# **MODUL III : MULTIPLE MODULE IN SPRING BOOT REST API**



**Kompetensi Pengetahuan**

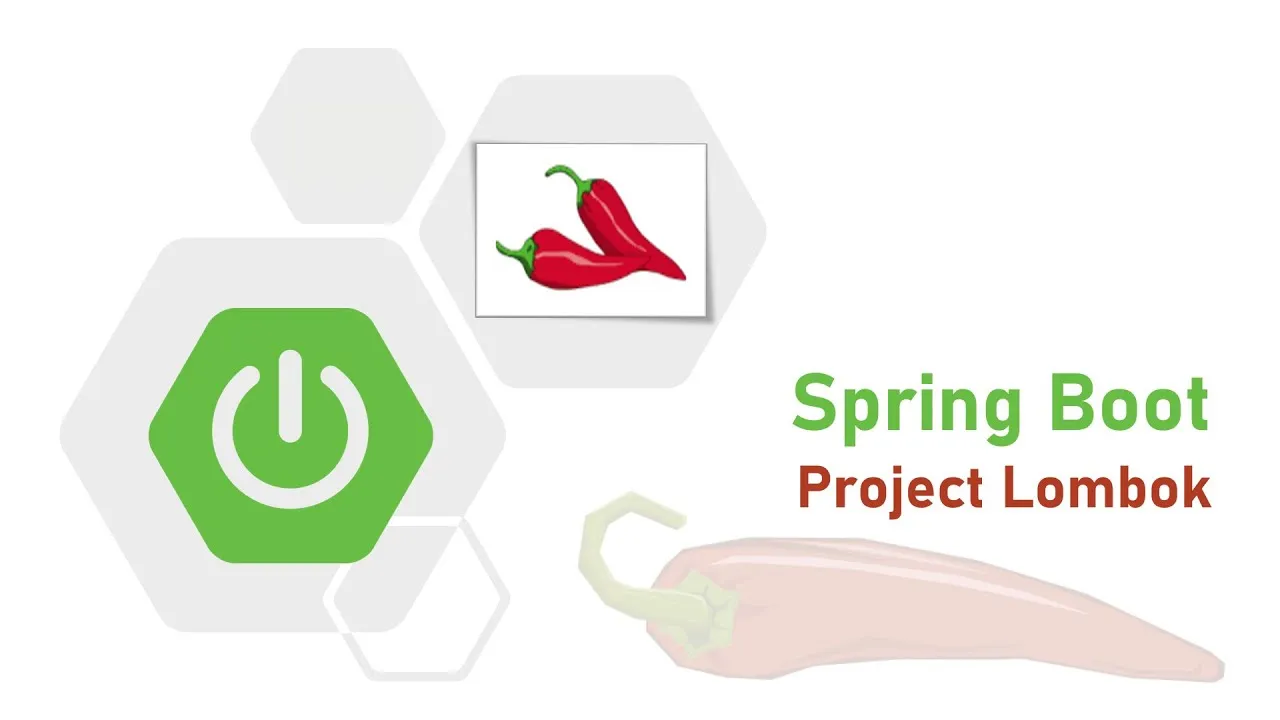
1. Peserta dapat memahami penggunaan lombok.
2. Peserta dapat memahami cara komunikasi antar modul

**Kompetensi Keterampilan**

1. Peserta dapat membuat Rest API Sederhana menggunakan Java Spring Boot
2. Peserta dapat melakukan build dan compile project
3. Peserta dapat menambahkan annotation dalam project
4. Peserta dapat melakukan konfigurasi pada pom.xml

Pada bagian modul ini akan dibuat sebuah REST API dengan mengambil topic Product Service yang berkaitan dengan manajemen data product sehingga akan dibutuhkan beberapa modul seperti modul product impl dan product api. Dengan adanya pemisahan modul product akan memudahkan dalam hal pengembangan dan maintenance ketika project sudah cukup *complex*. Adapun project yang digunakan masih lanjutan dari project pada bab sebelumnya.

