[개정판] Spring Boot를 이용한 RESTful API 개발

{ REST API } + Spring Boot

목차

- Section 0: Web Service & Web Application
- **Section 1: Spring Boot로 개발하는 RESTful API**
- Section 2: User Service API 추가
- Section 3: RESTful Service 기능 확장
- Section 4: Spring Boot API 사용
- Section 5: JPA 사용
- Section 6: REST API 설계 가이드

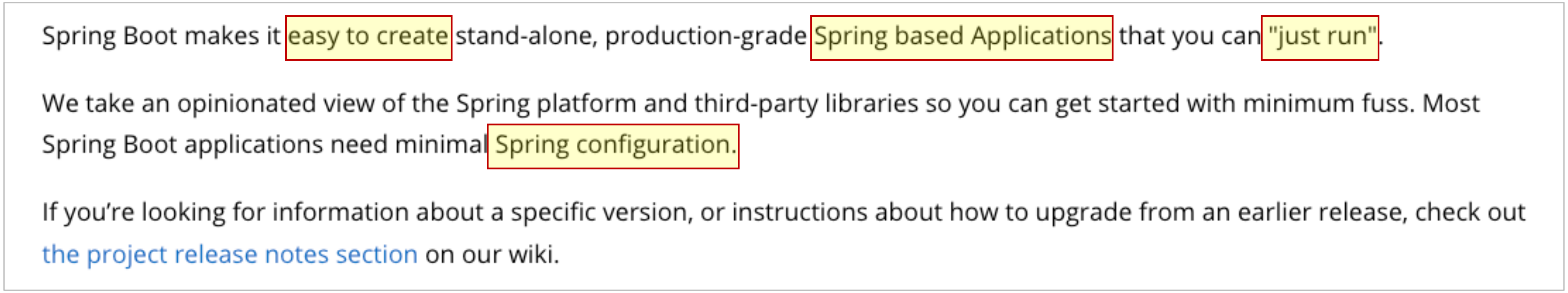
1. Section

Spring Boot로 개발하는 RESTful API

- Spring Boot 개요
- REST API 설계
- Spring Boot Project 생성, 실행
- HelloWorld Controller 추가
- HelloWorld Bean 추가
- DispatcherServlet, 프로젝트 동작 이해하기
- Path Variable

Spring Boot

- Spring Boot



Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

If you're looking for information about a specific version, or instructions about how to upgrade from an earlier release, check out the project release notes section on our wiki.

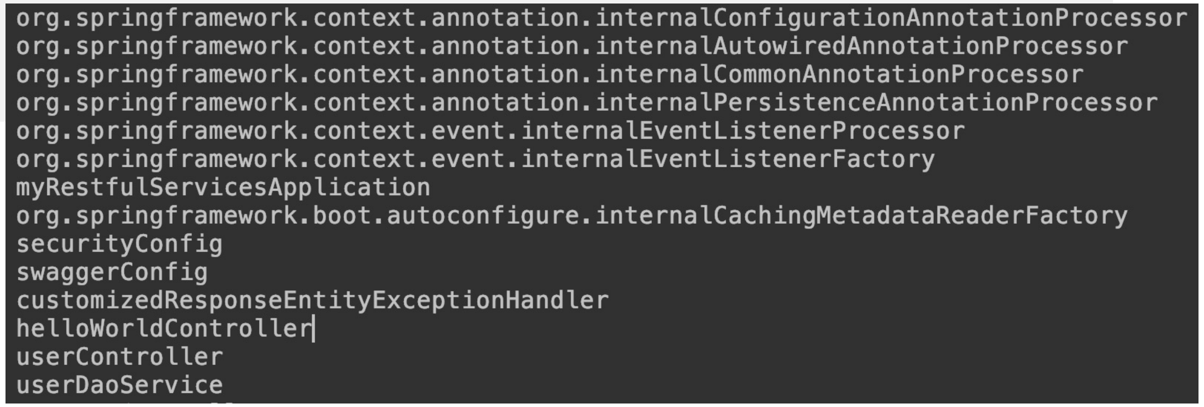
- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' component to simplify your build configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

https://start.spring.io/

Spring Boot

1) Spring Boot Application
2) Auto Configuration
3) Component Scan

```java
@SpringBootApplication
public class MyRestfulServicesApplication {

  public static void main(String[] args) {
    ApplicationContext application =  SpringApplication.run(MyRestfulServicesApplication.class, args);

    for (String str : application.getBeanDefinitionNames()) {
      System.out.println(str);
    }
  }
```



```
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.annotation.internalPersistenceAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
myRestfulServicesApplication
org.springframework.boot.autoconfigure.internalCachingMetadataReaderFactory
securityConfig
swaggerConfig
customizedResponseEntityExceptionHandler
helloWorldController
userController
userDaoService
```

