1. Introduction to Spring MVC

**Framework là gì ?**

* Là các ứng dụng phần mềm có tính trừu tượng(abstraction) cung cấp các tính năng chung và thông dụng có thể tùy biến để tạo nền những ứng dụng cụ thể khác nhau
* Mỗi framework cung cấp một phương pháo riêng biệt để xây dựng và triển khai ứng dụng.
* Mỗi framework bao gồm một môi trường tổng thể, tái sử dụng được nhằm cung cấp các chức năng và công cụ để hỗ trợ quá trình phát triển ứng dụng.
* **Lợi ích của framework**

+ Framework giải quyết các vấn đề thông dụng, giúp lập trình viên tập trung vào xử lý nghiệp vụ

+ Giúp tăng tốc độ phát triển

+ Cung cấp môi trường làm việc tiêu chuẩn, đẽ giao tiếp giữa các bên khi cùng tham gia phát triển

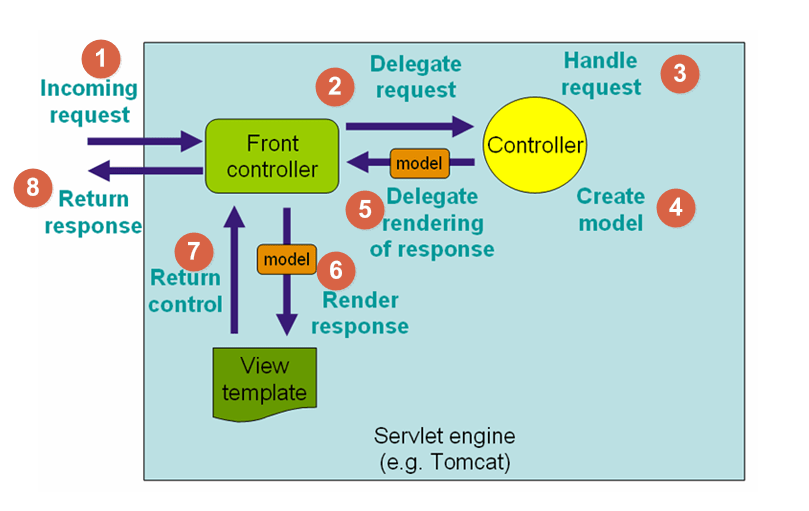
+ Các framework thường có cộng đồng phát triển lớn, các giải pháp đã được đánh giá và thử nghiệm, hệ sinh thái đầy đủ giúp nhanh chóng xây dựng được các giải pháp tùy biến

