**Framework** là một cấu trúc phần mềm được xây dựng sẵn, cung cấp một tập hợp các thư viện, công cụ, và nguyên tắc lập trình nhằm hỗ trợ nhà phát triển xây dựng và triển khai ứng dụng một cách hiệu quả và có tổ chức.



* **Thư viện mã nguồn**: Cung cấp các hàm và module giúp thực hiện nhanh các chức năng phổ biến, chẳng hạn như xử lý dữ liệu, kết nối cơ sở dữ liệu, hoặc quản lý giao tiếp mạng.
* **Kiến trúc phần mềm**: Quy định các quy tắc, mẫu thiết kế (design pattern) và cách tổ chức mã nguồn, đảm bảo tính nhất quán trong toàn bộ dự án.
* **Công cụ tích hợp**: Bao gồm trình biên dịch, công cụ kiểm thử tự động, hoặc trình quản lý gói, giúp tối ưu hóa quy trình phát triển và triển khai.





 **Spring Core** – Cung cấp **IoC (Inversion of Control) & Dependency Injection**, giúp quản lý các đối tượng Java một cách tự động.

 **Spring MVC** – Hỗ trợ xây dựng ứng dụng web theo mô hình **Model-View-Controller (MVC)**.

 **Spring Boot** – Giúp đơn giản hóa việc cấu hình và triển khai ứng dụng Spring, giảm thiểu cấu hình thủ công, có thể chạy độc lập với Tomcat, Jetty hoặc Undertow mà không cần triển khai trên một container riêng.

 **Spring Security** – Cung cấp cơ chế xác thực và phân quyền, bảo vệ API và dữ liệu.

 **Spring Data** – Hỗ trợ thao tác với cơ sở dữ liệu dễ dàng, đặc biệt với **Spring Data JPA** giúp làm việc với Hibernate nhanh chóng.

 **Spring Cloud** – Hỗ trợ xây dựng hệ thống microservices, tích hợp với Docker, Kubernetes.

A diagram of a software system

AI-generated content may be incorrect.

**Bước 1:** Yêu cầu đầu tiên sẽ được DispatcherServlet nhận.

**Bước 2:** DispatcherServlet sẽ được HandlerMapping ánh xạ yêu cầu của client đến Controller phù hợp.

**Bước 3:** Sau khi thực hiện tiến trình từ yêu cầu của client, nó thực thi các logic được xác định trong controller và cuối cùng trả về đối tượng ModelAndView. quay lại DispatcherServlet.

**Bước 4:** Bây giờ DispatcherServlet gửi Model object đến ViewResolver để có được trang xem thực tế.

**Bước 5:** Cuối cùng DispatcherServlet sẽ truyền đối tượng Model đến trang View để hiển thị kết quả.

**SpringBoot**

Cấu trúc source code của Spring Boot được dựa trên hai mô hình là mô hình MVC và mô hình 3 lớp.

## Mô hình ba lớp (three tier)

Đây là mô hình tổ chức source code rất phổ biến trong Spring Boot. Cụ thể, ứng dụng được chia làm 3 tầng (tier hoặc layer) như sau:

* **Presentation layer:** tầng này tương tác với người dùng, bằng View, Controller (trong MVC) hoặc API (nếu có).
* **Business logic layer:** Chứa toàn bộ logic của chương trình, các đa số code nằm ở đây
* **Data access layer:** Tương tác với database, trả về kết quả cho tầng business logic

Trong Spring Boot, thì có một số thành phần đại diện cho từng lớp:

* **Service:** chứa các business logic code
* **Repository:** đại diện cho tầng data access

A diagram of a structure

AI-generated content may be incorrect.

## Mô hình MVC

* **Model:** các cấu trúc dữ liệu của toàn chương trình, có thể đại diện cho trạng thái của ứng dụng
* **View**: lớp giao diện, dùng để hiển thị dữ liệu ra cho user xem và tương tác
* **Controller:** kết nối giữa Model và View, điều khiển dòng dữ liệu

Dữ liệu từ Model qua Controller sau đó được gửi cho View hiển thị ra. Và ngược lại, khi có yêu cầu mới từ View, thì sẽ qua Controller thực hiện thay đổi dữ liệu của Model.

**2. Áp dụng vào Spring Boot thì như nào?**

**2.1. Các thành phần quan trọng**

Kết hợp hai mô hình lại, chúng ta có được ứng dụng Spring Boot hoàn chỉnh, gồm các thành phần sau:

* Controller: trả về View (có chứa data sẵn, dạng trang HTML), hoặc Model thể hiện dưới dạng API cho View (View viết riêng bằng React, Vue, hoặc Angular).
* Service: chứa các code tính toán, xử lý. Khi Controller yêu cầu, thì Service tương ứng sẽ tiếp nhận và cho ra dữ liệu trả cho Controller (trả về Model). Controller sẽ gửi về View như trên.
* Repository: Service còn có thể tương tác với service khác, hoặc dùng Repository để gọi DB. Repository là thằng trực tiếp tương tác, đọc ghi dữ liệu trong DB và trả cho service.

Ơ thế còn Model và View thì đi đâu? Mình sẽ giải thích như sau:

* Model chỉ đơn giản là các đối tượng được Service tính toán xong trả về cho Controller.
* View thì có 2 loại, một là dạng truyền thống là trả về 1 cục HTML có data rồi. Lúc này Controller sẽ pass dữ liệu vào View và return về (Spring MVC có JSP hoặc template engine như Thymeleaf làm điều đó).
* View dạng 2 là dạng View tách riêng (không thuộc về project Spring boot). Thường có trong các hệ thống dùng API. View sẽ được viết riêng bằng React, Angular,... Controller sẽ đưa dữ liệu Model thông qua API cho View, và cũng nhận lại các yêu cầu qua API luôn.

A diagram of a flowchart

AI-generated content may be incorrect.