What's new in Swift 4

使用Codable解析JSON (https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/294)

如何自定义model对象的编码过程 (https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/296)

如何处理常见的JSON嵌套结构

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

欢迎回来，我们继续之前的话题。很多时候，服务器返回的JSON都不是一个“扁平”的结构，而是包含了各种嵌套，在这一节，我们就来处理如何把各种嵌套的情况，对应到Swift model。

用对象封装数组

第一种情况，是用对象封装数组，例如，服务器返回了多个视频信息的JSON：

```
{
    "list": [
        {
            "title": "How to parse JSON in Swift 4 - I",
            "series": "What's new in Swift 4",
            "created_by": "Mars",
            "type": "free",
            "created_at": "2017-08-23T01:42:42Z",
            "duration": "NaN",
            "origin": "Ym94dWVpby5jb20=",
            "url": "boxueio.com"
        },
        {
            "title": "How to parse JSON in Swift 4 - II",
            "series": "What's new in Swift 4",
            "created_by": "Mars",
            "type": "free"
            "created_at": "2017-08-23T01:42:42Z",
            "duration": "NaN",
            "origin": "Ym94dWVpby5jb20=",
            "url": "boxueio.com"
        }
    ]
}
```

为了把类似这种情况的JSON直接转型成model，我们只要定义下面这样的 `struct` 就好了：

```
struct EpisodeList: Codable {
    let list: [Episode]
}
```

这里，由于 `EpisodeList` 和 `Episode` 都是遵从 `Codable` 的，因此我们可以直接用之前的方式对JSON解码：

```
let list = try! decoder.decode(EpisodeList.self, from: data)
dump(list)
```

只是这次，对应的Model类型，变成了 `EpisodeList` 。执行一下，就能在控制台看到下面这样的结果：

```
▿ sss.EpisodeList
  ▿ list: 2 elements
    ▿ sss.Episode
      - title: "How to parse JSON in Swift 4 - I"
      - series: "What\'s new in Swift 4"
      - createdBy: "Mars"
      ...
```

数组作为JSON根对象

第二种情况，服务器还可能直接返回一个数组，而不使用对象封装它：

```
[
    {
        "title": "How to parse JSON in Swift 4 - I",
        "series": "What's new in Swift 4",
        "created_by": "Mars",
        "type": "free",
        "created_at": "2017-08-23T01:42:42Z",
        "duration": "NaN",
        "origin": "Ym94dWVpby5jb20=",
        "url": "boxueio.com"
    },
    {
        "title": "How to parse JSON in Swift 4 - II",
        "series": "What's new in Swift 4",
        "created_by": "Mars",
        "type": "free",
        "created_at": "2017-08-23T01:42:42Z",
        "duration": "NaN",
        "origin": "Ym94dWVpby5jb20=",
        "url": "boxueio.com"
    }
]
```

对这种情况，我们无须声明任何新的类型，只要在解码的时候，指定一个数组类型就好了：

```
let list = try! decoder.decode([Episode].self, from: data)

dump(list)
```

这次，我们就会看到这样的结果：

```
▿ 2 elements
  ▿ sss.Episode
    - title: "How to parse JSON in Swift 4 - I"
    - series: "What\'s new in Swift 4"
    - createdBy: "Mars"
    - type: sss.EpisodeType.free
    ...
```

纯数组中的对象带有唯一Key

第三种情况，可以看成是前面两种情况的组合，假设数组中的对象，是通过一个Key索引的：

```
[
    {
        "episode": {
            "title": "How to parse JSON in Swift 4 - I",
            "series": "What's new in Swift 4",
            "created_by": "Mars",
            "type": "free",
            "created_at": "2017-08-23T01:42:42Z",
            "duration": "NaN",
            "origin": "Ym94dWVpby5jb20=",
            "url": "boxueio.com"
        }
    },
    ...
]
```

对于这种情况，数组内的结构，可以用 Dictionary<String: Episode> 表示，而整个JSON，则是这种 Dictionary 的数组，于是，在解码的时候，我们只要把这个类型传递给它就好了：

```
let list = try! decoder.decode(
    [Dictionary<String: Episode>].self, from: data)

dump(list)
```

