4장

Goal :

1.Hello World Controller를 만들어 보기

2.Boot Run을 통해서 서버 실행해보기

먼저 해당 클래스가 스프링에서 관리하는 컨트롤러 bean이라는 표식을 해준다.

스프링에서는 annotation 기반으로 지정한다.

@Controller

@ResponseBody

위 2개 annotation은 아래 annotation으로 치환 가능하다.

@RestController

Ctrl+b 단축키로 annotation의 선언 파일 바로가기하여 볼 수 있다.

여기서 RestController 에 @Controller, @ResponseBody 어노테이션이 선언되어 있는 것을 볼 수 있다.

스프링 부트에는 톰캣 서버를 임베디드해서 사용할 수 있게 되어 있다.

5장.

Goal

1.SpringBoot 테스트 만들기

2.MockMvc 테스트 만들기



## [ 테스트 생성 단축키 ] Controller method에서 ctrl+shift+t를 누르면 test 디렉토리에 동일한 package에 동일한 이름으로 test class를 생성해준다.

@SpringBootTest : 해당 클래스가 spring boot test임을 표시하기 위함



@Test Annotation 추가할 때 org.junit 사용하면 Junit4를 사용할 때 사용하는 것이고,

Org.junit.jupiter.api 는 Junit5를 사용할 때 사용한다.



테스트 생성 시 Junit 5가 잘 설정되었는지, 의존성이 제대로 추가 되었는지, Test가 정상적으로 동작하는지 먼저 확인 필요하다.



정상 실행 시 test 문구 출력 확인 가능하다.

@Autowired의 의미 : 스프링 컨텍스트에서 bean을 주입하겠다는 의미이다.



Springboot start test에 포함된 assert를 이용해서 검증문 작성 가능

Build gradle에 아래 junit 추가 ( 이 강의와는 다른 assert 라이브러리 사용)





Import org.junit.Assert; 임포트 후 Assert.assertEquals 메서드로 테스트 실행 가능하였음.

## 또 하나의 테스트 방식이 있다. (스프링에서 제공하는 MockMvc 방식)

MockMvc란? 실제 객체와 비슷하지만 테스트에 필요한 기능만 가지는 가짜 객체를 만들어서 애플리케이션 서버에 배포하지 않고도 스프링 MVC 동작을 재현할 수 있는 클래스를 의미한다.

모의 HttpRequest와 HttpResponse를 만들어서 테스트를 진행하게 해주는 것이다.



위 그림에서 MockMvcBuilders.standaloneSetup().build()로 MockMvc 먼저 세팅

이후 MockMvc.perform() 메서드로 동작하게 한다.