## **Lombok Dependency (POJO)**

****

Sumber : <https://morioh.com/p/b6573ec02205>

Lombok adalah library/dependency didalam java yang secara otomatis dihubungkan ke editor dan build tools guna memperingkas penulisan code di dalam java terutama dengan menghilangkan method getter, setter, dan lain sebagainya. Langkah-langkah dalam pemasangan dependency lombok adalah sebagai berikut

1. Instalasi Lombok dengan version terupdate dengan menambahkan dependency lombok ke file pom

<dependencies>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.22</version>

<scope>provided</scope>

</dependency>

</dependencies>

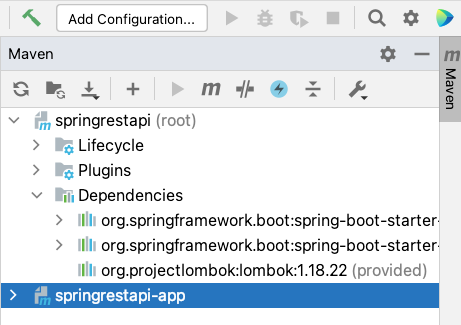
1. Selanjutnya masukkan dependency lombok ke dalam file pom ke project rest api



1. Lakukan **Reload maven** di dalam maven tool window dengan klik ikon reload

****

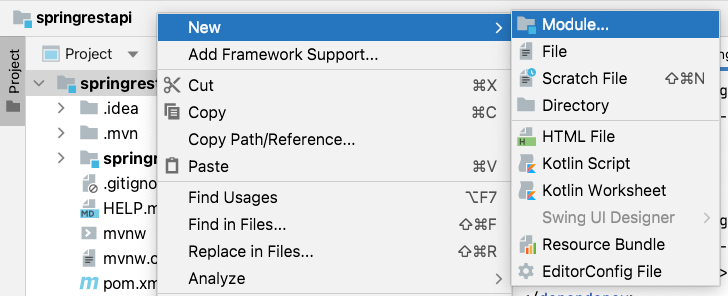
1. Pastikan dependency lombok sudah terdownload dan terdapat di dalam list dependency maven di dalam project springrestapi

