2.1 Introduction to Spring Boot, MVC Architecture, Tomcat and Dispatcher Servlet

🌐 ***Client-Server Architecture*** **–**

Client-Server Architecture is a **model** where ***two*** **types of devices** interact with each other:

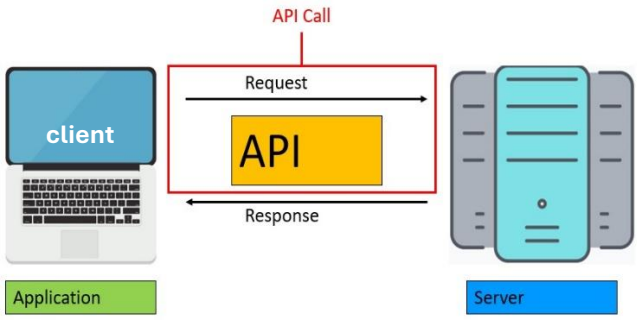
Components:

Client: The ***user-side*** (browser, app, device) that sends ***requests*** to access data or services.

Server: The ***provider-side*** that **processes the request**, runs the logic, accesses the database, and sends ***responses*** back.

How It Works:

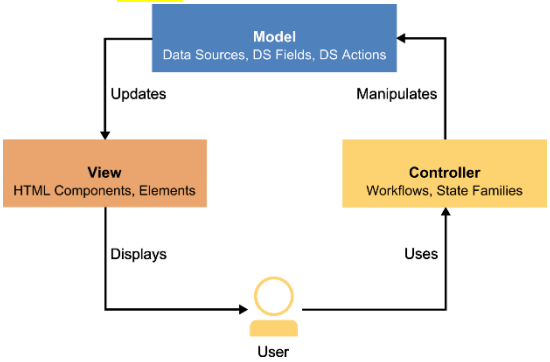
1. The ***client*** sends a ***request*** (e.g., fetch user data).
2. The ***server*** receives the **request**, performs operations (e.g., database query).
3. The ***server*** sends a ***response*** (e.g., user data in **JSON**).
4. The ***client*** displays the data or continues further action.



📡 **Role of API in Client-Server Architecture:**

API (Application Programming Interface) is the ***middleman*** that allows the ***client*** and ***server*** to ***communicate***.

What is MVC Architecture?

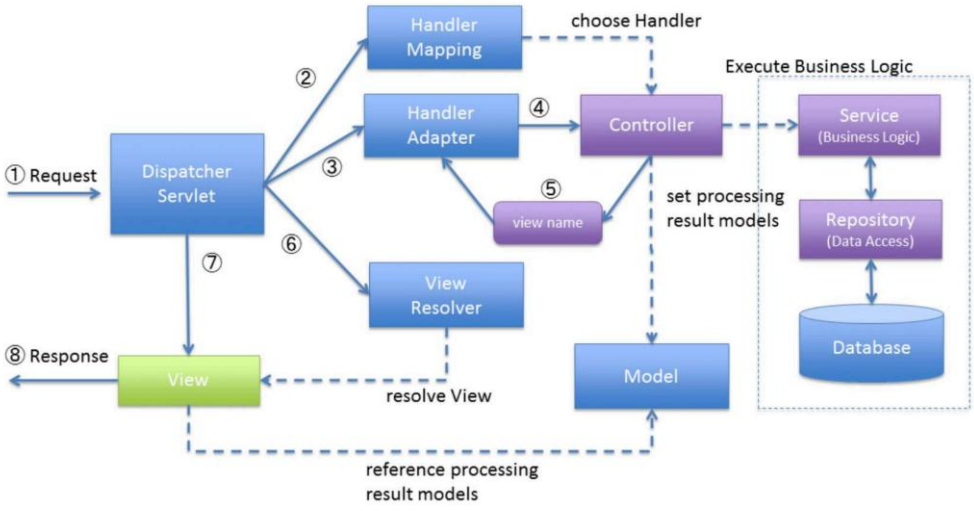


MVC stands for ***Model–View–Controller***, a ***design pattern*** used to build **organized** and **scalable applications** — especially web applications.

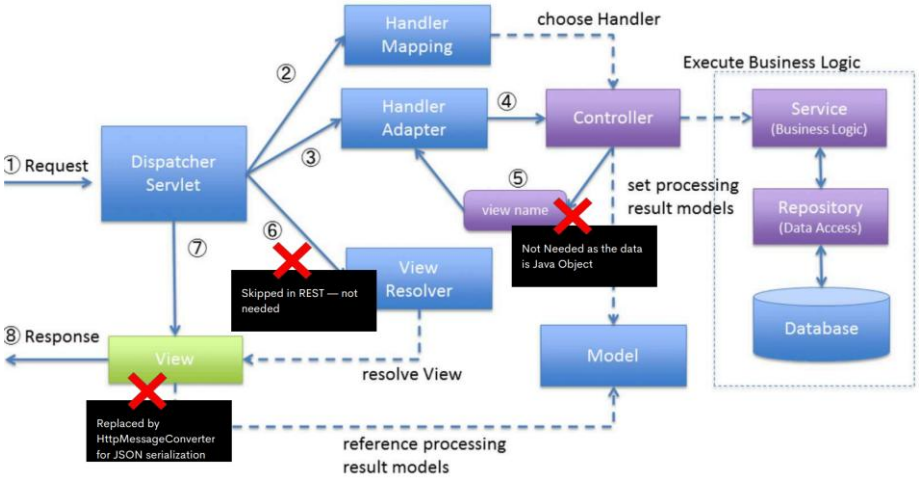
Components of MVC:

Component	Role
Model	🧠 Handles the ***data*** and ***business logic*** (e.g., database access, calculations).
View	🎯 Handles the ***UI*** – what the user sees (**HTML**, frontend templates, etc).
Controller	🎮 Handles ***user input***, **updates** the **Model**, and **decides** which **View** to show.

Spring MVC (Traditional Flow):



Spring MVC with REST APIs:



How does a Web Server works in Spring Boot?

Browser/Postman
↓
Embedded Tomcat
↓
DispatcherServlet
↓
HandlerMapping → which controller?
↓
HandlerAdapter → call controller method
↓
Controller method returns POJO/DTO
↓
HttpMessageConverter (e.g., Jackson → JSON)
↓
DispatcherServlet writes to HttpServletResponse
↓
Client gets JSON