* 1. Spring
* Spring phổ biến trong lập trình Java

Spring ra đời với nhiệm vụ của nó là thay thế các công nghệ Java dùng cho doanh nghiệp, vốn khá cồng kềnh và nặng nề như **EJB**. Spring đề xuất một giải pháp nhẹ nhàng, tinh gọn hơn so với EJB bằng cách bơm cho **POJO** sức mạnh vốn chỉ có ở EJB và các anh em của nó. Dần dần EJB và J2EE cũng cải tiến theo hướng của Spring: EJB cũng sử dụng POJO, triển khai ý tưởng DI(Dependency Injection) và AOP(Aspect-Oriented Programming).

Dù J2EE(hay JEE) có thể đuổi kịp được Spring, Spring vẫn không ngừng phát triển và vươn xa đến những vùng đất J2EE mới chỉ chập chững hoặc chưa từng đặt chân vào như: phát triển trên nền mobile, tích hợp social API, NoSQL database, cloud computing, big data,…

Xây dựng Web Application với Spring: Spring MVC là một framework được xây dựng trên nền của Spring. Nó vẫn bảo tồn được các tinh hoa của Spring và là một công cụ đắc lực hỗ trợ bạn trong vấn đề này. Dựa trên mô hình MVC kinh điển, Spring MVC sẽ giúp bạn xây dựng các web application linh hoạt và mạnh mẽ.

Tiếp cận với RESTful Service:

Spring Security và vấn đề bảo mật:

* Phát triển một ứng dụng Spring

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

Một cấu trúc Maven project hoặc Gradle project, bao gồm các dependency bắt buộc.

Tệp file xml (hoặc triển khai WebApplicationInitializer) khai báo DispatcherServlet của Spring.

Cấu hình Spring để cho phép sử Spring MVC.

Một class Controller sẽ gửi các phản hồi đến HTTP với yêu cầu “Hello World”.

Một máy chủ ứng dụng web chẳng hạn như Tomcat để triển khai ứng dụng.

* 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 (boilerplate Code), 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

Giải thích code

Quản lí dependencies dễ dàng hơn

• Đầu tiên sử dụng một số phụ thuộc có tên là spring-boot-starter- \*.

• Khi thêm phụ thuộc springboot-starter-web theo mặc định, nó sẽ thêm tất cả các thư viện thường được sử dụng trong khi phát triển các ứng dụng Spring MVC như spring-webmvc, jackson-json, validation-api và tomcat.

• Khi thêm phụ thuộc spring-boot-starter-data-jpa. Điều này kéo tất cả các phụ thuộc spring-datajpa và cũng thêm các thư viện Hibernate vì phần lớn các ứng dụng sử dụng Hibernate như là thực thi JPA.

Cấu hình tự động

• Không chỉ spring-boot-starter-web thêm tất cả các thư viện này mà còn cấu hình

các bean như DispatcherServlet, ResourceHandlers, MessageSource etc beans với các giá trị mặc định hợp lý.

• Khi thêm spring-boot-starter-thymeleaf nó không chỉ thêm dependencies vào thư viện thymeleaf mà còn cấu hình các beans ThymeleafViewResolver một cách tự động.

• Không cần định nghĩa bất kỳ nguồn dữ liệu nào của DataSource, EntityManagerFactory, TransactionManager,… nhưng chúng sẽ tự động được tạo. Nếu chúng ta có bất kỳ trình điều khiển cơ sở dữ liệu trong bộ nhớ như H2 hoặc HSQL trong classpath của chúng ta thì SpringBoot sẽ tự động tạo ra một DataSource trong bộ nhớ và sau đó đăng ký các bean EntityManagerFactory, TransactionManager tự động với các giá trị mặc định hợp lý. Do đang sử dụng MySQL, vì vậy cần cung cấp kết nối MySQL một cách chi tiết, rõ ràng. Các kết nối MySQL trong tệp application.properties và SpringBoot tạo một DataSource bằng các thuộc tính này.

Hỗ trợ Embedded Servlet Container

• Điều quan trọng là chúng ta chỉ cần tạo ra một lớp Java đơn giản dùng chú thích @SpringApplication có một phương thức main và chạy ứng dụng truy cập nó tại http://localhost:8080/.

