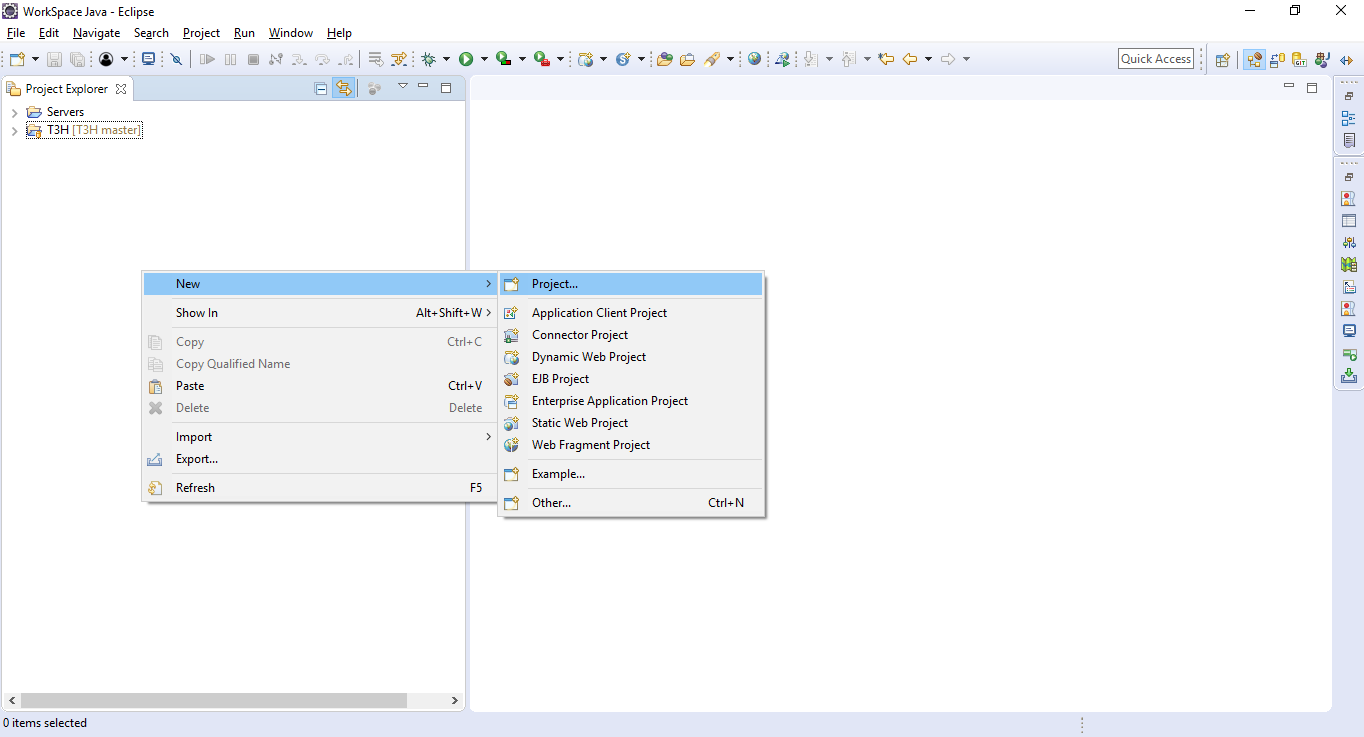
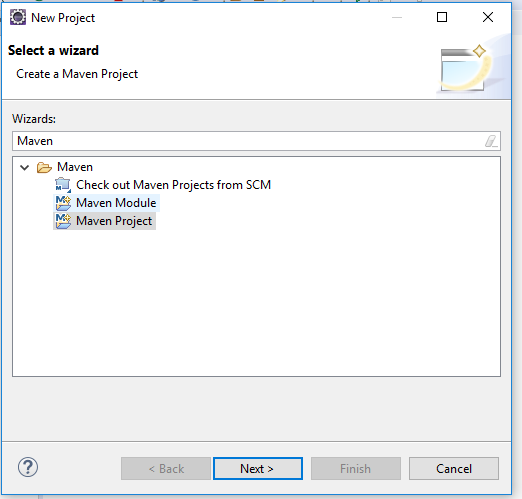
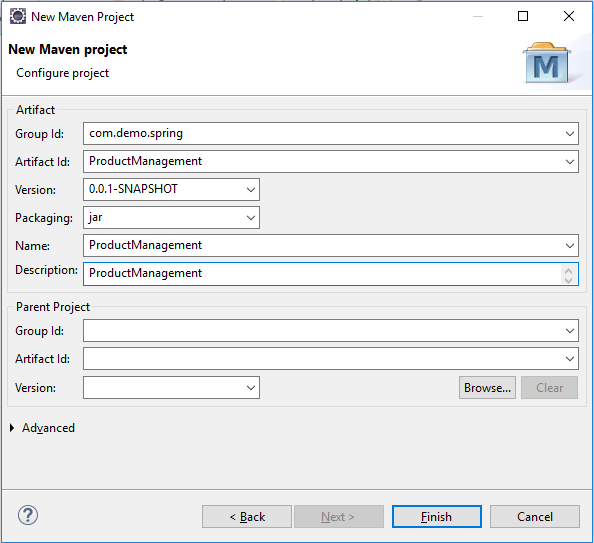
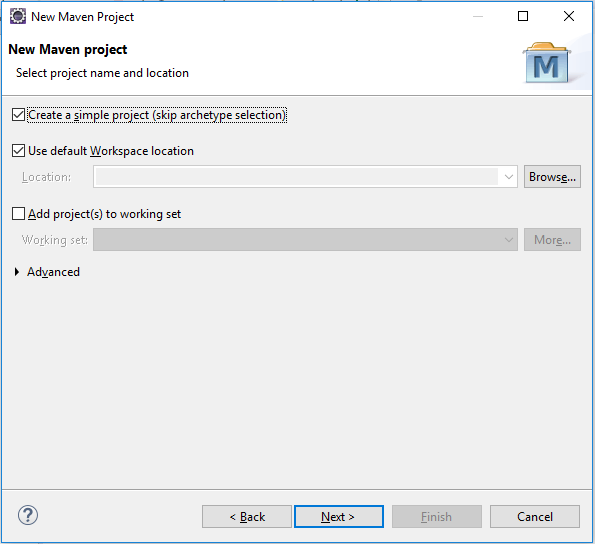
# Hello Worl với Spring Boot

## Tạo Project maven

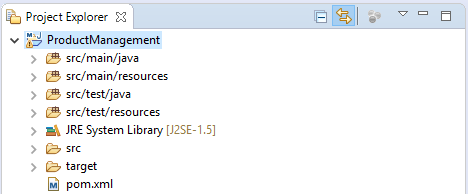
### Step1: Tạo Maven Project







Điền các thong số và chờ đợi Eclipse tạo project



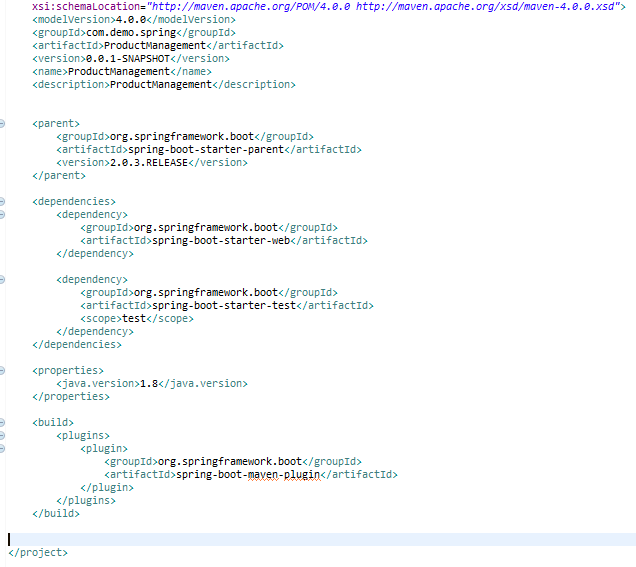
Kết quả Step 1 là hình trên. Nếu không thành công các bạn có thể tạo bằng tay thêm thư mục hoặc buil lại

