Todo IV - 进一步理解Subject的实际应用

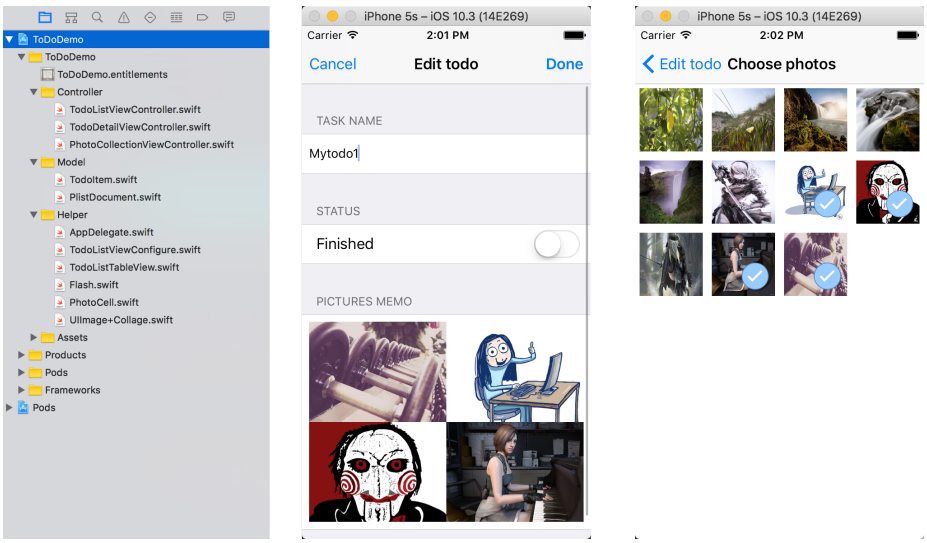
Back to series (/series/rxswift-101)

字号
字号
默认主题
金色主题
暗色主题

在了解了常用过滤型operator的基本概念和用法之后，在这段视频里，我们给之前的ToDoDemo添加一个功能，以此进一步了解这些operators在App开发中的用法。大家可以在这里下载(https://github.com/puretears/RxToDoDemo/tree/master/ToDoDemoStarter-IV)项目的初始模板，在这里下载 (https://github.com/puretears/RxToDoDemo/tree/master/ToDoDemoFinish-IV)项目的完成版本。

我们要做什么?

在开始之前，先来看下要完成的功能：



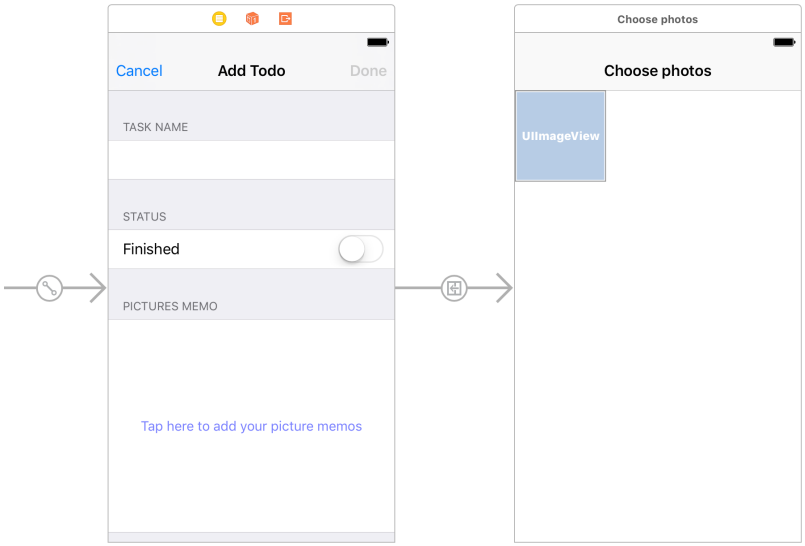
- 允许用户给Todo添加一个图片备忘；
- 允许用户从照片库中直接选择图片，并自动合成到一张图片中；

基本上，大的功能就这两部分，看起来不太复杂，但其中包含了诸多会用到过滤型operator的实现细节，我们将会一一看到。

对项目模板做了哪些修改?

