# **MODUL II : SPRING BOOT REST API STARTED**



**Kompetensi Pengetahuan**

1. Peserta dapat memahami cara membuat Rest API sederhana menggunakan java Spring Boot.
2. Peserta dapat memahami proses build dan compile project Rest API Java Spring Boot
3. Peserta dapat memahami annotation Rest API dalam Java Spring Boot
4. Peserta memahami management tools maven pada file pom.xml

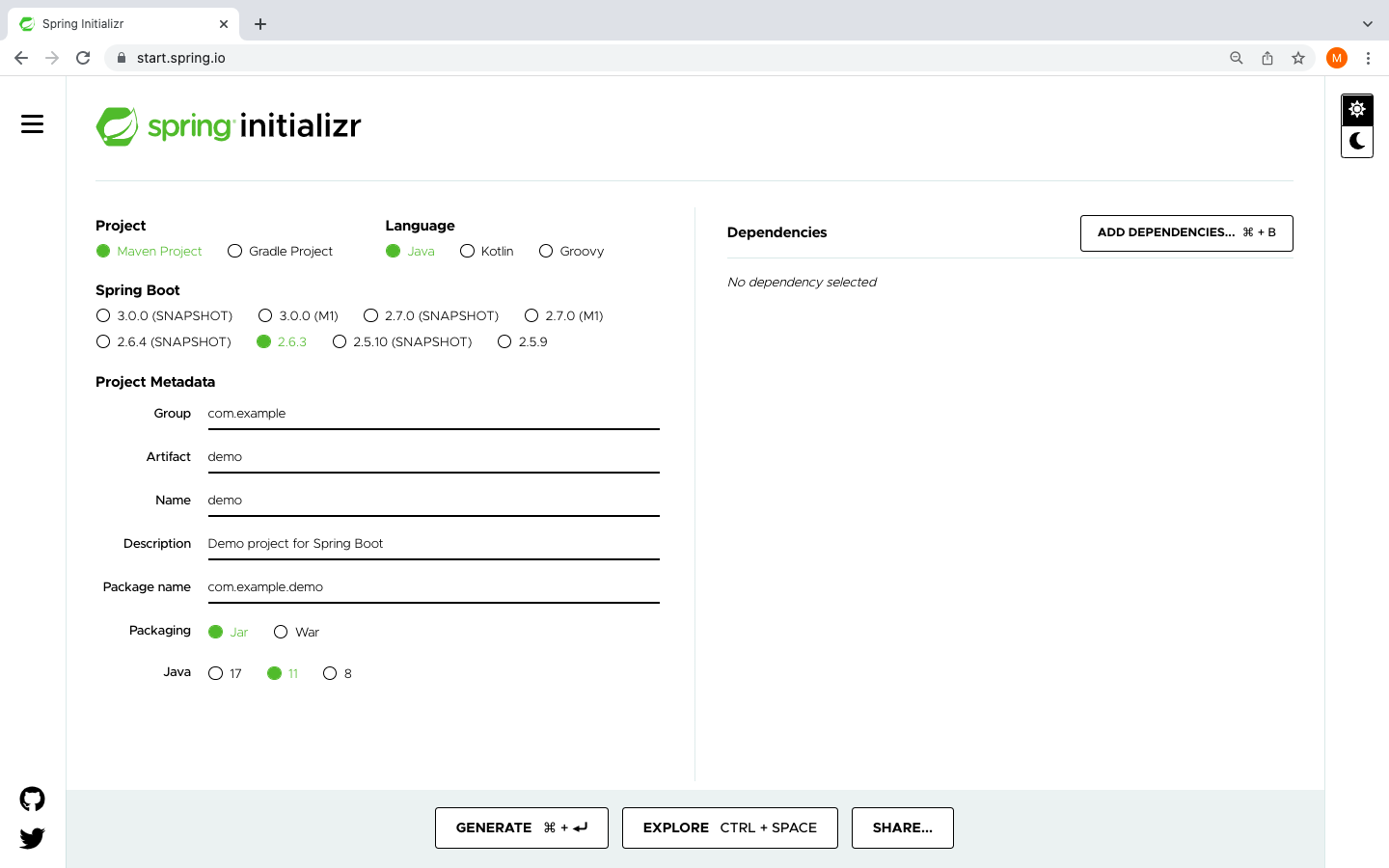
**Kompetensi Keterampilan**

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 dimulai cara pembuatan mulai dari awal membuat sebuah Rest API sederhana menggunakan Java Spring Boot hingga *testing* hasil pengembangan Rest API menggunakan postman.

## **Memulai Project dengan Spring Initializr**

Dalam membuat project dengan Spring Boot terlebih dahulu perlu melakukan konfigurasi awal dengan memasukkan data-data *identity project* dan kebutuhan *dependency* awal. Hal ini dapat dipermudah dengan menggunakan *Spring Initializr* yang dapat diakses di <http://start.spring.io>



1. Setelah muncul halaman web diatas lalu isikan data diatas sesuai data berikut
2. Pilih automation tools **Maven**
3. Language dipilih **Java**
4. Versi Spring Boot pilih **3.0.0** (Pilih versi stabil paling update)
5. Isikan data Project Metadata

* **Group**

Informasi perusahaan atau organisasi dari developer

Contoh : com.um

* **Artifact**

Nama project yang akan dibuat umumnya ditulis dengan lowercase

Contoh : springrestapi

* **Name**

Nama project dengan format tulisan bebas

Contoh : Spring Rest API

* **Description**

Deskripsi informasi lebih detail untuk project yang akan dibuat

* **Package Name**

Nama package identity project yang merupakan gabungan dari nama group dan artifact