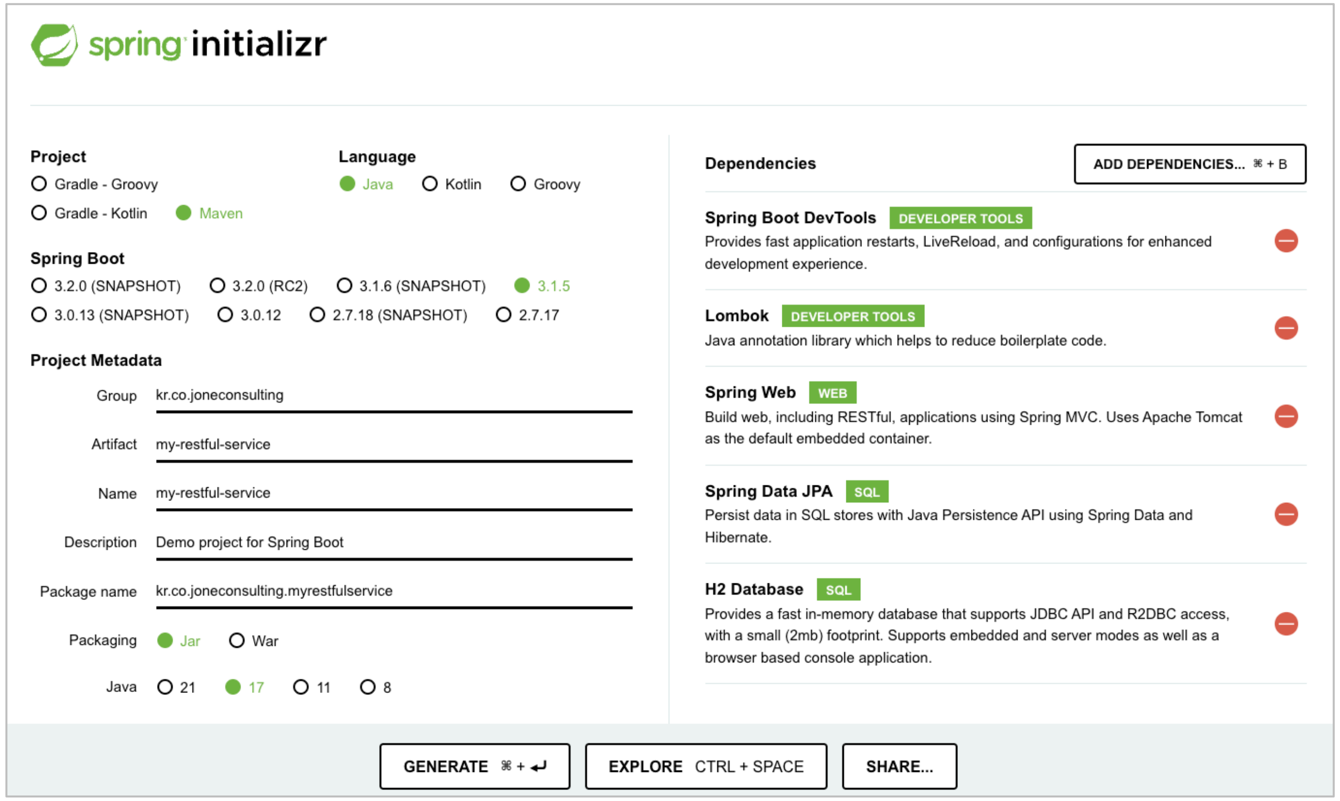
My RESTful Services

- REST overview
- RESTful Web Services
- Social Media Application
  - User → Posts

Description	REST API	HTTP Method
Retrieve all Users	/users	GET
Create a User	/users	POST
Retrieve one User	/users/{id}	GET
Delete a User	/users/{id}	DELETE
Retrieve all posts for a User	/users/{id}/posts	GET
Create a posts for a User	/users/{id}/posts	POST
Retrieve details of a User	/users/{id}/posts/{post_id}	GET

Initializing a RESTful Services Project

- 사전준비
  - JDK 1.8 +
  - IntelliJ IDEA Ultimate(https://www.jetbrains.com/idea/)
  - Postman(https://www.postman.com/downloads/) or curl(https://curl.haxx.se/windows/)
- https://start.spring.io/
- Dependencies
  - DevTools
  - Lombok
  - Web
  - JPA
  - H2



spring initializr

Project
Gradle - Groovy | Gradle - Kotlin | [✓ Maven]

Language
[✓ Java] | Kotlin | Groovy

Spring Boot
3.2.0 (SNAPSHOT) | 3.2.0 (RC2) | 3.1.6 (SNAPSHOT) | [✓ 3.1.5] | 3.0.13 (SNAPSHOT) | 3.0.12 | 2.7.18 (SNAPSHOT) | 2.7.17

Project Metadata

Group	kr.co.joneconsulting
Artifact	my-restful-service
Name	my-restful-service
Description	Demo project for Spring Boot
Package name	kr.co.joneconsulting.myrestfulservice
Packaging	[✓ Jar] War
Java	21 [✓ 17] 11 8

Dependencies ADD DEPENDENCIES... ⌘ + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

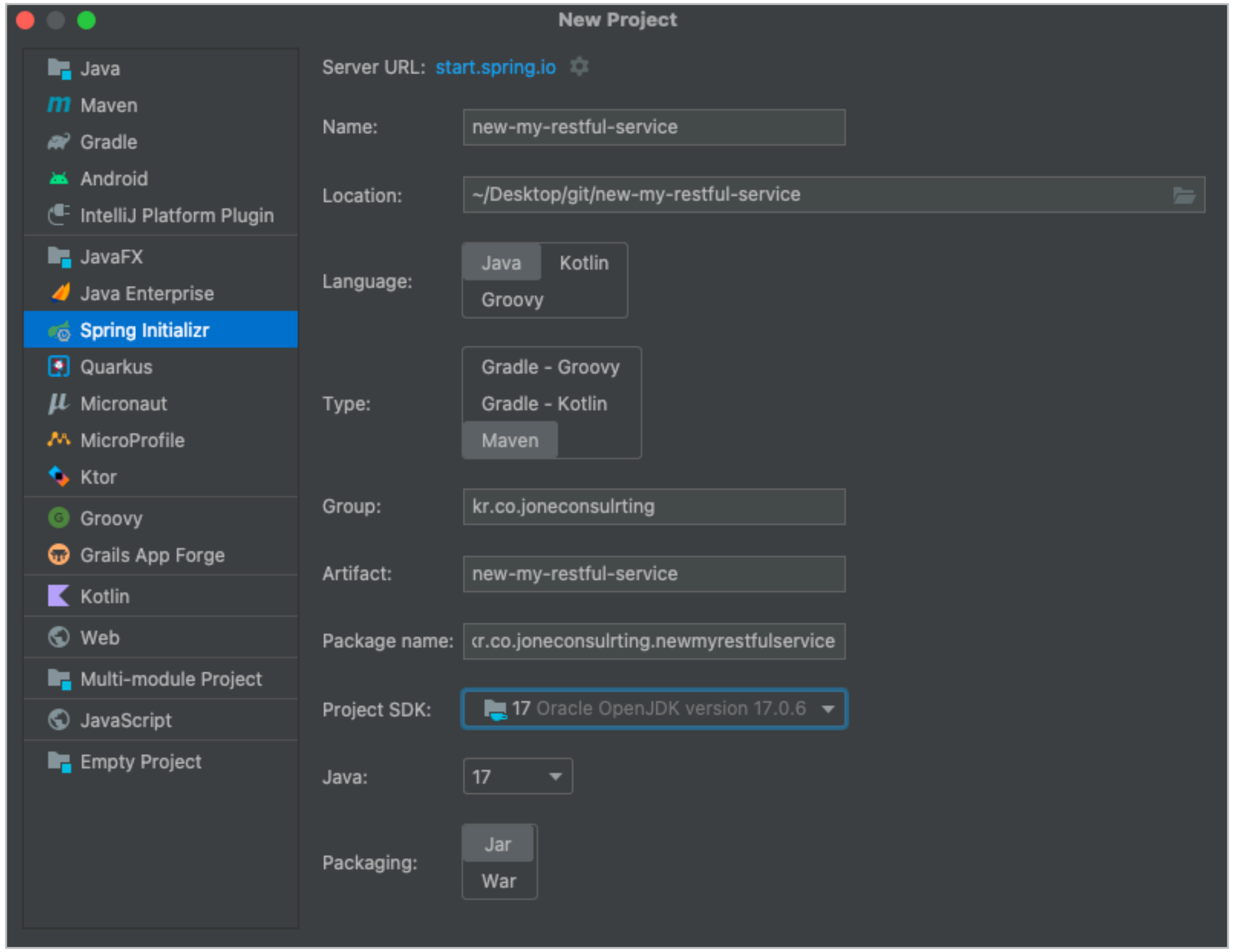
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Initializing a RESTful Services Project

