테스트 코드 작성으로 iOS 개발 효율성 높히기

✅ 테스트 가능하도록 코드 작성하기

Apple Developer Acedemy @ POSTECH 3rd

Ace

개요

1. 테스트 코드란 무엇인가?
2. 테스트 코드를 작성하는게 과연 효율적일까?
3. 테스트 가능한 코드 작성하기
4. 더 나아가기

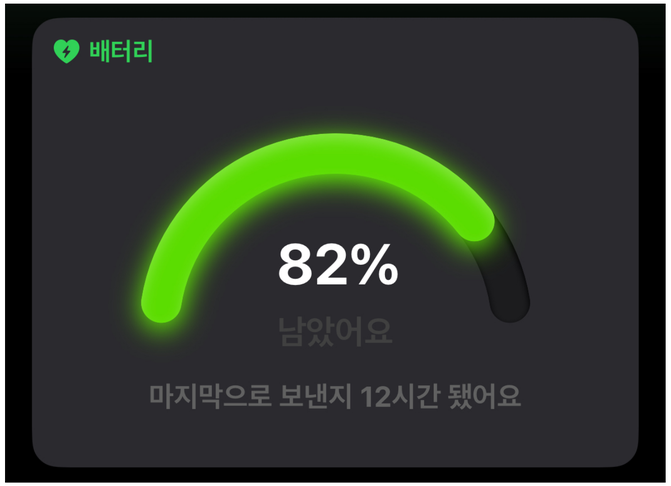
1. 테스트 코드란 무엇인가?

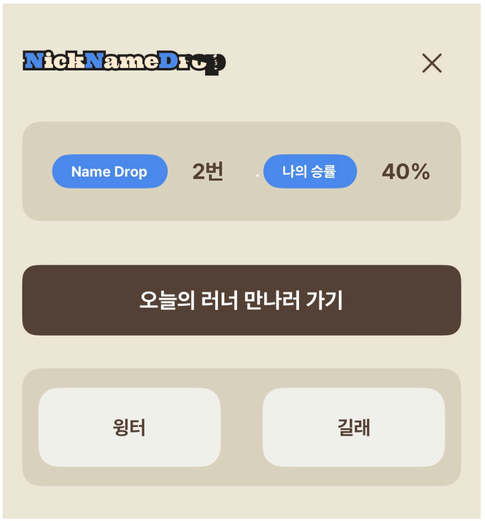
소프트웨어의 동작을 자동으로 테스트 하는 코드

ex) 사용자가 입력한 이메일 형식의 유효성을 검증하는 로직이 잘 작동하는지 테스트

ex) 사진들이 서버에서 잘 받아와지는지 테스트

ex) 로그인된 유저일 경우 메인 화면이 보이고, 비로그인 유저는 회원가입 화면이 보이는지 테스트

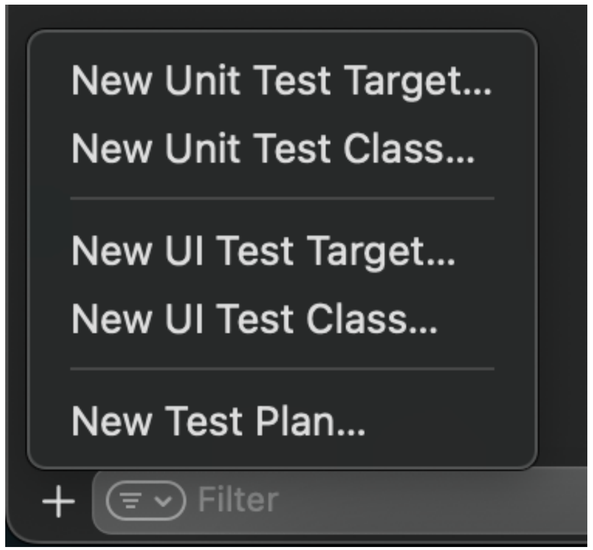




1. 테스트 코드란 무엇인가?

