**SERVLETS**

-A java program uses to translate the dynamic content that executes on a webserver(internet & intranet).

**Steps to Create J2EE Simple Web Application**

1)create the dynamic web project to develop **web application**.

2)created the first servlet by extending the **HttpServlet class**.

->copied the servlet.jar file to WEB-INF/lib folder.

->override doGet() method to HttpServlet.

3)created the plain **HTML** page.

4)added some configuration to **web.xml**.

5)build the project & generate the war file(export->war file).

6)copied the generated **war file** to webapps directory in server.

7)start the apache **web server**.

8)accessed the developed web application using web browser by typing **url**(<http://localhost:8080/projectname/javaclass> or html file.

9)accessed the html file**(static page)**.

10)accessed the servlet file**(dynamic page**).

**Web Application**

->web application is an application that is access over the network such as internet & intranet.

**Dynamic Page**

Dynamic pages are created at the time of request .this don’t exist before the request.

**Static Page**

->Static page is a web page that is deliver to the user exactly as it present at the web server.

->in case of static pages every client will see the same thing. they exist before the request.

**Build Process**

->it is the process of converting source code file and other necessary dependents into stand alone software archive**(war file)** that can be run on any operational environment.

**Jar File(Java Archieve)**

**->**it is typically use to aggregate many .class files and other necessary files(text files,images,properties files etc) into one file to distribute application software are libraries on java platform.

**War(web Archieve)**

->it consists of collections of source files(.class,servlets,jsp etc) and other necessary files(xml,properties files,html,image files etc) and other resources (dependent jar file) that together constitutes the web application.

**Ear files(Enterprise Archieve)**

**->** collection of .jar files, resources, classes, and multiple Web applications.

**Deployment Process**

**->**it is the process where a developed application is installed in an operational environment.

**Web Browser**

**->**it is a software that knows how to communicate with server.

->it allows user to make a request to web server and display the response given by the web server.

->it is interpreting the HTML code and rendering the information present in HTML.

**URL(Uniform Resource Locater)**

every resource on the web has its own unique address in the form of URL.

**Syntax**-protocol://domain:port/path?query\_string#fragment\_id

->**Protocol**

-tells the server, which operation protocol is used.

-most common protocol is HTTP(hyper text transfer protocol) & HTTPS(hyper text transfer protocol secure).HTTP encrypts any information we provide so it can’t be understood by hackers,who try to interrupt the page.

**->Domain**

-it’s a unique reference that identify in application in the network(internet/intranet).

**->Port**

-it’s a sixteen(2^16-65536) bit number that identifies specific software program on the server hardware.

-port represents a logical connection to a particular pieces of s/w running on the server h/w.

-wellknown port number are

8080-apache tomcat,3306-mysql db server,80-HTTP,443-HTTPS.

**->Path**

-its typically refers the combination of directory & file on the web server.

**->Query\_String**

**-**it is the commonly found in URL of dynamic pages.

-it is represented by “?” followed by one or more parameters,which are separated by “&” character.

**->Fragment**

-a fragment is a internal page reference named anchor.

-usually appears at the end of the URL begin with hash(#) character followed by an identifier.

-it refers to a section within a webpages.

Example-http://tomacat.apache.org/tomcat-6.0-doc/manager-howto.htm#undeploy\_an\_existing\_Application

**HTTP(Hyper Text Transfer Protocol)**

**->**protocol is a set of tools.

->browser and server both will communicate using HTTP protocol. the client sent Http request and server responds with a http response.

->the name implies most of the time ,http response contains html.

**HOW HTTP WORKS**

->HTTP runs on top of TCP/IP.

->transmission control protocol(TCP) is responsible for making sure that file send from one n/w node to another node as a complete file at destination,even though file is splited into multiple parts(Data Packets) when it is send.

->internet protocol(IP) is the underline protocol of TCP that moves the data packet from one n/w node to another node.

->http is another n/w protocol that has web specific features but depends on TCP/IP to get complete request and response from one node to another.

HTTP REQUEST

SERVER

CLIENT

HTTP RESPONSE

Key elements of HTTP request are: key elements of HTTP response are

1)HTTP method 1)status code

2)the page to access(URL) 2)response content type(html,image,text etc)which is

3)form parameters if any. Also known as MIME type.

3)actual content.

