Sử dụng Generics trong Swift

Nguyễn Văn Tú
36 bài viết

30 tháng 09, 2016 - 846 lượt xem

Lập trình

Lập trình Generic là cách để viết các func mà không quan tâm đến kiểu dữ liệu. Mảng là một ví dụ như thế, bạn có thể thêm phần tử có kiểu Int hoặc AnyObject vào mảng mà chưa cần biết kiểu của nó trước khi thêm vào.

1.Generic Functions

Đầu tiên tạo một playground: File/New/Playground...., bạn đặt tên Generics và chọn iOS và chọn Next/Create.

Bắt đầu với cách tạo generic function. Mục đích của chúng ta là kiểm tra xem 2 đối tượng có giống kiểu hay không. Nếu chúng có kiểu giống nhau thì kiểm tra xem giá trị có giống nhau hay không.

```
func sameType(one: Int, two: Int) -> Void {
    if(type(of: one) == type(of: two)) {

        print("Same")

    }
    else {
        print("not same type")
    }
}
```

Đối với func vừa viết thì nó chỉ kiểm tra giá trị kiểu Int. Nếu muốn kiểm tra kiểu String thì lại viết một hàm tương tự nhưng khác kiểu của tham số sao. Nhưng không dến dây chúng ta sẽ sử dụng generic type thì nó sẽ tổng quát hơn nhiều.

```
func sameType<T, E>(one: T, two: E) -> Void {
    if(type(of: one) == type(of: two)) {
        print("Same")

    }
    else {
        print("not same type")
    }

}
```

Ở đây cú pháp của nó chỉ ra kiểu để sử dụng là T và E được để trong <T, E>. Sau khi khai báo T, E thì có thể sử dụng trong func.

2. Generic Classes và Structures

Một queue là một dữ liệu có cấu trúc giống như list và stack, nhưng bạn có thể thêm vào ở đuôi và lấy ra ở đầu, nó cũng giống như NSOperationQueue và GCD(dispatch_async).

Khởi tạo một struct như sau:

```
struct Queue<Element> {
}
```

Queue là một generic với kiểu là Element. Thì với queue nó có thể enqueue(thêm dữ liệu vào) và dequeue(lấy dữ liệu ra).Thêm các đoạn mã sau để có thể enqueue và dequeue:

```
    private var elements = [Element]()
    init(array: [Element]) {
        self.elements = array
    }
    mutating func enqueue(newElement: Element) {
        elements.append(newElement)
    }

    mutating func dequeue() -> Element? {
        guard !elements.isEmpty else {
            return nil
        }
        return elements.remove(at: 0)
    }
```

Bản chất queue cũng là một mảng nên sẽ cần khai báo 1 mảng có kiểu dữ liệu "element" và 2 funcs là enqueue và dequeue.

Điều cần nhớ là kiểu Element có thể được sử dụng bất kỳ đâu khi ở trong thân struct.

Thêm đoạn mã sau để xem các câu lệnh ở trên có chạy đúng không

```
var q = Queue(array: [1, "tu", "trinhminhcuong", 3])

q.enqueue(newElement: 4)
q.enqueue(newElement: 2)

q.dequeue()
q.dequeue()
q.dequeue()
q.dequeue()
```

3. Protocols và Constraints

Trong nhiều trường hợp chúng ta muốn thao tác so sánh giữa các đối tượng ví dụ như tìm index của một phần tử trong mảng.

Như sau:

```
    func findIndex<T>(of valueToFind: T) -> Int? {

        for (index, value) in self.elements.enumerated() {

            if let elementValue = value as? T

            {
                if valueToFind == elementValue
                {
                    return index
                }
            }
        }
        return nil
    }
```

Ở đây trong bài này sử dụng optional binding để kiểm tra 2 biến cùng kiểu. Ở đây trình biên dịch sẽ báo lỗi "Binary operator == cannot be applied to two T operands". Vì T không hỗ trợ toán tử '=='. Cũng may là Swift cung cấp protocol 'Equatable'.

```
    func findIndex<T: Equatable>(of valueToFind: T) -> Int? {
        for (index, value) in self.elements.enumerated() {
            if let elementValue = value as? T
            {
                if valueToFind == elementValue
                {
                    return index
                }
            }
        }
        return nil
    }
```

Ngoài protocol 'Equatable' thì swift còn hỗ trợ protocol khác như 'Hashable và Comparable' để sử dụng toán tử '<' và '>'.