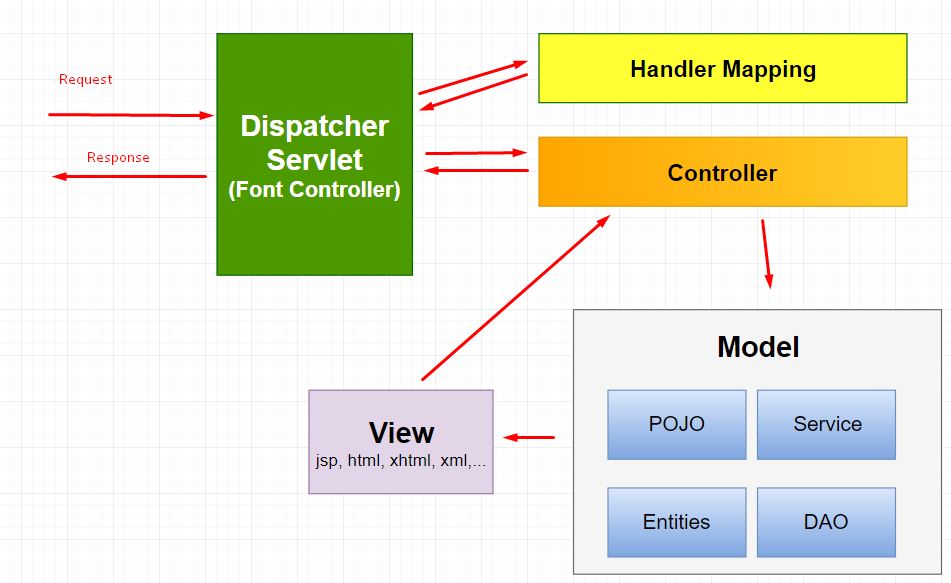
**Phân biệt framework and Library**

-

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Framework | Library |
| Cơ chế Inversion of Control | Framework nắm quyền điều khiển | Ứng dụng nằm quyền điều khiển (control) |
|  | Lớn hơn phức tạp hơn Library |  |
| Sử dụng function | Phải thay đổi cấu trúc code của dự án (project’s structure) theo các quy tắc của framework để có thể sử dụng | Sử dụng trực tiếp mà không cần thay đổi cấu trúc dự án |
|  | Là khung chương trình người dung bổ sung code và tuân theo quy tắc để tạo ra ứng dụng | Cung cấp các chức năng tiện ích hay các class để sử dụng trong quá trình xây dựng ứng dụng |
|  | Hoạt động chủ động nghĩa là có thể đưa ra các quyết định hoặc bị gọi bởi các Library hoặc ứng dụng nào đó | Hoạt động bị động nghĩa là chỉ được gọi khi chúng ta cần dùng |
|  |  |  |

Trình bày mô hình Spring MVC



* Request được gửi tới Front Controller (Dispatcher Servlet)
* Front Controller sử dụng Handler Mapping để biết được controller nào sẽ xử lý request đó
* Controller nhận request, gọi tới các class service thích hợp để xử lý yêu cầu
* Xử lý xong Controller sẽ nhân được model tử tầng Service hoặc DAO
* Controller gửi model vừa nhân được tới Front Controller
* Dispatcher Servlet sẽ tìm các mẫu view, sử dụng view resolver và truyền model vào nó
* View template, model, view page được build và gửi trả lại Front Controller
* Front Controller gửi một page view tới trình duyệt đẻ hiển thị nó cho người dung
* 
* Trong Spring MVC thì :
* M là các file POJO, Service, DAO thực hiện truy cập Database, xử lý business
* V là các file JSP, html, …
* C là Dispatcher Controller, Handler Mapping, Controller – thực hiện điều hướng các request.

IOC là gì ? Các cách triển khai DI

* Inversion Of Control Là một nguyên lý thiết kế trong công nghệ phần mềm, trong đó các thành phần nó dựa vào để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hương thủ tục truyền thống, nói ngắn gọn thì IOC là nguyên lý thiết kế được tạo ra để các code tuân thủ theo nguyên lý Dependency Inversion.Ví dụ các mô hình : Service Locator, Events, DI(Dependency Injection)
* Spring IOC :
* IOC Container chính là thành phần thực hiện IoC.
* Công việc của spring container là tạo các đối tượng rồi lắp ráp chúng lại với nhau cấu hình các dối tượng và quản lý vòng đời của chúng từ lúc được tạo ra cho đến khi bị hủy.
* Spring container sử dụng DI để quản lý thành phần, đối tượng đẻ tạo nên 1 ứng dụng, Các thành phần và đối tượng này gọi là Spring Bean
* IoC Container trong Spring có 2 kiểu là : BeanFactory và ApplicationContext.
* DI là gì?
* Là 1 trong những design pattern được sử dụng để triển khai IoC
* Nguyên tắc cơ bản của DI là làm cho module cấp cao phụ thuộc vào module cấp thấp thông qua injector

1. Spring Controller

**Controller là gì ?**

* @Controller dùng để đánh dấu 1 class là spring MVC controller ( dùng web application)
* @RestController dung để đánh dấu 1 class là restfull controller ( dung cho web service)

