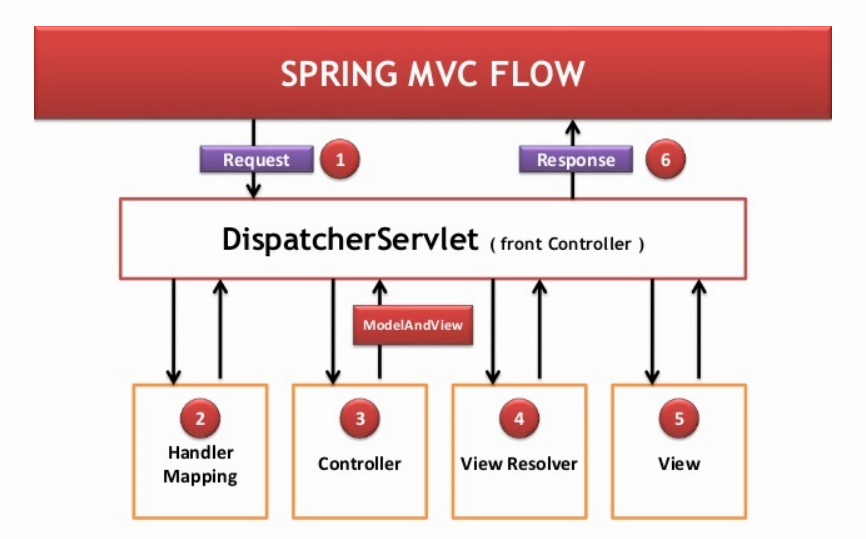
1. **Vẽ và trình bày cách Spring MVC xử lý request**



**Khi có một yêu cầu URL gữi đến các bước thực hiện sẽ như sau :**

1. Spring Font Controller (DispatcherServlet) nhận yêu cầu nó sẽ tìm đến Handler Mapping thích hợp.
2. Handler Mapping sẽ ánh xạ yêu cầu của client đến controller thích hợp.
3. Sau đó DispatcherServlet sẽ gửi yêu cầu đến Controller thích hợp.
4. Sau khi thực hiện tiến trình từ yêu cầu của client, nó thực thi các logic được xác định trong controller và cuối cùng trả về đối tượng ModelAndView.
5. Dựa trên các giá trị trong ModelAndView Controller mà nó sẽ view, có thể là JSP, FreeMaker,tiles do file cấu hình của chúng ta config.
6. Controller sẽ gửi đến cho Servlet container và cuối cùng Servlet container sẽ gửi kết quả cho người dùng
7. **Framework là gì ? Framework khác Library chỗ nào ?**

* Framework được xem như là bộ khung sẵn có của chương trình, chứa tập hợp thư viện phần mềm, API... hỗ trợ cho việc phát triển sao cho tiết kiệm thời gian và hợp lý nhất.
* Library là một tập hợp chức năng (function), lớp (class)... được cung cấp sẵn và có thể tái sử dụng.

\* Dưới đây là một số điểm khác nhau giữa Framework và Library:

* Framework có thể chứa nhiều Library bên trong nó.
* Library là tập hợp con nằm trong Framework.
* Khi sử dụng Framework, bạn cần thay đổi cấu trúc code trong dự án nhằm đáp ứng đúng theo nguyên tắc đề ra, có như vậy mới sử dụng được các chức năng của nó vào chương trình.
* Khi sử dụng Library, người dùng có thể dễ dàng áp dụng các chức năng vào chương trình mà không cần phải thay đổi cấu trúc code.
* Framework là khung chương trình, cho phép người dùng bổ sung các đoạn mã code, nhưng vẫn phải tuân thủ theo nguyên tắc mà nó đề ra.
* Library đơn giản chỉ là nơi cung cấp những tính năng tiện ích và class được sử dụng trong quá trình xây dựng, cũng như phát triển ứng dụng.
* Mọi hoạt động trong Framework đều được diễn ra một cách chủ động.
* Mọi hoạt động trong Library đều được diễn ra một cách bị động, khi ta cần và gọi đến chúng.

1. **@Autowired là gì ?**

Tính năng Autowiring của Spring framework cho phép bạn đưa sự phụ thuộc vào đối tượng một cách ngầm định. Bên trong nó sử dụng setter hoặc constructor tiêm.

Autowiring không thể được sử dụng để chèn các giá trị nguyên thủy và chuỗi. Nó chỉ hoạt động với tài liệu tham khảo.

1. **@Component có ý nghĩa gì ?**

Chúng ta có thể sử dụng @Component trên toàn ứng dụng để đánh dấu các bean là thành phần được quản lý của Spring . Spring chỉ chọn và đăng ký bean với  *@Component* và không tìm kiếm *@Service*  và  *@Repository* nói chung.

Chúng được đăng ký trong *ApplicationContext* vì bản thân chúng được chú thích bằng  *@Component.*

1. **Trình bày ý nghĩa của Controller**

Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view. Từ đó, Controller đưa ra dữ liệu phù hợp với người dùng. Bên cạnh đó, Controller còn có chức năng kết nối với model.

