1. **Giới thiệu sơ lược về Spring boot**
2. **Spring boot là gì?**

Spring Boot là một framework Java được sử dụng để xây dựng các ứng dụng và dịch vụ web dễ dàng và nhanh chóng. Nền tảng cung cấp các cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based.

Cách Spring Boot hoạt động nhằm tối ưu hóa quy trình phát triển ứng dụng Java. Điều này sẽ giúp nhà phát triển tập trung vào việc xây dựng tính năng chính của ứng dụng mà không cần phải lo lắng về cấu hình phức tạp.

1. **Nguồn gốc hình thành Spring boot từ đâu?**

Spring Boot xuất phát từ dự án Spring Framework, đây chính là một framework lập trình ứng dụng Java phổ biến. Spring Boot được phát triển bởi công ty Pivotal Software (nay thuộc VMware) với mục đích làm đơn giản hóa quá trình phát triển ứng dụng Spring. Công cụ sẽ giảm khối lượng công việc cần thiết để cấu hình và triển khai ứng dụng.

Bằng cách tự động hoá cấu hình và cung cấp các cấu hình mặc định thông minh, Spring Boot giúp nhà phát triển nâng cao sự tập trung vào việc xây dựng các tính năng chính của ứng dụng.

1. **Ba khả năng cốt lõi của Java Spring Boot**
2. Tự động cấu hình

Cấu hình opinionated approach (cho phép người dùng cấu hình nhưng sẽ trong một khuôn khổ nhất định)

1. Khả năng tạo các ứng dụng độc lập

Các tính năng này phối hợp với nhau giúp chúng ta có thể thiết lập ứng dụng dựa trên Spring khi chỉ cần thao tác cấu hình tối thiểu. Các ứng dụng Spring Boot cũng có thể được tối ưu hóa và chạy với Open Liberty.

1. Java và Spring framework

Mặc dù Java có thể dễ sử dụng và dễ học hơn các ngôn ngữ khác nhưng ngược lại khi build, debug và triển khai các ứng dụng Java thì lại rất phức tạp và ngày càng phức tạp hơn. Bởi vì số lượng biến sẽ tăng theo cấp số nhân khi chúng ta phát triển ứng dụng trong các lĩnh vực phổ biến, như stream nhạc online hay thanh toán trên thiết bị di động chẳng hạn. Bên cạnh đó, các developer khi viết ứng dụng kinh doanh cơ bản cùng lúc phải xử lý nhiều thư viện, plugin, log cũng như xử lý lỗi, tích hợp với các dịch vụ web và nhiều ngôn ngữ như C#, Java, HTML và các ngôn ngữ khác. Vì lẽ đó, người ta sẽ luôn có nhu cầu cực kỳ lớn đối với bất kỳ công cụ nào giúp họ phát triển ứng dụng Java một cách nhanh chóng và tiết kiệm tiền bạc nhất có thể.

**4. Các thành phần chính của Spring Boot:**

a) Spring Boot Starters:

* Là các dependency đóng gói sẵn, cung cấp các thư viện cần thiết cho một chức năng cụ thể.
* Ví dụ: spring-boot-starter-web cho ứng dụng web, spring-boot-starter-data-jpa cho JPA.

b) Spring Boot AutoConfigurator:

* Tự động cấu hình ứng dụng dựa trên các dependency có trong classpath.
* Giúp giảm thiểu cấu hình thủ công.

c) Spring Boot CLI:

* Command Line Interface để chạy và test ứng dụng Spring Boot nhanh chóng.

d) Actuator:

* Cung cấp các endpoint để giám sát và quản lý ứng dụng trong thời gian chạy.

e) Embedded Servers:

* Tích hợp sẵn các web server như Tomcat, Jetty, Undertow.

**5. Kiến trúc của Spring Boot**:

a) Lớp ứng dụng:

* Chứa logic nghiệp vụ và các controller.

b) Lớp AutoConfiguration:

* Tự động cấu hình các bean dựa trên classpath và cấu hình.

c) Lớp Spring Core Container:

* Quản lý vòng đời của các bean và dependency injection.

d) Lớp Spring Boot Starter:

* Cung cấp các dependency cần thiết cho các chức năng cụ thể.

e) Lớp Security (tùy chọn):

* Xử lý các vấn đề bảo mật như xác thực và phân quyền.

f) Lớp Data Access:

* Xử lý tương tác với cơ sở dữ liệu thông qua JPA hoặc JDBC.

g) Lớp Testing:

* Hỗ trợ kiểm thử đơn vị và tích hợp.
  1. **Những ưu điểm nổi bật của Spring boot**

Một số ưu điểm dưới đây đã giúp Spring Boot trở thành sự lựa chọn phổ biến cho việc xây dựng các ứng dụng và dịch vụ web trong hệ sinh thái Java.