•Thêm spring-boot-starter-web để bắt spring-boot-starter-tomcat tự động và khi chúng tôi chạy phương thức main (), nó bắt đầu tomcat như một vùng chứa nhúng để chúng ta không phải triển khai ứng dụng của mình trên bất kỳ máy chủ tomcat được cài đặt bên ngoài. Bằng cách này bạn có quan sát rằng loại packaging trong pom.xml là ‘jar’ chứ không phải ‘ửa’. Nhưng muốn sử dụng máy chủ Jetty thay vì tomcat thì sao? Đơn giản, loại trừ spring-bootstarter-tomcat từ spring-boot-starter-web và include spring-boot-starter-jetty vào.

**Tổng quan về Spring Boot**

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

Springframework đã tồn tại hơn một thập kỷ nay, cũng giống như bao framework khác, nó giúp rất nhiều trong việc xây dựng các ứng dụng doanh nghiệp ( Enterprise application) 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 đã biết, trong ứng dụng Spring để sử dụng các tính năng của nó chúng ta phải config rất nhiều bằng cách viết những file cấu hình khá là phức tạp, nào là web.xml để config 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 config bằng java, chúng ta vẫn chẳng thể tránh những config phiền toái ấy.

Ví dụ như việc config transaction, các filter hay quản lý các dependency chúng ta phải config trong những file xml hoặc trong code java. Việc config như thế đòi hỏi khá nhiều thời gian cũng công sức, thay vì phải ngồi config 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 config 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 config 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 config 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 config (automatic configuration): Spring boot có thể tự động config những thứ cơ bản mà hầu hết các ứng dụng Spring đều phải có.

- Khởi xướng 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 (tạm dịch là truyền động :D): Nó cho phép bạn 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 config 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 config 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 config 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 config 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 config trong mỗi lần tạo project?

Spring boot có thể tự động config những thứ tương tự vậy, nếu Spring boot phát hiện trong classpath của bạn có lib H2 thì nó sẽ tự config H2 vào ứng dụng của bạn. Nếu JdbcTemplate có trong classpath, nó sẽ config 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 config 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 config (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 config 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 config 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 config trong Spring application context

■ Những gì đã được Spring tự động config

■ 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

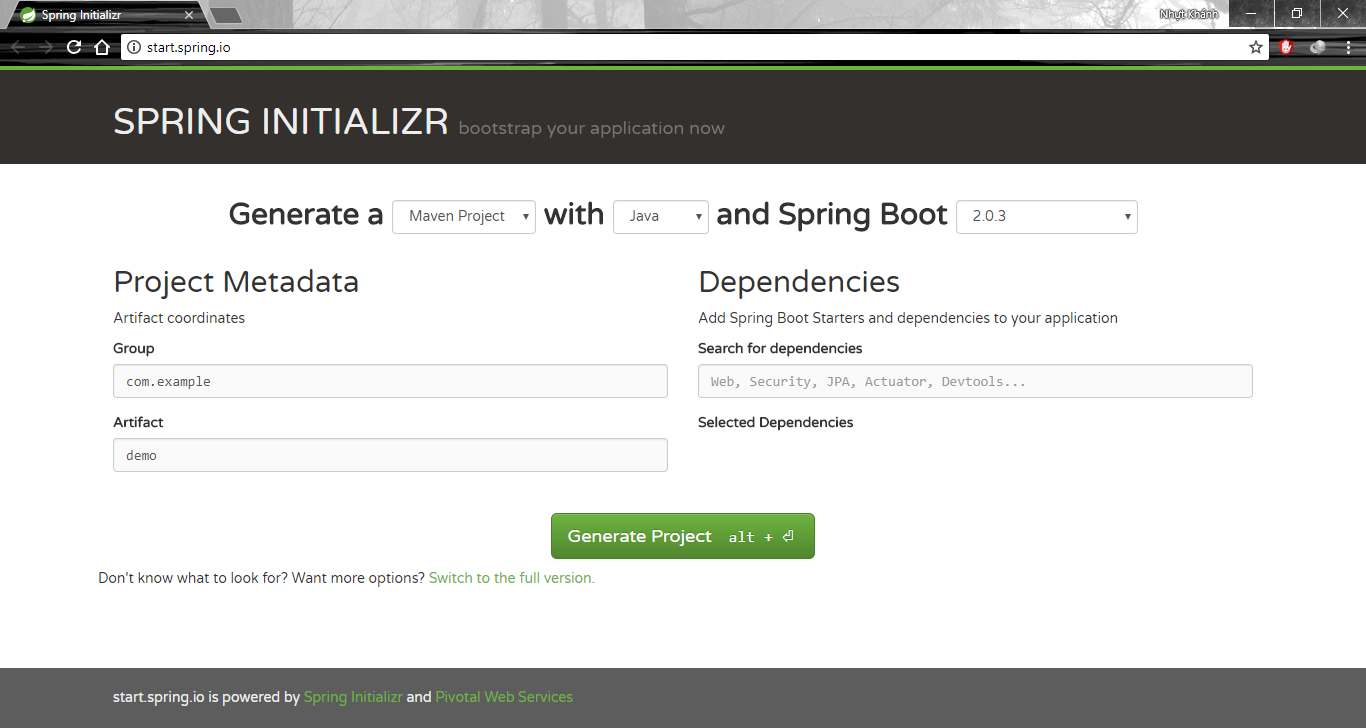
Actuator trình bày thông tin này bằng hai cách là thông qua web endpoints hoặc thông qua shell interface.

