**Các kiểu truyền dữ liệu giữa các viewcontroller**

Truyền dữ liệu giữa các UIViewController trong Swift là một nhiệm vụ phổ biến khi xây dựng ứng dụng iOS. Có nhiều cách khác nhau để thực hiện điều này tùy thuộc vào kiến trúc và yêu cầu cụ thể của ứng dụng. Dưới đây là một số phương pháp phổ biến:

1. ***Sử dụng*** ***Segue***

- **Segue** là một đối tượng trong **Storyboard** được sử dụng để xác định sự chuyển tiếp giữa hai ViewController. Khi bạn kích hoạt một **Segue**, nó sẽ tạo và chuyển sang **ViewController** đích. Bạn có thể sử dụng **Segue** để chuyển dữ liệu giữa các **ViewController** bằng cách chuẩn bị dữ liệu trước khi **Segue** được thực hiện.

**Các bước thực hiện:**

**a. Thiết lập Segue trong Storyboard**

1. **Kéo Segue vào Storyboard:**
   * Trong Storyboard, kéo một Button hoặc một đối tượng giao diện mà bạn muốn sử dụng để kích hoạt Segue.
   * Kết nối đối tượng này với ViewController đích bằng cách Ctrl-drag từ đối tượng đến ViewController đích và chọn loại Segue (ví dụ: Show, Present Modally, Push, v.v.).
2. **Đặt Identifier cho Segue:**
   * Chọn Segue trong Storyboard.
   * Trong bảng thuộc tính (Attributes Inspector), đặt một Identifier cho Segue (ví dụ: "showDetailSegue").

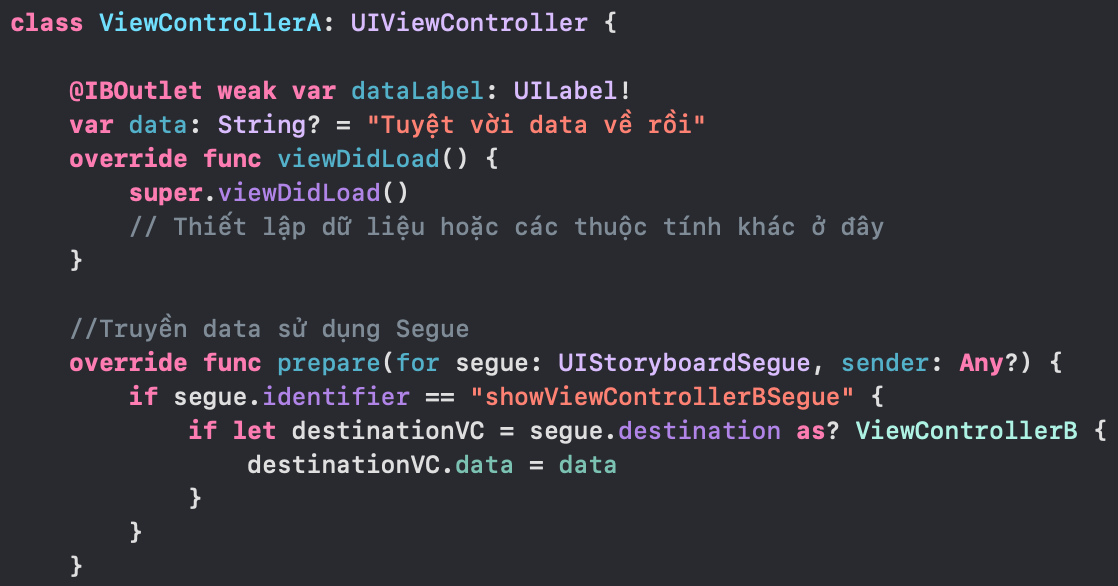
**b. Chuẩn bị dữ liệu trong ViewController gốc**

Trước khi Segue được thực hiện, bạn cần chuẩn bị dữ liệu và chuyển nó sang ViewController đích. Điều này được thực hiện trong phương thức prepare(for:sender:) của ViewController gốc.

