TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ TÀI**

**TÌM HIỂU VỀ SPRING BOOT**

*Người hướng dẫn*: **THẦY** **NGUYỄN CHÍ THIỆN**

*Người thực hiện*: **LÊ NHỰT KHÁNH - 51503360**

**PHẠM QUỐC THẮNG - 51503226**

Lớp **: 15050301**

Khoá  **: 19**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ TÀI**

**TÌM HIỂU VỀ SPRING BOOT**

*Người hướng dẫn*: **THẦY NGUYỄN CHÍ THIỆN**

*Người thực hiện*: **LÊ NHỰT KHÁNH - 51503360**

**PHẠM QUỐC THẮNG - 51503226**

Lớp **: 15050301**

Khoá  **: 19**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2018**

LỜI CẢM ƠN

Trước hết tôi xin tỏ lòng biết ơn sâu sắc đến Thầy Nguyễn Chí Thiện, thầy đã tận tình hướng dẫn, giúp đỡ tôi rất nhiều trong suốt quá trình thực hiện luận văn này. Sự hiểu biết sâu sắc về khoa học cũng như kinh nghiệm của thầy chính là tiền đề giúp tôi hoàn thành luận văn.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm để tài của riêng chúng tôi và được sự hướng dẫn của Thầy Nguyễn Chí Thiện;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Lê Nhựt Khánh*

*Phạm Quốc Thắng*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Khởi tạo một project với Spring Boot: biết được cách tạo một ứng dụng Web với Spring Boot và sự dễ dàng hơn trong việc thiết lập và phát triển ứng dụng so với các Spring khác.

Sử dụng Spring với JPA: Dùng JPA để thao tác với cơ sở dữ liệu trên Spring

Cách sử dụng Katharsis JSON API: Katharsis định nghĩa các tài nguyên có thể được chia sẻ trên một giao diện RESTful và một repository để xử lý chúng.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc518036444)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc518036445)

[TÓM TẮT iv](#_Toc518036446)

[MỤC LỤC 1](#_Toc518036447)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc518036448)

[CHƯƠNG 1 – TỔNG QUAN VỀ SPRING – SPRING BOOT 6](#_Toc518036449)

[1.1 Giới thiệu về Spring 6](#_Toc518036450)

[1.1.1 Khái niệm 6](#_Toc518036451)

[1.1.2 Phát triển một ứng dụng Spring 6](#_Toc518036452)

[1.1.3 Kiến trúc, các module của Spring Framework 7](#_Toc518036453)

[1.2 Spring Boot 8](#_Toc518036454)

[1.2.1 Spring boot là gì và tại sao phải sử dụng nó? 8](#_Toc518036455)

[1.2.2 Những lợi ích của Spring Boot trong việc phát triển các ứng dụng Spring 8](#_Toc518036456)

[1.2.2.1 Auto-configuration: 9](#_Toc518036457)

[1.2.2.2 Starter dependencies: 10](#_Toc518036458)

[1.2.2.3 Command-Line Interface: 11](#_Toc518036459)

[1.2.2.4 The Actuator: 12](#_Toc518036460)

[CHƯƠNG 2 PHÁT TRIỂN VỚI MỘT ỨNG DỤNG SPRING BOOT 13](#_Toc518036461)

[2.1 Sử dụng Spring Initializr Web 13](#_Toc518036462)

[2.2 Sử dụng công cụ Spring Tool Suite 16](#_Toc518036463)

[2.3 Sử dụng Spring Boot CLI 20](#_Toc518036464)

[CHƯƠNG 3 ỨNG DỤNG MINH HỌA 23](#_Toc518036465)

[3.1 Tổng quan về bài toán 23](#_Toc518036466)

[3.2 Các bước xây dựng 24](#_Toc518036467)

[3.2.1 training-lab-api project 24](#_Toc518036468)

[3.2.2 Áp dụng Spring Boot vào training-lab-api project 26](#_Toc518036469)

[3.2.3 Sử dụng JPA để truy cập dữ liệu 27](#_Toc518036470)

[3.2.4 Sử dụng Katharsis JSON API 31](#_Toc518036471)

[3.2.5 Unit Test cho Spring 37](#_Toc518036472)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1 Kiến trúc, các module của Spring Framework 7](#_Toc518036473)

[Hình 2 Spring Initializr Web 13](#_Toc518036474)

[Hình 3 Một project được tự động tải xuống 14](#_Toc518036475)

[Hình 4 Khởi chạy project 15](#_Toc518036476)

[Hình 5 404 Not Found 15](#_Toc518036477)

[Hình 6 Controller 16](#_Toc518036478)

[Hình 7 Nội dung được hiển thị trên web 16](#_Toc518036479)

[Hình 8 Tạo project 17](#_Toc518036480)

[Hình 9 Tạo project 18](#_Toc518036481)

[Hình 10 404 Not Found 19](#_Toc518036482)

[Hình 11 Controller 19](#_Toc518036483)

[Hình 12 Nội dung được hiển thị trên web 20](#_Toc518036484)

[Hình 13 Tạo project 20](#_Toc518036485)

[Hình 14 Project được tạo ra 20](#_Toc518036486)

[Hình 15 Chuyển đến thư mục làm việc của project 21](#_Toc518036487)

[Hình 16 Chạy ứng dụng 21](#_Toc518036488)

[Hình 17 Chạy ứng dụng 22](#_Toc518036489)

[Hình 18 404 Not Found 22](#_Toc518036490)

[Hình 19 Controller 23](#_Toc518036491)

[Hình 20 Cấu trúc thư mục project 25](#_Toc518036492)

[Hình 21 File pom.xml của project 25](#_Toc518036493)

[Hình 22 Thêm parent vào file pom.xml 26](#_Toc518036494)

[Hình 23 Thêm dependency spring-boot-starter-web 26](#_Toc518036495)

[Hình 24 Thêm dependency spring-boot-maven-plugin 27](#_Toc518036496)

[Hình 25 Application.java 27](#_Toc518036497)

[Hình 26 application.properties 27](#_Toc518036498)

[Hình 27 Thêm thư viện cần thiết 28](#_Toc518036499)

[Hình 28 Thư viện cho JPA và MySQL 29](#_Toc518036500)

[Hình 29 Cấu hình Database và JPA 29](#_Toc518036501)

[Hình 30 Subject.java 30](#_Toc518036502)

[Hình 31 CommonJpaRepository.java 30](#_Toc518036503)

[Hình 32 SubjectJpaRepository.java 31](#_Toc518036504)

[Hình 33 Katharsis property 32](#_Toc518036505)

[Hình 34 Katharsis dependency 32](#_Toc518036506)

[Hình 35 Thêm các chú thích 33](#_Toc518036507)

[Hình 36 SubjectRepository.java 33](#_Toc518036508)

[Hình 37 SubjectRepositoryImpl.java 34](#_Toc518036509)

[Hình 38 Thêm KatharsisConfigV3 34](#_Toc518036510)

[Hình 39 Dữ liệu được lấy dưới dạng raw 35](#_Toc518036511)

[Hình 40 Dữ liệu lấy dưới dạng json thông qua postman 35](#_Toc518036512)

[Hình 41 Tạo một subject mới thông qua POST request 35](#_Toc518036513)

[Hình 42 Dữ liệu mới được tạo 36](#_Toc518036514)

[Hình 43 Cập nhật một subject mới thông qua PATCH request 36](#_Toc518036515)

[Hình 44 Xóa dữ liệu thông qua DELETE request 36](#_Toc518036516)

[Hình 45 Dữ liệu sau khi thực hiện DELETE request 37](#_Toc518036517)

[Hình 46 Dependency mockito 38](#_Toc518036518)

[Hình 47 SubjectRepositoryImplTest 39](#_Toc518036519)

[Hình 48 SubjectRepositoryImplTest 40](#_Toc518036520)

CHƯƠNG 1 – TỔNG QUAN VỀ SPRING – SPRING BOOT

1.1 Giới thiệu về Spring

1.1.1 Khái niệm

Spring Framework, hay ngắn hơn là Spring, là một cấu trúc dùng để xây dựng và phát triển ứng dụng mã nguồn mở dành cho ngôn ngữ lập trình Java và được sử dụng bởi hàng triệu lập trình viên. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code…

Spring không bắt buộc người ta phải tuân theo một mô hình lập trình (programming model) cụ thể nào, nó như là một hình thức chủ yếu thay thế cho mô hình Enterprise JavaBean. Theo thiết kế, framework này giải phóng lập trình viên dùng Java, cho phép họ nhiều quyền tự do hơn, và đồng thời cung cấp một giải pháp tiện lợi, đầy đủ dẫn chứng bằng tài liệu, dễ dàng sử dụng, phù hợp với những thực hành thông dụng trong công nghệ phần mềm.

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: Dependency Injection và Aspect Oriented Programming.

Những tính năng core (cốt lõi) của Spring có thể được sử dụng để phát triển Java Desktop, ứng dụng mobile, Java Web. Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object)

1.1.2 Phát triển một ứng dụng Spring

Để phát triển một ứng dụng đơn giản như “Hello World” bằng Spring những điều cần làm là:

Tạo một project sử dụng **Maven** với các dependency cần thiết của **Spring MVC** và **Servlet API**.

