Chapter 6: Dependency Injection & Mocks

김성훈

Dependency Injection과 Mocks

- 테스트를 하는 과정에서 시스템이나 확장 서비스에 의존하지 않도록 해야 한다.
- 이 서비스들은 테스트에 대해서는 유용하거나 신뢰할 수 없다.
- 이번 챕터에서 배울 기법들은 에러 상태를 테스트하고, 실패 상태를 저장하고, 로직을 SDK로 부터 분리할 수 있게 한다.

What's up with fakes, mocks, and stubs?
test doubles

- 다른 코드가 각각의 SUT에 영향을 주기 때문에 대상에 대해 정확히 테스트하기 어렵다. 그래서 test double들을 이용해서 실제 코드와 분리할 수 있다.

- Stub: 가공된 응답값을 반환한다. 주로 비거나 없는 값을 반환하도록 구현한다.
- Fake: 로직을 가지고 있으며, 실제 데이터 대신에 테스트 데이터를 제공한다. 예를 들어 Fake 네트워킹 메니저는 네트워킹에 접근하는 대신에 로컬의 JSON을 읽어 제공한다.
- Mock: 다양한 동작을 증명할 때 사용한다. 실제 동작을 수행하며 기대되는 상태값을 제공한다.
- Partial mock: 일반적인 Mock은 실제 구현 객체를 완벽하게 대응하지만, Partial mock은 구현의 일부분만 재정의하여 사용한다. Partial mock은 구현 객체를 상속 받거나 프록시를 제공하여 구현한다.
- https://youtu.be/meTnd09Pf_M?t=2911

Understanding CMPedometer

- Core Motion의 CMPedometer를 통하여 사용자의 활동데이터를 수집할 수 있다.

CMPedometer 사용법

1. 만보계 이용 가능여부와 사용자의 허가를 확인한다.
2. 업데이트를 감시한다.
3. 사용자가 정지하거나, 목표에 도달하거나, Nessie에게 질 때까지 걸음수와 거리를 수집한다.

만보계 객체는 CMPedometerData를 받을 수 있는 CMPedometerHandler를 제공한다. 이 데이터 객체는 여행한 걸음수와 거리 정보를 가지고 있다.

설사 실제 기기에서 실행한다고 하더라도 TDD를 하며 CMPedometer를 사용하긴 힘들다. CMPedometer는 기기 상태에 의존하기 때문에 일관된 유닛테스트에 너무 많은 상태를 가진다.

시연

Mocking the pedometer

- Mock 객체를 생성하고 CMPedometer를 Mock 객체와 교체한다. 이를 위해 만보기의 구현과 인터페이스를 분리해야 한다.
- 이를 위해 Facade와 Bridge 패턴을 이용한다.

시연

Handling error conditions

- Mock 객체는 error 상태를 테스트하기 쉽게 한다.

- 지금까지 테스트 하면 다음과 같은 에러들을 만난다.
  - 걸음수 측정 불가능
  - 사용자가 기기에서 모션기록을 거부했다.

시연

Mocking a callback

- 또 다른 오류 상황인 콜백에서 오류가 넘어오는 경우도 확인해야 한다.
- 이 경우는 첫번째로 앱을 실행하여 허가 팝업이 나올 때 사용자가 거절하면 오류가 콜백으로 전달된다.

시연

Getting actual data

- 실제 데이터 업데이트를 테스트한다.
- 적절한 Mock을 만드는데 중요한 요소이다.
- CMPedometer는 CMPedometerData 객체로 실제 데이터를 전달한다.

시연

Making a functional fake

- 유닛 테스트는 로직을 입증하긴 좋지만 사용자 경험을 입증하긴 힘들다.
- 이를 위한 한가지 방법은 빌드해서 실제 기기에서 실행하는 방법이 있다. 그러나 이 방법은 걸어서 목표에 도달하도록 해야 한다.
- 시간이 너무 많이 드니 다른 방법을 하자.

시연

Wiring up the chase view

- chase view에서 Nessie와 사용자의 환경을 보여준다.
- 테스트 환경에서 사용자 상태를 표시하기 위해 partial mock을 사용할 수 있다.
- partial mock을 사용하면 메인 로직을 건드리지 않고 기능을 추가할 수 있다.

시연

Time dependencies

- 마지막으로 Nessie에 대한 로직이 필요하다. 시간을 기준으로 측정해야 하기 때문에 타이머를 이용한다.
- 타이머는 테스트 하기 힘들기로 악명이 높다.
- 타이머 테스트는 기대에 따라 잠재적으로 많은 기다림이 필요하다.

- 일반적으로 3가지 해결책이 있다.
    1. 테스트하는 동안 짧은 타이머를 사용한다.
    2. 타이머를 Mock으로 교환하여 콜백을 즉시 실행한다.
    3. 콜백을 직접 실행하여 앱 또는 사용자 수락 시간을 아낀다.

- 이번 테스트에서는 3번 해결책을 사용할 것이다.

시연

Challenge 시연

Key points

- Test double들은 다른 시스템과 분리하여 테스트 할 수 있도록 돕고, 시스템 SDK과 네트워킹환경, 타이머들로 부터 탈출 시켜준다
- Mock들은 테스트 수행환경에서 바꿔치기 한다. 그리고 Partial mock들은 객체의 일부분을 대신해 준다.
- Fake들은 테스트를 위한 데이터나 시뮬레이터 환경을 지원한다.