**HTTP METHODS**

->HTTP has eight different methods.servlets has implementation for these methods(excluding one).

|  |  |  |
| --- | --- | --- |
| **HTTP METHODS** | **RELATED SERVLET METHODS** | **DESCRIPTION** |
| HEAD | doHead() | Allows client to see only header. |
| TRACE | doTrace() | This method echoes trace the received resource so that client can see changes(it any) happened to the request.used for trouble shooting. |
| POST | doPost() | Allows client to submit data to be processed.which may resulting modifying the data in DB. |
| PUT | doPut() | Allows client to place a fileon the server and similar to sending file by FTP. |
| DELETE | doDelete() | Allows client to remove file from server. |
| OPTIONS | doOptions() | This method determines which http methods does server support and returns an appropriate response. |
| GET | doGet() | Allows a client to request the specified resource .doGet() method should only retrieve the data and should have no other effect. |
| CONNECT | No implementation. | There is no mechanism in servlet API for handling CONNECT hence it is not part of HttpServlet. |

**POPULAR HTTP STATUS CODE**

200-OK,404-FILE NOT FOUND,500-INTERNET SERVER ERROR

1xx-INFORMATION

2xx-OK

3xx-REDIRECTION

4xx-CLIENT ERROR

5xx-SERVER ERROR

**DEPLOYMENT DISCRIPTOR(DD)**

->DD is an xml file named as web.xml.it resides in application WAR file under the WEB-INF directory.

->DD describes the configuration for an application.

->when the webserver receives a request it uses the DD to map the URL,after request to the code that handles the request.

->below are the few benefits to get out of DD:-

-web.xml defines mapping between URL path and the servlet that handle request with those path.the web server uses the configurator to identify the servlet to handle a given request and call the servlet method that corresponds to the request.

-we can also specifies initialization parameters for that servlet.

<servlet>

<servlet-name>ServletSample</servlet-name>

<servlet-class>com.jspider.servlet.ServletSample</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ServletSample</servlet-name>

<url-pattern>/ServletSample</url-pattern>

</servlet-mapping>

->below are the biggest advantages is

1)security2)maintainbility3)pass the necessary information to the servlets.

->using DD we can customize server response where an error occours.the server can display an alternative page for a particular http status code or when a servlet throws a particular java exeception.

<error-page>

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

<location>/error</location>

</error-page>

**WEBSERVER**

->a webserver takes a user request and gives something back to the user.

->user request for a resource via webserver. the webserver takes this result, find the resources and returns it to the users.

->webserver are good at serving the static content.

->if the webserver can’t able to find the resources requested by the user then it give 404 not found error.

**LIMITATION OF WEBSERVER**

->webserver as the name implies served things and it won’t do any computation on the things it serve.

->following of the two things that webserver alone can’t do.

-generating dynamic content.

-interacting with database.

->in the above scenario ,it takes the help of other technologies (PHP,PYTHON,SERVLETS etc) to do the job.

**SERVLETS**

->it’s a java program used to generate dynamic content.

->they are not to tied to any specific protocol but most often used with HTTP protocol.

->servlets don’t have main method ,to run a servlets servlet container must be used.a servlet container (web container) is essentially a component of the webserver that interacts with servlets.a servlet container is responsible for managing life cycle of a servlets,mapping a URL to a particular servlets etc.

**ADVANTAGES OF SERVLET CONTAINER**

1)communication support-the container provides an easy way for servlet to talk to webserver.

2)life cycle management-the container controls the life and death of servlets.

3)multi threading support-the container automatically create a new thread for every request it receives. this thread dies when it is done with HTTP service method.

4)declarative security-with web.xml,which is used by servlet container,which is configure or modified security and other parameter without changing anything into source code.

5)jsp support-the container takes care of translating jsp into servlet.

**SERVLET HIRARCHY**

interfaces

SERVLET CONFIG

SERVLET

servlet:defines servlet lifecycle methods.

Servlet config:defines method which

provide initialization information to the

servlet.

->abstract class

GENERIC SERVLET

->not specific to any Protocol.

->abstract class

HTTP SERVLET

->only handles HTTP & HTTPS protocol.

Concrete class which overrides one of the

MYSERVLET

HttpServlet service() method[doXXX()]

**GENERIC SERVLET**

1)it’s a protocol independent servlets.

2)because service() method present in GenericServlet class is declared as abstract.genericServlet is an abstract class.

3)syntax of service() method.

Public abstract void service(ServletRequest req,ServletResponse res)throw ServletException,IOException;

->press f3 to see the class & ctrl,shift & t to see inside that.

**HTTPSERVLET**

1)it is an abstract class needs to be subclass to handle http request.

