**J2EE**

* used to develop distributed, dynamic, secured and multithreaded web-oriented application using Java platform

**Servlets**

* it’s a java program used to write server site business logic processing
* 2 types

**Generic Servlets**- It is a protocol independent as it is used for all types of protocol

**Syntax-**

* public abstract class GenericServlet extends Object implements Servlet,ServletConfig
* import javax.servlet.\*; for Generic Servlet

**Life cycle of generic servlet:**

**Methods-**

* init() - optional method used for initialization & invoked first and only once in the life cycle of the program .
* service() - mandatory method to write business logic processing.
* destroy() - optional method used for deallocation of memory.

**HTTP servlet** – protocol dependent as it is based on http protocol.

**Syntax**

* public abstract class HttpServlet extends GenericServlet
* import javax.servlet.http.\*; for HttpServlet

**Life cycle:**

* init() – optional
* doGet() / doPost() / doPut() / doDelete / doOption() / doTrace() / service()
* destroy()

**Difference between doGet() and doPost() -**

* Html program without method attribute or there is no html then it is get request so we write logic in doGet(). For post request there should be a html with method=POST then we write logic in doPost().
* In get request the parameters are transferred in url between ‘?’ and ‘&’ i.e. QueryString. In post request the parameters are directly transferred from one page to another page through Headers.
* Get request transfer only 2GB of data so it is only used for fetching purpose. Post request transfers unlimited data so it is used for storing purpose.

**Syntax**

* public void doGet(HttpServletRequest req,HttpServletRespose res) throws ServletException,IOException
* public void doPost(HttpServletRequest req,HttpServletRespose res) throws ServletException,IOException
* Sevlets should be configured in web.xml or using @WebServlet annotation
* response.setContentType("text/html"); - To write html tags inside the servlet program
* PrintWriter pw = response.getWriter(); - To print on the browser
* pw.println("<b>HellowWorld"); - To print on the browser
* System.out.println("Hello"); - To print in console
* pw.close(); - close pw

javax.servlet.http.\* package

**HttpServletRequest Interface** - used to read the data from the client request.

**Syntax:** public interface HttpServletRequest extends ServletRequest

**Methods :**

* String getParameter(String name): used to return single value from HTML page to servlet program
* Enumeration getParameterNames(): used only for printing all values at a time
* String[] getParameterValues(String name):if single control takes multiple values it is returned using this method, e.g. check box
* String getScheme(); return the name of the protocol
* String getProtocol(); return the version of the protocol
* String getServerName(); return name of the server
* int getServerPort(); return port number
* String getContextPath(); returns the name of the project
* String getServletPath; returns the url pattern of the servlet
* String getPathInfo(); returns the path i.e. located after the servlet path and QueryString
* String getRequestURI(); returns ContextPath , ServletPath and PathInfo
* String getQueryString(); returns the path between ? and &
* String getMethod(); returns the type of request, either GET or POST

**RequestDispatcherInterface-** used to dispatch a request from one servlet to other resources like HTML or JSP or another Servlet

**Syntax:**

* RequestDispatcher rd = request.getRequestDispatcher(”html or jsp or another servlet”);
* rd.forward(request,response) or rd.include(request,response)
* rd.forward()-used to forward the page to another resources like html or jsp or another servlet and doesn’t to the calling page redirection take place on server side so as a client we cant see anything in the url to which page it is redirected. It can redirect the resources only within the project / application
* rd.include()-used to include the output of another resources and returns back to the calling page. It can redirect the resources only within the project / application
* response.sendRedirect(“html or jsp or another servlet”) – used only to redirect to servlet program to another resources and we cant pass any parameters. Redirection takes place on client side we can clearly see in the URL to which page it has been redirected, This method can redirect the resouces out the application

**Attribute Methods-** Used to send some values in the background of the application

There are four attribute methods applicable for three objects

**Methods:**

* void setAttribute(String name, Object value)-used to set an attribute
* Object getAttribute(String name)-used to return a single attribute value
* Enumeration getAttributeName()-prints all the attributes at a time
* void removeAttribute(String name)-used to remove the value for the attribute

**Three objects-**

* request object- when we set an attribute using request object, it can retrieve only to the page we are sending the request
* context object- when we set an attribute using context object, it can be retrieve throughout the application
* session object-when we set an attribute using session object, we can retrieve in the page which participated in the session

**Headers-**

Whenever client give request to server certain default information is transferred between client and server, those default information is called headers.

