RxSwift Basics – Day 3

Younghwan Kim

RxSwift Basics

- Day 1 – Observable, Operator (Filter, Transform, Combine)
- Day 2 – Subject (flatMap, flatMapFirst, flatMapLatest)
- **Day 3 – Two VCs communications with Subject, RxCocoa (Button)**

- Day 4 – Sequential, Merged Observable Calls
- Day 5 – RxCocoa, UI Binding (Button, TextField, Label, TableView)

Advanced RxSwift

- Day 1 – Protocol-Oriented Programming, Protocol Extension, Associatetype
- Day 2 – Network Call, Generic Enum
- Day 3 – Binding Track Activity (show / hide ‘Loading’), Scan Operator
- Day 4 – Adding a Reactive Extension to Custom UI Element,
- 2 Way Binding, Advanced TableView – RxDataSources
- Day 5 – Schedulers (observeOn, subscribeOn),

  Unit Test (RxTest, RxBlocking)

Two VCs : delegate - 1

```
class ViewController: UIViewController {
...


    @IBAction func showTwoOptionsDelegateVC(_ sender: UIButton) {
        let mainStoryboard: UIStoryboard = UIStoryboard(name: "Main", bundle: nil)
        if let vc = mainStoryboard.instantiateViewController(withIdentifier: "TwoOptionsDelegateVC") as? TwoOptionsDelegateVC {
                vc.delegate = self
            self.show(vc, sender: nil)
        }
    }
....
}

extension ViewController:  TwoOptionsDelegate {
    func optionOneSelected() {
        print("Option One Selected : TwoOptionsDelegateVC")
    }

    func optionTwoSelected() {
        print("Option Two Selected : TwoOptionsDelegateVC")
    }
}
```

Two VCs : delegate - 2

```
protocol TwoOptionsDelegate: class {
    func optionOneSelected()
    func optionTwoSelected()
}

class TwoOptionsDelegateVC: UIViewController {

    @IBOutlet weak var optionOneButton: UIButton!
    @IBOutlet weak var optionTwoButton: UIButton!

     weak var delegate: TwoOptionsDelegate?
....


    @IBAction func optionOneClicked(_ sender: UIButton) {
        if let _delegate = delegate {
            _delegate.optionOneSelected()
            dismiss(animated: true, completion: nil)
        }
    }

    @IBAction func optionTwoClicked(_ sender: UIButton) {
        if let _delegate = delegate {
            _delegate.optionTwoSelected()
            dismiss(animated: true, completion: nil)
        }
    }
}
```

Two VCs : Observable - 1

```
class ViewController: UIViewController {

    let disposeBag = DisposeBag()

....


    @IBAction func showTwoOptionsObservableVC(_ sender: UIButton) {
        let mainStoryboard: UIStoryboard = UIStoryboard(name: "Main", bundle: nil)
        if let vc = mainStoryboard.instantiateViewController(withIdentifier: "TwoOptionsObservableVC") as? TwoOptionsObservableVC {

            vc.selectedOption.subscribe(onNext: { option in
                    print("Option \(option) Selected : TwoOptionsObservableVC")
                })
                .disposed(by: disposeBag)

            self.show(vc, sender: nil)
        }
    }
}
```

Two VCs : Observable - 2

```
class TwoOptionsObservableVC: UIViewController {

    @IBOutlet weak var optionOneButton: UIButton!
    @IBOutlet weak var optionTwoButton: UIButton!

    let disposeBag = DisposeBag()

    private let selectedOptionSubject = PublishSubject<Int>()
    var selectedOption: Observable<Int> {
        return selectedOptionSubject.asObservable()
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        let oneObservable = optionOneButton.rx.tap.map { _ in return 1 }
        let twoObservable = optionTwoButton.rx.tap.map { _ in return 2 }

        Observable.of(oneObservable, twoObservable)
            .merge()
            .subscribe(onNext: { [unowned self] option in
                self.complete(action: option)
            })
            .disposed(by: disposeBag)
    }

    func complete(action: Int) {
        dismiss(animated: true) { [unowned self] in
            self.selectedOptionSubject.onNext(action)
            self.selectedOptionSubject.onCompleted()
        }
    }

}
```

Lab

Two VCs communications using subject