**Cách khai báo bean:**

* Dùng annotation @Controller, @Service, @Repository, @Component
* Dùng kết hợp @Configuration + @Bean
* Khai báo bằng file .xml :

applicationContext.xml

dispatcher-servlet.xml

web.xml

**@RequestMapping và các thuộc tính**

* @RequestMapping dung để ánh xạ các request tới xử lý ở controller tương ứng
* Các thuộc tính:

1. Value ( url pattern)
2. Method (GET,POST,PUT,PATCH,DELETE)
3. Produces : định nghĩa kiểu dữ liệu trả về cho client

Tương ứng với cài đặt accept của request header

1. Consumes : định dạng kiêu dữ liệu từ client gửi lên server

Cài đặt content-type của request header

**@RequestParam**

* Sử dụng để lấy giá trị theo dạng key/value ( param động)
* Các thuộc tính :

1. Value: tên của parameter (key)
2. Required (true/false): param có bắt buộc hay không (giá trị mặc định là true)
3. defaultValue: gán giá trị mặc định cho param.

**@PatchVariable**

* Sử dụng để lấy giá trị tham số tĩnh trên url ( chỉ lấy value)
* Có thể sử dụng Regex.

**Handler Method**

* là những method dung để xử lý request của người dung và trả về kết quả.

**Các biến thể của @RequestMapping**

* @GetMapping
* @PostMapping
* @DeleteMapping
* @PutMapping
* @PatchMapping

1. Views Thymeleaf

**Thymeleaf là gì ?**

* Là bộ xử lý views được sử dụng cho các ứng dụng wed và các ứng dụng độc lập
* Có thể xử lý HTML, XML, JavaScript, CSS và cả text
* Cung cấp một cách thức xây dựng template hiện đại và dễ bảo trì
* Được xây dựng phù hợp với các tiêu chuẩn của wed, đặc biệt là HTML5
* Thymeleaf cho phép xử lý 6 loại template, mỗi loại được gọi là thymeleaf mode

**Các loại biểu thức trong thymeleaf:**

* Biểu thức với biến: ${…}.
* Biểu thức với thuộc tính : \*{…}.
* Biểu thức với message : #{…}.
* Biểu thức với URL: @{…}.
* Biểu thức với phân đoạn: ~{…}.

**Các hằng giá trị và các toán tử:**

* Hằng giá trị:

1. Hằng văn bản ‘…’
2. Hằng số: 0, 34, 3.0, 12.3, …
3. Hằng giá trị logic: true, false
4. Hằng null: null
5. Hằng token: one, sometext, main, …

* Thao tác với chuỗi:

1. Toán tử cộng chuỗi: +
2. Thay thế giá trị của biến: |The name is ${name}|

* Toán tử toán học: +, -, \*, /, %
* Toán tử một ngôi(lấy số đối) : -
* Toán tử logic : and, or, !, not
* Toán tử so sánh: >, <, >=, <=, ==, !=.
* Toán tử điều kiện: (if) ? (then), (if) ? (then) : (else), (value) ?: (defaultvalue)

**Fragment**

* Là các đoạn mã riêng biệt có thể được nhúng vào các vị trí khác nhau trong template
* Có thể sử dụng **th: insert** hoặc **th: replace** để nhúng **fragment**

Vòng lặp trong thymeleaf

* <tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}? 'odd'">

...

</tr>

* Các trạng thái trong vòng lặp th: each

1. Index: chỉ số của lần lặp hiện tại, bắt đầu từ 0
2. Count: chỉ số của lần lặp hiện tại, bắt đầu từ 1
3. Size: tổng số các phần tử
4. Current: phần tử của vòng lặp hiện tại
5. Even/odd: vòng lặp chẵn/lẻ
6. First: vòng lặp đầu tiên
7. Last: vòng lặp cuối cùng
8. Data Binding Form