* Tối ưu hóa quá trình phát triển: Spring Boot cung cấp cấu hình mặc định thông minh và tự động, giúp giảm thiểu việc cấu hình thủ công và tối ưu quá trình phát triển ứng dụng Java.
* Tích hợp tốt: Spring Boot tích hợp tốt với nhiều công nghệ và thư viện khác trong hệ sinh thái Spring Framework. Nền tảng cho phép hệ thống dễ dàng tích hợp các module và dịch vụ khác nhau mà không cần phải lo lắng về cấu hình phức tạp.
* Embedded server: Spring Boot đi kèm với các máy chủ nhúng như Tomcat, Jetty, hoặc Undertow. Đây là công cụ không thể thiếu trong việc triển khai ứng dụng một cách đơn giản mà không cần cấu hình thêm bất kỳ máy chủ nào khác.
* Tự động cấu hình: Spring Boot sử dụng cơ chế cấu hình tự động thông minh, cho phép ứng dụng tự cấu hình dựa trên các thư viện và module được sử dụng.
* Quản lý phụ thuộc: Spring Boot cung cấp các công cụ quản lý phụ thuộc mạnh mẽ như Maven hoặc Gradle, giúp quản lý các phụ thuộc của ứng dụng một cách hiệu quả.
* Monitoring và quản lý: Spring Boot cung cấp các công cụ hỗ trợ giám sát và quản lý ứng dụng dễ dàng, bao gồm Spring Boot Actuator cho việc giám sát và quản lý ứng dụng.
  1. **Nhược điểm của Spring Boot:**
* Kích thước ứng dụng lớn:
* Chi tiết: Spring Boot đóng gói nhiều thư viện và dependencies, dẫn đến kích thước JAR/WAR file lớn.
* Hệ quả: Tăng thời gian triển khai, chiếm nhiều không gian lưu trữ, và có thể làm chậm quá trình khởi động ứng dụng.
* Ví dụ: Một ứng dụng "Hello World" đơn giản có thể có kích thước lên đến 15-20MB, so với vài KB của một ứng dụng Java thuần.
* Học curve dốc:
* Chi tiết: Spring Boot sử dụng nhiều khái niệm và công nghệ phức tạp như Inversion of Control, Dependency Injection, AOP.
* Hệ quả: Người mới học có thể mất nhiều thời gian để nắm bắt đầy đủ các khái niệm và cách sử dụng hiệu quả.
* Ví dụ: Hiểu và sử dụng đúng các annotation như @Autowired, @Component, @Service có thể gây khó khăn cho người mới.
* Over-engineering cho các ứng dụng nhỏ:
* Chi tiết: Spring Boot cung cấp nhiều tính năng mà có thể không cần thiết cho các ứng dụng đơn giản.
* Hệ quả: Tăng độ phức tạp không cần thiết, làm chậm quá trình phát triển và triển khai cho các dự án nhỏ.
* Ví dụ: Sử dụng Spring Boot cho một ứng dụng CRUD đơn giản có thể là quá mức cần thiết.
* "Magic" configuration:
* Chi tiết: Auto-configuration của Spring Boot có thể khó hiểu và debug.
* Hệ quả: Có thể gây ra các vấn đề khó phát hiện khi cấu hình tự động không hoạt động như mong đợi.
* Ví dụ: Khi có conflict giữa các bean được auto-configured, việc xác định nguyên nhân có thể rất khó khăn.
* Phụ thuộc vào Spring ecosystem:
* Chi tiết: Spring Boot được thiết kế để hoạt động tốt nhất trong hệ sinh thái Spring.
* Hệ quả: Khó khăn khi tích hợp với các framework hoặc thư viện bên ngoài hệ sinh thái Spring.
* Ví dụ: Tích hợp một ORM khác ngoài Hibernate có thể gặp nhiều trở ngại.
* Vấn đề về phiên bản và tương thích:
* Chi tiết: Các phiên bản Spring Boot khác nhau có thể có sự khác biệt đáng kể.
* Hệ quả: Nâng cấp phiên bản có thể dẫn đến các vấn đề tương thích và yêu cầu thay đổi code đáng kể.
* Ví dụ: Chuyển từ Spring Boot 1.x lên 2.x đòi hỏi nhiều thay đổi trong cấu hình và code.
* Hiệu suất khởi động:
* Chi tiết: Quá trình auto-configuration và khởi tạo các bean có thể mất nhiều thời gian.
* Hệ quả: Thời gian khởi động ứng dụng có thể dài, đặc biệt là trong môi trường có tài nguyên hạn chế.
* Ví dụ: Một ứng dụng Spring Boot phức tạp có thể mất từ 10-30 giây để khởi động, so với vài giây của một ứng dụng Java thuần.
* Khó khăn trong việc tùy chỉnh sâu:
* Chi tiết: Mặc dù Spring Boot cung cấp nhiều tùy chọn cấu hình, việc thay đổi hành vi cốt lõi có thể phức tạp.
* Hệ quả: Khó khăn trong việc tùy chỉnh một số tính năng cụ thể hoặc thay thế các thành phần mặc định.
* Ví dụ: Thay đổi cách Spring Boot xử lý các exception hoặc tùy chỉnh quá trình khởi tạo ứng dụng có thể đòi hỏi nhiều công sức.
* Độ phức tạp của cấu hình:
* Chi tiết: Mặc dù Spring Boot giảm bớt cấu hình, nhưng đối với các ứng dụng phức tạp, cấu hình vẫn có thể trở nên rất phức tạp.
* Hệ quả: Khó khăn trong việc quản lý và hiểu rõ tất cả các cấu hình, đặc biệt trong các dự án lớn.
* Ví dụ: Quản lý nhiều profile, cấu hình cho nhiều môi trường khác nhau có thể trở nên phức tạp.
* Khả năng mở rộng hạn chế trong một số trường hợp:
* Chi tiết: Cấu trúc và cách tiếp cận của Spring Boot có thể không phù hợp cho tất cả các kịch bản mở rộng.
* Hệ quả: Có thể gặp khó khăn khi cần mở rộng ứng dụng theo các cách không chuẩn.
* Ví dụ: Triển khai một hệ thống phân tán phức tạp có thể đòi hỏi kiến trúc vượt ra ngoài mô hình tiêu chuẩn của Spring Boot.