**Cách hoạt động của String Boot**

**Cách tạo mới một Spring Boot project**

**Spring Initializr Web**

Bước 1: Truy cập vào trang web: **Spring Initializr Web**



Chọn Maven Project

Ngôn ngữ Java

Phiên bản Spring Boot 2.0.3

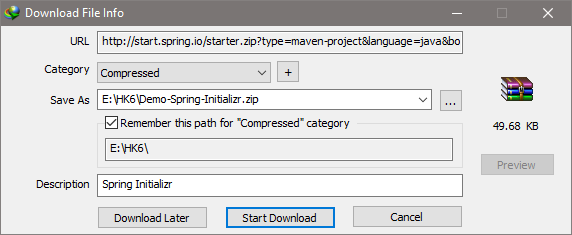
Group: kientapcongnghiep

Artifact: Demo-Spring-Initializr

Search for dependencies: Web

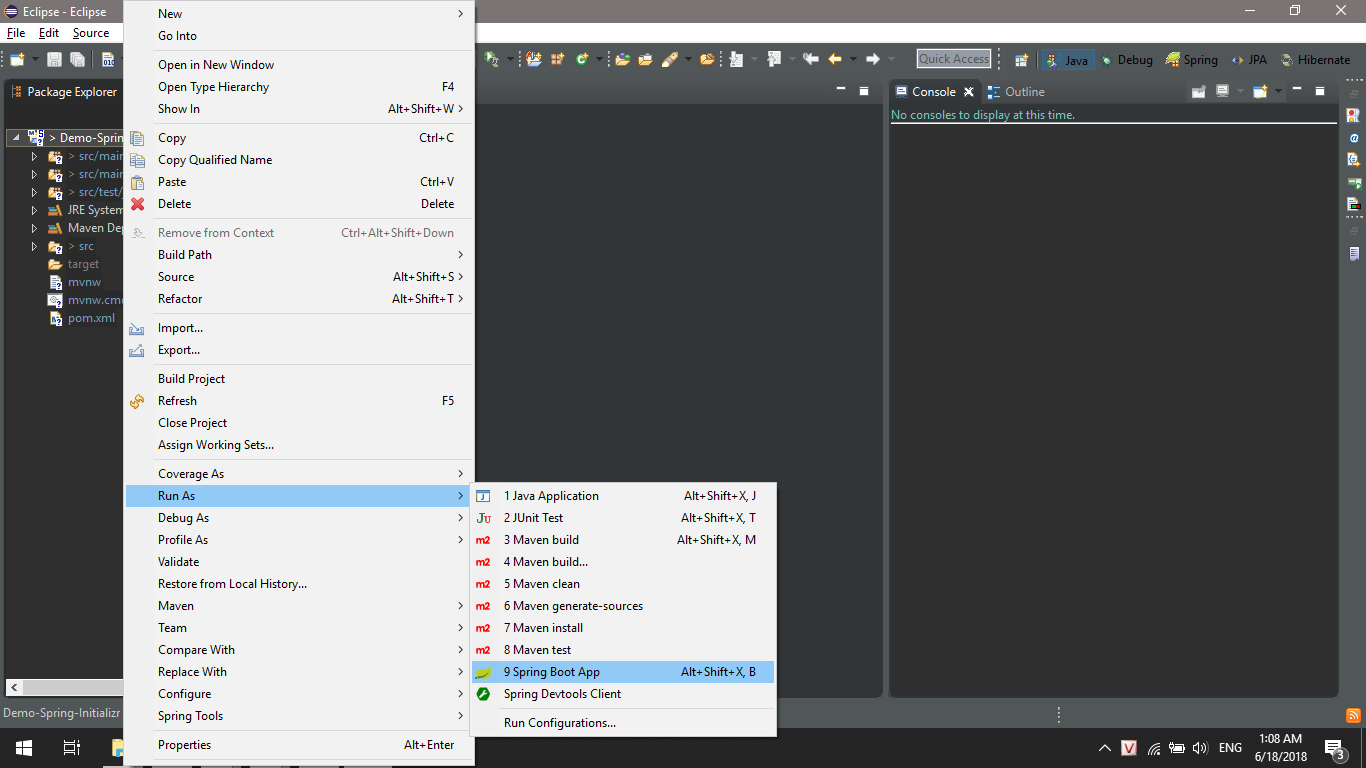
Chọn Generate Project

Một file vừa được tạo và tải xuống