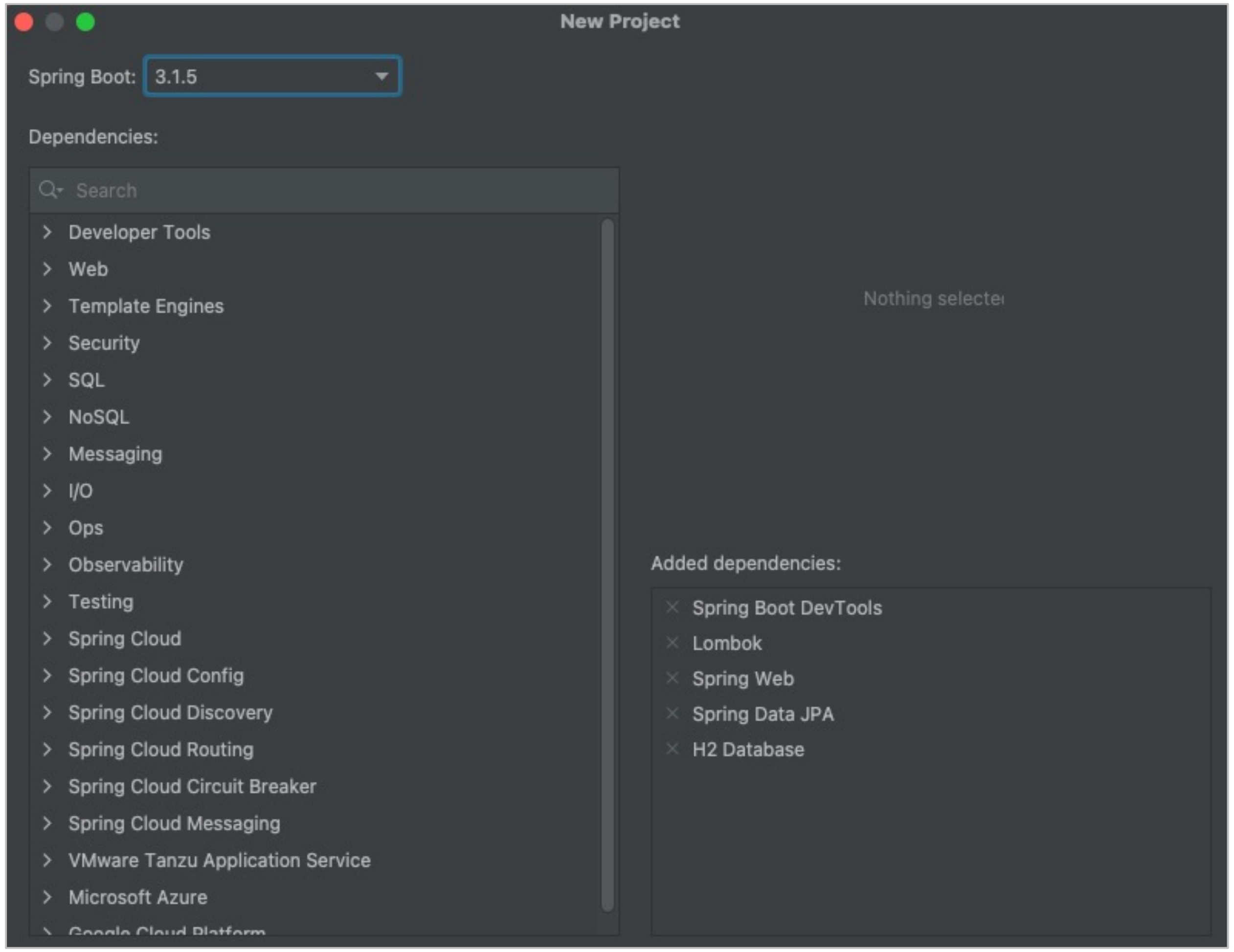
- IntelliJ IDEA > New Project > Spring Initializr > Spring project 생성



New Project

Java
Maven
Gradle
Android
IntelliJ Platform Plugin
JavaFX
Java Enterprise
Spring Initializr
Quarkus
Micronaut
MicroProfile
Ktor
Groovy
Grails App Forge
Kotlin
Web
Multi-module Project
JavaScript
Empty Project

Server URL: start.spring.io
Name: new-my-restful-service
Location: ~/Desktop/git/new-my-restful-service
Language: Java Kotlin Groovy
Type: Gradle - Groovy Gradle - Kotlin Maven
Group: kr.co.joneconsulrting
Artifact: new-my-restful-service
Package name: kr.co.joneconsulrting.newmyrestfulservice
Project SDK: 17 Oracle OpenJDK version 17.0.6
Java: 17
Packaging: Jar War



New Project

Spring Boot: 3.1.5

Dependencies:

