**********Servlet**********

- **Servlets:-**

- Web browser & servers.

- Servlets are web pages generated by the web servers in response to the request send by the clients.

- MIME type (Multipurpose Internet Mail Extension). Eg. text/pain, text/html, etc.

- CGI (Common Gateway Interface). C,C++,perl.

-> Performance of servlets is better than CGI.

-> CGI is not platform independent but servlets is platform independent.

-> Sun, Netscape and Microsoft web servers offer servlet API.

-> The programs developed for this API can be moved to any of these environment without recompiling them.

-> Java security manager will helps us to set restriction to access resources of server machine.

-> As we know java has great collection of java class libraries and all these are available for servlet.

- **Types of Servlets:**

1) Generic Servlets	2) HTTP Servlets
- Used javax.servlet package.	- Use javax.servlet.http package.
- Protocol independent	- Protocol dependent.
- It will execute service() method	- It will execute doGet() and doPost().

- **Servlet Life Cycle:**

init()

service()

destroy()

Step-1:

- User enter URL in web browser and generates HTTP request for this URL.

- This request transfer to appropriate web server.

Step-2:

- HTTP request received by web server. Server map this request to the particular servlet then servlet loaded into address space of web server.

Step-3:

- Server invokes the init() method of servlet. When servlet first loaded into the computer memory then this method will get called.

- The initialization parameter passed to the servlet so that it will configure itself.

public void init(ServletConfig config)

{

}

Step-4:

- Server invoke service() method. This method is used to process HTTP request.

- service() method accept the request parameter and process it and try to create HTTP response for the client.

- servlet available in server address space and it is able to process each HTTP request.

public void service(ServletRequest req,ServletResponse res)

{

}

Step-5:

- Server may take decision to unload the servlet from its memory. Server will use algorithm to take this decision.

- The server will invoke destroy() method to released the resources which are allocated for the servlet.

public void destroy()

{

}

- **Steps to create a servlet example:-**

There are given 6 steps to create a **servlet example.** These steps are required for all the servers.

The servlet example can be created by three ways:

1. By implementing Servlet interface.

2. By inheriting GenericServlet class

3. By inheriting HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(),doPost() etc.

- **Steps to create a servlet example**

1. Create a directory structure

2. Create a Servlet

3. Compile the Servlet

4. Create a deployment descriptor

5. Start the server and deploy the application

Here, we are going to use **apache tomcat** server in this example. The steps are as follows:

1. Create a directory structure

2. Create a Servlet

3. Compile the Servlet

4. Create a deployment descriptor

5. Start the server and deploy the project

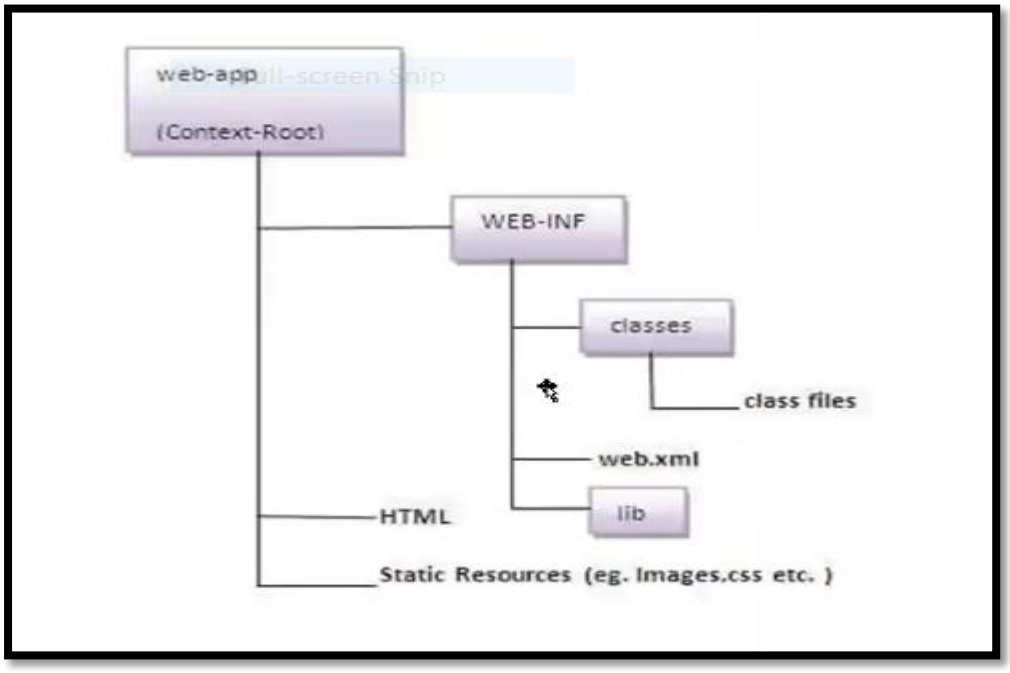
6. Access the servlet

1) Create a directory structures:-

The directory structure defines that where to put the different types of files so that web container may get the information and responds to the client.