### Fix lỗi Step1

* Cài maven riêng
* Chạy lệnh mvn eclipse:eclipse với thư mục Project
* Tạo Folder và add Resources

### Step2: Add thư viện Spring cho project

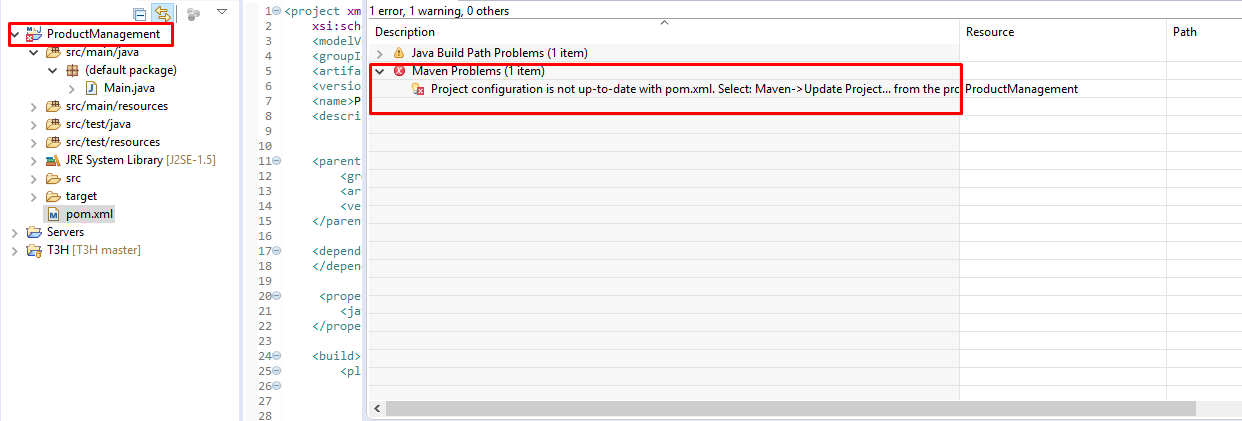
Tới file pom.xml và add thư viện và các thông số



Nội dung add mới

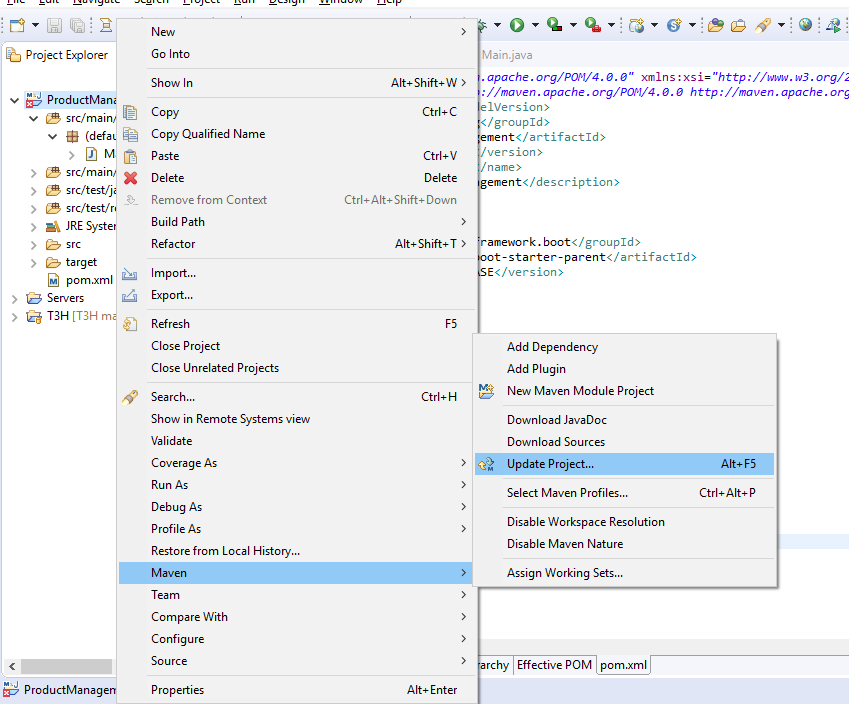
|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.3.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <properties>  <java.version>1.8</java.version>  </properties>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