1. **Tổng hợp các tính năng chính của Spring Boot**
2. **Spring Application**

Tiện ích Spring Application chịu trách nhiệm cho việc cấu hình và khởi động ứng dụng Spring Boot, cụ thể như:

A diagram of a shopping service

Description automatically generated

Spring Application tận dụng tính chất cấu hình tự động của Spring Boot để giảm thiểu sự phức tạp trong cấu hình ứng dụng. Hệ thống tự động quét các gói chứa các lớp và component của ứng dụng, giúp ứng dụng tự động phát hiện và cấu hình các bean cần thiết.

Tiện ích đem lại các cấu hình mặc định cho các ứng dụng web. Kèm theo đó là khả năng kích hoạt khả năng tích hợp với các máy chủ nhúng như Tomcat, Jetty, hoặc Undertow giúp việc triển khai ứng dụng web trở nên dễ dàng hơn.

Bên cạnh đó, Spring Application cho phép tải cấu hình ứng dụng từ nhiều nguồn khác nhau như các tập tin properties, biến môi trường, cấu hình trên command-line và nhiều nguồn cấu hình khác. Nhờ khả năng quản lý các sự kiện lúc khởi động và khởi tạo ứng dụng mà tiện ích đã giúp người phát triển có thể can thiệp vào quá trình khởi động của ứng dụng.

1. **Externalized Configuration**

Trong Spring Boot, tính năng này cho phép người dùng thực thi quá trình cấu hình ứng dụng bằng các tập tin cấu hình bên ngoài, biến môi trường mà không cần phải chỉnh sửa mã nguồn. Đây là thành phần nâng cao sự linh hoạt cho ứng dụng, cho phép cấu hình thay đổi mà không cần phải biên dịch lại mã nguồn. Một số tiện ích trên công cụ cần được kể đến chính là:

* External Configuration Files: Spring Boot cho phép bạn cấu hình ứng dụng bằng cách sử dụng các tệp tin cấu hình như YAML, Properties, hoặc JSON. Người dùng có thể chỉ định vị trí của các tệp tin này bằng cách sử dụng tham số dòng lệnh, biến môi trường hoặc vị trí mặc định.
* Setting Profiles: Bạn có thể sử dụng các profiles để cấu hình ứng dụng cho các môi trường khác nhau như development, testing, staging và production. Profiles cho phép bạn chỉ định cấu hình riêng biệt cho từng môi trường mà không cần phải thay đổi mã nguồn.
* Environment Variables: Spring Boot hỗ trợ cấu hình bằng biến môi trường, cho phép bạn điều chỉnh ứng dụng một cách dễ dàng mà không cần phải sửa đổi tập tin cấu hình.