Contoh : com.um.springrestapi

* **Packaging**

Format ekstensi dari hasil build source code project

Pilih format **Jar**

* **Java**

Versi java yang akan digunakan dalam pengembangan source code

Pilih versi yang stabil dan cukup update yakni 11

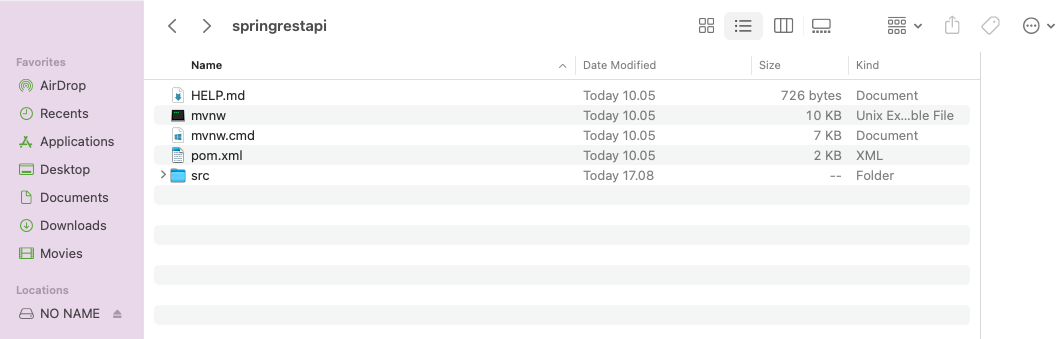
1. Dependency bisa dikosongkan terlebih dahulu nanti dependency bisa ditambahkan manual ketika di proses development pada file pom.xml
2. Setelah dilakukan pengisian data sesuai diatas selanjutnya klik Generate untuk mendapatkan project initialize yang akan kita gunakan dalam pengembangan Rest API



1. Setelah berhasil didownload *extract* hasil download initialize project tersebut dan letakkan di folder komputer masing-masing sesuai yang diinginkan sebagai contoh di Drive D, My Document, atau tempat lainnya



1. Hasil extract springrestapi.zip sebagai berikut



## **Open IntelliJ Application**

Selanjutnya pada bagian ini akan dimulai pengembangan project code menggunakan IDE IntelliJ dengan langkah sebagai berikut

1. Buka aplikasi IntelliJ untuk memulai proses coding dari initialize project yang telah dibuat sebelumnya



1. Pilih menu open untuk memilih project yang akan dibuka



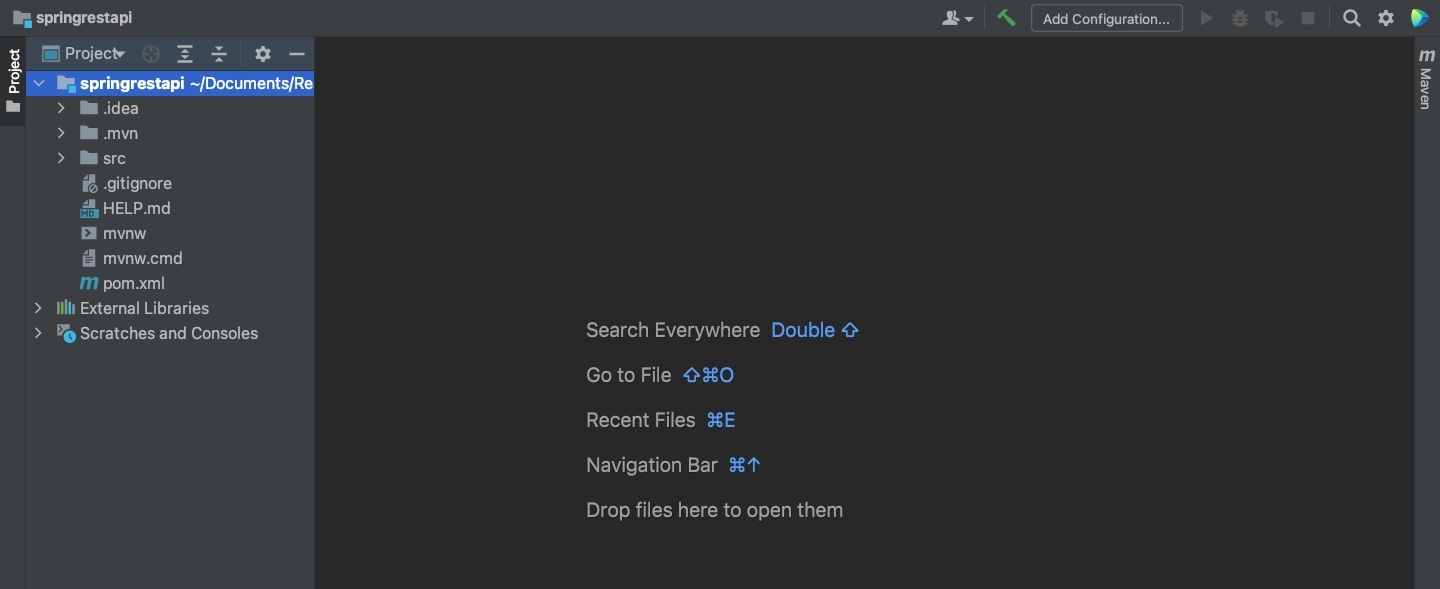
1. Pilih lokasi project yang sebelumnya telah tersimpan lalu klik open



1. Pilih tombol Trust Project untuk memberikan konfirmasi bahwa project yang akan dibuka bisa dipercaya



1. Selanjutnya akan muncul Main Window Project yang akan di kembangkan



## **Pom**

POM merupakan singkatan dari Project Object Model. File pom.xml terdiri dari informasi project dan informasi konfigurasi untuk maven dalam melakukan automation build seperti dependency, build directory, source directory, test source directory, plugins, golas, dll. Adapun file pom.xml dari project perlu diubah dengan langkah sebagai berikut

