* 1. Spring

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 Framework 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)

* Phát triển một ứng dụng Spring

Để phát triển một ứng dụng đơn giản như “Hello World” bằng Spring bạn cần làm gì?

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.

# 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. 

## 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

* 1. Spring Boot

Spring Boot giúp bạn dễ dàng tạo các ứng dụng dựa trên Spring, giúp chúng ta dễ dàng hơn trong việc thiết lập và phát triển ứng dụng Spring. Việc cấu hình Spring trở nên đơn giản



* Phát triển một ứng dụng Spring sử dụng Spring Boot

Nó rất dễ dàng để phát triển các ứng dụng dựa trên Spring.

Nó tránh việc phải viết nhiều mã nguyên mẫu, Annotations và các cấu hình XML.

Nó dễ dàng để bạn tương tác các ứng dụng Spring Boot với các hệ sinh thái của Spring như Spring JDBC, Spring ORM, Spring Data, Spring Security etc.

Nó cung cấp các Server nhúng (Embedded HTTP servers) như là Tomcat, Jetty .... để phát triển và test các ứng dụng web nhanh chóng và dễ dàng.

Nó cung cấp công cụ CLI (Command Line Interface) dể phát triển và test các ứng dụng Spring Boot từ các dòng lệnh rất dễ dàng và nhanh chóng.

Nó cung cấp rất nhiều các plugin để phát triển và test các ứng dụng Spring Boot nhanh chóng sử dụng các công cụ Build như Maven và Gradle

Nó cung cấp nhiều plugin để làm việc với các cơ sở dữ liệu nhúng (embedded database)  và các cơ sở dữ liệu lưu trữ trên bộ nhớ (in-memory Databases) một cách dễ dàng.

* **Các yếu tố cần thiết cho Spring Boot**

Cấu hình một cách tự động

Khởi tạo các dependencies

Command Line Interface(CLI)

Actuator

**Tổng quan về Spring Boot**

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

Spring framework đã xuất hiện từ rất lâu, nó hỗ trợ cho chúng ta rất nhiều trong việc xây dựng các ứng dụng doanh nghiệp bằng việc áp dụng các khái niệm như MVC, Inversion Of Control (IoC), AOP (aspect-oriented programming) với nhiều module mạnh mẽ cho security, dependency management,... Nhưng trong ứng dụng Spring để sử dụng các tính năng của nó chúng ta phải cấu hình rất nhiều bằng cách viết những file cấu hình khá là phức tạp, như là web.xml để cấu hình dispatcher servlet, nào là xxx-servlet.xml,...

Mặc dù trong Spring 2.5 đã giới thiệu annotation để giảm thiểu số file ấy mà thay bằng cấu hình bằng java, chúng ta vẫn chẳng thể tránh những cấu hình phiền toái này.

Ví dụ như việc config transaction, các filter hay quản lý các dependency chúng ta phải cấu hình trong những file xml hoặc trong code java. Việc cấu hình như thế đòi hỏi khá nhiều thời gian cũng công sức, thay vì phải ngồi cấu hình từng chút một như vậy thì để thời gian cho business logic có phải là hơn không? Hơn nữa, việc quản lý các dependency quả là một việc làm vô nghĩa, tại sao chúng ta phải đầu từ thời gian vào công việc này? Phải giải quyết xung đột giữa chúng hay những vấn đề về version?

Hiểu được như thế, các nhà phát triển đã cho ra Spring Boot 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.

Nói chung, Spring boot là một project được xây dựng dựa trên những cái chung mà trong các project spring nào cũng có, từ đó giúp giảm bớt những cấu hình rắc rối hay phải quản lý các version để tập trung vào code logic hơn, nâng cao chất lượng sản phẩm.

**Những cái hay ho mà Spring Boot đã tác động tới project Spring?**

Spring Boot đã mang lại rất nhiều những tiện ích đáng kinh ngạc trong việc phát triển các ứng dụng Spring, nhưng có 4 tiện ích chính mà nó đã thực hiện là:

- 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.

