**JAVA WEB DEVELOPMENT**

**Java EE/J2EE/Adv Java**:

* Java EE stands for Java Enterprise Edition
* Java EE is used to develop web applications and enterprise applications

**Components of Java EE**:

Web Components used to develop web applications

* Servlets
* JSP (Java Server Pages)

Enterprise Components used to develop enterprise applications (secured, distributed)

* EJB (Enterprise Java Beans)

Java EE applications are used by multiple clients and in order to make the Java EE application to be used by multiple clients we need to deploy(Copy) the application onto the server.

**Types of Servers** :

* Web Servers
* Application Servers

**Web Servers :**

Web servers are used to develop web applications only

* Apache Tomcat
* Jetty

**Servlets** :

Servlets are Java programs used to develop server side dynamic web applications.

Technologies used to develop server side dynamic web applications are

* Servlets
* JSP

**Life cycle of a Servlet :**

Life cycle is a collection of methods which gets executed in an order automatically

by the servlet container (server).

init() :

* called only once when Servlet is loaded
* generally we establish database connections

service() :

* called for every request

destroy() :

* called only once when Servlet is destroyed
* generally we close database connections

**Creating a Servlet :**

By creating a class which extends GenericServlet class **(or)**

By creating a class which extends HttpServlet class

**Using GenericServlet class :**

public class WelcomeServlet extends GenericServlet {

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException { //handle the request }

}

**Using HttpServlet class :**

public class WelcomeServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException { //handle the reques }

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException { //handle the request }

}

**Note :**

interface HttpServletRequest extends ServletRequest {}

interface HttpServletResponse extends ServletResponse {}

**GenericServlet class vs HttpServlet class** :

GenericServlet class is used for different protocols like ftp, smtp, http etc.

where as HttpServlet class is used for only http protocol

ftp => file transfer protocol

smtp => simple mail transfer protocol

http => hypertext transfer protocol

**Sending HTML as response** :

* response.setContentType("text/html") method is used to send HTML as response

**Reading HTML Form data into a Servlet :**

* HTML Form is used to accept the user input
* request.getParameter() is used to read HTML form data into a Servlet
* The return type of request.getParameter() is String

**response.sendRedirect()** method is used to redirect to other Servlets/JSPs/HTML pageslike below example.

* response.sendRedirect("/WebProj/login.html");

**GET vs POST methods :**

* In case of GET method the data goes along with the url (visible in URL) where as in case of POST method the data goes as a separate page (not visible in URL)
* In case of GET method we can send only limited data (255 characters) where as in case of POST method we can send unlimited data
* Performance of GET method is more than POST method

**ServletConfig interface :**

This interface is used to read initial parameters from web.xml file into a

particular Servlet.

**ServletContext interface :**

This interface is used to read context parameters (global) from web.xml file

into all Servlets of web application.

**Session Tracking :**

Session means moving from one page to another pages

It is used to track whether the user is coming from previous pages properly or not

Ex: Page1 -> Page2 -> Page3

**4 ways of Session Tracking :**

* Hidden Form Fields
* URL Rewriting
* Cookies
* HttpSession

**Hidden Form Fields :**

In this we track the user session by using hidden form fields created in previous

pages and read the value of hidden form fields in the next pages.

**URL Rewriting :**

In this we track the user session by sending the data along with the url to the next

pages

**Cookies :**

* Cookies are used to remember the user data
* Cookies are used to store the data onto the client browse

**Creating Cookies :**

Syntax : Cookie cookieObj = new Cookie("cookie-name","cookie-value");

Ex: Cookie c1 = new Cookie("username","admin");

response.addCookie(c1);//adds the cookie onto the client browser

**Reading Cookies :**

Cookie[] cookies = request.getCookies();

for(Cookie c : cookies)

out.println(c.getName()+":"+c.getValue());

getName() => returns the cookie-name

getValue() => returns the cookie-value

**HttpSession interface**  is used to track the user session by using attributes

Page1

HttpSession session = request.getSession();

session.setAttribute("attribute-name","attribute-value");

Page2

HttpSession session = request.getSession();

session.getAttribute("attribute-name");

**Filters :**

Filters are used for the following

* provide the preprocessing logic before the request is handled
* provide the postprocessing logic after the request is handle

**Listeners :**

Listeners are executed automatically based on the events like the following

* creating a ServletContext object
* creating a HttpSession object

**JSP (JAVA SERVER PAGES) :**

* JSP stands for Java Server Pages
* JSP is used to develop server side dynamic web applications like Servlets

**Servlets vs JSP :**

* In case of Servlets, Java code contains HTML code where as in case of JSP, HTML code contains Java code
* In case of Servlets, if the code is modified we need to recompile and redeploy the application where as in case of JSP, if the code is modified we need not recompile and redeploy the application
* JSP code is less than Servlet as in JSP we can use JSP Tags, JSP Implicit Objects and JSTL