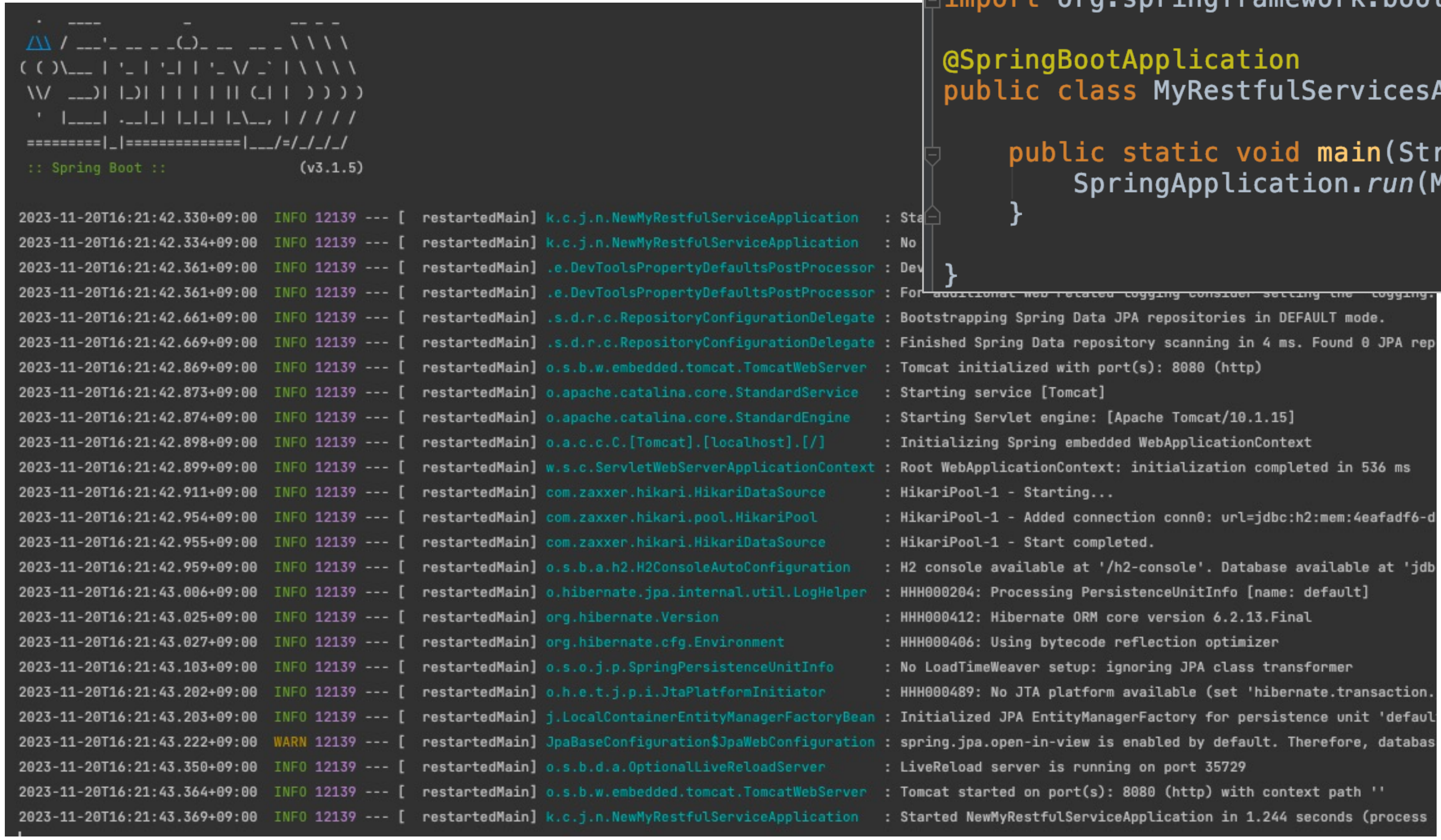
Search
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Messaging
VMware Tanzu Application Service
Microsoft Azure

Nothing selecte

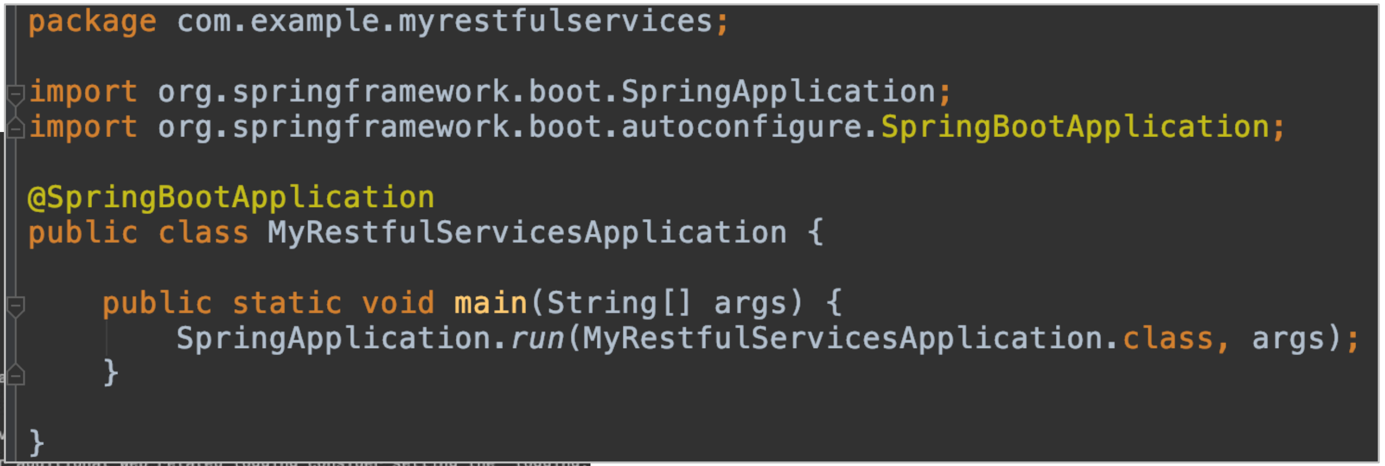
Added dependencies:
Spring Boot DevTools
Lombok
Spring Web
Spring Data JPA
H2 Database

