이대명

GDG 인천

언젠간 쓰게될지도 모르는것들 살펴보기

KOTLIN

앞으로의 내용이 모두 코틀린 ..

REALM

장점

- 빠르다 (write - 3.5x / read 2.5x)
- 문서의 한글화가 잘되어 있다
- 사용성이 좋다
- 지속적인 업데이트 / 피드백

세팅

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.0-beta4'
        classpath "io.realm:realm-gradle-plugin:2.0.2"
    }
}
```

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'realm-android'
apply plugin: 'kotlin-kapt'

android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
    defaultConfig {
        applicationId "com.moka.mokatoyapp"
        minSdkVersion rootProject.ext.minSdkVersion
```

BEST PRACTICE ??

사용하면서 고민이 생겼던 것들

- realm 인스턴스를 필요할때 가지고 와서 사용할것인가? fragment 의 멤버로 가지고 있으며 사용할것인가?
- 스레드간 이동이 자유롭지 않다
  - copyFromRealm / copy
- autoRefresh 를 사용할것인가 ?

INSERT

```
override fun insert(insert: (
    var copyTask: Task? = nul
    RealmHelper.onTransaction

        val number = realm.wh
        val taskId: Long
        if (null == number)
            taskId = 1
        else
            taskId = number.toLong() + 1

        val realmTask = realm.createObject(Task::class.java, taskId)
        insert(realmTask)
        realmTask.createdAt = DateUtil.timestampInSecond
        realmTask.updatedAt = DateUtil.timestampInSecond

        copyTask = realmTask.copy()
    }
    ob.onInsert(copyTask!!)
}
```

```
fun onTransaction(work: (realm: Realm) -> Unit) {
    val realm = instance
    realm.beginTransaction()
    work(realm)
    realm.commitTransaction()
    realm.close()
}
```

INSERT

```
override fun insert(insert: (Task) -> Unit) {
    var copyTask: Task? = null
    RealmHelper.onTransaction { realm ->

        val number = realm.where(Task::class.java).max("id")
        val taskId: Long
        if (null == number)
            taskId = 1
        else
            taskId = number.toLong() + 1

        val realmTask = realm.createObject(Task::class.java, taskId)
        insert(realmTask)
        realmTask.createdAt = DateUtil.timestampInSecond
        realmTask.updatedAt = DateUtil.timestampInSecond

        copyTask = realmTask.copy()
    }
    ob.onInsert(copyTask!!)
}
```

INSERT

```
override fun insert(insert: (Task) -> Unit) {
    var copyTask: Task? = null
    RealmHelper.onTransaction { realm ->

        val number = realm.where(Task::class.java).max("id")
        val taskId: Long
        if (null == number)
            taskId = 1
        else
            taskId = number.toLong() + 1

        val realmTask = realm.createObject(Task::class.java, taskId)
        insert(realmTask)
        realmTask.createdAt = DateUtil.timestampInSecond
        realmTask.updatedAt = DateUtil.timestampInSecond

        copyTask = realmTask.copy()
    }
    ob.onInsert(copyTask!!)
}
```

INSERT

```
override fun insert(insert: (Task) -> Unit) {
    var copyTask: Task? = null
    RealmHelper.onTransaction { realm ->

        val number = realm.where(Task::class.java).max("id")
        val taskId: Long
        if (null == number)
            taskId = 1
        else
            taskId = number.toLong() + 1

        val realmTask = realm.createObject(Task::class.java, taskId)
        insert(realmTask)
        realmTask.createdAt = DateUtil.timestampInSecond
        realmTask.updatedAt = DateUtil.timestampInSecond

        copyTask = realmTask.copy()
    }
    ob.onInsert(copyTask!!)
}
```

스레드간 이동이 자유롭지 않다

```kotlin
open class Task : RealmObject(), BaseDomain {

    @PrimaryKey
    var id: Long = 0
    var title: String = ""
    var content: String = ""
    var completed: Boolean = false
    var createdAt: Long = 0
    var updatedAt: Long = 0

}
```

```kotlin
fun Task.copy(): Task {
    val task = Task()
    task.id = this.id
    task.title = this.title
    task.content = this.content
    task.completed = this.completed
    task.createdAt = this.createdAt
    task.updatedAt = this.updatedAt
    return task
}

fun RealmResults<Task>.copy(): List<Task> {
    return this.map { it.copy() }
}
```

