

Introduction to Server-side Web Applications

Introduction to Server-Side Web Applications

(Designed as quick-reference notes for students or developers new to the topic)

1. What Is a Server-Side Web Application?

Aspect	Description
Definition	A program that runs on a web server, processes HTTP requests, performs business logic, accesses data stores, and returns dynamic responses (HTML, JSON, XML, etc.) to clients (browsers, mobile apps, other services).
Contrast with client-side	• Client-side code (HTML/CSS/JS) runs in the browser. • Server-side code runs on the server, hidden from the user, and can securely handle data, authentication, and heavy computation.
Typical workflow	1. Browser ? HTTP request (GET/POST/PUT/DELETE). 2. Server-side app receives request, routes it. 3. Executes business logic, queries databases, calls external services. 4. Generates response (HTML page, API payload, redirect, etc.). 5. Browser ? HTTP response.

2. Core Components

Component	Role	Common Technologies
Web Server	Accepts TCP connections, handles low-level HTTP, forwards requests to the app.	Apache, Nginx, Microsoft IIS, Caddy
Application Runtime	Executes the server-side code; provides request/response objects, session handling, etc.	Node.js, JVM, .NET CLR, PHP interpreter, Python interpreter
Framework / Library	Supplies routing, templating, ORM, security, middleware, etc., to speed development.	Express (Node), Django/Flask (Python), Spring (Java), ASP.NET Core, Laravel (PHP)
Database	Persists structured/unstructured data for the app.	PostgreSQL, MySQL, MongoDB, Redis, SQLite

Introduction to Server-side Web Applications

Component	Role	Common Technologies
Template Engine	Merges data with HTML (or other formats) to produce the final response.	EJS, Handlebars, Jinja2, Thymeleaf, Blade
Middleware	Small, reusable units that process requests/responses (e.g., logging, auth, compression).	Express middleware, Django middleware, ASP.NET pipeline
Static Asset Server	Serves CSS, JS, images, fonts, etc. (often the same web server or a CDN).	Nginx, Cloudflare CDN, AWS S3 + CloudFront

3. Request?Response Cycle (Detailed)

1. Client sends request

- Method: GET, POST, PUT, DELETE, PATCH, OPTIONS
- URL: /products/42
- Headers: Accept, Cookie, Authorization, etc.
- Body (optional): form data, JSON, multipart files.

2. Web server receives TCP packet, parses HTTP, forwards to the application (via CGI, FastCGI, WSGI, ASGI, or a reverse?proxy to a process manager).

3. Routing

- Framework matches URL + method to a *controller / handler* function.
- Example (Express): `app.get('/products/:id', handler)`

4. Middleware chain (optional)

- Executes in order: logging ? security headers ? body parsing ? authentication ? rate limiting ? ...

5. Controller / Handler

- Validates input (schema, sanitization).
- Performs business logic (calculations, calls to services).
- Interacts