1. **Profiles**

Profiles trong Spring Boot cho phép bạn định nghĩa và quản lý cấu hình ứng dụng dựa trên các môi trường khác nhau. Chẳng hạn như development, testing, staging và production mà không cần phải thay đổi mã nguồn.

Yếu tố này đã giúp ứng dụng hoàn toàn có thể chuyển đổi việc triển khai ở các môi trường khác nhau. Các tính năng chính của Profiles trong Spring Boot bao gồm:

* Định nghĩa Profiles: Bằng cách sử dụng tiền tố "application-" kèm theo tên profile, bạn có thể định nghĩa các tệp cấu hình riêng biệt cho từng profile. Ví dụ: "application-dev.yml" cho môi trường development, "application-prod.yml" cho môi trường production và cũng có thể sử dụng các properties tương ứng.
* Áp dụng Profiles: Bạn có thể chỉ định profile được sử dụng thông qua các cấu hình hoặc tham số dòng lệnh khi khởi chạy ứng dụng. Spring Boot sẽ tải cấu hình từ tập tin phù hợp với profile được chỉ định.
* Overriding: Profiles cho phép override các cấu hình mặc định được chỉ định trong tệp cấu hình chung. Điều này giúp bạn tuỳ chỉnh cấu hình mà không cần phải sửa đổi tập tin gốc.

1. **Logging**

Tính năng Logging cung cấp khả năng quản lý và ghi nhật ký hoạt động của ứng dụng một cách linh hoạt. Những hoạt động chính thường được thực hiện bao gồm:

A diagram of a computer network

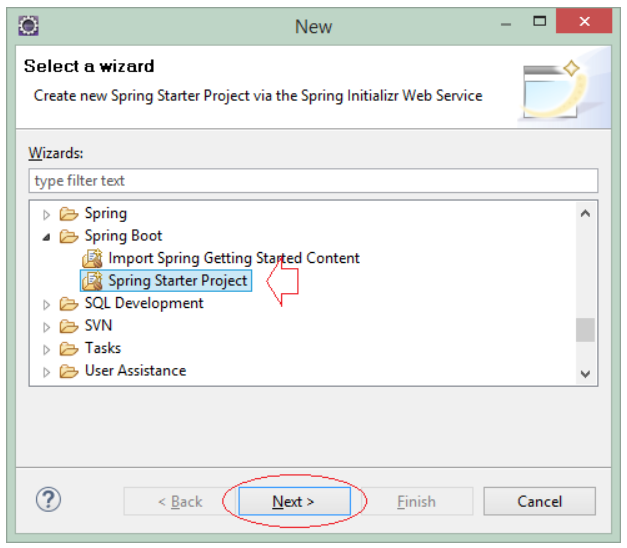
Description automatically generated

* Integration with Common Logging Frameworks: Spring Boot tích hợp mặc định với các framework logging thông dụng như Logback, Log4j2 và JDK (Java Util Logging). Nền tảng cho phép bạn lựa chọn framework phù hợp với nhu cầu của ứng dụng mà không cần phải cấu hình thêm.
* Default Configuration: Spring Boot cung cấp cấu hình mặc định cho logging, giúp bạn bắt đầu sử dụng logging mà không cần phải thực hiện các cấu hình phức tạp.
* Flexible Configuration: Bạn có thể tùy chỉnh cấu hình logging thông qua tệp cấu hình properties hoặc YAML, bao gồm cấu hình mức độ log, định dạng log, v.v.
* Log Levels: Spring Boot hỗ trợ các mức độ log thông thường như DEBUG, INFO, WARN, ERROR để giúp người dùng quản lý việc ghi log theo độ quan trọng.
* Logging Output: Bạn có thể chỉ định định dạng output cho việc ghi log như console, file hoặc cơ sở dữ liệu.