1. Buka file pom.xml di dalam root project



1. Buka file pom.xml di dalam root project lalu sesuaikan dengan yang ada dibawah yakni pada dependency spring-boot-starter diganti dengan spring-boot-starter-web



**Pom.xml (root project)**

*<?*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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<modules>

<module>springrestapi-app</module>

</modules>

<parent>

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

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

<version>3.0.0-M1</version>

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

</parent>

<groupId>com.um</groupId>

<artifactId>springrestapi</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springrestapi</name>

<description>Project Spring Rest API</description>

<properties>

<java.version>17</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>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</pluginRepository>

</pluginRepositories>

</project>

**Penjelasan Struktur POM :**

1. Modules

Menginformasikan child modul-modul yang terdapat dalam project tersebut

1. Parent

Dalam pom terdapat konsep parent dan child, dan untuk mendeklarasikan parent dapat dilakukan dengan tag parent. Sehingga semua dependency dari parent dapat dipakai oleh child.

1. Metadata

Berisi informasi tentang project yakni groupId, artifactId, version, name, dan description

1. Properties

Umumnya digunakan untuk mengatur sebuah variable di dalam file POM

1. Dependency

Sebuah deklarasi dari sebuah dependency/framework/library di dalam project

1. Dependencies

Merupakan satu set atau kesetuan dari satu atau lebih dependency

1. Build

Digunakan untuk menambahkan plugin ke dalam maven

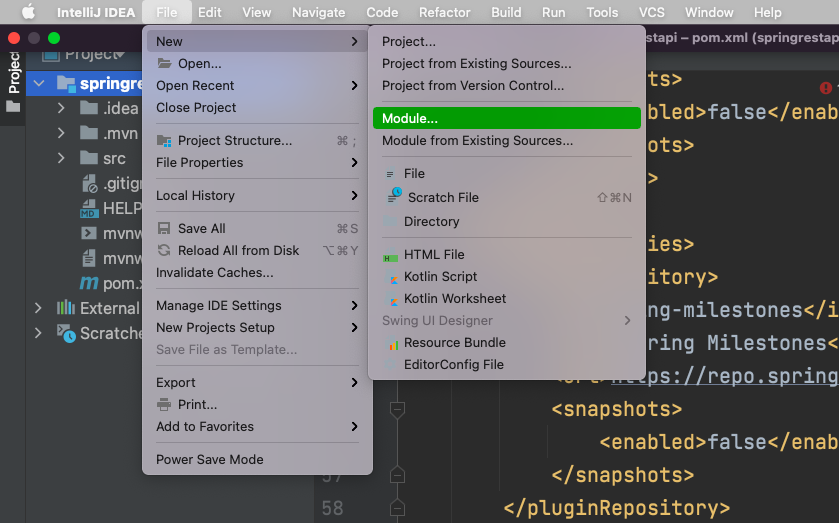
## **Module App Sederhana**

Selanjutnya pada bagian ini akan dibuat sebuah modul baru yang akan digunakan sebagai modul utama dalam pengembangan Rest API. Pada project sebelumnya sebenarnya bisa dilakukan langsung melakukan pengembangan pada folder src. Namun agar lebih modular maka pada project ini nantinya akan dibuat modul-modul.

