理解值语义的自定义类型

定义更复杂的值 - struct (https://www.boxueio.com/series/understand-value-types/ebook/170)

如何为值类型实现Copy-On-Write? - I (https://www.boxueio.com/series/understand-value-types/ebook/172)

# 不再只是“值替身”的enum

Back to series (/series/understand-value-types)

字号
字号
默认主题
金色主题
暗色主题

很多时候，我们需要用一组特定的值，来表达一个公共的含义。例如用1，2，3，4表示东、南、西、北：

```
let EAST  = 1
let SOUTH = 2
let WEST  = 3
let NORTH = 4
```

或者用一个字符串表示一年的月份：

```
let months = ["January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"]
```

在上面这些例子里，无论是用数字表示方向，还是用字符串表示月份，它们都有一个共同的问题：**我们让一个类型承载了本不属于他的语意**。因此我们无法安全的避免“正确的类型，却是无意义的值”这样的问题。例如：数字5表示什么方向呢？Jan.可以用来表示一月么？还有JAN呢？

因此，面对“把一组有相关意义的值定义成一个独立的类型”这样的任务，Swift为我们提供了一种类型安全的方式，叫做 enum 。

enum 并不是一个新生事物，几乎任何一种编程语言都有和 enum 类似的语法概念。但是Swift对 enum 做了诸多改进和增强，它已经不再是一个简单的“值的替身“。它可以有自己的属性、方法，还可以遵从 protocol 。在前面的章节里，我们也已经不止一次使用过了 enum ，和 struct 类似，这里，我们集中对 enum 的特性做一个总结。

## 定义一个enum

对于我们一开始提到过的方向和月份信息，我们可以用 enum 定义成这样：

```
enum Direction {
    case east
    case south
    case west
    case north
}

enum Month {
    case january, februray, march,
        april, may, june, july,
        august, september, october,
        november, december
}
```

这样，我们就用 enum 定义了两个新的类型，用来表示“方向”和“月份”，它们是两个有限定值的类型。然后，我们可以像下面这样，使用它们代表的值：

```
let NORTH = Direction.north
let JAN = Month.january
```

直观上看，使用 enum 比直接使用数字和字符串有很多“天生”的好处：一来我们可以借助IDE提供的auto complete避免输入错误；二来，使用 enum 是类型安全的，需要使用方向和月份内容的时候，不会发生“类型正确，值却无意义的情况”。

## 理解enum的“各种”value

在Swift里， enum 的值，可以通过不同的方式表达出来。而不像Objective-C，只能通过一个整数来替代。

## case自身就是enum的值

例如在上面的例子里，当我们当我们使用 Direction.north 时，我们就已经在使用一个 enum 的值了，它的 case 就是它的值本身，我们无需特意给它找一个“值替身”来表示。另外，如果编译器可以通过type inference可以推导出 enum 的类型，我们可以在读取值的时候，省掉 enum 的名字：

```
func direction(val: Direction) -> String {
    switch val {
    case .north, .south:
        return "up down"
    case .east, .west:
        return "left right"
    }
}
```

这个例子里，有两个地方值得注意：

- 因为val的类型可以通过type inference推导出是 Direction ，因此，在 case 里，我们可以省略掉 enum 的名字；
- 对于一个 enum 来说，它全部的值就是 switch 可能包含所有的 case ，因此在一个 switch...case... 里，只要列举了 enum 所有的 case ，它就被认为是exhaustive的，因此，switch 可以没有 default 分支；

## Raw value

和Objective-C不同，Swift的 enum 默认不会为它的 case “绑定”一个整数值。但这并不妨碍你手工给 case “绑定”一个，而这样“绑定”来的值，叫做raw values。

```
enum Direction: Int {
    case east
    case south
    case west
    case north
}
```

对于如此定义的 Direction ，Swift就会依次把 east / south / west / north “绑定”上1 / 2 / 3 / 4。但我们也可以像下面这样给所有的 case 单独指定值：

```
enum Direction: Int {
    case east  = 2
    case south = 4
    case west = 6
    case north = 8
}
```

或者，我们可以给所有的 case 指定一个初始值：

```
enum Month: Int {
    case January = 1, Februray, March,
        April, May, June, July,
        August, September, October,
        November, December
}
```

这样，Swift就会自动为其他月份“绑定”对应的整数值了。如果我们要读取 enum 的raw value，可以访问 case 的 rawProperty 方法：

```
let NORTH = Direction.north.rawValue // 1
let JAN = Month.January.rawValue // 1
```

## Associated value

Raw value的各种机制和方式，传统且易于理解，它最接近我们对传统 enum 的认知。但这并不是给 enum “绑定”值的唯一办法，在Swift里，我们甚至可以给每一个 case “绑定”不同类型的值。我们管这样的值叫做associated values。

例如，我们定义一个表达HTTP action的 enum：

```
enum HTTPAction {
    case get
    case post(String)
    case delete(Int, String)
}
```

我们在每一个需要有associated value的 case 后面放上和 case 对应的值的类型，就可以了。然后，我们可以这样来使用HTTPAction：

```
var action1 = HTTPAction.get
var action2 = HTTPAction.post("BOXUE")

switch action1 {
case .get:
    print("HTTP GET")
case let .post(msg):
    print("\(msg)")
case .delete(let id, let data):
    print("id = \(id), data = \(data)")
}
```

这个例子里，有两点是应该注意的：

- 不是每一个 case 必须有associated value，例如 .get 就只有自己的enum value；
- 当我们想“提取”associated value的所有内容时，我们可以把 let 或 var 写在 case 后面，例如 .post 的用法；
- 当我们想分别“提取”associated value中的某些值时，我们可以把 let 或 var 写在associated value里面，例如 .delete 的用法；

## enum是一个值类型，也可以是一个引用类型

在Swift里， enum 默认也是一个值类型，也就说，每一个 enum 对象，都只能有一个owner，因此，你无法创建指向同一个 enum 对象的多个引用。

但有一种特殊的情况，可以改变 enum 的这个属性。例如，我们之前实现过的链表(https://boxueio.com/series/advanced-collections/ebook/165)：

```
enum List {
    case end
    indirect case node(Int, next: List)
}
```

我们可以使用 indirect 修饰一个 case ，这样当一个 List 为 case node 时，它就变成了一个引用类型，多个 case node 可以指向同一个 List 对象：

```
let end = List.end
let list1 = List.node(1, next: end)
let list2 = List.node(2, next: end)
```

此时， list1 和 list2 就指向了同一个 end 对象。

## What's next?

以上，就是关于 enum 的主要内容，相比其它编程语言， enum 在值的呈现方式上更为强大和灵活。因此，我们应该掌握这个在其他语言中有些被边缘化的类型，以不断改进我们的代码质量。接下来，我们将详细讨论一个之前已经被多次提及的话题：copy on write。到底在哪些场景里适合使用COW？该如何为我们自定义的类型实现COW？在下一节里，我们就来一探究竟。

◀ 定义更复杂的值 - struct
(https://www.boxueio.com/series/understand-value-types/ebook/170)

如何为值类型实现Copy-On-Write？ - I ▶
(https://www.boxueio.com/series/understand-value-types/ebook/172)

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/)