2)a subclass of HttpServlet can override either service() method directly or atleast one http method.usually one of this

->doGet() for http GET request.

->doPost() for http POST request.

3)if we won’t override doXXX() method then default implementation in HttpServlet is invoked which intern return the error message along with the status code(405).

4)there is almost no reason to override the service() method.the service() method present in httpServlet class to handle Http request by dispatching them to the handler method for each http request type(doXXX() method).

**WHY HttpServlet IS ABSTRACT CLASS**

1)HttpServlet defines no abstract method however that designers of servlets didn’t want to simply use HttpServlet as our servlet class.thus to force us to extends HttpServet class into our servlet class thatsy class is marked as abstract.

**HOW DYNAMIC REQUEST IS HANDLED IN WEBSERVER**

SERVER MACHINE

SERVER

Service(req,res)

1

WEBSERVER APPLICATION

CLIENT

6

5

2

3

SERVLET

<html>

</html>

SERVLET CONTAINER

THREAD

4

SERVICE()

doXXX()

<html>

</html>

1)client request for a dynamic resource.lets say dynamic page with current date & time.

2)webserver doesn’t find the resource within itself then it handover that request to servlet container.

3)->servlet container creates ServletRequest & ServletResponse object.

->it finds the correct servlet based on URL using web.xml to handle this request.

->container creates a thread for that request and invoke service() method(through run() method) by passing ServletRequest & ServeletResponse object).

->depending on the HTTP method in the request the service method calls relative doXXX() method.let say in our case doGet() method.

4)doGet() method constructs brand new html page along with current date & time and stuff the page into a response object.the container converts the response object into a http response and send it back to the webserver.then container deletes the request & response object.

5)webserver gets the response from servlet container as for as webserver concern the html from the container is a static page.

6)client get the requested resource that is html page with current date & time.

**DIFFERENCE BETWEEN GENERIC SERVLET & HTTPSERVLET**

|  |  |
| --- | --- |
| **GENERIC SERVLET** | **HTTP SERVLET** |
| Protocol independent. | Protocol dependent.support only http & https. |
| Abstract class because service() method declared as abstract. | Abstract class but non of the methods are abstract. |
| If we extends GenericServlet class then we have to provide implementation for service() method. | There is almost no reason to override service() method hence we have to worry about overriding doXXX() methods. |
| It belongs to java.servlet package. | It belongs to java.servlet.http packages. |

**SERVLET REQUEST**

->interface.

ServletRequest I

->provides request information to a particular servlet

->a servlet request provides request data including parameter

Name & values,attributes and an input stream.

->interface.

->it extends the ServletRequest interface to provide request

HttpServletRequest I

information for HttpServlet.

**SERVLET RESPONSE**

->interface.

ServletResponse I

->it assist’s the servlet in sending a response.

-> to send character data use PrintWriter.

->to send binary data use ServletOutputStream

->to create multiple response (binary & text data)use

servletOutput Stream and manage the character sections

HttpServletResponse I

manaually.

->interface

->it extends the ServletResponse interface to provide Http

Specific functionality in sending a response.

->it has methods to access Http headers & cookies.

**CONTENT TYPE(MIME TYPE)**

1)content type are sometimes referred as **Multi purpose Internet Mail Extensions(MIME)** type,tells the browsers that what type of content its going to receive.so that it can prepare itself to handle the response data.e.g.open a media player to handle media content.open adobe reader to handle PDF contents.

2)default content type is text or HTML.

3)e.g text/html,text/pdf,video/quicktime,image/jpeg,application/x-zip.

**SERVLET LIFECYCLE**

CONTAINER LOADS THE

SUCCESS SERVLET ,WHENEVER IT RCV REQ

FAILURE

SUCCESS

SUCCESS

REQ

RES

WEB.XML

INSTANTIATE,INVOKE DEFAULT CONSTRUCTOR

I

SERVLET CONTAINER

INITIALIZE,INVOKE INIT() METHOD

DoXXX()

SERVICE,INVOKE SERVICE() METHOD

DESTROY,INVOKE DESTROY() METHOD

Life cycle of a servlets is controlled by servlet container.when a request comes to a servlet then container performs following steps.

1)load the servlet class and executes the static block of the servlet(if any).

2)create the instance of the class by invoking default constructor.

3)initializes the servlet by calling init() method after the servlet instance is created but before servlet can service any client request.

4)invoke the service() by passing request & response object.

5)when a container wants to unload the servlet.it calls to destroy() method before unloading the servlet from service.