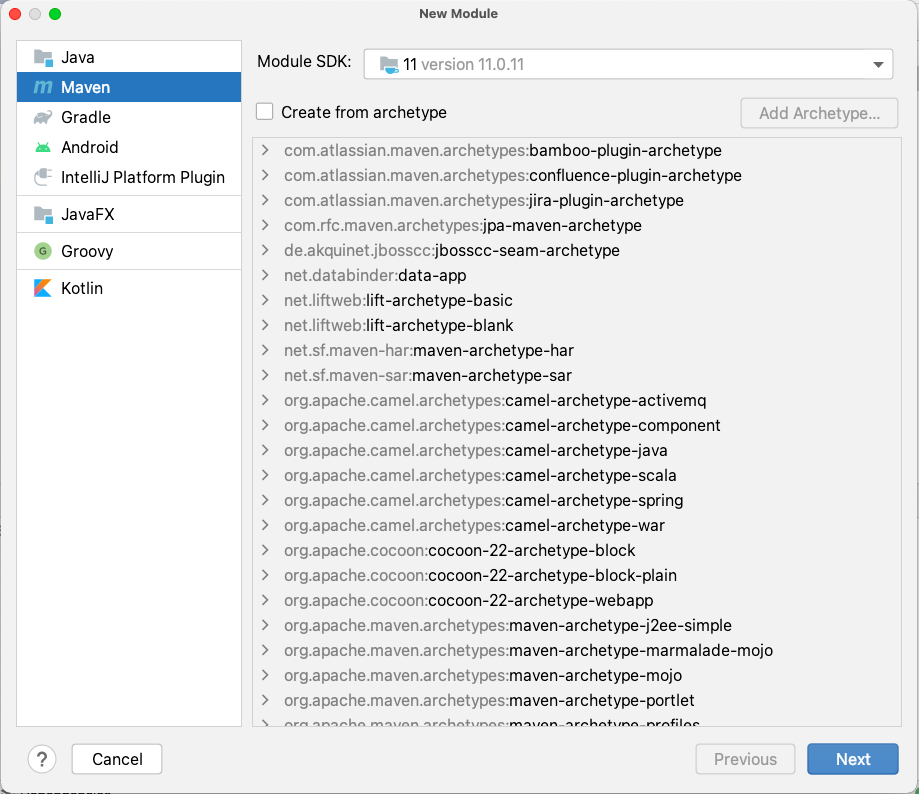
****

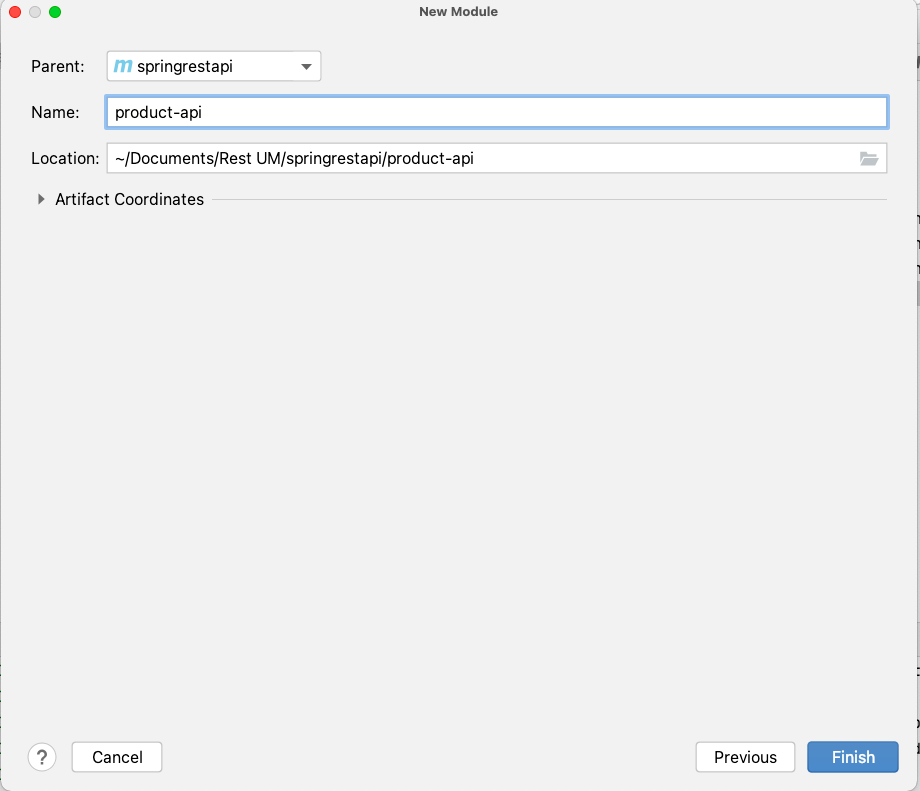
## **Modul Product API**

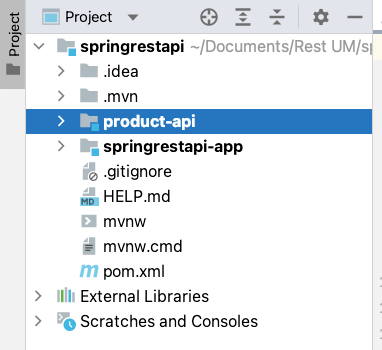
Selanjutnya pada sub modul ini akan ditambahkan modul product API yang akan digunakan untuk membuat end point mengenai product dengan langkah sebagai berikut

1. Buat sebuah modul baru dengan nama product-api

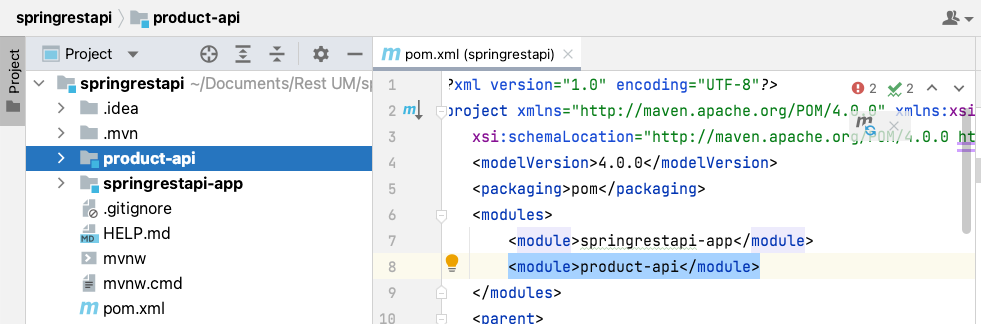


****

****

****

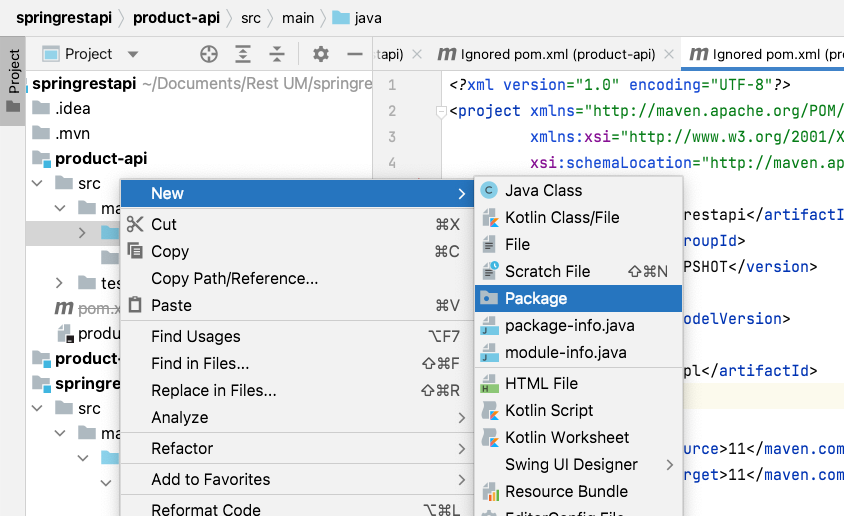
1. Pastikan di dalam pom parent sudah terdapat modul product-api