在继续之前，先来看下基于上一次完成的版本，我们对模板进行了哪些修改。

对storyboard和view controller的修改



首先是storyboard，在 TodoDetailViewController 对应的UI上，底部添加了一个 UIButton ，稍后，我们会把合成之后的图片设置为这个按钮的背景图片，用于显示用户选择的结果。点击这个button，会跳转到图片选择UI，为此：

- 我们给这个按钮添加了一个segue；
- 在 TodoDetailViewController 中添加了两个属性：一个 @IBOutlet 表示按钮，一个 UIImage 表示合成后的图片；

TodoDetailViewController

```
class TodoDetailViewController: UITableViewController {
    // ...
    fileprivate var todoCollage: UIImage?
    @IBOutlet weak var memoCollageBtn: UIButton!
    // ...
}
```

- 在 TodoDetailViewController.viewDidLoad 方法中，添加了初始化图片备忘按钮背景的代码：

```
override func viewDidLoad() {
    // ...

    if todoItem.pictureMemoFilename != "" {
        let url = getDocumentsDir().appendingPathComponent(
            todoItem.pictureMemoFilename)

        if let data = try? Data(contentsOf: url) {
            self.memoCollageBtn.setBackgroundImage(
                UIImage(data: data),
                for: .normal)

            self.memoCollageBtn.setTitle("", for: .normal)
        }
    }

    // ...
}
```

- 在 TodoDetailViewController 中添加了以下方法，其中： resetMemoBtn / setMemoBtn 用于重置和设置按钮的默认样式； savePictureMemos 用于保存为用户合成的图片并返回图片的名称； setMemoSectionHeaderText 用于在用户选择完图片后，提示对应的图片个数。稍后，我们会逐步实现这些方法；

```
extension TodoDetailViewController {
    fileprivate func resetMemoBtn() { // ... }
    fileprivate func setMemoBtn(bkImage: UIImage) { // ... }
    fileprivate func savePictureMemos() -> String { //... }
    func setMemoSectionHederText() { // ... }
}
```

PhotoCollectionViewController

另外，为了显示照片库中的所有图片，我们添加了一个 PhotoCollectionViewController ，其中实现了用 UICollectionView 显示照片，并通过一个蓝色的对勾显示选中图片的效果。

添加的helper方法

在项目的*Helper group*中，新添加了两个文件：

- *PhotoCell.swift*，其中定义了 PhotoCollectionViewController 中使用的collection cell；
- *UIImage+Collage.swift*，其中，为 UIImage 添加了一个 extension ，包含了用于缩放图片（ scale ）、合成用户选中的图片（ collage ）以及根据图片内容比较两个图片是否相等（ isEqual (lhs: rhs:) ）的方法；

对model的修改

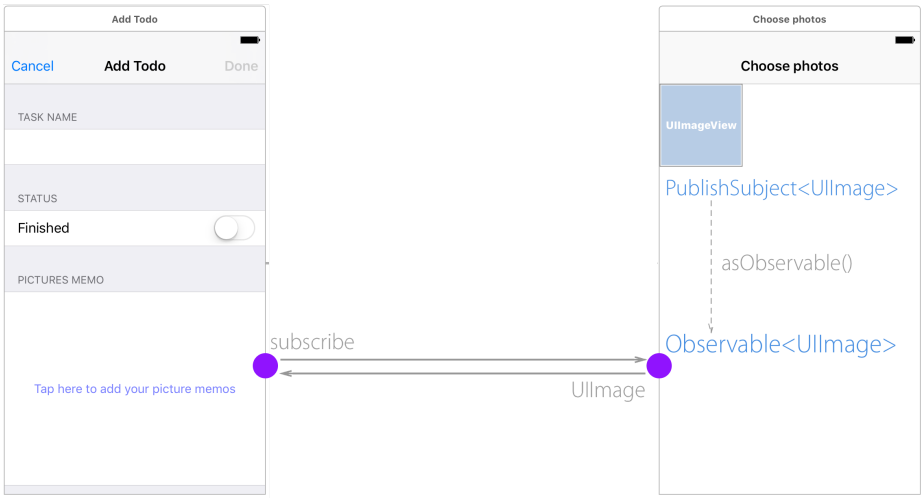
在 TodoItem 中，新添加了一个属性 pictureMemoFilename ，表示每一个todo对应的图片名，默认是空字符串。并对 TodoItem 中对应的 init 以及序列化方法进行了修改：

```
class TodoItem: NSObject, NSCoding {
    // ...
    var pictureMemoFilename: String = ""
    // ...
}
```

