What's new in Swift 4

SE-0169改进的private访问权限

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/234)

Dictionary初始化以及常用操作的诸多改进

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/236)

更智能安全的Key Value Coding

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Swift里，有一种间接访问类属性的方法，叫做 #keyPath 。来看下面的例子：

```
class Foo: NSObject {
    @objc var bar = "bar"
    @objc var baz = [1, 2, 3, 4]
}

var foo = Foo()
print(foo.bar)
foo.bar = "BAR"
```

除了像上面这样直接访问属性之外，我们还可以这样：

```
var bar = foo.value(forKeyPath: #keyPath(Foo.bar))
foo.setValue("BAR", forKeyPath: #keyPath(Foo.bar))
```

这就是Cocoa中的KVC机制，在Objective-C中它可以很好的工作，但移植到Swift之后，它的不足就显现出来了：

- value(forKeyPath:) 方法返回的类型是 Any? ，这样我们就失去了类型信息，错误的赋值会直接导致运行时错误；
- 只有 NSObject 的派生类才支持这种访问机制，进而导致了只有在 Darwin 平台上才可以使用；

类型更安全的KeyPath

为此，Swift 4中设计了更智能的KeyPath，像这样：

```
let barKeyPath = \Foo.bar
var bar = foo[keyPath: barKeyPath]
foo[keyPath: barKeyPath] = "BAR"
```

其中， \Foo.bar 就是Swift 4中新的key path用法，它是一个独立的类型，带有属性的类型信息，因此，编译器会发现错误类型的赋值，而不会把这个错误推迟到运行时：

```
// error: cannot assign value of type 'Int' to type 'String'
foo[keyPath: barKeyPath] = 1
```

适应性更好的KeyPath

除了类型安全之外，新的KeyPath用法适应性也更好，我们无须要求 Foo 是 NSObject 的派生类，也不必用 @objc 修饰属性，把 Foo 改成这样，之前的代码仍旧可以正常工作：

```
class Foo {
    var bar = "bar"
    var baz = [1, 2, 3, 4]
}
```

甚至，我们都无须要求 Foo 是一个 class ， struct 属性同样可以使用新的KeyPath：

```
struct Foo {
    var bar = "bar"
    var baz = [1, 2, 3, 4]
}
```

可以在KeyPath中使用下标操作符（尚未实现）

SE-0161 (https://github.com/apple/swift-evolution/blob/master/proposals/0161-key-paths.md)中提议的新型KeyPath当前还没有全部实现完成，但在它的设计里，KeyPath中是可以使用下标操作符的，因此，为了访问 baz 中的第1个元素，我们本可以这样：

```
// error: INTERNAL ERROR: feature not implemented
foo[keyPath: \Foo.baz[1]]
```

但由于，这个功能在录制这个视频的时候还没有实现，因此，我们会看到一个编译错误。这并不重要，我们只要了解这个用法就好了。

SE-0169改进的private访问权限
(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/234)

Dictionary初始化以及常用操作的诸多改进
(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/236)