Initializing a RESTful Services Project

- Spring project 실행
  - Embedded tomcat 실행



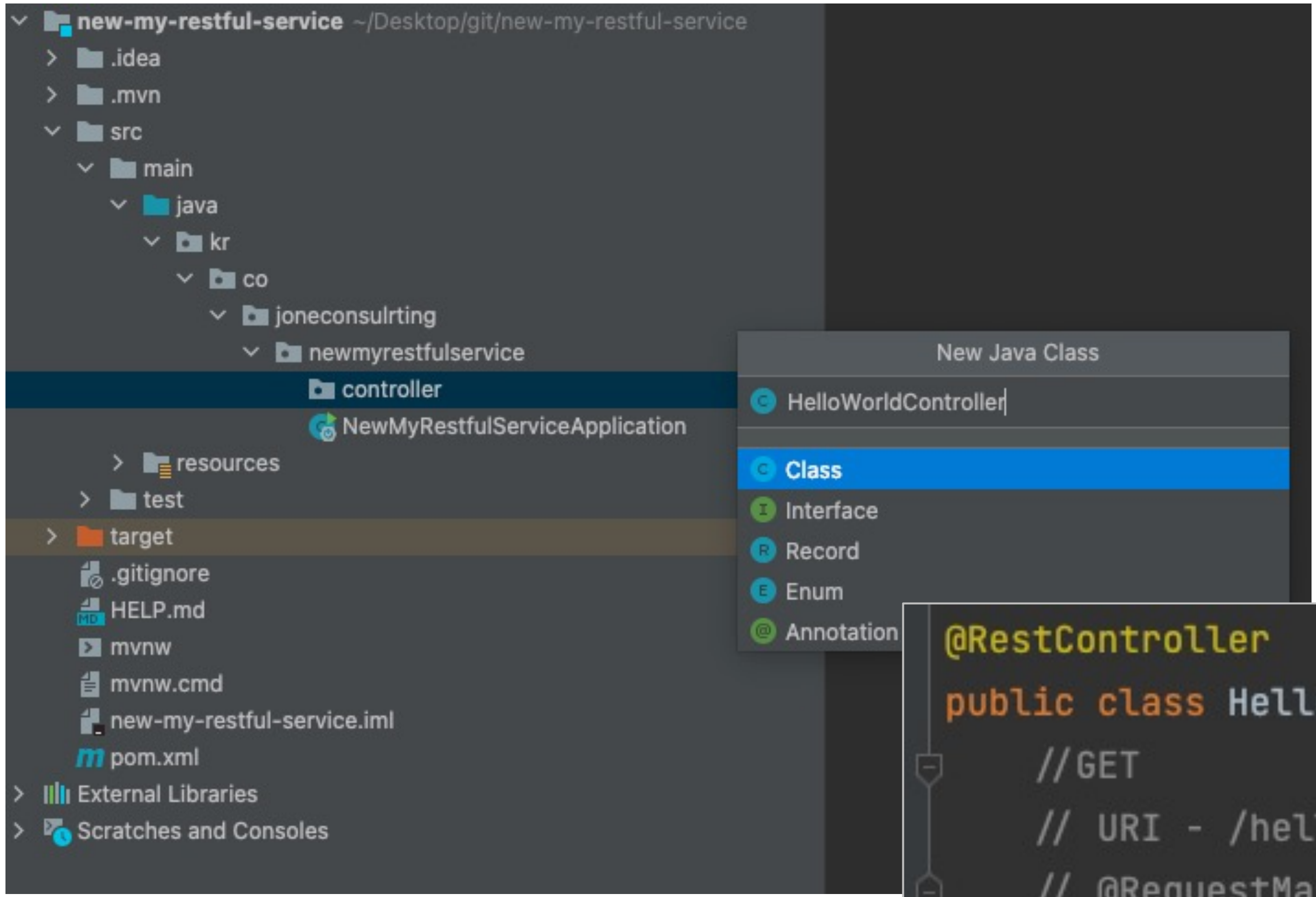
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.5)

2023-11-20T16:21:42.330+09:00  INFO 12139 --- [  restartedMain] k.c.j.n.NewMyRestfulServiceApplication   : Sta
2023-11-20T16:21:42.334+09:00  INFO 12139 --- [  restartedMain] k.c.j.n.NewMyRestfulServiceApplication   : No
2023-11-20T16:21:42.361+09:00  INFO 12139 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Dev
2023-11-20T16:21:42.361+09:00  INFO 12139 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For
2023-11-20T16:21:42.661+09:00  INFO 12139 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-11-20T16:21:42.669+09:00  INFO 12139 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 4 ms. Found 0 JPA rep
2023-11-20T16:21:42.869+09:00  INFO 12139 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-11-20T16:21:42.873+09:00  INFO 12139 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-11-20T16:21:42.874+09:00  INFO 12139 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.15]
2023-11-20T16:21:42.898+09:00  INFO 12139 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-11-20T16:21:42.899+09:00  INFO 12139 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 536 ms
2023-11-20T16:21:42.911+09:00  INFO 12139 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2023-11-20T16:21:42.954+09:00  INFO 12139 --- [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:4eafadf6-d
2023-11-20T16:21:42.955+09:00  INFO 12139 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2023-11-20T16:21:42.959+09:00  INFO 12139 --- [  restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Database available at 'jdb
2023-11-20T16:21:43.006+09:00  INFO 12139 --- [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-11-20T16:21:43.025+09:00  INFO 12139 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.2.13.Final
2023-11-20T16:21:43.027+09:00  INFO 12139 --- [  restartedMain] org.hibernate.cfg.Environment            : HHH000406: Using bytecode reflection optimizer
2023-11-20T16:21:43.103+09:00  INFO 12139 --- [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2023-11-20T16:21:43.202+09:00  INFO 12139 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.
2023-11-20T16:21:43.203+09:00  INFO 12139 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'defaul
2023-11-20T16:21:43.222+09:00  WARN 12139 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, databas
2023-11-20T16:21:43.350+09:00  INFO 12139 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2023-11-20T16:21:43.364+09:00  INFO 12139 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2023-11-20T16:21:43.369+09:00  INFO 12139 --- [  restartedMain] k.c.j.n.NewMyRestfulServiceApplication   : Started NewMyRestfulServiceApplication in 1.244 seconds (process
```

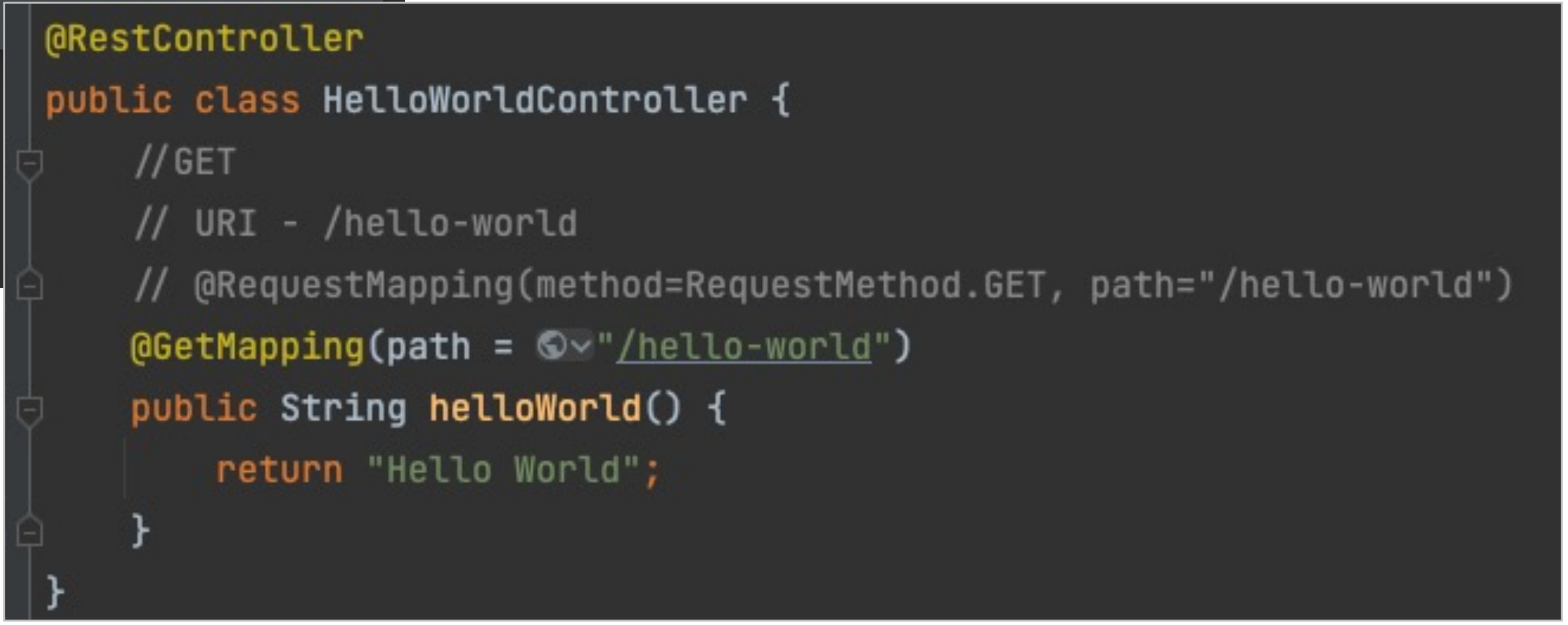