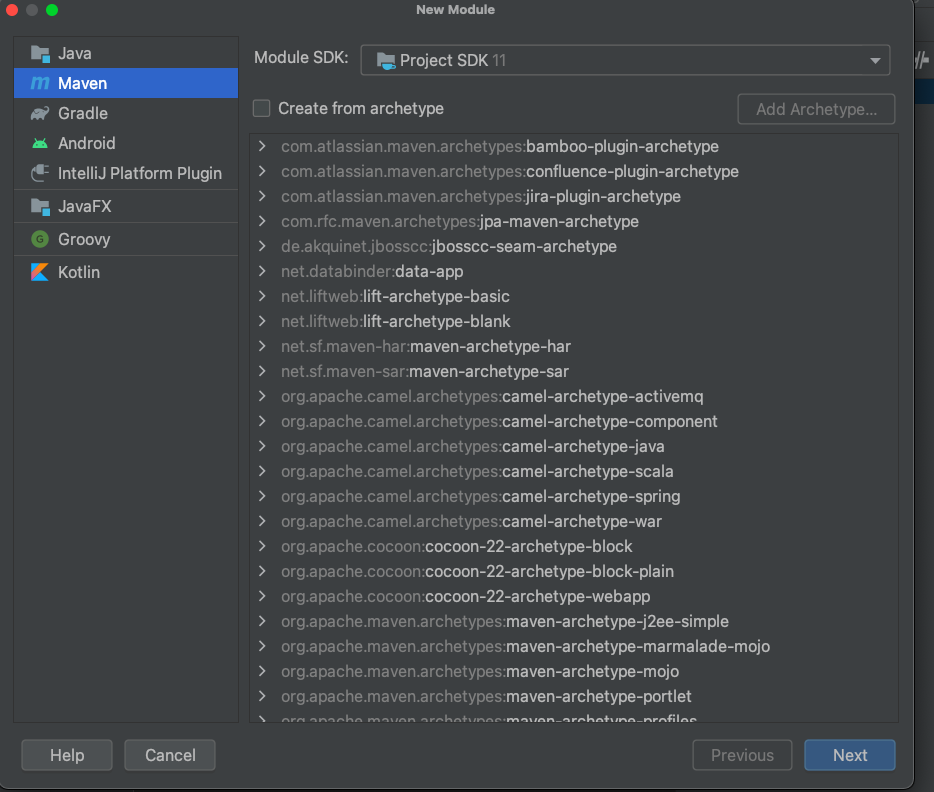
 

Adapun langkah-langkah membuat 1 modul sederhana sebagai berikut

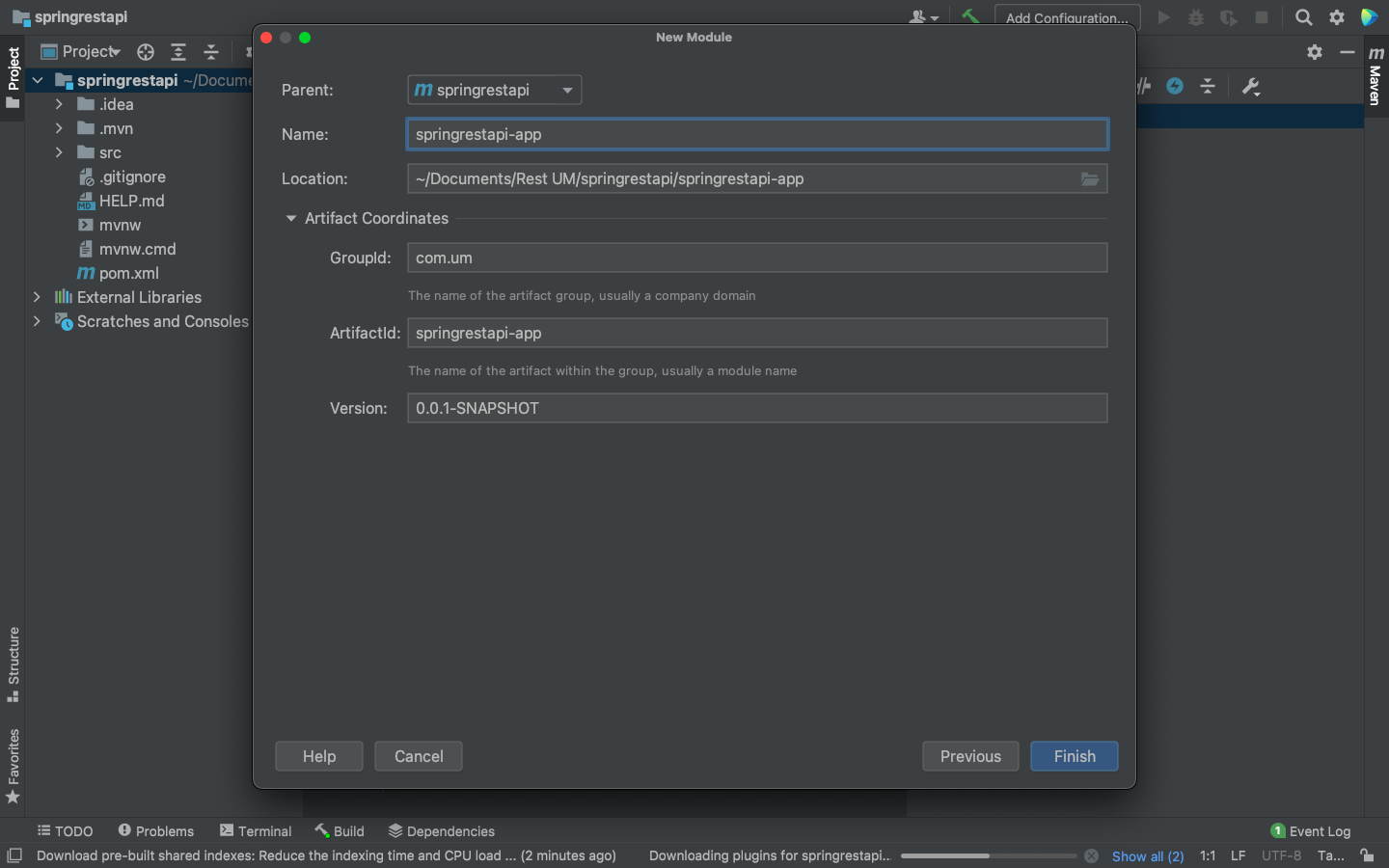
1. Buat sebuah module dengan klik **New** lalu pilih menu **Module**



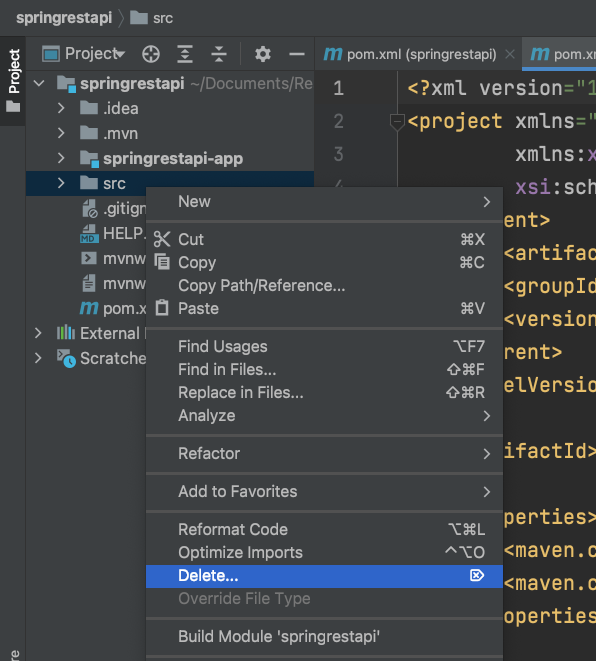
1. Pilih **maven** dan Pilih Java **JDK min 11** lalu klik tombol next pada pojok kanan bawah



