Java web Development

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:

1. 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.

First, understand servlet:

What is Servlet

A Servlet is a Java program that runs within a web server; it receives the requests and responds to them using related protocols (Usually HTTP). The Servlets are capable enough to respond to any type of request; they are commonly used to make the application functional.

We can create a static website using only HTML and CSS, but when it comes to dynamic, we need a server-side programming language. For these applications, Java provides Servlet technology, which contains HTTP-specific servlet classes.

The **javax.servlet** and **javax.servlet.http** packages contain interfaces and classes for creating servlets. All servlets should implement the Servlet interface, which defines life-cycle methods. To implement a generic service, we can use the GenericServlet class by extending it. It provides **doGet** and **doPost** methods to handle HTTP-specific services.

Why are the Servlets Useful?

Web servers are capable enough to serve static HTML requests, but they don't know how to deal with dynamic requests and databases. So, we need a language for dynamic content; these languages are PHP, Python, Java, Ruby on Rails, etc. In Java, there are two technologies Servlet and JSPs, that deals with dynamic content and database. Java also provides frameworks such as Spring, Spring Boot, Hibernate, and Struts to use the servlet and JSP easily.

The Servlets and JSPs are server-side technologies that extend the functionality of a web server. They support dynamic response and data persistence. We can easily create a web application using these technologies.

Servlet:-

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

Servlet echnology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology

Servlet is a technology which is used to create a web application.

Servlet is an API that provides many interfaces and classes including documentation.

Servlet is an interface that must be implemented for creating any Servlet.

Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.

Servlet is a web component that is deployed on the server to create a dynamic web page.

**Servlet Container**

Provide web server with servlet support

Execute and manage servlets E.g., Tomcat Must conform to the following lifecycle contract

Create and initialize the servlet Handle zero or more service calls from clients  Destroy the servlet and then garbage collect it

Three types

Standalone container

Add-on container

Embeddable container Usually execute all servlets in a single JVM

**Loading Servlet**

 Server calls servlet’s init() method

After the server constructs the servlet instance and before the servlet handles any requests  init()  Servlet initialization is defined  May be called … When the server starts When the servlet is first requested, just before the service() method is invoked  At the request of the server administrator

Removing Servlet

Server calls the destroy() method

After the servlet has been taken out of service and all pending requests to the servlet have completed or timed out

destroy() :-Resources acquired should be freed up A chance to write out its unsaved cached info Last step before being garbage collected

HttpServletRequest

Encapsulate all information from the client request

HTTP request header and request body

Methods to retrieve data

Inherited from ServletRequest

getParameter()

getParameterNames()

getParameterValues()

getInputStream()

getReader()

HttpServletResponse

Encapsulate all data to be returned to client

HTTP response header and response body (optional)

Set HTTP response header Primitive manipulation

setStatus(), setHeader(), addHeader()

Convenience methods

setContentType(), sendRedirect(), sendError()

Set HTTP response Body

Obtain a PrintWriter or ServletOutputStream to return data to the client

getWriter(), getOutputStream()

**JSP**

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

#### 4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The JSP pages follow these phases:

Translation of JSP Page

Compilation of JSP Page

Classloading (the classloader loads class file)

Instantiation (Object of the Generated Servlet is created).

Initialization ( the container invokes jspInit() method).

Request processing ( the container invokes \_jspService() method).

Destroy ( the container invokes jspDestroy() method).

# Java JDBC Tutorial

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

* JDBC-ODBC Bridge Driver,
* Native Driver,
* Network Protocol Driver, and
* Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.

It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular interfaces of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular classes of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

### Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

Register the Driver class

Create connection

Create statement

Execute queries

Close connection

To connect Java application with the MySQL database, we need to follow 5 following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class:**The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:**The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.
3. **Username:**The default username for the mysql database is **root**.
4. **Password:**It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.

But we can make this object to move forward and backward direction by passing either TYPE\_SCROLL\_INSENSITIVE or TYPE\_SCROLL\_SENSITIVE in createStatement(int,int) method as well as we can make this object as updatable by:

Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,

                     ResultSet.CONCUR\_UPDATABLE);

# PreparedStatement interface

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Let's see the example of parameterized query:

String sql="insert into emp values(?,?,?)";

As you can see, we are passing parameter (?) for the values. Its value will be set by calling the setter methods of PreparedStatement.

**Why use PreparedStatement?**

### Improves performance: The performance of the application will be faster if you use PreparedStatement interface because query is compiled only once.