UNIT-3	SERVER-SIDE PROGRAMMING
Servlets: Java Servlet Architecture - Servlet Life Cycle - Form GET and POST actions - Session Handling Understanding Cookies - DATABASE CONNECTIVITY: JDBC perspectives, JDBC program example - JSP: Understanding Java Server Pages - JSP Standard Tag Library (JSTL) - Creating HTML forms by embedding JSP code.	

SERVLETS

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

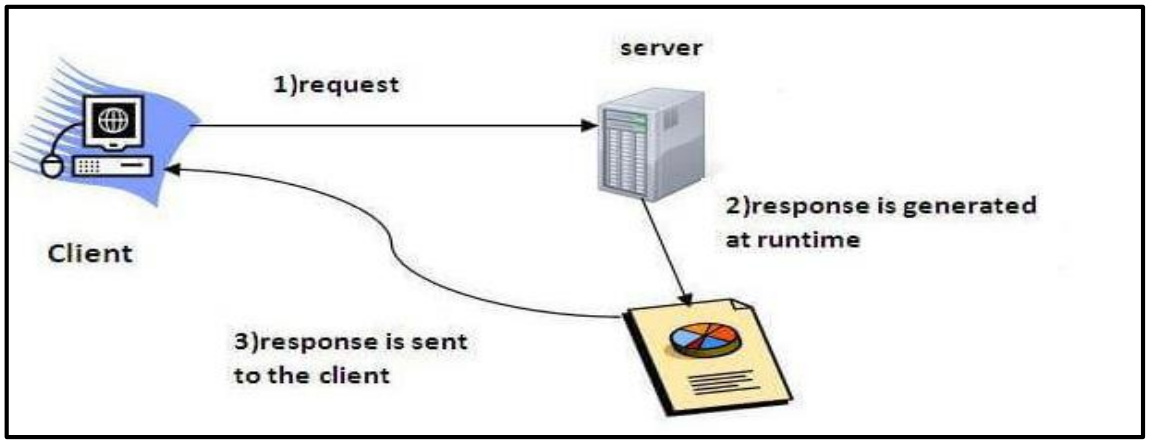
Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to the CGI technology. We have discussed the CGI disadvantages and Servlet advantages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

There **are** many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

1. What is a Servlet?

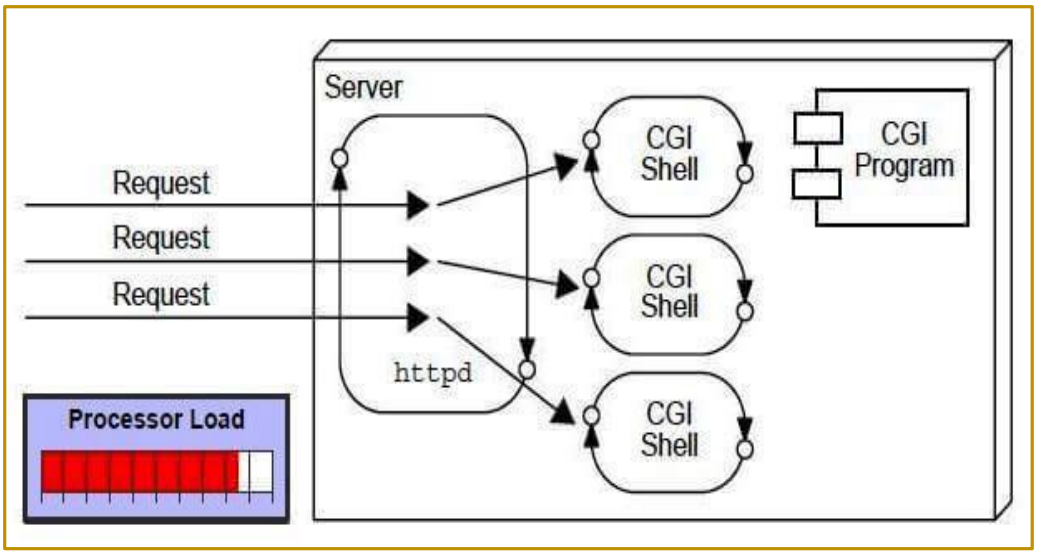
Servlet can be described in many ways, depending on the context.

- 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.



2. CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.

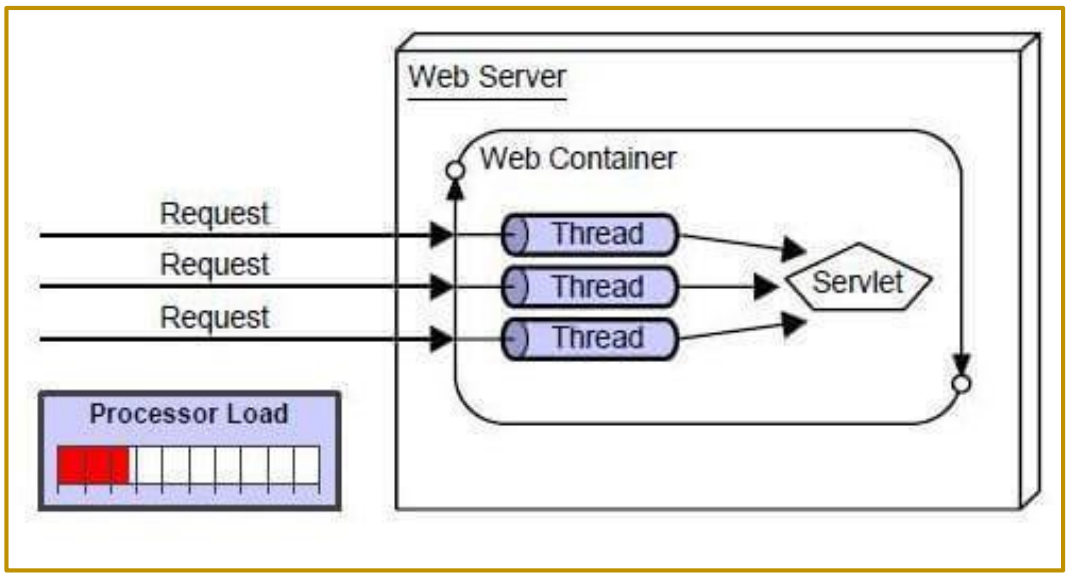


3. Disadvantages of CGI

There are many problems in CGI technology:

- If the number of clients increases, it takes more time for sending the response.
- For each request, it starts a process, and the web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.

4. Advantages of Servlet



There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

JAVA SERVLET ARCHITECTURE

What is Java Servlet?

Java Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.

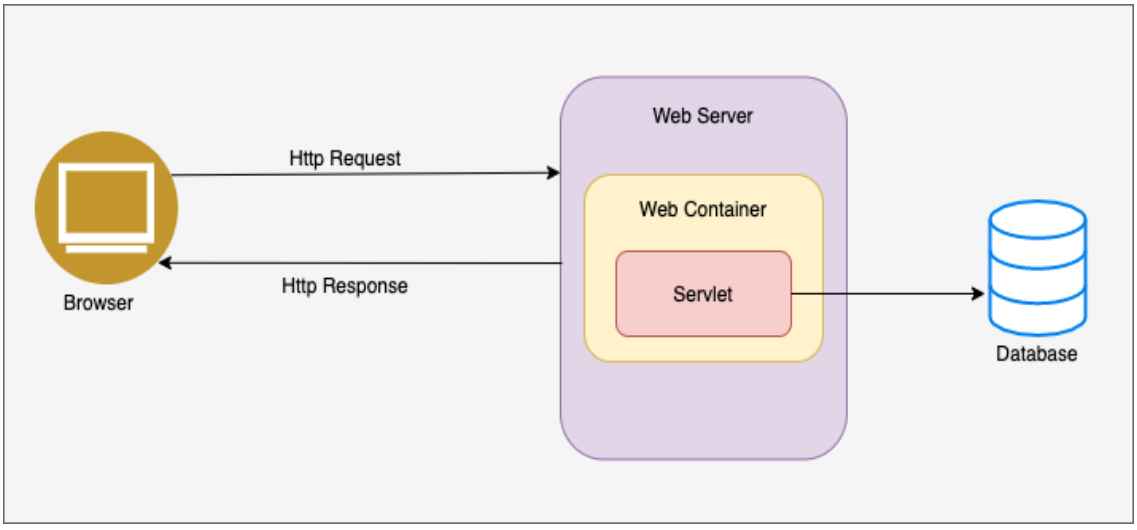
Properties of Java Servlet

The properties of Servlets are as follows:

- Servlets work on the server side.
- Servlets are capable of handling complex requests obtained from the web server.

