise false.
15) void setSecure(boolean secure) → it is used to create a secured cookie.
16) boolean isHttpOnly() → it returns true if the cookie is httponly type.

void SetHttpOnly(boolean http) → if http is true it will add an httpOnly attribute to the Cookie if http is false it will remove the http only attribute from the Cookie.

week 7a:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
Public class week7a extends Httpservlet
{
Public void doGet(HttpservletRequest req, Httpservlet
                                    Response res)
throws IoException, servlet Exception
{
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html><head><title>Cookie</title></head>");
out.println("<body>");
out.println("<h1>Create Cookies to send to the browser
                                    </h1>");
out.println("<form name='Cookie' method='post' action=
                                    'week7a'>");
out.println("<table><tr><td> username:</td><td>");
Out.println("<input type="text" name="user" </td></tr>");
out.println("<tr><td> password:</td><td>");
Out.println("<input type='password' name='pass'></td></tr>
                                    <table><br>");
Out.println("<input type='submit' value='login'>  
            <input type='reset' value='clear'>");
Out.println("</form></body></html>");
String user = req.getparameter("user");
String pass = req.getparameter("pass");
if(user!=null && pass!=null)
 {
  Cookie CK = new Cookie(user, pass);
   CK.setMaxAge(30);
   res.addCookie(CK);
out.println("<h2> you just sent the following Cookies</h2>");
out.println("<p> username:"+user);
```

```
out.println("<br> Password:" +pass);
}
Cookie [] Cookies = req.getCookies();
if(Cookies != null && Cookies.length > 0)
{
out.println("<h2> your browser sending the following
                                    Cookies </h2>");
for(int j=0; j<Cookies.length;j++)
 {
 Cookie C = cookie[j];
out.println("<p> userName:" +C.getName());
 out.println("<br> Password:" +C.getValue());
 }
}
else
{
Out.println("<h2> your browser is not sending any
                                    Cookies </h2>");
 }
}
public void dopost( HttpServletRequest req, HttpServletResponse
{                                                    res)
dopost doGet(req, res);
  }
 }
```

Http Servlet Class:

It is a class which is used to perform various http request and responses it is always extended by Generic servlet. It has the following methods

1) void doGet(HttpServletRequest req, HttpServletResponse res) throws IoException, ServletException → Handling HttpGet

2) void doPost(HttpServletRequest req, HttpServletResponse res) throws IoException, ServletException → Handling HttpPost

3) void doDelete(HttpServletRequest req, HttpServletResponse res) throws IoException, ServletException → Handling HttpDelete

4) void doPut(HttpServletRequest req, HttpServletResponse res) throws IoException, ServletException → Handling HttpPut()

6) void doHead(HSR req, HSR res)
throws IOE, SE → Handling http head

7) void doOptions(HSR req, HSR res)
throws IOE, SE → Handling http options request

8) void doTrace(HSR req, HSR res)
throws IOE, SE → Handling http trace request

2) long getLastModified(Http ServletRequest req) → returns the time when the requested resource is last modified.

9) void service(HttpServletRequest res, HttpservletResponse res)
throws IOE, SE → It is used by the server to handle all http request.

Handling HttpRequest and Responses:

We can handle Http Request and responses by using the following method.

1) doGet() → it is used to handle an http getRequest.
2) doPost() → it is used to handle an http postRequest.
3) doDelete() → it is used to delete particular resource at server side based on http deleteRequest
4) doHead() → it is used to display a header information like host header date, header port Number and creation time etc. based on http Head Request.
5) doput() → it is used to put a particular resource at server side based on http put Request.
6) doTrace() → it is used to trace the user information based on http Trace Request.

But generally, will use doGet and doPost methods to handle httpRequest and Responses.

Handling Http GET Request:-

We can handle Http GET Request using doGet() while we are using Get() in form we have to call doGet() in servlet to handle Http Get Request while we are using doGet(). The parameters of http GetRequest are as part of url.

For Example, if we are submitting a form which contains an user name then the user name will appear in the url as follows

http://localhost:8080/cse/week7a?user=csc

(directory name: cse; servlet name: week7a)

Ex:-
```
<html>
<body>
<form name="Sample" method="Get" action="simple">
userName : <input type="text" name="user">
<input type="submit" value="Submit">
</form>
</body>
</html>
public class demoGet extends HttpServlet
{
public void doGet(HttpServletRequest req, HttpServlet
                                    Respons res)
throws IoException, servletException
{
res.setContenEType("text/html");
printWriteH out=res.getWriteH();
out.println("welcome "+req.getparameteH("user"));
}
}
```

Handling Http post Request:

We can handle Http post Request by using doPost() in servlet while sending post request we have to mention the form() type as post type. So, that in servlet we have to use doPost() to handle HttpPost Request.

* If you are using doPost() in servlet the HttpPost Request parameters are not as part of url. This parameters will not appeared in the url. The url simply as follows.

http://localhost:8080/cse/week7a

Ex:
```
<html>
<body>
<form name="Sample" method="post" action="simple">
userName : <input type="text" name="user">
<input type="submit" value="submit">
</form>
</body>
</html>
```

