Còn từ bài này, chúng ta sẽ học về các mảng chính mà người ta áp dụng **Spring Boot** vào, đó là **Lập trình Web** hoặc **Restful Web Service**.

Để bắt đầu, chúng ta sẽ bắt đầu với @Controller

**Cài đặt**

Trong bài này, tôi có sử dụng thêm thư viện spring-boot-starter-thymeleaf. Đây là một **Template Engine** hỗ trợ chúng ta tạo ra các file html để trả về thông tin cho người dùng. Về cơ bản là như vậy, còn chi tiết tôi sẽ giới thiệu riêng ở bài sau.

Tạm thời hãy cứ để nó là một thư viện trong chương trình của bạn để chúng ta có thể chạy suôn sẻ.

*pom.xml*

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>me.loda.spring</groupId>

<artifactId>spring-boot-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-learning</name>

<description>Everything about Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--spring mvc, rest-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- thymeleaf -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

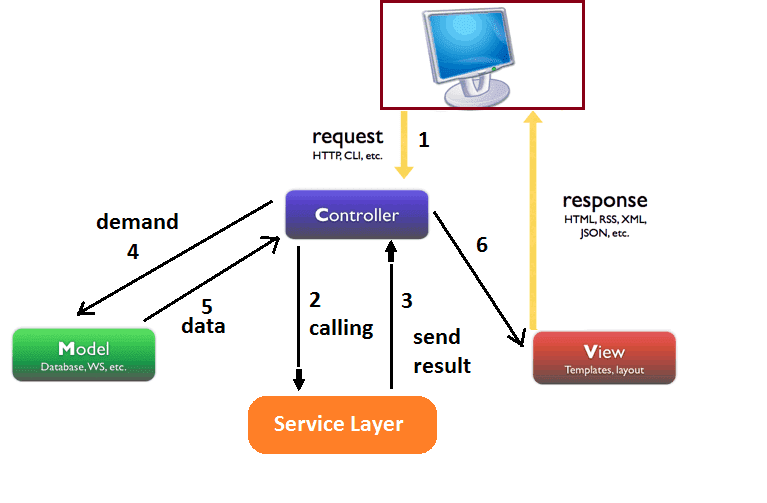
Cấu trúc thư mục:



**@Controller**

[Trong bài viết số 4](https://loda.me/spring-boot-4-component-vs-service-vs-repository-loda1557627097246), khi nói về @Service và @Repository tôi đã đề cập tới kiến trúc trong **Spring Boot**.

Để xây dựng một trang web với **Spring Boot**, bạn sẽ cần tuân theo quy trình như hình dưới đây:



@Controller là nơi tiếp nhận các thông tin request từ phía người dùng. Nó có nhiệm vụ đón nhận các yêu cầu (kèm theo thông tin request) và chuyển các yêu cầu này xuống cho tầng @Serivce xử lý logic.

**HTML**

Để tạo ra một trang Web, bạn sẽ cần tạo ra các trang html để trả về cho người dùng.

Mặc định trong **Spring Boot**, các file html này sẽ được lưu trữ trong thư mục resources/templates như sau:



**Spring Boot + Thymeleaf** sẽ tìm kiếm các file này theo tên.

ví dụ "index" sẽ tương ứng với "index.html".

**Hello World**

Chúng ta sẽ tạo ra một Server Web đơn giản để bạn dễ dàng hiểu @Controller làm việc như nào.

*WebController.java*

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

// Đánh dấu đây là một Controller

// Nơi tiếp nhận các reqquest từ phía người dùng

@Controller

public class WebController {

// Đón nhận request GET

@GetMapping("/") // Nếu người dùng request tới địa chỉ "/"

public String index() {

return "index"; // Trả về file index.html

}

}

*index.html*

<!DOCTYPE html>

<html>

<head>

<title>Hello World</title>

</head>

<body>

<h1>Đây là một trang web</h1>

</body>

</html>

Chạy thử chương trình:

*App.java*

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class App {

public static void main(String[] args) {

SpringApplication.run(App.class, args);

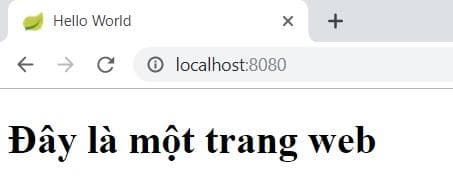
}

}

Lúc này Server đã ON trên port 8080.