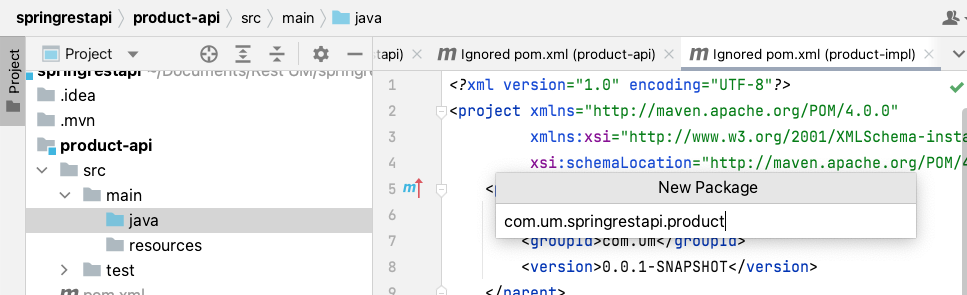
**

1. Buat sebuah package di dalam modul product-api dengan nama com.um.springrestapi.product

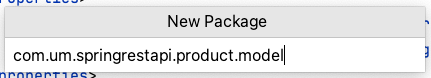
*Nama package ditambahkan dengan nama service yang akan dibuat dimana akan dibuat sebuah product service maka hanya perlu tambahkan .product setelah com.um.springrestapi sehingga menjadi* ***com.um.springrestapi.product***

1. Buat sebuah package dengan nama artifact yang sama dan tambahkan **product** untuk menandakan sebagai service sehingga menjadi com.um.springrestapi.product

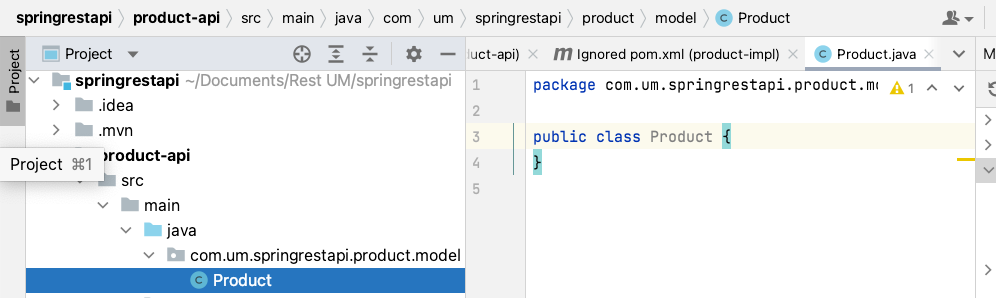
**

**

1. Di dalam package product tambahkan package lagi untuk menyimpan data model dengan nama model

**

1. Buat sebuah file java dengan nama Product yang akan digunakan sebagai POJO

**

1. Tambahkan annotation @Data, @Builder, @NoArgsConstructor, dan @AllArgsConstructor dalam class Product. Penjelasan masing-masing annotation tersebut adalah



**Product.java**

import lombok.\*;

@Data

@Builder

@NoArgsConstructor

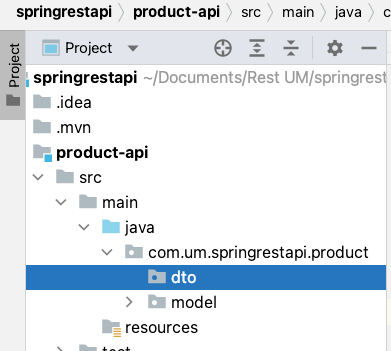
@AllArgsConstructor(access = AccessLevel.*PRIVATE*)

public class Product {

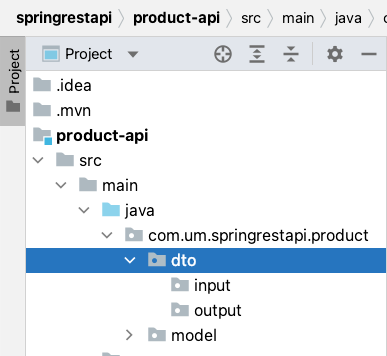
}

* @Data : merupakan annotation dari dependency lombok yang dapat otomatis dapat menggunakan method getter dan setter dalam setiap property
* @Builder : merupakan annotation yang mempunyai mekanisme untuk menggunakan Builder tanpa menulis kode boilerplate.
* @NoArgsConstructor : merupakan annotation untuk generate sebuah constructor dalam class tanpa sebuah parameter
* @AllArgsConstructor : merupakan annotation untuk generate sebuah constructor dengan semua parameter yang ada dalam property