Swift 에서의 테스트는 어떻게 생겼나

- XCTest, Testing 모듈

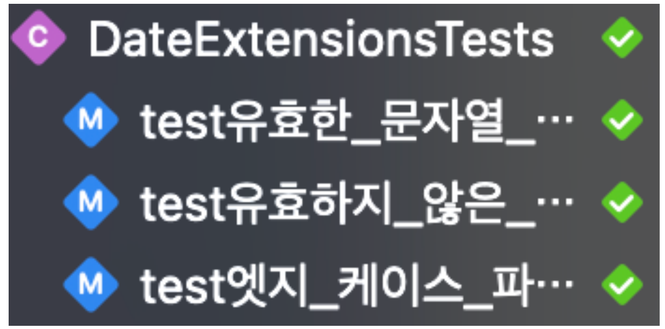



```swift
class DateExtensionsTests: XCTestCase {

    func test유효한_문자열_날짜로_파싱() {
        let validDateString = "2024-05-30"

        let validDate = validDateString.parseDate()

        let formatter = DateFormatter()
        formatter.dateFormat = Constants.DateFormat.YEAR_MONTH_DAY_DATEFORMAT
        let expectedDate = formatter.date(from: validDateString)

        XCTAssertEqual(validDate, expectedDate, "유효한 문자열은 파싱 성공")
    }

    func test유효하지_않은_문자열_날짜로_파싱() {
        let invalidDateString = "INVALID DATE!"

        let invalidDate = invalidDateString.parseDate()

        XCTAssertEqual(invalidDate, Date(timeIntervalSince1970: 0), "유효하지 않은 문자열은 파싱 실패.")
    }
```

1. 테스트 코드란 무엇인가?

테스트 코드 작성시 이점

- 버그가 발생하기 이전에 미리 발견 가능
- 코드 변경 및 리팩토링 할 때 안정성 확보 (+ 개발자의 심신 안정)
- 의도와 동일하게 작동하는지 검증 가능
- QA 역할 어느정도 대체 가능



마음이 편안해지는 초록색 체크표시들

1. 테스트 코드란 무엇인가?

iOS 개발에는 대표적으로 다음과 같은 테스트가 있다.

단위 테스트

함수 같은 로직을 작은 단위로 테스트 하는 것
(최소 단위의 로직을 테스트 하는 것은 명확하고 안전하며 어렵지 않다)

UI 테스트

로그인 회원가입 과정 등 기능의 흐름을 테스트 하는 것
(변수가 많아서 복잡하고 구현하기 어렵다.)

2. 테스트 코드 작성하는게 과연 효율적일까

- 같은 로직을 왜 두번 작성하냐
- 구현 코드 작성할 시간도 촉박하다
- 어차피 사람이 버그를 다 잡지 못한다.
  테스트 코드에도 빈틈이 생긴다.
- 막상 잘 안쓰는 회사 많다.

2. 테스트 코드 작성하는게 과연 효율적일까

그럼에도 불구하고 시도해야 하는 이유

테스트 가능한 코드는

품질이 좋은 코드의 중요한 특징 중 하나이다.

2. 테스트 코드 작성하는게 과연 효율적일까

그럼에도 불구하고 시도해야 하는 이유

Why?

테스트 가능한 코드의 특징

- 모듈화가 잘 되어있다.
- 리팩토링 안정성이 높다.
- 상태와 로직이 분리되어있다.
- 확장성이 좋다.

-> 테스트 "가능" 하게만 만들어도 위에 조건들이 충족된다.

3. 테스트 가능한 코드 작성하기

뭘 테스트 해야하지?

