\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Test 코드

1) build.gradle에

dependencies 에 추가

테스트 기능을 제공하는 spring-boot-start-test 추가

2) 인텔리제이의 [Gradle]을 누른다음 -- 새로고침

3) 테스트 코드는 작성한 코드가 의도대로 잘 동작하고 예상치못한

문제가 없는지 확인할 목적이다.

4) 테스트코드란?

test 디렉터리에서 실행한다.

test 코드에도 다양한 패턴이 있다.

given-when-then

given - 테스트 실행을 준비하는 단계

when - 테스트를 진행하는 단계

then - 테스트 결과를 검증하는 단계

5) 스프링부트는 어플리케이션을 테스트 하기 위한 도구와

애너테이션을 제공한다.

JUnit : 자바프로그래밍 언어용 단위 테스트 프레임워크

Spring Test & Spring Boot Test : 스프링부트 어플리케이션을 위한 통합Test 지원

AssertJ : 검증문인 어셜션을 작성하는 데 사용되는 라이브러리

6) JUnit은 자바 언어를 위한 단위 테스트 프레임워크이다.

단위테스트란 -- 작성한 코드가 의도대로 작동하는지 작은 단위로

검증하는것을 의미한다.

\*\* JunitTest.java

@Test 애너테이션으로 메서드를 호출할때마다 새 인스턴스를 생성

독립 Test가능

@DisplayName -- 테스트이름을 명시

@Test -- 테스트를 수행하는 메서드

JUnit은 테스트끼리 영향을 주지 않도록 각 테스트를 실행할 때마다

테스트를 위한 실행 객체를 만들고 테스트가 종료되면 실행객체를

삭제한다.

JUnit에서 제공하는 assertEquals()로 a+b와 sum이 같은지 확인

\*\* JunitCycleTest.java

@BeforeAll : 전체 테스트를 시작하기 전에 1회 실행하므로

메서드는 static으로 선언

@BeforeEach : 테스트 케이스를 시작하기 전에 매번 실행

@AfterAll : 전체 테스트를 마치고 종료하기 전에 한번만 실행

@AfterEach : 각 테스트 케이스를 종료하기 전 매번 실행

AssertJ 검증문 가독성 높이기

AsserJ는 Junit과 함께 사용해 검증문의 가독성을 높여주는 라이브러이다.

assertThat(a+b).isEqualtTo(sum);

a+b를 더한값이 sum과 같아야 한다.

의미로 명확하다. 다양한 메서드가 존재한다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestController.java \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@SpringBootTest -- 테스트용 어플리케이션 컨텍스트 생성

메인 어플리케이션에 추가하는 @SpringBootApplication이

있는 클래스를 찾고, 그 클래스에 포함되어 있는 빈을 찾은 다음

테스트용 어플리케이션 컨텍스트를 만든다.

@AutoConfigureMockMvc

MockMvc를 생성하고 자동으로 구성하는 애너테이션

MockMvc는 어플리케이션을 서버에 배포하지 않고도

테스트용 MVC환경을 만들어 요청 및 전송, 응답기능을

제공하는 유틸리티 클래스이다.

컨트롤러를 테스트할때 사용되는 클래스이다.

@BeforeEach

테스트를 실행하기 전에 실행하는 메서드에 적용하는

애너테이션

MockMvcSetUp()메서드를 실행해

MockMvc를 설정해준다.

@AfterEach

테스트를 실행한 이후에 실행하는 메서드에 적용하는

애너테이션

lombok -- 게터, 세터, 생성자와 같이 클래스를 만들 때

반복하여 입력하는 코드를 줄여주기 위한 라이브러이다.

\*\*\* TEST절차

Given : 멤버를 저장

When : 멤버 리스트를 조회하는 API를 호출한다.

Then : 응답코드가 200 OK이고 반환값은 값중에 0번째 요소의

id와 name이 저장된 값과 같은지 확인한다.

perform() 메서드는 요청을 전송하는 역할을 하는 메서드

결과로 ResultActions객체를 받는다.

ResultActions객체는 반환값을 검증하고

확인하는 andExpect() 메서드를 제공 해준다.

accept()메서드는 요청을 보낼 때 무슨 타입으로 응답을 받을지

결정하는 메서드이다. (JSON으로 받는다.)

andExpect()메서드는 응답을 검증한다.

jsonPath() -- JSON 응답값의 값을 가져오는 역할을 하는 메서드이다.

저장된 값과 같은지 확인한다.

