Signaler med ReactiveCocoa

-------o-------o-------x

`RACSignal`

Objekt som skickar värden

RACSignal

```
RACSignal *getSignal = [RACSignal getRequest:@"movie/12123"];
```

RACSignal

```
RACSignal *getSignal = [RACSignal getRequest:@"movie/12123"];

[getSignal subscribeNext:^(NSDictionary *json) {
    NSLog(@"did fetch movie: %@",json);
}];
```

Error

-----x

Error

```
RACSignal *getSignal = [RACSignal getRequest:@"movie/12123"];

[getSignal subscribeNext:^(NSDictionary *json) {
    NSLog(@"did fetch movie: %@",json);
} error:^(NSError *error) {
    NSLog(@"request failed with error: %@",error);
}];
```

Complete

----------|

Complete

```
RACSignal *getSignal = [RACSignal getRequest:@"movie/12123"];

[getSignal subscribeCompleted:^{
    NSLog(@"completed!");
}];
```

Operatorer

---------A--------|

---------B--------|

En operator är en metod som från en signal returnerar en ny signal.

Operatorer

- map:
- flattenMap:
- filter:
- delay:
- repeat
- take:
- merge:
- collect:

Map

Omvandlar en signals värden till nya värden

Map

```
[RACSignal getRequest:@"movie/12123"]
```

Map

```
[[RACSignal getRequest:@"movie/12123"]
// Map json to Video
map:^id(NSDictionary *json) {
    return [[Movie alloc] initWithDictionary:json];
}]
```

Map

```
[[[RACSignal getRequest:@"movie/12123"]
// Map json to Video
map:^id(NSDictionary *json) {
    return [[Movie alloc] initWithDictionary:json];
}] subscribeNext:^(Movie *movie) {
    NSLog(@"did fetch movie with title: %@",movie.title);
}];
```

Flatten map

Skapar en helt ny signal från ett skickat värde

Flatten map

```
// Get the movie
[RACSignal getMovieWithId:@1231]
```

.

Flatten map

```
// Get the movie
[[RACSignal getMovieWithId:@1231]

// Get the director
flattenMap:^RACStream *(Movie *movie) {
    return [RACSignal getDirectorWithId:movie.directorId];
}]
```

.

Flatten map

```
// Get the movie
[[[RACSignal getMovieWithId:@1231]

// Get the director
flattenMap:^RACStream *(Movie *movie) {
    return [RACSignal getDirectorWithId:movie.directorId];
}]

// Get the directed movies
flattenMap:^RACStream *(Director *director) {
    return [RACSignal getMoviesWithDirector:director.directorId];
}]
```

.

Flatten map

```
// Get the movie
[[[[RACSignal getMovieWithId:@1231]

// Get the director
flattenMap:^RACStream *(Movie *movie) {
    return [RACSignal getDirectorWithId:movie.directorId];
}]

// Get the directed movies
flattenMap:^RACStream *(Director *director) {
    return [RACSignal getMoviesWithDirector:director.directorId];
}]

// Get imdb ratings for all movies
flattenMap:^RACStream *(NSArray *movies) {

    NSArray *signals = [movies arrayByMappingWithBlock:^id(Movie *movie) {
        return [RACSignal getImdbRatingForMovieWithId:movie.movieId];
    }];
    return [[RACSignal merge:signals] collect];

}]
```

.

Flatten map

```
// Get the movie
[[[[[RACSignal getMovieWithId:@1231]

// Get the director
flattenMap:^RACStream *(Movie *movie) {
    return [RACSignal getDirectorWithId:movie.directorId];
}]

// Get the directed movies
flattenMap:^RACStream *(Director *director) {
    return [RACSignal getMoviesWithDirector:director.directorId];
}]

// Get imdb ratings for all movies
flattenMap:^RACStream *(NSArray *movies) {

    NSArray *signals = [movies arrayByMappingWithBlock:^id(Movie *movie) {
        return [RACSignal getImdbRatingForMovieWithId:movie.movieId];
    }];
    return [[RACSignal merge:signals] collect];

}]

// Calculate the avarage rating
map:^id(NSArray *ratings) {

    __block CGFloat sum = 0;
    [ratings enumerateObjectsUsingBlock:^(NSNumber *rating, NSUInteger idx, BOOL *stop) {
        sum += rating.floatValue;
    }];
    return @(sum / ratings.count);
}]
```

.

Flatten map

```
// Get the movie
[[[[[[RACSignal getMovieWithId:@1231]

// Get the director
flattenMap:^RACStream *(Movie *movie) {
    return [RACSignal getDirectorWithId:movie.directorId];
}]

// Get the directed movies
flattenMap:^RACStream *(Director *director) {
    return [RACSignal getMoviesWithDirector:director.directorId];
}]

// Get imdb ratings for all movies
flattenMap:^RACStream *(NSArray *movies) {

    NSArray *signals = [movies arrayByMappingWithBlock:^id(Movie *movie) {
        return [RACSignal getImdbRatingForMovieWithId:movie.movieId];
    }];
    return [[RACSignal merge:signals] collect];

}]

// Calculate the avarage rating
map:^id(NSArray *ratings) {

    __block CGFloat sum = 0;
    [ratings enumerateObjectsUsingBlock:^(NSNumber *rating, NSUInteger idx, BOOL *stop) {
        sum += rating.floatValue;
    }];
    return @(sum / ratings.count);
}]

subscribeNext:^(NSNumber *directorRating) {
    NSLog(@"The director's average rating on imdb is: %@",directorRating);
}];
```

Exempel

starta sparande

1. Terms and condition visas.

2. Användaren svarar på “Know Your Customer” frågor.

3. Användaren matar in sitt personnummer.

4. Om användaren saknar bank-id appen informeras han om att detta krävs.

5. Applikationen anropar MobileBankIdIdentification/ med användarens personnummer som ett headervärde.

6. Applikationen anropar MobileBankIdIdentification/ <referense_id>.

7. Bankid-applikationen öppnas så att användaren kan signera.

8. Applikationen anropar MobileBankIdIdentification/<referense_id> varannan sekund tills den returnerar progress code `COMPLETE`

9. Applikationen anropar ConfigurationValues/

10. Applikationen anropar SavingApplications/

11. Applikationen anropar POST SignOrder/ med application id

12. Applikationen anropar POST SignOrder/ med sparmålet

13. Bankid-applikationen öppnas så att användaren kan signera.

14. Applikationen anropar POST SignOrder/ varannan sekund tills det returnerade SignOrder objektet har state `COMPLETE`.

KVO

Länkar

- http://github.com/andersfrank/movies
- http://nshipster.com/reactivecocoa/
- http://github.com/hsjunnesson/RACFlickrSearch
- http://github.com/hsjunnesson/CrayolaColors