SR-1446 关于内嵌类型的两种改进

Back to series (/series/what-is-new-in-swift-31)

字号
字号
默认主题
金色主题
暗色主题

这一节，我们来聊聊和内嵌类型有关的话题。在Swift 3.1里，内嵌类型有了两方面的改进：

- 普通类型的内嵌类型可以直接使用其外围类型的泛型参数，此时它仍旧是一个普通类型；
- 泛型类型的内嵌类型可以拥有和其外围类型完全不同的泛型参数；

我们通过实现一个简单的链表，来介绍这两个改进。

首先，定义链表本身，它当然是一个泛型类，其中的泛型参数，表示链表中元素的类型：

```
class List<T> {

}
```

其次，为了表示链表中的节点，我们需要给 List 添加一个内嵌类 Node：

```
class List<T> {
    class Node<T> {
        var value: T
        var next: Node<T>?

        init(value: T, next: Node<T>?) {
            self.value = value
            self.next = next
        }
    }
}
```

但这里，就有一个问题了，在 Node<T> 中使用的 T 和 List<T> 中的T是同一个类型么？其实，我们的本意是，它们应该是同一个类型。为了去掉这种歧义，在Swift 3.1里，我们可以把代码改成这样：

```
class List<T> {
    class Node: CustomStringConvertible {
        var value: T
        var next: Node?

        init(value: T, next: Node?) {
            self.value = value
            self.next = next
        }

        var description: String {
            var nextText = "End"

            if let next = self.next {
                nextText = String(describing: next.value)
            }

            return "[value: \(self.value) next: \(nextText)]"
        }
    }
}
```

这就是我们刚才提到的内嵌类型的第一个特性，尽管 Node 是一个普通类型，但它可以直接使用 List<T> 中的泛型参数，此时 Node.value 的类型就是 List 中元素的类型， next 的类型，就是 Node<T> 。并且，为了方便稍后看到结果，我们还给 Node 实现了 CustomStringConvertible 。

接下来，我们再来看一个内嵌类型需要自己独立泛型参数的情况。假设，我们希望用一个独立的类型抽象 List<T> 的底层存储：

```
class List<T> {
    // ...
    class Storage<U> {
        var head: U? = nil
        var curr: U? = nil
    }
}
```

这就是我们要介绍的内嵌类型的第二个改进，内嵌类型可以和其外围类型有不同的泛型参数。这里我们刻意使用了 U 来表示 List 使用的存储单元的类型。实际上，这里，即便我们使用 Storage<T> ，在 Storage 的定义内部， T 也是一个全新的类型，并不是 List 中 T 的类型，为了避免歧义，我们最好还是用一个全新的字母，避免给自己带来不必要的麻烦。当然，这里我们仅仅是为了示意，并没有实际编写一个真正的buffer，只是定义了两个位置， head 和 curr ，分别表示链表的头和当前位置。

定义好这些类型之后，我们就可以实现链表了。首先，给 List<T> 添加两个属性，表示存储和元素个数：

```
class List<T> {
    // ...
    var storage: Storage<Node> = Storage()
    var count = 0
}
```

然后，再添加一个用于演示的 push 方法：

```
class List<T> {
    // ...
    func push(element: T) {
        let node = Node(value: element, next: nil)

        if storage.head == nil {
            storage.head = node
        }

        if storage.curr != nil {
            storage.curr?.next = node
        }

        storage.curr = node
        count += 1
    }
}
```

第三，我们让 List 也实现 CustomStringConvertible：

```
extension List: CustomStringConvertible {
    var description: String {
        var desc = ""
        var pos = storage.head

        while pos != nil {
            desc += (pos!.description + "\n")

            pos = pos!.next
        }

        return desc
    }
}
```

这里要说明一点的是，当我们为一个泛型类型添加 extension 的时候，是无须使用泛型参数的。

最后，我们就可以用下面的代码来试一下了：

```
var l: List<Int> = List<Int>()

l.push(element: 2)
l.push(element: 4)
l.push(element: 6)
l.push(element: 8)

print(l)
// [value: 2 next: 4]
// [value: 4 next: 6]
// [value: 6 next: 8]
// [value: 8 next: End]
```

◀ SR-1009 使用具象类型约束泛型参数 (https://www.boxueio.com/series/what-is-new-in-swift-31/ebook/211)

返回视频 ▶ (/series/what-is-new-in-swift-31)

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/)