**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!

**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!

**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.

**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

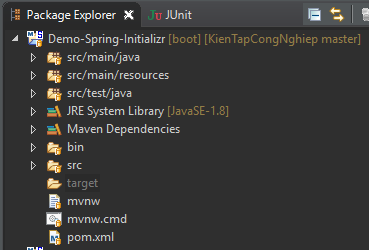
■ Chỉ ra HTTP request gần đây được thực hiện bởi ứng dụng

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

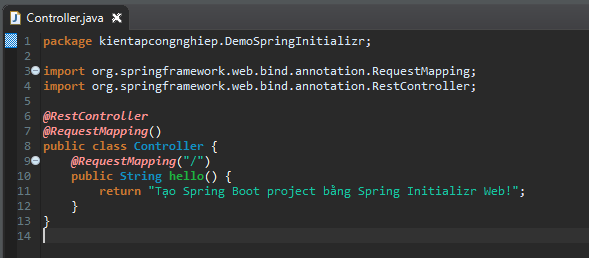
Actuator trình bày thông tin này bằng hai cách là thông qua web endpoints hoặc thông qua shell interface.

**Cách hoạt động của String Boot**

Lấy 1 project vừa mới tạo là: Demo-**Spring-Initializr-Web**



Hình 1 Demo-**Spring-Initializr-Web project**

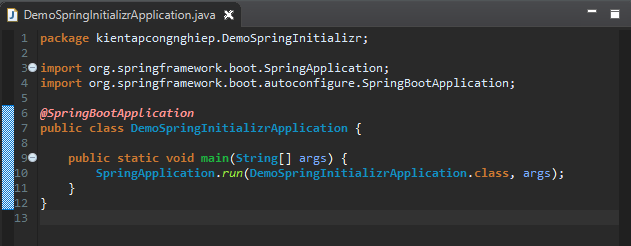


Hình 2 Controller

Khi truy cập vào địa chỉ <http://localhost:8080/> thì màn hình sẽ hiện kết quả như sau:

Dễ dàng nhìn thấy chúng ta không cần khai báo bất kì tập tin cấu hình nào như: web.xml, annotation @Configuration hay bất kỳ bean nào trong Spring container nhưng dụng vẫn chạy bình thường và xuất nội dung ra màn hình

Để ứng dụng có thể chạy được như vậy thì chúng ta cần dùng đến annotation @SpringBootApplication được khai báo trong class **DemoSpringInitializrApplication.java**



Hình 3 DemoSpringInitializrApplication

Chính annotation @SpringBootApplication này đã quét các thành phần và có thể xác định cấu hình bổ sung trên class DemoSpringInitializrApplication. Một annotation @SpringBootApplication có thể được sử dụng để bật ba tính năng là:

@EnableAutoConfiguration : Bật cơ chế cấu hình tự động của Spring Boot

@ComponentScan: bật @Component quét những package mà ứng dụng được đặt tại đó

@Configuration cho phép đăng ký đậu phụ trong ngữ cảnh hoặc nhập các lớp cấu hình bổ sung

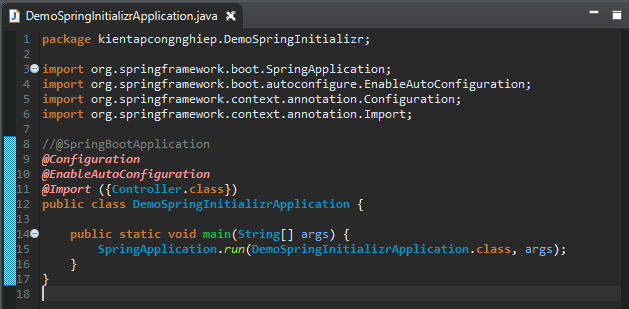
Sử dụng Annotation @SpringBootApplication thì tương đương với việc sử dụng @EnableAutoConfiguration, @ComponentScan và @Configuration trong đó annotation @EnableAutoConfiguration là quan trọng nhất. Với @EnableAutoConfiguration annotation, Spring Boot sẽ tự động cấu hình cho ứng dụng của chúng ta dựa vào classpath, các annotations và các thông tin cấu hình mà chúng ta định nghĩa.