Java Servlets Architecture

Servlet Architecture can be depicted from the image itself as provided below as follows:



Execution of Java Servlets

Execution of Servlets basically involves Six basic steps:

1. The Clients send the request to the Web Server.
2. The Web Server receives the request.
3. The Web Server passes the request to the corresponding servlet.
4. The Servlet processes the request and generates the response in the form of output.
5. The Servlet sends the response back to the webserver.
6. The Web Server sends the response back to the client and the client browser displays it on the screen.

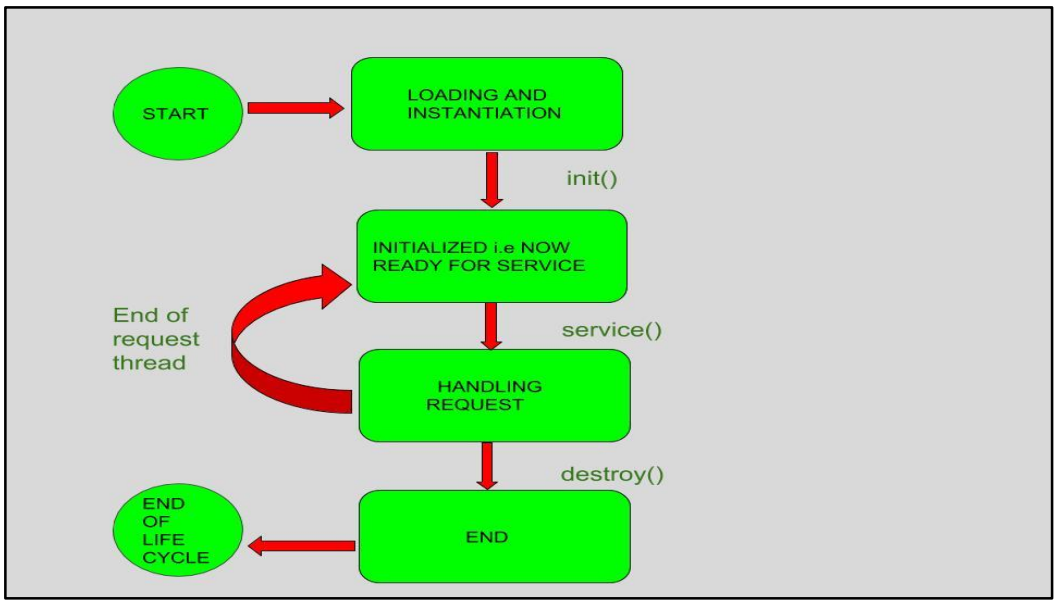
SERVLET LIFE CYCLE

The web container maintains the life cycle of a servlet instance. The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling.
- Destroying the Servlet.

1) Loading a Servlet

The first stage of the Servlet lifecycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages:



- Initializing the context, on configuring the Servlet with a zero or positive integer value.
- If the Servlet is not preceding stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request.

2) Initializing a Servlet:

After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object.

The container initializes the Servlet object by invoking the Servlet.init (ServletConfig) method which accepts ServletConfig object reference as parameter.

The Servlet container invokes the Servlet.init (ServletConfig) method only once, immediately after the Servlet.init (ServletConfig) object is instantiated successfully. This method is used to initialize the resources, such as JDBC data source.

3) Handling request:

After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request:

- It creates the **ServletRequest** and **ServletResponse** objects.
- After creating the request and response objects it invokes the Servlet.service (ServletRequest, ServletResponse) method by passing the request and response objects.

4) Destroying a Servlet:

When a Servlet container decides to destroy the Servlet, it performs the following operations,

- It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
- After currently running threads have completed their jobs, the Servlet container calls the **destroy()** method on the Servlet instance.

FORM GET AND POST ACTIONS

HTTP methods declare what action is to be performed on the data that is submitted to the server. The HTML <form> method Attribute is used to specify the HTTP method used to send data while submitting the form.