\*\*\*\*\*\*\*\* 테스트 코드 패턴 연습하기

QuizController.java

/quiz 패스로 GET요청이 오면 quiz() 라는 메서드에서 요청을 처리한다.

이 메서드는 요청 파라미터의 키가 code이면 int자료형의 code변수와

매핑되며, code값에 따라 다른 응답을 보낸다.

quiz 패스로 POST 요청이 오면 quiz2()라는 메서드에서 요청을 처리한다.

이 메서즈는 요청 값을 Code라는 객체로 매핑하 후에 value값에 따라

다른 응답을 보낸다.

\*\*\* record -- 패핑할 객체로 사용하기 위해 선언한 코드이다.

데이터 전달을 목적으로 하는 객체를 더 빠르고 간편하게

만들기 위한 기능

레코드를 사용하면 필드, 생성자, 게터

equals(), hashCode(), 등등의 함수를 자동으로 생성한다.

\*\*\* MockMvc를 이용한 테스트 목적

- MockMvc를 이용하여 컨트롤러의 동작을 테스트하는 데 사용

- 컨트롤러의 엔드포인트를 호출하여 HTTP 클라이언트의 요청을 모방하고

적절한 응답을 확인하기 위해 테스트를 수행

- 이러한 테스트 과정을 통해 애플리케이션의 서비스 로직이나

API 엔드포인트가 의도한 대로 동작하는지 확인하고,

버그를 발견하고 수정하는 데 도움을 주는 것

\*\*\* MockMvc

- 스프링 프레임워크에서 제공하는 테스트용 라이브러리

이 라이브러리를 사용하면 Spring MVC 컨트롤러의 단위 테스트를 쉽게 작성

- MockMvc를 사용하면 HTTP 요청을 작성하고 컨트롤러의 응답을 검증

이를 통해 통합 테스트를 실행하지 않고도

컨트롤러의 동작을 확인

@ExtendWith(MockitoExtension.class) // Junit5 - Mockito 연동

class CodeControllerTest {

private MockMvc mockMvc; // HTTP 호출을 위한 MockMVC 사용

@BeforeEach

void setUp() {

mockMvc = MockMvcBuilders.standaloneSetup(codeController).build();

}

@Test

public void testExample() throws Exception {

mockMvc.perform(get("/example")) // "/example"로 GET 요청 수행

.andExpect(status().isOk()) // 상태 코드 200인 성공적인 응답을 기대합니다

.andExpect(content().string("expected")) // 응답 내용이 "expected"와 같을 것으로 기대합니다

.andExpect(jsonPath("$.property").value("expected")) // JSON 속성 "property"의 값이 "expected"와 같을 것으로 기대합니다

.andExpect(view().name("expectedView")) // 뷰 이름이 "expectedView"와 같을 것으로 기대합니다

.andExpect(model().attribute("attributeName", "expectedValue")) // 모델 속성 "attributeName"의 값이 "expectedValue"와 같을 것으로 기대합니다

.andExpect(redirectedUrl("expectedUrl")); // "expectedUrl"로의 리다이렉트를 기대합니다

}

}

\*\*\* ResultActions

- MockMvc를 사용하여 실행한 ‘HTTP 요청에 대한 결과

이를 통해 컨트롤러의 응답을 검증하고 원하는 동작을 수행

\*\*\* MvcResult

- MockMvc에서 수행된 MVC 요청의 결과에 대한 상세한 정보를 제공

이 클래스는 응답 상태, 헤더, 내용 등과 같은 정보를 추출하기 위한

다양한 메서드를 포함

@ExtendWith(MockitoExtension.class) Mockito를 사용하여 모킹하기 위해 테스트 클래스에 적용됩니다.

@WebMvcTest 웹 MVC 테스트를 위해 스프링 컨텍스트를 구성합니다.

@AutoConfigureJsonTesters JSON 테스트를 위해 JsonTester의 자동 구성을 활성화합니다.

@AutoConfigureMockMvc MockMvc를 자동으로 구성하는 데 사용됩니다.

@Mock 모킹 대상 객체를 생성하여 주입합니다.

@MockBean Spring 컨텍스트에서 Mock 객체를 생성하여 주입합니다.

@SpringBootTest 스프링 부트 애플리케이션의 통합 테스트를 위해 스프링 컨텍스트를 구성합니다.

@BeforEach 각각의 테스트 메서드가 실행되기 전에 실행되는 메서드를 지정합니다.

@Test 테스트 메서드를 지정합니다.