**INIT() METHOD**

**1)SYNTAX-**

Public void init() throws ServletException

Public void init(ServletConfig config) throws ServletException

**2)How It Works**

1)container calls init() method after the servlet instance created before servlet can service any client request.

2)init() method gives us a chance to initialize the servlet before handling any client request like opening a DB connection,read data from properties file etc.

3)during initialization ,servlet has access to two key objects.

ServletConfig

ServletContext

4)this method is called only once in the life cycle of a servlet.

5)if we don’t override init() method then one from the GenericServlet is called.

**3)Do We override the method?**

Possibly,if we have a initialization code then this may override this method.

**4)EXAMPLE**

Version 1- Public void init() throws ServletException{//initialization code}

Version 2-Public void init(ServletConfig config) throws ServletException

{super.init(Config);

// initialization code}

1)first line of version 2 is init() method should be “super.init(Config)”.if an error occurs during an initialization phase then servlet throws a ServletException.then servlet container release the servlet instance and put the garbage collector.in this status destroy() method is not called.

**SERVICE() METHOD**

**1)SYNTAX-**

Public void service(ServletRequest req,ServletResponse res) throws ServletException ,IOException;

**2)HOW IT WORKS**

1)when a request comes,the container starts the threading and call service() method of the servlet.

2)this method looks at request,gets the Http method and invokes the mathching doXXX() method.

3)this method is called whenever client request come in.that is the method is called one or more times in servlet life cycle.

**3)DO WE OVERRIDE THIS METHOD?**

NO,we should not override the service() method.our job is to override doXXX() method & lets service() method is worry about calling right doXXX() method.

**DESTROY() METHOD**

**1)SYNTAX-**

Public void destroy();

**2)HOW IT WORKS**

1)when container wants to unload the servlet,it calls destroy() method before unloading the servlet from service.

2)the decision of when to destroy servlet instance the rest on the shoulder of container.

3)this method is called only once in the life cycle of a servlet.

4)if we don’t override the destroy() method then one from the GenericServlet is called.

**3)DO WE OVERRIDE THIS METHOD?**

POSSIBLY,if we want to perform some cleanup operation then we can make use of this method.

**CAN SERVLET HAVE CONSTRUCTOR?**

1)we can definitely have default constructor in our servlet.even we can use this constructor for initialization .but this type of approach is not so common.the common approach is to used init() method.

2)it should not only have parameterized constructor in our servlet.in this case servlet initialization fails.but we can have parameterized constructor along with default.in this case servlet will get successfully initialized.

**WHAT DETERMINED REQUEST IS OF TYPE GET OR POST?**

1)a simple hyperlink will always means a GET request.

<a href=”http://localhost:8080/studentapp/CurrentDateTime”>click here</a>

2)method =”post” in the form means post request.

3)method=”get” in the form means get request.

4)no method declaration in the form means get request.

**DIFF. BETWEEN doGet() & doPost METHOD**

|  |  |
| --- | --- |
| **doGet() method** | **doPost() method** |
| 1)this method allows a client to request for a resource. | 1)allows a client to submit data to be processed. |
| 2)it is a default method. | 2)its not default method.we have to explicitly declare method=”post” in html form. |
| 3) not secure because form data will get appended to URL. | 3) secure because data will be part of HttpRequest.. |
| 4) limited amount of information,we can send(1024 char). | 4) unlimited amount of data or size of the data we can send. |
| 5) we can’t send or upload file using get(resume upload). | 5) we can send one or more file using post. |
| 6) we can bookmark the get request. | 6) we can’t bookmark. |

**String[] getParameterValue(String arg0) method**

1)this method is present in the interface ServletRequest and returns an array of string containing all of the values of the given request parameters or return null ,if the parameter does.t exist.

**String getParameter(String arg0) method**

1)this method is also present in servletRequest interface and the value of all the request parameter as a string or returns null if the parameter doesn’t exist.

**ServletContext**

1)it is an interface use by servlet container to pass information to all the servlets ,which are part of a web application.hence there will be only one servlet context for the entire web application and the component of the web application can share it.

2)this allows application to have initialization parameters specified outside the code and they can be changed without touching the source code.

3)the configuration information(one or many) is declared in web.xml under <context-param></context-param>.

4)we can get the initialization parameters at runtime but we can’t set them.

**ServletConfig**

1)it is an interface use by servlet container to pass information to a particular servlet during its initialization.

2)this allows application to have initialization parameters specified outside the code and they can be changed without touching the source code.

