**Mô hình MVC (Model–View–Controller)** chia ứng dụng thành ba thành phần chính, mỗi thành phần đảm nhiệm một trách nhiệm riêng biệt:

1. **Model ( M );**

Đây là một bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là cầu nối giữa 2 thành phần View và Controller. Model thể hiện dưới hình thức là một cơ sở dữ liệu hoặc có khi đơn giản chỉ là một file XML bình thường. Nó hiện rõ các thao tác với cơ sở dữ liệu như cho phép xem, truy xuất, xử lý dữ liệu,…

1. **View ( V );**

Đây là phần giao diện (theme) dành cho người dùng. Nơi mà người dùng có thể lấy được các thông tin dữ liệu của MVC thông qua thao tác truy vấn như tìm kiếm hoặc thông qua các website. Thông thường, các ứng dụng web sẽ sử dụng MVC View như một phần của hệ thống, nơi các thành phần HTML được tạo ra. Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của người dùng để tương tác với phần Controller. Tuy nhiên, View lại không có mối quan hệ trực tiếp với Controller. Đồng thời, nó cũng không lấy được dữ liệu từ Controller mà chỉ có chức năng hiển thị yêu cầu và chuyển cho Controller.

1. **Controller ( C ):**

Bộ phận này có nhiệm vụ xử lý các yêu cầu của người dùng đưa đến thông qua View. Từ đó, C sẽ đưa ra những dữ liệu phù hợp nhất cho người dùng. Ngoài ra, Controller còn có chức năng kết nối với model.

**Tại sao nên dùng MVC trong dự án Java Spring**

1. **Tách biệt mối quan tâm (Separation of Concerns)**
   * Mỗi lớp (Model, View, Controller) chỉ tập trung vào một nhiệm vụ duy nhất, giúp mã nguồn gọn gàng, rõ ràng và dễ bảo trì.
2. **Hỗ trợ sẵn trong Spring Framework**
   * Spring MVC cung cấp loạt annotation (@Controller, @Service, @Repository, @RestController,…) và cơ chế dispatch servlet, giúp cấu hình và triển khai nhanh chóng.
3. **Dependency Injection và cấu hình linh hoạt**
   * Spring quản lý toàn bộ lifecycle của bean, cung cấp DI (IoC container), giảm thiểu mã khởi tạo thủ công và tăng khả năng mở rộng.
4. **Dễ dàng kiểm thử**
   * Nhờ tách biệt rõ ràng, bạn có thể mock hoặc stub từng thành phần (ví dụ: mock service khi test controller) bằng Spring Test hoặc Mockito.
5. **Tối ưu cho phát triển Web và RESTful API**
   * Spring MVC hỗ trợ sâu cho việc xây dựng API (các annotation về mapping, content negotiation, exception handling…), phù hợp với kiến trúc microservices.
6. **Tái sử dụng và mở rộng**
   * Logic nghiệp vụ trong Model (thường đánh dấu @Service hoặc @Repository) có thể dùng chung cho nhiều controller, hoặc dễ dàng thay thế implementation khác (ví dụ: chuyển từ JPA sang MyBatis).
7. **Cộng đồng và hệ sinh thái phong phú**
   * Spring là framework phổ biến nhất trong thế giới Java, có rất nhiều tài liệu, plugin và các thư viện hỗ trợ (Spring Security, Spring Data, Spring Cloud…), giúp giải quyết nhanh các bài toán thực tế.

**Kết luận:**  
Ứng dụng MVC với Spring MVC không chỉ mang lại cấu trúc rõ ràng, dễ mở rộng và kiểm thử, mà còn tận dụng được toàn bộ sức mạnh của hệ sinh thái Spring, giúp dự án Java phát triển nhanh chóng, ổn định và dễ dàng bảo trì về lâu dài.

