

Swift中的异常和错误处理

[◀ NSError是如何桥接到Swift原生错误处理的?](#)[如何处理closure参数会发生的错误? ▶](#)<https://www.boxueio.com/series/error-handling/ebook/200><https://www.boxueio.com/series/error-handling/ebook/202>

Swift中的错误是如何映射到NSError的?

[⌕ Back to series \(/series/error-handling\)](#)

把Swift中的 `Error` 移植到Objective-C, 相对而言倒是个简单很多的事情。Swift会根据 `enum` 的名字自动生成默认的 `error domain`, 并从0开始, 为每一个 `enum` 中的 `case` 设置 `error code`。

为了看到映射的结果, 首先, 我们得把上一节中的 `struct Car` 改成一个 `NSObject` 的派生类, 并给它添加一个 `memberwise init` 方法:

```
class Car: NSObject {
    var fuelInLitre: Double

    init(fuelInLitre: Double) {
        self.fuelInLitre = fuelInLitre
    }

    // ...
}
```

然后, 在 `Sensor.m` 中, 我们先包含Swift类在Objective-C中的头文件:

```
#import "SwiftErrorsInOC-Swift.h"
```

就可以在Objective-C中使用 `class Car` 了。然后, 我们定义一个全局函数 `startACar()`:

```
// In Sensor.m
NSObject* startACar() {
    Car *car = [[Car alloc] initWithFuel:5];

    NSError *err = nil;
    [car startAndReturnError: &err];

    if (err != nil) {
        NSLog(@"Error code: %ld", (long)err.code);
        NSLog(@"Error domain: %@", err.domain);

        return nil;
    }

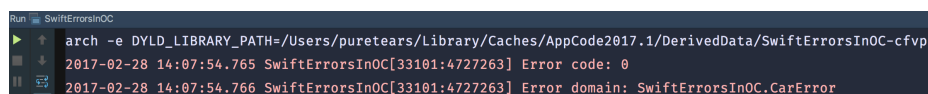
    return car;
}
```

在上面的代码里可以看到, 由于Swift中的 `Car.start()` 是一个 `throws` 方法, 在OC里, 它会被添加一个 `AndReturnError` 后缀, 并接受一个 `NSError **` 类型的参数。然后, 当 `err` 不为 `nil` 时, 我们向控制台打印了 `start` 抛出的错误映射到OC的结果。

由于 `car` 对象的 `fuel` 只有5, 所以这个调用是一定会产生 `NSError` 的。为了在Swift中调用这个方法, 我们在 `Sensor.h` 中添加下面的声明:

```
// In Sensor.h
NSObject* startACar();
```

然后, 在 `main.swift` 里, 我们直接调用 `startACar`, 就能在控制台看到类似这样的结果:



```
Run SwiftErrorsInOC
arch -e DYLD_LIBRARY_PATH=/Users/puretears/Library/Caches/AppCode2017.1/DerivedData/SwiftErrorsInOC-cfvp
2017-02-28 14:07:54.765 SwiftErrorsInOC[33101:4727263] Error code: 0
2017-02-28 14:07:54.766 SwiftErrorsInOC[33101:4727263] Error domain: SwiftErrorsInOC.CarError
```

在这里, 自动生成的 `NSError` 对象的 `code` 是0, `domain` 是“项目名.Swift中 `enum` 的名字”。当然, 这只是最基本的映射。在Swift 3里, 除了 `Error` 之外, 还添加了一些新的 `protocol`, 帮助我们进一步定制自动生成的 `NSError` 对象的属性。

LocalizedError

🔍 字号

🌑 字号

🖌️ 默认主题

🖌️ 金色主题

🖌️ 暗色主题

第一个要介绍的，是 `LocalizedError`，它的定义是这样的：

```
protocol LocalizedError : Error {
    /// A localized message describing what error occurred.
    var errorDescription: String? { get }

    /// A localized message describing the reason for the failure.
    var failureReason: String? { get }

    /// A localized message describing how one might recover from the failure.
    var recoverySuggestion: String? { get }

    /// A localized message providing "help" text if the user requests help.
    var helpAnchor: String? { get }
}
```