HTTP Protocol provides several methods, and the HTML Form element is able to use two methods to send user data:

- **GET method** - used to request data from a specified resource
- **POST method** - used to send data to a server to update a resource

Syntax:

```
<form method="get | post">
```

The GET Method

The HTML GET method is used to get a resource from the server. For example,

```
<form method="get" action="www.rec.com/search">
   <input type="search" name="location" placeholder="Search.." />
   <input type="submit" value="Go" />
</form>
```

When we submit the above form by entering **Chennai** in the input field, the request sent to the server will be **www.rec.com/search/?location=Chennai.**

The HTTP GET method adds a query string at the end of the URL to send data to the server. The query string is in the form of key-value pair followed **by ? symbol**.

From the URL, the server can parse the user-submitted value where:

- **key -** location
- **value -** California

The POST method

The HTTP POST method is used to send data to the server for further processing. For example,

```
<form method="post" action="www.programiz.com/user">
   <label for="firstname">First name:</label>
   <input type="text" name="firstname" /><br />
   <label for="lastname">Last name:</label>
   <input type="text" name="lastname" /><br />
   <input type="submit" />
</form>
```

When we submit the form, it will add the user input data to the body of the request sent to the server. The request would look like

```
POST /user HTTP/2.0
Host: www.rec.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
firstname=Robin&lastname=Williams
```

The data sent is not easily visible to the user. However, we can check the sent data using special tools like the browsers' dev tools.

SESSION HANDLING

Session-handling customization is the process of manipulating server responses in such a way that application state information is preserved during load testing.

HTTP is a “stateless” protocol, which means that each time a client requests a Web page, the client establishes a new connection with the Web server, and the server does not retain track of prior requests.

The conversation of a user over a period of time is referred to as a session. In general, it refers to a certain period of time.

The recording of the object in session is known as tracking.

Session tracking is the process of remembering and documenting customer conversations over time. Session management is another name for it.

Why is Session Tracking Required?

Because the HTTP protocol is stateless, we require Session Tracking to make the client-server relationship stateful.

Session tracking is important for tracking conversions in online shopping, mailing applications, and E-Commerce applications.

The HTTP protocol is stateless, which implies that each request is treated as a new one. As you can see in the image below.

Deleting Session Data

- Delete your whole session. To delete an entire session, use the public void invalidate() function.
- Setting Session Timeout You may set the timeout for a session separately by calling the public void setMaxInactiveInterval(int interval) function.

- Log the user out On servers that support servlets 2.4, you may use the logout method to log the client out of the Web server and invalidate all of the users' sessions.
- web.xml Configuration If you're using Tomcat, you may set the session timeout in the web.xml file, in addition to the ways listed above.

```
<session-config>
  <session-timeout>20</session-timeout>
</session-config>
```

- The timeout is specified in minutes and overrides Tomcat's default timeout of 30 minutes.

Session Tracking Employs Four Different techniques

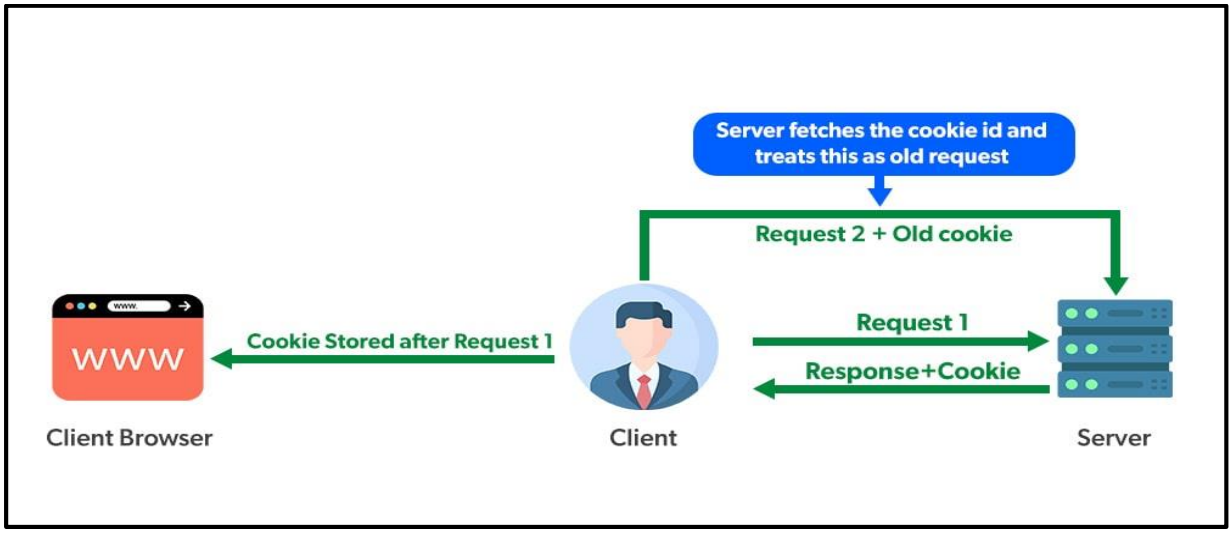
1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

COOKIES

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.

