



Spring Boot Basic

스프링 부트 기초

스프링 부트 퀵 스타트



한국기술교육대학교
온라인평생교육원

학습내용

- 스프링 부트 프로젝트 구조 이해
- 스프링 부트 설정(Properties, Yaml) 이해
- 웹 애플리케이션 작성

학습목표

- 스프링 부트로 만든 프로젝트의 구조를 설명할 수 있다.
- Properties와 Yaml 파일을 통해 스프링 부트 프로젝트를 제어할 수 있다.
- REST 방식의 웹 애플리케이션을 작성할 수 있다.



스프링 부트 프로젝트 구조 이해

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

① 소스 폴더

src/main/java

- 일반적인 자바 클래스 작성

src/main/resources

- 자바가 아닌 파일(XML 또는 Properties) 작성

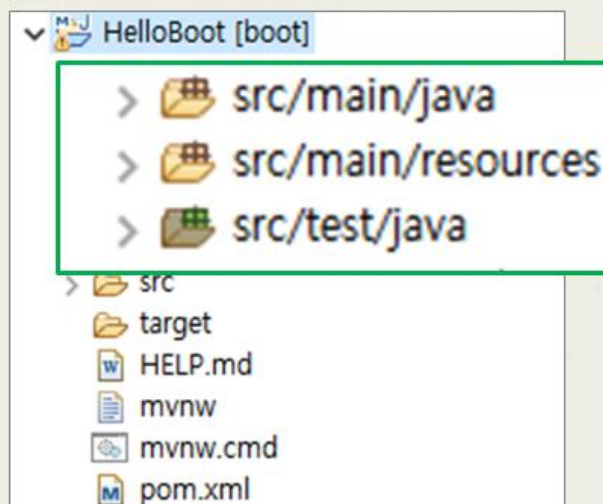
src/test/java

- Junit 기반의 테스트 코드 작성

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

① 소스 폴더





스프링 부트 프로젝트 구조 이해

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

② src/main/resources 소스 폴더

static 폴더

- 이미지나 HTML 같은 정적인 웹 콘텐츠 저장

templates 폴더

- 타임리프(Thymleaf)같은 템플릿 기반의 웹 리소스 저장

application.properties 파일

- 프로젝트 전체에서 사용할 프로퍼티 정보 저장

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

③ Maven Dependencies

- Maven에 의해 프로젝트 Path에 등록된 라이브러리 목록
- pom.xml 파일에 <dependency>를 추가하여 라이브러리 다운로드 가능

라이브러리 다운로드 물리적 위치

- “C:\Users\컴퓨터이름\m2\repository”

스프링 부트 프로젝트 구조 이해

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

③ Maven Dependencies

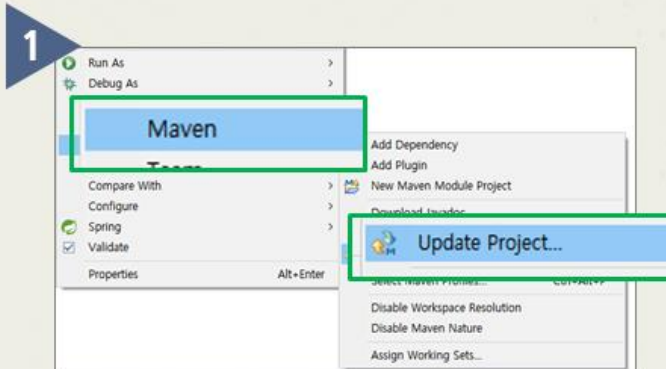
- 라이브러리 다운로드 과정에서 문제가 생긴 경우
Maven → Update Project...를 실행