了解了以上修改之后，我们就可以实现对应的功能了。第一个要完成的，就是合成用户选择的图片，并把它显示出来。

用Rx的方式实现图片的选取与合成

这个功能实现的起点，在 PhotoCollectionViewController ，我们需要通知 TodoDetailViewController 用户选择的每一张图片，进而在 TodoDetailViewController 里，完成图片的合成、显示以及保存。这个过程用一张图来表示，就是这样的：



PhotoCollectionViewController

首先，来完成 PhotoCollectionViewController 的部分，在它的定义里，添加下面的代码：

```
class PhotoCollectionViewController: UICollectionViewController {
    // ...

    fileprivate let selectedPhotosSubject = PublishSubject<UIImage>()
    var selectedPhotos: Observable<UIImage> {
        return selectedPhotosSubject.asObservable()
    }
    let bag = DisposeBag()

    // ...
}
```

同样，我们用了之前视频里介绍的一个技巧，为了避免来自外部“伪造”的图片选中事件，我们让 selectedPhotosSubject 的访问级别是 fileprivate ，然后，把它Observable的一面，用另外一个属性暴露给选中图片事件的订阅者。

其次，在 UICollectionView 单元格被选中的处理方法里，添加下面的代码：

```
override func collectionView(_ collectionView: UICollectionView,
                             didSelectItemAt indexPath: IndexPath) {
    // 1. Get photo object
    let asset = photos.object(at: indexPath.item)
    // 2. Flip the checked status
    if let cell = collectionView.cellForItem(at: indexPath) as? PhotoCell
 {
        cell.selected()
    }

    imageManager.requestImage(for: asset,
        targetSize: view.frame.size,
        contentMode: .aspectFill,
        options: nil,
        resultHandler: { [weak self] (image, info) in
            guard let image = image, let info = info else { return }

            if let isThumbnail = info[PHImageResultIsDegradedKey] as? Bool
,
               !isThumbnail {
                // 3. Trigger event if the image is not an icloud
                // thumbnail.
                self?.selectedPhotosSubject.onNext(image)
            }
        })
}
```

其中，前半部分获取图片对象、反选单元格的代码都很简单，重点其实只有 `requestImage` 的 `requestHandler` 参数，在这里，只要图片库中的图片不是iCloud中的缩略图，我们就把它作为 `selectedPhotosSubject` 的 `.next` 事件发送。

最后，在 `PhotoCollectionViewController.viewWillDisappear` 方法里，我们给 `selectedPhotosSubject` 发送完成事件，表示结束：

```
override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
    selectedPhotosSubject.onCompleted()
}
```

这样， `PhotoCollectionViewController` 的部分就完成了。接下来，我们完成 `TodoDetailViewController` 的部分。

TodoDetailViewController

首先，给 `TodoDetailViewController` 添加一个 `Variable` ，它的值类型是 `[UIImage]` ，用来保存用户选中的所有图片：

```
class TodoDetailViewController: UITableViewController {
    fileprivate let images = Variable<[UIImage]>([])
    // ...
}
```

其次，在 `TodoDetailViewController.prepare(segue:sender:)` 方法里，我们先获取之前添加的 `selectedPhotos` ：

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let photoCollectionViewController =
        segue.destination as! PhotoCollectionViewController
    images.value.removeAll()
    resetMemoBtn()

    let selectedPhotos = photoCollectionViewController.selectedPhotos
    // ...
}
```

第三，我们从 `selectedPhotos` 中订阅到用户选择的所有图片：

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    // ...
    _ = selectedPhotos.subscribe(onNext: { image in
            self.images.value.append(image)
        }, onDisposed: {
            print("Finished choose photo memos.")
        })
    // ...
}
```