Mở Eclipse giải nén và add project vừa được tạo

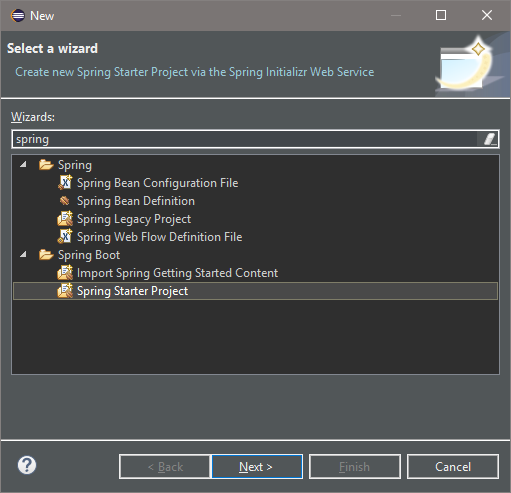
Click chuột phải vào Project vừa add chọn Run as chọn Spring Boot Application



Như vậy một Spring Boot Project đã được tạo.

**Dùng Spring Tool Suite**

Tạo một Spring Starter Project



Name: Demo-Spring-Tool-Suite

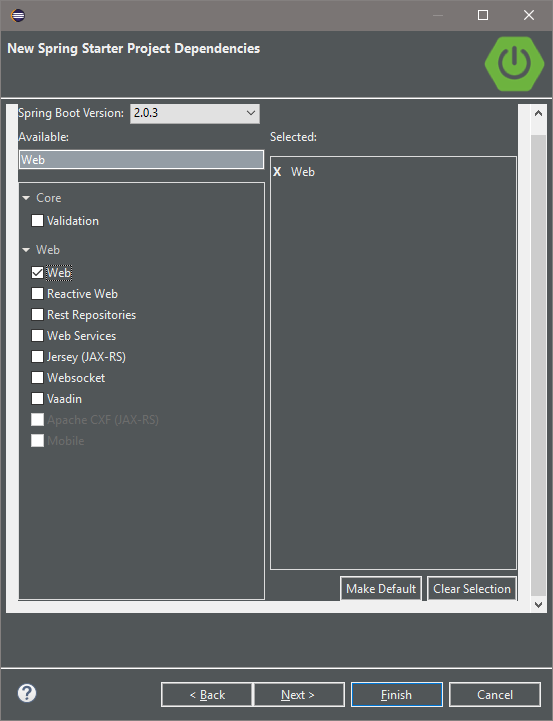
Location: Chọn nơi lưu Project

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