**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC QUY NHƠN**



**BÁO CÁO GIỮA KỲ PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG**

**Tên đề tài: FRAMEWORK SPRING MVC**

**Người hướng dẫn: *Ts. Nguyễn Thanh Bình***

**Nhóm sinh viên thực hiện: *Nhóm 8***

**Lớp: *KTPM k45***

**Thành viên:**

**Đào Văn Phận – 4551190043**

**Nguyễn Đông Dương – 4551190010 (leader)**

**Đoàn Trần Mạnh Cường – 4551190004**

**Nguyễn Văn Dũng - 4551190009**

**Bình Định, năm 2025**

## I. Tổng quan framework

### 1. Giới thiệu chung, sơ lượt lịch sử, version

Spring MVC là một module trong Spring Framework, được thiết kế để xây dựng các ứng dụng web dựa trên mô hình Model-View-Controller (MVC). Nó giúp tạo ra các ứng dụng web linh hoạt, mạnh mẽ và dễ bảo trì.

Spring Framework được tạo bởi Rod Johnson và giới thiệu lần đầu vào tháng 6 năm 2003.

Trong năm 2004, Spring Framework đã nhận được sự hưởng ứng rộng rãi nhờ việc cung cấp khung lập trình hướng khía cạnh (AOP), giúp mẫu hình lập trình này trở nên phổ biến trong cộng đồng Java.

**Các phiên bản chính:**

* **Spring 1.0 (2004):** Phiên bản đầu tiên, đặt nền móng cho các tính năng cốt lõi của framework.
* **Spring 2.0 (2006):** Bổ sung nhiều cải tiến, bao gồm hỗ trợ tốt hơn cho lập trình hướng khía cạnh và tích hợp với các framework khác.
* **Spring 3.0 (2009):** Giới thiệu hỗ trợ cho Java 5 và các cải tiến về cấu hình.
* **Spring 4.0 (2013):** Hỗ trợ Java 8 và các cải tiến về hiệu suất.
* **Spring 5.0 (2017):** Giới thiệu Spring WebFlux, hỗ trợ lập trình phản ứng (reactive programming).
* **Spring 6.0 (2022):** Cập nhật để hỗ trợ các tính năng mới của Java và cải thiện hiệu suất

