**Junit이란? (Junit.4.12)**

- Java에서 독립된 단위테스트(Unit Test)를 지원해주는 프레임워크이다.

- assert method로 테스트 케이스의 수행결과를 판별한다. (ex: assertEquals(예상값, 실제값))

- jUnit4 이후 여러 annotation지원(@Test, @Before, @After 등)

- @Test메서드가 호출할 때 마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지게 한다.

**단위테스트를 하는 이유**

1. 코드변경으로 인해 문제가 발생 시 단시간 내로 이를 파악하고 수정할 수 있도록 도와준다.
2. 기능에 대한 불확실성을 감소시킬 수 있다.
3. 개발단계 초기에 문제점을 쉽게 발견할 수 있게 도와준다.
4. 시스템에 대한 실제 문서(테스트 케이스)를 제공한다. 단위테스트 자체가 문서로 활용될 수 있다.

@Value : property의 key를 가지고 특정한 값을 호출할 수 있음.(키를 정확히 입력 or 예외처리必)

@ConfigurationProperties : root prefix를 활용하여 원하는 객체를 바인딩 가능하게 함

**제공되는 Annotation기반 기능**

@SrpingBootTest :

-통합테스트를 제공하는 기본적인 spring boot test annotation

- 실제 구동되는 어플리케이션과 똑 같은 어플리케이션 컨텍스트를 로드하여 테스트하기 때문에 하고싶은 테스트 모두 수행 가능

- @RunWith를 사용하면 내장된 runner를 사용하는 대신 정의된 러너 클래스를 사용하지만 @SpringBootTest를 사용하려면 JUnit실행에 필요한 SpringJUnit4ClassRunner 클래스를 상속받은 @RunWith(SpringRunner.class)로 사용해야한다.

- 프로파일 환경마다 다른 데이터소스를 갖는다면 어떻게 해야 할까?

A: @ActiveProfiles(“local”)과 같은 방식으로 원하는 프로파일 환경 값을 부여하면 됨

- @Transactional을 사용 시 테스트를 마치고 나서 수정된 데이터가 롤백 됨

@WebMvcTest

- 웹에서 테스트하기 힘든 컨트롤러를 테스트하는데 적합하며 웹상에서 요청(request)와 응답(response)에 대해 테스트할 수 있음,

- Controller를 본 서비스에 만들어 놓은 상태에서 test디렉토리 아래쪽에 테스트 코드용으로 XXX..Tests.java로 파일생성 후 작성

- MockMvc.perform(get(해당 주소)).andExpect().andExpect()… andDo();

- Mock이란?

Test double 이라고도 하며 실제로 사용되어야 하는 객체의 대역을 의미한다.

테스트 하려는 객체와 연관된 객체를 사용하기 어렵고 모호할 때 대신 해줄 수 있는 객체를 Test Double 이라고 함.

이러한 객체는 대상의 행위를 검증할 때 사용하기 때문에 객체가 가지고

있어야하는 기본적인 정보를 반드시 가지고 있다. 기본적으로 Mock은 일일이 클래스를 만들기보단 org.Mockito라는 라이브러리를 사용한다.

Ex) @MockBean

Private BookService bookService;

@DataJpaTest

- JPA관련 테스트 설정만을 로드 함. 데이터베이스의 설정이 정상적인가, JPA를 사용하여 데이터를 제대로 생성, 수정, 삭제하는지 등의 테스트가 가능함

- JPA테스트가 끝날 때 마다 자동으로 테스트에 사용한 데이터를 롤백한다.

- 어떤 데이터베이스를 사용할지에 대해서도 설정이 가능함

Ex) @AutoConfigureTestDatabase(connection = H2)

**추가적으로 공부가 필요한 기능의 경우 JPA에 대해서 공부할 필요가 있을 것 같습니다.**

@RestClientTest

- REST통신의 데이터형으로 사용되는 JSON 형식이 예상대로 응답을 반환하는지 등을 테스트 할 수 있음

- @RestClientTest는 테스트 대상이 되는 service.class 를 주입받는다.

- private MockRestServiceServer server;를 통하여 클라이언트와 서버 사이의 REST 테스트를 위한 객체를 준비함

- @Test의 method로는 요청에 대해 응답과 기댓값이 같은지 테스트한다.

- 지정 경로로 요청을 보내면 리소스 폴더에 만들어 놓은 ex) test.json 파일의 데이터로 응답을 주도록 설정하며 컨트롤러에서 가져온 기댓값(test.json)과 실제로 가져온 값과 비교하여 일치하는지 확인함.

@JsonTest

JSON테스트를 지원하며 @JsonTest는 JSON의 직렬화, 역직렬화를 수행하는 라이브러리인 Gson과 Jackson API의 테스트를 도와줌.

- isEqualsTo : 해당값이 일치하는지

- isNull : 해당값이 NULL인지

- isEqualToJson : 각 파일에 정의한 내용들이 일치하는지(JSON파일과 변환한 문자열)

- hasJsonPathStringValue : 해당 값이 존재하는지(Has)

- extractingJsonPathStringValue : 해당 값을 가지고 있는지(Extracting)

추후 정리 예정기능(annotation)

@Before

@After