Một tập tin **web.xml** để khai báo **DispatcherServlet** của **Spring MVC**.

Một tập tin cấu hình của **Spring MVC**.

Một class **Controller** trả về một trang “Hello World” khi có request đến.

Cuối cùng là phải có một web server dùng để triển khai ứng dụng lên chạy.

1.1.3 Kiến trúc, các module của Spring Framework

Spring được chia làm nhiều module khác nhau, tùy theo mục đích phát triển ứng dụng mà ta dùng 1 trong các module đó. Dưới đây là kiến trúc tổng thể của Spring Framework.



Hình 1 Kiến trúc, các module của Spring Framework

Test: Tầng này cung cấp khả năng hỗ trợ kiểm thử với JUnit và TestNG.

Spring Core Container: Bao gồm các module spring core, beans, context và expression languate (EL)

Spring core, bean cung cấp tính năng IOC và Dependency Injection.

Spring Context hỗ trợ đa ngôn ngữ (internationalization), các tính năng Java EE như EJB, JMX.

Expression Language được mở rộng từ Expresion Language trong JSP. Nó cung cấp hỗ trợ việc setting/getting giá trị, các method cải tiến cho phép truy cập collections, index, các toán tử logic…

AOP, Aspects and Instrumentation: Những module này hỗ trợ cài đặt lập trình hướng khía cạnh (Aspect Oriented Programming), hỗ trợ tích hợp với AspectJ.

Data Access / Integration: Nhóm này bao gồm JDBC, ORM, OXM, JMS và module Transaction. Những module này cung cấp khả năng giao tiếp với database

Web hay còn gọi là Spring MVC Nhóm này gồm Web, Web-Servlet… hỗ trợ việc tạo ứng dụng web.

Spring còn phát triển nhiều project con như: Spring MVC, Spring Security, Spring Boot, Spring Batch, Spring Social, Spring IO, Spring Cloud, Spring Mobile, Spring for Android, Spring Session. Trong đó Spring Boot là một framework giúp chúng ta phát triển cũng như chạy ứng dụng một cách nhanh chóng.

1.2 Spring Boot

1.2.1 Spring boot là gì và tại sao phải sử dụng nó?

SpringBoot là một framework giúp xây dựng các ứng dụng dựa trên Spring một cách nhanh chóng và dễ dàng. Nó sinh ra với mục tiêu chính là giảm thiếu (hoặc hoàn toàn) các cấu hình lùm xùm trong Spring, giúp những lập trình viên có thể tập trung vào business logic hơn từ đó nâng cao chất lượng của sản phẩm. Tất cả những gì bạn cần là viết mã logic cho application hoặc rất rất ít các cấu hình nguyên thủy của Spring.

1.2.2 Những lợi ích của Spring Boot trong việc phát triển các ứng dụng Spring

Tự động cấu hình (automatic configuration): Spring boot có thể tự động cấu hình những thứ cơ bản mà hầu hết các ứng dụng Spring đều phải có.

Khởi tạo các phụ thuộc (starter dependencies): Bạn chỉ việc khai báo những loại tính năng bạn cần, nó sẽ đảm bảo các phụ thuộc cần thiết sẽ được thêm vào project.

Giao diện command-line (The command-line interface): Đây là sự lựa chọn, Spring boot cho phép bạn chỉ cần viết code logic không cần xây dựng project như truyền thống.

The Actuator: Nó cho phép bạn giám sát những gì đã xảy ra bên trong một ứng dụng đang chạy.

Những tiện ích đó sẽ giúp đơn giản hóa việc phát triển ứng dụng Spring theo cách riêng của mình.

1.2.2.1 Auto-configuration:

Trong bất cứ ứng dụng Spring nào, bạn sẽ phải cấu hình sử dụng Java code hoặc xml hoặc cả 2. Ví dụ bạn viết một ứng dụng Spring có kết nối tới csdl quan hệ thì có thể bạn sẽ cấu hình như sau:

@Bean

public JdbcTemplate jdbcTemplate(DataSource dataSource) {

return new JdbcTemplate(dataSource);

}

Đoạn code trên là cách đơn giản để tạo ra một bean JdbcTemplate và nó sẽ phải phụ thuộc vào DataSource. Vì vậy, DataSource sẽ được inject vào, điều đó có nghĩa là bạn phải tạo ra một DataSource bean nữa để inject vào đó. Giá sử bạn muốn sử dụng embedded H2 database vào thì DataSource bạn phải cấu hình như sau:

@Bean

public DataSource dataSource() {

return new EmbeddedDatabaseBuilder()

.setType(EmbeddedDatabaseType.H2)

.addScripts('schema.sql', 'data.sql')

.build();

}

Phương thức build() sẽ trả về một DataSource. Ui có vẻ dài dòng và rắc rối quá nhỉ? Nhưng đó chỉ là một phần nhỏ trong những gì mình phải cấu hình trong toàn ứng dụng.

Hơn nữa, mọi ứng dụng Spring khi sử dụng H2 thì để phải sử dụng 2 phương thức (tạo bean) trên. Điều đó có nghĩa là những cấu hình đó nó như là một cái khuôn cho các ứng dụng khác sử dụng H2. Nó phổ biến và mang tính chung chung như vậy, tại sao chúng ta phải bỏ công cấu hình trong mỗi lần tạo project?

Spring boot có thể tự động cấu hình những thứ tương tự vậy, nếu Spring boot phát hiện trong classpath của bạn có lib H2 thì nó sẽ tự cấu hình H2 vào ứng dụng của bạn. Nếu JdbcTemplate có trong classpath, nó sẽ cấu hình và có thể tự động inject vào bất cứ bean nào mà bạn viết. Vì vậy, chúng ta sẽ bỏ quả được một thứ khác rắc rối rồi. So good!

1.2.2.2 Starter dependencies:

Quản lý dependencies là một vấn đề khá rắc rối trong khi xây dựng project. Thư viện nào bạn sẽ cần thêm vào project? Group và Artifact của nó là gì? Version nào bạn cần? Sersion đó có phù hợp với các thư viện khác không? Có xung đột gì không?

Nó thật là khó chịu, tốn thời gian biết bao vì trong một project thì biết bao lib phụ thuộc, không lẽ chúng ta phải nhớ từng version của nó, rồi phải test thử có xung đột hay không, rồi lỡ có thì lại thay rồi lại test, cái vòng luẩn quẩn.

Spring boot hoàn toàn có thể giúp bạn việc này, ví dụ bạn muốn xây dựng một REST API sử dụng Spring trả về Json thì bạn sẽ phải cấu hình trong build.gradle (nếu dùng Gradle) hoặc POM.xml (nếu dùng Maven) các lib sau:

■ org.springframework:spring-core

■ org.springframework:spring-web

■ org.springframework:spring-webmvc

■ com.fasterxml.jackson. core:jackson-databind

■ org.hibernate:hibernate-validator

■ org.apache.tomcat.embed:tomcat-embed-core

■ org.apache.tomcat.embed:tomcat-embed-el

■ org.apache.tomcat.embed: tomcat-embed-logging-juli

Hoặc có thể bạn add bằng tay vào classpath :D. Có vẻ hơi mệt đây, đừng lo, Spring boot sẽ giúp bạn. Ví dụ bạn muốn xây dựng web application, bạn chỉ việc khai báo "web starter" org.springframework.boot:spring-boot-starter-web trong file cấu hình (build.gradle hoặc POM.xml).

Nếu bạn muốn dùng JPA thì khi báo "jpa starter" org.springframework.boot:spring-boot-starter-jpa. Spring sẽ tự động add các thư viện cần thiết vào classpath, các lib này đã được test cẩn thận nên bạn không phải lo về vấn đề conflict hay version gì nữa. Hehe... quá đã!

1.2.2.3 Command-Line Interface:

Ngoài khả năng tự động cấu hình và khởi xướng các thư viện phụ thuộc (starter dependencies) để các lập trình viên có thể focus vào việc viết code hơn, thì Spring Boot cũng cung cấp một tính năng rất thú vị khác đó là Spring Boot CLI (Command-Line Interface).

Spring Boot CLI làm cho ứng dụng của chúng ta giảm lượng code đáng kể khi loại bỏ được số lượng lớn các dòng import rắc rối. Vậy CLI làm thế nào để biết RequestMapping hay RestController nằm trong packages nào và những class nào được thêm vào classpath?

Câu trả lời ngắn gọn đó là Spring boot CLI đã phát hiện các kiểu dữ liệu đó đang được sử dụng và nó biết starter dependencies nào được thêm vào để nó làm việc. Mỗi lần các dependencies đó được thêm vào classpath thì Spring Boot sẽ cấu hình những thứ cần thiết và chắc chắn rằng DispatcherServlet và Spring MVC đã được kích hoạt và các controller có thể phản hồi các HTTP request.

Spring Boot CLI là một sự lựa chọn nhỏ trong những gì mà Spring Boot có thể làm được. Mặc dù nó mang lại sức mạnh rất lớn và đơn giản hóa việc phát triển ứng dụng Spring, nhưng nó cũng được giới thiệu là một phong cách lập trình không theo nguyên tắc (unconvention development model). Nếu bạn không ưng phong cách này thì đó không phải là vấn đề, bạn có thể không sử dụng nó vì đơn giản, đây là một sự lựa chọn.