Cookies are the textual information that is stored in key-value pair format to the client's browser during multiple requests. It is one of the state management techniques in session tracking.

How Cookie Works?



Basically, the server treats every client request as a new one so to avoid this situation cookie are used. When the client generates a request, the server gives the response with cookies having an id which are then stored in the client's browser.

Thus if the client generates a second request, a cookie with the matched id is also sent to the server. The server will fetch the cookie id, if found it will treat it as an old request otherwise the request is considered new.

Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

1) Non-persistent cookie

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

2) 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 sign-out.

Advantage of Cookies

- Simplest technique of maintaining the state.
- Cookies are maintained at client side.

Disadvantage of Cookies

- It will not work if cookie is disabled from the browser.
- Only textual information can be set in Cookie object.

DATABASE CONNECTIVITY – JDBC PERSPECTIVES

JDBC (Java Database Connectivity) is an API in Java that enables applications to interact with databases. It allows a Java program to connect to a database, execute queries, and retrieve and manipulate data.

By providing a standard interface, JDBC ensures that Java applications can work with different relational databases like MySQL, Oracle, PostgreSQL, and more.

It is a part of **JavaSE** (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.

- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

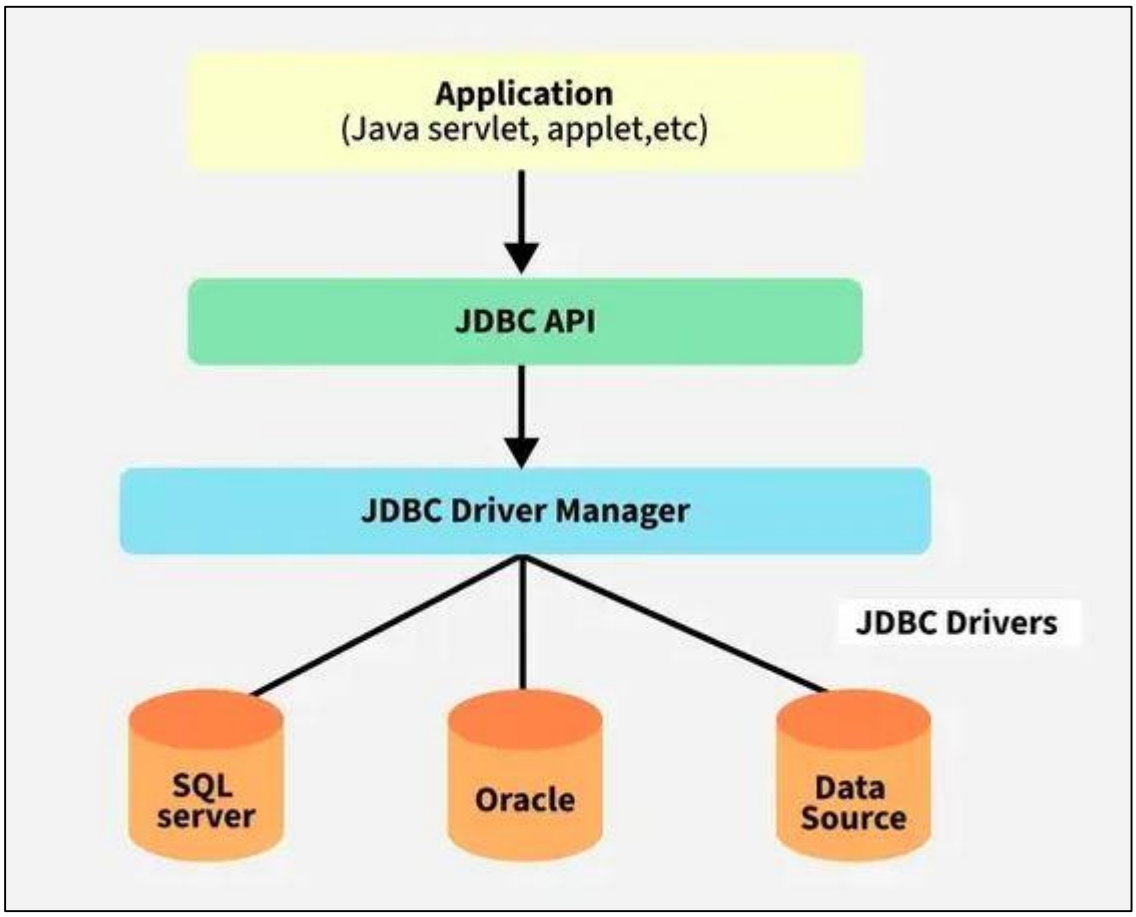
Why Should We Use JDBC?