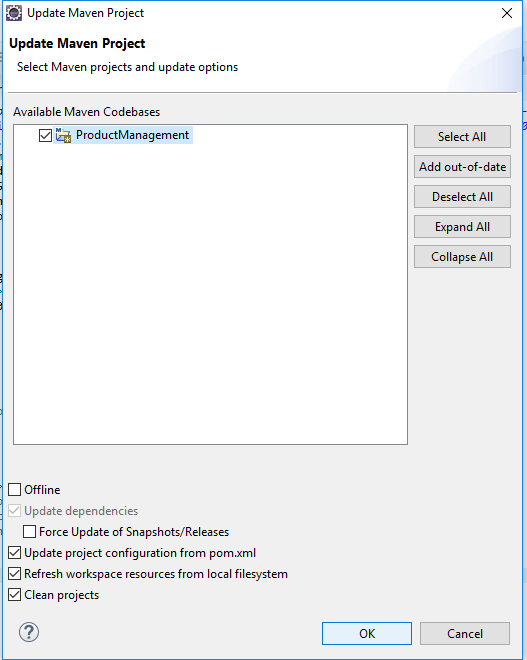
Hãy lưu lại và Build project lại



Khi bị lỗi project bạn có thể thấy gợi ý của Eclipse như trên

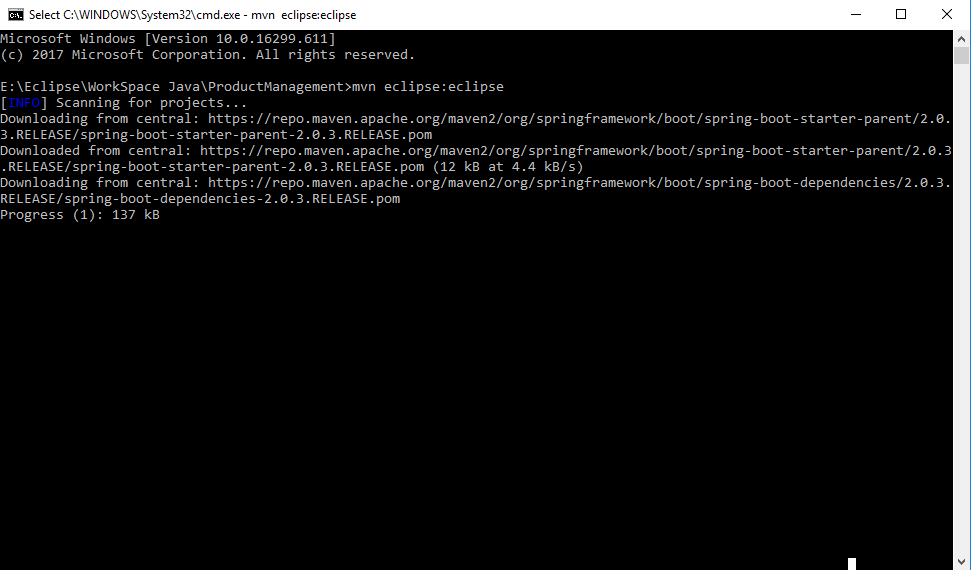
Chuột phải Project và Chọn như hình

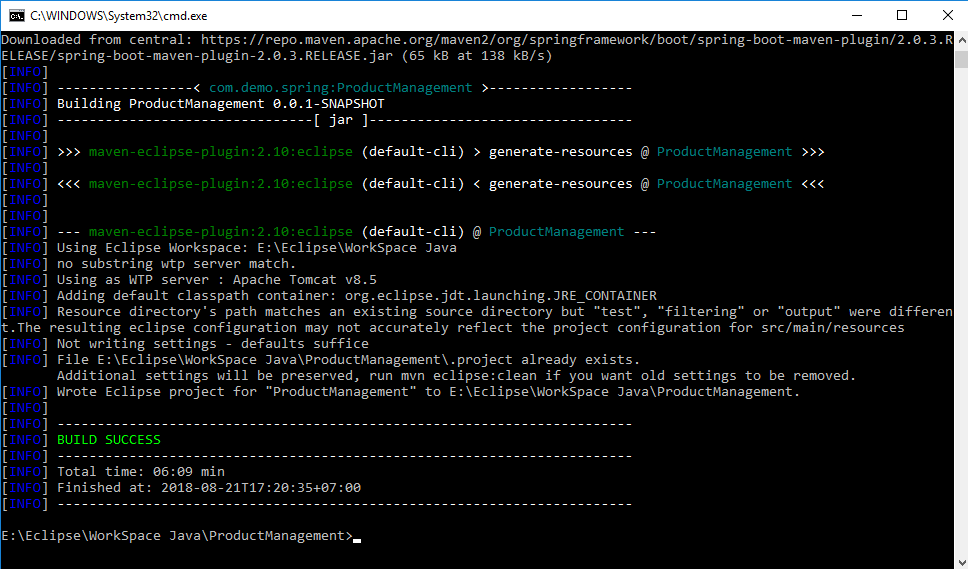




### Fix lỗi Step 2

* Có thể trỏ tới thư mục chứa project và chạy lệnh maven với maven cài riêng





Nếu không cài maven riêng hãy cố gắng buil lại project khi Build project thành công

Và update lại thư viện của project như trên Maven > Update project

## Tạo Demo với Project Spring Boot

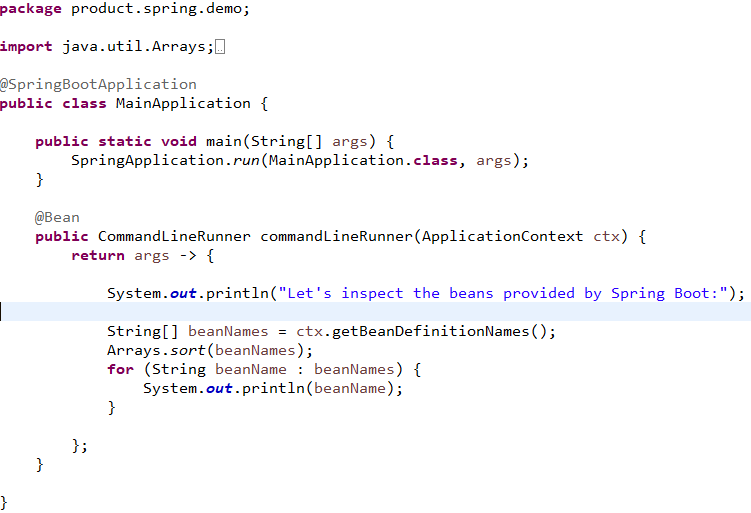
### Link tham khảo

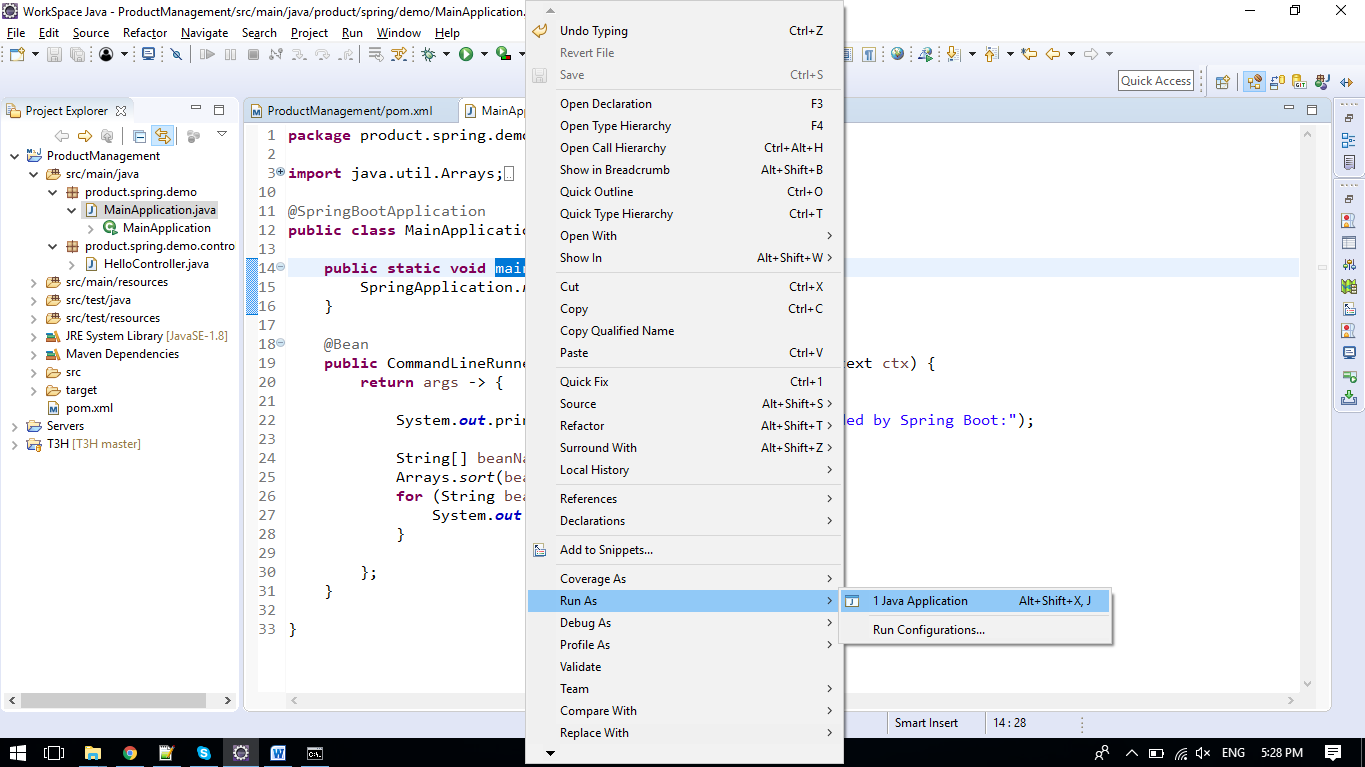
##### https://spring.io/guides/gs/spring-boot/

