Nguyễn Toàn Thắng 20222062  
Công nghệ Java  
DCCNTT 13.10.11

**Đặc Điểm Nổi Bật của Spring MVC:**

1. Kiến trúc Model-View-Controller (MVC) Rõ Ràng: Đây là đặc điểm cốt lõi. Spring MVC tuân thủ nghiêm ngặt nguyên tắc MVC, đảm bảo sự tách biệt rõ ràng giữa ba tầng:

Model: Tập trung vào dữ liệu và logic nghiệp vụ. Nó không biết gì về cách dữ liệu được hiển thị.

View: Chịu trách nhiệm hiển thị thông tin cho người dùng. Nó thụ động và nhận dữ liệu từ Controller để hiển thị.

Controller: Điều phối tương tác giữa Model và View. Nó nhận yêu cầu từ người dùng, gọi các phương thức trong Model để xử lý dữ liệu, và sau đó chọn View thích hợp để hiển thị kết quả.

1. DispatcherServlet là Front Controller: Đây là trái tim của Spring MVC. Tất cả các yêu cầu gửi đến ứng dụng đều đi qua DispatcherServlet. Nó hoạt động như một bộ điều phối trung tâm, chịu trách nhiệm:

Nhận yêu cầu HTTP.

Tìm Handler (Controller) phù hợp để xử lý yêu cầu.

Gửi yêu cầu đến Handler đã chọn.

Xử lý kết quả trả về từ Handler (ModelAndView).

Chọn View thích hợp để hiển thị.

Render View để tạo ra response gửi về cho client.

1. Handler Mapping Linh Hoạt: Spring MVC cung cấp nhiều chiến lược để ánh xạ các yêu cầu đến các Controller khác nhau. Các phổ biến bao gồm:

@RequestMapping: Annotation mạnh mẽ để ánh xạ các đường dẫn URL, phương thức HTTP (GET, POST, PUT, DELETE), tham số, header, và media type đến các phương thức xử lý trong Controller.

Các HandlerMapping tích hợp sẵn khác như BeanNameUrlHandlerMapping (ánh xạ dựa trên tên bean của Controller).

1. ViewResolver Đa Dạng: Spring MVC hỗ trợ nhiều công nghệ View khác nhau thông qua cơ chế ViewResolver. Điều này cho phép bạn dễ dàng tích hợp các template engine như JSP, Thymeleaf, FreeMarker, hoặc thậm chí trả về dữ liệu ở các định dạng khác như JSON hoặc XML. ViewResolver chịu trách nhiệm tìm kiếm và tạo ra đối tượng View thực tế dựa trên tên logic trả về từ Controller.
2. Data Binding và Validation Mạnh Mẽ: Spring MVC cung cấp cơ chế mạnh mẽ để tự động chuyển đổi dữ liệu từ các tham số request (form data, query parameters) thành các đối tượng Java (data binding). Nó cũng tích hợp sẵn framework Validation (ví dụ: Bean Validation API với các annotation như @NotNull, @Size, @Email) giúp dễ dàng xác thực dữ liệu đầu vào.
3. Hỗ trợ Form Handling: Spring MVC cung cấp các tag library (ví dụ: Spring Form tag library trong JSP) giúp đơn giản hóa việc tạo và xử lý các form HTML, bao gồm cả việc hiển thị lỗi validation.
4. Interceptors: Cho phép bạn can thiệp vào quá trình xử lý request trước, trong và sau khi Controller được thực thi. Interceptors hữu ích cho các tác vụ như logging, kiểm tra quyền, xử lý authentication/authorization.
5. Exception Handling: Spring MVC cung cấp các cơ chế mạnh mẽ để xử lý các exception xảy ra trong quá trình xử lý request, cho phép bạn hiển thị các trang lỗi thân thiện hoặc trả về các response lỗi có cấu trúc.
6. Hỗ trợ nhiều loại Response: Controller có thể trả về nhiều loại response khác nhau, không chỉ là trang HTML mà còn có thể là JSON, XML, file download, hoặc redirect đến một URL khác.
7. Tích hợp tốt với hệ sinh thái Spring: Là một phần của Spring Framework, Spring MVC được hưởng lợi từ tất cả các tính năng cốt lõi của Spring như Dependency Injection (DI) và Aspect-Oriented Programming (AOP), giúp xây dựng các ứng dụng linh hoạt và dễ bảo trì.

**Công Dụng Chính của Spring MVC:**

1. Xây dựng ứng dụng web có cấu trúc: Mô hình MVC giúp tổ chức code một cách logic, làm cho ứng dụng dễ hiểu, dễ quản lý và dễ mở rộng hơn so với việc viết code một cách lộn xộn.
2. Phát triển giao diện người dùng động: Spring MVC cho phép tạo ra các trang web tương tác, nơi dữ liệu được hiển thị động dựa trên yêu cầu của người dùng và trạng thái của ứng dụng.
3. Xử lý yêu cầu và phản hồi HTTP: Nó cung cấp một framework mạnh mẽ để xử lý các yêu cầu từ trình duyệt và tạo ra các phản hồi thích hợp.
4. Quản lý dữ liệu và tương tác với backend: Controller đóng vai trò trung gian trong việc lấy dữ liệu từ Model (có thể là database, API bên ngoài, hoặc các nguồn dữ liệu khác) và chuyển nó đến View để hiển thị. Nó cũng xử lý các yêu cầu cập nhật dữ liệu từ người dùng.
5. Xây dựng RESTful APIs: Spring MVC cũng là một lựa chọn tuyệt vời để xây dựng các RESTful APIs, nơi các tài nguyên được truy cập và thao tác thông qua các phương thức HTTP chuẩn (GET, POST, PUT, DELETE). Với các annotation như @RestController và @ResponseBody, việc xây dựng APIs trả về JSON hoặc XML trở nên đơn giản.
6. Đảm bảo tính bảo mật: Spring Security, một dự án khác trong hệ sinh thái Spring, thường được tích hợp với Spring MVC để cung cấp các tính năng bảo mật như authentication (xác thực người dùng) và authorization (phân quyền truy cập).
7. Kiểm thử dễ dàng: Sự tách biệt các thành phần trong mô hình MVC giúp việc viết các unit test và integration test cho từng phần (Controller, Model, View) trở nên dễ dàng hơn, đảm bảo chất lượng của ứng dụng.
8. Tóm lại, Spring MVC là một framework mạnh mẽ và linh hoạt để xây dựng các ứng dụng web hiện đại. Nó cung cấp một kiến trúc rõ ràng, nhiều tính năng hữu ích và tích hợp tốt với hệ sinh thái Spring, giúp các nhà phát triển tập trung vào logic nghiệp vụ của ứng dụng thay vì phải lo lắng về các chi tiết kỹ thuật phức tạp của việc xử lý request/response trong môi trường web.