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But ODBC API uses ODBC driver that 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 Architecture



Explanation:

- **Application:** It is a Java applet or a servlet that communicates with a data source.
- **The JDBC API:** It allows Java programs to execute SQL queries and retrieve results. Key interfaces include Driver, ResultSet, RowSet, PreparedStatement, and Connection. Important classes include DriverManager, Types, Blob, and Clob.
- **DriverManager:** It plays an important role in the JDBC architecture. It uses some database-specific drivers to effectively connect enterprise applications to databases.
- **JDBC drivers:** These drivers handle interactions between the application and the database.

JDBC Components

There are generally **4 main components of JDBC** through which it can interact with a database. They are as mentioned below:

1. **JDBC API**

   It provides various methods and interfaces for easy communication with the database. It includes two key packages

   - **java.sql**: This package, is the part of **Java Standard Edition (Java SE),** which contains the core interfaces and classes for accessing and processing data in relational databases.
   - **javax.sql**: This package is the part of **Java Enterprise Edition (Java EE),** which extends the capabilities of **java.sql** by offering additional features like connection pooling, statement pooling, and data source management.

2. **JDBC Driver Manager**

   Driver manager is responsible for loading the correct database-specific driver to establish a connection with the database. It manages the available drivers and ensures the right one is used to process user requests and interact with the database.

3. **JDBC Test Suite**

   It is used to test the operation (such as insertion, deletion, updating) being performed by JDBC Drivers.

4. **JDBC Drivers**

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

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

Steps to Connect to MySQL Database Using JDBC

Step 1: Load the JDBC Driver

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

Step 2: Establish a Connection

```
Connection connection = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/your_database",
"your_username",
"your_password"
);
```

Step 3: Create a Statement

```
Statement statement = connection.createStatement();
```

Step 4: Execute a Query

```
String query = "INSERT INTO students (id, name) VALUES
(101, 'John Doe')";
int rowsAffected = statement.executeUpdate(query);
System.out.println("Rows affected: " + rowsAffected);
```

Step 5: Close the Connection

```
statement.close();
connection.close();
```

Key Features

- **Platform Independence**: It enables database operations across different platforms.
- **Standard API**: It provides a uniform interface for various databases.

- **Support for Multiple Databases**: It works with popular databases like MySQL, PostgreSQL, Oracle, etc.
- **Extensibility**: It offers features like batch processing, connection pooling, and transaction management.

