초격차 패키지 Online.

API 설계

PART1 | 애노테이션 기반 설계

@Controller, @RestController

PART2 | 함수 기반 설계

함수형 프로그래밍으로 만드는 API

PART3 | 요청, 응답의 설계

Handler Methods의 활용

PART4 | @ControllerAdvice

공통 에러 응답을 설계하는 법

PART5 | 컨트롤러 테스트

Spring Web API를 테스트하는 방법

PART6 | TDD 방식으로 복습하기

테스트 주도 개발 과정 맛보기

API 설계

6 TDD 방식으로 복습하기

Test Driven Development (TDD)

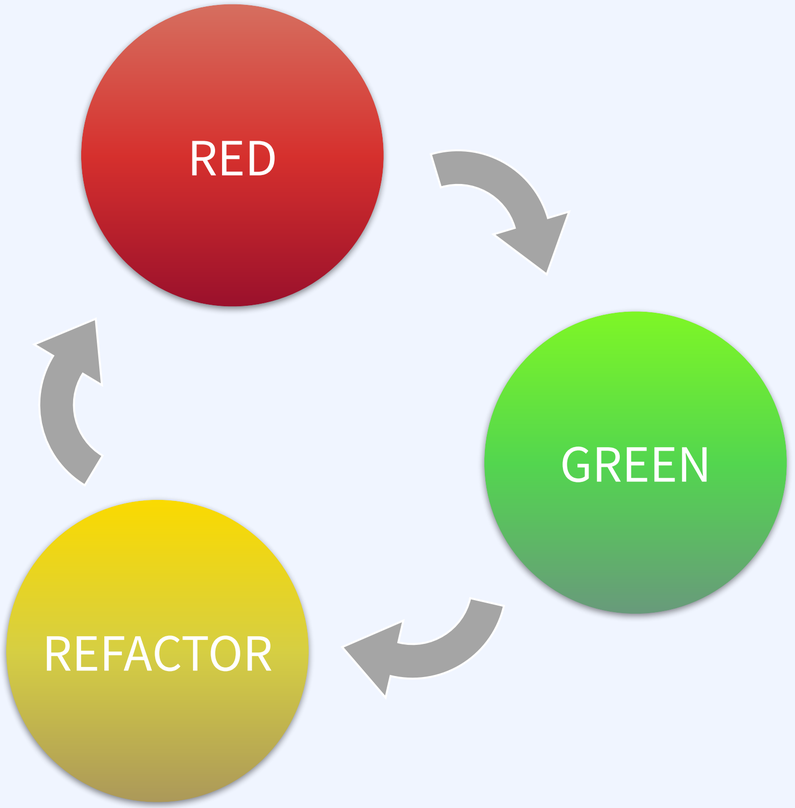
테스트 주도 개발 기법

- 프로그램의 설계와 구현, 사고의 흐름을 테스트 중심으로 생각하는 개발 방법
- 개발 순서의 변화
  - as-is: 구현한다 -> 테스트한다
  - to-be: 테스트를 만든다 -> 구현한다
- 주요 키워드: 익스트림 프로그래밍 (XP), 애자일, 폭포수 모델, Test-First Programming

RED - GREEN - REFACTOR

TDD 개발 사이클

1. RED: (실패하는) 테스트를 짠다. (요구사항의 명세)
2. GREEN: 테스트를 성공시킨다. (구현)
3. REFACTOR: 구현 코드를 고도화(리팩토링)한다.



Given - When - Then

테스트의 구조를 표현하는 방법 (a.k.a. 3A, Arrange - Act - Assert)