Bạn hãy vào trình duyệt theo đường dẫn <http://localhost:8080/>

Kết quả sẽ như sau:



**Giải thích 1**

Bản thân @Controller Cũng là một @Component nên nó sẽ được **Spring Boot** quản lý.

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Component

public @interface Controller {

@AliasFor(

annotation = Component.class

)

String value() default "";

}

**Spring Boot** sẽ lắng nghe các request từ phía người dùng. và tùy theo đường dẫn path là gì, nó sẽ mapping tới hàm xử lý tương ứng trong @Controller.

Như ví dụ trên, tôi sử dụng GET vào địa chỉ localhost:8080/ ( đường dẫn là /). **Spring Boot** sẽ gọi tới hàm có gắn @GetMapping("/") và yêu cầu hàm này xử lý request này.

Trong ví dụ trên, tôi chỉ trả về một file index.html là xong, không cần nghiệp vụ gì khác.

**Ví dụ 2**

Chúng ta mở rộng thêm một chút, để tạo trang **Hello Your Name!**

Cấu trúc thư mục:



Mở rộng WebController như sau:

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

// Đánh dấu đây là một Controller

// Nơi tiếp nhận các reqquest từ phía người dùng

@Controller

public class WebController {

// Đón nhận request GET

@GetMapping("/") // Nếu người dùng request tới địa chỉ "/"

public String index() {

return "index"; // Trả về file index.html

}

@GetMapping("/about") // Nếu người dùng request tới địa chỉ "/about"

public String about() {

return "about"; // Trả về file about.html

}

@GetMapping("/hello")

public String hello(

// Request param "name" sẽ được gán giá trị vào biến String

@RequestParam(name = "name", required = false, defaultValue = "") String name,

// Model là một object của Spring Boot, được gắn vào trong mọi request.

Model model

) {

// Gắn vào model giá trị name nhận được

model.addAttribute("name", name);

return "hello"; // trả về file hello.html cùng với thông tin trong object Model

}

}

*index.html*

<!DOCTYPE html>

<html>

<head>

<title>Hello World</title>

</head>

<body>

<h1>Đây là một trang web</h1>

<a href="/about">About</a>

<form method="get" action="/hello">

<input type="input" name="name">

<button type="submit">Submit</button>

</form>

</body>

</html>

*about.html*

<!DOCTYPE html>

<html>

<head>

<title>Hello World</title>

</head>

<body>

<h1>Loda</h1>

<a href="https://loda.me">Website</a>

<a href="https://github.com/loda-kun/spring-boot-learning">Github</a>

</body>

</html>

*hello.html*

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<title>Hello World</title>

</head>

<body>

<h1 th:text="'Hello, ' + ${name}"></h1>

<a href="/">Trang chủ</a>

</body>

</html>

Chạy chương trình:

*App.java*

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class App {

public static void main(String[] args) {

SpringApplication.run(App.class, args);

}

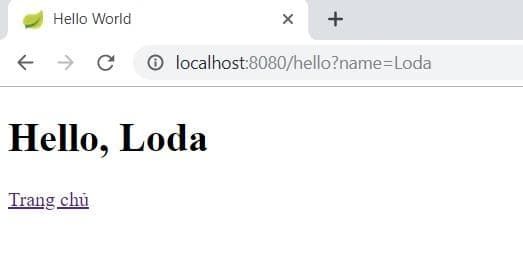
}

<http://localhost:8080/>



Nhập tên của bạn vào ô và submit. Nó sẽ tạo một request GET tới địa chỉ /hello kèm theo param ?name=your\_name.

