Reactive Programming in Swift

RxDataSource创建UITableView - I (https://www.boxueio.com/series/reactive-programming-in-swift/ebook/81)

RxDelegate代理UITableView事件 (https://www.boxueio.com/series/reactive-programming-in-swift/ebook/83)

RxDataSource创建UITableView - II

Back to series (/series/reactive-programming-in-swift)

字号
字号
默认主题
金色主题
暗色主题

在上个视频中，使用 `rx_itemsWithCellIdentifier` 固然方便，但也有一个缺陷，就是我们只能创建包含一个section的table。如果我们要创建多个section，就需要自定义一个 `RxDatasouce` ，当然，这也并不是什么难事儿。我们修改一下这个例子，让Github的搜索结果10个一组分组显示。

首先，介绍一个新的类型： `SectionModel<Section, ItemType>` 。这是RxDatasource中定义的一个类型。它有两个泛型参数，第一个表示Section的值，第二个表示用于构建每一个Section里table cell内容的对象。

> 在这里要特别说明的是， `SectionModel` 的第二个参数ItemType， `SectionModel` 会在内部创建一个 `[ItemType]` ，用来初始化Section中的每一个table cell。

我们在 `ViewController` 里，用这个 `SectionModel` 创建一个自定义的data source：

```
typealias SectionTableModel =
    SectionModel<String, RepositoryModel>
let dataSource =
    RxTableViewSectionedReloadDataSource<
        SectionTableModel
    >()
```

这个data source的名字有点长， `RxTableViewSectionedReloadDataSource` ，这是RxDatasource中为创建带有section的 `UITableView` 定义的类型。

接下来，我们把Github的返回值，10个一组，装到 `SectionModel` 里。在 `ViewController` extension里，添加一个新的方法 `createGithubSectionModel` ：

```
private func createGithubSectionModel(
    repoInfo: [RepositoryModel]
) -> [SectionTableModel] {

    var ret: [SectionTableModel] = []
    var items: [RepositoryModel] = []

    if (repoInfo.count <= 10) {
        let sectionLabel = "Top 1 - 10"
        items = repoInfo

        ret.append(SectionTableModel(
             model: sectionLabel, items: items))
    }
    else {
        for i in 1...repoInfo.count {
            items.append(repoInfo[i - 1])

            if (i / 10 != 0 && i % 10 == 0) {
                let sectionLabel =
                    "Top \(i - 9) - \(i)"

                ret.append(
                    SectionTableModel(
                        model: sectionLabel,
                        items: items))

                items = []
            }
        }
    }

    return ret
}
```

这样，我们就得到了一个数组，数组的索引就是 UITableView 中section的索引，数组的值，就是初始化对应section里所有table cell的 SectionTableModel 对象。

这样，在 repositoryName.rx_text 的 onNext 里，我们就可以直接，订阅 dataSource 了。把之前创建 UITableView 的代码都先注释掉。换成下面的代码：

```
Observable.just(
    self.createGithubSectionModel(
        repositoryModelArray))
    .bindTo(
        self.searchResult.rx_itemsWithDataSource(
            dataSource))
    .addDisposableTo(self.bag)
```