JDBC Program Example

```
import java.sql.*;


public class SelectExample {
   static final String DB_URL = "jdbc:mysql://localhost/Student";
   static final String USER = "guest";
   static final String PASS = "guest123";
   static final String QUERY = "SELECT id, first, last, age FROM Students";
   public static void main(String[] args) {
      // Open a connection
      try(Connection conn = DriverManager.getConnection(DB_URL, USER,
PASS);
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(QUERY);) {
         // Extract data from result set
         while (rs.next()) {
            // Retrieve by column name
            System.out.print("ID: " + rs.getInt("id"));
            System.out.print(", Age: " + rs.getInt("age"));
            System.out.print(", First: " + rs.getString("first"));
            System.out.println(", Last: " + rs.getString("last"));
         }
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Output

Now let us compile the above example as follows −

C:\> **javac SelectExample.java**

C:\>

When you run **SelectExample**, it produces the following result −

C:\> **java SelectExample**

Connecting to database...

Creating statement...

ID: 100, Age: 18, First: Zara, Last: Ali

ID: 101, Age: 25, First: Mahnaz, Last: Fatma

ID: 102, Age: 30, First: Zaid, Last: Khan

ID: 103, Age: 28, First: Sumit, Last: Mittal

JSP: Understanding Java Server Pages

JavaServer Pages (JSP) is a server-side technology that creates dynamic web applications. It allows developers to embed Java code directly into HTML or XML pages and it makes web development more efficient.

JSP is an advanced version of Servlets. It provides enhanced capabilities for building scalable and platform-independent web pages.

How is JSP More Advantageous than Servlets?

JSP simplifies web development by combining the strengths of Java with the flexibility of HTML. Some of the advantages of JSP over Servlets are listed below:

- JSP code is easier to manage than Servlets as it separates UI and business logic.
- JSP minimizes the amount of code required for web applications.
- Easily generates content dynamically in response to user interactions.
- It provides access to the complete range of Java APIs for robust application development.
- JSP is suitable for applications with growing user bases.

Key Features of JSP

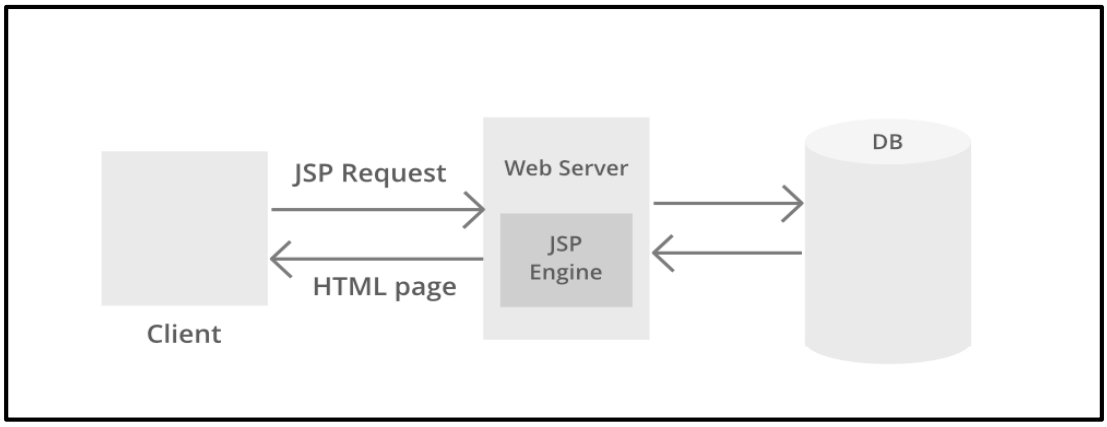
- **Platform Independence:** Write once, run anywhere.
- It simplifies database interactions for dynamic content.

- It contains predefined objects like request, response, session, and application reduce development time.
- It has built-in mechanisms for exception and error management.
- It supports custom tags and tag libraries.

JSP Architecture

JSP follows a three-layer architecture:

1. **Client Layer:** The browser sends a request to the server.
2. **Web Server Layer:** The server processes the request using a JSP engine.
3. **Database/Backend Layer**: Interacts with the database and returns the response to the client.



Steps to Create a JSP Application

JSP allows us to embed Java code within HTML pages and making it easy to create dynamic content. Let us start with a simple exercise to convert an existing HTML file into a JSP file.

Steps to Create a JSP

1. Take any HTML file you have previously created.
2. Change the file extension from .html to .jsp.
3. Load the new .jsp file in browser.

When we load a JSP file for the first time:

- The JSP is converted into a Java file.
- The Java file is compiled into a servlet.
- The compiled servlet is loaded and executed.

Adding Dynamic Content with JSP

Here is an example to demonstrate how JSP can generate dynamic content:

hello.jsp:

```
<!DOCTYPE html>
<html>
<body>
    Hello! The time is now <%= new java.util.Date() %>
</body>
</html>
```

Why Use JSP?

JSP is powerful because it allows us to:

- Embed Java logic directly into HTML.
- To create dynamic pages that respond to user actions.
- To customize content for each user or session.

JSP Standard Tag Library (JSTL)

The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications.

JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating the existing custom tags with the JSTL tags.

JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description
Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.

Formatting tags	The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

Example of JSTL:

The <c:set> Tag with body

variable can be set by using the <c:set> tag within the body of another tag as in the following code snnipate:

