

Mô hình MVC (Model-View-Controller) là một kiến trúc phần mềm chia ứng dụng thành ba thành phần riêng biệt nhằm tách biệt giao diện, luồng xử lý và dữ liệu. Điều này giúp phân chia rõ ràng các khía cạnh khác nhau của ứng dụng (giao diện người dùng, logic xử lý, dữ liệu) và tăng tính linh hoạt cùng khả năng tái sử dụng mã . Cụ thể, MVC tách biệt input logic, business logic và UI logic một cách rạch ròi, đồng thời duy trì sự liên kết lỏng (loose coupling) giữa các thành phần .

**Vai trò của Model, View, Controller**

**Model**: Lưu trữ và xử lý dữ liệu cùng logic nghiệp vụ của ứng dụng. Model thường là các lớp POJO hoặc dịch vụ (Service) thao tác với cơ sở dữ liệu. Nói cách khác, Model đóng gói toàn bộ dữ liệu ứng dụng .

**View**: Chịu trách nhiệm hiển thị dữ liệu với người dùng. View nhận dữ liệu từ Model và sinh ra giao diện (HTML) cho trình duyệt hoặc giao diện người dùng khác. View render dữ liệu trong Model thành HTML .

**Controller**: Tiếp nhận và xử lý các yêu cầu từ người dùng. Khi có yêu cầu (request), Controller sẽ gọi Model/Service để xử lý dữ liệu, sau đó đưa kết quả vào Model và chọn một View để hiển thị. Controller xử lý yêu cầu và chuyển dữ liệu đến View để render . Như vậy Controller là cầu nối giữa Model và View.

**Spring áp dụng mô hình MVC (Spring MVC)**

Trong Spring Framework (và Spring Boot), MVC được triển khai thông qua Spring MVC. Mọi yêu cầu HTTP đầu tiên đều được DispatcherServlet – một servlet trung tâm – tiếp nhận và quản lý theo pattern “front controller” . Quy trình cơ bản trong Spring MVC như sau:

**DispatcherServlet** (Front Controller): khi có request HTTP vào ứng dụng, nó được bắt bởi DispatcherServlet .

**HandlerMapping**: DispatcherServlet dùng HandlerMapping để xác định Controller phù hợp xử lý request, dựa trên annotation (như @RequestMapping hoặc @GetMapping trên Controller) .

**Controller**: Controller nhận request, gọi các lớp Service/DAO để xử lý nghiệp vụ, thêm dữ liệu vào Model rồi trả về ModelAndView (chứa model và tên view) .

**ViewResolver**: DispatcherServlet nhờ ViewResolver tìm đúng trang giao diện (ví dụ file JSP, Thymeleaf) tương ứng với tên view do Controller trả về . Model (dữ liệu) được chuyển vào View và render thành HTML.

**Phản hồi kết quả**: Cuối cùng HTML kết quả được gửi ngược lại cho trình duyệt để hiển thị cho người dùng

**Ưu điểm của mô hình MVC trong Spring .**

**Tách biệt các tầng, dễ kiểm thử**: MVC phân chia rõ ràng ba lớp Controller, Model, View. Nhờ vậy mỗi thành phần có thể được phát triển và kiểm thử độc lập. Ví dụ trong Spring MVC, các lớp có thể được unit test riêng lẻ rất dễ dàng . View trong Spring cũng có thể tích hợp nhiều công nghệ khác nhau (JSP, Thymeleaf, Freemarker, v.v.) mà không ảnh hưởng tới logic ứng dụng .

**Hỗ trợ phát triển song song**: Các lập trình viên có thể làm việc đồng thời trên các phần khác nhau (một người xây dựng giao diện, người khác viết logic xử lý), do Model, View và Controller độc lập. Chính vì vậy, Spring MVC “cho phép phát triển nhanh và song song” nhiều thành phần trong dự án.

**Liên kết lỏng, dễ bảo trì**: Nhờ mô hình phân lớp, thay đổi một phần (ví dụ cập nhật giao diện hay nghiệp vụ) sẽ ít ảnh hưởng đến các phần khác. Tutorialspoint nhận xét rằng MVC tạo ra “sự liên kết lỏng giữa các thành phần” , và GeeksforGeeks cũng ghi nhận Spring MVC “làm cho việc cập nhật ứng dụng dễ dàng hơn” nhờ kiến trúc đa lớp rõ ràng . Điều này giúp dễ bảo trì và mở rộng hệ thống.

**Tăng tốc phát triển và gỡ lỗi**: Mô hình rõ ràng giúp gia tăng năng suất phát triển (khi chia công việc) và thuận lợi cho việc debug. Spring MVC vốn “dựa trên các lớp POJO” nên đơn giản, đồng thời hỗ trợ cấu hình bằng annotation nên việc phát triển và đọc mã nguồn nhanh chóng hơn.s