What's new in Swift 4

SE-0148使用泛型下标操作符

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/238)

使用Codable解析JSON

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/294)

SE-0156 Subtype existential

Back to series (/series/what-is-new-in-swift-4)

字号

字号

默认主题

金色主题

暗色主题

在Objective-C里，声明变量的时候，我们可以用下面的形式表达类型和protocol的关系：

```
id<Protocol1, Protocol2>
Base<Protocol>*
```

例如，第一个就表示任意遵从 `Protocol1` 和 `Protocol2` 的类型，第二个则表示遵从 `Protocol` 的 `Base` 类。为了实现同样的功能，SE-0156 (https://github.com/apple/swift-evolution/blob/master/proposals/0156-subclass-existentials.md)为Swift 4添加了下面的特性。

用AnyObject约束任意遵从protocol的类

当我们要约束**任意一个遵从 P 的类**时，可以这样：

```
protocol P {}
struct S : P {}
class C : P {}
class D { }

let u: AnyObject & P = C()
let v: P & AnyObject = C()
```

在上面的例子里， `AnyObject` 表示任意一个 `class` ， `P` 表示要遵从的 `protocol` ，用 `&` 把它们连接起来，就达成了我们要对类型进行的约束。不过，就像你看到的一样，它们的先后并没有关系。理解了它的含义之后，就不难理解，下面的代码，都会触发编译错误：

```
// !! Compile time error: `S` is not a class !!
let t: AnyObject & P = S()
// !! Compile time error: `D` is not conform to P !!
let w: AnyObject & P = D()
```

用具体的类名约束遵从protocol的类以及派生类

除了 `AnyObject` 之外，我们也可以限制某个遵从了protocol的具体类以及它的派生类，像这样：

```
protocol P {}
struct S : P {}
class C {}
class D: P {}
class E: C, P {}

let w: C & P = E()
```

其中， `C & P` 这样的写法，表示遵从了 `protocol P` 的类 `C` 以及 `C` 的派生类。因此，下面两种情况都会导致编译错误：

首先，不能限制 `struct` 类型的 `protocol` 约束：

```
let u: S & P
```

其次，不是的 `C` 的派生类无法通过编译：

```
let v: C & P = D()
```

另外，当我们在 `protocol` 约束中，同时使用了 `AnyObject` 和具体的类名，则具体的类名会覆盖 `AnyObject` ，也就是说，下面这两种表达方式是相同的：

```
let w: C & P = E()
let w: AnyObject & C & P = E()
```

而下面的代码同样无法通过编译：

```
let v: AnyObject & C & P = D()
```

实际上，关于protocol-constraints的用法，还有一些更复杂的情况，例如，在约束中串联多个 `class` 或者 `typealias` ，大家可以在SE-0165 (https://github.com/apple/swift-evolution/blob/master/proposals/0156-subclass-existentials.md)中找到关于这个特性的详细规格。这里，我们就不一一展开了，最重要的是，我们要知道，在Swift 4里，多了一种可以进一步约束类型的用法。

带来的潜在影响

基于这个特性，之前的一些Objective-C的代码桥接到Swift 4，采用protocol-constraints语法的时候，会带来源代码的不兼容。来看下面的例子：

```
@interface MyViewController
- (void)setup:(
    nonnull UIViewController<
    UITableViewDataSource,
    UITableViewDelegate>*)tableViewController;
@end
```

当上面这段代码桥接到Swift 3时，会忽略掉 `setup` 参数的 `UITableViewDataSource` 和 `UITableViewDelegate` 的约束，变成这样：

```
class MyViewController {
    func setup(
        tableViewController: UIViewController) {}
}
```

于是，在Swift里，我们可以给 `setup` 传递任何一个 `UIViewController` 对象，甚至这个对象没有遵从 `UITableViewDataSource` 和 `UITableViewDelegate` 。

但桥接到Swift 4后， `MyViewController` 就会变成这样：

```
class MyViewController {
    func setup(
        tableViewController:
            UIViewController &
            UITableViewDataSource &
            UITableViewDelegate) {}
}
```

这时，对于那些没有遵从 `UITableViewDataSource` 和 `UITableViewDelegate` 的 `UIViewController` 对象，如果传递给 `setup` ，就无法通过编译了。

◀ SE-0148使用泛型下标操作符 (https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/238)

使用Codable解析JSON ▶ (https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/294)

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/)