```java
package com.example.myrestfulservices;

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

@SpringBootApplication
public class MyRestfulServicesApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyRestfulServicesApplication.class, args);
    }

}
```

Creating a HelloWorld Service

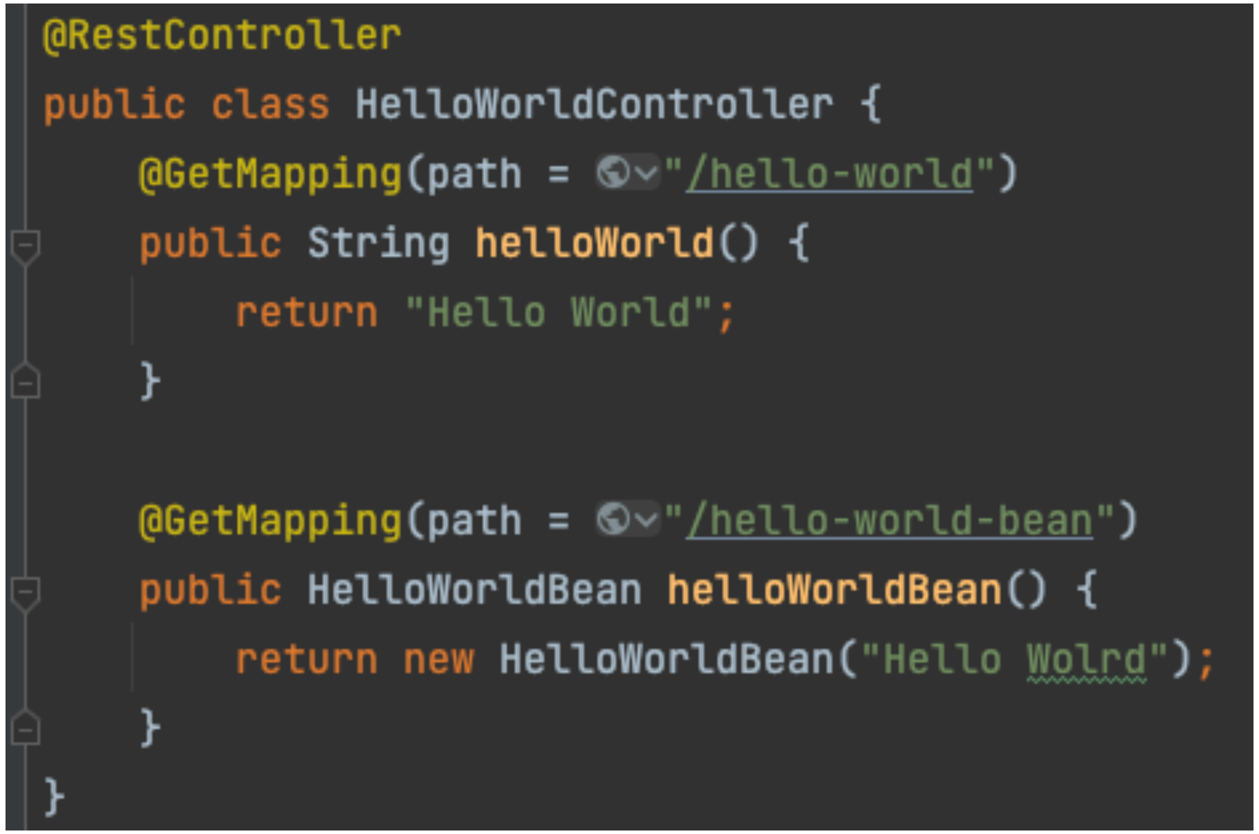
njone company



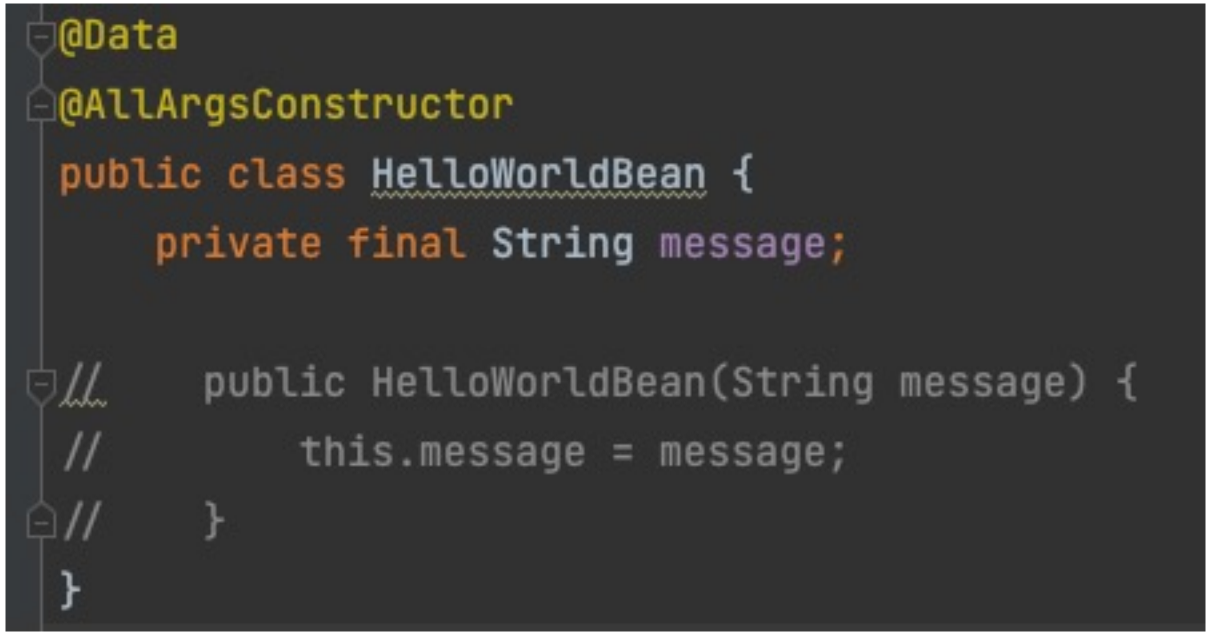


