Introduction to RxJava for Android

Kunal Malhotra
kunal.malhotra1@ibm.com
@kunal_forever

IBM Code

Agenda

- What is RxJava ?
- The Problem.
- Why Rx ?
- The Basic.
- Let’s CODE.

WHAT IS RxJAVA ?

What is RxJava?

RxJava is a Java VM implementation of ReactiveX (Reactive Extensions): a library for composing asynchronous and event-based programs by using observable sequences

The Problem
How to deal with execution context ?

Threads

```
Handler handler = new Handler(Looper.getMainLooper());
    new Thread(){
    @Override
    public void run() {
        final String result = doHeavyTask();
        handler.post(new Runnable() {
            // or runUIThread on Activity
            @Override public void run() {
                showResult(result);
                }
            });
        }
     }.start();
```

Threads

Pros:

- Simple

Cons:

- Hard way to deliver results in UI thread
- Pyramid of Doom

AsyncTask

```
new AsyncTask(){
	@Override
	protected String doInBackground(Void... params)
	{
		return doHeavyTask();
	}
	@Override
	protected void onPostExecute(String s)
	{
		showResult(s);
	} }.execute();
```

AsyncTask

Pros:

- Deal with main thread

Cons:

- Hard way to handling error
- Not bound to activity/fragment lifecycle
- Not composable
- Nested AsyncTask
- “Antek Async”

Why Rx?

- Because multithreading is hard
- Execution context
- Powerful operators
- Composable
- No Callback Hell
- Etc ...

Basic

The basic building blocks of reactive code are Observables and Subscribers. An Observable emits items; a Subscriber consumes those items.

The smallest building block is actually an Observer, but in practice you are most often using Subscriber because that's how you hook up to Observables.

The Basic

Observables often don't start emitting items until someone explicitly subscribes to them

Enough talk, let’s see some code ...