Để thực hiện việc cấu hình một cách tự động, đầu tiên Spring Boot sẽ kiểm tra classpath trong project của mình, và bởi vì dependency của có spring-boot-starter-web nên Spring Boot sẽ cấu hình ứng dụng là một ứng dụng web. Thêm nữa, Spring Boot sẽ coi Controller là một web controller dựa vào annotation @Controller và @RequestMapping.

@SpringBootApplication cũng cung cấp bí danh để tùy chỉnh các thuộc tính của @EnableAutoConfiguration và  @ComponentScan.

Không có tính năng nào trong số này là bắt buộc và chúng ta có thể thay thế chú thích đơn này bằng bất kỳ tính năng nào mà nó cho phép

Chúng ta có thể thay thế như thế này

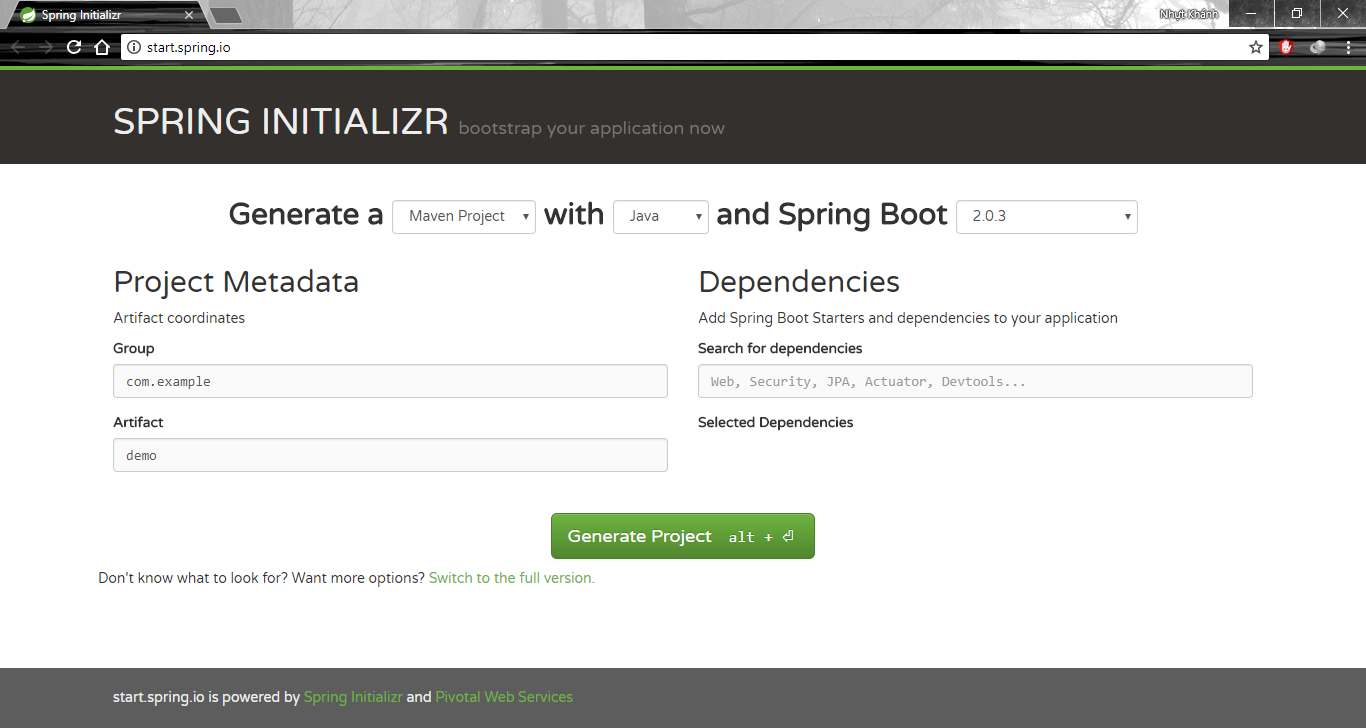


Hình 4

**Cách tạo mới một Spring Boot project**

**Sử dụng Spring Initializr Web**

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



Hình 5 **Spring Initializr Web**

Chọn: **Maven Project**

Ngôn ngữ: **Java**

Phiên bản Spring Boot: **2.0.3**

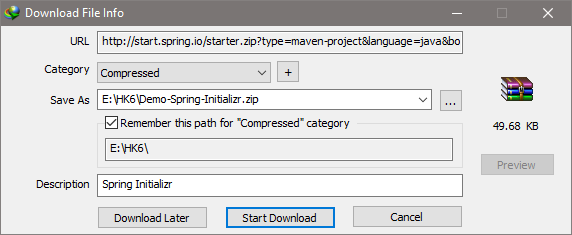
Group: **kientapcongnghiep**

Artifact: **Demo-Spring-Initializr**

Search for dependencies: **Web**

Chọn **Generate Project**

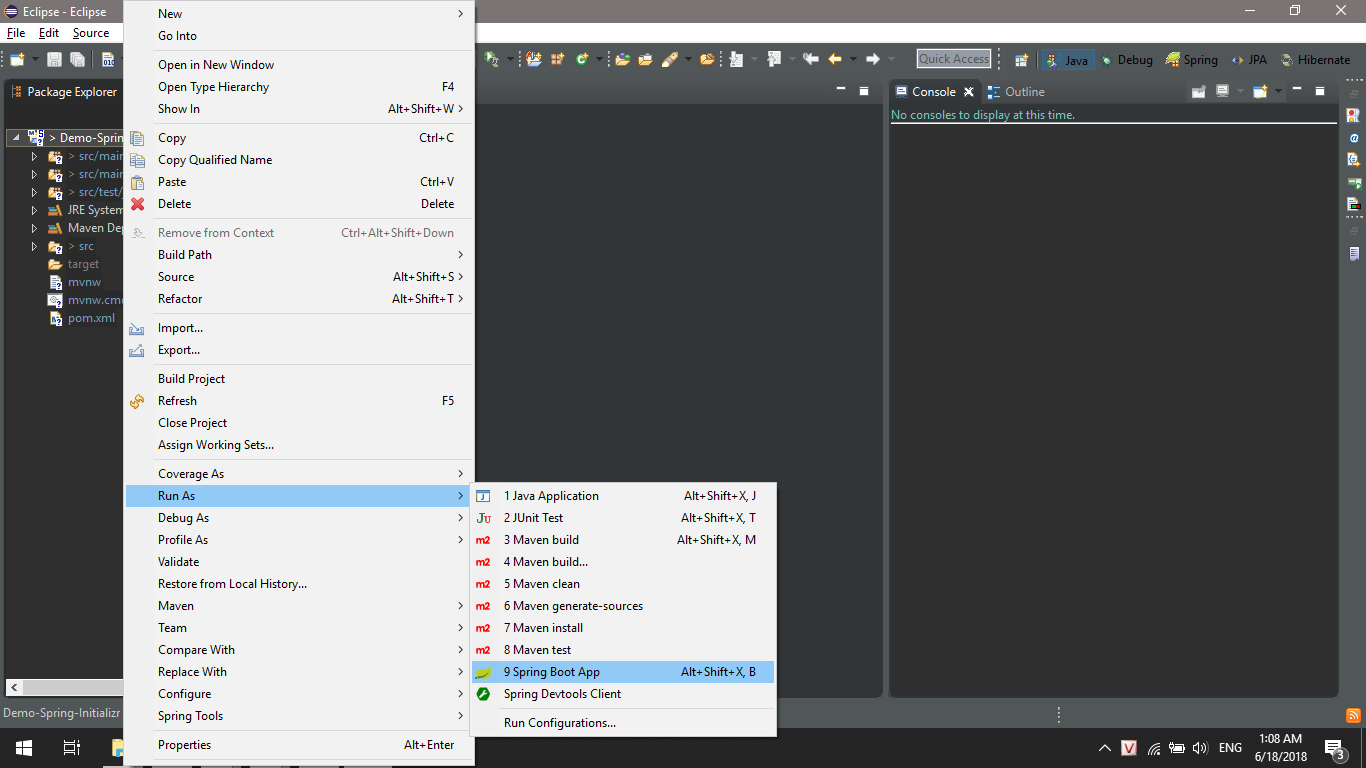
Một file vừa được tạo và tải xuống



Hình 6 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 7 Khởi chạy project

Như vậy một Spring Boot Project đã được tạo.