1.2.2.4 The Actuator:

Đây là một khả năng hữu ích trong những khả năng đáng chú ý của Spring Boot. Các thành phần trên nhằm đơn giản hóa việc phát triển ứng dụng Spring nhưng Actuator thì cung cấp khả năng kiểm tra những gì đã xảy ra bên trong một ứng dụng đang chạy, bao gồm những chi tiết như:

■ Những bean nào đã được cấu hình trong Spring application context

■ Những gì đã được Spring tự động cấu hình

■ Biến môi trường, system properties, configuration properties và command-line arguments nào có hiệu lực trong ứng dụng của bạn

■ Trạng thái hiện tại của threads và hỗ trợ ứng dụng của bạn

■ Chỉ ra HTTP request gần đây được thực hiện bởi ứng dụng

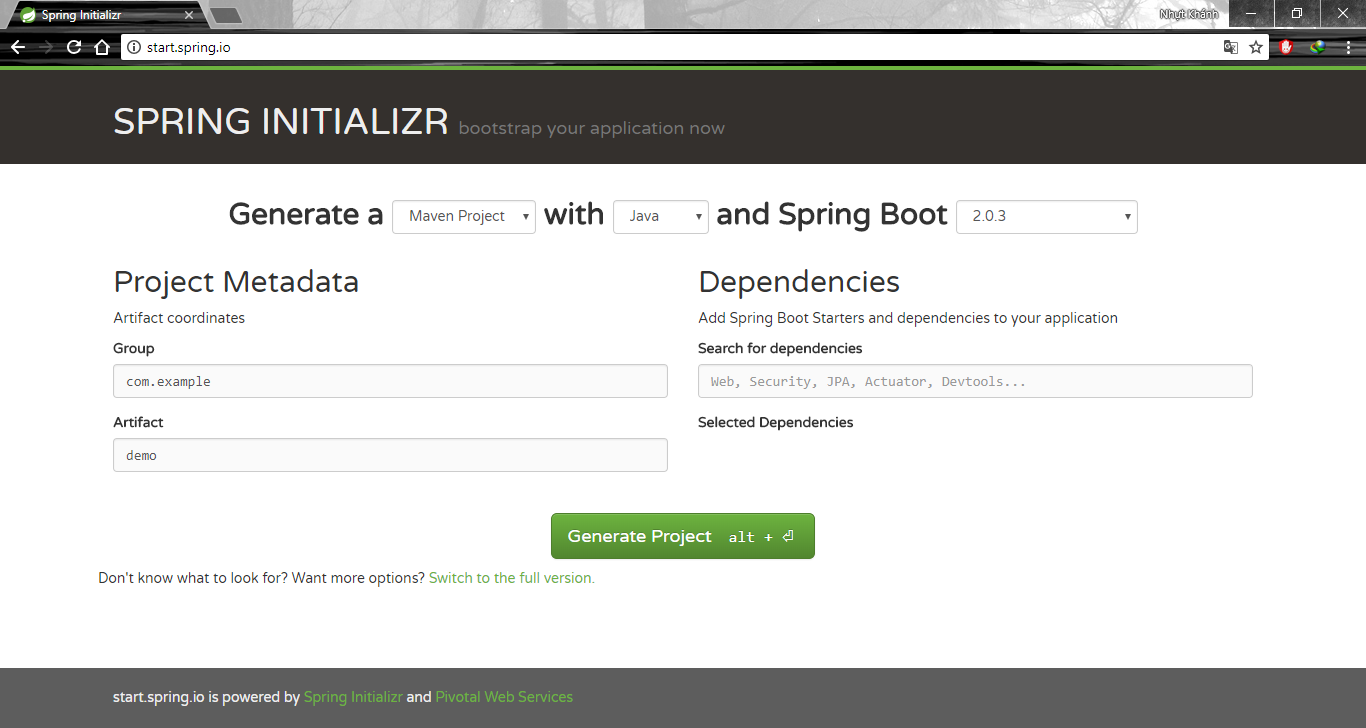
■ Các số liệu liên quan đến việc sử dụng bộ nhớ, bộ dọn rác, web requests và nguồn dữ liệu sử dụng

Actuator trình bày thông tin này bằng hai cách là thông qua web endpoints hoặc thông qua shell interface.

CHƯƠNG 2 PHÁT TRIỂN VỚI MỘT ỨNG DỤNG SPRING BOOT

2.1 Sử dụng Spring Initializr Web

Bước 1: Truy cập vào trang web của **Spring Initializr Web: start.spring.io**



Hình 2 Spring Initializr Web

Chọn: **Maven Project**

Ngôn ngữ: **Java**

Phiên bản Spring Boot: **2.0.3**

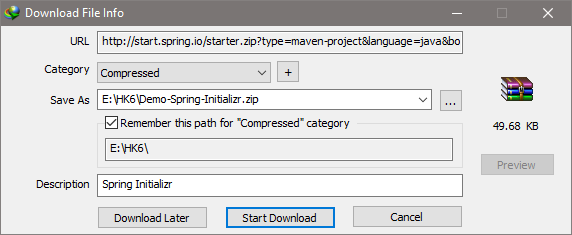
Group: **kientapcongnghiep**

Artifact: **Demo-Spring-Initializr**

Search for dependencies: **Web**

Chọn **Generate Project**

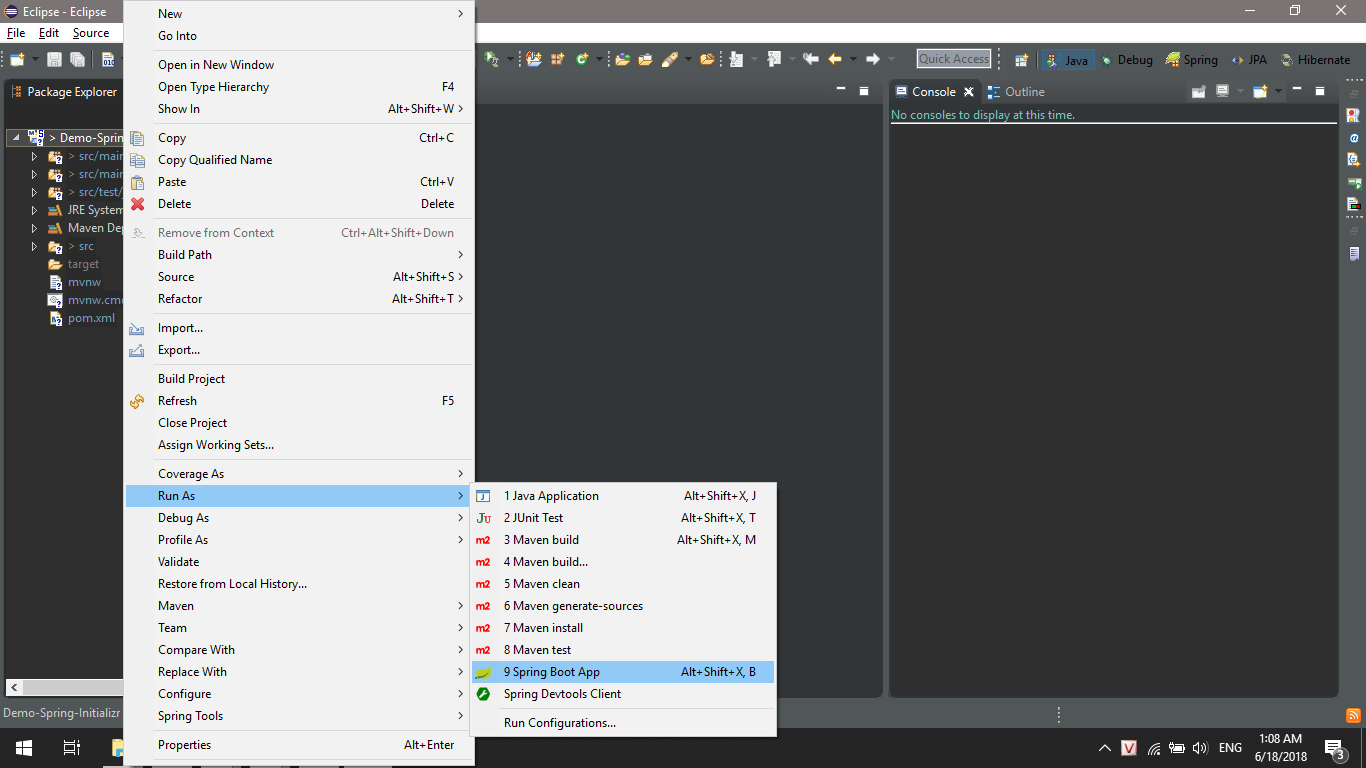
Một file vừa được tạo và tải xuống



Hình 3 Một project được tự động tải xuống

Giải nén và add project vừa được tạo vào Eclipse

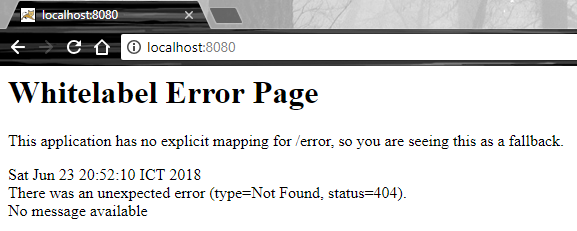
Click chuột phải vào Project vừa add chọn **Run as** chọn **Spring Boot Application**



Hình 4 Khởi chạy project

Như vậy một Spring Boot Project đã được tạo.