스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

③ Maven Dependencies

- 라이브러리 다운로드 과정에서 문제가 생긴 경우
Maven → Update Project...를 실행



스프링 부트 프로젝트 구조 이해

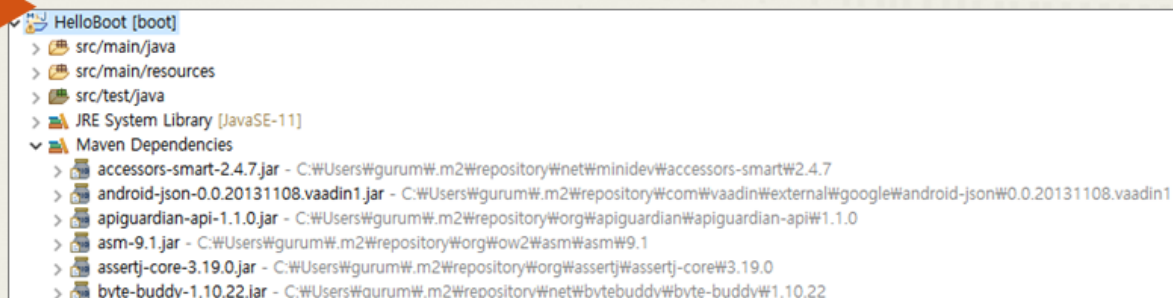
스프링 부트 프로젝트 구조 이해

1 스프링 부트 프로젝트 구조

③ Maven Dependencies

- 라이브러리 다운로드 과정에서 문제가 생긴 경우
Maven → Update Project...를 실행

2

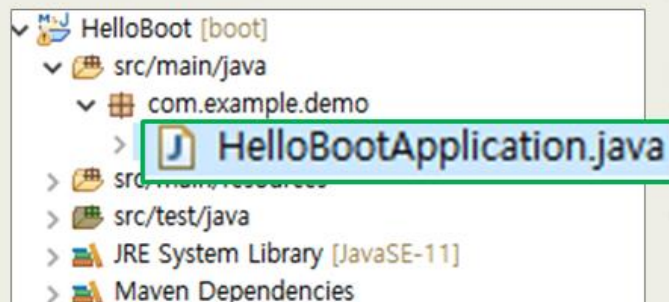


스프링 부트 프로젝트 구조 이해

2 스프링 부트 애플리케이션 실행하기

① 웹 애플리케이션으로 실행

- 프로젝트 생성 시 지정한 Package에 “프로젝트이름 + Application” 형태의 자바 애플리케이션이 있음






스프링 부트 프로젝트 구조 이해

스프링 부트 프로젝트 구조 이해

2 스프링 부트 애플리케이션 실행하기

① 웹 애플리케이션으로 실행

- 스프링 부트 애플리케이션은 기본적으로 웹 애플리케이션으로 실행되며, 내장 톰캣이 8080 포트로 구동됨
- `WebApplicationType`을 `NONE`으로 설정하여 일반 자바 애플리케이션으로 실행할 수도 있음



```

:: Spring Boot ::
(v2.5.1)

2021-06-22 19:32:32.855 INFO 7412 --- [main] com.example.demo.HelloBootApplication : Starting HelloBootApplication using Java 15.0.2 on
2021-06-22 19:32:32.857 INFO 7412 --- [main] com.example.demo.HelloBootApplication : No active profile set, falling back to default prof
2021-06-22 19:32:33.473 INFO 7412 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-06-22 19:32:33.483 INFO 7412 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache/2.5.18 (Ubuntu)]
2021-06-22 19:32:33.483 INFO 7412 --- [main] org.apache.catalina.core.StandardContext : Initializing Spring WebApplicationFactory: org.s
2021-06-22 19:32:33.548 INFO 7412 --- [main] o.a.c.c.C.[Tomcat] : Initializing Tomcat
2021-06-22 19:32:33.549 INFO 7412 --- [main] w.s.c.ServletWebServer : Initializing ServletWebServer
2021-06-22 19:32:33.838 INFO 7412 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http)
2021-06-22 19:32:33.846 INFO 7412 --- [main] com.example.demo.HelloBootApplication : Started HelloBootApplication in 1.265 seconds (JVM
  
```



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

① application.properties 파일 작성

- 스프링 부트는 애플리케이션에서 사용하는 설정 정보를 외부 프로퍼티로 분리시킴으로써 자바 소스 수정을 최소화 함

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

① application.properties 파일 작성

application.properties

- 애플리케이션의 환경을 관리하는 설정 파일

```
## WebApplicationType 설정
spring.main.web-application-type=none
```




스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

② 프로퍼티 적용 순서

- 프로퍼티 설정은 자바 소스보다 우선순위가 높음
- 자바 소스에서 `WebApplicationType`을 `SERVLET`으로 설정해도 `properties`에서 `NONE`으로 설정하면 자바 애플리케이션으로 동작함

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

③ 애플리케이션 배너

1 애플리케이션 배너 제거

- 자바 소스를 이용하여 배너 제거

```
@SpringBootApplication
public class HelloBootApplication {

    public static void main(String[] args) {
        SpringApplication application =
            new SpringApplication(HelloBootApplication.class);
        application.setWebApplicationType(WebApplicationType.SERVLET);
        application.setBannerMode(Banner.Mode.OFF);
        application.run(args);
    }
}
```