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

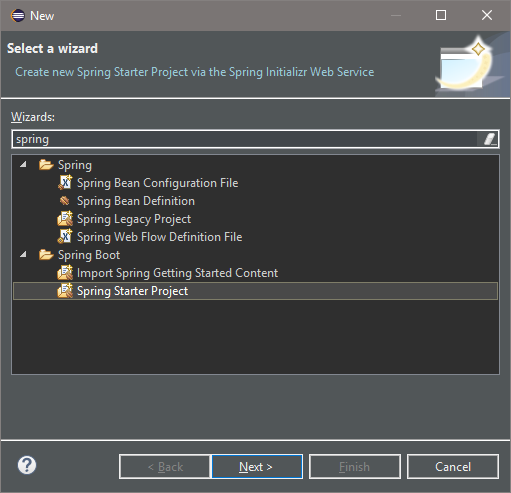
Chúng ta đang nhìn thấy một lỗi là **404 Not Found**, nguyên nhân là vì trong project mà chúng ta vừa tạo không có một **Controller** nào xử lý cho request đến trang chủ.

Lúc này project chưa có controller nên ta phải thêm vào

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

**Sử dụng công cụ Spring Tool Suite**

Tạo một Spring Starter Project



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

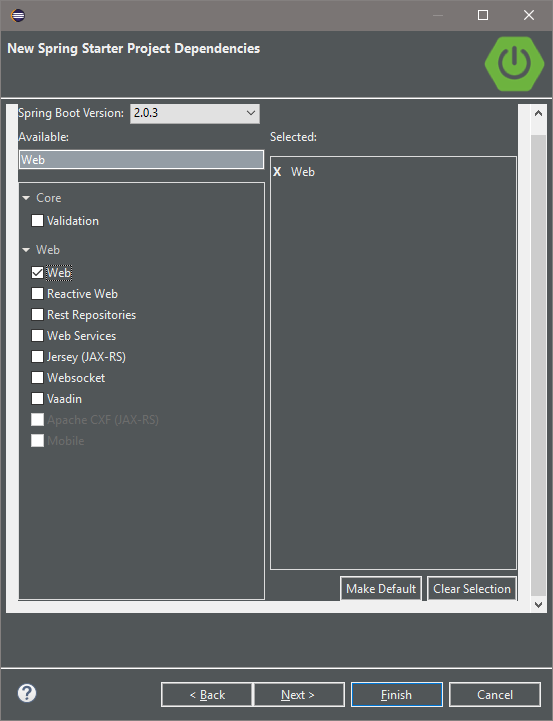
Location: **Chọn nơi lưu Project**

Type: **Maven**

Language: **Java**

Group: **kientapcongnghiep**

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



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**

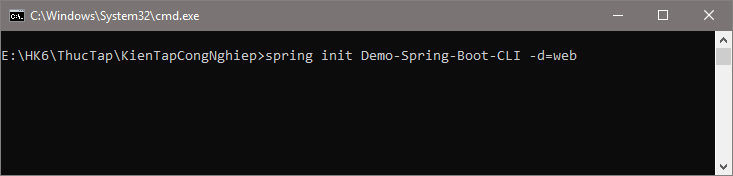
Mở trình duyệt và truy cập vào địa chỉ <http://localhost:8080>

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.

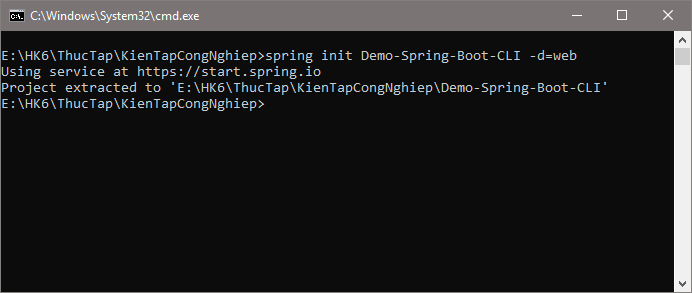
Lúc này project chưa có controller nên ta phải thêm vào