The Sun Microsystem defines a unique standard to be followed by all the server vendors. Let's see the directory structure that must be followed to create the servlet.

Diagram:-



As you can see that the servlet class file must be in the classes folder. The web.xml file must be under the WEB-INF folder.

2) Create a Servlet:-

There are three ways to create the servlet.

1. By implementing the Servlet interface

2. By inheriting the **GenericServlet** class

3. By inheriting the **HttpServlet** class

The HttpServlet class is widely used to create the servlet because it provides methods to handle http requests such as doGet(),doPost, etc.

In this example we are going to create a servlet that extends the HttpServlet class. In this example, we are inheriting the HttpServlet class and providing the implementation of the doGet() method. Notice that get request is the default request.

DemoServlet.java:-

```
//Generic Servlet Demo Program.

import javax.servlet.*;
import java.io.*;

public class HelloServlet extends GenericServlet
{
  public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
  {
            res.setContentType("text/html");//Setting the Content
Type
            PrintWriter pw=res.getWriter();//get The Stream To Write
the data.

            //Writing html in the Stream.
            pw.print("<html><body>");
            pw.print("<b>Welcome to World Of Servlet
Programming!!!");
            pw.print("</body></html>");
            pw.close();//Closing the Stream.
  }
}
```

3) Compile the servlet:-

For compiling the Servlet, jar file is required to be loaded. Different Servers provide different jar:

Jar file	Server
1) servlet-api.jar	Apache Tomcat
2) weblogic.jar	Weblogic
3) javaee.jar	Glassfish
4) javaee.jar	JBoss

Two ways to load the jar file

1. set classpath

2. paste the jar file in JRE/lib/ext folder

Put the java file in any folder. After compiling the java file, paste the class file of servlet in WEB-INF/classes directory.

4) Create the deployment descriptor (web.xml file):-

The **deployment descriptor** is an xml file, from which Web Container gets the information about the servlet to be invoked.

There are many elements in the web.xml file. Here is given some necessary elements to run the simple servlet program.

web.xml file

```
<web-app>

<servlet>

<servlet-name>vjtechacademy</servlet-name>
 <servlet-class>HelloServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>vjtechacademy</servlet-name>

<url-pattern>/welcome</url-pattern>

</servlet-mapping>

</web-app>
```

Description of the elements of web.xml file

There are too many elements in the web.xml file. Here is the illustration of some elements that is used in the above web.xml file. The elements are as follows:

<web-app >represents the whole application

<servlet> is sub element of <web-app> and represents the servlet.

<servlet-name> is sub element of <servlet>represents the name of the servlet.

<servlet-class>is sub element of <servlet> represent the class of the servlet.

<servlet-mapping> is sub element of <web-app>. It is used to map the servlet.

<url-pattern> is sub element of <servlet-mapping.> This pattern is used at client side to invoke the servlet.

5) Start the Server and deploy the project:-

To start Apache Tomcat server, double click on the startup.bat file under apache-tomcat/bin directory.

One Time Configuration for Apache Tomcat Server

You need to perform 2 tasks:

1. set JAVA_HOME or JRE_HOME in environment variable (It is required to start server).

2. Change the port number of tomcat (optional). It is required if another server is running on same port (8080).

6) How to deploy the servlet project:-

Copy the project and paste it in the webapps folder under apache tomcat.

7) How to access the servlet:-

Open browser and write

http://hostname:portno/contextroot/urlpatternofservlet. For example:

```
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.*;
public class HttpServletDemo extends HttpServlet
{
        public void doGet(HttpServletRequest req,HttpServletResponse
        res)throws ServletException,IOException
        {
                res.setContentType("text/html");
                PrintWriter=res.getWriter();
                String name=req.getParameter("nm");
                pw.print("<html><body>");
                pw.print("<br>Welcome"+name);
                pw.print("</body></html>");
                pw.close();
        }
}
```

Web.xml

```
<web-app>
   <servlet>
          <servlet-name>SVMP14</servlet0-name>
          <servlet-class>HttpServletDemo</servlet-class>
  </servlet>
    <servlet-mapping>
```

```
            <servlet-name>SVMP14</servlet-name>
            <url-pattern>/college</url-pattern>
    </servlet-mapping>
</web-app>
```

loginPage.html

```
<html>
<body>

<form name="myform"
      Action=""http://localhost:9696/SVMP14/college" method="GET">
      Enter Your Name:<input type="text" name="nm">
     <input type="submit" value="Submit" name="b1">
</form>
</body>
</html>
```

****

- **Cookies in Servlet:**

A cookie is a small piece of information that is persisted between the multiple client requests.