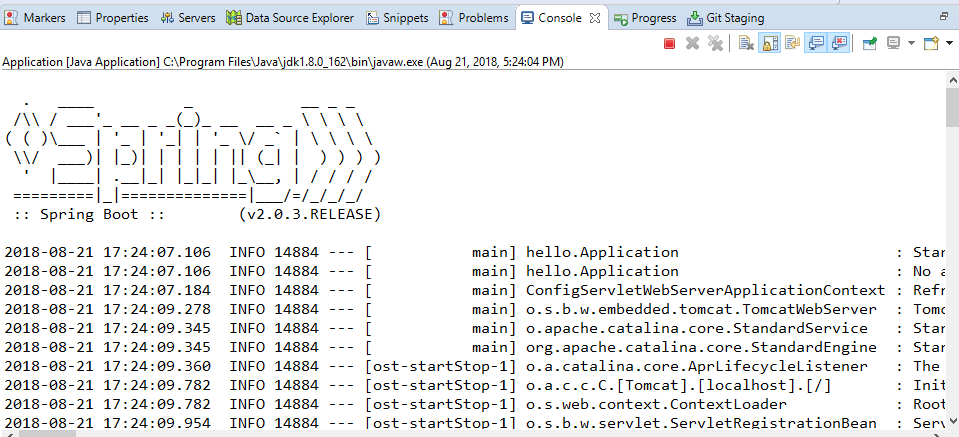
### Step 3 Tạo Main Application

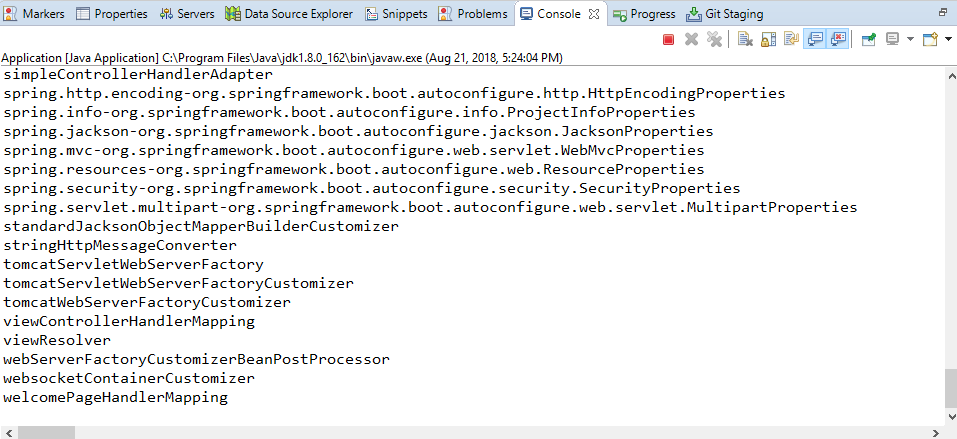
|  |
| --- |
| package product.spring.demo;  import java.util.Arrays;  import org.springframework.boot.CommandLineRunner;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.context.ApplicationContext;  import org.springframework.context.annotation.Bean;  @SpringBootApplication  public class MainApplication {  public static void main(String[] args) {  SpringApplication.run(MainApplication.class, args);  }  @Bean  public CommandLineRunner commandLineRunner(ApplicationContext ctx) {  return args -> {  System.out.println("Let's inspect the beans provided by Spring Boot:");  String[] beanNames = ctx.getBeanDefinitionNames();  Arrays.sort(beanNames);  for (String beanName : beanNames) {  System.out.println(beanName);  }  };  }  } |

Hãy tạo controller và chạy thử kết quả









Hình kết quả là các bean tạo sẵn của Spring Boot nói riêng và Spring nói chung

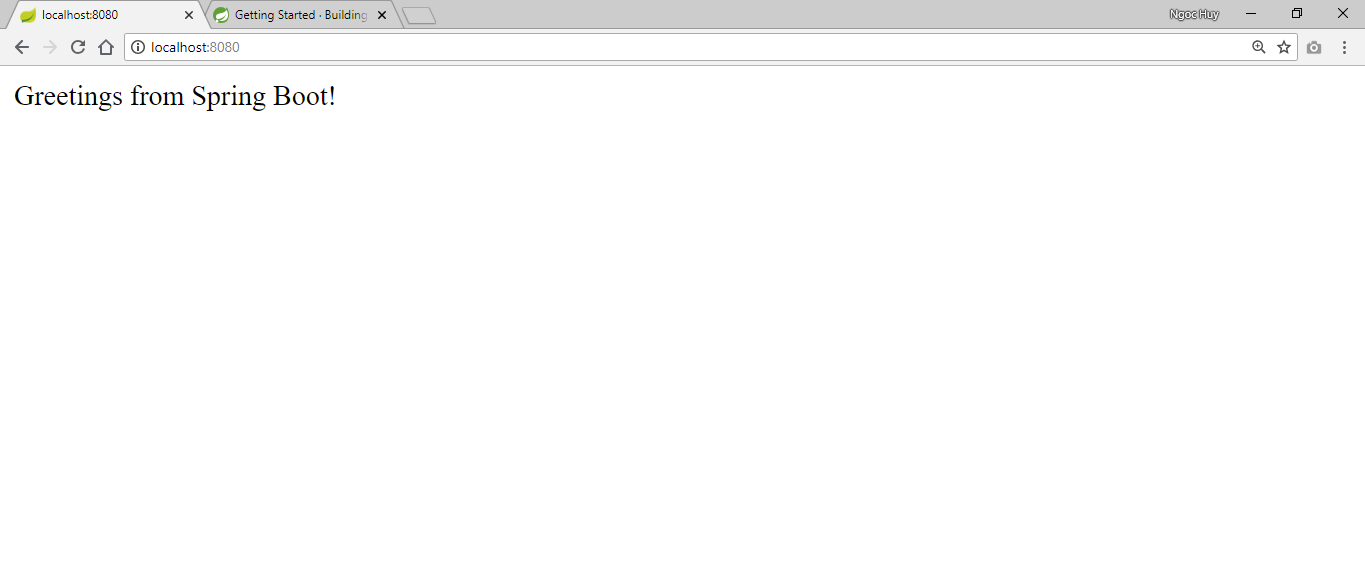
### Fix lỗi trong Step3

Step này không có lỗi nếu như bước 2 đã thành công, hãy build lại project.

### Step 4: Tạo Controller và chạy web

|  |
| --- |
| package product.spring.demo.controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  @RestController  public class HelloController {  @RequestMapping("/")  public String index() {  return "Greetings from Spring Boot!";  }  } |

Khi tạo controller này hãy chạy lại Main Application và kiểm tra kết quả với đường dẫn trình duyệt web