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

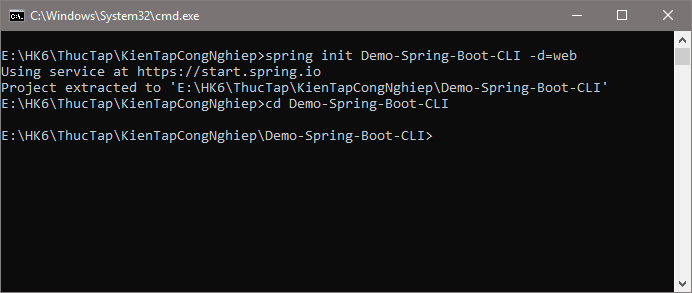
Dùng lệnh spring init Demo-Spring-Boot-CLI –d=web để tạo một Spring Boot project



Hình 8 Tạo project

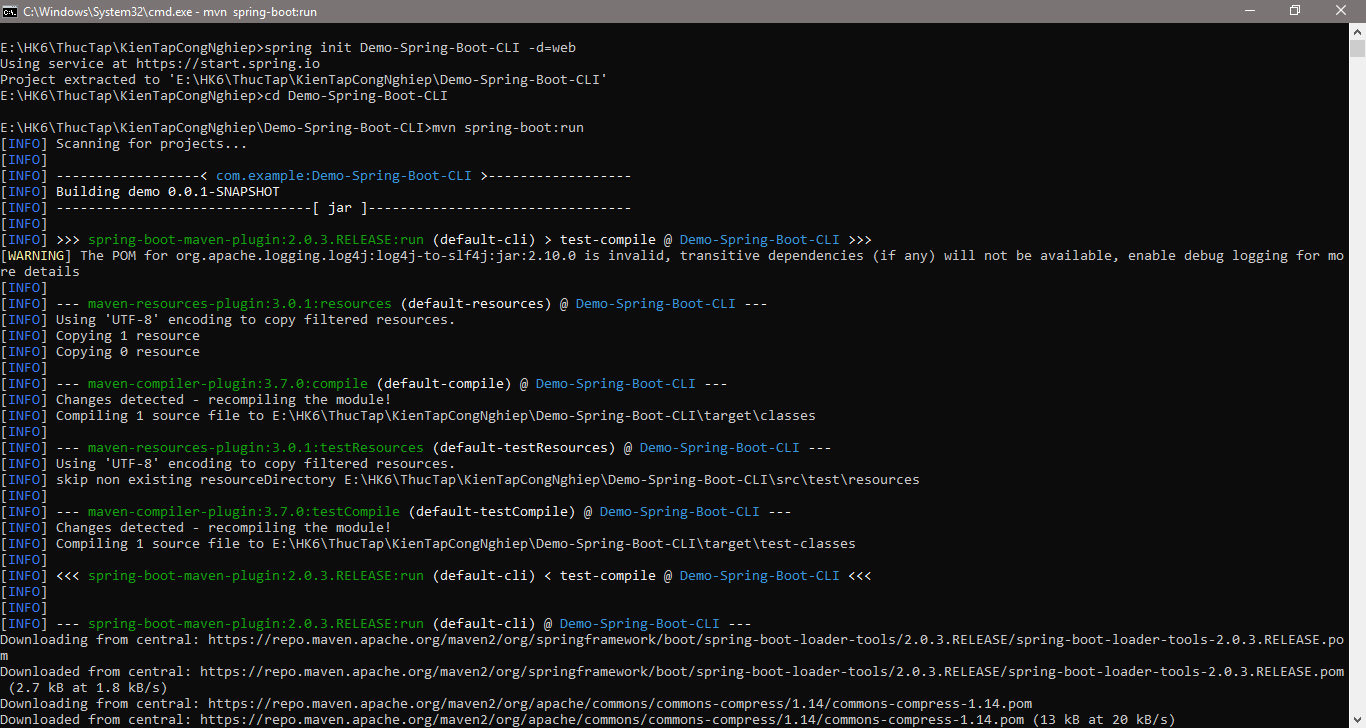


Hình 9 Project được tạo ra

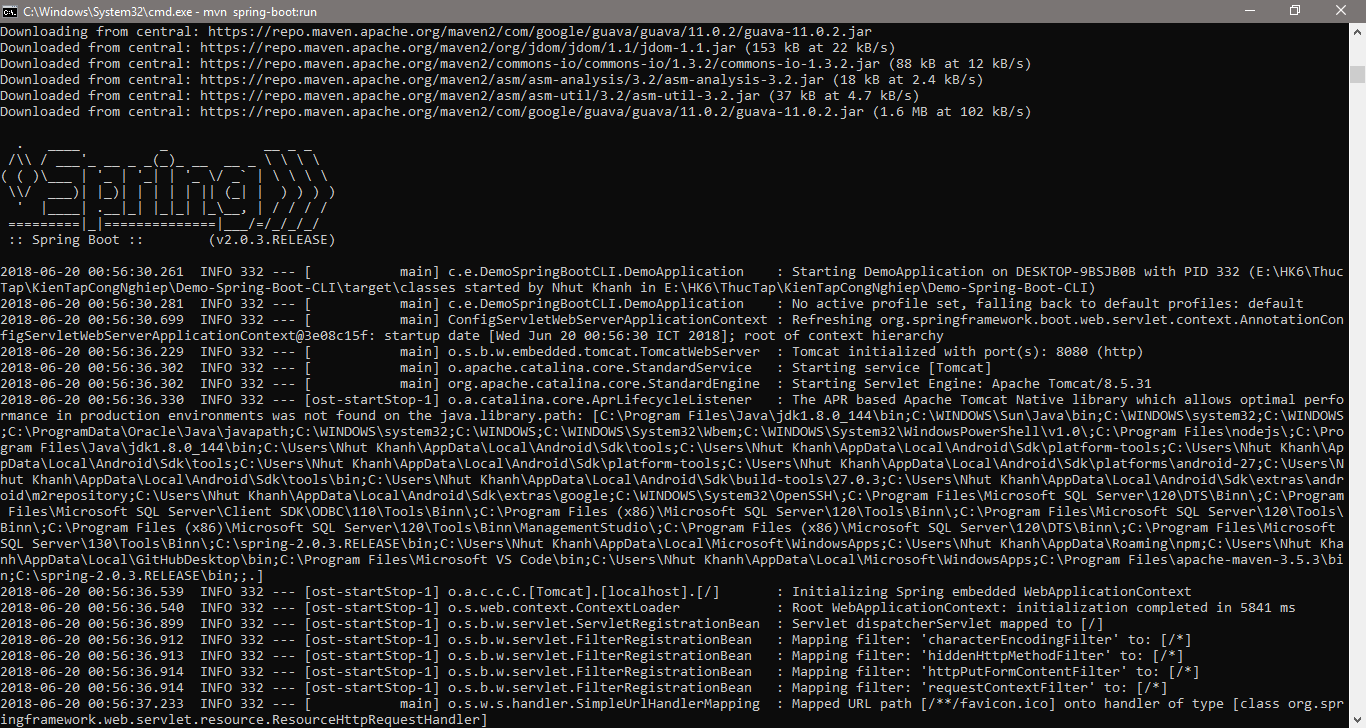


Hình 10 Chuyển đến thư mục làm việc của project

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



Hình 11 Chạy ứng dụng



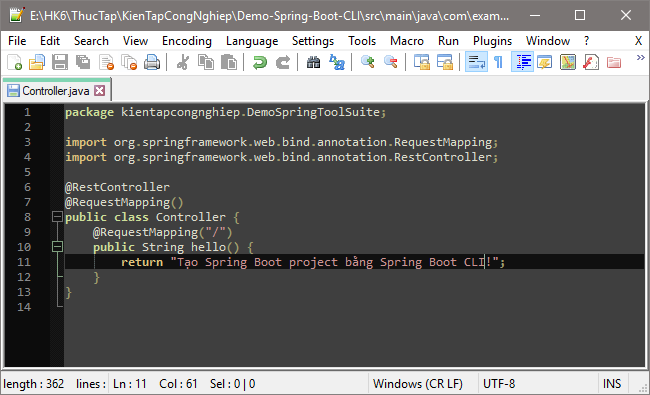
Hình 12 Chạy ứng dụng

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

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

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



Hình 13 Controller

https://lcdung.top/gioi-thieu-ve-spring-boot-spring-boot-la-gi/