와플스튜디오 세미나

스프링 부트 세미나 4

과제

- 다음 과제를 위해 api endpoint 통일
- 3,4 과제는 TEST / 배포 과제
- 테스트용 클라이언트도 준비 중

- 공통 피드백
  - *entityManager를 끌어쓰는 방식 지양*

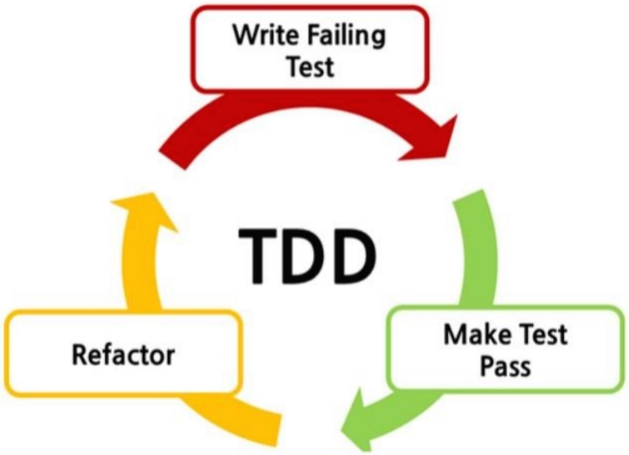
목차

- TEST
- 배포+
  - *CI/CD*
  - *웹서버*
- Api 문서
- Caching
- Functional controller

- 심화 내용을 더욱 하고 싶다.

TDD (Test Driven Development)

- 테스트 주도 개발
  - *큰 규모의 소프트웨어 개발에서 지향하는 방식*
  - *테스트 코드 작성 후 개발*
- 설계 단계에서 프로그래밍 목적을 반드시 미리 정의해야만 하고, 또 무엇을 테스트해야 할지 미리 정의해야 한다.
  - *반복적인 단계가 진행되며 코드의 버그가 줄고 재설계 시간이 감소*



Test

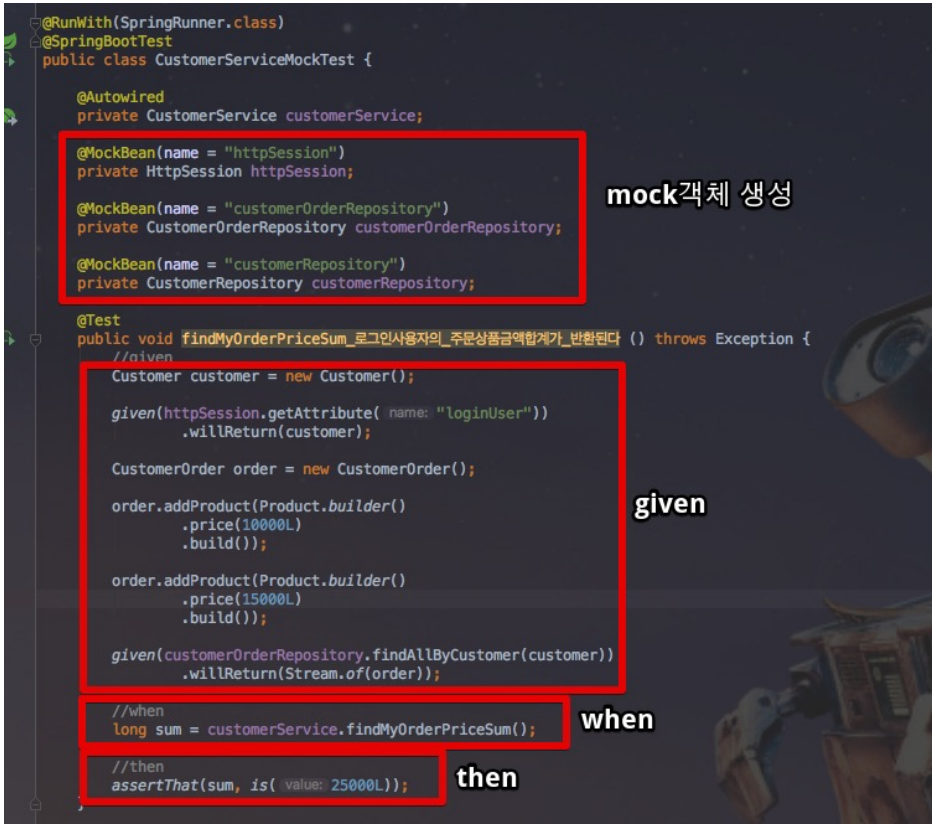
- Integration test
  - *통합된 서버 자체의 동작을 테스트*
- Unit test
  - *각각의 component 단위를 테스트*
  - *하나의 단위만을 테스트 하는 것이므로 의존성이 없어야 한다.*