Ví dụ tôi nhập "Loda" nó sẽ request tới <http://localhost:8080/hello?name=Loda>



**Giải thích 2**

Tới đây bạn hãy tham chiếu đường dẫn request với hàm xử lý nó:

//http://localhost:8080/hello?name=Loda

@GetMapping("/hello")

public String hello(

// Request param "name" sẽ được gán giá trị vào biến String

@RequestParam(name = "name", required = false, defaultValue = "") String name,

// Model là một object của Spring Boot, được gắn vào trong mọi request.

Model model

) {

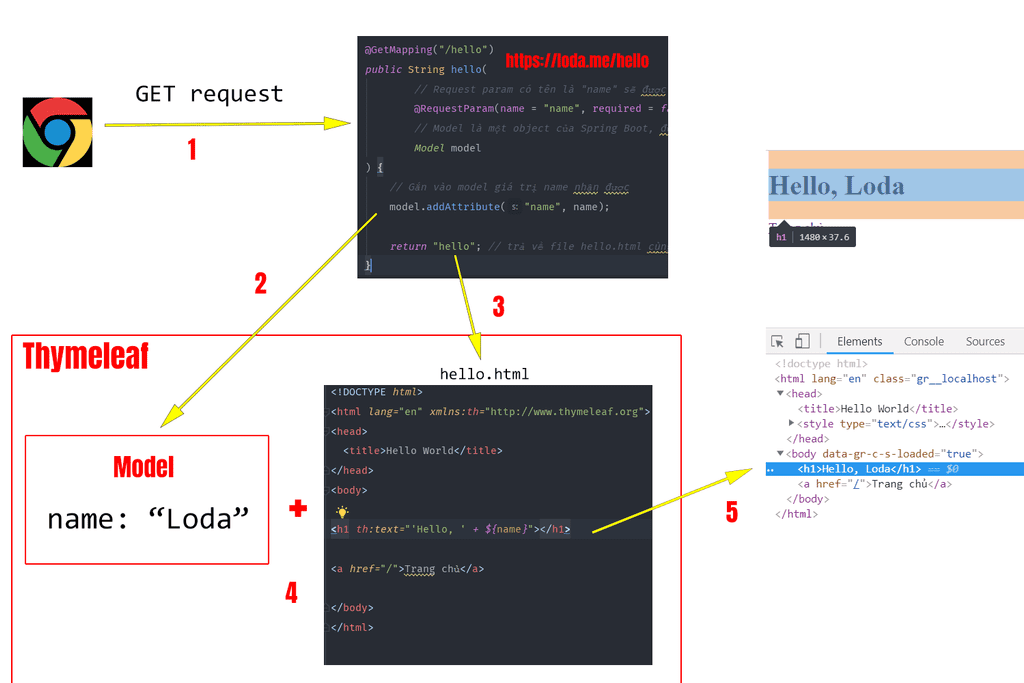
// Gắn vào model giá trị name nhận được

model.addAttribute("name", name);

return "hello"; // trả về file hello.html cùng với thông tin trong object Model

}

Khi request lên, chúng ta nhận được giá trị của name và tiếp tục gán nó vào Model.



Model ở đây là một object được **Spring Boot** đính kém trong mỗi response.

Model chứa các thông tin mà bạn muốn trả về và **Template Engine** sẽ trích xuất thông tin này ra thành html và đưa cho người dùng.

Trong file hello.html tôi lấy giá trị của name trong Model ra bằng cách sử dụng cú pháp của Thymeleaf

<h1 th:text="'Hello, ' + ${name}"></h1>

Chi tiết về Thymeleaf tôi sẽ giới thiệu ở bài sau, tạm thời tới đây bạn đã nắm được cơ bản một trang web với **Spring Boot** là như nào.