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.	

1️⃣ Introduction to Server-Side Programming

Server-Side Programming refers to programs that run on the **web server** to generate dynamic web content before it is sent to the client (browser).

👉 Instead of sending the same static HTML every time, the server processes logic, databases, authentication, etc., and then returns customized output.

Example:
When you log in to Gmail, the inbox you see is generated server-side.

2️⃣ How Server-Side Programming Works

Step-by-Step Flow

1. User opens a browser.
2. User sends a request (URL / form submission).
3. Request reaches the **Web Server**.
4. Server runs server-side script.
5. Script may interact with database.
6. Server generates HTML response.
7. Response sent back to browser.

Architecture:

Client → Request → Server → Processing → Response → Client

3️⃣ Client-Side vs Server-Side Programming

Feature	**Client-Side**	**Server-Side**
Runs on	Browser	Server
Languages	JavaScript	PHP, Python, Java, Node.js
Speed	Faster (local)	Slower (network involved)
Security	Less secure	More secure
DB Access	No	Yes

4️⃣ Server-Side Scripting Languages

Common languages used:

- **PHP**
- **Python**
- **Java (JSP / Servlets)**
- **Node.js**
- **Ruby**
- **.NET**

Example

PHP:

```
<?php
echo "Hello from Server";
?>
```

Python (Flask):

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
   return "Hello Server Side"
```

5️⃣ Features of Server-Side Programming

- Dynamic page generation
- Database interaction
- User authentication
- Session management
- File handling
- Form processing
- Security validation

6️⃣ Web Server

A **Web Server** is software that handles HTTP requests.

Examples:

- Apache
- Nginx
- IIS

Functions:

- Receives requests
- Runs scripts
- Sends responses
- Manages sessions

7️⃣ Database Connectivity

Server-side programs connect to databases to store and retrieve data.

Popular Databases:

- MySQL
- PostgreSQL
- MongoDB

- Oracle

Example Process

1. User submits login form
2. Server script runs
3. Query sent to DB
4. Result returned
5. Access granted/denied

8️⃣ Form Handling

Forms send user data to the server.

HTML Form Example:

```
<form method="POST" action="login.php">
  Username: <input type="text" name="user">
  <input type="submit">
</form>
```

Server processes input securely.

9️⃣ Session Management

HTTP is stateless → server uses sessions to track users.

Techniques:

- Sessions
- Cookies
- Tokens

Uses:

- Login systems
- Shopping carts
- User dashboards

🔟 Cookies

Cookies are small data stored in the browser but set by the server.

Types:

- Session cookies
- Persistent cookies

Uses:

- Remember login
- Preferences
- Tracking

1️⃣1️⃣ Authentication & Authorization

- **Authentication** → Who are you? (Login)
- **Authorization** → What can you access?

Example:

- Student login
- Admin panel access

1️⃣2️⃣ Advantages of Server-Side Programming

✅ Dynamic content
✅ Secure data handling
✅ Database integration
✅ Centralized control
✅ Scalable applications

1️⃣3️⃣ Disadvantages

❌ Server load increases
❌ Slower response vs client-side
❌ Requires hosting
❌ Development complexity

1️⃣4️⃣ Applications of Server-Side Programming

- E-commerce websites
- Social media platforms
- Online banking
- Learning Management Systems
- Email services

1️⃣5️⃣ Example: Login Process Flow

1. User enters username/password
2. Data sent to server
3. Server validates with DB
4. Session created
5. Dashboard displayed

1️⃣6️⃣ MVC Architecture (Overview)

Model → Data & DB
View → UI
Controller → Logic

Used in:

- Django
- Laravel
- Spring
- ASP.NET

Java

Java entered the programming world in 1995, and since then has skyrocketed to become one of the most widely used languages globally, ranking 6th in popularity. Not just a language, Java is a robust platform and ecosystem that flexes its muscles across various domains. From crafting desktop applications to building massive web portals and services, Java has got you covered. It even stretches its reach to software development for devices like PCs, tablets, smartphones and beyond.

Advantages:

- **Scalable and simple**

Java shines when it comes to creating scalable applications. Its object-oriented approach allows for constructing large, flexible and extensible web applications that can run multiple instances simultaneously.

- **Multi-threading**

Harnessing the power of independent threads, Java thrives in applications that can run multiple instances simultaneously.

- **Open-source code**

With Java's open-source code, a thriving community has sprouted. They've crafted a treasure trove of free libraries and frameworks, turbocharging server-side development.

- **Enhanced security**

Java takes security seriously. Its virtual machine diligently checks Java byte codes, ensuring viruses don't wreak havoc. Plus, its security model and reusable code testing add extra layers of protection.

Disadvantages:

- **Licensing**

For commercial use, a license is required, adding a bureaucratic hurdle.

- **Performance quirks**

Java can be a bit slow due to its reliance on a computer virtual machine for compilation. Memory cleanup issues, thread blocking and caching settings can hinder performance.

- **Native design challenges**

Creating a native design in Java often requires third-party libraries and tools.

- **Complex and verbose code**

Java’s syntax can sometimes be overwhelming, leading to verbose and intricate code.

Introduction to Java Servlets

Java Servlet is a Java program that runs on a Java-enabled web server or application server. It handles client requests, processes them and generates responses dynamically. Servlets are the backbone of many server-side Java applications due to their efficiency and scalability.

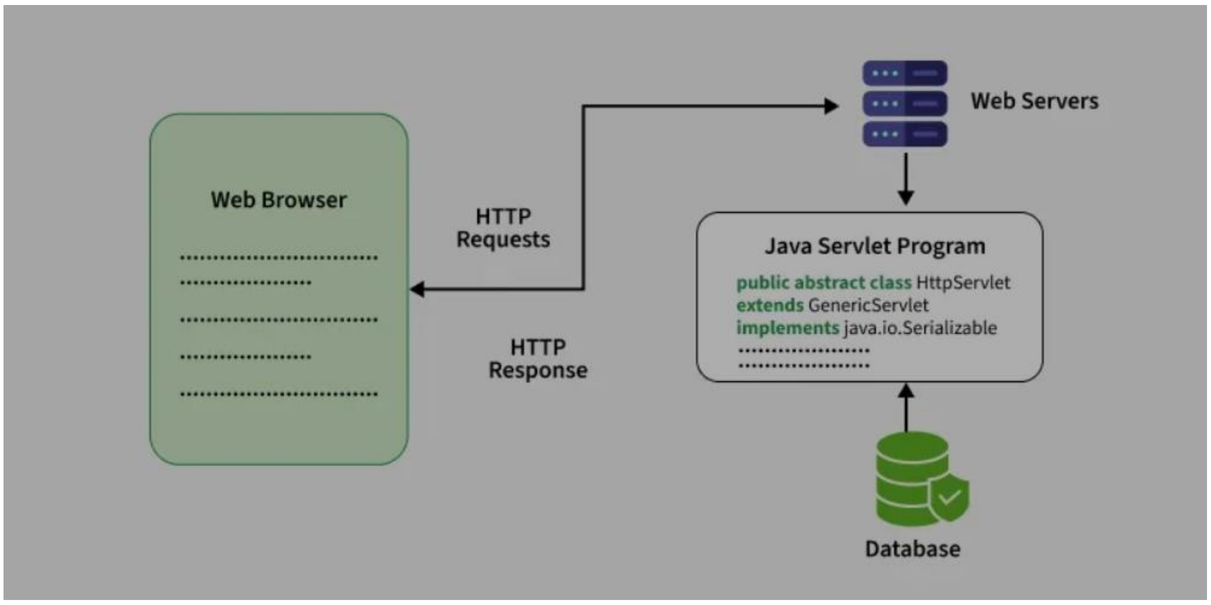
Features of Java Servlets

- Work on the server-side.
- Efficiently handle complex client requests.
- Generate dynamic responses.
- Provide better performance compared to older technologies like CGI.
- Highly scalable in enterprise-level web applications.

Java Servlets Architecture

Java servlets container play a very important role. It is responsible for handling important tasks like load balancing, session management and resource allocation, it make sure that all the requests are process efficiently under high traffic. The container distribute requests accross multiple instances, which helps improve the system performance.

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



The Role of a Servlet Container

The servlet container (sometimes called the servlet engine) is the core component that enables a standard web server to work with Java servlets. It provides essential services:

- Lifecycle Management: Manages the loading, initialization, and eventual destruction of servlets.
- Communication: Decodes incoming HTTP requests and encodes the dynamic responses generated by the servlet.
- Session Management: Maintains user sessions and tracks user progress across multiple web pages (e.g., for shopping carts).
- Security: Handles authentication and authorization processes.
- Resource Management: Manages static and dynamic resources within the web application's structured directory hierarchy.

Key Web Servers and Servlet Containers

Popular web servers and application servers that run servlets include:

- Apache Tomcat: The most widely used open-source servlet container, which also functions as a standalone web server. It is the official reference implementation for Java Servlet and JSP specifications.

The Role of JavaServer Pages (JSP)

JSP technology builds upon servlets, providing a more intuitive way to create dynamic web pages by embedding Java code directly within HTML markup.

- **Definition:** JSP pages are text-based documents with a .jsp extension that contain static HTML and dynamic content elements (tags, scriptlets, expression language).
- **Workflow:** A JSP engine first translates a JSP page into a Java servlet and then compiles it into bytecode. This generated servlet then handles subsequent client requests.
- **Advantages:**
    - **Simplified Development:** Easier for front-end developers and designers to work with, as they start with HTML and embed Java logic.
    - **Separation of Concerns:** Encourages better separation of presentation logic (HTML) from business logic (Java code, often in JavaBeans or separate servlets).