**Data Binding là gì ?**

* Là cơ chế liên kết dữ liệu đầu vào hoặc đầu ra với các đối tượng trong model
* Giúp cho việc tương tác với dữ liệu trở nên dễ dàng
* Khi sử dụng data binding các form đều được liên kết với một đối tượng biểu diễn dữ liệu ở phía sau
* Khi tương tác với form, dữ liệu trên form sẽ được tự động chuyển đổi thành các thuộc tính của đối tượng liên kết với nó
* Data Binding hỗ trợ chuyển đổi dữ liệu (data conversion) và validate dữ liệu
* Data Binding trong Spring hoạt động dựa trên Data Binder

**Thẻ form và các thẻ thao tác với form của thư viện spring-form.tld.**

* Được sử dụng để tạo một form
* Ngoài các thuộc tính của thẻ form trong html còn có thêm các thuộc tính quan trọng

commandName(modelAttribute) : tên của model được liên kết với form.

* Các thẻ của spring-form.tld:

Button, checkbox, checkboxes, errors, form, hidden, input, label, option, options, password, radiobutton, radiobuttons, select, texarea.

* Các thuộc tính quan trọng:

Path: tên của thuộc tính liên kết với trường hiện tại

Items: danh sách các hạng mục của các trường như select, options, radiobuttons, checkboxes.

1. Database ORM

**ORM**

* Là ky thuật liên kết giữa các đối tượng trong lập trình với các dối tượng trong database
* Là quá trình biến đổi qua lại giữa 1 record trong database và 1 object trong OOP
* ORM có nhiều implement

1. Hibernate là ORM Framework nổi tiếng nhất và sử dụng rộng rãi trong java

**Thẻ form và các thẻ thao tác với form của thư viện spring-form.tld.**

* Được sử dụng để tạo một form

**Entity.**

* Là đối tượng đại diện cho dữ liệu trong ứng dụng
* Entity thường là POJO(Plain Old Java Object)- Chỉ mô tả dữ liệu, không xử lý nghiệp vụ
* Mỗi entity sẽ mapping với 1 table trong DB
* Được khai báo bằng annotation @Entity
* Có constructor không tham số có AM là public
* Không được khai báo final
* Các thuộc tính có AM là private, protected hoặc default

1. JPA
2. Spring Data Repository

Spring Data JPA

* Là 1 module của Spring framework, mục đich là giảm thiểu việc thực hiện quá nhiều bước để có thể implement được JPA. Spring Data JPA là một phần của Spirng Data và nó hỗ trợ Hibernate 5, OpenJPA 2.4 và EclipseLink 2.6.1.

Các interface trong Spring Data Repository

* Repository<T, ID>: Là interface chính của Spring Data JPA, chỉ dung để đánh dấu/phân loại mà không khai báo các phương thức
* CrudRepository<T,ID>: Hỗ trợ thực hiện các câu lênh CRUD cơ bản
* PagingAndSortingRepository<T, ID>: Kế thừa từ CrudRepository, hỗ trợ thêm khả năng phân trang và sắp xếp
* ReactiveCrudRepository<T, ID>
* ReactiveSortingRepository<T, ID>
* RevisionRepository<T, ID>
* RxJava2CrudRepository< T, ID>
* RxJava2SortingRepository<T, ID>

T: là kiểu dữ liệu của entity muốn quản lý

ID : là kiểu dữ liệu của id của entity muốn quản lý

Converter và Formatter

* Hỗ trợ chuyển đổi dữ liệu nhập vào sang kiểu dữ liệu thích hợp
* Converter là các thành phần sử dụng chung cho toàn bộ hệ thống, có thể sử dụng converter ở bất kì tầng nào của ứng dụng
* Formatter chỉ được thiết kế để sử dụng ở tầng web

Phương thức addResourceHandlers()

* Giúp đăng kí các thư mục chứa tài nguyên tĩnh như file ảnh, video, css, JS,…
* Các tài nguyên tĩnh có thể được đặt trong thư mục của ứng dụng, trong classpath hoặc các vị trí khác.