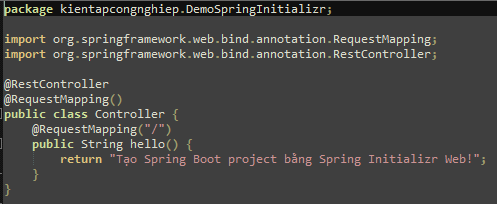
Mở trình duyệt và truy cập vào địa chỉ <http://localhost:8080>



Hình 5 404 Not Found

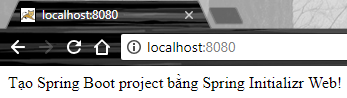
Chúng ta đang nhìn thấy một lỗi là **404 Not Found**, nguyên nhân là vì trong project mà chúng ta vừa tạo không có một **Controller** nào xử lý cho request đến trang chủ.

Lúc này project chưa có controller nên ta phải thêm vào



Hình 6 Controller

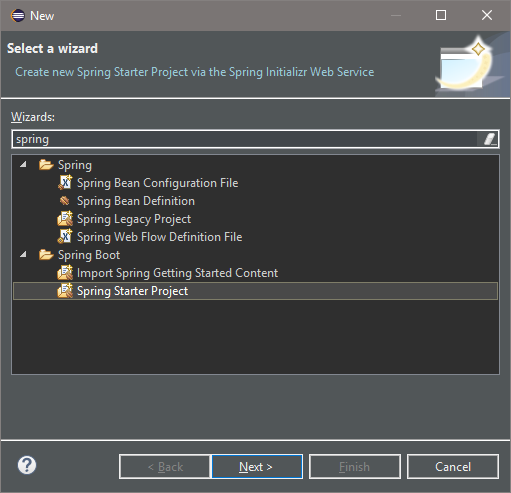
Sau đó chạy lại ứng dụng và truy cập vào địa chỉ <http://localhost:8080>



Hình 7 Nội dung được hiển thị trên web

2.2 Sử dụng công cụ Spring Tool Suite

Tạo một Spring Starter Project



Hình 8 Tạo project

Name**: Demo-Spring-Tool-Suite**

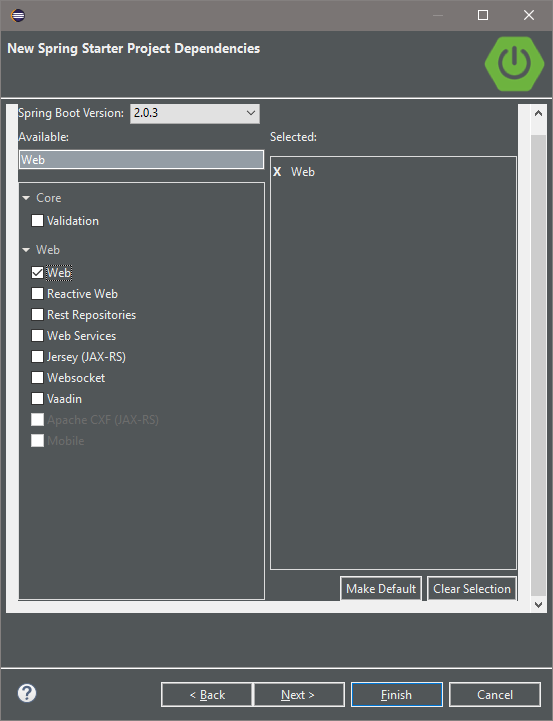
Location: **Chọn nơi lưu Project**

Type: **Maven**

Language: **Java**

Group: **kientapcongnghiep**

Artifact: **Demo-Spring-Tool-Suite**



Hình 9 Tạo project

Chọn Next chọn Spring Boot Version: **2.0.3**

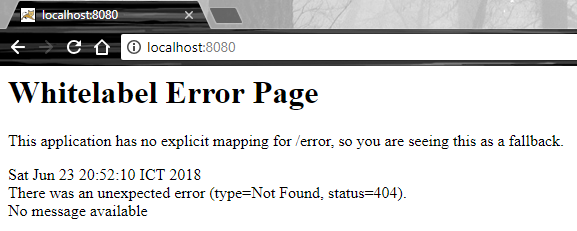
Tick vào **“Web”**

Chọn **Finish**

Như vậy một Spring Boot Project đã được tạo.

Click chuột phải vào Project vừa add chọn **Run as** chọn **Spring Boot Application**

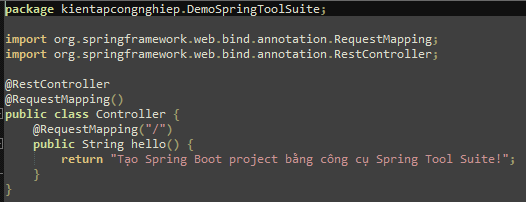
Mở trình duyệt và truy cập vào địa chỉ <http://localhost:8080>



Hình 10 404 Not Found

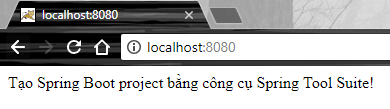
Chúng ta đang nhìn thấy một lỗi là **404 Not Found**, nguyên nhân là vì trong project mà chúng ta vừa tạo không có một **Controller** nào xử lý cho request đến trang chủ.

Lúc này project chưa có controller nên ta phải thêm vào



Hình 11 Controller

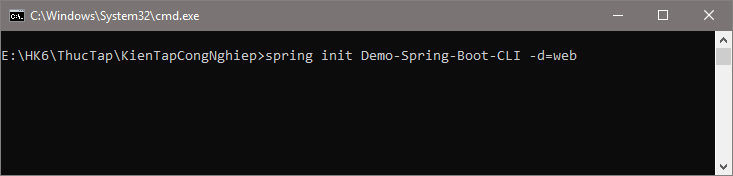
Sau đó chạy lại ứng dụng và truy cập vào địa chỉ <http://localhost:8080>



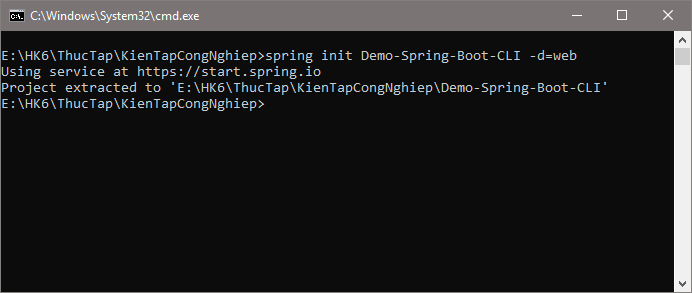
Hình 12 Nội dung được hiển thị trên web

2.3 Sử dụng Spring Boot CLI

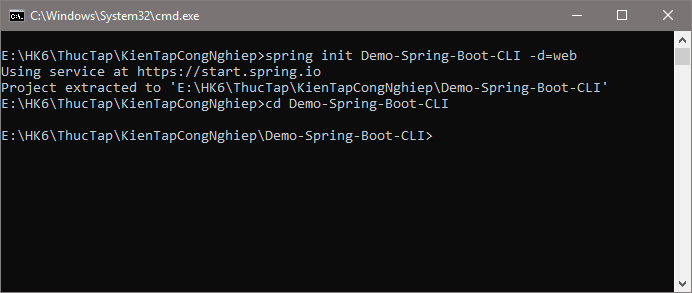
Dùng lệnh **spring init Demo-Spring-Boot-CLI –d=web** để tạo một Spring Boot project



