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

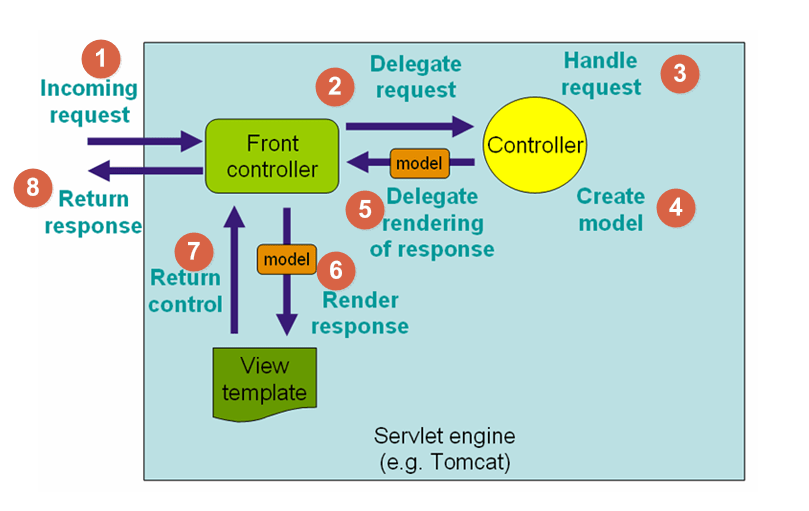
+ Các framework thường có cộng đồng phát triển lớn, các giải pháp đã được đánh giá và thử nghiệm, hệ sinh thái đầy đủ giúp nhanh chóng xây dựng được các giải pháp tùy biến

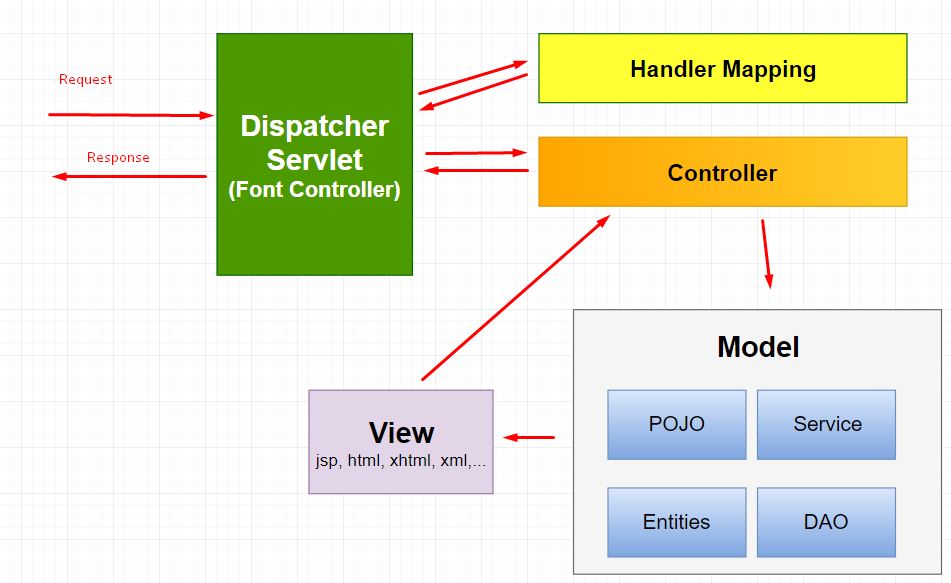
**Phân biệt framework and Library**

-

|  |  |  |
| --- | --- | --- |
| 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
2. JPA
3. Spring Data Repository
4. Validation
5. AOP Exception Handle
6. Session and Cookie
7. Web service and RESTful service
8. AJAX
9. I18N
10. Spring Security
11. Automated Testing
12. Spring Boot
13. Deployment