RxCocoa

Updated 28 Mar 2024

Info

Lifecycle

Observables are entities capable of notifying subscribers that some data has arrived or changed, pushing values to be processed.

For this reason, the correct place to subscribe to an observable while working in view controllers is inside viewDidLoad. This is because you need to subscribe as early as possible, but only after the view has been loaded.

Subscribing in a different lifecycle event might lead to missed events, duplicate subscriptions, or parts of the UI that might be visible before you bind data to them.

Therefore, you have to create all subscriptions before the application creates or requests data that needs to be processed and displayed to the user.

Observables are entities capable of notifying subscribers that some data has arrived or changed, pushing values to be processed. Subscription returns a Disposable which lets you cancel the subscription when necessary.

Binding observables:

think of the relationship as a connection between two entities:

- A producer, which produces the value
- A consumer, which processes the values from the producer
- A consumer cannot return a value. This is a general rule when using bindings in RxSwift

`bind(to:)` is used to bind an observable to another entity. It's required that the consumer conforms to ObserverType, a write-only entity that can only accept new events but cannot be subscribed to.

With RxSwift the only ObserverType is Subject. We can also use `bind(to:)` on Relays. Those `bind(to:)` methods are separate overloads since Relays don't conform to ObserverType.

`bind(to:)` is an alias for `subscribe()`. Calling `bind(to: observer)` will internally call `subscribe(observer)`.

Traits

Traits are a group of Observable type-conforming objects, which are specialized for creating straightforward, easy-to-write code with UI.

Traits provide compile-time guarantee and predictable rules when dealing with UI.

The RxCocoa Traits are:

- ControlProperty and ControlEvent
  - can't error out
  - represents properties of object that can both be readable & writable
  - is used to listen to UI events
- Driver
  - can't error out
  - share resources b/c they are derived from an entity called SharedSequence
    - Driver automatically gets `share(replay: 1)`
  - suitable for modeling state (due to its replay strategy)
- Signal
  - can't error out
  - share resources b/c they are derived from an entity called SharedSequence
    - Signal gets `share()`
  - useful for modeling events (due to its replay strategy)

Name	Events			Shares Effect	Replay	Scheduler
	next	error	complete			
Observable	✔	✔	✔	✘	✘	Any
PublishSubject	✔	✔	✔	✘	✘	Any
BehaviorSubject	✔	✔	✔	✘	✔	Any
Driver	✔	✘	✔	✔	✔	**Main**
Signal	✔	✘	✔	✔	✘	**Main**
Completable	✘	✔	✔	✔	N/A	Any
Single	✔ **(1)**	✔	✘	✘	✘	Any
Maybe	✔ **(1)**	✔	✔	✘	✘	Any
BehaviorRelay	✔	✘	✘	✘	✔	Any
PublishRelay	✔	✘	✘	✘	✘	Any

RxSwift - RxCocoa Entities and Traits