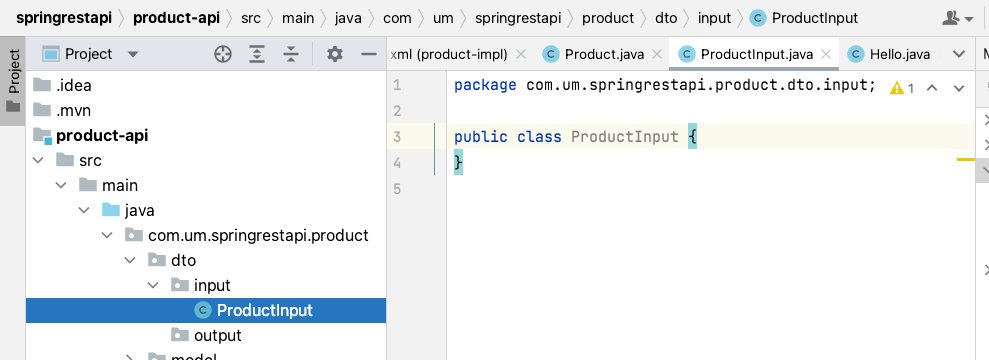
1. Tambahkan package dengan nama dto yang akan digunakan untuk **Data Transfer Object (DTO)**

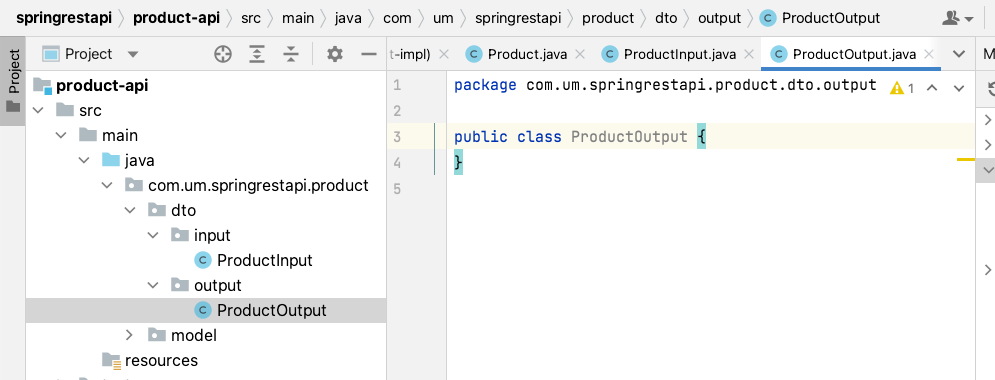


1. Tambahkan package **input** dan **output** di dalam package dto



1. Tambahkan kelas java ProductInput.java di dalam package input dan kelas java ProductOutput.java di dalam package output





1. Tambahkan annotation dan atribut sesuai dengan kebutuhan untuk kelas ProductInput



**ProductInput.java**

import lombok.\*;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor(access = AccessLevel.*PRIVATE*)

public class ProductInput {

private String name;

private String description;

private Double price;

}

1. Tambahkan annotation dan atribut sesuai dengan kebutuhan untuk kelas ProductOutput



**ProductOutput.java**

import lombok.\*;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor(access = AccessLevel.*PRIVATE*)

public class ProductOutput {

private Long id;

private String name;

private String description;