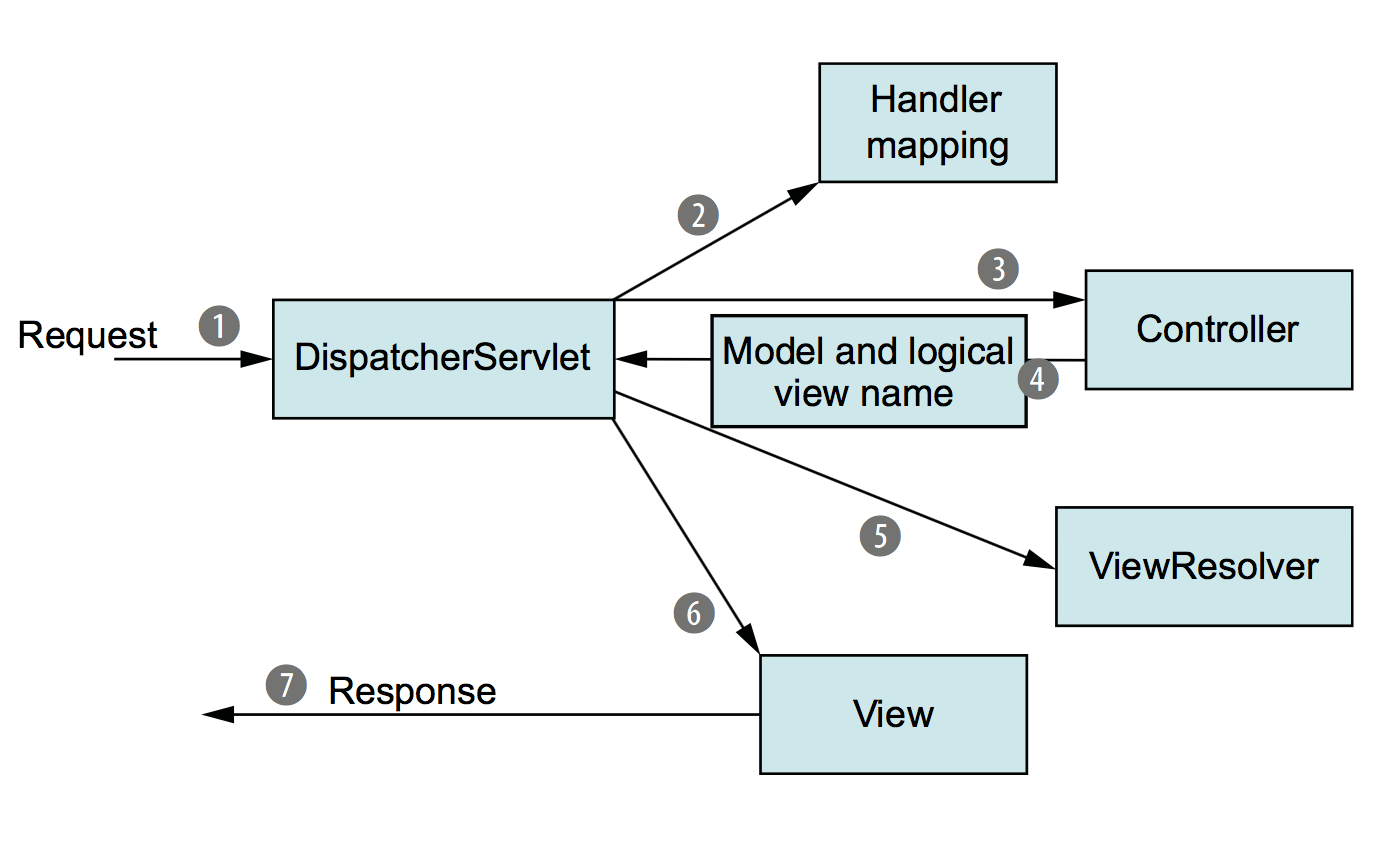
### 2.Nguyên lý và kỹ thuật phổ biến

Mô hình MVC:

* **Model:** Xử lý dữ liệu và logic nghiệp vụ.
* **View:** Hiển thị dữ liệu cho người dùng.
* **Controller:** Nhận yêu cầu, xử lý và trả về View.

Luồng xử lý:

1. Người dùng gửi yêu cầu HTTP.
2. DispatcherServlet nhận yêu cầu và xác định Controller phù hợp.
3. Controller xử lý yêu cầu, gọi Service hoặc DAO.
4. Kết quả được gửi về DispatcherServlet.
5. ViewResolver tìm và hiển thị View.
6. Kết quả được gửi về trình duyệt người dùng.



**Nguyên tắc thiết kế chính:**

* **Inversion of Control (IoC):** Giúp tách biệt việc khởi tạo và quản lý đối tượng, cho phép inject các phụ thuộc vào đối tượng thay vì để đối tượng tự tạo ra chúng.
* **Aspect-Oriented Programming (AOP):** Hỗ trợ tách biệt các mối quan tâm chéo (cross-cutting concerns) như logging, transaction management, giúp code rõ ràng và dễ bảo trì hơn.

**Kỹ thuật phổ biến**

* **Dependency Injection (DI):** Giúp quản lý các phụ thuộc giữa các thành phần trong ứng dụng một cách linh hoạt.
* **Annotation-Based Configuration:** Sử dụng các chú thích (annotation) như @Controller, @RequestMapping, @Autowired để cấu hình và quản lý các thành phần trong ứng dụng.
* **Data Binding và Validation:** Tự động chuyển đổi dữ liệu từ request thành các đối tượng Java và thực hiện kiểm tra tính hợp lệ của dữ liệu.
* **Integration với các View Technologies:** Hỗ trợ tích hợp với các công nghệ view như JSP, Thymeleaf, Freemarker để hiển thị dữ liệu.
* **Exception Handling:** Cung cấp cơ chế xử lý ngoại lệ tập trung, giúp quản lý và xử lý lỗi một cách hiệu quả.
* **Asynchronous Request Processing:** Hỗ trợ xử lý các yêu cầu bất đồng bộ, cải thiện hiệu suất và khả năng mở rộng của ứng dụng.

### 3. Thành phần chính

1. **DispatcherServlet (Front Controller):** Đây là thành phần trung tâm của Spring MVC, chịu trách nhiệm nhận mọi yêu cầu từ client và phân phối chúng đến các thành phần xử lý thích hợp như Controller.
2. **Controller:** Xử lý các yêu cầu từ DispatcherServlet, tương tác với Model để xử lý nghiệp vụ và trả về View tương ứng để hiển thị kết quả cho người dùng.
3. **Model:** Đại diện cho dữ liệu và logic nghiệp vụ của ứng dụng. Model thường bao gồm các Plain Old Java Objects (POJOs) chứa dữ liệu và các phương thức để xử lý nghiệp vụ.
4. **View:** Chịu trách nhiệm hiển thị dữ liệu từ Model cho người dùng. View thường được xây dựng bằng các công nghệ như JSP, Thymeleaf hoặc các template engine khác.
5. **HandlerMapping:** Xác định Controller nào sẽ xử lý một yêu cầu cụ thể dựa trên URL hoặc các thuộc tính khác của yêu cầu.
6. **ViewResolver:** Xác định View nào sẽ được sử dụng để hiển thị dữ liệu dựa trên tên View được trả về từ Controller.
7. **HandlerAdapter:** Giao tiếp giữa DispatcherServlet và các Controller, cho phép DispatcherServlet gọi phương thức xử lý thích hợp của Controller.
8. **HandlerInterceptor:** Cho phép can thiệp vào quá trình xử lý yêu cầu trước hoặc sau khi Controller xử lý, thường được sử dụng cho các chức năng như xác thực, ghi log hoặc quản lý phiên làm việc.
9. **LocaleResolver:** Xác định và lưu trữ thông tin về ngôn ngữ và vùng miền của người dùng, giúp ứng dụng hỗ trợ đa ngôn ngữ.
10. **MultipartResolver:** Hỗ trợ xử lý các yêu cầu có chứa tệp tin tải lên, giúp việc quản lý tệp tin trong ứng dụng trở nên dễ dàng hơn.