1. Isikan name untuk nama modul sebagai contoh springrestapi-app, selanjutnya untuk nama groupId bisa disamakan dengan nama group id di root project yakni com.um serta nama artifactId akan digunakan identitas modul ketika akan digunakan oleh modul lain

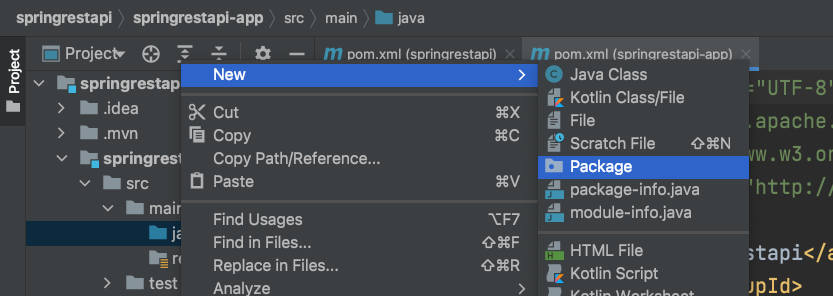


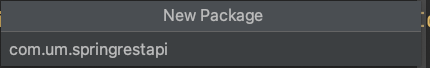
1. Selanjutnya karena project ini akan berbasis modular maka folder **src** yang ada di dalam root project bisa di hapus dengan cara klik kanan pada folder src lalu pilih menu Delete



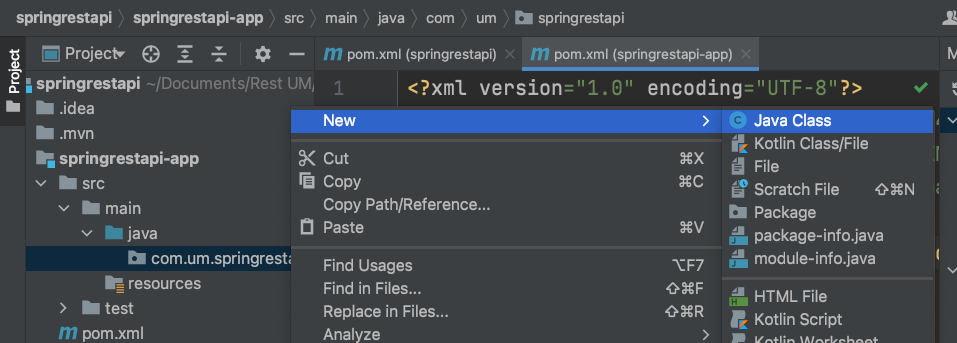
1. Setelah itu pilih tombol **reload maven** guna bisa mendapatkan *dependency* yang belum didapatkan sesuai *maven* di *root project* maupun di modul. Adapun letak tombol *refresh* ada di  *tool window maven* berada di sebalah kanan dan pilih *icon* *refresh*

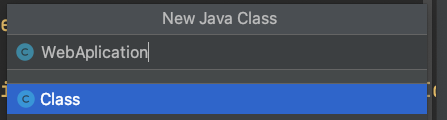
****

1. Di dalam folder **java** di dalam modul springrestapi-app tambahkan sebuah package terlebih dahulu, dimana nama package adalah nama group ditambah nama artifactid dari nama project yakni **com.um.springrestapi**



1. Buat sebuah file java dengan nama WebApplication di dalam package com.um.springrestapi





1. Di dalam file java tersebut tambahkan annotation @SpringBootApplication dan fungsi main yang di dalam tambahkan code run spring boot



**WebAplication.java**

package com.um.springrestapi;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class WebAplication {

public static void main(String[] args) {

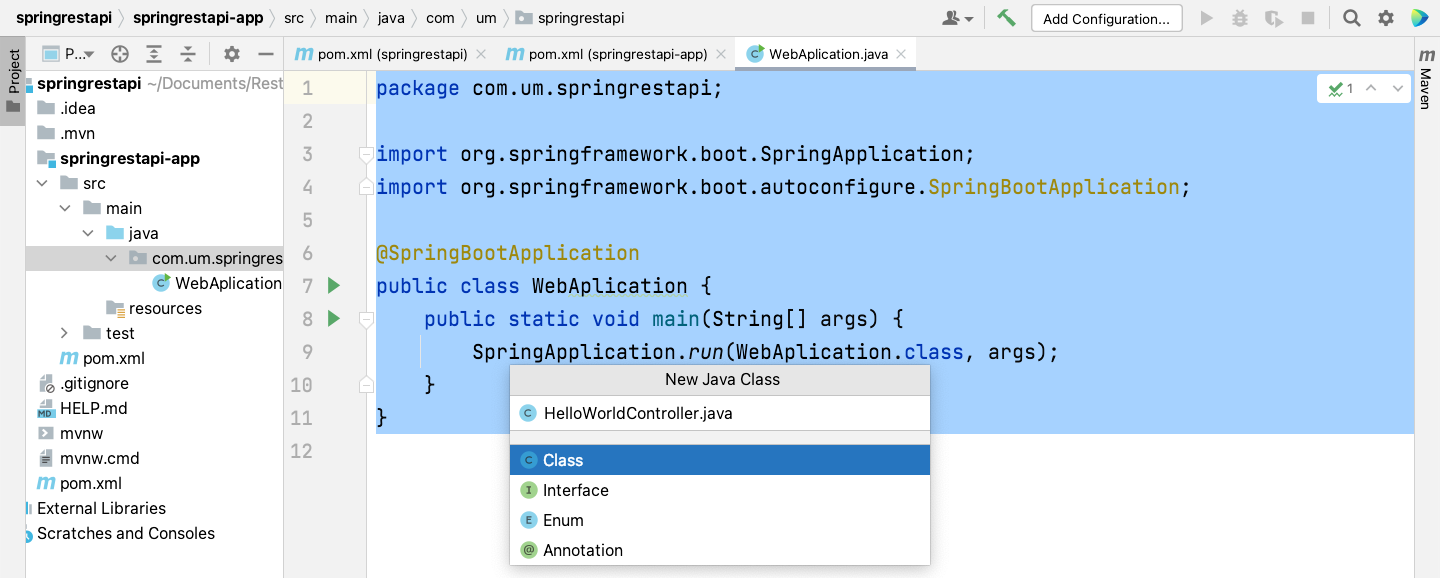
SpringApplication.*run*(WebAplication.class, args);