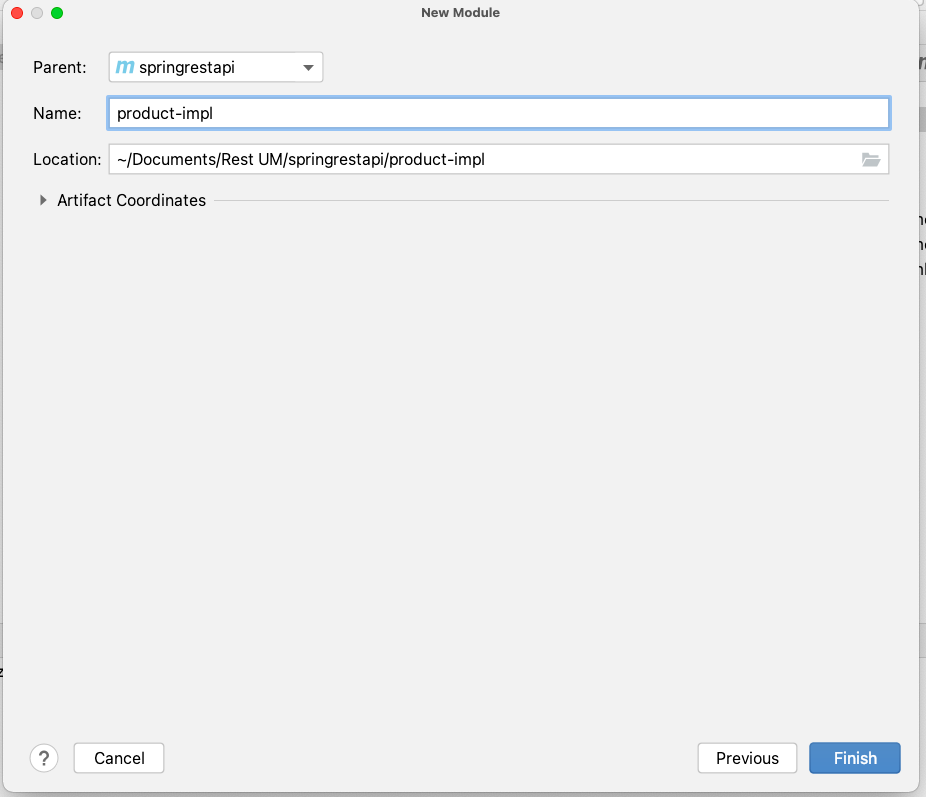
private Double price;

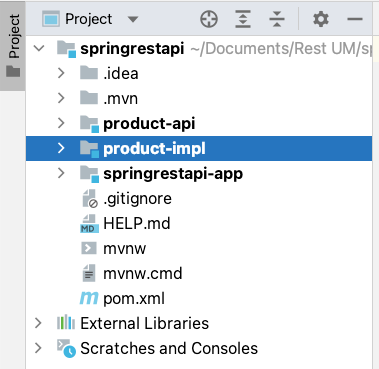
}

## **Modul Product Impl**

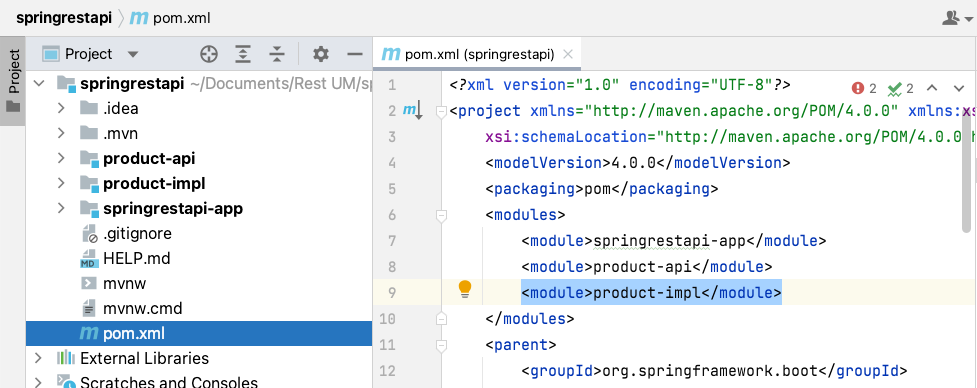
Pada bagian ini akan dibuatkan modul baru yakni product-impl yang akan digunakan sebagai implementasi dari endpoint dari sebuah product. Adapun cara menambah dan code yang perlu ditambahkan dalam modul Product Impl adalah sebagai berikut

1. Buat sebuah modul baru dengan nama product-impl

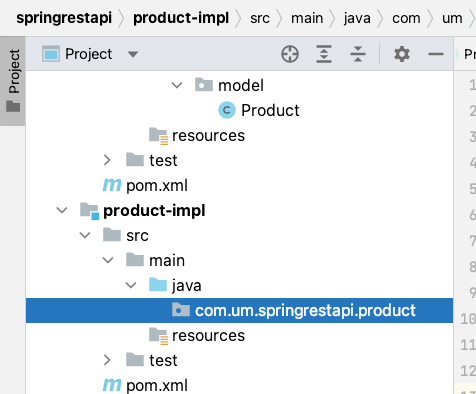
****

****

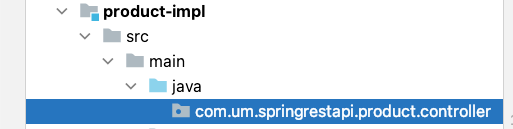
1. Pastikan di dalam pom parent sudah terdapat modul product-impl