3)this configuration information(one or many) is declare in web.xml under <init-param></init-param> inside the <servlet></servlet> tag

4)we can get this initialization parameter at run time but we can’t set that.

**DIFF BETWEEN ServletConfig & ServletContext**

|  |  |
| --- | --- |
| **ServletConfig** | **ServletContext** |
| 1)one per servlet. | 1)one per web application. |
| 2)ServletConfig reference is obtained by calling getServletConfig() method. | 2)it is obtained by calling getServletContext() method. |
| 3)on web.xml ServletConfig parameters are declared under <init-param></init-param>. | 3)declared under <context-param></context-param> |
| 4)ServletConfig holds the reference of a ServletContext. | 4)ServletContext doesn’t holds the reference of a servletConfig. |

**FULLFILL(DIRECT SERVLET)---BORROW(ONLY INTERNAL SERVLET)-----REDIRECT(INTERNAL SERVLET OR EXTERNAL PAGE)**

**REDIRECT**

1)if we don’t want to respond the request then we can redirect the request to different URL.it can be internal or external URL.

2)when we redirect the request,it always be a get request.

3)redirect makes browser to do the work when servlet .when servlet does the redirect its like asking browser to call someone else.the user see the new URL to browser.

4)we can’t call sendRedirect() after the response is committed.in other words if we have already written something to screen its too late to do the redirect.in this case we will get illegal state exception.

Res.sendRedirect(“http://www.google.com”);

Res.sendRedirect(“http://localhost:8080/studentsapp/FormExample”);

Res.sendRedirect(“FormExample”);

**RsquestDispatcher**

1)this interface is present in javax.servlet package and contains following two methods.

->void forward(ServletRequest req,Servletresponse res);

->void include(ServletRequest req,Servletresponse res);

**Forward()**

1)this method forwards a request to another resources on the same servlets.that resources can be servlet,jsp page or simple html page.

2)if servlet don’t want to handle the request within itself ,it can forward that request to an another request.

3) when servlet forward the request,it will invoke corresponding method at destination.i.e. if servlet is forwarding the request from doGet() then it will invoke doGet() at destination and if servlet is forwarding the request from doPost() then it will invoke doPost() at destination.

4)when servlet forwards the request to another servlet and if the corresponding doXXX() method doesn’t exist in that servlet then we get error message at the response.

5)in the case of forward,browser will not have any clue on what went behind the screen.browser required for something and get it.

6)it doesn’t matter how its work i.e.everythings happens internally within the servlet container.

7)in case of forward(),user doesn’t see a new url in the browser.

8)we can’t cal forward() method after the response is committed.in this case user will get illegal state exception.

9)to forward the request,call forward() method on RequestDispatcher object.there are two ways we can get the RequestDispatcher object.

**->from Servletrequest object**

RequestDispatcher dispatcher=req.getRequestDispatcher(“FormExample”);

dispatcher.forward(req,res)

**->from ServletContext object**

Servletcontext context=getServletContext();

RequestDispatcher dispatcher=context.getRequestDispatcher(“/FormExample”);

Dispatcher.forward(req,res)

In this case URL should start with “/”.otherwise we get illegalArgumentException(405).

**Include() method**

1)this method includes the response from the given resources(servlet,jsp,html) within the caller response.

2)when we include the content of one servlet into one or another then it includes the content or response of corresponding method in that servlet i.e. if we are invoking include in doGet() then include the contents of doGet() method of another servlet and if we are invoking include in doPost() then include the contents of doPost() method of another servlet.

3)when we include the contents of one servlet into another and corresponding doXXX() method doesn’t exist in another servlet then it will not throws any exception.in this case just ignore include() statement.

RequestDispatcher dispatcher=req.getRequestDispatcher(“FormExample”);

dispatcher.include(req,res)

**DIFFERENT WAYS TO PASS THE IFORMATION TO SERVLET**

1)using Form.

2)using ServletContext & ServletConfig initialization parameters(init-param).

3)using query parameters

http://localhost:8080/studentApps/CurrentDateTime?fname=praveen &lname=singh.

String fname=req.getParameter(“fname”);

String lname=req.getParameter(“lname”);

http://localhost:8080/studentApps/CurrentDateTime?vehicle=car & vehicle =bus.

