앱 안정성 및 확장성 강화를 위한

안드로이드 아키텍처

-의존성 주입-

[Part 5. 의존성 주입 2](#_Toc124149195)

[1. 의존성 주입(DI, Dependency Injection) 개념 2](#_Toc124149196)

[2. Dagger 사용법 4](#_Toc124149197)

[3. Dagger Hilt 사용법 5](#_Toc124149198)

[4. Hilt 테스트 6](#_Toc124149199)

[5. Dagger 고급편 7](#_Toc124149200)

[6. DI 적용을 위한 조언과 Best Practices 8](#_Toc124149201)

[참고 링크 9](#_Toc124149202)

# Part 5. 의존성 주입

## 1. 의존성 주입(DI, Dependency Injection) 개념

1.1 의존성 주입이란 무엇인가?

-의존성(Dependency) : 사용되는 객체 인스턴스

ㄴ코드에서 두 모듈 간의 연결, 두 클래스 간의 관계

ㄴ일반적으로 두 모듈(클래스) 중 하나가 다른 하나를 어떤 용도를 위해 사용한다.

-의존성 주입(DI) : 클라이언트가 아닌 잘 조직화 된 곳에서 일관성 있는 형태로 객체를 생성하고, 필요한 곳에 전달되도록 코드를 조직화하는 기법

cf. 서비스 : 전달된 객체(의존성)

cf. 클라이언트 : 객체(의존성)를 사용하는 곳. 서비스는 클라이언트 상태의 일부가 된다.

cf. DI라이브러리 : Dagger, Hilt, Koin

|  |
| --- |
| //Example1: Not DI  class NewsRemoteDataSource {  private val newsApi: NewsApi = ProdNewApi() //유닛테스트 작성이 어려움  }  //Example2: DI  class NewsRemoteDataSource(private val newsApi: NewsApi) {  //…  } |

**1.2 의존성이 위험한 이유**

-하나의 모듈이 바뀌면 의존한 다른 모듈까지 변경이 이루어진다.

-유닛 테스트 목적 자체가 다른 모듈로부터 독립적으로 테스트하는 것을 요구하기 때문에 테스트가 어려워진다.

**1.3 DI가 필요한 이유**

-코드를 재사용 가능하게 한다.

ㄴ클라이언트 코드의 변경 없이 손쉽게 다른 형태의 DataSource인스턴스를 만들 수 있다.

ㄴ객체의 생성 과정이 바뀌더라도 클라이언트 코드에 영향을 주지 않는다.

-클라이언트는 인터페이스로만 객체를 알고있으면 되므로, 보다 나은 설계를 가능하게 하고 유지보수가 편리해진다.

ㄴ추상화가 쉬워진다.

ㄴ멀티모듈에서 모듈별 의존성을 떼어내는 데에 매우 유용하다.

-테스트 코드 작성이 쉬워진다.

|  |
| --- |
| // Example1: Not DI  class NewsRemoteDataSource() {  private val newsApi: NewApi = when(BuildConfig.type()) {  PROD -> ProdNewsApi()  TEST -> FakeNewsAPi()  }  }  //Example2: DI  val fakeNewsRemoteDataSource = NewsRemoteDataSource(FakeNewsApi()) |

**1.4 Injector 클래스**

-DI를 위한 인스턴스화가 일어나는 클래스

-클라이언트가 사용할 서비스를 지정하는 대신 Injector 가 클라이언트에게 사용할 서비스를 알려준다.

-서비스를 호출하는 클라이언트는 해당 서비스를 구성하는 방법을 알 필요가 없다. 따라서 클라이언트는 외부 클래스인 Injector에게 서비스 구성을 위임한다.

**1.5 DI 프레임워크를 사용하는 이유**

-DI과정에서 많은 양의 보일러 플레이트 코드가 필요하다.

-Injector 클래스를 유지보수 하는 것은 프로젝트가 커지면 힘들어진다.

-앱 빌드의 설정에 따라 다른 종류의 Injector를 구현하는 것은 상당히 많은 양의 중복 코드를 만든다.

cf. JSR-330 표준 프레임 워크 : Dagger(+Hilt, +Anvil)

## 2. Dagger 사용법

2.1 Dagger 란?

## 3. Dagger Hilt 사용법

## 4. Hilt 테스트

## 5. Dagger 고급편

## 6. DI 적용을 위한 조언과 Best Practices

# 참고 링크

[Android] 의존성 주입(Dependency Injection)이란?

https://onlyfor-me-blog.tistory.com/373

Test Coverage

의존성 주입 라이브러리 Hilt

<https://developer.android.com/training/dependency-injection/hilt-testing>