Certain headers have String as a value, int as a value, date as a value. The diiferent headers are :-

* Accept - the file type the browser accept text/html
* Accept-language - the language the browser accept
* Accept-charset - character set the browser accept,”ISO-8859-1”
* Accept-Encoding - encoding technique the browser use like gzipand browser knows how to decode
* User-Agent - the type of browser the client uses
* Hosts - return server name and port number
* Content-length - for post messages how much data is attached
* Cookie - content session ID
* From - email address of requester

**Methods –**

* String getHeader(String headerName)-returns a single header that takes String as value
* Int getIntHeader(String headerName)-returns a single header that takes int as a value
* Long getDateHeader(String headerName)-returns a single header that takes date as a value
* Enumeration getHeaderNames()-prints all headers at a time
* Enumeration getHeaders(String headernames)-if single header take multiple values

**Cookie[] getCookies() -** used to retrieve the cookie information in the browser.

* HttpSession getSession();
* HttpSession getSession(boolean on);
* Boolean isRequestSessionIdFromCookie();
* Boolean isRequestSessionIdFromURL();
* Boolean isRequestSessionIdIsValid();

**HttpServletResponse Interface** – used to write response to the client request

**Syntax**

public interface HttpServletResponse extends ServletResponse

**Methods-**

* void setContentType(String type)
* PrintWriter getWriter()throws IOException-used to display the response in the form of character
* ServletOutputStream getOutputStream()throws IOException-used to display the response in the form of bytes
* void addCookie(Cookie c)
* void sendRedirect(String url)
* void encodeURL(String url)-used to send session id alonfg with the url
* void setHeader(String headerName,String value)
* void setIntHeader(String headername,int value)
* void setDateHeader(String headername,long value)

**SingleThreadModel Interface**-by default all servlets are multithreaded if you want the servlet to accept only single request at a time then we implement single thread model interface which is a marker interface i.e. which doesn’t contain any methods

Ex- public class sample extends HtttpServlets implements SingleThreadModel{}

**Cookie class-**

It is a temporary storage of internet file which contain name, path, domain,expiry date of the web page and session-id.

* Cookie()
* Cookie(String name,String value)
* Void setDomain(String domain)
* String getDomain()
* Void setPath(String path)
* String getPath()
* Void setMaxAge(int age)
* Int getMaxAge()
* Void setName(String name)
* String getName()
* Void setValue(String val)
* String getValue()
* Void getSecure(boolean on)
* Boolean isSecure()

**Creation of cookie**

1. Create the object for the cookie class
2. Add the cookie at the time of response using addCookie()
3. Retrieve the cookie information using getCookies()

**ServletConfig Interface –** used to access initialization parameter in servlet program which is configured in web.xml using <init-param> or using @WebInitParam annotation. Initialization parameter are the parameters that can be accessed only by a particular servlet

**Syntax:**

ServletCongif sc = getServletConfig()

**Methods:**

* String getInitParameter(String name): used to return single initialization parameter
* Enumeration getInitParameterName(): prints all initialization parameters at a time

**ServletContext Interface-** used to access application parameter in all servlet which is configured only in web.xml using <context-param>. Application parameters are parameters that can be accessed throughout the application. Context object also used to set and get the attribute throughout the application.

**Syntax:**

ServletContext sc = getServletContext()

**Methods:**

* String getInitParameter(String name)
* Enumeration getInitParameterNames()
* void setAttribute(String name, Object value)
* Object getAtrribute(String name)
* Enumeration getAttributeNames()
* void removeAttribute(String name)

**Filter Interface-** used to do preprocessing of request and post-processing of response. It is a java program which implements javax.servlet.filter interface. Filter is configured in web.xml using <filter>

and <filter-mapping> or using @WebFilter annotation

**Methods-**

* init() - optional method used for initialization int the first and only once in the life cycle of a program.
* doFilter(ServletRequest req, ServletResponse res, FilterChain c) – mandatory method to write preprocessing and post-processing
* destroy() – optional method used for deallocation

**HttpSession Interface:**J2EE is used to develop web oriented application which is focused on HTTP protocol which is a stateless protocol means we can transfer the information only from one page to another page. If we want to transfer the information to multiple pages then we create session. Session is used to maintain the stability or persistence of the web page by creating a unique session id until we click log-out button or close the browser.

Session is used in case of user authentication and stored in Cookies. Browsers which do not support Cookies, stores the Session ID in the URL.