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

③ 애플리케이션 배너

1 애플리케이션 배너 제거

- 외부 프로퍼티 설정을 통해 배너를 제거할 수도 있음

```
## WebApplicationType 설정
spring.main.web-application-type=none

# Application Banner 설정
spring.main.banner-mode=off
```

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

③ 애플리케이션 배너

2 사용자 정의 배너

- 스프링 부트는 src/main/resources에 banner.txt 파일이 없으면 기본 배너를 출력



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

③ 애플리케이션 배너

2 사용자 정의 배너

- 사용자 정의 배너는 src/main/resources 소스 폴더에 banner.txt 파일로 작성
- 배너 파일에는 외부 프로퍼티를 사용할 수 있음

banner.txt

```
#####
SPRING BOOT START ^^
${spring-boot.formatted-version}
#####
```

스프링 부트 설정 (Properties, Yaml) 이해

1 외부 프로퍼티 이용하기

③ 애플리케이션 배너

2 사용자 정의 배너

- 배너 파일의 이름과 위치를 변경할 수 있음

application.properties

```
## WebApplicationType 설정
spring.main.web-application-type=none

# Application Banner 설정
spring.main.banner-mode=console
spring.banner.location=banner/custom_banner.txt
```



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

2 프로퍼티 파일과 프로퍼티 객체

① 서버 포트 변경

- 프로퍼티 파일을 이용하면 내장 톰캣 서버의 포트 번호를 변경할 수 있음
- 포트 번호를 -1로 설정하면 랜덤한 포트가 할당됨

```
# WebApplicationType 설정
spring.main.web-application-type=servlet

# Application Banner 설정
spring.main.banner-mode=console
spring.banner.location=banner/custom_banner.txt

# Tomcat Server Port 변경
server.port=8888
```

스프링 부트 설정 (Properties, Yaml) 이해

2 프로퍼티 파일과 프로퍼티 객체

② 프로퍼티 클래스

- application.properties에 설정한 프로퍼티 정보는 실제 애플리케이션이 동작할 때 프로퍼티 객체에 매핑됨



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

2 프로퍼티 파일과 프로퍼티 객체

② 프로퍼티 클래스

- server로 시작하는 프로퍼티 설정이 매핑되는 ServerProperties 클래스의 일부 코드 발췌

```
package org.springframework.boot.autoconfigure.web;

import org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties(prefix = "server", ignoreUnknownFields = true)
public class ServerProperties {

    /**
     * Server HTTP port.
     */
    private Integer port;

    public Integer getPort() {
        return this.port;
    }

    public void setPort(Integer port) {
        this.port = port;
    }

}
```

스프링 부트 설정 (Properties, Yaml) 이해

3 Yaml 파일

- ‘야물’이라고도 불림
- XML이나 JSON과 마찬가지로 데이터의 의미와 구조를 쉽게 전달하기 위한 파일
- 기존의 XML이나 JSON 보다 쉽게 작성할 수 있고 가독성이 뛰어나



스프링 부트 설정 (Properties, Yaml) 이해

스프링 부트 설정 (Properties, Yaml) 이해

3 Yaml 파일

- Properties 설정을 Yaml 파일로 대체할 수 있음
- src/main/resources 소스 폴더에 있는 application.properties 파일 제거 후 application.yml 파일로 대체함(들여쓰기에 주의)

```
# WebApplicationType 설정
spring:
  main:
    web-application-type: none

# Application Banner 설정
banner-mode: off

# Tomcat Server Port 변경
server:
  port: 8888
```



웹 애플리케이션 작성

웹 애플리케이션 작성

1 REST 컨트롤러

① 컨트롤러 빈 등록

- @RestController를 사용하여 REST 컨트롤러를 구현
- 컨트롤러 클래스는 반드시 main 클래스의 하위 패키지에 위치해야 함

웹 애플리케이션 작성

1 REST 컨트롤러

① 컨트롤러 빈 등록

- @GetMapping은 GET 방식의 요청을 처리
- 리턴한 데이터를 브라우저에 그대로 전달하기 때문에 별도로 View 화면을 만들 필요가 없음

```
package com.example.demo.web;

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

@RestController
public class BoardController {
    public BoardController() {
        System.out.println("==> BoardController 생성");
    }

    @GetMapping("/hello")
    public String hello(String name) {
        return "Hello : " + name;
    }
}
```



웹 애플리케이션 작성

웹 애플리케이션 작성

1 REST 컨트롤러

② REST 컨트롤러 구현

- 각 웹 요청 방식에 따른 어노테이션을 사용하여 메소드를 구현함

@GetMapping • GET 요청 처리(SELECT)

@PostMapping • POST 요청 처리(INSERT)

