

如何在不同的Observables之间跳转

⌕ Back to series (/series/rxswift-101)

在平时的编程中，我们还会遇到另一类问题，两个Observable中的内容，在逻辑上，是有关联的。例如，当我们提交一个表单的时候，就同时存在着两个事件序列，一个是提交按钮的点击事件序列；另一个是表单内值的序列。每次点击提交按钮的时候，都需要知道当前表单中的内容，以便发送给服务器。这时，我们就需要一种方式，在不同的Observable中进行切换。

🔍 字号

🔍 字号

🖌️ 默认主题

🖌️ 金色主题

🖌️ 暗色主题

模拟表单提交

为了模拟我们在一开始提到的表单提交场景，我们先创建两个Observables：

```
let textField = BehaviorSubject<String>(value: "boxu")
let submitBtn = PublishSubject<Void>()
```

其中，我们把 textField 定义成了一个 BehaviorSubject<String>，表示输入框的当前值是 boxu。而 submitBtn 则表示提交按钮的点击事件序列。

现在，为了模拟点击按钮的时候，订阅到表单的当前值，我们可以使用 withLatestFrom operator：

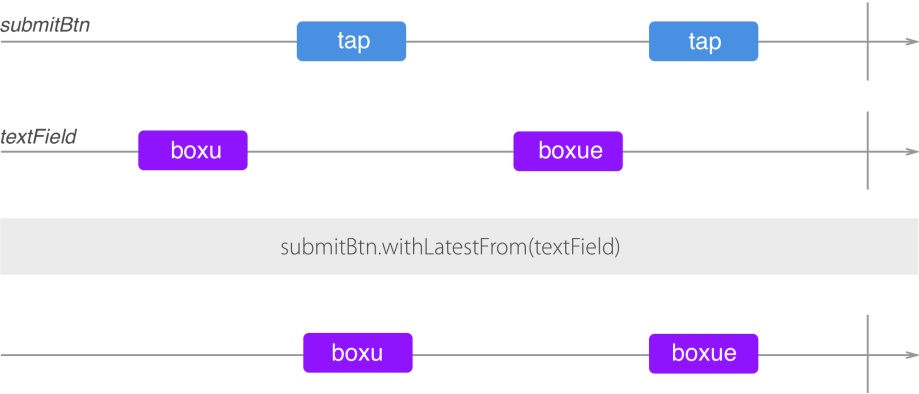
```
submitBtn.withLatestFrom(textField)
    .subscribe(onNext: { dump($0) })
    .addDisposableTo(bag)
```

上面的代码可以理解为，每当 submitBtn 中有事件发生的时候，就读取 textField 中的最新事件。这样，就实现submit按钮被点击的时候，订阅到当前表单内容的效果了。我们用下面的事件序列试一下：

```
submitBtn.onNext()
textField.onNext("boxue")
submitBtn.onNext()
```

执行一下就可以在控制台看到，一定会订阅到两个结果：第一次，是 submitBtn.onNext() 之后，此时 textField 中的值是 boxu；然后，我们用 textField.onNext("boxue") 模拟了 textField 的值被修改的事件；最后，当我们第二次调用 submitBtn.onNext() 的时候，就会订阅到 boxue 了。

用序列图表示，就是这样的：



在多个Observables之间进行跳转

除了把一个Observable中的事件作为跳转到另外一个Observable的“触发器”之外，我们还可以让订阅到的内容在不同的Observable之间跳转。

为了理解这个过程，我们来看代码从提交到测试这个工作流的场景。首先，我们定义两个事件序列，一个表示编码，一个表示测试：

```
let coding = PublishSubject<String>()
let testing = PublishSubject<String>()
```

其次，为了表示从提交到测试的往复过程，我们得定义一个 `Observable<Observable<String>>`，它的事件类型是 `Observable<String>`，表示了 `coding` 或者 `testing` workflow：

```
let working = PublishSubject<Observable<String>>()
```

第三，为了能模拟在 `coding` 和 `testing` 这两个Observables中切换的过程，我们对 `working` 使用 `switchLatest()` operator：

```
working.switchLatest()
    .subscribe(onNext: { dump($0) })
    .addDisposableTo(bag)
```

第四，我们来看下面这个事件序列：

```
working.onNext(coding)
coding.onNext("version1")

working.onNext(testing)
testing.onNext("FAILED")

working.onNext(coding)
coding.onNext("version1")

working.onNext(testing)
testing.onNext("PASS")
```

它看着有点儿复杂，我们来分别看一下：

第一对，`working.onNext(coding)` 表示当前的 workflow 是 `coding`，然后，我们用 `coding.onNext("version1")` 表示提交了一个版本；此时，我们会订阅到字符串 `Version1`。

第二对，我们用 `working.onNext(testing)` 表示当前 workflow 切换到了 `testing`。这样，我们就只能订阅到 `testing` 中发生的事件了。于是，我们用 `testing.onNext("FAILED")` 表示测试失败了，我们会订阅到 `FAILED` 事件。

第三对，我们用 `working.onNext(coding)` 又回到了 `coding`。现在，又可以订阅到 `coding` Observable 中的事件了。我们用 `coding.onNext("version1")` 模拟又提交了一次 `Version1`。

最后，我们又切换到了 `testing`，在 `testing.onNext("PASS")` 之后，就会订阅到 `PASS` 事件了。

综上所述，执行一下上面的代码，就会看到下面这样的结果：

```
===== switchLatest =====
- "version1"
- "FAILED"
- "version1"
- "PASS"
```

理解了 `switchLatest` 的用法之后，我们还可以用下面的事件序列试一下：

```
working.onNext(coding)
coding.onNext("version1")

working.onNext(testing)
testing.onNext("FAILED")

coding.onNext("version2") // Cannot subscribe this event
```

这次，我们在切换到 `testing` Observable 之后，用 `coding.onNext("version2")` 模拟了提交一个新版本。但实际上，我们并不会订阅到这个事件。大家可以自己试一下。

What's next?

以上，就是在不同的Observable之间切换事件的用法。我们用三节内容，和大家分享了合并序列本身、合并序列事件以及在事件序列之间切换的方法。实际上，我们使用过的所有这些operators，有一个共同的名字，叫做：**combine operators**。在继续之前，我们应该确保已经掌握了它们的含义和用法。从下一节开始，我们来看另外一大类和时间有关的operators。



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

2017 © Boxue, All Rights Reserved. 京ICP备15057653号-1 (http://www.miibeian.gov.cn/) 京公网安备 11010802020752号 (http://www.beian.gov.cn/portal/registerSystemInfo?recordcode=11010802020752)

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/)