1. Validation
2. AOP Exception Handle
3. Session and Cookie
4. Web service and RESTful service
5. AJAX
6. I18N
7. Spring Security
8. Automated Testing
9. Spring Boot

Spring Boot :

* Là giải pháp cho việc chuyển đổi cấu hình một project độc lập của Spring
* Giúp cho việc bắt đầu một dự án Spring trở nên dễ dàng và nhanh chóng
* Được thể hiện thông qua một loạt các quy tắc nhằm mục đích :

1. Cung cấp các ứng dụng được cấu hình và đóng gói đầy đủ có thể chạy độc lập
2. Giảm thiểu tối đa sự phức tạp của việc cấu hình project cũng như quản lý các thư viện được cung cấp từ bên thứ 3 ngoài Spring

* Spring Boot đưa ra 4 đặc điểm để phục vụ tối đa cho mục đích làm đơn và dễ dàng hơn trong việc phát triển dự án Spring
* Spring Boot là một công nghệ mang lại nhiều lợi ích, giúp cho quá trình phát triển một dự án Spring nhanh chóng và dễ dàng hơn rất nhiều.
* • Dự án Spring Boot khởi chạy từ hàm Main[] nằm trong class được xác định qua annotation @SpringBootApplication.
* • Một ứng dụng Spring Boot có thể khởi chạy độc lập

1. **Spring Boot Starters:**

* Cung cấp 1 single dependency chứa những dependencies có chung đặc điểm
* Rút ngắn danh sách dependency cần khai báo cho 1 dự án Spring
* Giảm bớt công việc phải làm trong quá trình khai báo và quản lý các dependencies
* Đảm bảo tất cả dependencies hoạt động ổn định cùng nhau
* Tạo sự rõ rang dễ đoán được tác dụng từ việc đặt tên các dependencies

1. **Auto-configuration**

* Đặc điểm auto-configuration của Spring Boot sẽ tự động cấu hình dự án dựa trên các dependencies đã được thêm vào Gradle/Maven.
* Khởi động auto-configuration thông qua antonation @EnableAutoconfiguration hoặc @SpringBootApplication
* Ưu điểm:

• Giảm thiểu rất nhiều khối lượng đội tượng cần cấu hình khi bắt đầu một dự án Spring.

• Làm đơn giản cấu trúc một dự án Spring giúp người phát triển tập trung nhiều hơn vào phát triển logic nghiệp vụ.

• Giúp việc chuyển đổi giữa các công nghệ, thư viện dễ dàng hơn

1. Command-line interface (CLI)

• SpringBoot-CLI là một giao diện người dùng thao tác qua dòng lệnh hỗ trợ cậu lệnh và cú pháp SpringBoot.

• SpringBoot-CLI tận dụng sự mạnh mẽ của Spring-boot-starter và Auto-configuration để chạy được nhưng dòng lệnh nhập vào mà không cần khái báo dependencies, configuration…

1. Actuator

SpringBoot Actuator là một project con nhúng bên trong SpringBoot project.

• SpringBoot Actuator cũng cấp một số dịch vụ giúp theo dõi, kiểm soát hoạt động của SpringBoot Application giống như những publish web services.

• Quản lý các điểm kết thúc (endpoints):

• GET /autoconfig: Thông tin về auto-configuaration của SpringBoot

• GET /beans: Danh sách @Bean đang cấu hình trong ứng dụng.

• GET /evn: Lấy danh sách các môi trường và hệ điều hành đang chạy ứng dụng.

• GET /health: Lấy tình trạng ứng dụng gồm đang hoạt động, đã dừng, dung lượng ổ cứng total, free.

• POST /shutdown: Cho phép tắt ứng dụng

• Gửi thông tin lỗi của ứng dụng.

• Gửi thông tin tổng quan về quá trình chạy của ứng dụng.

• Đưa ra framework hỗ trợ kiểm định, kiểm soát các events sinh ra trong quá trình chạy của ứng dụng.

• Trong đó tính năng quản lý các điểm kết thúc là quan trọng và hữu ích, hay được sử dụng nhất trong một dự án SpringBoot.

1. Deployment