Reactive Java (RxJava)

kabisa

ARTISANS OF INFORMATION TECHNOLOGY

Introduction

- Why RxJava?
- Observables / Examples
- Reactive Functional Programming / Examples
- Async / Sync
- Error Handling
- Workshop and….. 🍕

Why RxJava?

“It frees you from tangled webs of callbacks, and thereby makes your code more readable and less prone to bugs.” - reactivex.io

“With ReactiveX you can later change your mind, and radically change the underlying nature of your Observable implementation, without breaking the consumers of your Observable.” - reactivex.io

Bonus:
Reactive support in multiple languages

Push vs Pull

	Single Items	Multiple Items
Synchronous	T getData()	Iterable<T> getData()
Asynchronous	Future<T> getData()	***Observable<T> getData()***

Creating An Observable

```
// Emits elements from 1 - 10 to the Observer
Observable<Integer> numbers = Observable.range(1,10);

// Emits elements "Hello" and "RxJava!" to the Observer
List<String > words = Arrays.asList("Hello", "RxJava!");
Observable<String> observable = Observable.fromIterable(words);

// We need to subscribe to the Observable before anything happens
observable.subscribe(System.out::println);
```

Reactive Functional Programming

- Transforming functions
  - Map
- Filtering functions
  - Filter
- Combining functions
  - Zip

Examples

```
Observable<Integer> numbers = Observable.range(1, 10);

// [2, 4, 6, 8, 10]
numbers.filter(x -> x % 2 == 0);

// ["#1", "#2", "#3", "#4", "#5", "#6", "#7", "#8", "#9", "#10"]
numbers.map(x -> "#" + x);

// 55
numbers.reduce((x,y) -> x + y);
```

Examples

```
List<String> alphabet = Arrays.asList("A", "B", "C", "D", “E”);

Observable<Integer> numbers = Observable.range(1,5);
Observable<String> letters = Observable.fromIterable(alphabet);

// [ 1A, 2B, 3C, 4D, 5E ]
Observable.zip(numbers, letters, (num, let) -> num + let);

// [ 1, 2, 3, 4, 5, A, B, C, D, E ]
Observable.concat(numbers, letters);
```

Async / Sync

- SubscribeOn
- Schedulers
    - newThread
    - computation
    - IO

Examples

```
// Emits elements “Hello” and “RxJava!” to the Observer
List<String > words = Arrays.asList("Hello", "RxJava!");
Observable<String> observable = Observable.fromIterable(words);

// We want to iterate on a different thread (async)
observable.subscribeOn(Schedulers.newThread())

// We need to subscribe to the Observable before anything happens
observable.subscribe(System.out::println);
```

Error Handling

"The ability for the producer to signal to the consumer that an error has occurred (an Iterable throws an exception if an error takes place during iteration; ***an Observable calls its observer's onError method****)"*

- reactivex.io

Examples

```
 // Emits elements “Hello” and “RxJava!” to the Observer
 List<String > words = Arrays.asList("Hello", "RxJava!");
 Observable<String> observable = Observable.fromIterable(words);

words.subscribe(
  System.out::println,
  error -> System.out.println("Handle the error”)
);

words.onErrorReturnItem("Return default for error");
```

Workshop.just(🍕);

Let's Get Coding!

https://github.com/Vreijsen/rx-java-workshop

1. Try to change the existing Java code to RxJava code.

2. Add an extra service to connect to (Optional).

Pro Tip: Use .blockingGet()

References

- http://rxmarbles.com/
- http://reactivex.io/
- https://github.com/ReactiveX/RxJava/wiki
- https://kabisa.nl