1. 중요 로직 테스트
2. 경계값 테스트
3. 예외 테스트
4. 최대한 많은 코드 테스트

https://shoulditestprivatemethods.com/

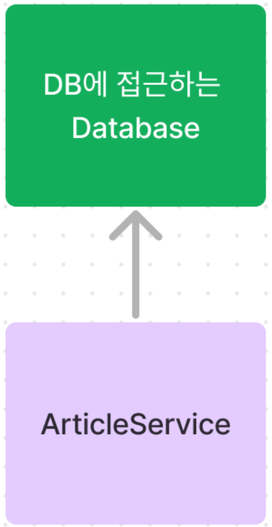
3. 테스트 가능한 코드 작성하기

테스트 불가능한 구조 예시

```swift
struct Article {
    let id: Int
    let content: String
    var isFavorite: Bool = false
}
```

```swift
class Database {
    func loadArticles() -> [Article] {
        return // DB에서 불러온 실제 데이터 반환
    }
}
```

```swift
class ArticleService {
    private var articles: [Article]
    private var articleScore: Int
    private var database: Database // 데이터베이스 의존성

    init(database: Database) {
        self.articles = repository.loadArticles()
        self.articleScore = 0
        self.database = database
    }

    // 아티클의 상태에 따라 점수 계산
    func calcArticleScoreStatus() {
        var score = 0
        for article in articles {
            score += article.isFavorite ? 20 : 10
        }

        // 내부 상태 변경
        articleScore = score
    }
}
```



DB 서버에 의존, 메소드가 값 반환이 아닌 클래스 상태 변경 -> 테스트 어렵다.

3. 테스트 가능한 코드 작성하기

테스트 가능한 구조 예시

```swift
struct Article {
    let id: Int
    let content: String
    var isFavorite: Bool = false
}


protocol ArticleRepository {
    func loadArticles() -> [Article]
}


class Database: ArticleRepository {
    func loadArticles() -> [Article] {
        return // DB에서 불러온 실제 데이터 반환
    }
}
```

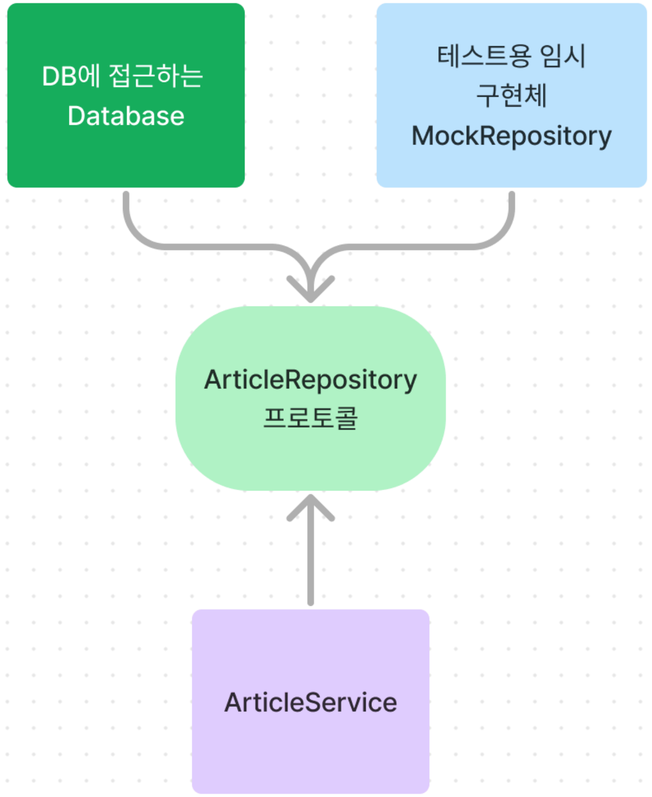
```swift
class ArticleService {
    private var articles: [Article]
    private let repository: ArticleRepository // 의존성 주입

    init(repository: ArticleRepository) {
        self.articles = repository.loadArticles()
        self.repository = repository
    }

    // 테스트 용이하도록 값 반환하게 변경
    func calculateScore() -> Int {
        var score = 0
        for article in articles {
            score += article.isFavorite ? 20 : 10
        }

        return score
    }
}
```