```java
@RestController
public class HelloWorldController {
    //GET
    // URI - /hello-world
    // @RequestMapping(method=RequestMethod.GET, path="/hello-world")
    @GetMapping(path = "/hello-world")
    public String helloWorld() {
        return "Hello World";
    }
}
```

Change the HelloWorld Service



```java
@RestController
public class HelloWorldController {
    @GetMapping(path = "/hello-world")
    public String helloWorld() {
        return "Hello World";
    }

    @GetMapping(path = "/hello-world-bean")
    public HelloWorldBean helloWorldBean() {
        return new HelloWorldBean("Hello Wolrd");
    }
}
```



```java
@Data
@AllArgsConstructor
public class HelloWorldBean {
    private final String message;

//    public HelloWorldBean(String message) {
//        this.message = message;
//    }
}
```

- *execute project*

Spring Boot Configuration

- Spring Boot 동작 원리
- Spring Boot의 설정파일: application.yml or application.properties

```
logging.level.org.springframework = debug
```

application.properties → 설정이름=값

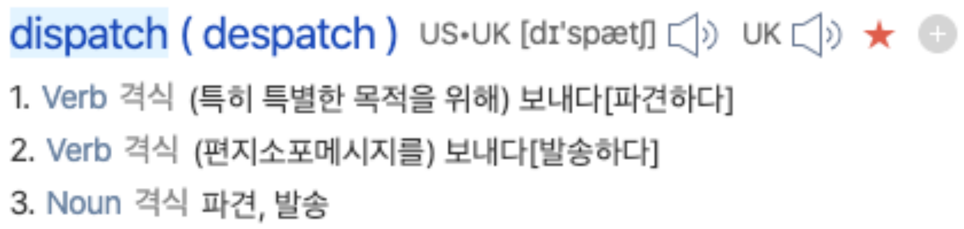
```
logging:
  level:
    org.springframework: debug
```

application.yml → 설정이름:값

- Spring Boot Auto Configuration
  - DispatcherServletAutoConfiguration
  - ErrorMvcAutoConfiguration
  - HttpMessageConvertersAutoConfiguration → JSON convert