```
<c:set var="bookname">
My book: java 8
</c:set>
<c:out value="${bookname}"/>
```

In the preceding code snippet, the value for the bookname variable is set in the body of the <c:set> tag. Here <c:out> tag is used for printing the output.

How to download and install JSTL:

1. Download JSTL.jar and Standard.jar file from here (or you will get these from your native Apache tomcat installation too!).
2. Now put both the files into your 'WEB-INF/lib' folder.
3. After this, add them to classpath also.
4. Finally, you can use JSTL into your project.

Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

Creating HTML forms by embedding JSP code

SimpleForm.html

```
<html>
  <body>
    <h1>Please tell me about yourself</h1>

    <form action="SimpleFormHandler.jsp" method="get">
      Name:
      <input type="text" name="firstName" placeholder="Enter Firstname" />
      <input type="text" name="lastName" placeholder="Enter Lastname" /><br />
      <br />Sex: <input type="radio" checked name="sex" value="male" />Male
      <input type="radio" name="sex" value="female" />Female
      <p>
        What Java primitive type best describes your personality:
        <select name="javaType">
          <option value="boolean">boolean</option>
          <option value="byte">byte</option>
          <option value="char" selected>char</option>
          <option value="double">double</option>
          <option value="float">float</option>
          <option value="int">int</option>
          <option value="long">long</option>
        </select>
        <br /><br />
        <input type="submit" />
      </p>
    </form>
  </body>
</html>
```

SimpleFormHandler.jsp

```
<html>
 <body>
        <%
             // Grab the variables from the form.
             String firstName =  request.getParameter("firstName");
            String lastName =   request.getParameter("lastName");
            String sex = request.getParameter("sex");
            String javaType =  request.getParameter("javaType");
        %>
        <%-- Print out the  variables. --%>
        <h1>Hello, <%=firstName%> <%=lastName%>!</h1>
         I see that you are <%=sex%>. You know, you remind me of a
        <%=javaType%> variable I once knew.
 </body>
</html>
```

Output


Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/SimpleForm
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop
Hello, JSP Programmer!
I see that you are male. You know, you remind me of a char variable I once knew.
Document: Done (20.9 secs)

JSTL (JavaServer Pages Standard Tag Library) is a collection of useful tags that simplify tasks commonly required in JSP pages, such as conditional statements, iteration, and formatting. By using JSTL tags, you can avoid embedding Java code directly into your JSP pages (e.g., using scriptlets <% %>), leading to cleaner, more maintainable code.

JSTL Tags

There JSTL mainly provides five types of tags: **Tag Name**	**Description**
Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<c:set var="message" value="Hello, world!" />
<p>${message}</p>
```

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<c:if test="${user != null}">
    <p>Welcome, ${user.name}!</p>
</c:if>
<c:if test="${user == null}">
    <p>Please log in.</p>
</c:if>
```

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<c:forEach var="item" items="${items}">
    <p>${item}</p>
</c:forEach>
```

In JSP (JavaServer Pages), the <%= %> tag is used to embed Java code directly into the HTML content. This is known as a scriptlet. The code inside the <%= %> tag is executed on the server and the result is inserted into the HTML output sent to the client.

When to Use <%= %> in JSP:

You use <%= %> in JSP to insert dynamic content or Java expressions into the HTML output. The code inside the <%= %> is evaluated on the server side, and the resulting value is included in the response sent to the browser.

In JSP (JavaServer Pages), the <% %> tag is used to define a scriptlet, which allows you to embed Java code directly into your JSP page. This code is executed on the server side before the page is rendered and sent to the browser.

When to Use <% %> in JSP

The <% %> tag is used when you need to embed Java logic that processes or manipulates data before generating the output. This can include things like variable declarations, loops, conditional logic, or calling methods. The code inside these tags is executed on the server, and its result is not directly inserted into the HTML; instead, it controls the page's behavior.

```
<%
   public String getGreeting()
{
      return "Welcome to JSP!";
   }
%>
<h1><%= getGreeting() %></h1>
```

```
<%
   int a = 5;
   int b = 3;
%>
<p>The sum of <%= a %> and <%= b %> is <%= a + b %>.</p>
```