**Propagating Errors**

**throws**

Từ khóa “throws” là cần thiết trong Swift nếu 1 function hay 1 method throws 1 lỗi. Các lỗi sẽ được ném ra 1 cách tuần tự, nhưng để bong bóng lỗi quá xa nguồn được coi là 1 hành động lỗi. Ý nghĩa của propagation trong khắp codebase làm tăng khả năng lỗi sẽ thoát khỏi việc xử lí thích hợp, vì vậy throws là 1 nhiệm vụ đảm bảo propagation được ghi lại trong code và hiển nhiên là dành cho coder

rethrows

Hầu hết các ví dụ bạn đã nhìn thấy cho đến nay sử dụng throws, vậy về rethrows?

Rethrows nói với trình biên dịch rằng chức năng này chỉ throw 1 lỗi khi function parameter throw 1 lỗi. 1 ví dụ nhanh có thể tìn thấy dưới đây, (không cần thêm vào playground):

func doSomethingMagical(magicalOperation: () throws -> MagicalResult) rethrows -> MagicalResult {

return try magicalOperation()

}

Tại đó, SomethingMagical(\_:) sẽ chỉ throw 1 lỗi nếu magicalOperation cung cấp 1 function throws. Nếu nó thành công sẽ chả về 1 MagicalResult instead

### Manipulating Error Handling Behavior

**defer**

Mặc dù auto-propagation sẽ phục vụ bạn tốt trong hầu hết trường hợp, có những tình hướng bạn sử dụng code của bạn để ném ra các lỗi.

Câu lệnh defer là 1 cơ chế cho phép hành động “dọn dẹp” được thực hiện bất cứ khi nào phạm vi hiện tại đã bị thoát, chẳng hạn là như khi 1 function hay 1 method đã được trả về. Nó rất hữu ích cho việc quản lí các nguồn tài nguyên cần được sắp xếp gọn gàng hay các hành động không thành công, do đó nó đặc biệt hữu ích trong việc xử lí lỗi

Để thấy cách hoạt động, thêm đọn code sau vòa Witch Structure:

func speak() {

defer {

print("\*cackles\*")

}

print("Hello my pretties.")

}

Thêm đoạn sau vào cuối playground :

func exampleThree() {

print("") *// Add an empty line in the debug area*

let witchThree = Witch(name: "Hermione", familiar: nil, hat: nil)

witchThree.speak()

}

exampleThree()

Trong debug console, bạ có thể thấy những lời nói nhảm (witch cackle) sau tất cả những thứ cô nói.

Điều thú vị là. Câu lệnh defer được thực thi theo thứ tự ngược lại tại nơi nó được viết

Thêm 1câu lệnh defer :

func speak() {

defer {

print("\*cackles\*")

}

defer {

print("\*screeches\*")

}

print("Hello my pretties.")

}

Câu lênh in ra có làm bạn bất ngờ? Ah, điều kì diệu của prefer

**Nhiều điều thú vị hơn nữa với Errors**

Việc đưa các câu lệnh vào bên trong Swift mang đến cho ngôn ngữ này sự phù hợp cới nhiều ngôn ngữ phổ biến khác và tách Swift từ NSError-based được tìm thấy trong Objective-C. Objective-C, với hầu hết các thành phần, trực tiếp dịch và phân tích tĩnh trong trình biên dịch là một điều tuyệt với bạn cần nắm bắt.

Mặc dù do-catch và các chức năng liên quan có ý nghĩa rất quan trọng trong các ngôn ngữ khác, nhưng trong swift nó được đối xử như các câu lệnh khác. Điều này đảm bảo nó hiệu quả và hiệu quả hơn

Nhưng chỉ vì bạn có thể tạo ra các tùy chỉnh lỗi và ném chúng xung quanh, không có nghĩa là bạn bạn nên làm điều đó. Bạn thực sự cần xây dựng hướng dẫn khi throw và catch lỗi cho mỗi dự án mà bạn đảm nhận. Tôi muốn đề nghị 1 số điều sau:

Đảm bảo các loại lỗi được đạt tên rõ ràng trên codebase của bạn

Sử dụng optionals tại nơi có 1 lỗi tồn tại

Sử dụng custom errors tại chỗ có nhiều hơn 1 lỗi tồn tại

Không cho phép 1 lỗi được truyền quá xa source

Tương lai của việc xử lí lỗi trong swift

Một vài ý tưởng cho việc xử lí lỗi đang được bàn luận trên các diễn đàn về Swift. Mọt trong những khái niệm được bàn nhiều nhất là untyped propagation.

“

Chúng tôi tin rằng chúng ta có thể mở rộng mô hình hiện tại để hỗ trợ untyped propagation cho các universal errors. Làm tốt điều này, và đặc biệt là làm việc đó à không hoàn toàn hi sinh kích thức code và chất lượng, sẽ thấy tầm quan trọng của việc lập kế hoạch và sự hiểu biết sâu sắc. Vì lí do này nó được coi là nằm ngoài phạm vi của Swift 2.0

”

Cho dù bạn thích ý tưởng sẽ có sự thay đổi lớn về việc xử lí lỗi trong Swift 3, hoặc hạnh phúc với những gì hiện tại, nó là 1 điều tốt đẹp nếu biết cách rõ ràng việc xử lí lỗi đang được bản luận và cải thiện ngôn ngữ phát triển

Đi đâu từ đây?

Bạn có thể download toàn bộ playground ở đây cho bài hướng dẫn này

Để đọc thêm, tôi khuyên bạn nên đọc các bài sau, 1 số trong đó đã được tham chiếu trong suốt bài hướng dẫn này

* [Swift Apprentice, Chapter 21 – Error Handling](https://www.raywenderlich.com/store/swift-apprentice)
* [Failable Initializers](http://bit.ly/1RMpOY3)
* [Factory Method Pattern](https://en.wikipedia.org/wiki/Factory_method_pattern)
* [Pyramid of Doom](https://en.wikipedia.org/wiki/Pyramid_of_doom_(programming))

Nếu bạn quan tâm đến những cái có thể là bịa đặt trong Swift 3, tôi khuyên nên đọc những dề nghị mở, đọc Swift Language Proposals để biết thêm chi tiêt. Nếu bạn quan tâm , tại sao không submit của riêng bạn

Hi vọng bây giờ bạn đã bị mê hoặc bởi việc xử lí lỗi trong Swift. Nếu bạn có bất kì câu hỏi nào hoặc ý kiến về hướng dẫn, hãy tham gia thảo luận ở dưới