SPRING CORE

DAY-1 09/11/2025

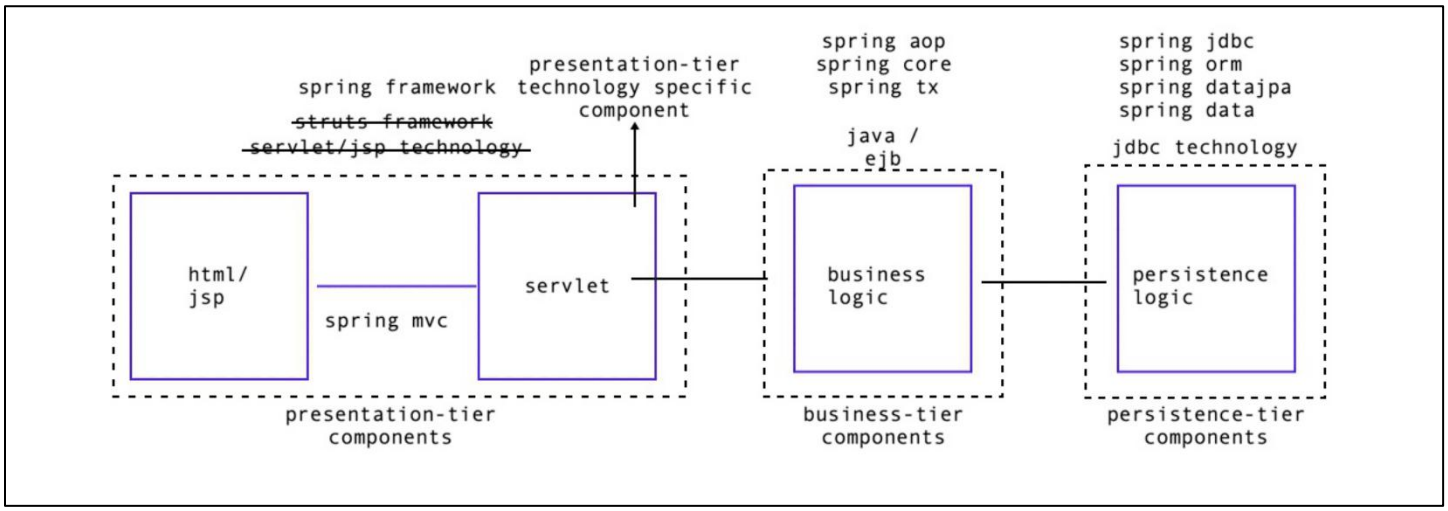
1. Overview of Web Application Components

A typical web application is built using **3-tier architecture**, ensuring modularity, reusability, and maintainability. These tiers divide responsibilities clearly among presentation, business and persistence layers.

a. Presentation Tier Components
   a. Handles **user interaction** and acts as the **face of the application.**
   b. Responsible for **receiving requests from the end-user** and **sending response back.**
   c. Examples: HTML, JSP Servlets, Spring MVC.
   d. These components **define how the user interacts with the System – not what business operation happen.**

b. Business Tier Components
   a. Contains **business logic –** the core rules and operations of the application.
   b. Validates data, applies computations, enforces business rules, and communicates with persistence layer.
   c. Typically implemented **Java classes, Spring Core, Spring AOP, Spring TX, or EJBs.**

c. Persistence Tier Components
   a. Responsible for **data storage, retrieval, and management.**
   b. Interact with underlying databases or files.
   c. Technologies: Spring JDBC, Spring ORM, Spring Data JPA, Hibernate, etc.
   d. Encapsulated through DAO (Data Access Object) classes.



2. Why We Shouldn't Write Business & Persistence Logic Inside Servlets

Even though Servlets can technically handle all logic, it's considered **bad design practice.**
There are **two main reasons:**

Reason 1: Servlets Are Presentation-Tier Specific Components

- A Servlet is tightly coupled with **Servlet/JSP technology.**
- If in future, the project migrates from **Servlets** to another presentation framework (e.g., **Struts, JSF, Spring MVC,** or **Portlets**), all business and persistence logic written inside the Servlet will also need to be re-written.
- This means we lose not only the presentation logic but also all functional code.
- Result: **High redevelopment cost, poor reusability,** and **tight coupling** between layers.

Reason 2: Servlets Are Managed by Servlet Container

- A Servlet is controlled by the Servlet Container (e.g., Tomcat, Jetty).
- It cannot be instantiated or invoked directly by other classes outside the container.
- Therefore, if business or persistence logic is embedded in a Servlet, it becomes non-reusable by any other class or context (like CLI tools, batch jobs, or REST services).
- This breaks the principle of separation of concerns (SoC) and reusability.

Summary Points:

Servlets should focus only on handling HTTP requests/responses, delegating business logic to Service Layer and persistence logic to DAO Layer.

3. Advantages of Layered (Tiered) Architecture

Layer	Responsibility	Benefit
Presentation Tier	Handles input/output, user interaction	Easy to modify UI without affecting logic
Business Tier	Holds core logic & rules	Centralized rules, easier testing & reuse
Persistence Tier	Handles data storage/retrieval	Easier DB maintenance and flexibility

Key Benefits:

- Separation of Concerns: Each tier performs a distinct role.
- Reusability: Logic can be reused across multiple apps or interfaces.
- Maintainability: Easier to debug, test, and modify one layer without touching others.
- Flexibility: UI or DB tech can change independently of the core logic.
- Scalability: Layers can be scaled or distributed as needed.