메소드가 상태를 변경하는게 아닌 값 반환, 의존성 주입 받아서 결합도 낮춤 -> 테스트 가능

3. 테스트 가능한 코드 작성하기

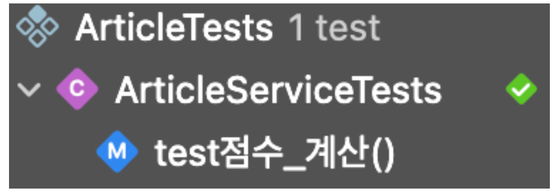
테스트 코드 예시

```
import XCTest
@testable import TechBlogNotifications

class MockRepository: ArticleRepository {
    func loadArticles() -> [Article] {
        return [
            Article(id: 1, content: "테스트1", isFavorite: true),
            Article(id: 2, content: "테스트2")
        ]
    }
}

final class ArticleServiceTests: XCTestCase {
    func test점수_계산() {
        let mockRepository = MockRepository()
        let articleService = ArticleService(repository: mockRepository)

        let score = articleService.calculateScore()

        XCTAssertEqual(score, 30, "점수 계산 테스트 통과 실패")
    }
}
```



프로토콜을 준수하는 임시 구현체 클래스 MockRepository 만들어서

데이터베이스 없어도 ArticleService 클래스의 메소드 테스트 성공!



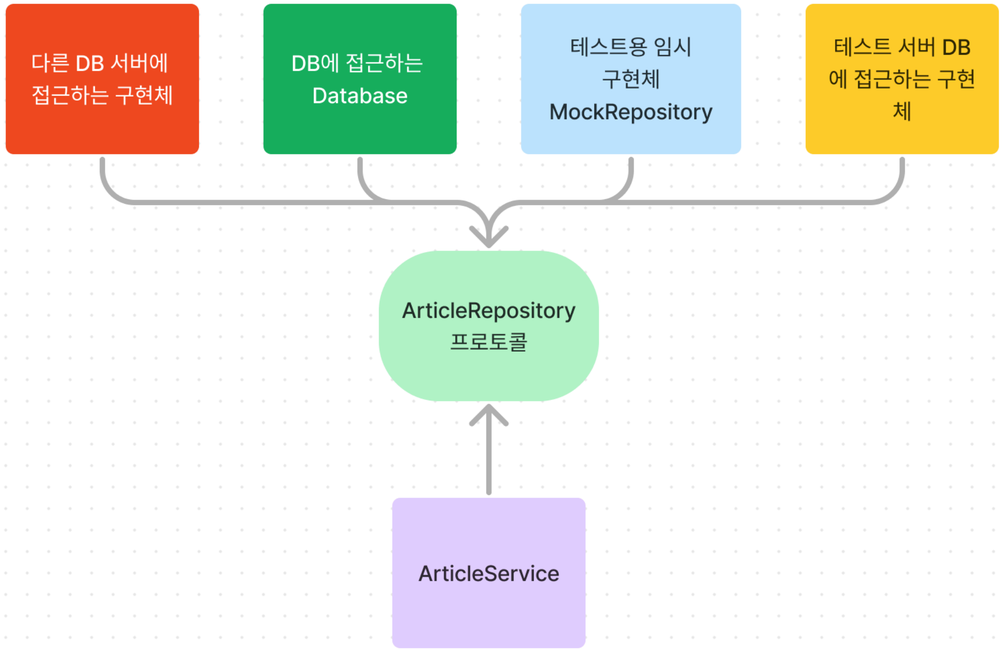
3. 테스트 가능한 코드 작성하기

테스트 가능한 구조 예시

테스트 가능한 코드를 만들다 보니

1. 의존성 주입
2. 코드 단순화
3. 확장성 증가

가 되면서 코드 품질이 좋아졌다.



4. 더 나아가기

TDD?

- 테스트 코드가 잘 작성된 코드는, 코드의 품질이 좋다는 뜻일 확률이 높다.
- 테스트 코드 작성이 부담된다면, GPT-4 를 활용해봐도 좋다.

- 좋은 테스트 코드가 무엇인지 알고싶으면 단위 테스트의 FIRST 원칙
- 테스트 주도 개발 방법론 (TDD)도 학습해서 적용해보는 경험을 쌓아보자.

TDD 개념 정리 블로그 링크

감사합니다.

✅ 질문 있으면 해주세요!

Apple Developer Acedemy @ POSTECH 3rd

Ace