执行下，就会看到这样的结果：

```
▿ 2 elements
  ▿ 1 key/value pair
    ▿ (2 elements)
      - key: "episode"
      ▿ value: sss.Episode
        - title: "How to parse JSON in Swift 4 - I"
        - series: "What\'s new in Swift 4"
        - createdBy: "Mars"
        ...
```

更一般的复杂情况

在这一节最后，我们结合之前说过的这些情况，看一个更一般的例子，假设我们要给视频播放的页面传递一个包含所有要显示内容的JSON，它看上去是这样的：

```
let response = """
{
    "meta": {
        "total_exp": 1000,
        "level": "beginner",
        "total_duration": 120
    },
    "list": [
        {
            "title": "How to parse JSON in Swift 4 - I",
            "series": "What's new in Swift 4",
            "created_by": "Mars",
            "type": "free",
            "created_at": "2017-08-23T01:42:42Z",
            "duration": "NaN",
            "origin": "Ym94dWVpby5jb20=",
            "url": "boxueio.com"
        },
        {
            "title": "How to parse JSON in Swift 4 - II",
            "series": "What's new in Swift 4",
            "created_by": "Mars",
            "type": "free",
            "created_at": "2017-08-23T01:42:42Z",
            "duration": "NaN",
            "origin": "Ym94dWVpby5jb20=",
            "url": "boxueio.com"
        }
    ]
}
"""
```

为了把这段JSON自动转型成Swift model，我们新建一个 `struct`：

```
struct EpisodeMeta: Codable {
    let total_exp: Int
    let level: EpisodeLevel
    let total_duration: Int

    enum EpisodeLevel: String, Codable {
        case beginner
        case intermediate
        case advanced
    }
}
```

它对应JSON头部视频信息的部分，这里，由于视频难度属于视频信息的一部分，我们把 `EpisodeLevel` 定义成了一个内嵌类型。

接下来，JSON的后半段，是系列中每一个视频的具体信息，这种情况我们已经处理过了，这里，我们把之前定义的类型整理一下：

```
struct Episode: Codable {
    let title: String
    let series: String
    let createdBy: String
    let type: EpisodeType
    let createdAt: Date
    let duration: Float
    let origin: Data
    let url: URL

    enum CodingKeys: String, CodingKey {
        case title
        case series
        case createdBy = "created_by"
        case type
        case createdAt = "created_at"
        case duration
        case origin
        case url
    }

    enum EpisodeType: String, Codable {
        case free
        case paid
    }
}
```

可以看到，我们把 EpisodeType 也变成了 Episode 的内嵌类型。最后，我们定义一个表示页面数据的 struct：

```
struct EpisodePage: Codable {
    let meta: EpisodeMeta
    let list: [Episode]

    struct EpisodeMeta: Codable {
        /// ...
    }

    struct Episode: Codable {
        /// ...
    }
```

这样， EpisodePage 就完全对应我们上面提到的JSON结构了，接下来，使用用 decode 方法解码就好：

```
let page = try! decoder.decode(
    EpisodePage.self, from: data)

dump(page)
```

执行一下，就能看到下面这样的结果了：

```
▿ sss.EpisodePage
  ▿ meta: sss.EpisodePage.EpisodeMeta
    - total_exp: 1000
    - level: sss.EpisodePage.EpisodeMeta.EpisodeLevel.beginner
    - total_duration: 120
  ▿ list: 2 elements
    ▿ sss.EpisodePage.Episode
      - title: "How to parse JSON in Swift 4 - I"
      - series: "What\'s new in Swift 4"
      - createdBy: "Mars"
```

What's next?

以上，就是这一节的内容，了解了各种常见的嵌套处理方式之后，下一节，我们来看如何通过 Codable 进一步定制编码和解码的行为。

(https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/294) (https://www.boxueio.com/series/what-is-new-in-swift-4/ebook/296)

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/)