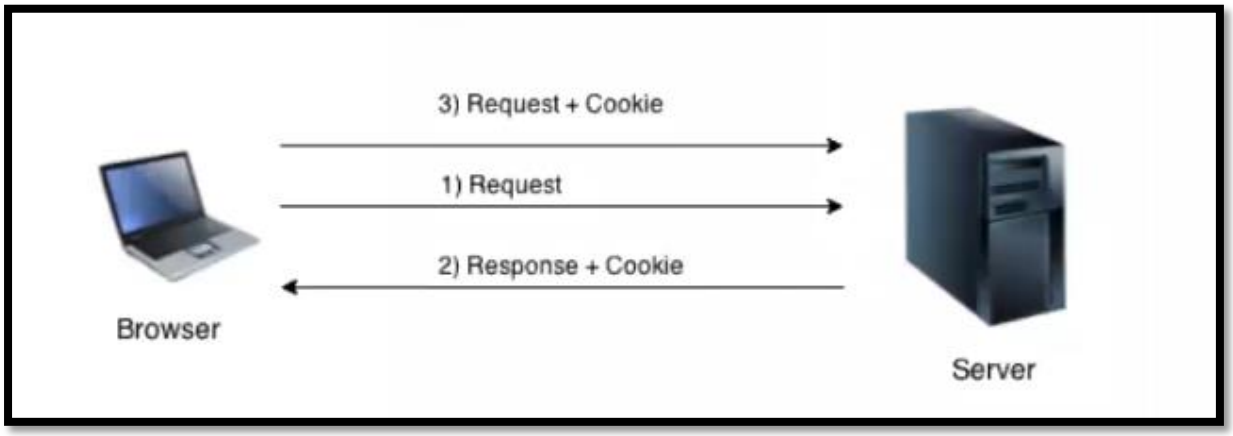
A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

- **How Cookie works:**

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache

of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

Diagram:-



- **Types of Cookie:-**

There are 2 types of cookies in servlets.

1. Non-persistent cookie

2. Persistent cookie

- **Non-persistent cookie:-**

It is **valid for single session** only. It is removed each time when user closes the browser.

- **Persistent cookie:-**

It is **valid for multiple session**. It is not removed each time when user closes the browser. It is removed only if user logout or signout.

- **Advantage of Cookies:-**

1. Simplest technique of maintaining the state.

2. Cookies are maintained at client side.

- **Disadvantage of Cookies:-**

1. It will not work if cookie is disabled from the browser.

2. Only textual information can be set in Cookie object.

- **Cookie class:-**

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

- **Constructor of Cookie class**

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

- **Useful Methods of Cookie Class:-**

There are given some commonly used method of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.

public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

- **Other methods required for using Cookies:-**

For adding cookie or getting the value from the cookie. we need some methods provided by other interfaces. They are:

1. public void addCookie(Cookie ck):method of HttpServletResponse interface is used to add cookie in response object.

2. public Cookie[] getCookies():method of HttpServletRequest interface is used to retum all the cookies from the browser.

- **How to create Cookie?**

Let's see the simple code to create cookie.

```
Cookie ck=new Cookie("user","vjtechacademy");//creating cookie object
response.addCookie(ck);//adding cookie in the response
```

- **How to delete Cookie?**

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

```
Cookie ck=new Cookie("user",""):://deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

- **How to get Cookies?**

Let's see the simple code to get all the cookies.

```
Cookie ck[]=request.getCookies();
 for(int i=0;i<ck.length;i++)
{
    out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and
        value of cookie
}
```

1)

```
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.*;
public class AddCookie extends HttpServlet
{
  public void doGet(HttpServletRequest req,HttpServletResponse res)throws
        ServletException,IOException
 {
         res.setContentType("text/html");
        String name=req.getParameter("cname");
         String value=req.getParameter("cvalue");
         Cookie ck=new Cookie(nm,value);
         res.addCookie(ck);
         PrintWriter pw=res.getWriter();
         pw.println("<html><body>");
         pw.println("<b> Your Cookie Added Successfully!!!");
         pw.println("</body></html>");
         pw.close();
  }
}
```

Web.xml

```
<web-app>
   <servlet>
```

```
        <servlet-name>vjtech</servlet0-name>
        <servlet-class>AddCookie</servlet-class>

    </servlet>
    <servlet-mapping>
        <servlet-name>vjtech</servlet-name>
            <url-pattern>/ajp</url-pattern>
    </servlet-mapping>
</web-app>
```

HomePage.html