Hình 13 Tạo project

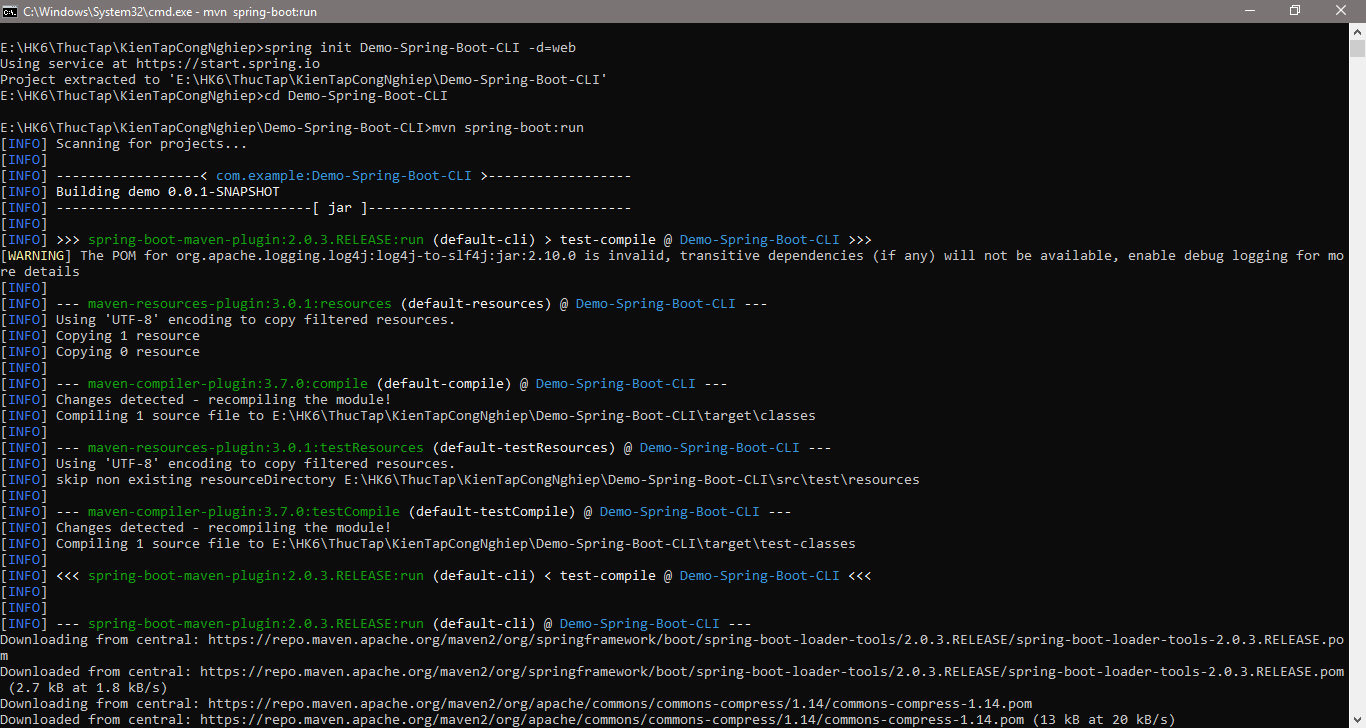


Hình 14 Project được tạo ra

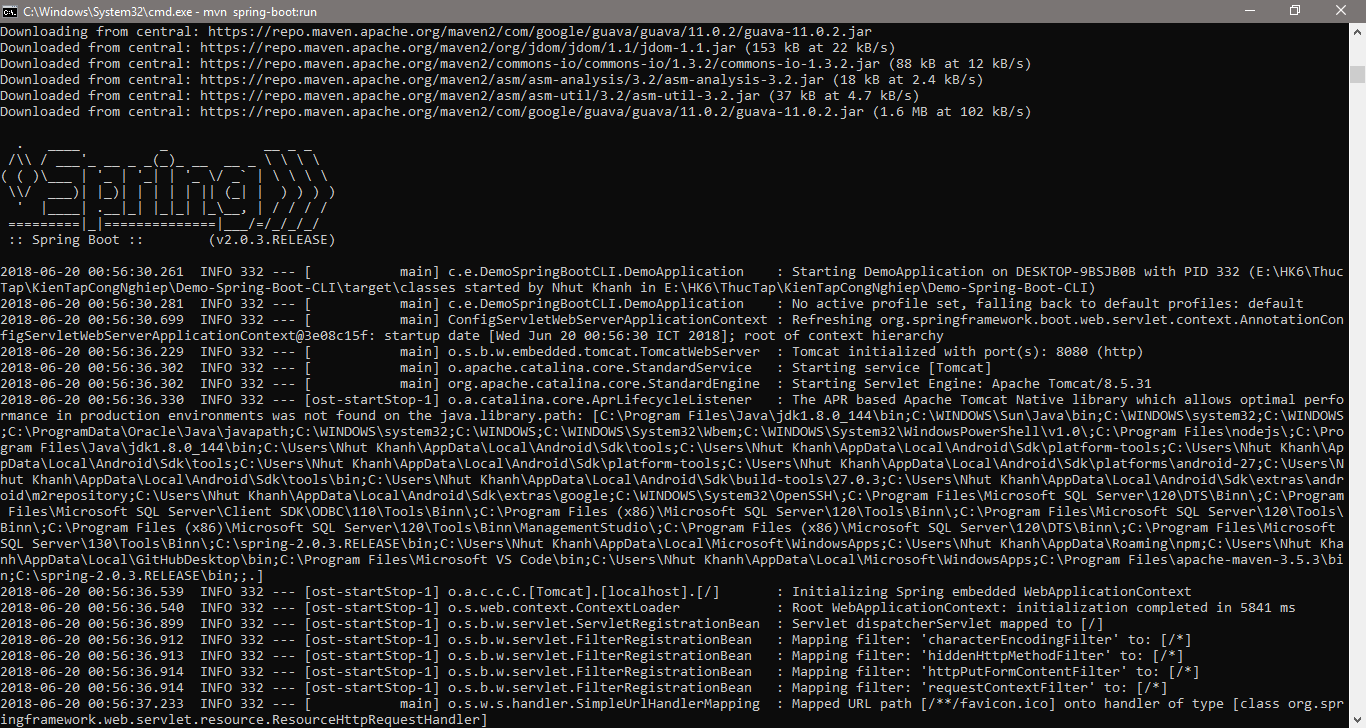


Hình 15 Chuyển đến thư mục làm việc của project

Để chạy ứng dụng ta dùng lệnh: **mvn spring-boot:run**

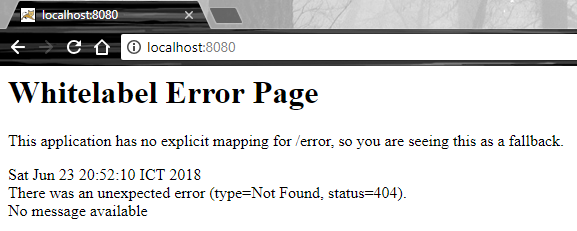


Hình 16 Chạy ứng dụng



Hình 17 Chạy ứng dụng

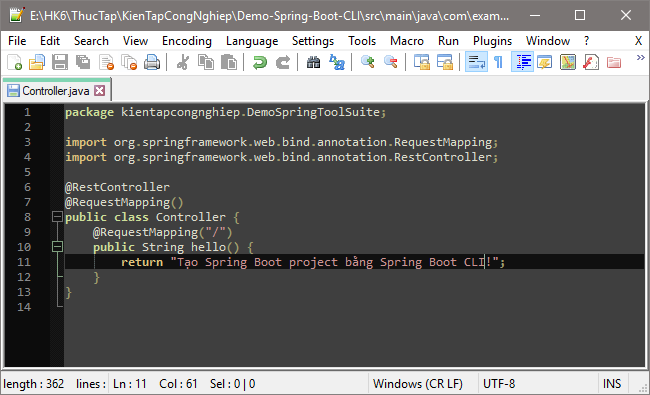
Mở trình duyệt và truy cập vào địa chỉ <http://localhost:8080>



Hình 18 404 Not Found

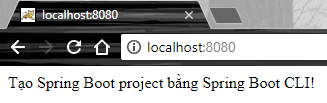
Chúng ta đang nhìn thấy một lỗi là **404 Not Found**, nguyên nhân là vì trong project mà chúng ta vừa tạo không có một **Controller** nào xử lý cho request đến trang chủ.

Lúc này project chưa có controller nên ta phải thêm vào



Hình 19 Controller

Sau đó chạy lại ứng dụng và truy cập vào địa chỉ <http://localhost:8080>



CHƯƠNG 3 ỨNG DỤNG MINH HỌA

3.1 Tổng quan về bài toán

Khi sinh viên hoặc nhân viên mới vào, họ sẽ được training theo 1 chủ đề nào đó (subject, for example: java). Trong subject thì có nhiều phân vùng (area, for example: jpa, java core, spring core, …) cần được training. Và sau mỗi phần, cần được làm bài kiểm tra, tức là làm bài kiểm tra dựa trên ngân hàng câu hỏi (questions) trong area nào đó.

Các chức năng chính của hệ thống như sau:

         Có thể thêm/xóa/sửa subject/area

         Có thể import questions

         Có thể thiết kế các bài kiểm tra

         Có thể giao bài cho user làm bài kiểm tra

         User có thể làm bài kiểm tra và biết được kết quả đạt/không đạt.

Các chức năng mở rộng của hệ thống như sau:

         Có thể thêm/xóa/sửa delivery group/ delivery center/ project

         Có thể tạo request resource từ các project

         Có thể import danh sách sinh viên từ các trường đại học gởi về

         Có thể tạo/xóa/sửa user cho các sinh viên/ nhân viên mới

         ….