1. **Tạo Spring Boot Project**
2. **Tạo project**

Trên **Eclipse** tạo một project sử dụng **Spring Tool Suite (STS)**:

* **File/New/Other...**



* **Name:** HelloSpringBoot
* **Group:** org.o7planning
* **Artifact:** HelloSpringBoot
* **Package:** org.o7planning.hellospringboot

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Project đã được tạo ra, dưới đây là cấu trúc mặc định của Project:

A screenshot of a computer program

Description automatically generated

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/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<**modelVersion**>4.0.0</**modelVersion**>

<**groupId**>org.o7planning</**groupId**>

<**artifactId**>HelloSpringBoot</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**packaging**>jar</**packaging**>

<**name**>HelloSpringBoot</**name**>

<**description**>Hello Spring Boot</**description**>

<**parent**>

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

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

<**version**>2.0.0.RELEASE</**version**>

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

</**parent**>

<**properties**>

<**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>

<**project.reporting.outputEncoding**>UTF-8</**project.reporting.outputEncoding**>

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

</**properties**>

<**dependencies**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-test</**artifactId**>

<**scope**>test</**scope**>

</**dependency**>

</**dependencies**>

<**build**>

<**plugins**>

<**plugin**>

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

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

</**plugin**>

</**plugins**>

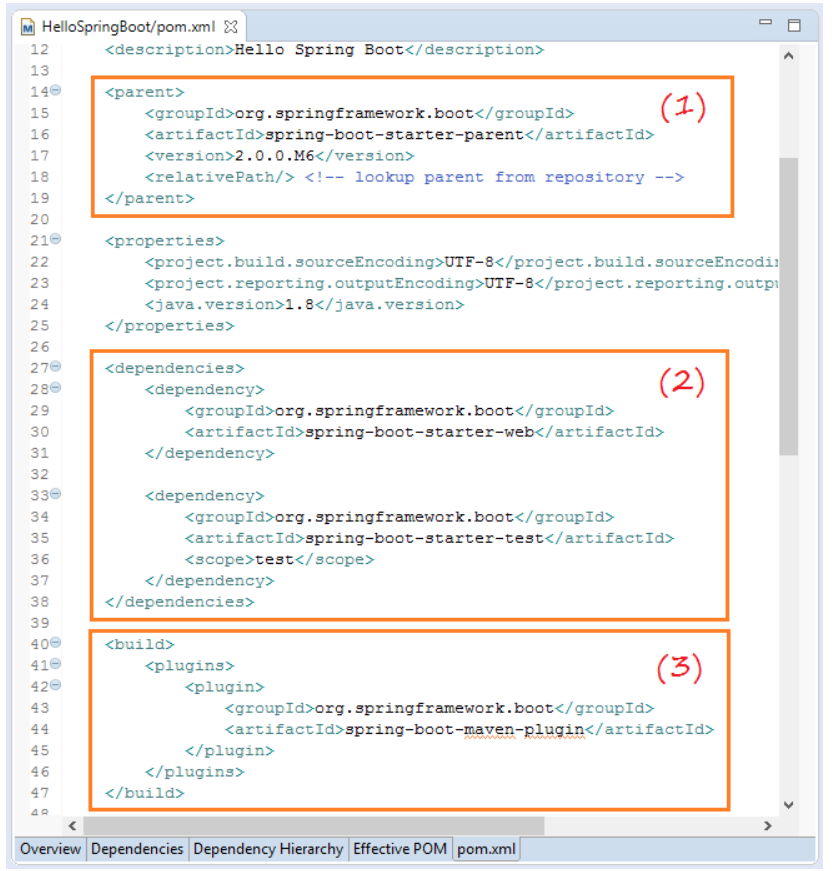
</**build**>

</**project**>

1. **Một vài giải thích**

Ở bước trên bạn vừa tạo ra một **Spring Boot Project**, hãy xem nội dung ***pom.xml***:

* Có 3 điểm cần chú ý trong **pom.xml** (1),(2), (3) như hình minh họa dưới đây, **Spring Boot** giúp bạn đơn giản hơn trong việc khai báo các thư viện Spring.



* **spring-boot-starter-parent**

spring-boot-starter-parent là một project sẵn có trong **Spring Boot**. Các thư viện phụ thuộc 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 của bạn.

* <https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-parent>

\*\* spring-boot-starter-parent \*\*

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-parent -->

<**parent**>

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

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

<**version**>2.0.0.RELEASE</**version**>

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

</**parent**>

* **spring-boot-starter-web**

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 phụ thuộc **spring-boot-starter-web**.