@PutMapping • PUT 요청 처리(UPDATE)

@DeleteMapping • DELETE 요청 처리(DELETE)

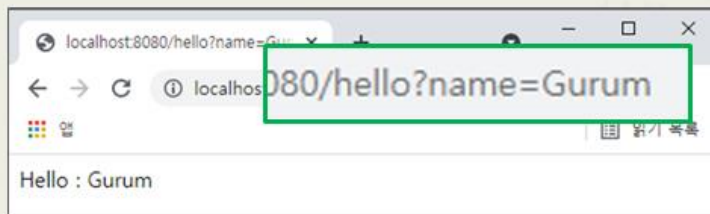
웹 애플리케이션 작성

1 REST 컨트롤러

③ REST 컨트롤러 실행

1 WebApplicationType을 Servlet으로 설정하고 웹 애플리케이션으로 실행

2 브라우저에서 GET 방식으로 hello?name=Gurum 요청 전송





웹 애플리케이션 작성

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

① 자바 객체 리턴하기

- @RestController가 VO(Value Object)나 컬렉션 형태의 객체를 리턴하는 경우는 자동으로 JSON으로 변환되어 전송됨

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

① 자바 객체 리턴하기

```
1 package com.example.demo.web;

import java.util.Date;

public class BoardVO {
    private int seq;
    private String title;
    private String writer;
    private String content;
    private Date createDate = new Date();
    private int cnt = 0;

    // Getter/Setter Method
    // toString Method
}
```



웹 애플리케이션 작성

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

① 자바 객체 리턴하기

2

```
@GetMapping("/getBoard")
public BoardVO getBoard() {
    BoardVO board = new BoardVO();
    board.setSeq(1);
    board.setTitle("테스트 제목...");
    board.setWriter("테스터");
    board.setContent("테스트 내용입니다.....");
    board.setCreateDate(new Date());
    board.setCnt(0);
    return board;
}
```

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

② Lombok 적용하기

- VO 클래스를 작성할 때 Lombok을 이용할 수 있음

@Getter • Getter 메소드를 Generation함

@Setter • Setter 메소드를 Generation함

@ToString • toString 메소드를 Generation함

@Data • @Getter, @Setter, @ToString 등을 모두 포함함



웹 애플리케이션 작성

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

② Lombok 적용하기

- BoardVO 객체를 리턴하는 메소드 실행 결과

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

② Lombok 적용하기

참고 크롬 브라우저에 JsonView를 설치했을 때의 결과

```

{
  seq: 1,
  title: "테스트 제목...",
  writer: "테스터",
  content: "테스트 내용입니다.....",
  createDate: "2021-06-25T06:26:34.470+00:00",
  cnt: 0
}
  
```



웹 애플리케이션 작성

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

③ 컬렉션 리턴하기

- 검색 결과가 한 건이 아니라 여러 건일 경우, `java.util.List` 같은 컬렉션에 VO 객체들을 저장하여 리턴

웹 애플리케이션 작성

2 VO(Value Object) 객체 사용하기

③ 컬렉션 리턴하기

```
@GetMapping("/getBoardList")
public List<BoardVO> getBoardList() {
    List<BoardVO> boardList = new ArrayList<BoardVO>();
    for (int i = 1; i <= 10; i++) {
        BoardVO board = new BoardVO();
        board.setSeq(i);
        board.setTitle("제목" + i);
        board.setWriter("테스터");
        board.setContent(i + "번 내용입니다.");
        board.setCreateDate(new Date());
        board.setCnt(0);
        boardList.add(board);
    }
    return boardList;
}
```



실전 스프링 부트

스프링 부트 퀵스타트 // 실전 스프링 부트

Spring Boot Basic

- src/main/java
- com.mycompany
- com.mycompany.controller
- BoardController.java
- com.mycompany.domain
- BoardVO.java
- src/main/resources
- static
- templates
- application.properties
- src/test/java
- IRE System Library (Javase-11)
- Maven Dependencies
- src
- target
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

```

1 package com.mycompany;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.WebApplicationType;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6
7 @SpringBootApplication
8 public class HelloBootApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(HelloBootApplication.class, args);
12         SpringApplication application = new SpringApplication(HelloBootApplication.class);
13         application.setWebApplicationType(WebApplicationType.NONE);
14         application.run(args);
15     }

