RxJava

Douglas Drumond
douglas.drumond@movile.com

whoami

movile

we think mobile

Por que RxJava?

Eventos de UI

Eventos do domínio

Concorrência

RxJava

Observer pattern

Iterator pattern

Functional programming

Lambda

Retrolambda

```
export JAVA_HOME=/path/java8
export JAVA7_HOME=/path/java7
```

Retrolambda

```
plugins {
    id "me.tatarka.retrolambda" version "3.2.2"
}

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8   } }
```

Retrolambda

```
view.setOnClickListener(new View.
OnClickListener() {
        @Override
        public void onClick(final View v) {
            ...
        }
    });
```

Retrolambda

```
view.setOnClickListener(v -> {


});
```

Básico

Observables

Subscribers

Operadores

movile

Observables

Emite itens

Chama onNext, onComplete ou onError

Hot or Cold

Subscriber

Implementa

- onNext
- onComplete
- onError

Operadores

flatMap

filter

debounce

create

defer

etc

movile

Criando Observable e Subscriber

DEMO

Operators

Subscribers reagem, não mudam

Operadores transformam dados

Operators

DEMO

Exemplo com UI

DEMO

Mais operadores

filter

Ex:

```
.filter(query -> !TextUtils.isEmpty(query))
```

movile

Threads

subscribeOn

observeOn

NewThreadScheduler

~~HandlerThreadScheduler~~

Rede

DEMO

Tratamento de erros

```
Observable.create(...)
        .map(s -> exception(s))
        .map(s -> exception2(s))
        .subscribe(new Subscriber<String>() {
                @Override public void onNext(String s) { Log.d(TAG, s); }
                @Override public void onCompleted() { Log.d(TAG, "uhu!"); }
                @Override public void onError(Throwable e) { Log.e(TAG, "ops!"); }
        });
```

Ciclo de vida da Activity

.cache

.unsubscribe ao rodar

.subscribe ao restaurar

Use RxLifecycle

Google Play Services

https://github.com
/mcharmas/Android-ReactiveLocation

https://github.com
/zmarkan/Reactive-PlayServices

Google Play Services

```
ReactiveLocationProvider locationProvider = new
ReactiveLocationProvider(context);
locationProvider.getLastKnownLocation()
    .subscribe(new Action1<Location>() {
        @Override
        public void call(Location location) {
            doSthImportantWithObtainedLocation(location);
        }
    });
```

Testes

```
@Test
public void shouldLoadTwoUsers() throw Exception {
    TestSubscriber<User> testSubscriber
                        = new TestSubscriber<>();
    databaseHelper.loadUser().subscribe(testSubscriber);
    testSubscriber.assertNoErrors();
    testSubscriber.assertReceivedOnNext(
                        Arrays.asList(user1, user2))
}
```

Testes

https://github.com/ribot/assertj-rx + BlockingObservable

Testes (exemplo)

```
assertThat(observable.toBlocking()).
completes();


assertThat(observable.toBlocking())
    .completes()
    .listOfValuesEmitted()
    .containsExactly("a", "b", "c");
```

Referências

http://reactivex.io/
https://github.com/ReactiveX/RxJava/wiki
/Alphabetical-List-of-Observable-Operators
https://speakerdeck.com/benjchristensen
https://github.com/mutexkid/rxjava-koans

Referências

https://github.com/trello/RxLifecycle

https://github.com/JakeWharton/RxBinding

https://github.com/ReactiveX/RxAndroid/

Código

https://github.com/
douglasdrumond/KnightsOfLambdaCalculus

Obrigado

movile

Contato

douglas.drumond@movile.com

@douglasdrumond +DouglasDrumond
www.cafelinear.com

talentos@movile.com