**

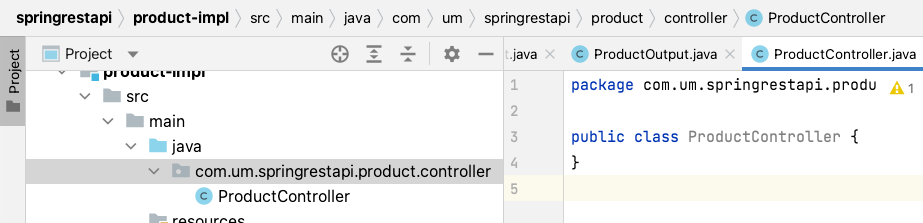
1. Buat sebuah package product service dengan nama artifact sebelumnya di dalam module product-impl



1. Buat sebuah package dengan nama controller di dalam module product-impl



1. Buat sebuah class ProductController di dalam package controller



1. Dikarenakan nantinya module product-impl akan mememanggil kelas yang ada di dalam module product-api sehingga perlu ditambahkan di file pom.xml module product-impl berupa dependency product-api



**pom.xml (modul product-impl)**

...

<dependencies>

<dependency>

<groupId>com.um</groupId>

<artifactId>product-api</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

</dependencies>

...

1. Tambahkan controller untuk method get dan method post



**ProductController.java**

import com.um.springrestapi.product.dto.input.ProductInput;

import com.um.springrestapi.product.dto.output.ProductOutput;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/products")

public class ProductController {

@GetMapping("/{id}")

public ResponseEntity<ProductOutput> getOne(@PathVariable Long id) {

return ResponseEntity.*ok*(ProductOutput.*builder*().build());

}

@GetMapping

public ResponseEntity<List<ProductOutput>> getOne() {

return ResponseEntity.*ok*(null);

}

@PostMapping

public ResponseEntity<ProductOutput> insertOne(ProductInput productInput) {

return ResponseEntity.*ok*(ProductOutput.*builder*().build());

}

}

1. Running project dan testing di dalam postman

## **Swagger**

Pada bagian sub modul ini akan di generate sebuah documentation dari sebuah API yang telah dikembangkan ke dalam swagger. Swagger sendiri dari *official documentation* (swagger.io, 2021) merupakan sebuah UI untuk mendeskripsikan RESTful API menggunakan JSON . Adapun langkah-langkah dalam generate swagger di spring boot adalah sebagai berikut

1. Tambahkan dependency swagger di file pom.xml dalam module springrestapi-app



**Pom.xml (module springrestapi-app)**

. . .

<dependencies>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

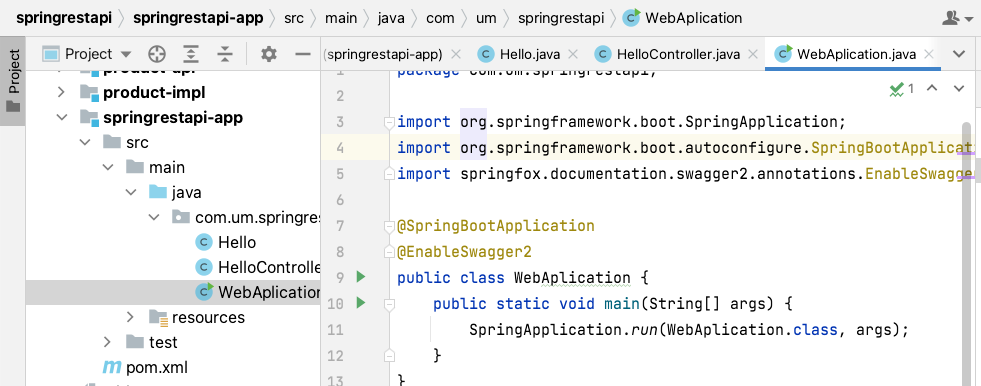
<version>2.9.2</version>

</dependency>

</dependencies>

. . .

1. Tambahkan annotation @EnabledSwagger2 di dalam kelas application untuk memberikan indentity bahwa project ini akan menggunakan SwaggerApi document



1. Tambahkan bean untuk beberapa konfigurasi dari swagger letakkan di dalam class WebApplication

…

@Bean

public Docket api() {

return new Docket(DocumentationType.*SWAGGER\_2*)

.select()

.apis(RequestHandlerSelectors.*basePackage*(this.getClass().getPackageName()))

.paths(PathSelectors.*any*())

.build()

.useDefaultResponseMessages(false);