并且，Swift为 `LocalizedError` 中的每一个属性都提供了默认值 `nil`，因此，你可以只定义自己需要的部分就好了。例如，对于我们的 `CarError` 来说，可以把它改成这样：

```
enum CarError: LocalizedError {
    case outOfFuel
}
```

然后，通过 `extension` 给它添加额外信息：

```
extension CarError: LocalizedError {
    var recoverySuggestion: String? {
        return "Switch to e-power mode"
    }
}
```

这样，在OC的 `startACar` 实现里，我们就可以通过访问 `NSError` 的 `localizedRecoverySuggestion` 属性来读取恢复建议了：

```
NSObject* startACar() {
    // ...
    if (err != nil) {
        // ...
        NSLog(@"Recovery suggestion: %@",
              err.localizedDescription);
        return nil;
    }

    // ...
}
```

CustomNSError

另外一个加入到Swift的 `protocol` 是 `CustomNSError`，我们可以通过它自定义 `NSError` 中的 `code` / `domain` / `userInfo`。

```
extension CarError: CustomNSError {
    static let errorDomain = "CarErrorDomain"

    var errorCode: Int {
        switch self {
        case .outOfFuel:
            return -100
        }
    }

    var errorUserInfo: [String: Any] {
        switch self {
        case .outOfFuel:
            return [
                "LocalizedDescription":
                "U r running out of fuel"
            ]
        }
    }
}
```

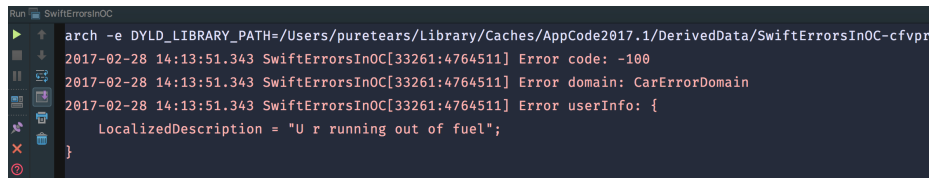
尽管在SE-0112 (<https://github.com/apple/swift-evolution/blob/master/proposals/0112-nerror-bridging.md>)的约定里, `errorDomain` 是一个computed property, 但至少在XCode 8.2.1中, 它只能定义成一个type property。不过想来也合理, 一个 `NSError` 对象只需要一个error code就可以了, 我们也没什么计算它的必要。

接下来, 把 `startACar` 的定义改成这样:

```
NSObject* startACar() {
    // ...
    if (err != nil) {
        NSLog(@"Error domain: %@", err.domain);
        NSLog(@"Error code: %ld", (long)err.code);
        NSLog(@"Error userInfo: %@", err.userInfo);
    }

    // ...
}
```

我们就能在控制台看到自定义的结果了:



```
Run SwiftErrorsInOC
arch -e DYLD_LIBRARY_PATH=/Users/puretears/Library/Caches/AppCode2017.1/DerivedData/SwiftErrorsInOC-cfvp...
2017-02-28 14:13:51.343 SwiftErrorsInOC[33261:4764511] Error code: -100
2017-02-28 14:13:51.343 SwiftErrorsInOC[33261:4764511] Error domain: CarErrorDomain
2017-02-28 14:13:51.343 SwiftErrorsInOC[33261:4764511] Error userInfo: {
    LocalizedDescription = "U r running out of fuel";
}
```

What's next?

了解了和 `NSError` 的交互之后, 下一节中, 我们来看另外一个常用的错误处理场景。如何正确处理函数类型的参数返回的错误呢?

d

◀ [NSError是如何桥接到Swift原生错误处理的?](#)

(<https://www.boxueio.com/series/error-handling/ebook/200>)

[如何处理closure参数会发生的错误?](#) ▶

(<https://www.boxueio.com/series/error-handling/ebook/202>)



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