第四，让 image 是一个Observable，我们订阅它的值并更新UI。在 TodoDetailViewController.viewDidLoad 方法里，订阅这个 Variable：

```
override func viewDidLoad(){
    // ...

    images.asObservable().subscribe(onNext: {
        [weak self] images in
        guard let `self` = self else {
            return
        }
        guard !images.isEmpty else {
            self.resetMemoBtn()
            return
        }

        /// 1. Merge photos
        self.todoCollage = UIImage.collage(images: images,
            in: self.memoCollageBtn.frame.size)

        /// 2. Set the merged photo as the button background
        self.setMemoBtn(bkImage: self.todoCollage ?? UIImage())
    }).addDisposableTo(bag)

    // if ...
}
```

订阅后的处理逻辑很简单，就像代码注释中说明的那样，分成两个步骤：

1. 合成所有用户选择的图片；
2. 把合成后的图片设置为按钮的背景；

要注意的是，**这段订阅的代码一定要放在 viewDidLoad 中初始化UI的代码前面**，否则，当用户打开编辑Todo的时候，由于此时还未选择任何图片，订阅代码会重置 memoCollageBtn 的状态，我们就看不到之前合成的图片了。

最后，在处理Done按钮的 IBOutlet 方法里，我们保存合成后的图片，并结束 todoSubject 序列。

```
@IBAction func done() {
    todoItem.name = todoName.text!
    todoItem.isFinished = isFinished.isOn
    todoItem.pictureMemoFilename = savePictureMemos()

    todoSubject.onNext(todoItem)
    todoSubject.onCompleted()
    dismiss(animated: true, completion: nil)
}
```

What's next?

至此，保存图片备忘的基本功能就实现了，但是，我们还没有在合成图片的时候，动态的修改对应的 section header text，这关系到一个值得思考的问题：同一个Observable可以订阅多次么？下一节，我们就来讨论这个话题，并实现修改section header text的功能。

⏮ 常用的获取事件操作符
(https://www.boxueio.com/series/rxswift-101/ebook/241)

Todo V - 理解重复订阅Observable的行为 ⏭
(https://www.boxueio.com/series/rxswift-101/ebook/243)

职场漂泊的你，每天多学一点。

从开发、测试到运维，让技术不再成为你成长的绊脚石。我们用打磨产品的精神去传播知识，把最新的移动开发技术，通过简单的图表， 清晰的视频，简明的文字和切实可行的例子一一向你呈现。让学习不仅是一种需求，也是一种享受。

泊学动态

一个工作十年PM终创业的故事（二） (https://www.boxueio.com/after-the-full-upgrade-to-swift3)
Mar 4, 2017

人生中第一次创业的"10有" (https://www.boxueio.com/founder-chat)
Jan 9, 2016

猎云网采访报道泊学 (http://www.lieyunwang.com/archives/144329)
Dec 31, 2015

What most schools do not teach (https://www.boxueio.com/what-most-schools-do-not-teach)
Dec 21, 2015

一个工作十年PM终创业的故事（一） (https://www.boxueio.com/founder-story)
May 8, 2015

泊学相关

关于泊学 ›

加入泊学 ›

泊学用户隐私以及服务条款 (HTTPS://WWW.BOXUEIO.COM/TERMS-OF-SERVICE)

版权声明 (HTTPS://WWW.BOXUEIO.COM/COPYRIGHT-STATEMENT)

联系泊学

Email: 10[AT]boxue.io (mailto:10@boxue.io)

QQ: 2085489246

友情链接 SwiftV (http://www.swiftv.cn) | Seay信息安全博客 (http://www.cnseay.com) | Swift.gg (http://swift.gg/) | Laravist (http://laravist.com/) | SegmentFault (https://segmentfault.com) | 靛青K的博客 (http://blog.dianqk.org/)