Kết quả chạy: localhost:8080

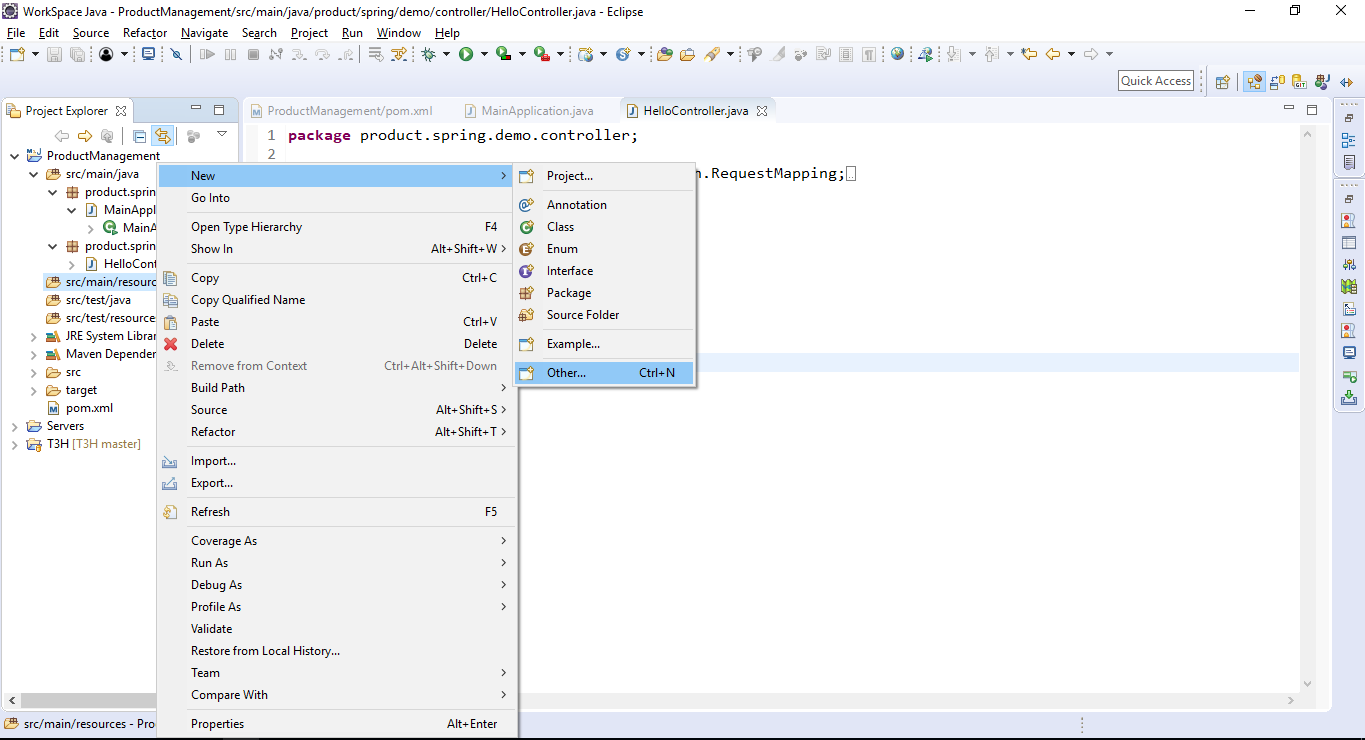
### Lưu ý Step4

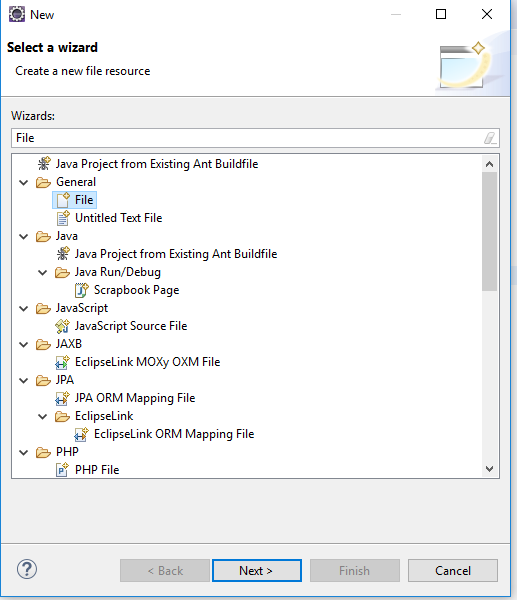
* Mặc định Spring Boot chạy với Tomcat là 8080
* Tomcat được nhúng trong Project, Khi Spring Boot chạy nó sẽ khởi động Tomcat này

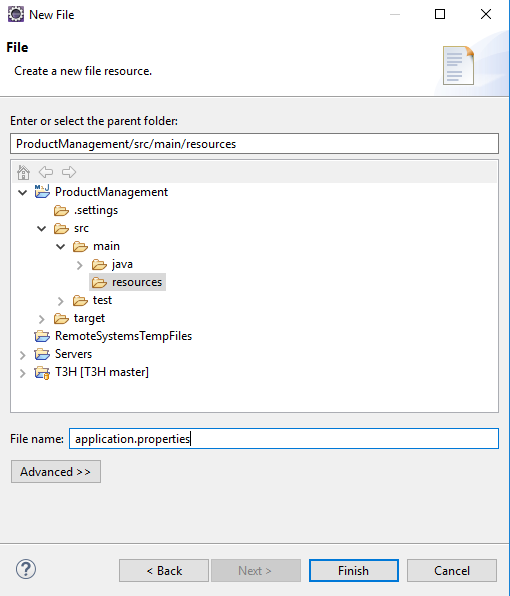
### Step 5: Tạo file cấu hình ứng dụng application.properties

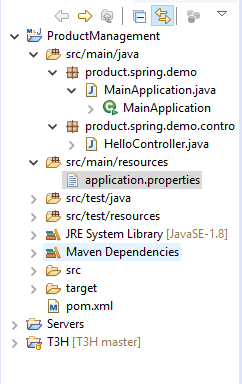
Cách hay nhất trong Spring Boot là sử dụng application.properties để config ứng dụng

Hãy làm theo hình sau. Chuột phải vào 



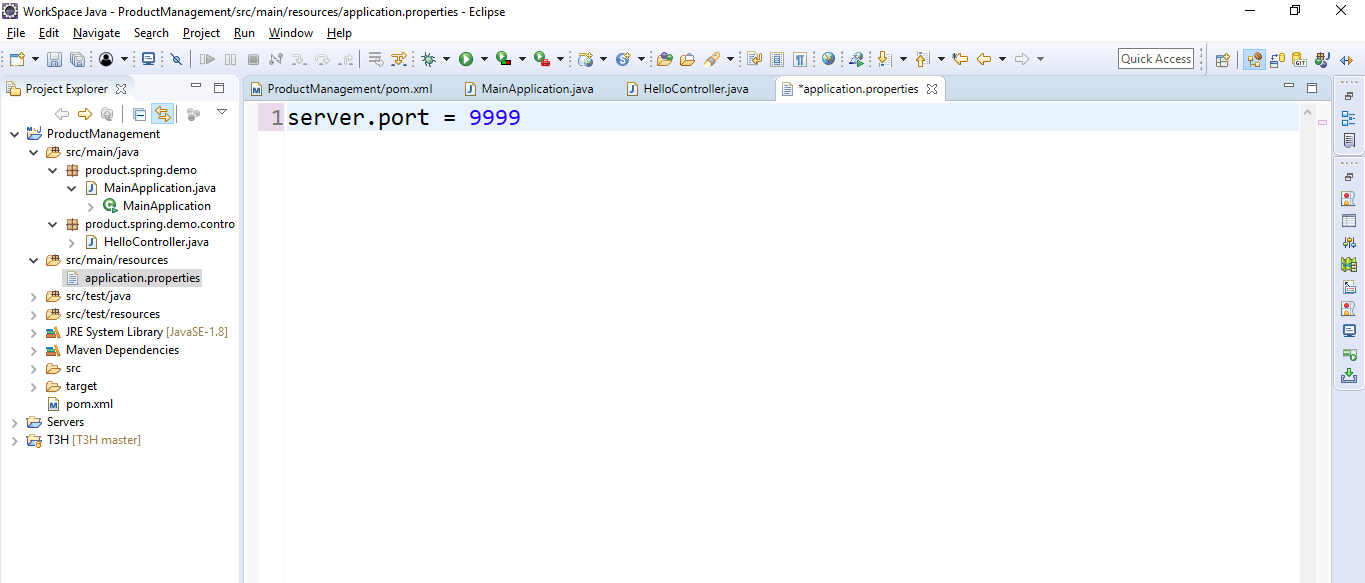






### Step 6: Cấu hình cổng cho ứng dụng trong application.properties

Hãy cấu hình port bạn muốn:

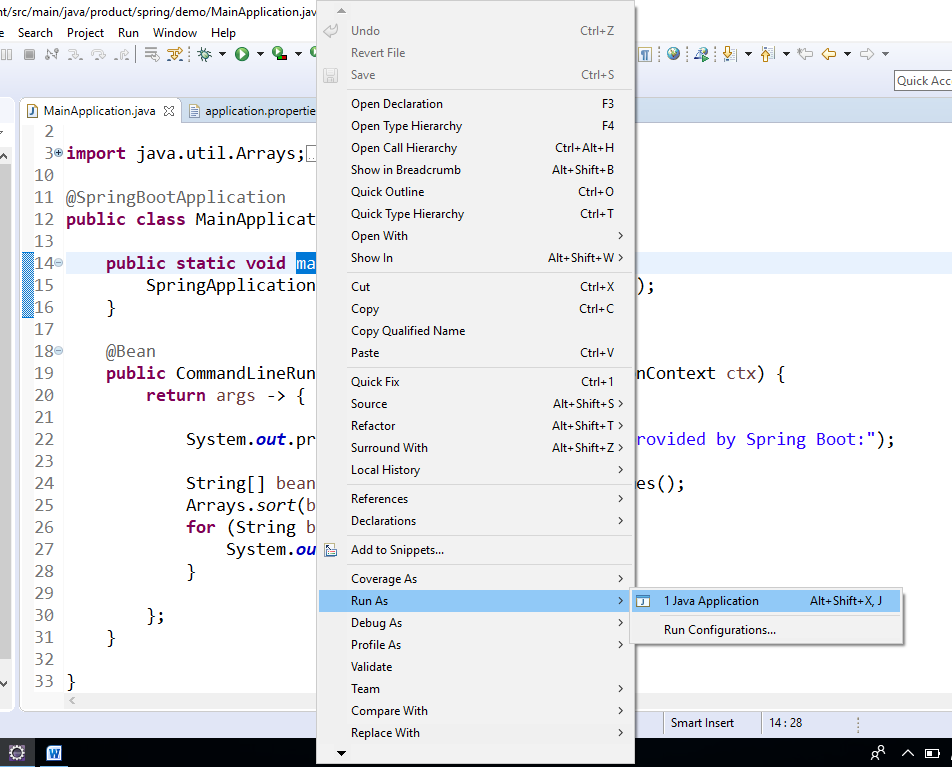


Cấu hình port 9999

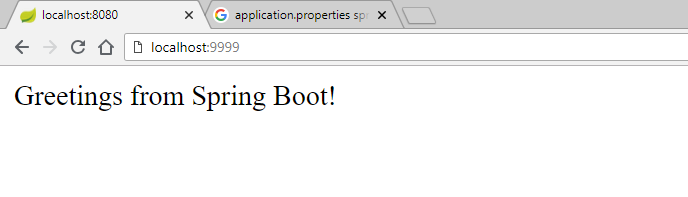
Lưu lại và khởi động và kiểm tra

Sử dụng server.port = 9999 trong file application.properties để cấu hình ứng dụng chạy cổng 9999

Kết quả khi chạy



Kết quả



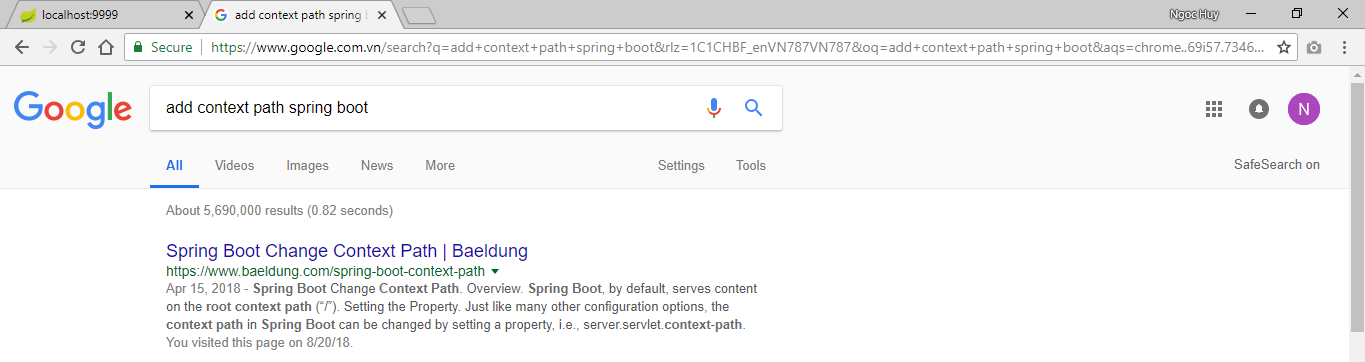
Kết quả

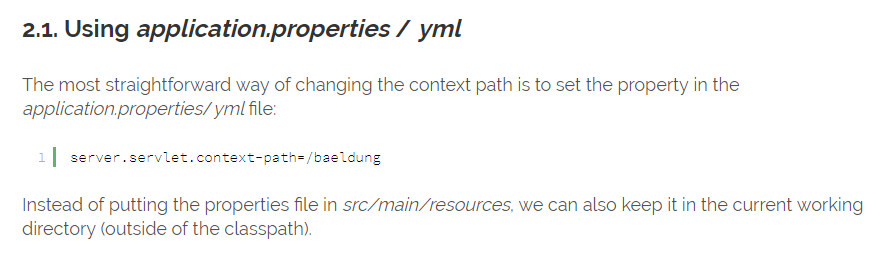
### Step7: Cấu hình context-path ứng dụng

Context Path có thể tạm hiểu là đường tới trang chủ của ứng dụng

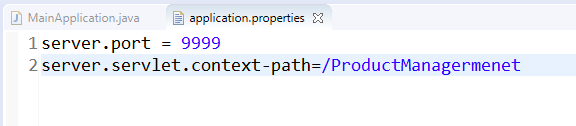
Nếu không cấu hình có thể hiểu là localhost:port là context path mặc định của ứng dụng Spring Boot

Nếu chúng ta muốn localhost:port /ProductManagement thì làm thế nào?



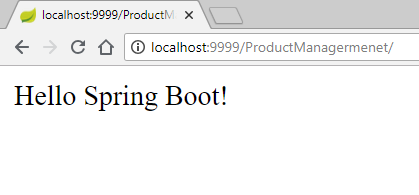
Chủ động tìm kiếm

Hướng dẫn sử dụng context-path

Hãy add cấu hình sau:



### Kết quả bước 6 và 7



### Step8: Add dev-tool Spring Boot

Dùng devtool để không phải khởi động ứng dụng mỗi lần chạy. Kết hợp auto build của Eclipse và dev tool chúng ta sẽ không phải khởi động lại ứng dụng mỗi lần chạy

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <optional>true</optional>  </dependency> |

Hãy Maven => Update Project khi add thư viện vào ứng dụng

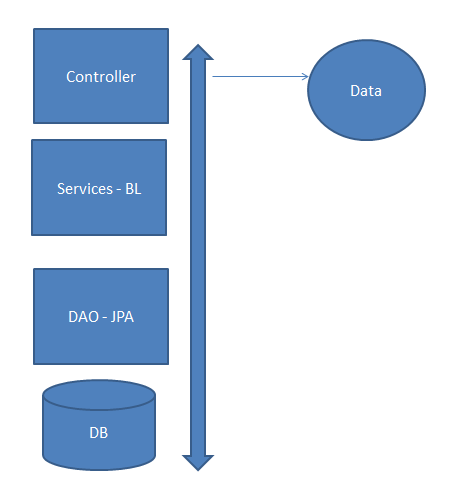
Với phần này chúng ta có thể coi project hiện tại là Hello với Spring MVC và Spring Boot

## Nâng cấp ứng dụng từ phần 2

### Link tham khảo mô hình

##### https://techtalk.vn/mo-hinh-3-lop-co-gi-hay.html

### Mô hình ứng dụng



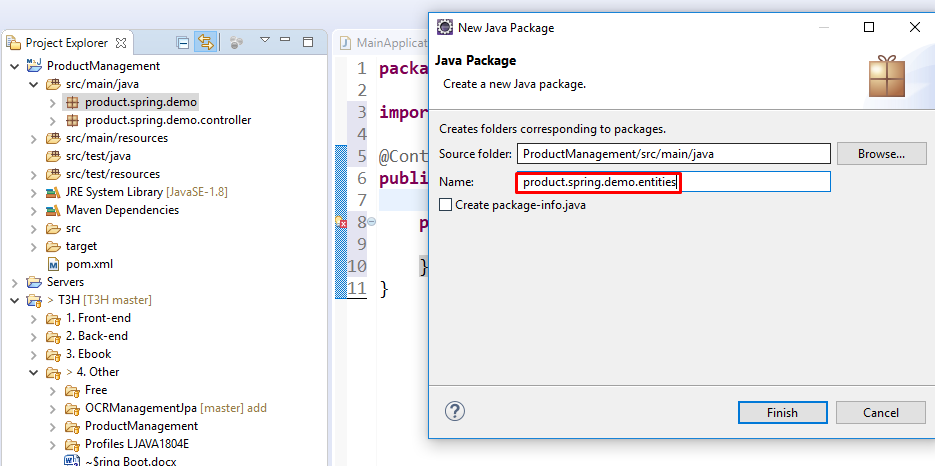
Mô hình giữa các lớp ứng dụng

### Step9: Tạo package

Hãy tạo các package riêng biệt ứng với mỗi phần

|  |  |
| --- | --- |
| product.spring.demo | Chứa main ứng dụng |
| product.spring.demo.controller | Chứa các controller cho web services |
| product.spring.demo.controller.web | Tạo các controller cho Spring MVC |
| product.spring.demo.entities | Chứa các entities class mapping với bảng |
| product.spring.demo.services | Chứa tầng Service |
| product.spring.demo.dao | Chứa Tâng Dao kết nối với Cơ sở dữ liệu |
| product.spring.demo.bean | Chứa các model, bean, value object,… |
| product.spring.demo.vo | Chứa các value object |
| //Toto | //Todo |