- Given (Arrange): 상태(state)의 정의 - 테스트를 수행할 때 전제 조건
- When (Act): 동작 - 테스트 실행
- Then (Assert): 검증 - 동작의 결과(actual) vs. 예상값(expected)

```java
@DisplayName("[API][GET] 단일 장소 조회 - 장소 있는 경우, 장소 데이터를 담은 표준 API 출력")
@Test
void givenPlaceId_whenRequestingExistentPlace_thenReturnsPlaceInStandardResponse() throws Exception {
    // Given
    int placeId = 1;

    // When
    ResultActions resultActions = mvc.perform(get( urlTemplate: "/api/places/" + placeId));

    // Then
    resultActions
            .andExpect(status().isOk())
            .andExpect(content().contentTypeCompatibleWith(MediaType.APPLICATION_JSON))
            .andExpect(jsonPath( expression: "$.data").isMap())
            .andExpect(jsonPath( expression: "$.data.placeType").value(PlaceType.COMMON.name()))
            .andExpect(jsonPath( expression: "$.data.placeName").value( expectedValue: "랄라배드민턴장"))
```

WHY TDD? WHY 테스트?

("협업(기업) 코딩" 환경에서) 왜 이걸 해야 할까?

- 내가 지금 뭘 하려는지 명확히 안다는 사실을, 스스로 지속적으로 확인한다.
  - 개발이 지연되는 이유 중 하나는, 막막해서 멍때리기 때문
- 내가 지금 뭘 하려는지 명확히 안다는 사실을, 동료와 소스 코드로 공유하고 소통(코드 리뷰)한다.
  - as-is 1: 개발 계획을 별도의 문서로 공유함
  - as-is 2: 개발 계획을 구현 코드로 공유함
  - to-be: 개발 계획을 테스트 코드로 공유함

최고의 효율로 테스트 개발을 하려면

TDD가 기존 개발 방식보다 효율적이기 위해서는

- 테스트 설계 흐름에 익숙해야 합니다.
  - 사람의 요구사항을 프로그램이 할 수 있는 기능으로 변환하기
  - 기능을 단위 기능으로 세분화하기
  - 기능의 관계와 상호작용을 설계하기
- 테스트 작성 기술에 익숙해야 합니다.

결론: 계속, 꾸준히, 많이, 동료와 함께 하세요.

테스트 내용의 발전 과정 (막연한 사람을 위한 가이드)

1. 메인 요구사항의 기본 목표 위주로만 우선 테스트를 작성

- 날짜를 yyyymmdd 포맷으로 입력하면, 정산일(D+3)을 계산해준다.

테스트 내용의 발전 과정

2. 메인 요구사항 기본 + 세부 목표를 테스트로 작성

- 날짜를 yyyymmdd 포맷으로 입력하면, 정산일(D+3)을 계산해준다.
- 입력값이 없으면, 사용자에게 적절한 경고를 표시해준다.
- 2000.01.01 이전 날짜 입력은, 최초 정산일(2000.01.01)을 출력해준다.
- 매월 말일은, 빠른 정산일(D+2)을 계산해준다.

테스트 내용의 발전 과정

3. 메인 요구사항 기본 + 세부 + 더욱 구체적인 기능적 고려 요소

- 날짜를 yyyymmdd 포맷으로 입력하면, 정산일(D+3)을 계산해준다.
  - yyyy-mm-dd 는? yymmdd는? 20210000 은? 20210231 은?
- 입력값이 없으면, ~~사용자에게 적절한 경고를 표시해준다.~~ -> 에러로 응답한다 (표현 방법은 위임)
  - null 은? "" 는? " " 는? " " 는?
- 2000.01.01 이전 날짜 입력은, 최초 정산일(2000.01.01)을 출력해준다.
  - 19991230 은? 19991231 은? 20000101 은? 20000102 은?
- 매월 말일은, 빠른 정산일(D+2)을 계산해준다.
  - 20210101? 20210102? 20210130? 20210131? 20210201? 20210228? 20201231?

Reference

- https://en.wikipedia.org/wiki/Test-driven_development
- https://ko.wikipedia.org/wiki/테스트_주도_개발
- https://martinfowler.com/bliki/TestDrivenDevelopment.html
- https://martinfowler.com/bliki/GivenWhenThen.html