

RxSwift Basics – Day 5

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)



Simple Table View

```
@IBOutlet var tableView: UITableView!

func bindTableView() {
    let cities = Observable.of(["Lisbon", "Copenhagen", "London", "Madrid",
    "Vienna"])

    cities
        .bind(to: tableView.rx.items) {
            (tableView: UITableView, index: Int, element: String) in
            let cell = UITableViewCell(style: .default, reuseIdentifier:
"cell")
            cell.textLabel?.text = element
            return cell
        }
        .disposed(by: disposeBag)
}
```



Simple Table View

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



Table View – Multi Cell

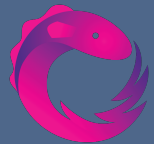
```
enum MyModel {  
    case text(String)  
    case pairOfImages(UIImage, UIImage)  
}  
  
let observable = Observable<[MyModel]>.just([  
    .textEntry("Paris"),  
    .pairOfImages(UIImage(named: "EiffelTower.jpg")!, UIImage(named:  
"LeLouvre.jpg")!),  
    .textEntry("London"),  
    .pairOfImages(UIImage(named: "BigBen.jpg")!, UIImage(named:  
"BuckinghamPalace.jpg")!)  
])
```



Table View – Multi Cell

```
observable.bind(to: tableView.rx.items) {
    (tableView: UITableView, index: Int, element: MyModel) in
    let indexPath = IndexPath(item: index, section: 0)

    switch element {
    case .textEntry(let title):
        let cell = tableView.dequeueReusableCell(withIdentifier: "titleCell",
        for: indexPath) as! TextCell
        cell.titleLabel.text = title
        return cell
    case .pairOfImages(let firstImage, let secondImage):
        let cell = tableView.dequeueReusableCell(withIdentifier:
        "pairOfImagesCell", for: indexPath) as! ImagesCell
        cell.leftImage.image = firstImage
        cell.rightImage.image = secondImage
        return cell
    }
}
.disposed(by: disposeBag)
```



TextField – Binding

```
class ViewController: UIViewController {  
    @IBOutlet weak var usernameTextField: UITextField!  
    @IBOutlet weak var passwordTextField: UITextField!  
    @IBOutlet weak var loginButton: UIButton!  
  
    let viewModel = LoginViewModel()
```

```
    usernameTextField.rx.text  
        .orEmpty  
        .bind(to: viewModel.username)  
        .disposed(by: disposeBag)
```

```
    passwordTextField.rx.text  
        .orEmpty  
        .bind(to: viewModel.password)  
        .disposed(by: disposeBag)
```

```
    viewModel.isValid  
        .bind(to: loginButton.rx.isEnabled)  
        .disposed(by: disposeBag)
```




TextField – Binding

```
struct LoginViewModel {  
  
    let username = BehaviorRelay<String>(value: "")  
    let password = BehaviorRelay<String>(value: "")  
  
    let isValid: Observable<Bool>  
  
    init() {  
        isValid = Observable.combineLatest(self.username.asObservable(), self.password.asObservable())  
        { (username, password) in  
            return username.count > 0 && password.count > 0  
        }  
    }  
}
```



Observable

- ▬ Create login and get profile observables

Sequential Call – Two Observables

RxCocoa UI Binding – button, textfield, TableView

Subject, Relay