3.2 Các bước xây dựng

3.2.1 training-lab-api project

Mở cmd ở thư mục cần lưu project

Gõ dòng lệnh sau: **mvn archetype:generate -DgroupId=com.tma -DartifactId=training-lab-api -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

Trong đó:

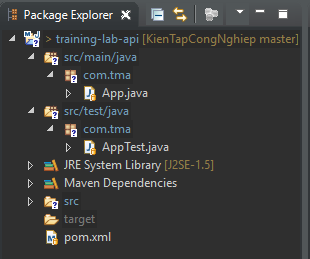
**archetype:generate** – tạo một project mới dựa trên một Archetype (khuôn mẫu) có sẵn

**DgroupId** – thường đặt theo tên của tổ chức hoặc nhóm tạo ra dự án: com.tma

**DartifactId** – thường lấy theo tên viết tắt của dự án, cũng là tên của project mà ta tạo: training-lab-api

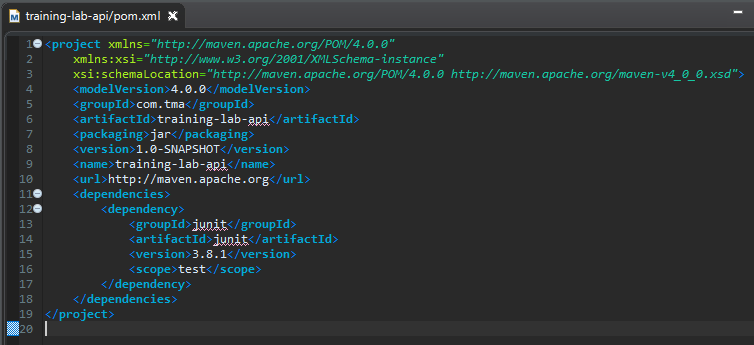
**DarchetypeArtifactId** – là loại dự án sẽ được tạo: Maven Quickstart project “để tạo cái maven của bạn với cấu trúc thư mục đơn giản và dành cho người mới học maven(đương nhiên là chỉ có đủ chức năng cho starter)”

Import project vừa tạo vào Eclipse ta được cấu trúc thư mục như sau:



Hình 20 Cấu trúc thư mục project

File **pom.xml** là nơi khai báo tất cả những gì liên quan đến dự án được cấu hình qua maven, như khai báo các dependency, version của dự án, tên dự án, repossitory, …



Hình 21 File pom.xml của project

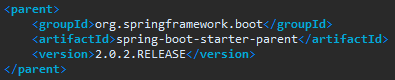
1.0-SNAPSHOT là version của project

Dependencies là những thư viện của dự án và được lưu ở thư mục Maven Dependencies ở cấu trúc thư mục hình 11.

3.2.2 Áp dụng Spring Boot vào training-lab-api project

Thêm parent vào file pom.xml

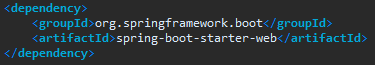
**spring-boot-starter-parent** là một project sẵn có trong **Spring Boot**. Các thư viện dependency cơ bản đã được khai báo trong **spring-boot-starter-parent**, project của bạn chỉ cần thừa kế nó. Chỉ cần khai báo **<parent>** trong file **pom.xml** của project.



Hình 22 Thêm parent vào file pom.xml

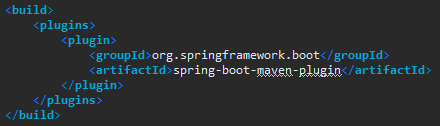
Do đang phát triển một json webservice nên chúng ta cần thêm dependency **spring-boot-starter-web vào file pom.xml**

Các **"Starter"** khác chỉ đơn giản là cung cấp phụ thuộc mà bạn có khả năng cần thiết khi phát triển một loại hình cụ thể của ứng dụng. Chẳng hạn khi bạn phát triển một ứng dụng web, bạn cần một dependency **spring-boot-starter-web**.



Hình 23 Thêm dependency spring-boot-starter-web

**Spring-boot-maven-plugin** là plugin cung cấp các thư viện cần thiết giúp project của bạn có thể chạy trực tiếp mà không cần triển khai trên một **Web Server**. Nó giúp tạo ra một file **jar** có thể thực thi (Executable)



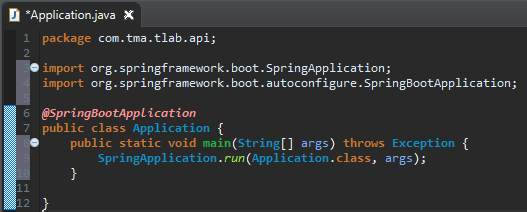
Hình 24 Thêm dependency spring-boot-maven-plugin

Xóa 2 class App và AppTest trong package com.tma

Tạo package tlab.api trong com.tma

Tạo class Application trong package vừa tạo

Chỉnh sửa class Application để trở thành trình khởi chạy với Spring Boot. Thêm @SpringBootApplication trước class Application và phương thức main().



Hình 25 Application.java

Để thay đổi port của ứng dụng Spring ta tạo file **application.properties** và sau đó thêm server.port = 9090 vào tệp này để thay đổi từ port 8080 sang 9090



Hình 26 application.properties

3.2.3 Sử dụng JPA để truy cập dữ liệu

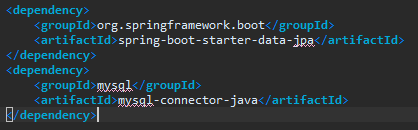
Trong **JPA** mỗi một lớp **Model** sẽ tương ứng với một bảng trong cơ sở dữ liệu. Có rất nhiều bảng trong cơ sở dữ liệu vì vậy sẽ có rất nhiều lớp **Model**. Thường xuyên phải làm việc với các **Model**, và phải viết các lớp **Repository** để thao tác với dữ liệu thông qua các **Model** này nó sẽ làm mất rất nhiều thời gian. **Spring Data JPA** là một thư viện của **Spring**. Theo quy tắc của **Spring Data JPA** bạn chỉ cần định nghĩa một interface mở rộng interface **Repository<T, ID>**, và khai báo tên các phương thức để thao tác với dữ liệu của **Model** này. **Spring Data JPA** sẽ tự tạo một lớp implements interface.

Thêm một số thư viện cần thiết cho ứng dụng Spring



Hình 27 Thêm thư viện cần thiết

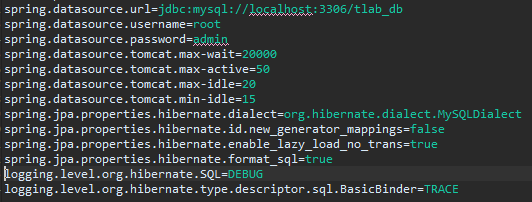
Thêm dependency spring-boot-starter-data-jpa để lưu trữ và truy xuất dữ liệu trong cơ sở dữ liệu quan hệ. Thêm dependency mysql-connector-java để kết nối với MySQL



Hình 28 Thư viện cho JPA và MySQL

Tạo một cơ sở dữ liệu là tlab\_db với các bảng như sau

Để **Spring** có thể kết nối vào Database thì cần phải cấu hình các thông số về database và jpa trong tập tin **application.properties**



Hình 29 Cấu hình Database và JPA

Tạo package model để tạo các class tương ứng là một class đại diện cho một bảng trong cơ sở dữ liệu. Các trường trong lớp này sẽ tương ứng với các cột trong bảng.

Chúng ta sẽ tạo lớp **Subject** để đại diện cho bảng **subject** trong database. Các **JPA Annotation** sẽ được sử dụng để chú thích trên các trường để mô tả cách ánh xạ giữa các trường và các cột của bảng. Các ánh xạ này là 1-1, mỗi trường tương ứng với 1 cột trong bảng.

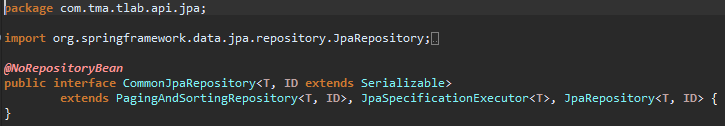


Hình 30 Subject.java

Các bảng còn lại tương tự.

Tạo package jpa để có các phương thức để thao tác với từng model

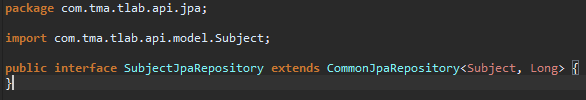
Tạo interface CommonJpaRepository trong package com.tma.tlab.api.jpa.



Hình 31 CommonJpaRepository.java

Tạo interface SubjectJpaRepository trong package com.tma.tlab.api.jpa

Interface SubjectJpaRepository extends interface **CommonJpaRepository <Subject, Long>**, nó có các phương thức để thao tác với model Subject. **Spring Data JPA** sẽ tự động tạo một lớp implements interface này tại thời điểm chạy của ứng dụng.



Hình 32 SubjectJpaRepository.java

3.2.4 Sử dụng Katharsis JSON API

Katharsis cung cấp các phương tiện để dễ dàng hiển thị các tài nguyên trên giao diện REST. Katharsis bao hàm hầu hết các loại request đặc tả JSON API như: GET, POST, PATCH, DELETE

Có một số annotations có thể được gán cho các model. Theo mặc định, tất cả các trường của model được phản ánh trong giao tiếp API JSON ngoại trừ các trường tổng hợp.

JsonApiResource: Đây là chú thích quan trọng nhất định nghĩa tài nguyên. Nó đòi hỏi tham số kiểu được định nghĩa được sử dụng để tạo thành một URL và trường kiểu trong JSON được truyền.

JsonApiId: Định nghĩa một trường sẽ được sử dụng như một định danh của một tài nguyên. Mỗi tài nguyên yêu cầu chú thích này phải có mặt trên một trường có kiểu thực hiện Serializablehoặc thuộc kiểu nguyên thủy.