String[] vehicle=req.getParameterValues(“vehicle”);

4)using attributes

Install fiddler2

**DIFFERENT BETWEEN SendRedirect() & forward()**

|  |  |
| --- | --- |
| **sendRedirect()** | **forward()** |
| 1)are redirect. | 1)are redirect dispatcher. |
| 2)redirect makes browser to do the work i.e.it happens at browser side. | 2)browser will not have any clue on what went behind the scene i.e it happens at the server side. |
| 3)in case of redirect users see the diff. url in the browser. | 3)url of browser doesn’t change. |
| 4)we can redirect the request to  ->within the application  ->to an another application display on the same server  ->to an external application. | 4)we can forward the request only within the application. |
| 5)when we redirect the request ,it will always be a request of type GET. | 5)when we forward the request,then it invokes the corresponding method at destination. |
| 6)sendRedirect() is presenting HttpservletResponse. | 6)forward() is representing RequestDispatcher which can be obtained by HttpServletRequest or ServletContext. |
| 7)slower in operation. | 7)faster in operation. |

**ATTRIBUTE**

1)attributes are simply name-value pair where name is string and value is the object.which means that any java object can be an attributes value.

2)attributes are used to share information from one servlet or jsp to another servlet or jsp.

**ATTRIBUTE TYPES**

There are three types of attribute.

1)context attribute(application scope)

2)request attribute(request scope)

3)session attribute(session scope)

->this three attribute can be get or set by using following object respectively.

1)javax.servlet.ServletContext

2)javax.servlet.ServletRequest

3)javax.servlet.http.HttpSession

->the following methods are present in above three object which can be used to get or set attribute.

1)void setAttribute(string arg0,Object arg1)

->Binds an object to a given attribute name.if the attribute with the same name exists then this method will replace the existing object with the new object.

2)Object getAttribute(String name)

->This method returns an attribute with the given name or returns null if there are no attribute by that name exists.

3)void removeAttribute(String name)

->remove the attribute with the given neme.after removal subsequent calls to get the attribute to retrieve the attribute value to return null.

**DIFFERENT BETWEEN CONTEXT,REQUEST & SESSION**

|  |  |  |
| --- | --- | --- |
| **CONTEXT** | **REQUEST** | **SESSION** |
| 1)available to entire application. | 1)available to specific request. | 1)available to specific session. |
| 2)context scope are lifetime of the application. | 2)request scope are lifetime of the thread that is handling this request. | 2)session scope. |
| 3)context attributes are handled by ServletContext interface. | 3)ServletRequest. | 3)HttpSession. |

**DIFFERENT BETWEEN AN ATTRIBUTES AND PARAMETERS**

|  |  |
| --- | --- |
| **ATTRIBUTE** | **PARAMETER** |
| 1)return type is object. | 1)return type is string. |
| 2)we can set attribute. | 2)can’t set parameters. |
| 3)three types of attribute.  ->context  ->request  ->session | 3)three types of parameter.  ->context init parameter  ->servlet init parameter.  ->request parameter(form data,query string) |

**COOKIES**

HTTP REQUEST

**CLIENT**

**SERVER**

HTTP RESPONSE

|  |  |
| --- | --- |
| **CLIENT** | **SERVER** |
| Whenever browser makes a request to some server,it always checks whether it has any cookies within it,which are tagged to that server if yes,then it will send those cookies to server along with the “request”.  1)lets say browser makes the first request to “localhost”.it checks whether it has any cookies within it which are tagged to localhost.since it is a first request ,it won’t find & since request won’t have any cookies.  3)browser receives the response ,gets the cookie & stores it into its memory.  4)browser makes second request.this time cookies is send along with the request. | Sever gets the request along with cookies(if any).it process the request & it “may/may not” send the cookies (by settings its lifetime) along with the “response”.  2)server gets the request ,it process the request ,creates the cookie & set its lifetime & send it to browser along with the response.  5)server gets the request along with cookie,it process the request,its “may/may not ” send the cookies along with the response. |

1)cookies are little piece of information(name-value string pair) exchange between browser & server.

2)since sends the cookie to the browser and browser return the cookie whenever it makes request to the same server.

3)a cookie value can unique identity by client and hence cookies are commonly using session management.

4)the browser returns cookies to the servlet container by adding cookies to HttpRequest and hence cookies c n be retrieve from HttpRequest by using **getCookies()** method.

5)by default cookies live only as long as session.

6)one session ends cookies disappear(that’s how JSESSIONID cookie works) but it is possible to make cookies alive even after session expire.

7)there are two types of cookies.

->**Persistent Cookies-**

**-**it will survive even after a user session expired.if a persistent cookies as its age set to one year,then within a year cookies will send back to server whenever user need a request to same server by using same browser.