Integration Test

- 실제 의존관계를 모두 포함시키고 api 리퀘스트를 통한 검증
- Spring MVC에서는 MockMvc (테스트 클라이언트) 를 통해 request를 날리고 response를 검증하는 단계로 진행
- 주요 어노테이션
  - @SpringBootTest: @SpringBootAppIlication이 붙은 클래스를 찾아 어플리케이션 컨텍스트 구동, 모든 빈을 스캔함, 비용이 아주 큼
  - *@AutoConfigureMockMvc*

Unit test

- Class 하나에 대해 의존관계를 배제시킨 테스트
- 의존관계들에 @MockBean을 사용해 mocking
- 주요 어노테이션
  - *@ExtendWith: 스프링에 특화된 테스트 기능사용 -> X -> JUnit5 전용 스프링테스트컨텍스트(SpringExtension)를 생성해줌(핵심은 비용이 작음)*
  - *@MockBean: 가짜 클래스 생성 후 어플리케이션 컨텍스트에 스프링빈으로 등록 (@SpyBean)*
  - *@BeforeEach, @BeforeAll*



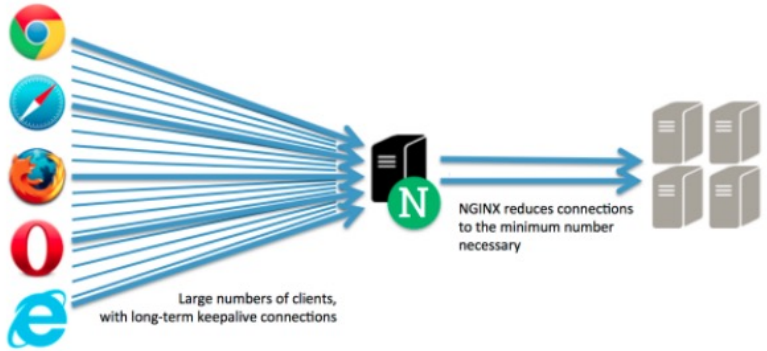
CI/CD

- CI (continuous integration)
  - *한국말로 지속적 통합*
  - *변경사항이 생길 때마다 새로운 코드 변화에 대한 테스트 실행*
  - *이를 통한 추가 변경사항 검증 및 오류 탐색*
- CD (continuous deployment)
  - *지속적 배포*
  - *배포 자동화를 통해 업데이트가 필요할 때마다 배포 가능*
- 주요 툴
  - *Github action*
  - *Jenkins*
  - *Travis ci*

Web Server

- Nginx
  - *web server의 일종*
  - *클라이언트로부터 HTTP 요청을 받아 정적인 페이지 및 파일로 응답.*
  - *웹 어플리케이션 서버에 앞서 요청을 관리하고 배분하여 효율적인 동작을 가능하게 함*
  - *하나의 instance 내에서도 가상 호스트(서버) 개념을 사용해, 단일 IP 내에서 여러 웹 배포 가능*

https://www.youtube.com/watch?v=Zimhvf2B7Es



Api 문서

- Client와 소통하기 위해 api spec을 문서로 정리해야 함
  - *기본적으로 endpoint, parameter, request body, header, response body, status code 등의 정보가 포함되어야 한다.*
  - *Stoplight, confluence, notion 등으로 작성*
- 문서 자동화
  - *REST docs*
  - *swagger*