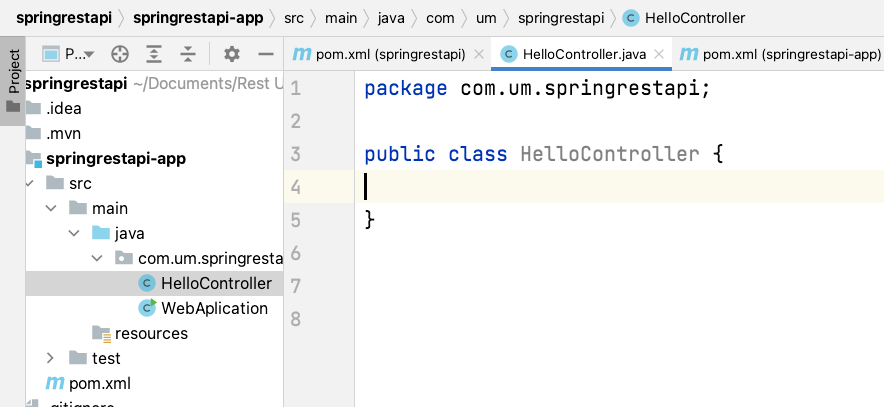
}

}

@SpringBootApplication merupakan annotation yang digunakan untuk informasi kepada compiler bawah file application yang kita kembangkan menggunakan Framework Spring Boot

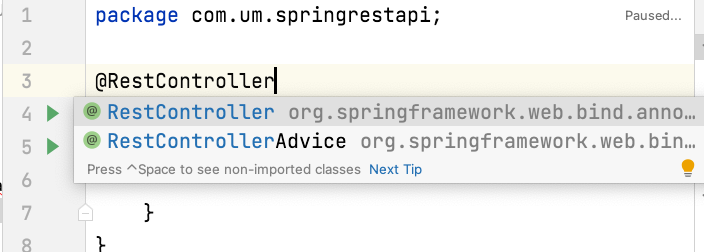
1. Selanjutnya buat sebuah file class HelloWorldController.java





1. Tambahkan annotation **@RestController** jika belum muncul pastikan lagi di dalam pom.xml sudah mengimport **spring-boot-starter-web.** 

* *Annotation* **@RestController** *digunakan untuk memudahkan dalam hal pembuatan Restful API service secara sederhana.*
* *Dalam hal ini class tersebut dapat menerima request dari client dan mengembalikan value nilai ke client berupa data JSON. Dalam memberikan nilai kembalian tidak perlu menentukan format pengembalian nilai karena spring sudah mendukung HTTP Message converter dan Jackson 2 yang ada di classpath project ini secara otomatis digunakan untuk konversi object ke JSON.*
* *Anotasi tersebut seperti penggabungan dari anotasi* ***@controller*** *dan* ***@respondbo***





**HelloController.java**

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

}

1. Tambahkan annotation @RequestMapping([nama path endpoint]) setelah anotasi @RestController

* *Annotation* **@RequestMapping** *digunakan untuk melakukan handling alamat endpoint agar dapat di handle oleh kelas dan method yang telah ditentukan*



**HelloController.java**

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/helloworld")

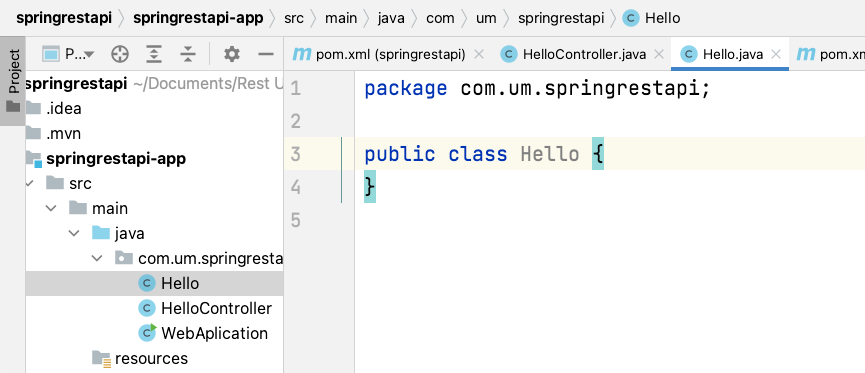
public class HelloController {

}

1. Buat sebuah POJO (Plain Old Java Object) class dengan nama Hello yang digunakan sebagai model response dari simple Rest API yang akan dibuat



* ***POJO*** *adalah kelas java dimana kelas tersebut berdiri sendiri, tidak ketergantungan dengan kelas lain (tidak extends kelas lain, dan tidak implements interface lain)*
* ***POJO*** *ini sendiri sering digunakan sebagai model untuk sebuah data tertentu*
* *POJO umumnya terdiri dari atribut model berupa type dan nama variable atau object beserta metode getter dan setter.*
* *Hal ini dikarenakan atribute variable atau object di dalam pojo bersifat private maka untuk memberikan nilai menggunakan method* ***Setter*** *serta untuk mengembil nilai menggunakan* ***Getter***



1. Selanjutnya isikan atribut dengan tipe String dengan nama hello dan inisial value “Hello World” serta tambahkan method getter dan setter.