->**NonPersistent Cookies-**

**-**nonpersistent cookies lives as long as session.once clients quits the browser or after session expire cookies disappear.

**COOKIES RELATED METHOD**

**1)void HttpServletResponse.addCookie(Cookie arg0)**

-this method adds the specify cookies to response .this method can be called multiple times to set more than one cookies.

**2)Cookie[] HttpServletRequest.getCookies()**

-returns an array containing all the cookies object or return null if the request object have no cookie.

**3)void Cookie.setMaxAge(int expiery)**

-set the maximum age of cookies in seconds.

-a positive value indicates that cookies will expire after that many seconds have passed.

-a negative value means that cookies is not stored persistently i.e. it will be deleted when the browser exit.

-0 value causes the cookie to be deleted immediately.

**4)void Cookie.getName()**

Returns the name of the cookies.the name can’t be changed after creation.

**5)void Cookie.getValue()**

Returns the present value of the cookies.

**SESSION**

1)session is a collection of http request ,over a period of time between a client and server.

2)when a session is created ,its life is set .if lifetime is expired ,session is disappeared and its associated resource are garbage collected.

3)session can also be prematurely destroyed.

<session-config>

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

</session-config>

**SESSION TRACKING**

1)HTTP is a stateless protocol and hence even a user user sends a sequence of request from the same browser ,the server will not be able to tell that those are from the same user.

2)to address to this problem,servlets provide a HttpSession API.this is a high level interface that allows server to remember a set of information relevant to a particular user on going transaction.so that it can retrieve the information to handle any future request from the same user.

3)whenever we need to remember some information about what user does,start a new session and attach the information with this session.behind the scene servlet container will assign a unique sessionid(JSESSIONID ) to the session and sends this ID to browser with the 1st response.browser will send back this ID whenever it makes request to the server.this way server will able to trace the request coming from same user and will retrieve whatever the information that has been stored at serverside.

4)HttpSession user one of the two mechanism to handle session.

-Cookies

-URL rewriting

**URL REWRITING**

1)if the user browser supports cookies then servlet container makes use of cookies to handle session.if the browser doesn’t support cookies or the user explicitly disable cookies then servlet container makes use of URL rewriting by appending JSESSIONID to the end of every URL(only if we explicitly tell container to encode URL).all the details are handled by HttpSession automatically ant it happens behind the scene.developer need not to worry about handling the things.

SERVER

A new unique session ID is created and user’s ongoing transaction info is stored at server side using session attributes agaist this ID as shown JSESSIONID=1234,name=value,some obj=any java object. ->if need to edit the session attribute(add/replace/remove)whish are stored against the supplied JSESSIONID at server side.

once user clicks on the logout link,invalidate the session and remove all the attributes against JSESSIONID=1234

CLIENT

Initial request(no session) a new

Cookie with name=JSESSIONID &

Value=1234 is sent to browser

along with 1st res.

3)whenever user makes any future request ,browser will send this cookie to the servlet container.

4)user is redirected to login page.

**HTTPSESSION CONTEXT**

1)**void setMaxInactiveInterval(int argo)**

-it specifies valid session time is seconds.a negative time indicates that session will never timeout.

2)**void invalidate()**

-invalidate the current session and unbinds any objects bound to it.

3)refer attribute section for attribute related method.

**OTHER SESSION RELATED METHODS**

1)**HttpSession HttpServletRequest.getSession()**

-returns the current session associated with the current request.if request doesn’t have the session,this method creates a new session.

2) **HttpSession HttpServletRequest.getSession(Boolean arg0)**

-if arg equal to true,return the current HttpSession associated with the current request or if there is no session exist,returns the new session.

2)if args equal to false,return the current HttpSession associated with the current request or returns null if there is no session exist.

3) **String HttpServletResponse.encodeURL(String URL)**

**String HttpServletResponse.encodeURL(String URLS)**

-this two method ,in course the specified URL by appending SESSIONID.

-use encodeURL() to encode form,forward and hyperlink URLS and use encode redirect URL to encode redirect URLS.

**WORKING WITH HTTPSESSION**

1. **Create the Session**

-once user logs into the application and after successful authentication ,create the session for the 1st time which can be done by

HttpSession session=req.getSession();

HttpSession session=req.getSession(true);

After this below things happens behind the scene.

-servlet container creates the unique sessionID and sends it to the user browser(using cookie where name=JSESSIONID and value=unique session) along with the 1st response.

-whenever user makes any future request,browser sends this cookie to servlet container.

-servlet container stores any ongoing transaction related information using session attributes at server side against this ID.