Hãy tự tạo các package theo ứng dụng của mình và hãy chủ động tìm hiểu về các mô hình tốt nhất



## Tạo web service với Spring Boot

### Link tham khảo

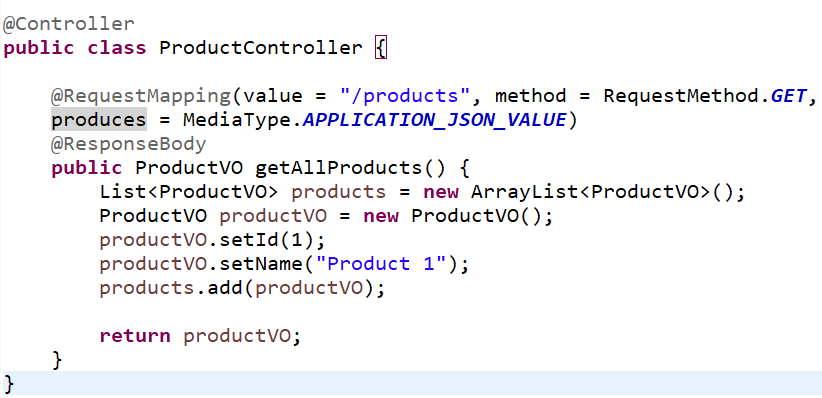
Hãy tham khảo link sau với RESTful service

##### <https://spring.io/guides/gs/rest-service/>

##### <https://springframework.guru/spring-framework-annotations/>

### Step10: Tạo Controller cho web service

Có rất nhiều controllers đây là một controllers cho việc quản lý Product. Và mỗi controller có rất nhiều phương thức quản lý các hành vi với mỗi đối tượng





### Step 10: Giải thích

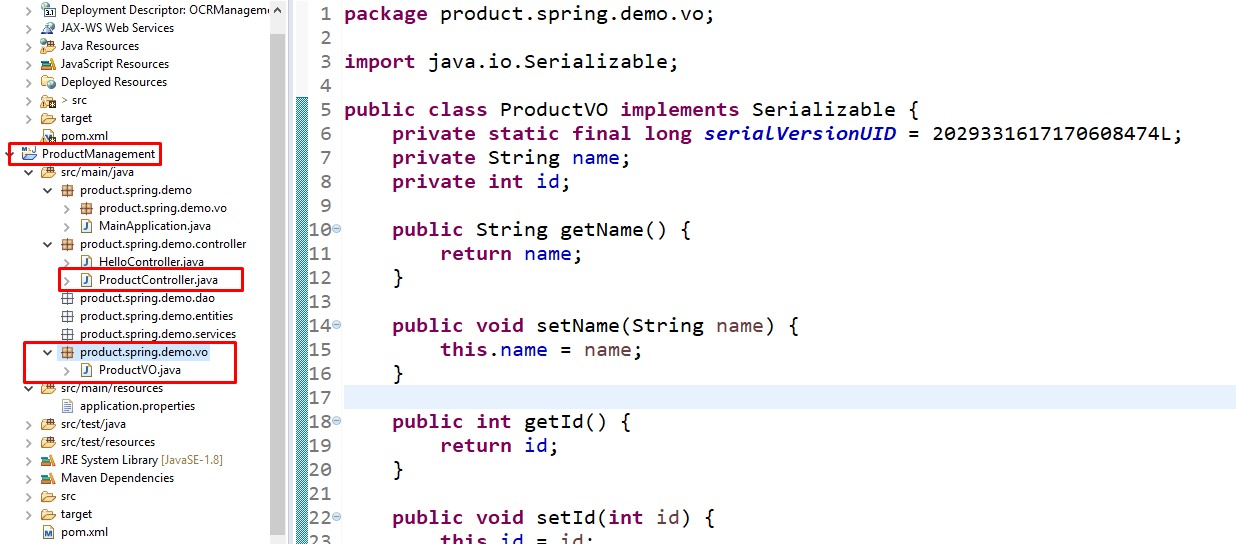
|  |  |
| --- | --- |
| @Controller | Để spring có thể scan quét được |
| @RequestMapping => value | Mapping URL: context path + value |
| @RequestMapping => method | Metho với URL trên |
| @RequestMapping => produces | Loại dữ dữ liệu được sử dụng |
| @ResponseBody | Dữ liệu trả về được nhúng trong Body của response |

### Step 10: Lưu ý

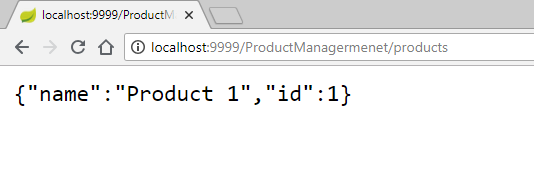
* **Dùng thì mới tạo** .Hãy tạo các controllers tạo các web services với khi chúng ta sử dụng. Nếu chúng ta không sử dụng hiện tại và tương lai thì cũng không nên tạo làm gì. Và code như vậy sẽ tạo ra rác trong ứng dụng
* Hãy tham khảo cách dùng với: **produces** và **consumes** với các method GET, POST trong Spring

### Step 11: Tạo POJO – Value Object cho Service

Bước này có thể tạo trước bước 10, đó là chúng ta tạo một class đơn giản cho Service mình muốn



### Step 11: Kết quả Kiểm tra service



### Step 12: Sử dụng Swagger để test WebService

Swagger là một “plugin” thú vị trong quá trình phát triển Service.

Các bạn có thể chạy trình duyệt để kiểm tra kết quả Step 11. Nhưng để quản lý và hiển thị các controller để test thì chỉ có thể Swagger là mạnh mẽ.