POST search_query/

강의를 검색합니다. `tags/:year/:semester/`로부터 교과 구분, 학점, 학년, 강사, 학과, 교양영역 정보를 받아올 수 있습니다. input의 각 member는 AND 연산이고 time_mask를 제외한 member 안의 원소들은 OR 연산입니다.

member	description	example
year	연도	`2016`
semester	학기 ('1', 'S', '2', 'W'가 아닌 1, 2, 3, 4)	`1`
title	Optional. 강의명. 정규표현식을 통해 유사한 강의명을 모두 검색합니다.	`"수학 및 연습 1"` or `"수및연 1"`
classification	Optional. 교과 구분 배열	`["전필", "전선"]`
credit	Optional. 학점 배열	`[2, 3]`
course_number	Optional. 교과목 번호 배열	`["034.334", "32.513"]`
academic_year	Optional. 학년 배열	`["3학년", "4학년"]`
instructor	Optional. 강사 배열	`["하순희", "엄현상"]`
department	Optional. 학과 배열	`["컴퓨터공학부", "국어교육과"]`
category	Optional. 교양영역 배열	`["문화와 예술", "체육"]`
etc	Optional. 영어 강의 혹은 군휴학 원격수업을 검색할 수 있습니다. `"E"`: 영어진행 강의, `"MO"`: 군휴학 원격수업	`["E"]`
time_mask	Optional. 시간 검색. number 7개의 배열을 받습니다. 각 entry는 30bit 길이의 non-negative numeric bitmask입니다. 첫번째 원소가 월요일이고 마지막 원소는 일요일입니다. MSB가 0교시이며 2bit가 1시간입니다. **Exact match가 아닌 해당 mask 안에 있는 강의를 검색합니다.**	화요일과 목요일, 각 13교시부터 1시간 동안 진행할 때 `[0, 12, 0, 12, 0, 0, 0]`
offset	Pagination offset. Default = 0	`0`
limit	Pagination limit. Default = 20	`20`

Sample Input

```
{
    "year":2016,
    "semester":1,
    "title":"공연실1"
}
```

Sample Output

```
[
  {
    "year": 2016,
    "semester": 1,
```

https://haddoddo.tistory.com/entry/Spring-Boot-STS-%EC%8A%A4%ED%94%84%EB%A7%81%EB%B6%80%ED%8A%B8%EC%97%90-swagger-%EC%97%B0%EB%8F%99-%EB%B0%8F-%EC%82%AC%EC%9A%A9%EB%B2%95

Caching

- 어떨 때 쓰면 좋을까?
  - *무거운 query가 자주 발생할 때*
  - *잦은 요청이 발생할 때*
  - *여러 사용자가 공통으로 공유하는 데이터일 때*
  - *client에게 주어야 할 데이터가 꼭 최신이거나 정확할 필요가 없을 때*
- 주로 Redis를 사용
- 만료 시간을 지정 가능

Functional controller

- Annotation 기반의 controller 보다 더욱 custom 하기 쉬움 (spring의 제약에서 벗어남)

```
@Bean
fun notification():RouterFunction<ServerResponse> {
    return route()
        .path("/v2/notifications", builder -> builder
            .GET("", serviceHandler::getNotifications)
            .POST("", serviceHandler::createNotification)
            .GET("/{notificationId}", serviceHandler::getNotification)
            .DELETE("/{notificationId}", serviceHandler::removeNotification)
            .PUT("/{notificationId}", serviceHandler::updateNotification)
        )
        .build();
}
```

https://www.baeldung.com/spring-mvc-functional-controllers

다음 주제

- Functional programming
- Reactive programming
- Webflux

함수형 프로그래밍
반응형 프로그래밍: https://www.youtube.com/watch?v=KDiE5qQ3bZI
쓰레드와 멀티프로세싱 개념