Dispatcher Servlet

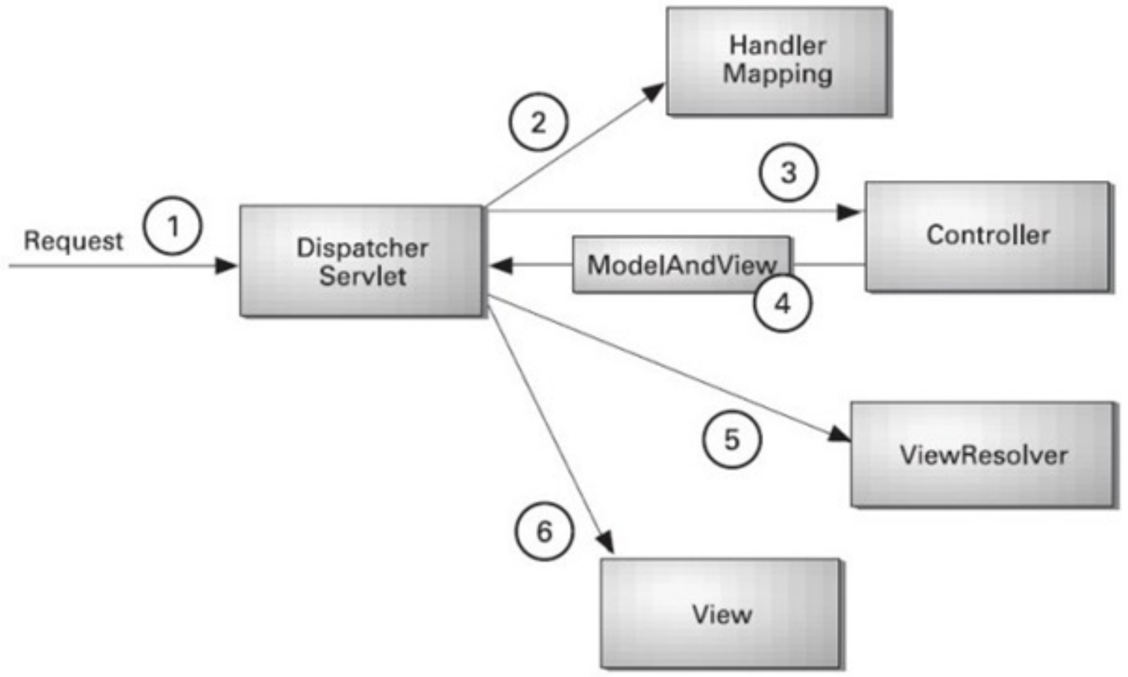
- DispatcherServlet → '/'



dispatch (despatch) US•UK [dɪ'spætʃ] UK ★

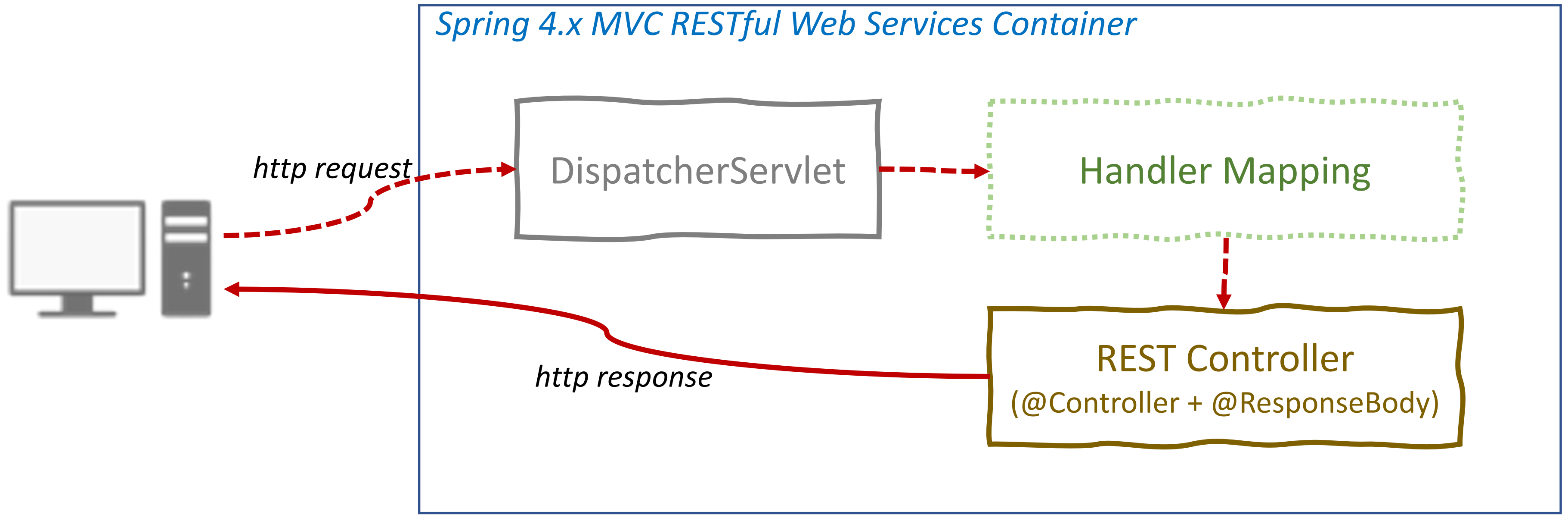
1. Verb 격식 (특히 특별한 목적을 위해) 보내다[파견하다]
2. Verb 격식 (편지소포메시지를) 보내다[발송하다]
3. Noun 격식 파견, 발송

- 클라이언트의 모든 요청을 한곳으로 받아서 처리
- 요청에 맞는 Handler로 요청을 전달
- Handler의 실행 결과를 Http Response 형태로 만들어서 반환



RestController

- RestController
  - Spring4부터 @RestController 지원
  - @Controller + @ResponseBody
  - View를 갖지 않는 REST Data(JSON/XML)를 반환


Spring 4.x MVC RESTful Web Services Container
http request
DispatcherServlet
Handler Mapping
REST Controller
(@Controller + @ResponseBody)
http response

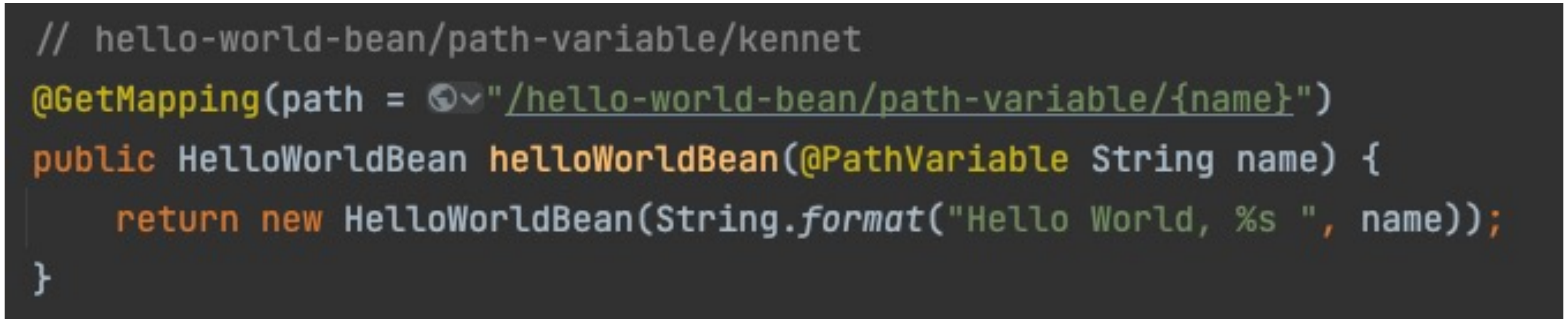
Path Variable

- Path Variable

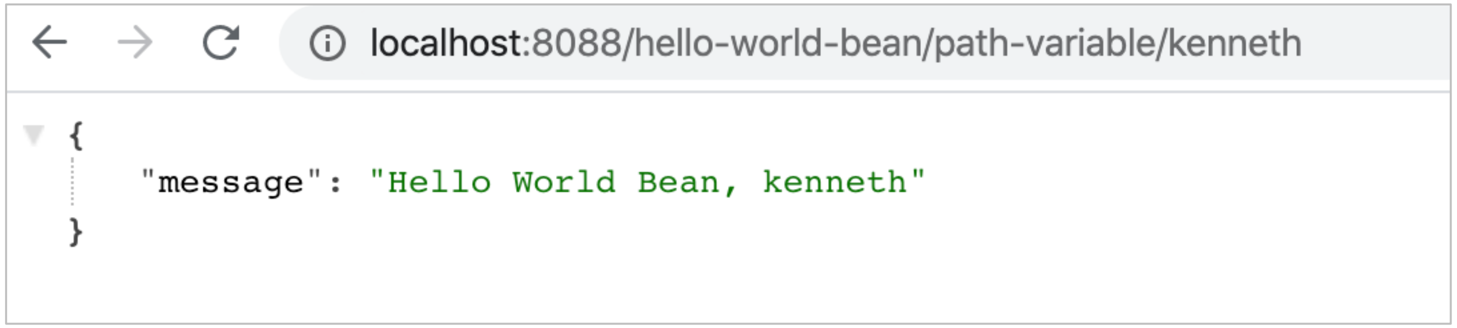
http://localhost:8080/books/

http://localhost:8080/books/1 or http://localhost:8080/books/123

http://localhost:8080/books/**{book_id}**



```java
// hello-world-bean/path-variable/kennet
@GetMapping(path = "/hello-world-bean/path-variable/{name}")
public HelloWorldBean helloWorldBean(@PathVariable String name) {
    return new HelloWorldBean(String.format("Hello World, %s ", name));
}
```



localhost:8088/hello-world-bean/path-variable/kenneth

```
{
    "message": "Hello World Bean, kenneth"
}
```