Swagger tự động quét các phương trong các Controller

#### Add thêm thư viện

|  |
| --- |
| <!--For create test case -->  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger2</artifactId>  <version>2.7.0</version>  </dependency>  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger-ui</artifactId>  <version>2.7.0</version>  </dependency> |

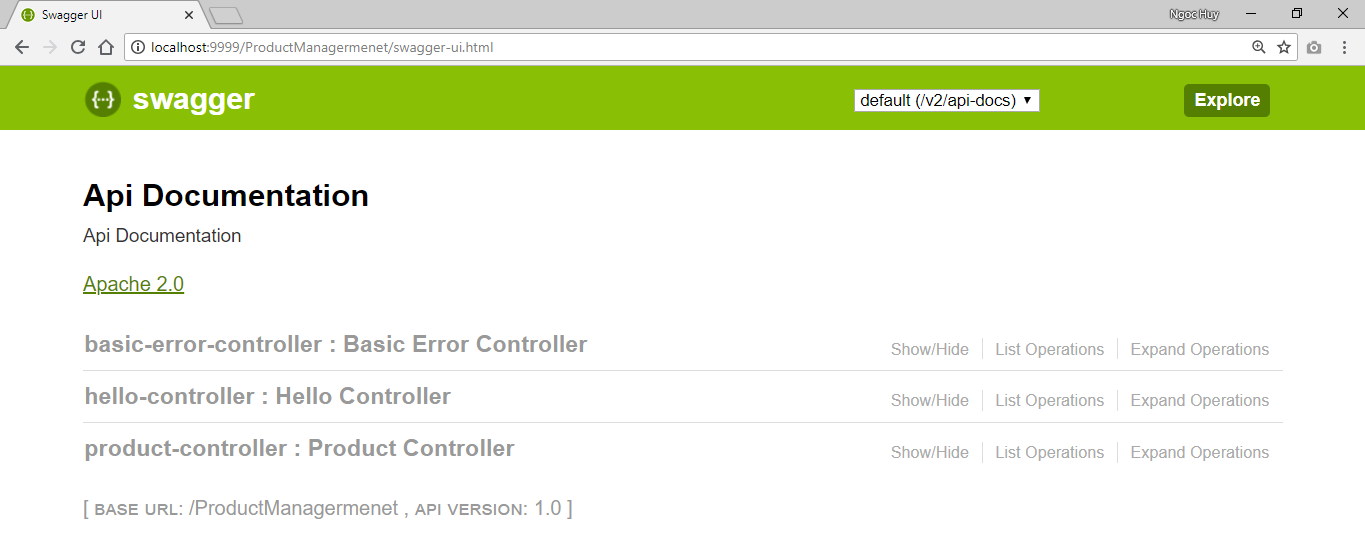
#### Build lại và Update Project

#### Add Bean Swagger



|  |
| --- |
| @EnableSwagger2  **public** **class** SwaggerConfig {  @Bean  **public** Docket api() {  **return** **new** Docket(DocumentationType.***SWAGGER\_2***).select().apis(RequestHandlerSelectors.*any*())  .paths(PathSelectors.*any*()).build();  }  } |

#### Test với đường dẫn: Context-Path + /swagger-ui.html



<http://localhost:9999/ProductManagermenet/swagger-ui.html>

### Project thời điểm phần 4



## Appache Tiles với Spring MVC, Tomcat

Với JSP để tạo template thì Appche Tiles được sử dụng khá phổ biến và dùng dễ dàng

### Step 13: Thư viện, JSP, Appche Tiles

#### Thư viện cho JSP

Muốn sử dụng JSP thì phải sử dụng thư viện hỗ trợ JSP trong Spring Boot

|  |
| --- |
| <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency> |

#### Add Appche Tiles

|  |
| --- |
| <!-- Tiles API -->  <!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-api%20 -->  <dependency>  <groupId>org.apache.tiles</groupId>  <artifactId>tiles-api</artifactId>  <version>3.0.8</version>  </dependency>  <!-- Tiles Core -->  <!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-core%20 -->  <dependency>  <groupId>org.apache.tiles</groupId>  <artifactId>tiles-core</artifactId>  <version>3.0.8</version>  </dependency>  <!-- Tiles Servlet -->  <!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-servlet%20 -->  <dependency>  <groupId>org.apache.tiles</groupId>  <artifactId>tiles-servlet</artifactId>  <version>3.0.8</version>  </dependency>  <!-- Tiles JSP -->  <!-- http://mvnrepository.com/artifact/org.apache.tiles/tiles-jsp%20 -->  <dependency>  <groupId>org.apache.tiles</groupId>  <artifactId>tiles-jsp</artifactId>  <version>3.0.8</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.tiles/tiles-request-api -->  <dependency>  <groupId>org.apache.tiles</groupId>  <artifactId>tiles-request-api</artifactId>  <version>1.0.6</version>  </dependency> |

### Step 14: Cấu hình Project với Tomcat

Project hiện tại có hàm main chạy Run as với Java Application. Nhưng làm sao để chạy trên Server có sẵn ?

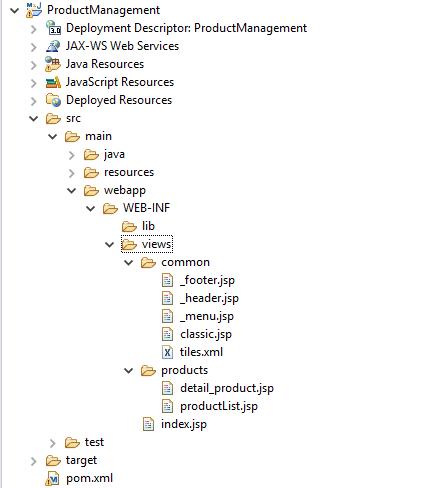
#### Add Buil Project dạng War

|  |
| --- |
| <packaging>war</packaging> |

Khi add tới đây thì ứng dụng có thể build ra dạng war file để deploy trên bất kì server Java nào

#### Hãy Build và Update Project với Maven

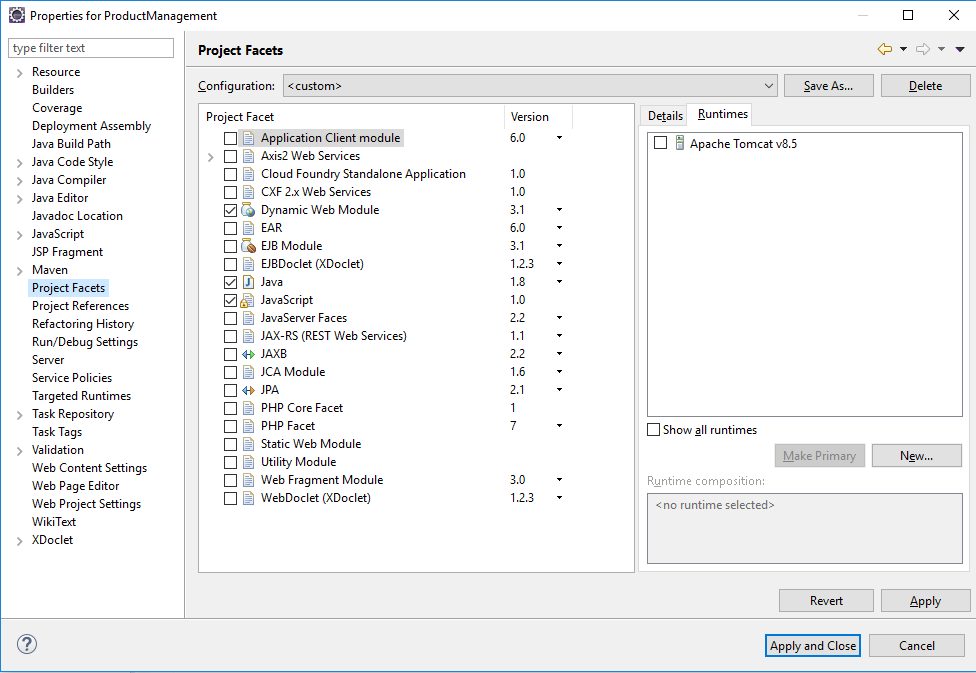
#### Add thư mục Web



Hãy Tạo thư mục các File như trên

#### Kiểm tra lại Cấu hình Project - Project Pacets

Chuột phải Project > **Properties** và Chọn **Project Pacets**



### Step 15: Cấu hình ViewResole, CSS, JS,..

#### Tham khảo

##### https://www.baeldung.com/spring-mvc-view-resolver-tutorial

Website thì cần đọc CSS và JS, Bootstrap, Font, Các Trang JSP

|  |
| --- |
| server.port = 9999  server.servlet.context-path=/ProductManagermenet  #web mvc  spring.mvc.view.prefix:/WEB-INF/views/  spring.mvc.view.suffix:.jsp  security.ignored=/css/\*\*,/js/\*\*,/images/\*\*,/font/\*\* |

Đây là cấu hình ViewResole và đọc CSS và JS