JsonApiRelation: Cho biết liên kết với một giá trị đơn lẻ hoặc tập hợp các tài nguyên. Loại trường như vậy phải là tài nguyên hợp lệ.

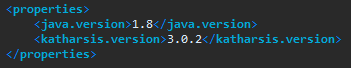
ResourceRepositoryV2: được sử dụng để hoạt động trên tài nguyên. Mỗi tài nguyên phải có triển khai repository tương ứng. Nó bao gồm các phương thức cơ bản cung cấp cho tài nguyên. Các phương pháp như sau:

findAll(QuerySpec querySpec) Tìm kiếm tất cả các tài nguyên. Một thể hiện của QuerySpec có thể được sử dụng nếu cần thiết. Nếu không có tài nguyên có thể được tìm thấy một Iterable rỗng hoặc null phải được trả lại. Nó sẽ trả về các thực thể có quan hệ liên quan.

save(S entity) Lưu một tài nguyên. Nó không nên lưu các mối quan hệ liên quan. Tài nguyên trả về phải bao gồm mã định danh được gán cho cá thể tài nguyên. Phương pháp này có thể vừa tạo tài nguyên mới vừa cập nhật tài nguyên hiện có.

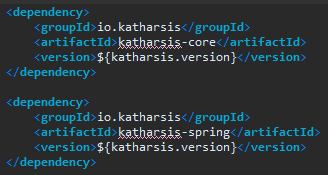
delete(ID id) Loại bỏ một tài nguyên được xác định bởi tham số id.

Thêm Katharsis property



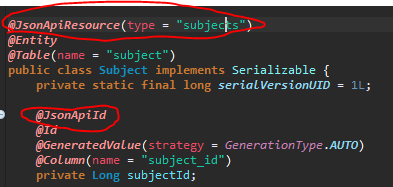
Hình 33 Katharsis property

Và thêm Katharsis dependency



Hình 34 Katharsis dependency

Thêm @JsonApiResource (type = "subjects") và @JsonApiId để mô tả nguồn 'subjects'

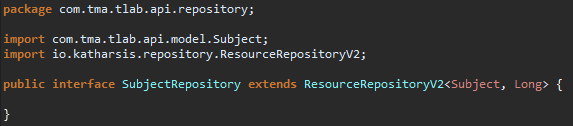


Hình 35 Thêm các chú thích

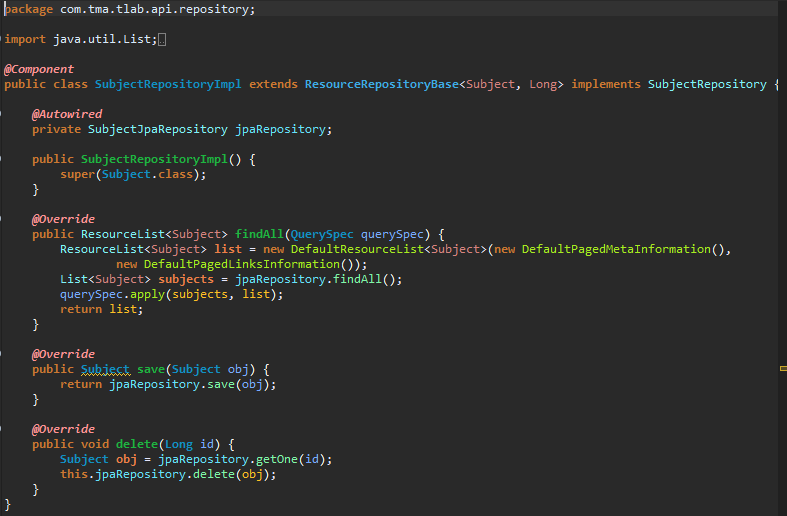
Các class còn lại làm tương tự

Tạo package repository

Tạo class SubjectRepository và class SubjectRepositoryImpl trong package reponsitory