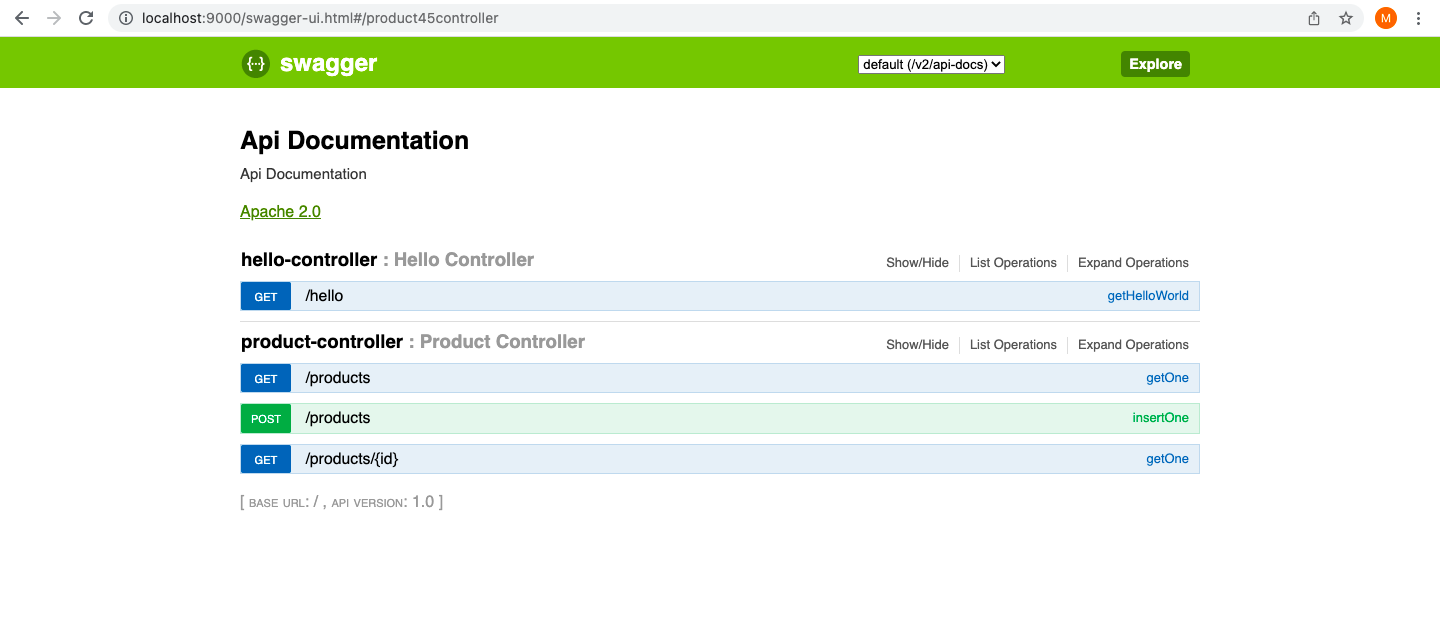
}

…

1. Running springrestapi-app module bisa dengan melalui IDE di IntellJ dengan klik tombol Run di Web Aplication



1. Access localhost:8080/swagger-ui.html



# 

## **Rangkuman Modul III**

1. **Lombok** adalah library/dependency didalam java yang secara otomatis dihubungkan ke editor dan build tools guna memperingkas penulisan code di dalam java terutama dengan menghilangkan method getter, setter, dan lain sebagainya. Annotation pending di Lombok yakni

* **@Data** : merupakan annotation dari dependency lombok yang dapat otomatis dapat menggunakan method getter dan setter dalam setiap property
* **@Builder** : merupakan annotation yang mempunyai mekanisme untuk menggunakan Builder tanpa menulis kode boilerplate.
* **@NoArgsConstructor** : merupakan annotation untuk generate sebuah constructor dalam class tanpa sebuah parameter
* **@AllArgsConstructor** : merupakan annotation untuk generate sebuah constructor dengan semua parameter yang ada dalam property

1. **Swagger** merupakan sebuah UI untuk mendeskripsikan RESTful API menggunakan JSON



## **Tugas Praktikum (Ketrampilan) Modul III**

1. Buat sebuah Rest API multiple modul seperti materi di modul III
2. Tambahkan untuk end point pada multiple modul untuk update dan delete product dan sesuaikan respond dari rest api dengan standard genaral
3. Generate Documentation API Swagger dari Rest API Spring Boot dengan multiple modul diatas



# 

## **Tugas Pilihan Ganda (Pengetahuan) MODUL III**

1. Buka Link Berikut (Token : 333)

[**https://forms.gle/JUc4GDiBgysZWxTT7**](https://forms.gle/JUc4GDiBgysZWxTT7)

1. Isikan identitas dan Jawab Soal Pilihan Ganda dalam form tersebut dengan waktu 10 menit
2. Jumlah Soal terdapat 10 Soal