```
Public class demopost extends HttpServlet
{
Public void dopost(HttpServletRequest req, HttpServletResponse
                                                    res)
throws IOException, ServletException
{
res.SetContentType("text/html");
PrintWriter out = res.getWriter();
out.println("welcome" + req.getparameter("user"));
}
}
```

Session tracking:

Http is a stateless protocol so that it can't maintain data (state) of the user but in servlets we can maintain a state of user by creating a session for user without sessions. if any user sends an http request those requests are treated as independent but in sessions those requests are treated as dependent.

We can create a session using GetSession(). It has the following syntax

```
HttpSession getsession();
Httpsession getSession(boolean);
```

getsession: This method creates a new session

getsession(true) → It will create a new session if there is no existed session.

getsession(false) → If they are existed session these method returns the reference of existed session.

We can set attributes using SetAttribute() and we can get attributes using getAttribute() and we can also get a session id using getId(). we can access creation time using getCreationTime() and we can also get the last accessing time using getlastAccessingTime().

Ex:- week7b.html as in before page week6b.html

week7b.java

```
public class week7b extends HttpServlet
{
public void doPost(HSR req, HSR res)
                throws IOException, servletException
{
res.SetContentType("text/html");
PrintWriter out = res.getwriter();
 String user = req.getparameter("user");
 String pass = req.getparameter("pass");
 HttpSession Session = req.getSession();
  Integer count =new Integer(1);
 if (session.isNew())
 {
 Session.SetAttribute("username", user);
 out.println("<h1> welcome to session </h1>");
 else
 {
out.println("<h2> welcome back to session </h1>");
 count = (Integer) session.getAttribute("vcount");
 count = count+1;
}
Session.setAttribute("vcount", count);
 out.println("<center><a href="week7bb"> Logout </a></center>");
out.println("<center><h1> session Information </h1></center>");
 out.println("<center>");
 out.println("<table border='5'>");
 out.println("<th> Session Info </th> <th> value </th>");
out.println("<tr> <td> username </td> <td>"+ session.getAttribute
                                ("username")+"</td> </tr>");
out.println("<tr><td> id </td><td>"+ session.getId()+"</td>
                                                </tr>");
out.println("<tr><td> creation Time </td> <td>"
            +new Date(session.getCreationTime()))+"</td></tr>");
out.println("<tr><td> last Accessed Time </td> <td>"
            +new Date(session.getlastAccessedTime()+
                                "</td></tr>");
out.println("<tr><td> No. of visits </td><td>"
            +session.getAttribute("vcount")+"</td></tr>");
}
}
```

week7bb.java

```
Public Class week7bb extends HttpServlet
{
Public void Service(HSR req, HSR res)
 throws IOE, SE
{
res.SetContentType("text/html");
PrintWriter out = res.getWriter();
HttpSession session = req.getSession(false);
 session.invalidate();
out.println("<center> <h1> you are logged out successfully
                                                <h1></center>");
out.println("<a href='week7b.html'> Login<b></center>");
}
}
```

Accessing database with JDBC using Servlets:

1) we have to Register drivers and Create a Connection with JDBC. In servlets we can create these two steps in first method of servlet lifecycle i.e; init()

2) In doPost() or doGet() we can process the http request or responses and we can also create and execute Sql statements.

3) we can close the Connections and Statements inside destroy()

write a java program to process the data base activity using Servlets

```
import java.io.*;
import javax.Servlet.*;
import javax.Httpservlet.*;
import javax.sql.*;
Public Class week7c extends HttpServlet
{
Connection Con = null;
PreparedStatement ps;
ResultSet rs;
Public void init() throws ServletException
{
try
{
```

```
Class.forName("oracle.jdbc.oracleDriver");
con = DriverManager.getConnection("jdbc:oracle:thin:
                                  cse/cse@localhost:1521/xe");
}
catch(Exception e)
{
System.out.println(e);
}
}
public void doPost(HttpServletRequest req, HttpServletResponse
                                                    res)
throws IOException, ServletException
{
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<html><head><title>Registration</title></head>");
out.println("<body><center>");
out.println("<h1>Registration success! The details as
                                        follows</h1>");
out.println("<table border='5'>");
out.println("<th>userName</th><th>password</th><th>
            email</th><th>phone number</th>");
try
{
PS = con.prepareStatement("insert into Reg values(?,?,?,?)");
ps.setString(1, req.getparameter("user"));
ps.setString(2, req.getparameter("pass"));
ps.setString(3, req.getparameter("email"));
ps.setString(4, req.getparameter("phone"));
ps.executeUpdate();
PS = con.prepareStatement("select * from Reg");
RS = PS.executeQuery();
while(RS.next())
{
out.println("<tr><td>"+RS.getString(1)+"</td>");
out.println("<td>"+RS.getString(2)+"</td>");
out.println("<td>"+RS.getString(3)+"</td>");
out.println("<td>"+RS.getString(4)+"</td></tr>");
}
out.println("</table>");
```

```
out.println("<h2> <a href="week7c.html"> New Registration
                                    </a></h2>");
out.println("</center></body></html>");
}
catch(SQLException e)
{
S.O.P(e);
}
}
public void destroy()
{
try
{
con.close();
ps.close();
rs.close();
}
catch(SQLException e)
{
S.O.println(e);
}
}
}
```

week 6c.html as week7c.html