<**dependencies**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-test</**artifactId**>

<**scope**>test</**scope**>

</**dependency**>

</**dependencies**>

* + **spring-boot-maven-plugin**

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)

<**plugins**>

<**plugin**>

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

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

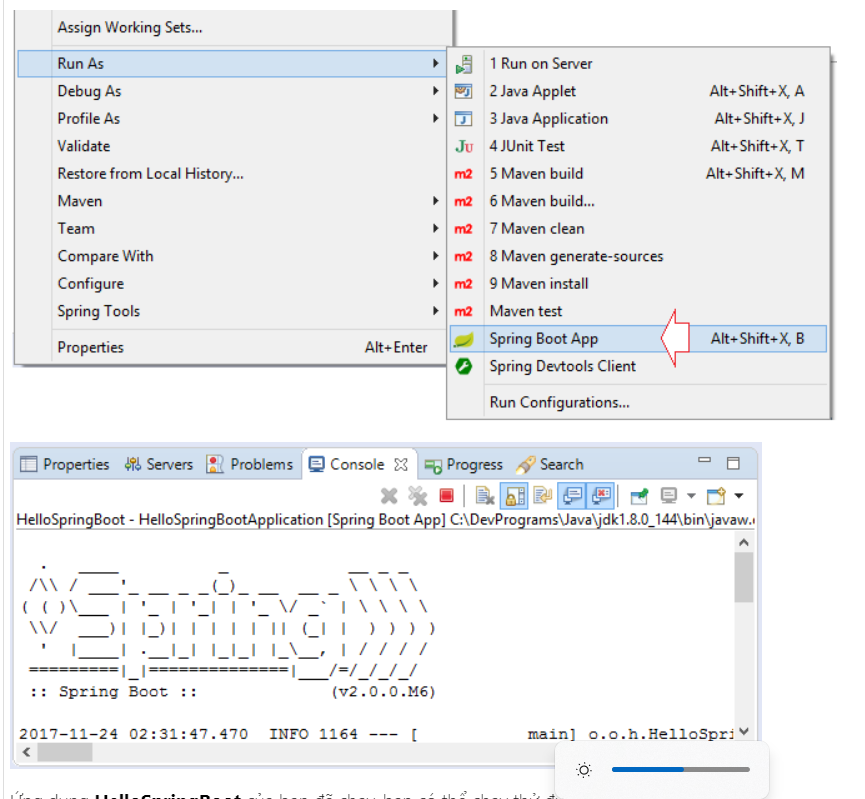
</**plugin**>

<!-- ... -->

</**plugins**>

1. **Chạy Spring Boot project**

Ở bước trên bạn đã tạo ra **HelloSpringBoot** project bởi **Eclipse & Spring Tool Suite**, và chưa có bất kỳ thay đổi nào. Tất cả mọi thứ được tạo ra mặc định. Bây giờ bạn cần một vài bước để chạy ứng dụng.



Ứng dung **HelloSpringBoot** của bạn đã chạy, bạn có thể chạy thử đường dẫn:

A screenshot of a computer error page

Description automatically generated

Có thông báo lỗi **"404 Not found"** khi bạn chạy đường dẫn trên, tuy nhiên không có vấn đề gì nghiêm trọng bởi vì project của bạn chưa có bất kỳ trang nào.

Tạo file **style.css** trong thư mục **static**:

A screenshot of a computer program

Description automatically generated

style.css

**h1** {

**color**: red;

}

Và chạy lại ứng dụng với đường dẫn:A screenshot of a computer

Description automatically generated

1. **Spring Boot chạy thế nào?**

Khi bạn tạo một **"Spring Boot Web Appp"**, có 1 class được tạo ra (generate) bởi công cụ **"Spring Tool Suite"**:

* HelloSpringBootApplication

A screenshot of a computer

Description automatically generated

Ứng dụng của bạn được bắt đầu bởi việc thực thi class HelloSpringBootApplication.

Class này được chú thích bởi **@SpringBootApplication**.

HelloSpringBootApplication.java

**package** org.o7planning.hellospringboot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** **HelloSpringBootApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(HelloSpringBootApplication.class, args);

}

}

Chú thích **@SpringBootApplication** là tương đương với việc sử dụng **@Configuration**, **@EnableAutoConfiguration** và **@ComponentScan** với các thuộc tính mặc định của chúng.

Như vậy **@SpringBootApplication** giúp bạn tự động cấu hình **Spring**, và tự động quét (Scan) toàn bộ project để tìm ra các thành phần Spring (Controller, Bean, Service,...)

A screenshot of a computer

Description automatically generated