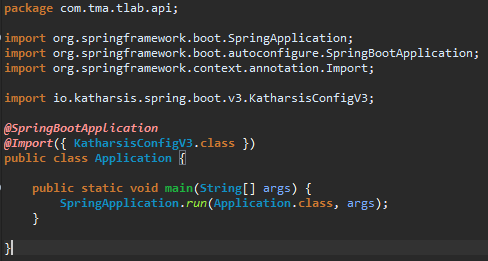
Hình 36 SubjectRepository.java



Hình 37 SubjectRepositoryImpl.java

Các class còn lại tương tự

Thêm KatharsisConfigV3 vào class Application



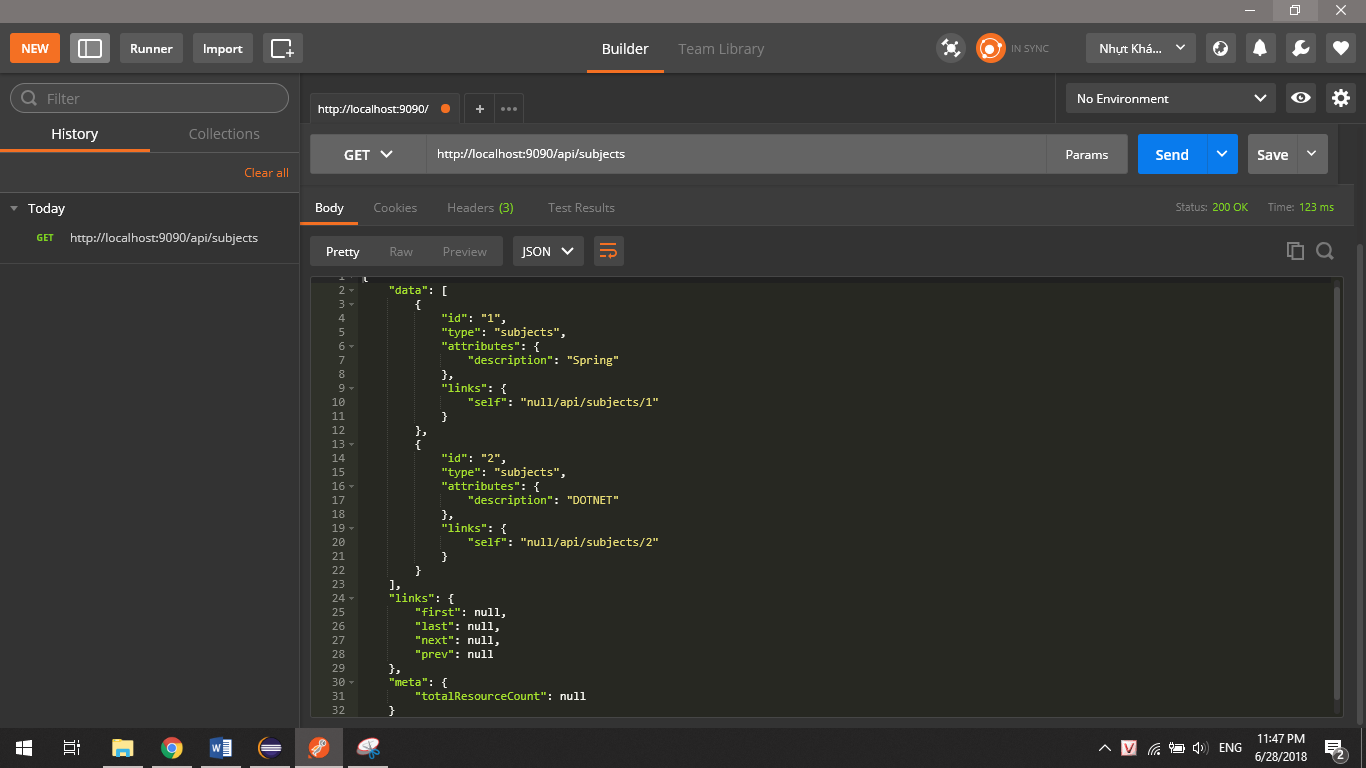
Hình 38 Thêm KatharsisConfigV3

Thêm katharsis.pathPrefix=/api vào file application.properties

Truy cập vào <http://localhost:9090/api/subjects> sẽ hiển thị json như sau

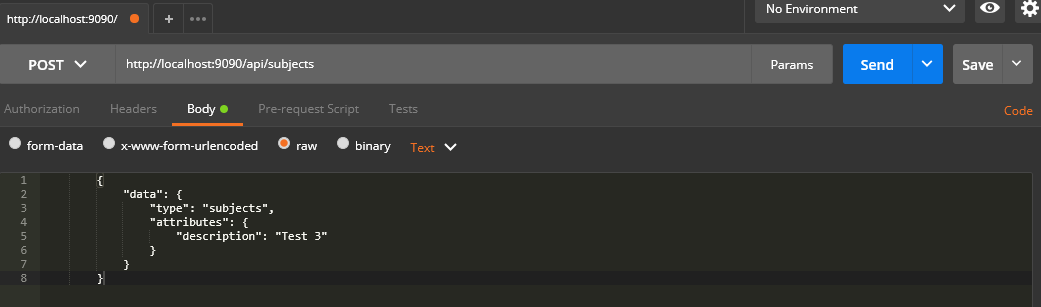


Hình 39 Dữ liệu được lấy dưới dạng raw

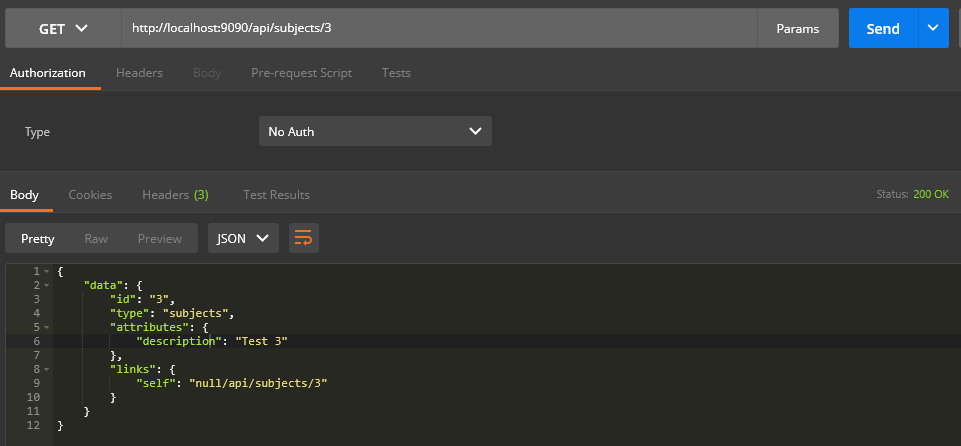
Để tiện hơn chúng ta cài postman và truy cập lại <http://localhost:9090/api/subjects>

Hình 40 Dữ liệu lấy dưới dạng json thông qua postman

Chúng ta có thể dùng lệnh POST để tạo một subject mới vào bảng subject trong cơ sở dữ liệu

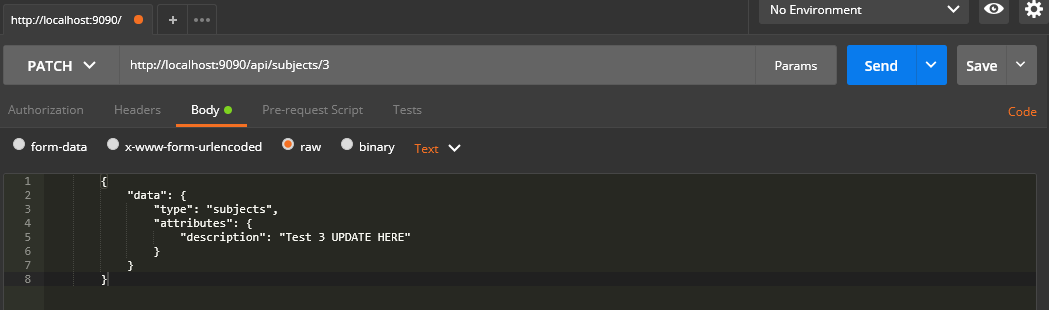


Hình 41 Tạo một subject mới thông qua POST request

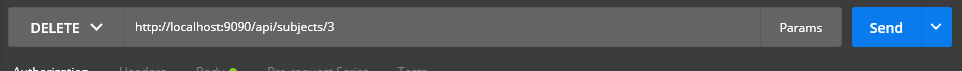


Hình 42 Dữ liệu mới được tạo

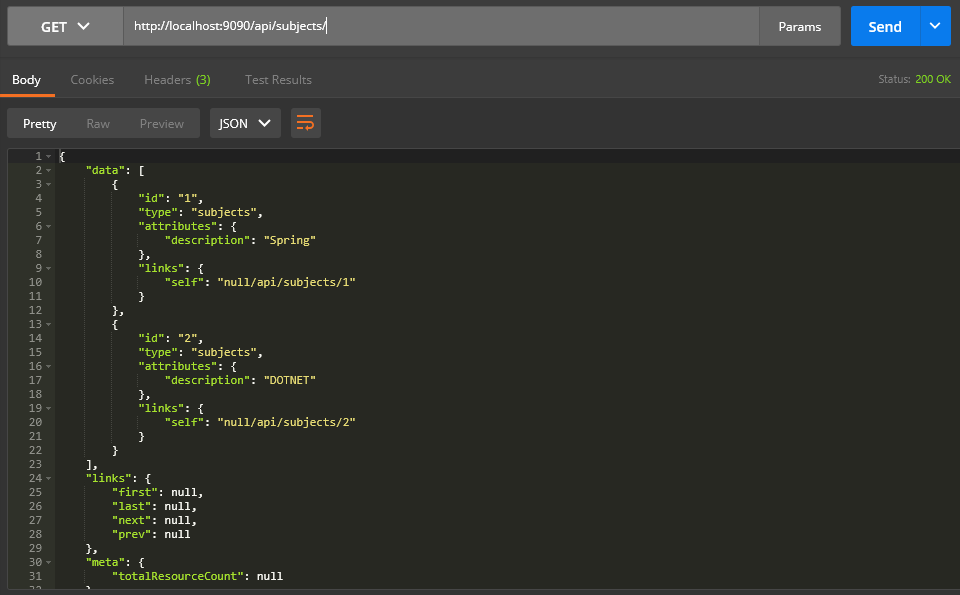
Dùng lệnh PATCH để chỉnh sửa một subject mới vào bảng subject trong cơ sở dữ liệu



Hình 43 Cập nhật một subject mới thông qua PATCH request



Hình 44 Xóa dữ liệu thông qua DELETE request



Hình 45 Dữ liệu sau khi thực hiện DELETE request

3.2.5 Unit Test cho Spring

JUnit là một framework dùng cho việc tạo các test case để kiểm thử các method của các đối tượng trong chương trình Java.

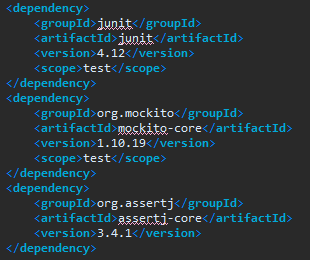
Trong nhiều trường hợp, method được kiểm thử gọi đến những phương thức của một interface nào đó mà interface đấy chưa được implements bới bất kỳ class nào. Khi đó, trong Unit Test, ta cần giả lập các hàm của interface được sử dụng. Và Mockito là một framework thường dùng để giả lập kiểm thử, tạo giả một đối tượng cài đặt interface, giả lập lời gọi hàm đến đối tượng giả này và chỉ định kết quả trả lại cho lời gọi hàm giả lập đó.

**Các hàm hay sử dụng trong Mockito**

Mockito.when(T methodCall): dùng để giả lập một lời gọi hàm nào đó được sử dụng bên trong method đang được kiểm thử.

Mockito.when(): thường đi kèm với .thenReturn(), .thenAnswer(), .thenThrow() để chỉ định kết quả trả lại.

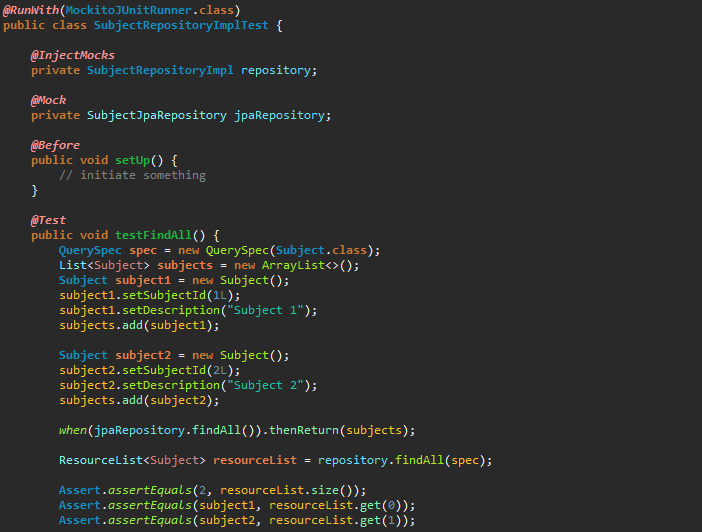
Thêm dependency mockito



Hình 46 Dependency mockito

Tạo package com.tma.tlab.api.repository trong thư mục src\test\java

Tạo class SubjectRepositoryImplTest để viết unit test cho SubjectRepositoryImpl



Hình 47 SubjectRepositoryImplTest

Vai trò của các annotation trong SubjectRepositoryImplTest trên.

[@Mock](https://viblo.asia/u/Mock) sẽ tạo ra mock implementation cho dependency SubjectJpaRepository.

[@InjectMocks](https://viblo.asia/u/InjectMocks) là dành đối tượng đc test, ở đây chính là SubjectRepositoryImpl.



Hình 48 SubjectRepositoryImplTest

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. https://viblo.asia/p/mockito-tu-a-den-z-WAyK8xAoKxX
2. https://o7planning.org/vi/11669/spring-boot

**Tiếng Anh**

1. http://www.vogella.com/tutorials/Mockito/article.html
2. https://katharsis-jsonapi.readthedocs.io/en/latest/
3. **Keith**, Mike, **Schincariol**, Merrick Pro JPA 2 - 2nd Edition

Craig Walls, *Spring Boot in Action*

K Siva Prasad Reddy, *Springboot learn by example sample*