What is a Web Application

A web application is computer software that can be accessed using any web browser. Usually, the frontend of a web application is created using the scripting languages such as HTML, CSS, and JavaScript, supported by almost all web browsers. In contrast, the backend is created by any of the programming languages such as Java, Python, Php, etc., and databases. Unlike the mobile application, there is no specific tool for developing web applications; we can use any of the supported IDE for developing the web application.

Web Server and Client

The web server is a process that handles the client's request and responds. It processes the request made by the client by using the related protocols. The main function of the webserver is to store the request and respond to them with web pages. It is a medium between client and server. For example, Apache is a leading webserver.

A client is a software that allows users to request and assist them in communicating with the server. The web browsers are the clients in a web application; some leading clients are Google Chrome, Firefox, Safari, Internet Explorer, etc.

HTML and HTTP

The HTML stands for HyperText Markup Language; it is a common language for Web Server and Web Client communication. Since both the web server and web client are two different software components of the web, we need a language that communicates between them.

The HTTP stands for HyperText Transfer Protocol; it is a communication protocol between the client and the server. It runs on top of the TCP/IP protocol.

Some of the integral components of an HTTP Request are as following:

**HTTP Method:** The HTTP method defines an action to be performed; usually, they are GET, POST, PUT, etc.

**URL:** URL is a web address that is defined while developing a web application. It is used to access a webpage.

**Form Parameters:** The form parameter is just like an argument in a Java method. It is passed to provide the details such as user, password details on a login page.

What is URL

URL stands for Universal Resource Locator used to locate the server and resource. It is an address of a web page. Every web page on a project must have a unique name.

A URL looks like as follows:

* <http://localhost:8080/SimpleWebApplication/>

Where,

**http or https:** It is the starting point of the URL that specifies the protocol to be used for communication.

**Localhost:** The localhost is the address of the server. When we run our application locally, it is called localhost; if we deployed our project over the web, then it is accessed by using the domain name like "javatpoint.com". The domain name maps the server to IP addresses.

**8080:** This is the port number for the local server; it is optional and may differ in different machines. If we do not manually type the port number in the URL, then by default, the request goes to the default port of the protocol. Usually, the port no between 0 to 1023 are reserved for some well-known services such as HTTP, HTTPS, FTP, etc.

We have discussed all the major components of a web application. Let's move towards our main motive How to build a web application in Java.

**Servlet:**

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client

and databases or applications on the HTTP server.

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet

The servlet is initialized by calling the init () method.

The servlet calls service() method to process a client's request.

The servlet is terminated by calling the destroy() method.

Finally, servlet is garbage collected by the garbage collector of the JVM.

The init() Method:

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So,

it is used for one-time initializations, just as with the init method of applets.

public void init() throws ServletException {

// Initialization code...

}

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming

from the client( browsers) and to write the formatted response back to the client.

public void service(ServletRequest request,

ServletResponse response)

throws ServletException, IOException{

}

The service () method is called by the container and service method invokes doGe,

doPost, doPut, doDelete, etc. methods as appropriate.

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections,

halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

COOKIES:

Cookies are small text files that are used to maintain the different session when a user online for shopping different items from same web store.

Cookies are send by the web server to the client browser.

A cookie contains the following contents

The name of the cookie

The value of the cookie

The expiration date of the cookie

The valid path of cookie

The domain of cookie

The secure connection

servlet to database :

Step 1: Creation of Database and Table in MySQL. ...

Step 2: Implementation of required Web-pages. ...

Step 3: Creation of Java Servlet program with JDBC Connection. ...

Step 4: To use this class method, create an object in Java Servlet program. ...

Step 5: Get the data from the HTML file.

**JSP:**

Elements of JSP

The elements of JSP have been described below:

The Scriptlet:

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Any text, HTML tags, or JSP elements you write must be outside the scriptlet.

Following is the syntax of Scriptlet:

<% code fragment %>

You can write the XML equivalent of the above syntax as follows:

<jsp:scriptlet>

code fragment

</jsp:scriptlet>

JSP declaration:

<%! declaration; [ declaration; ]+ ... %> eg.<%! int i = 0; %> <%! int a, b, c; %> <%! Circle a = new Circle(2.0); %>

You can write the XML equivalent of the above syntax as follows:

<jsp:declaration>

code fragment

</jsp:declaration>

JSP Expression:

A JSP expression element contains a scripting language expression that is evaluated,

converted to a String, and inserted where the expression appears in the JSP file.

Following is the syntax of JSP Expression:

<%= expression %>

You can write the XML equivalent of the above syntax as follows:

<jsp:expression>

expression

</jsp:expression>

JSP Comments:

<%-- This is JSP comment --%>

JSP Directives

A JSP directive affects the overall structure of the servlet class. It usually has the following

form:

<%@ directive attribute="value" %>

There are three types of directive tag:

Directive Description:

<%@ page ... %> Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.

buffer Specifies a buffering model for the output stream.

autoFlush Controls the behavior of the servlet output buffer.

contentType Defines the character encoding scheme.

errorPage Defines the URL of another JSP that reports on Java unchecked runtime exceptions.

isErrorPage :Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.

<%@ include ... %> Includes a file during the translation phase.

<%@ include file="relative url" >

<%@ taglib ... %> Declares a tag library, containing custom actions, used in the page

<%@ taglib uri="uri" prefix="prefixOfTag" >

JSP Actions:

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard:

<jsp:action\_name attribute="value" />

Action elements are basically predefined functions. Following table lists out the available JSP

Actions:

jsp:include Includes a file at the time the page is requested.

jsp:useBean Finds or instantiates a JavaBean.

jsp:setProperty Sets the property of a JavaBean.

jsp:getProperty Inserts the property of a JavaBean into the output.

jsp:forward Forwards the requester to a new page.

jsp:plugin Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin.

jsp:element Defines XML elements dynamically.

jsp:attribute Defines dynamically-defined XML element's attribute.

jsp:body Defines dynamically-defined XML element's body.

jsp:text Used to write template text in JSP pages and documents.