Thứ 3 18/01/2022

RxCocoa

I. Label, Button, TextField

1. Sử dụng với RxCocoa:

- Trong rxcocoa ta được cung cấp wrapper để thực hiện các công việc reactive với các thành phần UI mà không cần gọi thành phần delegate làm trung gian nữa.
- Để gọi các wrapper này ta gọi .rx để gọi extension cho các wrapper này.
- Ví dụ:

```
textField.rx.text
    .subscribe(onNext: { [weak self] text in
        guard let self = self else { return }
        self.label.text = self.textField.text
        })
    .disposed(by: disposeBag)
```

- Trong ví dụ trên ta theo gọi rx.text để truy cập wrapper kiểu ControlProperty. ControlProperty là các observable chạy trên main thread để thực hiện các công việc cập nhật UI.
- Trong ví dụ trên ta lắng nghe cho các element mà được emit bởi observable text của textField, kế tiếp ta thực hiện update label sử dụng

element được emit.

2. Sử dụng bind(to:)
    - Ta có thể gọi ngắn các bước subscribe sử dụng convienent method bind(to:). Ví dụ

```
textField.rx.text
    .map{ $0?.count ?? 0 > 5 }
    .bind(to: button.rx.isEnabled)
    .disposed(by: disposeBag)
```

    - Chú ý: ở 2 ví dụ trên ta chỉ thực hiện cập nhật UI cho label và button ở trong các sự kiện onNext() của textField.rx.text.
    - Nếu ta mà thay đổi property text ở bên ngoài thì sẽ không thực hiện các công việc cập nhật này.
3. Sử dụng với Button:
    - Với button ta có thể không cần đặt @IbAction hay đặt function cho khi nào mà button được nhấn. Để bắt khi nào mà button được nhấn TouchUpInside ta sử dụng wrapper tap của extension rx. Ví dụ:

```
button.rx.tap
    .throttle(.seconds(3), scheduler: MainScheduler.instance)
    .subscribe(onNext: { [weak self] in
        guard let self = self else { return }
        self.clearAll()
    })
    .disposed(by: disposeBag)
```

+ Ở dòng .throttle ta thực hiện nhận chỉ 1 input của button trong 3 giây.
+ Kế tiếp ta thực hiện gọi method clearAll() để thực hiện xoá textField và label.

- Nếu ta muốn bắt các event khác của button thì ta có thể gọi controlEvent(_:) cho từng event khi được nhấn của button.
- Ví dụ:

```
button.rx
    .controlEvent(UIControl.Event.touchUpInside)
    .subscribe(onNext: { print("Button TouchUpInside") })
    .disposed(by: disposeBag)
```

II. TableView

1. Datasource

- Ta có thể thực hiện đặt 1 dataSource cho tableView 1 cách nhanh chóng mà không cần đặt dataSource như sau:
- Tạo 1 observable mà có element dạng array làm dataSource

```
let data = ["Hello", "Bonjour", "Stop", "End"]
```

```
let dataSource = Observable<[String]>.just(data)
```

- Bind dataSource đó cho tableView sử dụng function items(cellIdentifier:)

```
.bind(to: tableView.rx.items(cellIdentifier: "cell")){
    index, model, cell in
    cell.textLabel?.text = model
}
.disposed(by: disposeBag)
```

+ Index: là row của cell trong tableView
+ Model: là element bên trong array của observable.
+ Cell: cell của tableView.

2. Delegate Method

- Tương tự như dataSource ta có thể setup delegate method didSelectRowAtIndexPath của tableView mà không cần phải làm gì nhiều ngoài sử dụng wrapper modelSelectec(_:) của rx extension. Ví dụ:

```
tableView.rx
    .modelSelected(String.self)
    .subscribe(onNext: { model in
        print("\(model) was selected")
    })
    .disposed(by: disposeBag)
```

III. Collection View

- Tương tự như tableView.