What's new in Swift 4

如何自定义JSON的解码过程

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/297)

如何让model兼容多个版本的API

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/299)

如何编码和解码带有派生关系的model

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

这一节，我们来看当model存在继承关系的时候，是如何与JSON完成自动转换的，它的默认行为，和我们想象的有点儿不太一样。

假设，我们有下面这两个类：

```
class Point2D: Codable {
    var x = 0.0
    var y = 0.0
}

class Point3D: Point2D {
    var z = 0.0
}
```

由于这两个类包含的都是简单属性，我们希望直接让它们遵从 `Codable`，实现JSON到model的自动转化，但事情却并不如我们想象的这么简单。在接下来的内容里，为了方便观察对象编码的结果，我们定义了一个全局 `encode` 函数，在后面的视频中，我们也会使用它：

```
func encode<T>(of model: T) throws where T: Codable {

    let encoder = JSONEncoder()
    encoder.outputFormatting = .prettyPrinted

    let data = try encoder.encode(model)
    print(String(data: data, encoding: .utf8)!)
}
```

这基本就是我们之前手动编码对象的封装，很简单。唯一要注意的就是我们在最后，对泛型参数 `T` 进行了类型约束，要求它遵从 `Codable`。接下来，我们创建一个 `Point3D` 对象，然后对它编码：

```
var p1 = Point3D()
p1.x = 1
p1.y = 1
p1.z = 1

encode(of: p1)
```

这段代码会得到什么结果呢？你可能会想，当然是包含xyz的JSON啊。但执行一下，你会看到这样的结果：

```
{
  "x" : 1,
  "y" : 1
}
```

`z` 呢？如果你第一次执行这段代码，一定会感到奇怪。**默认 `Codable` 中的默认 `encode` 方法并不能正确处理派生类对象**。因此，当我们的model是派生类时，要自己编写对应的编码和解码的方法。

先来看编码，我们先实现基类的部分：

```
class Point2D: Codable {
    var x = 0.0
    var y = 0.0

    private enum CodingKeys: String, CodingKey {
        case x
        case y
    }

    func encode(to encoder: Encoder) throws {
        var container = encoder.container(keyedBy: CodingKeys.self)
        try container.encode(x, forKey: .x)
        try container.encode(y, forKey: .y)
    }
}
```

唯一需要注意的，就是基类中的 CodingKeys 被修饰成了 private 。因为派生类中，也要定义它自己的 CodingKeys ，我们要避免它被派生类继承。

然后是派生类的部分：

```
class Point3D: Point2D {
    var z = 0.0

    private enum CodingKeys: String, CodingKey {
        case z
    }

    override func encode(to encoder: Encoder) throws {
        var container =
            encoder.container(keyedBy: CodingKeys.self)
        try container.encode(z, forKey: .z)
    }
}
```

现在，你应该想：这下总该没问题了吧。但如果执行下，你就会发现这样的结果：

```
{
  "z" : 1
}
```

这时，也许你马上就意识到问题所在了，在派生类的 encode 方法里，先调用基类版本的 encode 就应该正确了：

```
override func encode(to encoder: Encoder) throws {
    try super.encode(to: encoder)
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(z, forKey: .z)
}
```

重新执行一下，就会得到正确的结果了：

```
{
  "x" : 1,
  "y" : 1,
  "z" : 1
}
```

在派生类的实现里，基类和派生类的属性共享了同一个 container 。但这样的方式并不利于我们了解其中哪部分属于基类，哪部分属于派生类。为了能在编码后的结果中方便区分，我们得让它们使用不同的容器：

```
override func encode(to encoder: Encoder) throws {
    var container =
        encoder.container(keyedBy: CodingKeys.self)
    try super.encode(to: container.superEncoder())

    try container.encode(z, forKey: .z)
}
```

这里，我们使用了 container.superEncoder() 获得了派生类中，专门用于编码基类的容器，并把它传递给了基类的编码方法。现在，重新执行一下，我们会看到下面这样的结果：

```
{
  "super" : {
    "x" : 1,
    "y" : 1
  },
  "z" : 1
}
```

其中，基类和派生类的部分就很清楚了。但我们还可以进一步改进这个结果，把派生类中的 `CodingKeys` 改成这样：

```
private enum CodingKeys: String, CodingKey {
    case z
    case point_2d
}
```

然后，在派生类的 encode 方法里，指定编码基类时的key：

```
override func encode(to encoder: Encoder) throws {
    var container =
        encoder.container(keyedBy: CodingKeys.self)
    try super.encode(to:
        container.superEncoder(forKey: .point_2d))

    try container.encode(z, forKey: .z)
}
```

重新执行下，结果就会这样：

```
{
  "z" : 1,
  "point_2d" : {
    "x" : 1,
    "y" : 1
  }
}
```

理解了编码的过程之后，解码的过程是类似的，无非就是把共享容器，或使用基类独立容器的代码写在 `init(from decoder: Decoder)` 方法里，大家可以试着自己写写，我们就不再重复了。

What's next?

了解了派生类的编码方式之后，下一节，我们来看一个经常会遇到的场景：如何通过 `UserInfo` ，在API版本更新的过程中，为保持兼容性提供辅助信息。

◀ 如何自定义JSON的解码过程
(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/297)

如何让model兼容多个版本的API ▶
(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/299)

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/)