```
<html>
<body>
 <form action="http://localhost:8484/CookieDemo/ajp" method="get">
  Enter CookieName:<input type="text" name="cname"><br><br>
  Enter Cookie Value:<input type="text" name="cvalue"><br><br>
  <input type="submit" value="Create Cookie" name="b1">
 </form>
 </body>
 </html>
```

2)

- **GetCookieServlet-**

```
import javax.servlet.http.*;
import java.io.*;
```

```
import javax.servlet.*;
public class GetCookieServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req,HttpServletResponse res)throws
        ServletException,IOException
 {
          res.setContentType("text/html");
         Cookie ck[]=req.getCookies();
         PrintWriter pw=res.getWriter();
         pw.println("<html><body>");
         for(int i=0;i<ck.length;i++)
        {
          pw.print("<br>"+ck[i].getName()+" "+ck[i].getValue());
         }
         pw.println("</body></html>");
         pw.close();
 }
}
```

Web.xml

```
<web-app>
      <servlet>
          <servlet-name>vjtech1</servlet-name>
          <servlet-class>GetCookieServlet</servlet-class>
```

```
        </servlet>
        <servlet-mapping>
            <servlet-name>vjtech1</servlet-name>
                <url-pattern>/get</url-pattern>
        </servlet-mapping>
</web-app>
```

****

- **Session and Session Tracking:-**

getSession()- This Method will create HttpSession object.

```
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.*;
import java.util.*;
public class DateServlet extends HttpServlet
{
 public void doGet(HttpServletRequest req,HttpServletResponse res)throws
   ServletException,IOException
 {
    HttpSession hs=req.getSession(true);
    res.setContentType("text/html");
     PrintWriter pw=res.getWriter();
     pw.println("<html><body>");
     Date d1=(Date)hs.getValue("date");
```

```
        pw.print("<b>Last Access Time:"+d1);
        pw.println("</body></html>");
        pw.close();
     }
   }
```

Web.xml

```
  <web-app>
       <servlet>
         <servlet-name>vjtech2</servlet0-name>
         <servlet-class>DateServlet</servlet-class>
       </servlet>
       <servlet-mapping>
           <servlet-name>vjtech2</servlet-name>
                <url-pattern>/getsession</url-pattern>
       </servlet-mapping>
   </web-app>
```

****

- **JSP:**

- JSP stands for Java Server Page.

- JSP is similar to ASP(Active Server Page).

- JSP used at server side.

- JSP do not <SERVLET> tag in an html file.

- Every block of code is called as scriplet.

- The scriplet is begin with <% and ends with %>

- JSP has got four predefined variables.

1) request: This is servlet request and is object of HttpServletRequest class.

2) response: This is servlet respone and is object of HttpServletResponse class.

3) out: It is an output writer and is an object of PrintWriter.

4) in: It is input reader and is an object of BufferedReader class.

- Extension of JSP file is .jsp. Eg hello.jsp

- If we want run above jsp file then we can use URL http://server:port/hello.jsp.

Example of hello.jsp file

```
<html>
<body>
<%
        if(request.getParameter("name")==null)
        {
                out.println("No user entered");
        }
        else
        {
                out.println("Hello
                "+request.getParameter("name"));
        }
%>
```

```
</body>

</html>
```

- **Expression and Directives:-**

- As we know that JSP use scriplets. JSP also use expressions and directive.

- A JSP expression begin with <%= and ends with %>.

- A JSP directive begin with <%@ and ends with %>. We use directive like <%@ varname="value" %>

There are total six variables that can be used.

1) content_type:

E.g - <%@ content_type="text/plain" %>

2) import:

E.g - <%@ import="java.io.*,java.util.*,java.lang.*"%>

3) extends:

E.g - <%@ extends="SuperClass1" %>

4) implements:

E.g - <%@ implements="interface name" %>

5) method:

E.g - <%@ method="doGet" %>

6) language:

E.g - <%@ language="java" %>

- **Life Cycle of JSP Page:-**

Following steps are involved in JSP Life Cycle:

1) Compilation

2) Initialization

3) Execution

4) Cleanup

- **JSP Compilation:-**

- Here, we can compile the jsp page.

- In compilation process, following phases are present

I) Parsing the JSP

II) Converting JSP into Servlet

3) Compilation of the servlet.

- **JSP Initialization:-**

- It is used to initialize database connection, create lookup tables and open files, etc.

- **JSP Execution:-**

In this phase, JSP engine trying to process request and will try to generate response.

```
public void _jspService(HttpServletRequest req, HttpServletResponse res)
{
        //body
}
```

- **JSP Cleanup:-**

-In this phase, JSP got removed from the container. it means, whatever the resources allocated for the JSP got removed.

Inspiring Your Success

VJTech Academy…