```

1. Properties 파일을 이용하여 스프링 부트 애플리케이션 제어하기

- Zulu(OpenJDK) 16 사용
- Spring Tools 4 for Eclipse 4.0.10 사용

Markers | Properties | Snippets | Problems | Console | Progress

```

2021-11-20 10:35:39.407 INFO 228916 --- [main] com.mycompany.HelloBootApplication : Starting
2021-11-20 10:35:40.148 INFO 228916 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
2021-11-20 10:35:40.157 INFO 228916 --- [main] o.apache.catalina.core.StandardService : Starting
2021-11-20 10:35:40.157 INFO 228916 --- [main] org.apache.catalina.core.StandardEngine : Starting
2021-11-20 10:35:40.231 INFO 228916 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing
2021-11-20 10:35:40.231 INFO 228916 --- [main] w.s.c.ServletWebServerApplicationContext : Root Web
2021-11-20 10:35:40.559 INFO 228916 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
2021-11-20 10:35:40.568 INFO 228916 --- [main] com.mycompany.HelloBootApplication : Started
2021-11-20 10:36:00.925 INFO 228916 --- [nio-8888-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing
2021-11-20 10:36:00.925 INFO 228916 --- [nio-8888-exec-1] o.s.web.servlet.DispatcherServlet : Initializing
2021-11-20 10:36:00.926 INFO 228916 --- [nio-8888-exec-1] o.s.web.servlet.DispatcherServlet : Completed

```

실습단계

스프링 부트 프로젝트를 외부 Properties 파일 혹은 설정 파일을 이용하여 제어

스프링 부트 프로젝트에 웹 애플리케이션을 추가하여 브라우저에서 테스트

src/main/java에 작성

src/main/resources → application.properties을 통해 스프링 부트 애플리케이션 설정 가능

JAVA 애플리케이션에서 웹 애플리케이션 타입을 어떻게 설정했는가에 따라 웹 또는 JAVA 애플리케이션으로 동작

Properties에 설정 값이 일반 JAVA 설정보다 우선 순위가 높음

톰캣 서버의 포트번호를 8080이 아닌, 8888로 설정

자동 완성 기능 이용

저장 시 UTF-8 저장 팝업창이 뜨면 Save, 프로퍼티 설정

기존에 실행된 애플리케이션 중지

메인 애플리케이션 실행

웹 애플리케이션 타입이 servlet으로 적용되어 내부적으로 톰캣 서버가 구동됨



실전 스프링 부트

실습단계
포트번호가 8888로 변경됨
Properties 파일과 동일한 설정을 yml 파일로도 작성 가능
주의! Properties 키 들여쓰기
Properties 설정과 의미가 동일함, servlet 혹은 NONE으로 설정 가능
포트번호를 8888로 설정
애플리케이션이 계속 실행되면 포트 충돌이 발생할 수도 있기 때문!
톰캣 서버가 구동되는 웹 애플리케이션으로 동작함
내장 톰캣 구동 시 포트 번호는 8888로 변경됨



실전 스프링 부트

2. REST 기반의 웹 애플리케이션 작성

```

1 # 애플리케이션 모드 설정
2=spring:
3   main:
4     web-application-type: servlet
5
6 # 서버 포트 설정
7=server:
8   port: 8888
9

```

mpany.HelloBootApplication : Starting HelloBootApplication using Java 16.0.1 on DESKTOP-N8EIBLT with PI

mpany.HelloBootApplication : No active profile set, falling back to default profiles: default

embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8888 (http)

.catalina.core.StandardService : Starting service [Tomcat]

he.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.55]

C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

rvletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 890 ms

embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8888 (http) with context path ''

2.5)

웹 애플리케이션 작성

실습단계
웹 애플리케이션 작성
BoardVO.java 클래스 작성
멤버 변수는 private으로 설정
6개의 멤버변수(seq, title, writer, content, createDate, cnt) 선언
<Alt> + <Shift> + <S> 단축키 이용
Board controller.java 클래스 생성
컨트롤러 클래스를 @RestController 이용
컨트롤러 메소드 return 데이터를 자동으로 JSON으로 변환해 리턴
@GetMapping : Get 방식으로 요청
매개변수로 받은 name 값에 "Hello"를 붙여 리턴
getBoard() 메소드는 BoardVO 객체를 리턴
리턴된 BoardVO가 JSON 형태로 변형되어 리턴
getBoardList() 메소드 실행



실전 스프링 부트

실습단계
여러 개 BoardVO 객체가 1부터 10까지 생성, 저장, 리턴될 것
List도 JSON 형태로 변환되어 리턴될 것
브라우저에서 테스트
웹 스토어에서 'JSON Viewer' 확장 프로그램 설치
"Hello"와 입력한 파라미터 값 리턴
getBoard : 게시글 상세 정보 열람
getBoardList : 글 목록 열람