4. Associated Types

Khi bạn định nghĩa một protocol, thì đôi khi nên sử dụng associated types như là một phần của protocol để để kiến soát cùng như thao tác hơn. Khi khai báo chỉ cần từ khoá associatedtype "tên", hơn nữa nó cung cấp luôn các func để so sánh "==, > ,<...."

```
protocol Container {
    associatedtype ItemType
    mutating func append(_ item: ItemType)
    var count: Int { get }
    subscript(i: Int) -> ItemType { get }
}
```

Container protol định nghĩa với các chức năng:

Thêm phần tử bằng func append(_:)

Lấy ra số lượng phần tử

Lấy ra phần tử theo chỉ số index và trả về ItemType

5. Where Clause

Swift cũng cho phép sử dụng where clauses với generic. Generic where clause cho phép xác định một associated type phải tuân thủ theo một protocol hoặc điều kiện nào đó. Một generic where clause bắt đầu với từ khoá where tiếp theo là các ràng buộc cho associated type. Generic clause được viết ở bên phải dấng sau kiểu trả về của func. Ví dụ dưới đây là func allItemsMatch, nó kiểm tra xem 2 đối tượng truyền vào có các phần tử giống nhau hay không.

```
func allItemsMatch<C1: Container, C2: Container>
    (_ someContainer: C1, _ anotherContainer: C2) -> Bool

    where C1.ItemType == C2.ItemType, C1.ItemType: Equatable {
    // Check that both containers contain the same number of items.
        if someContainer.count != anotherContainer.count {
            return false
        }
    // Check each pair of items to see if they are equivalent.
        for i in 0..<someContainer.count {
            if someContainer[i] != anotherContainer[i] {
                return false
            }
        }
        // All items match, so return true.
        return true
}

extension Array: Container{}
var a1 = Array(arrayLiteral: 4)
a1.append(1)
var a2 = Array(arrayLiteral: 4)
a2.append(1)
let a = allItemsMatch(a1, a2)
```

6.Tóm lại

Bài này đã tôi đã giới thiệu qua cơ bản và tác dụng của generics về các khai báo sử dụng ở functions, class, struct,.

Nếu bạn có thể hiểu rõ và vận dụng tốt thì có thể xử lý những vấn đề phực tạp tốt hơn.

> *Khóa học lập trình di động tại Techmaster:*
>
> - *Lập trình iOS căn bản đến nâng cao.*
> - *Thành thạo React Native qua 3 ví dụ.*
>
> *Để cài đặt MacOSX lên phần cứng không phải Apple liên hệ chuyên gia cài Hackintosh:*
>
> - *Nguyễn Minh Sơn: 0128706565?34*
> - *Huỳnh Minh Sơn: 0936225565*
> - *Website: caidatmacos.com*

Tham gia ngay khoá học lập trình iOS, hình thức học tập rất linh hoạt cho bạn lựa chọn và sẽ có nhức học phí khác nhau tuỳ theo bạn chọn học Online, Offline hoặc Flip/Learning(Kết hợp giữa Online và Offline). Ngoài ra bạn có thể tham gia thực tập toàn thời gian tại Techmaster để rút ngắn thời gian học và tăng cơ hội việc làm.

BÀI VIẾT LIÊN QUAN

Replica redis

Học trực tuyến trong thời đại COVID-19: Ưu và nhược điểm là gì?

Sử dụng Javascript Form Validation

Tất cả những gì bạn cần biết về DevOps

Tạo RESTFul với NODEJS: Tổng quan

KHOÁ HỌC HAY

PYTHON cơ bản cho nghề PHÂN TÍCH DỮ LIỆU

Web cơ bản HTML5, CSS3 và Javascript Online

Lập trình di động Flutter cho IOS - Android

Python phân tích xử lý dữ liệu

Nhập môn Machine Learning

Thiết kế - Lập trình cơ sở dữ liệu

Javascript căn bản - Tổng hợp 12 game huyền thoại

Web for Kids - Lập trình Web cần bản cho trẻ em

Git căn bản đến nâng cao

Lập trình Golang căn bản

Bình luận

** Vui lòng đăng nhập trước khi bình luận.*