1. **Trình bày ý nghĩa của ModelAndView Interface ?**

Interface Model để truyền dữ liệu từ Controller sang View để hiển thị .

Giao diện cuối cùng để chuyển các giá trị cho một chế độ xem là ModelAndView .

Giao diện này cho phép chúng tôi chuyển tất cả thông tin được yêu cầu bởi Spring MVC trong một lần trả lại.

1. **Định nghĩa URI với các phương thức khác nhau như GET, POST, PUT, PATH, DELETE**

GET: được sử dụng để lấy thông tin từ sever theo URI đã cung cấp.  
HEAD: giống với GET nhưng response trả về không có body, chỉ có header  
POST: gửi thông tin tới sever thông qua các biểu mẫu http( đăng kí chả hạn..)  
PUT: ghi đè tất cả thông tin của đối tượng với những gì được gửi lên  
PATCH: ghi đè các thông tin được thay đổi của đối tượng.  
DELETE: xóa tài nguyên trên server.  
CONNECT: thiết lập một kết nối tới server theo URI.  
OPTIONS: mô tả các tùy chọn giao tiếp cho resource.  
TRACE: thực hiện một bài test loop - back theo đường dẫn đến resource.

**GET VS POST**

2 loại request get post gần như đều rất quen thuộc với chúng ta. với một ứng dụng web được thiết kế theo restful thì get chỉ dùng để lấy dữ liệu và post chỉ dùng để đẩy dữ liệu lên. Nhưng, khi đi phỏng vấn thì chúng ta sẽ thường được hỏi: "Liệu get có thể được dùng để đấy dữ liệu lên hay post có thể được dùng để lấy dữ liệu về hay không ?" thú thật với các bạn là trước đây tôi đã trả lời là không thể :man\_vampire: Hoàn toàn có thể các bạn nhé. Nhưng có chúng ta không nên sử dụng như vậy, nó phá vỡ các quy tắc thiết kế. Một chút khác biệt dễ nhận thấy giữa get và post là get thì không có body. Khi dùng get để truyền dữ liệu lên sever chúng ta thấy rằng tất cả các paramater đều bị hiển thị trên url của request, xét về khía cạnh bảo mật thì điều này thật là tệ.  
Post thì khác, nó giấu parameters trong body và mã hóa chúng đi, ngăn cản các phần tử trung gian ăn cắp nội dung. Nhưng post chỉ có tính an toàn đối với client, còn với sever thì lại khác. Các method như post, put, delete bị coi là unsafe và not idempotent cho server.

**POST/PUT/PATCH**

Điểm khác biệt giữ post và put đơn giản là put là idempotent còn post thì không, bạn sẽ nhận được thông báo lỗi khi gửi một request post với cùng 1 nội dung 2 lần nhưng put thì không, nó luôn trả về kết quả như nhau.  
post: tạo mới  
put: ghi đè(toàn bộ) hoặc tạo mới 1 resource  
patch: cập một 1 phần của resource

**SAFE**

một method được coi là safe khi nó không làm thay đổi trạng thái "sate" của server. Nói cách khác, an toàn là chỉ đọc mà không làm thay đổi bất kì điều gì. Các method được coi là safe chỉ có: GET, HEAD và OPTIONS.  
Unsafe: PUT, DELETE, POST và PATCH.

**IDEMPOTENT**

các method được coi là idempotent khi nó có thể thực hiên n + 1 lần mà vẫn trả lại 1 kết quả như ban đầu.

vì điều này nên các method safe thì đều idempotent. Nhưng unsafe chưa chắc đã idempotent.

1 số lưu ý: header dài tối đa 8kb và cũng phụ thuộc cả vào trình duyệt  
body thì limit của nó tùy trình duyệt.  
url không dài quá 2 nghìn kí tự (ror)

1. **Template là gì ? Template Engine là gì ? Thymeleaf là gì ?**

* Template là các mẫu layout đã được dàn trang, có bố cục được thiết kế sẵn theo các màu sắc, bố cục ngẫu nhiên bất kỳ. Khi có nhu cầu sử dụng, người dùng sẽ tải các mẫu template này, thêm vào các nội dung, thông tin hay chỉnh sửa lại về mặt đồ họa theo mục đích của mình, vậy là họ đã có một sản phẩm mang dấu ấn riêng của mình.
* **Template Engines** là công cụ giúp chúng ta tách mã HTML thành các phần nhỏ hơn mà chúng ta có thể sử dụng lại trên nhiều tập tin HTML.
* Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text.
* Mục tiêu chính của thymeleaf là mang lại các template tự nhiên, đồng nhất, đơn giản (nature templates) cho công việc phát triển.