GET

```
fun onInstance(work: (realm: Realm) -> Unit) {
    val realm = instance
    work(realm)
}
```

```
override fun getTasks(filterStatus: Int): Observable<Task> {
    var observable: Observable<Task>? = null

    RealmHelper.onInstance { realm ->
        var query = realm.where(Task::class.java)

        query = when (filterStatus) {
            TaskListFragment.ACTIVE_TASKS -> query.equalTo("completed", false)
            TaskListFragment.COMPLETED_TASKS -> query.equalTo("completed", true)
            else -> query
        }

        observable = query.findAll()
                .asObservable()
                .filter { it.isLoaded }
                .first()
                .flatMap {
                    Observable.from(it.copy())
                }
                .doOnCompleted { realm.close() }
    }
    return observable!!
}
```

RX ANDROID

REACTIVEX IS A LIBRARY
FOR COMPOSING
ASYNCHRONOUS AND
EVENT-BASED PROGRAMS
BY USING OBSERVABLE SEQUENCES.

ReactiveX
An API for asynchronous programming
with observable streams

쉽게 써볼수 있는곳

- DB 에서 데이터를 들고와 adapter 에 set
- 네트워크 요청
- RxBinding
- EventBus (subject)

DB

```
override fun getTasks(filterStatus: Int): Observable<Task> {
    var observable: Observable<Task>? = null

    RealmHelper.onInstance { realm ->
        var query = realm.where(Task::class.java)

        query = when (filterStatus) {
            TaskListFragment.ACTIVE_TASKS -> query.equalTo("completed", false)
            TaskListFragment.COMPLETED_TASKS -> query.equalTo("completed", true)
            else -> query
        }

        observable = query.findAll()
                .asObservable()
                .filter { it.isLoaded }
                .first()
                .flatMap {
                    Observable.from(it.copy())
                }
                .doOnCompleted { realm.close() }
    }
    return observable!!
}
```

DB

```
fun loadTask(filterStatus: Int) {
    adapterModel.clear()
    taskRepository
            .getTasks(filterStatus)
            .observeOn(Schedulers.io())
            .map { task -> adapterModel.getItemList().add(TaskItemData(task)) }
            .observeOn(AndroidSchedulers.mainThread())
            .filter { isAttached }
            .subscribe({}, Throwable::printStackTrace, {
                if (adapterModel.getItemList().size > 0)
                    view!!.refreshAdapterList()
                else
                    view!!.showNoTasks()
            })
}
```

스케쥴링

DB

```
fun loadTask(filterStatus: Int) {
    adapterModel.clear()
    taskRepository
            .getTasks(filterStatus)
            .observeOn(Schedulers.io())
            .map { task -> adapterModel.getItemList().add(TaskItemData(task)) }
            .observeOn(AndroidSchedulers.mainThread())
            .filter { isAttached }
            .subscribe({}, Throwable::printStackTrace, {
                if (adapterModel.getItemList().size > 0)
                    view!!.refreshAdapterList()
                else
                    view!!.showNoTasks()
            })
}
```

오퍼레이터 연산

DB

```
fun loadTask(filterStatus: Int) {
    adapterModel.clear()
    taskRepository
            .getTasks(filterStatus)
            .observeOn(Schedulers.io())
            .map { task -> adapterModel.getItemList().add(TaskItemData(task)) }
            .observeOn(AndroidSchedulers.mainThread())
            .filter { isAttached }
            .subscribe({}, Throwable::printStackTrace, {
                if (adapterModel.getItemList().size > 0)
                    view!!.refreshAdapterList()
                else
                    view!!.showNoTasks()
            })
}
```

