

# FastRxJava

이승민

뱅크샐러드 Android Developer / GDE Android Korea

[rfrost77@gmail.com](mailto:rfrost77@gmail.com)

# 테스트란?



2019

# 테스트란?

어떤 동작을 호출했을 때,

1. 값이 원하는 형태가 되는지 검증
2. 원하는 동작을 연계해서 수행하는가 검증



2019

```
@Test
fun loginTest() {
    loginHelper.login()
    // isLogin 값이 True가 되었는가?
    assertTrue(loginHelper.isLogin)
}
```



2019

```
@Test
fun loginTest() {
    loginHelper.login()
    // isLogin 값이 True가 되었는가?
    assertTrue(loginHelper.isLogin)
}
```



2019

```
@Test
fun loginTest() {
    loginHelper.login()
    // isLogin 값이 True가 되었는가?
    assertTrue(loginHelper.isLogin)
}
```



2019

```
@Test
fun searchTest() {
    presenter.searchGithubRepos(searchText)
    // view.showRepos(repo)가 동작하는가?
    Mockito.verify(view).showRepos(repos)
}
```



2019

```
@Test
fun searchTest() {
    presenter.searchGithubRepos(searchText)
    // view.showRepos(repo)가 동작하는가?
    Mockito.verify(view).showRepos(repos)
}
```



2019



```
@Test
fun searchTest() {
    presenter.searchGithubRepos(searchText)
    // view.showRepos(repo)가 동작하는가?
    Mockito.verify(view).showRepos(repos)
}
```



2019

# 테스트 원리

검증하고자 하는 메소드를 수행시켰을 때,

1. 원하는 값의 `assert()`가 true
2. 원하는 동작의 `verify()`가 true



2019

# 테스트 원리

검증하고자 하는 메소드를 수행시켰을 때,

1. 원하는 값의 `assert()`가 true
2. 원하는 동작의 `verify()`가 true





테스트

# RxJava Test

`test().assert()`

# RxJava Test - Schedulers

**io, computation 비동기 안됨**

**trampolin, TestScheduler**

- 1. 주입받기**
- 2. RxJavaPlugins**

# RxBinding



# RxBinding

**RxJava API for 안드로이드 UI**



# RxBinding

**안드로이드 UI Method를 Rx로 Wrapping**

# RxBinding 장점

안드로이드 UI Method를 Rx로 Wrapping

확장성

# 확장성

UI 동작을 Operator로 가공 가능

throttle

merge

combineLatest

# 원리

**View를 들고있는 Observable**

# Memory

**View lifecycle에서 다루면 auto dispose()**

**Activity -> Observable -> View -> Observer**

**Activity가 죽으면 함께 사라짐**



# Room

DB

데이터 저장소

# DB 종류

## Relational DB

## nosql



# Relational DB

## column으로 명세

id	name	age	location
1	이승민	20	서울
2	홍길동	23	부산

← Column

row / data

# Relational DB 조회/편집

## 쿼리

**select from where**

**insert**

**update**

**delete**

# Relational DB 장점

**명세가 있어 연산이 빠르다**

# nosql

## no 명세

```
[
  {
    id: 1,
    name: 이승민,
    age: 20
  },
  {
    id: 1,
    name: 홍길동,
    location: seoul
  },
]
```

**nosql 장점**

**명세 자유도가 높다**

**안드로이드 DB**

**SQLite**

**Relational DB**

# SQLite

```
object FeedReaderContract {  
    // Table contents are grouped together in an anonymous object.  
    object FeedEntry : BaseColumns {  
        const val TABLE_NAME = "entry"  
        const val COLUMN_NAME_TITLE = "title"  
        const val COLUMN_NAME_SUBTITLE = "subtitle"  
    }  
}
```

명세 정의

```
private const val SQL_CREATE_ENTRIES =  
    "CREATE TABLE ${FeedEntry.TABLE_NAME} (" +  
        "${BaseColumns._ID} INTEGER PRIMARY KEY," +  
        "${FeedEntry.COLUMN_NAME_TITLE} TEXT," +  
        "${FeedEntry.COLUMN_NAME_SUBTITLE} TEXT)"  
  
private const val SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS ${FeedEntry.TABLE_NAME}"
```

테이블 생성

# ORM

```
@Parcelize
data class User(
    @SerializedName(value: "id")
    val id: Long,
    @SerializedName(value: "login")
    val userName: String,
    @SerializedName(value: "avatar_url")
    val avatarUrl: String,
    @SerializedName(value: "followers_url")
    val followersUrl: String,
    @SerializedName(value: "repos_url")
    val reposUrl: String
) : Parcelable
```

## Object Relation Mapping



**ORM**

**Room by Google**

# Realm

## 새로운 DB

### SQLite vs Realm

### **NOT** Room vs Realm

# Room

## Annotation 기반

## Rx 지원



# 부록

# 제가 실무레벨로 활용하는 안드로이드 기술스택

**언어:** Kotlin, Java

**Elephant:** Retrofit, Glide, Rxjava

**DB:** SQL, Room

**Architecture:** MVP / CleanArchitecture / Multimodule

**DI:** Koin, Dagger2

# 제가 잘 모르는 안드로이드 기술스택

**Architecture:** MVVM

**Library:** DataBinding, AAC ViewModel, LiveData

**Thread:** Coroutines

**하나를 해도 제대로 알면  
다른 기술은 쉽게 이해할 수 있습니다  
근본은 똑같습니다**

**응용으로 가치를 만들면서도  
원리를 탐구하세요**

**안드로이드 개발자 귀하입니다**  
**함께 힘냅시다**



끝

이승민

뱅크샐러드 Android Developer / GDE Android Korea

[rfrost77@gmail.com](mailto:rfrost77@gmail.com)