4. Example – How It All Connects

a. End-User – interacts through browser (HTML/JSP).
b. Servlet (Presentation Layer) – receivers request, validates input format, calls Business Layer.
c. Business Logic Layer – performs validation, rules, calculations, then call Persistence Layer.
d. Persistence Layer – performs actual CRUD operations with the database.
e. Response – flows back in reverse: Persistence – Business – Servlet – JSP/HTML – User.

5. Core Principle Demonstrated

"Keep Presentation, Business, and Persistence Layers Independent.".

Mixing all logic into Servlets Causes:

- Tight coupling with the servlet container.
- Loss of flexibility for technology change.
- Reduced maintainability and testability.
- Poor code organization and duplication.

6. Conclusion

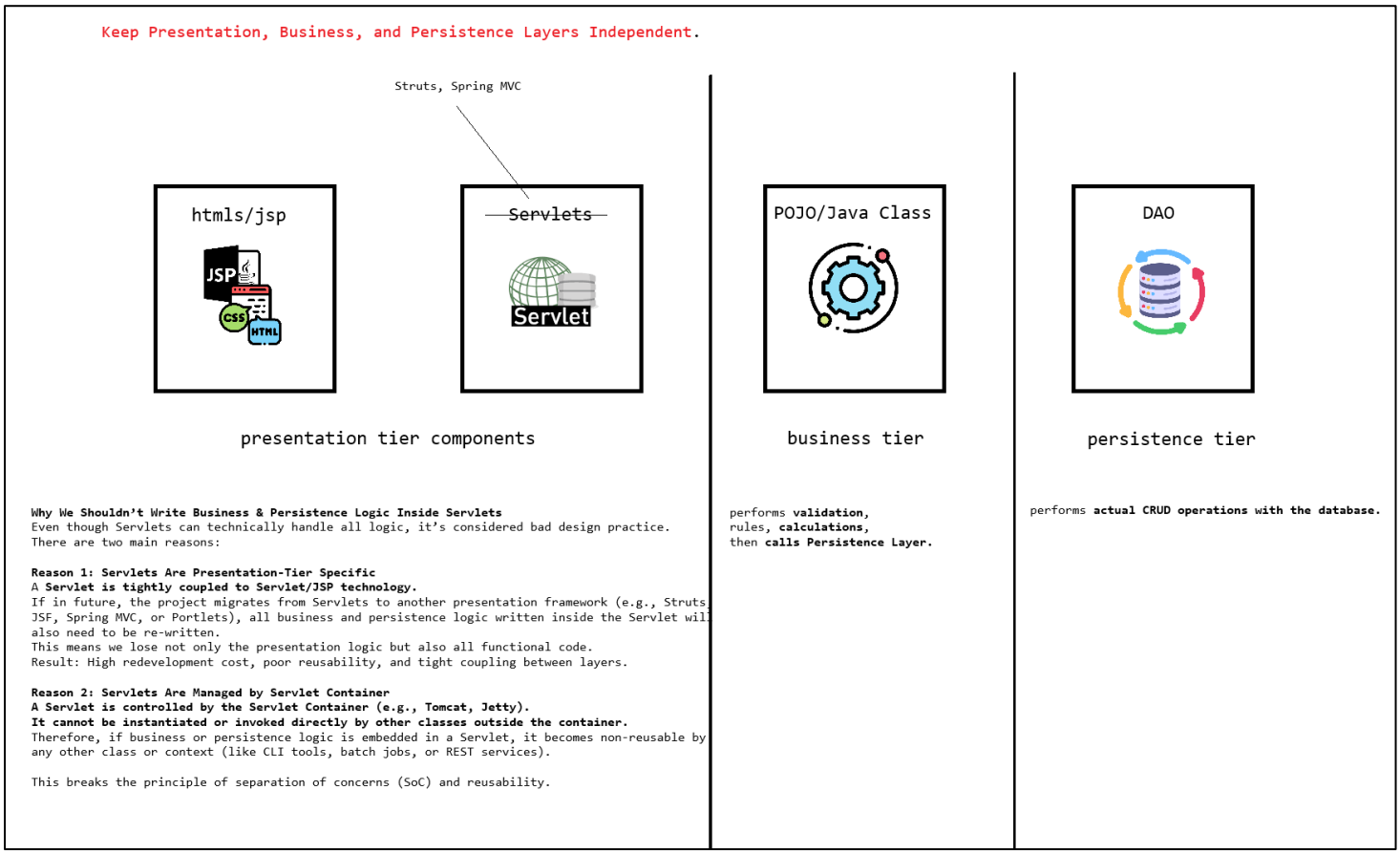
A well-structed web application separates concerns into three layers:

- **Presentation Tier:** User Interface (HTML, JSP, Servlet, Spring MVC).
- **Business Tier:** Application logic (Java classes, Spring Core, EJB).
- **Persistence Tier:** Data handling (JDBC, JPA, Hibernate)

By avoiding business and persistence logic in Servlets, we achieve:

- Better maintainability.
- Reusability across technologies.
- Clear modular structure.
- Reduced code duplication

.



picture 1.0 (3 Tier Architecture)