Swift 3 的第一印象

为什么String不是一个Collection类型?

(https://www.boxueio.com/series/swift-up-and-running/ebook/121)

使用Tuple打包数据

(https://www.boxueio.com/series/swift-up-and-running/ebook/4)

基于unicode的字符串常用操作

Back to series (/series/swift-up-and-running)

字号
字号
默认主题
金色主题
暗色主题

获取前缀

好吧，经历了上一节的各种“说教”，也许现在你真的已经做好准备要用全新的姿态面对Swift中的字符串处理了。然而，当你想获取一个字符串中的前缀时，根据我们前面讲到的内容，你写下了下面的代码：

```
let beg = cafee.startIndex
let end = cafee.index(cafee.startIndex, offsetBy: 3)
cafee[beg ..< end] // Caf
```

看到这，估计你还是会觉得很受挫折，毕竟这难道不是 `cafee[0 ..< 3]` 就能解决的问题么？为了避免上面这样的代码，Swift提供了一个叫做 `prefix(_:)` 的方法，它返回一个特定的 `CharacterView`：

```
String(cafee.characters.prefix(3)) // Caf
```

当我们用这个view创建 `String` 对象的时候，就可以看到结果和之前是一样的。但这样用起来，就方便多了。

遍历字符串中的每一个字符

有时候，我们需要遍历字符串中的每一个字符，通过 `characters` 属性，我们可以很容易办到：

```
var mixStr = "Swift很有趣"

for (index, value) in mixStr.characters.enumerated() {
    print("\(index): \(value)")
}

// 0: S
// 1: w
// 2: i
// 3: f
// 4: t
// 5: 很
// 6: 有
// 7: 趣
```

上面的代码，会返回注释中的结果。其中 `enumerated` 方法会返回一个Tuple序列，其中的每一个元素表示对应字符在字符串中位置以及值。从结果就可以看到，Swift可以很好的识别字符串中的中英文字符，让unicode字符串处理起来和普通的字符数组一样。

插入内容

当我们要在字符串中插入内容时，要先获得目标位置的 `Index` 对象，然后使用 `insert` 方法。例如：

```
if let index = mixStr.characters.index(of: "很") {
    mixStr.insert(contentsOf: " 3.0".characters, at: index)
    // "Swift 3.0很有趣"
}
```

这样，我们就在原来“很”字的位置，插入了“ 3.0”。

基于Range的查找和替换

借助于由 `String.Index` 构成的 `Range` 对象，我们可以很方便的替换某个范围的字符串。例如，要把 `mixStr` 中的“很有趣”，替换成“is interesting”，可以这样：

```
if let cnIndex = mixStr.characters.index(of: "很") {
    // 2. Replace a specific range
    mixStr.replaceSubrange(
        cnIndex ..< mixStr.endIndex,
        with: " is interesting!")
    // Swift 3.0 is interesting!
}
```

首先，我们使用 `cnIndex ..< mixStr.endIndex` 得到了要替换的字符串的 `Range` 对象。然后，直接调用 `replaceSubrange(_:with:)`，用 `with` 中的值替换 `Range` 中的值就可以了；

字符串切片

在上节中，我们提到过 `String.characters` 是有集合类型的行为的，因此，最简单的获取子串的方式就是截取 `String.characters`，得到一个新的 `CharacterView`，并用这个view，生成新的 `String` 对象。例如，我们要截取 `mixStr` 中的“interesting”部分，可以这样：

```
let swiftView = mixStr.characters.suffix(12).dropLast()
String(swiftView) // interesting
```

这样，我们先用 `suffix` 截掉了头部的"Swift 3.0 is"，再用 `dropLast` 方法去掉了末尾的“!”，但别忘了，此时，我们针对 `mixStr.characters` 的操作，得到的是一个 `String.CharacterView` 对象，我们需要用这个view，生成一个新的 `String` 。

另外一类和子串相关的操作就是按照一定规则把之前的字符分割成多个部分，先来看个例子：

```
let strViews = mixStr.characters.split(separator: " ")
strViews.map(String.init)
// ["Swift", "3.0", "is", "interesting!"]
```

其中，我们可以把 `strViews` 理解为一个 `String.CharacterView` 集合。然后用 `map` 方法把每集合中的每一个view都生成一个新的 `String` 对象，最后，就得到了一个包含每一个子串的数组 `Array<String>` 。

除了用特定字符来分割子串，我们还可以用一个closure来指定分割条件。例如，把 `mixStr` 中的每个奇数位置的字符当成分隔符：

```
var i = 0
let singleCharViews = mixStr.characters.split { _ in
    if i > 0 {
        i = 0
        return true
    }
    else {
        i += 1
        return false
    }
}

singleCharViews.map(String.init)
// ["S", "i", "t", "3", "0", "i", " ", "n", "e", "e", "t", "n", "!"]
```

What's next?

在这节内容里，我们了解了unicode环境中，一些常用的字符串操作。基本上，就是先找到要操作内容的 `String.CharacterView.Index` 对象，然后对 `String.characters` 这个集合进行处理。至此，我们对 `String` 类型的讨论就先告一个段落了。在下段视频中，我们将介绍一个可以方便打包不同类型数据的工具： `Tuple` 。

⏮ 为什么String不是一个Collection类型? 使用Tuple打包数据 ⏭

(https://www.boxueio.com/series/swift-up-and-running/ebook/121) (https://www.boxueio.com/series/swift-up-and-running/ebook/4)

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/)