**Hello.java**

package com.um.springrestapi;

public class Hello {

String hello = "Hello World";

public String getHello() {

return hello;

}

public void setHello(String hello) {

this.hello = hello;

}

}

1. Tambahkan method getHelloWord dengan tipe return value di dalam method berupa **ResponseEntity** dari sebuah Object POJO Hello yang telah dibuat sebelumnya serta tambahkan annotation @GetMapping.



**HelloController.java**

@RestController

@RequestMapping("/hello")

public class HelloController {

@GetMapping

public ResponseEntity<Hello> getHelloWorld() {

return ResponseEntity.*ok*(new Hello());

}

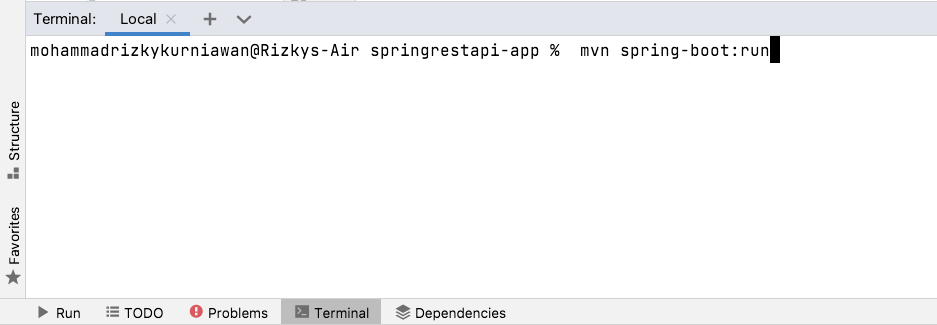
}

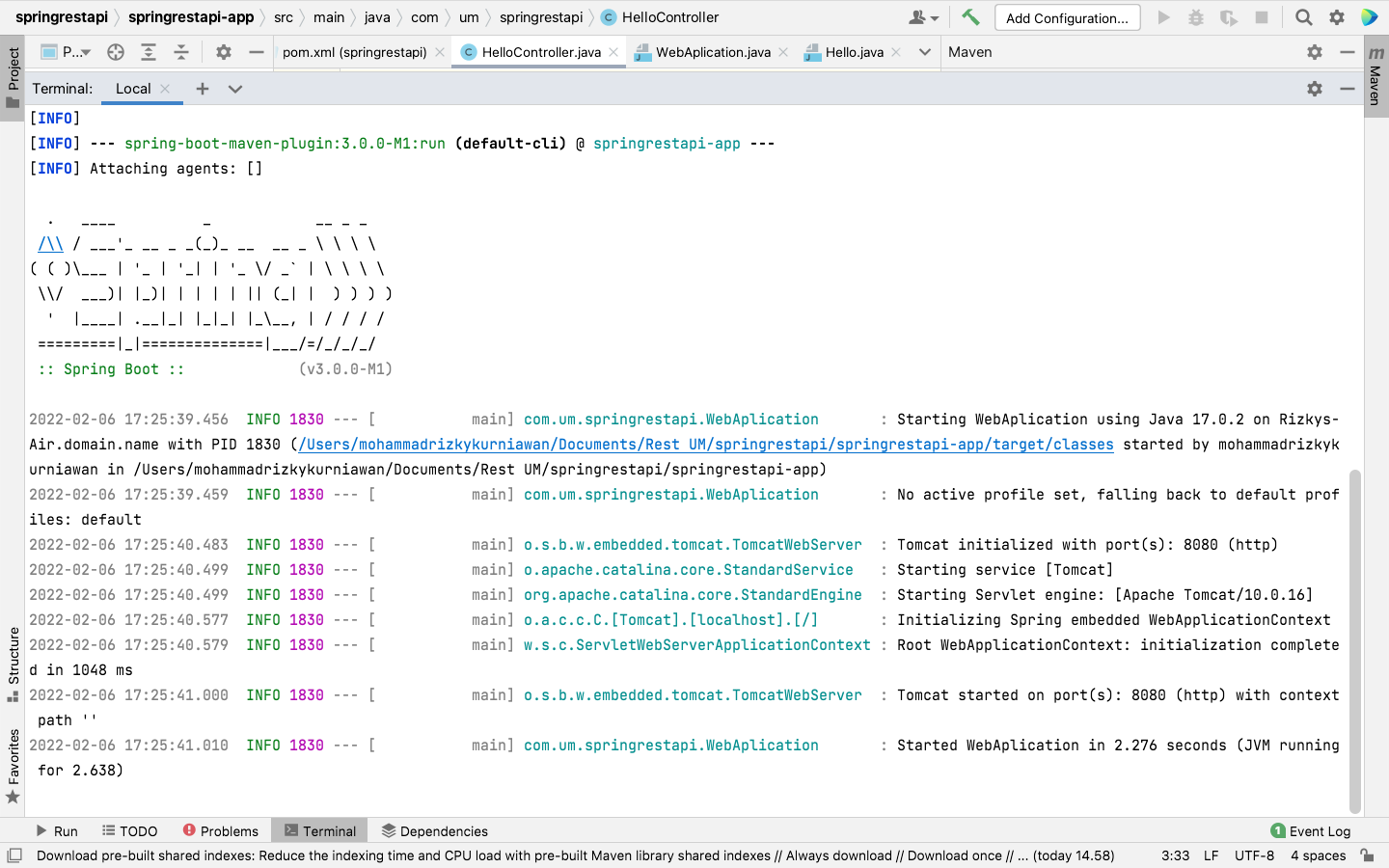
* Pada saat memberikan return value response berupa ResponseEntity panggil dengan method OK dalam arti akan memberikan return Success atau status code 200 beserta initial object POJO Hello.
* Annotation **@GetMapping** mempunyai kesamaan seperti memberikan method annotation @RequestMapping(method = RequestMethod.POST) akan menandakan ketika client melakukan request dengan menggunakan method get sesuai dengan end point pada annotation @RequestMapping maka method yang dibawahnya akan terpanggil. Untuk method lain yang bisa digunakan @PostMapping, @PutMapping, dan @DeleteMapping.
* Sehingga method getHelloWorld akan dapat menghandle request dari client ketika melakukan request dengan metode GET dan endpoint “[base\_url]/hello” dengan memberikan value respond JSON dari POJO Hello yang telah dibuat sebelumnya dengan status OK.