**JSP Architecture :**

***Two Phases***

* **Translation Phase**
* When a request is made to a JSP page, the JSP page is translated into a Servlet only once until JSP page changes
* **Request Process Phase**
* the request is handled by the translated servlet

**JSP Life Cycle :**

jspInit():

* called only once when JSP page is loaded
* generally we establish Database connections

\_jspService():

* called for every request
* cannot be overridden

jspDestroy():

* called only once when JSP page is destroyed
* generally we close database connections

**JSP Tags :**

* *Directive Tags :*
* page directive tag => <%@page .... %>
* include directive tag => <%@include .... %>
* taglib directive tag => <%@taglib .... %>
* *Scriptlet Tags :*
* Declaration tag => <%! ... %>
* Expression tag => <%= ... %>
* Script tag => <% ... %>
* Action Tags
* <jsp:include>
* <jsp:forward>
* <jsp:param>
* <jsp:useBean>
* <jsp:setProperty>
* <jsp:getProperty>

**Custom Tags** used to create user defined tags in JSP

**page directive tag :**

<%@page ... %>

* used to import the Java packages

Ex: <%@page import = "java.util.\*" %>

<%@page import = "java.sql.\*" %>

or

<%@page import = "java.util.\*,java.sql.\*" %>

* used to set the content type

Ex:<%@page contentType="text/html" %> //response.setContentType("text/html");

<%@page contentType="application/vnd.ms-excel" %>

* used to handle the exceptions

**include directive tag :**

<%@include ... %>

* used to include other JSP pages

Ex: <%@include file="header.jsp" %>

...

*<%@include file="footer.jsp" %>*

**Declaration Tag**

<%! ... %>

* used to declare variables and to define methods the code will go into the translated servlet class as data members and methods

Ex: <%! int n = 100; %>

<%! void xxx() { ... } %>

<%! void jspInit() { ... } %>

**Expression Tag :**

<%= ... %>

* used to access the value of the variable and to call methods
* the code goes into \_jspService() of translated servlet class

Ex: <%= n %> //100 will get displayed on client browser

<%= xxx() %>

<%= new java.util.Date() %>

**Script Tag :**

<% ... %>

* used to include the Java Statements
* the code goes into \_jspService() of translated servlet class

Ex: <% for(int i=1;i<=10;i++)

out.println(i); %>

**Note**

* HttpJspBase extends HttpServlet
* JspWriter extends PrintWriter

**config implicit object :**

config is an implicit object of ServletConfig interface which is used to read

initial parameters from web.xml file into a particular JSP file

**application implicit object :**

application is an implicit object of ServletContext interface which is used to

read context parameters (global) from web.xml file into all JSP pages of web application

**session implicit object:**

Session Tracking is used to track whether the user is coming from previous pages

properly or not

Ex: Page1 -> Page2 -> Page3

Page1 -> Page2 -> Page4

session is an implicit object of HttpSession interface which is used to track the

user session by using attributes

**pageContext implicit object :**

pageContext is an implicit object PageContext class and it is used to set, get and

remove attributes from the following scopes

* page
* request
* session
* application

**Note:** In JSP, page scope is the default scope

**Handling Exceptions in JSP :**

In Java try { statements; }

catch(Exception e){ //handle}

In case of JSP, if an exception occurs in a JSP page we can redirect to another JSP page to handle the exception

Ex: <%@page errorPage="ShowErrors.jsp" %>

exception is an implicit object of Throwable class is used to handle exceptions in JSP

**JSP Implicit Objects :**

* out
* request
* response
* config
* application
* session
* page
* pageContext
* exception

**JSP Action Tags :**

* <jsp:include>
* <jsp:forward>
* <jsp:param>
* <jsp:useBean>
* <jsp:setProperty>
* <jsp:getProperty>

**<jsp:include> and <jsp:forward> :** These tags are used to include other JSP files

In case of <jsp:include> the response is handled by all the JSP pages where as

In case of <jsp:forward> the response is handled by only the last page

**<%@include ... %> directive vs <jsp:include> tag :**

In case of <%@include ...%> directive the content will get placed during translation phase where as in case of <jsp:include> tag the content will get placed during request process phase

<%@include ... %> directive is static where as <jsp:include> tag is dynamic

**<jsp:param> :**

It is used to send the parameters from one JSP page to another JSP page

**Using JavaBeans in JSP :**

A JavaBean is a public class with private properties and public setters and getter methods for every property

Ex: public class Book {

private int bno;

public void setBno(int bno) {

this.bno = bno;

}

public int getBno() {

return bno;

}

**JSTL :**

* JSTL stands for JSP Standard Template Library
* JSTL is created by Apache Company using Custom Tags of JSP
* In order to use the JSTL tags we need to copy "jstl1.2.jar" into WEB-INF/lib folder