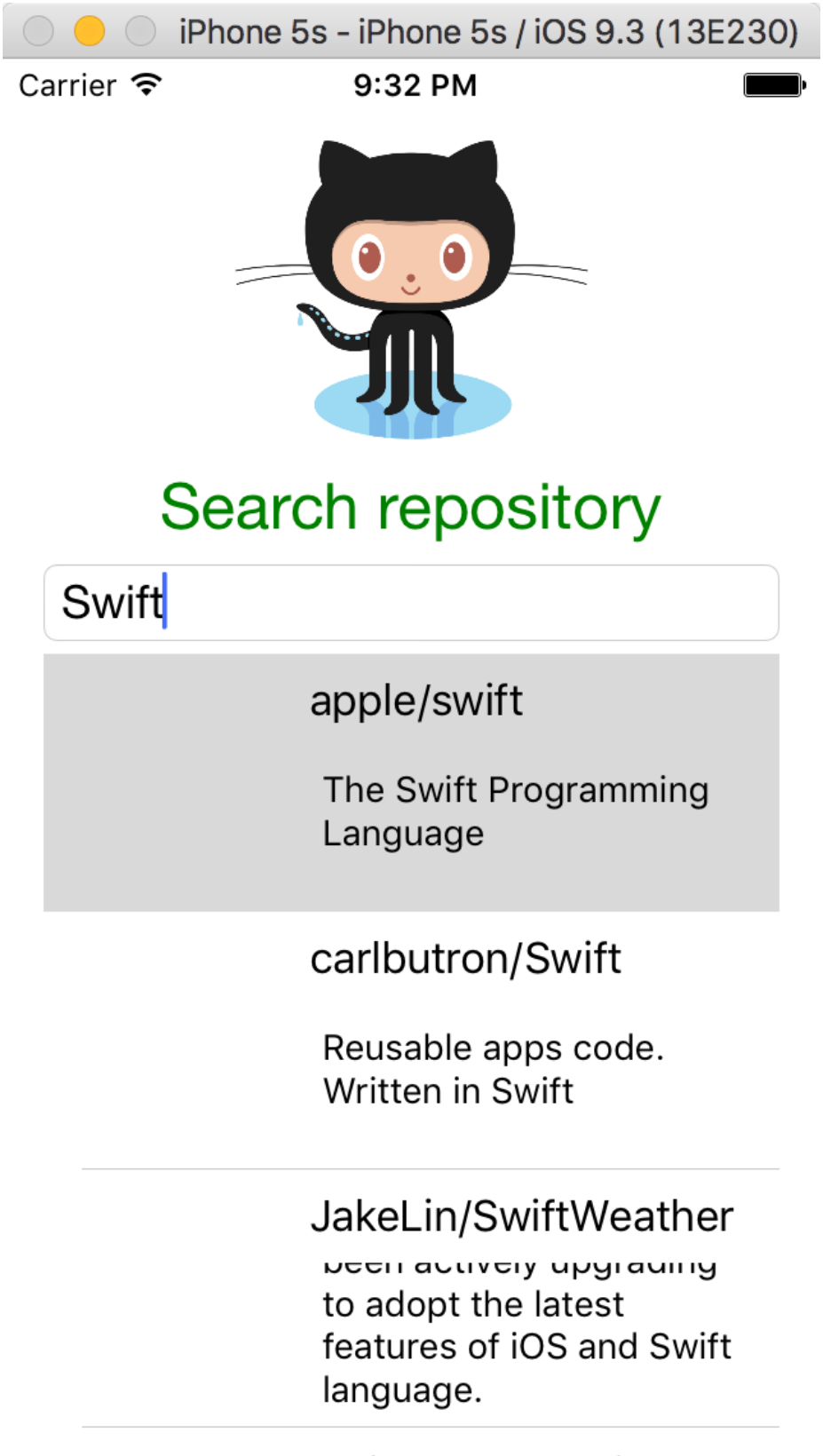
最后，我们要告诉dataSource，如何根据订阅到的内容生成table cell。这和我们上个视频中设置的 curriedArgument 是类似的。在 viewDidLoad 方法里，添加下面的代码：

```
self.dataSource.configureCell = {
    (_, tv, indexPath, element) in
    let cell =
        tv.dequeueReusableCellWithIdentifier(
            "RepositoryInfoCell",
            forIndexPath: indexPath)
            as! RepositoryInfoTableViewCell

    cell.name.text = element.name
    cell.detail.text = element.detail

    return cell
}
```

Command + R 编译执行，可以看到，结果和我们想象的略有出入：



在 UITableView 里，Section header没有显示出来，选中一个Cell时，高亮效果也没有自动消失。这是由于我们还没有设置 UITableView 的delegate。当然，可以尝试用传统的方式为 UITableView 设置 delegate，它们可以和RxSwift很好的协同工作。

Next?

这就是，我们这段视频的内容。也许新事物有点多。我们了解了 subscribe 的一个新的用法，以及通过 RxDatasource创建 UITableView 的两种方法。在下一段视频中，我们将自己实现一个DelegateProxy，来和大家分享如何用reactive的方式，实现 UITableView delegate。