1. Running project spring boot hello Rest API yang telah dibuat. Terdapat 2 cara yang sering digunakan developer untuk melakukan running project yakni
2. **Cara 1** : Menggunakan **Maven Tool Window** di sebalah kanan, pilih module/project yang akan dirunning, lalu pilih plugins, lalu pilih spring:boot, lalu pilih menu spring-boot:run



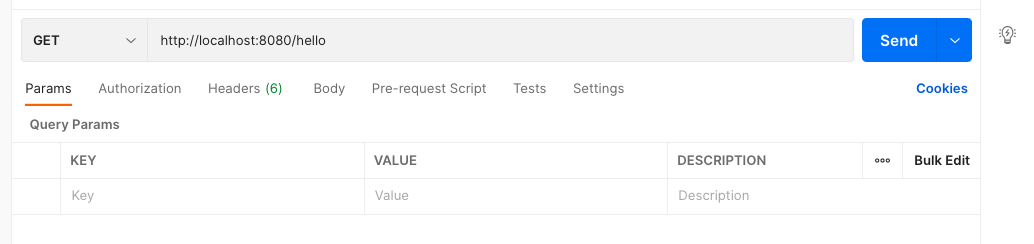
1. Cara 2 : Menggunakan perintah cli maven melalui terminal window dengan mengetikkan **mvn spring-boot:run**

****

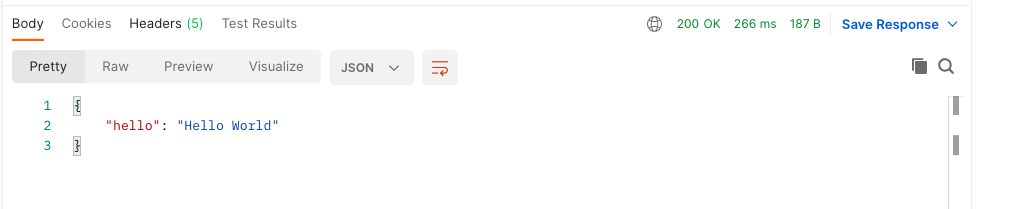
****

Secara default spring boot rest api akan dijalankan di dalam **port 8080** sehingga untuk melakukan test pada rest api spring tersebut menggunakan alamat base URL : [**http://localhost:8080**](http://localhost:8080)

1. Testing simple api yang telah dibuat ke dalam aplikasi test api Postman dengan path “Hello” sehingga url lengkap yang akan dilakukan test adalah Base URL + Path Endpoint yakni [**http://localhost:808**](http://localhost:8080)**0/hello** dengan metode **GET**



Selanjutnya klik tombol Send maka akan tampil respond JSON dan status responde 200 (OK)



# 

## **Rangkuman Modul II**

1. **Spring Initializr** merupakan cara membuat inisiasi project Spring Boot secara sederhana
2. **POM** merupakan singkatan dari Project Object Model. File pom.xml terdiri dari informasi project dan informasi konfigurasi untuk maven dalam melakukan automation build seperti dependency, build directory, source directory, test source directory, plugins, golas, dll.
3. Terdapat annotation penting dalam pembuatan rest api spring boot sederhana yakni
4. **@RestController** digunakan untuk menunjukkan bahwa class yang akan digunakan untuk Rest API sebagai controller atau yang berhubungan dengan client
5. **@RequestMapping** digunakan untuk melakukan routing agar spring dapat melakukan pemetaan atas web request agar dapat di handle oleh kelas dan method yang telah ditentukan
6. **@GetMapping** akan memanggil method yang diidentifikasi ketika client melakukan request dengan menggunakan method get. Untuk method lain yang bisa digunakan @PostMapping, @PutMapping, dan @DeleteMapping



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

1. Buat sebuah Rest API sederhana Hello seperti materi di modul II
2. Buatlah sebuah endpoint baru dengan ketentuan berikut
   1. Method Post dengan alamat [http://localhost:8080/](http://localhost:8080/hy)hello dengan memberikan sebuah response dengan status 200

{

“hello” : “post world”

}

* 1. Method Delete denga alamat [http://localhost:8080/](http://localhost:8080/hy)hello dengan memberikan sebuah response dengan status 200 namun tidak perlu memberikan data response
  2. Method Put denga alamat [http://localhost:8080/](http://localhost:8080/hy)hello dengan memberikan sebuah response dengan status 200

{

“hello” : “put world”

}

# 

## **Tugas Pilihan Ganda (Pengetahuan) Modul II**

1. Buka Link Berikut (Token : 222)

[**https://forms.gle/q9JrLotZXfXkHos18**](https://forms.gle/q9JrLotZXfXkHos18)

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