1. **Validate the Session**

-once the session is created for the subsequent request validate the session that is in each page of the application(wherever necessary validate the session).

HttpSession session=req.getSession(**false**);

**if**(session!=**null**)

{

session.invalidate();

res.sendRedirect("LoginError?loginError=logout Successful");

}

**else**

{

//validate session

//generate the proper response.

}

1. **Invalidate the Session**

**-a** session gets invalidate in following steps.

1)on its timeout

2)when user logouts the application.

3)when application or server goes down.

**CONFIGURE SESSION TIMEOUT**

There are two ways,where we can set timeout.

**1)in web.xml**

<session-config>

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

</session-config>

**1)in programs**

HttpSession session=req.getSession(true);

session.setMaxInactiveInterval(60);

**WHEN USER LOGOUT**

-when user wants to logout the application,invoke the invalidate method on session object.

HttpSession session=req.getSession(**false**);

**if**(session!=**null**)

{

session.invalidate();

res.sendRedirect("LoginError?loginError=logout Successful");

}

**else**

{

res.sendRedirect("LoginError?loginError=you have not logged in,please login");

}

**URL REWRITING**

1)in handling user session URL rewriting comes into the picture only if we tells server to encode URL.

2)if encode our URL then container will first attempt to use cookies for session management and if cookies approach fails then call back to URL rewriting .

3)we can’t use URL rewriting method inside static pages.so if our application is dependent on session,we must carefully review out decisions of using static pages.

**URL REWRITING IN CASE OF FORWARD**

String forwardURL=res.encodeURL("ListStudents");

RequestDispatcher dispatcher=req.getRequestDispatcher(forwardURL);

dispatcher.forward(req,res);

**URL REWRITING IN CASE OF REDIRECT**

String errMsg="LoginError?loginError=please provide valid credentials";

String encodeErrorMsg=res.encodeRedirectURL(errMsg);

res.sendRedirect(encodeErrorMsg);

**URL REWRITING IN CASE OF HYPERLINK**

String

editEncodeURL=res.encodeURL("./ModifyUser?regno="+datalist.getRegno()+"&action=edit");

String deleteEncodeURL=res.encodeURL("./ModifyUser?regno="+datalist.getRegno()+"&action=delete");

out.println("<td>"+"<a href="+editEncodeURL+">edit</a>"+"</td>");

out.println("<td>"+"<a href="+deleteEncodeURL+">delete</a>"+"</td>");

**SingleThreadModel SERVLET**

->this servlet implement single thread model interface which ensure that servlet handle only one request at a time i.e. no two threads will execute concurrently the service() method of that servlet.to accomplish this each thread uses a free servlet instance from the pool as shown below.

WEB SERVER

THREAD

THREAD

THREAD

THREAD

SERVLET POOL

REQUEST

SERVLET INSTANCE

REQUEST

REQUEST

SERVLET INSTANCE

REQUEST

SERVLET INSTANCE

SERVLET INSTANCE

->SingleThreadModel Interface is a marker interface.

->SingleThreadModel servlets does not solve **all thread safety issue.**

->for every session attributes and static variables can still be accessed by multiple request on multiple threads at the same time.

->it is recommended that developer takes other means to resolve this issues instead of implementing this interface.

**public** **class** ListStudents **extends** HttpServlet implements SingleThreadModel

{

//servlet code goes here

}

**public** **class** ListStudents **extends** HttpServlet

{

**protected** synchronized **void** doPost(HttpServletRequest req, HttpServletResponse res)

**throws** ServletException, IOException

{

//code for response

}

**public** **class** ListStudents **extends** HttpServlet

{

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res)

**throws** ServletException, IOException

{

Synchronized(this){//generate the response}

->marker interface used for polymorphism.

**Filters**

1. Suppose I want to do some validation on form data before sending to servlet and I want to add some output rendering with response before sending to client then I have to use filters.
2. It means we can do some pre-servlet and post servlet operation then I can use filters.
3. It load at deployment time or container start only.

**Listner**

1. **Use to do some additional operation at the time of deployment or request creation or session creation then we have to use listener. It will start at** container start only before filter.

**Three Types**

**->ServletContextListner**

**Execute this listener at container start only and will use until container close. So suppose I have to initialize something on login then I can use.**

**->RequestListner**

**Execute this listener at the time of sending the request. e.g: count the user request.**

**-> ServletSessionListner**

**Execute this listener at the time of session creates and will Use throughout the session. e.g: count the no of user login.**