**Creation of Session:**

* HttpSession hs = request.getSession() or HttpSesssion hs = request.getSession(true);
* This method creates a new session with a unique Session ID
* HttpSession Interface is used to access extra information about the session.

**Methods:**

* String getId(): returns the Session ID
* long getCreationTime(): returns date and time in mili-seconds from Jan 1st, 1970 till when the session was creted
* long getLastAccessedTime(): return date and time in mili-seconds from Jan 1st, 1970 till the session lastly accessed
* boolean isNew(): to check whether the session is new or old
* void invalidate(): used to expire the session
* void setAttribute(String name, Object value)
* Object getAttribute(String name)
* Enumeration getAttributeNames()
* void removeAttribute(String name)
* void setMaxInactiveInterval(long sec): used to expire the session after certain time in seconds
* to expire the session after certain time in Minutes we configure using @WebSession or in web.xml using

<webapp>

<session-config>

<session-timeout>10</session-timeout>

</session-config>

</web-app>​

**ExpressionLanguage or EL –** without using scriplet or expression if we want to print in the browser then we use expression language. In jsp by default EL is disabled we have to enable the EL by

* Using page directive element , to enable EL for that particular page

<%@ page isELIgnored=”false” %>

* To enable EL in all the pages we use web.xml

<web-app>

<jsp-config>

<jsp-property-group>

<url-pattern>/\*.jsp</url-pattern>

<el-enabled>true</el-enabled>

</jsp-property-group>

</jsp-config>

</web-app>

**Syntax**

${expression}

**EL Operators**

* Arithmetic operator: +, -, \*, /or div, % or mod
* Relational operator: >= or ge, <= or le, > or gt, < or lt, == or eq, != or ne
* Logical operator: && or AND, || or OR, ! or NOT

**EL Implicit Objects**

* param : used to get a single parameter from HTML program
* paramValues: used to return multiple values for single control
* pageScope: return the page levelled value
* requestScope: return the request scope levelled value
* sessionScope: returns the session scope levelled value
* applicationScope: returns the application scope levelled value
* initParam: used to access application parameter
* header: used to return a single header value
* headerValue: used to return header with multiple value
* cookie: used to retrieve cookie information
* pageContext: used to access jsp implicit object

**JSTL Framework** (jsp standard tag library)- it is a framework used to develop the web page completely with the help of tags

**Installation:** Download JSTL-1.2 jar and put inside lib folder since JSTL is a framework.

We have to use taglib directive element to give the information about JSTL tags to Tomcat server

**Types-**

* core tags: used to declare the variable, printing the output, conditional statements, iteration and catching the exception

<%@ taglib uri=” http://java.sun.com/jsp/jst/core” prefix=”c” %>

* sql tags: used to do database processing
* xml tags: used to xml processing
* fmt tags: used for formatting date, number, currency etc.

**Core tags-**

* <c:set>- used to declare a variable in different scope
* <c:out>- used to remove the value for the attribute
* <c:remove>- used to remove the value for the attribute
* <c:if>- used to check single condition
* <c:choose>, <c:when>, <c:otherwise>- all 3 are used to check multiple condition
* <c:forEach>- used to iterate over the data
* <c:forTokens>- used to split the collection of data based on delimiter
* <c:catch>- used to catch the exception
* <c:redirect>- used to redirect from one page to another page

**Collection Pooling**

It is a pattern used by software application to connect to databases using pre-created set of reusable connection objects. When a new connection is needed, an existing connection is retrieved from the pool. When the thread using the connection is completed it is placed back in the pool for used by another thread. For using connection pool, we need Apache-DBCP library provided with commons-collection, commons-dbcp, commons-pool jar files.

**DAO PATTERNS**

With DAO design patterns we use the following components: -

* The models which transfers from one layer to another
* The interface which provides a flexible design
* The interface implementation which is the concrete implementation of the persistent logic

**Advantages: -**

* While changing your persistent mechanism, service layer doesn’t even have to know where the data comes from, all changes are needed to done in the DAO layer only.
* Dao pattern emphasize on the low coupling between different components of the application. So the view layer has no dependency on the DAO layer and only service layer depends on it.
* As the persistent logic is completely separate it is easier to write unit testing.
* As wo work with interfaces with DAO pattern, it also emphasizes the style of work with interfaces instead of implementation which is an excellent Oops style of programming.
* DAO design pattern used to emphasize on keeping persistent logic separate so our components are loosely coupled.

**SINGLETON PATTERN**

To implement singleton pattern we provide private constructor to restrict instantiation of the class from other classes.