1. **Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?**

**Thymeleaf** cung cấp cho bạn vòng lặp **'each'**, và bạn có thể sử dụng nó thông qua thuộc tính (attribue) **th:each**. Đây là vòng lặp duy nhất được hỗ trợ trong **Thymeleaf**.

Trong một vài tình huống bạn muốn một đoạn nào đó của **Thymeleaf Template** xuất hiện trong kết quả nếu một điều kiện nhất định nào đó được đánh giá là **true** (Đúng). Để làm được điều này bạn có thể sử dụng thuộc tính (attribute) **th:if**.

Một thuộc tính (attribute) khác bạn cũng có thể sử dụng đó là **th:unless**, nó là phủ nhận của **th:if**.

1. **Data Binding là gì?**

Data binding là một thư viện hỗ trợ cho phép bạn liên kết các UI component (thành phần giao diện người dùng) trong layout với các nguồn dữ liệu trong ứng dụng của bạn bằng cách sử dụng định dạng khai báo thay vì lập trình.

1. **ORM là gì ?**

ORM (**O**bject **R**elational **M**apping), là một kỹ thuật/cơ chế lập trình thực hiện ánh xạ CSDL sang các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java, C# …(các table tương ứng các class, mối ràng buộc giữa các table tương ứng quan hệ giữa các class ‘has a’ , ‘is a’).

# **Ưu nhược điểm của ORM**

## **Ưu điểm**

* OOP: ORM giúp lập trình viên tập trung vào lập trình hướng đối tượng
* Tính độc lập: Làm việc được với nhiều loại database(hệ quản trị cơ sở dữ liệu), nhiều kiểu dữ liệu khác nhau. Dễ dàng thay đổi loại database hơn. Các câu lệnh SQL không phụ thuộc vào loại database.
* Đơn giản, dễ sử dụng: Hỗ trợ HSQL, cung cấp nhiều nhiều API truy vấn.
* Năng suất hơn: viết code ít hơn, dễ hiểu hơn. Phù hợp các case CRUD (Create, Read, Update, Delete)
* Khả năng sử dụng lại code.

## **Nhược điểm**

* Khả năng truy vấn bị hạn chế, nhiều trường hợp ta vẫn phải dùng native SQL để truy vấn database.
* Khó tối ưu câu lệnh SQL (do câu lệnh SQL được ORM tự động sinh ra).

1. **JPA là gì ?**

JPA (Java Persistence API) là 1 giao diện lập trình ứng dụng Java, nó mô tả cách quản lý các mối quan hệ dữ liệu  trong ứng dụng sử dụng **Java Platform**.

**Java Platform** (**nền tảng Java**) là một tập hợp các chương trình giúp phát triển và chạy các chương trình được viết bằng ngôn ngữ lập trình Java.

**API** là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của **Application Programming Interface** – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng.

1. **@RequestMapping làm gì ?**

**@RequestMapping**  là một trong những annoation sử dụng nhiều nhất trong Spring MVC.

Annotation @RequestMapping được sử dụng để map request với class hoặc method xử lý request đó.

@RequestMapping có thể được áp dụng với controller class hoặc method trong controller class.

1. **Hibernate là gì ?**

Hibernate là một thư viện ORM (Object Relational Mapping) mã nguồn mở giúp lập trình viên viết ứng dụng Java có thể map các objects (pojo) với hệ quản trị cơ sở dữ liệu quan hệ, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cớ dữ liệu quan hệ.

**Persistence object** Chính là các POJO object map với các table tương ứng của cơ sở dữ liệu quan hệ. Nó như là những “thùng xe” chứa dữ liệu từ ứng dụng để ghi xuống database, hay chứa dữ liệu tải lên ứng dụng từ database.

**Session Factory** Là một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong Hibernate configuration. Mỗi một database phải có một session factory.

Tỉ dụ nếu ta sử dụng MySQL, và Oracle cho ứng dụng Java của mình thì ta cần có một session factory cho MySQL, và một session factory cho Oracle.

**Hibernate Session** Mỗi một đối tượng session được Session factory tạo ra sẽ tạo một kết nối đến database.

**Transation** Là transaction đảm bảo tính toàn vẹn của phiên làm việc với cớ sở dữ liệu. Tức là nếu có một lỗi xảy ra trong transaction thì tất cả các tác vụ thực hiện sẽ thất bại.

**Query**  
Hibernate cung cấp các câu chuy vấn HQL (Hibernate Query Language) tới database và map kết quả trả về với đối tượng tương ứng của ứng dụng Java.

1. **Entity là gì ?**

## @Entity

@Entity được sử dụng để chú thích một class là một Entity.

Thuộc tính name của @Entity là không bắt buộc. Việc chỉ định rõ name của @Entity cho phép viết ngắn câu HSQL.

Entity khớp với một bảng lấy theo tên theo thứ tự ưu tiên:

1. name trong @Table.
2. name trong @Entity.
3. name của class.
4. **Các câu hỏi hay về hibernate**

https://vn.bitdegree.org/huong-dan/hibernate-la-gi/