스케쥴링

NETWORK

```
@POST("/v1/api/signup")
fun signUp(@Body signUpRequestBody: SignUpRequestBody): Observable<Response<SignUpRes>>
```

```
fun reqSignUp() {
    val signUpRequestBody = SignUpRequestBody()
    signUpRequestBody.email = view!!.getEmail()
    signUpRequestBody.password = view!!.getPassword()

    view!!.showLoadingDialog()
    mocaApi.signUp(signUpRequestBody)
            .subscribeOn(Schedulers.io())
            .distinctUntilChanged()
            .cache()
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(getSignUpSubscriber())
}

private fun getSignUpSubscriber(): Subscriber<Response<SignUpRes>> =
        object : Subscriber<Response<SignUpRes>>() {

            override fun onError(error: Throwable) {
                if (null != view && view!!.isAdded()) {
                    view!!.dismissLoadingDialog()
                    view!!.showToast("인터넷 연결을 확인해주세요")
                }
            }

            override fun onCompleted() {
                if (null != view && view!!.isAdded())
                    view!!.dismissLoadingDialog()
            }
```

EVENT BUS

```
private PublishSubject<Boolean> dateChange;

public void sendDateChange(Boolean o) {
    if ( null == dateChange )
        dateChange = PublishSubject.create();

    dateChange.onNext(o);
}

public PublishSubject<Boolean> onDateChange() {
    if ( null == dateChange )
        dateChange = PublishSubject.create();

    return dateChange;
}
```

RX BINDING

```
fun setOnClickStopAlarm() {
    RxView.clicks(stopAlarmView)
            .map {
                currentCount++
                setCurrentCountText(currentCount)
            }
            .debounce(700, TimeUnit.MILLISECONDS)
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe({
                if (currentCount == releaseCount) {
                    activity.stopService(Intent(activity, WakeUpService::class.java))
                    StartActivityUtil.restartMainActivity(activity)
                }
                else {
                    currentCount = 0
                    setCurrentCountText(0)
                    showToast("${releaseCount}번 맞게 클릭하지 않으셨어요 :(\n다시 눌러주세요 !!")
                }
            }, { e -> }, {})
            .put(compositeSubscription)
}
```

700m 이하에서 연속해서 item 이 에밋되면

마지막 item 이 700m 뒤에 에밋됨

더 많은 사용예들 ..

https://github.com/gdgand/android-rxjava

ANKO

뷰를 XML 대신 KOTLIN 으로 그릴수 한 DSL

세팅

```
/* Kotlin & Anko */
compile rootProject.ext.kotlin
compile rootProject.ext.anko
compile rootProject.ext.anko_appcompat
compile rootProject.ext.anko_design
compile rootProject.ext.anko_recyclerview
compile rootProject.ext.anko_cardview
```

장점

- 빠르다 (xml 에 비해 300%)
- findViewById 안해도 된다 ..
  - 뷰 들을 바로 인스턴스로 받을수 있다

단점

- 프리뷰가 안된다

ANKO

```
class AppLockActivityUI : AnkoComponent<AppLockActivity> {
    lateinit var lockNum1: TextView
    lateinit var lockNum2: TextView
    lateinit var lockNum3: TextView
    lateinit var lockNum4: TextView
    lateinit var description: TextView

    override fun createView(ui: AnkoContext<AppLockActivity>) = with(ui) {
        linearLayout {
            orientation = LinearLayout.VERTICAL
            linearLayout {
                orientation = LinearLayout.VERTICAL
                textView("하 루 하 루") {
                    lparams(width = matchParent, height = wrapContent) {
                        topMargin = dip(24)
                        bottomMargin = dip(44)
                    }
                    id = R.id.textView_applock_app_name
                    gravity = Gravity.CENTER
                    textColor = R.color.base_text_edit_text
                    typeface = Typeface.DEFAULT_BOLD
                    textSize = 16f
                }
                textView("암호 입력") {
                    lparams(width = matchParent, height = wrapContent) {
                    }
                    id = R.id.textView_applock_label_01
                    gravity = Gravity.CENTER
                    typeface = Typeface.DEFAULT_BOLD
                    textSize = 16f
                }
                description = textView("암호를 입력해주세요") {
                    lparams(width = matchParent, height = wrapContent) {
                        topMargin = dip(7)
                        bottomMargin = dip(10)
                    }
                    id = R.id.textView_applock_label_02
                    gravity = Gravity.CENTER
                    textColor = R.color.app_gray_color
                    textSize = 13f
                }
                linearLayout {
                    lparams(width = matchParent, height = dip(14)) {
                        topMargin = dip(12)
                    }
                    id = R.id.linearLayout_lockNumbers
                    orientation = LinearLayout.HORIZONTAL
```

감사합니다