### 5. Điểm mạnh điểm yếu của framework, các loại ứng dụng có thể xây dựng từ framework

**Điểm mạnh:**

**+Kiến trúc linh hoạt, dễ mở rộng:** Spring MVC tuân theo mô hình Model-View-Controller (MVC) giúp tách biệt rõ ràng các thành phần Controller, Model và View, làm cho ứng dụng dễ bảo trì và mở rộng.

**+Hỗ trợ nhiều View Technology:** Spring MVC không bị ràng buộc với một công nghệ View cụ thể. Bạn có thể sử dụng JSP, Thymeleaf, FreeMarker, hoặc các công nghệ frontend khác như React, Angular, Vue.js.

**+Tích hợp tốt với các công nghệ khác:** Spring MVC có thể dễ dàng tích hợp với **Spring Boot**, **Spring Security**, **Spring Data JPA**, **Hibernate**, **Microservices**, giúp xây dựng các hệ thống từ nhỏ đến lớn.

**+Hỗ trợ lập trình hướng đối tượng và lập trình khai báo:** Spring sử dụng mạnh mẽ **Dependency Injection (DI)** và **Aspect-Oriented Programming (AOP)**, giúp tách biệt các logic xử lý và cải thiện khả năng tái sử dụng mã nguồn.

**+Hỗ trợ Restful API dễ dàng:** Với Spring MVC, bạn có thể tạo các **RESTful Web Services** một cách dễ dàng bằng cách sử dụng **@RestController**, giúp phát triển ứng dụng API nhanh chóng.

**+Bảo mật tốt:** Spring hỗ trợ **Spring Security**, giúp bảo vệ ứng dụng khỏi các lỗ hổng bảo mật phổ biến như **CSRF, XSS, SQL Injection**.

**+Cộng đồng lớn, tài liệu phong phú:** Spring MVC có một cộng đồng phát triển mạnh, tài liệu chi tiết, và nhiều nguồn tài nguyên học tập, giúp dễ dàng tiếp cận và tìm kiếm giải pháp khi gặp vấn đề.

**Điểm yếu:**

**+Độ phức tạp cao:** Cấu hình Spring MVC có thể phức tạp, đặc biệt khi sử dụng XML Configuration thay vì Java-based Configuration.

**+Yêu cầu hiểu biết sâu về Spring Core:** Để sử dụng Spring MVC hiệu quả, bạn cần nắm vững các khái niệm như **Bean, Dependency Injection, Application Context**, có thể gây khó khăn cho người mới.

**+Hiệu suất có thể bị ảnh hưởng nếu không tối ưu:** So với các framework nhẹ hơn như Express.js (Node.js), Flask (Python), Spring MVC có thể tiêu tốn tài nguyên nhiều hơn nếu không được cấu hình và tối ưu hợp lý.

**+Tích hợp với Frontend có thể mất thời gian:** Khi phát triển ứng dụng web với frontend hiện đại (React, Angular, Vue), việc kết hợp với Spring MVC có thể mất thời gian để cấu hình **CORS, RESTful API, JWT Authentication**, v.v.

**Các loại ứng dụng có thể xây dựng từ Spring MVC:**

Spring MVC có thể được sử dụng để phát triển nhiều loại ứng dụng khác nhau, từ hệ thống đơn giản đến các hệ thống lớn, bao gồm:

**1. Ứng dụng Web**

Xây dựng các website có backend mạnh mẽ với JSP, Thymeleaf, hoặc frontend hiện đại (React, Angular, Vue).

**2. Ứng dụng Quản lý**

Hệ thống quản lý doanh nghiệp như ERP, CRM, quản lý nhân sự, kho hàng, kế toán.

**3. Ứng dụng API (RESTful Web Services)**

Xây dựng API backend cho ứng dụng mobile hoặc frontend SPA (Single Page Application).

**4. Ứng dụng Microservices**

Spring MVC có thể được sử dụng để phát triển các microservices, nhưng thường Spring Boot sẽ được ưu tiên hơn.

**5. Ứng dụng WebSockets & Real-time**

Có thể xây dựng ứng dụng hỗ trợ WebSockets như chat, thông báo thời gian thực.

**6. Ứng dụng SaaS**

Spring MVC có thể được sử dụng để xây dựng các hệ thống SaaS với khả năng mở rộng cao.

**7. Ứng dụng tài chính, ngân hàng**

Spring MVC hỗ trợ bảo mật mạnh mẽ, phù hợp để xây dựng hệ thống giao dịch, quản lý tài chính.