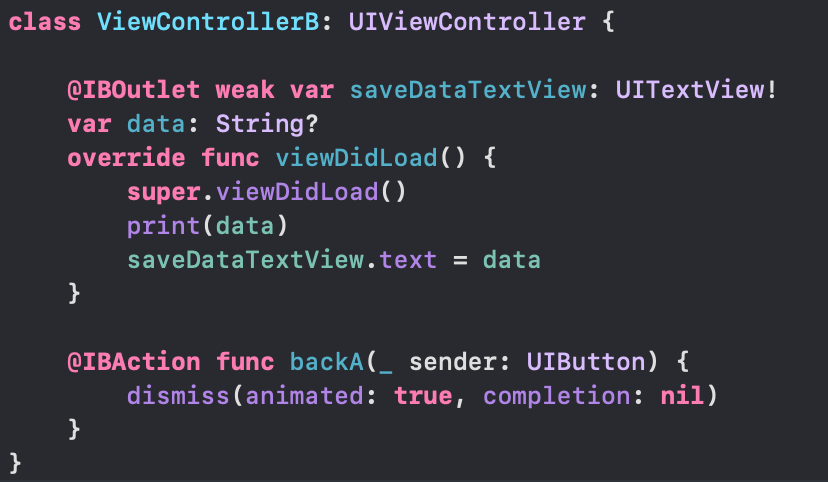
**Ví dụ:**

Giả sử bạn có một ViewController gốc (**ViewControllerA**) và một ViewController đích (**ViewControllerB**). Bạn muốn chuyển một chuỗi từ **ViewControllerA** sang **ViewControllerB**.

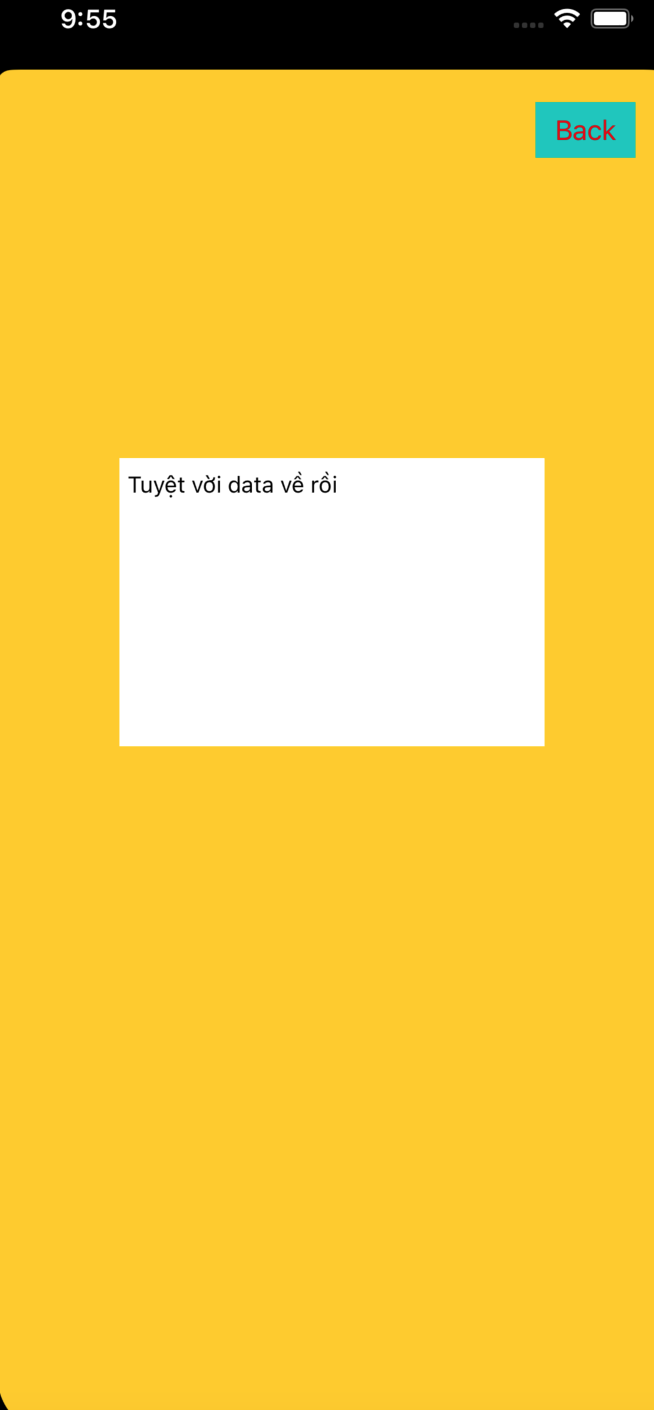
* **ViewControllerA.swift**

****

* **ViewControllerB.swift**



Kết quả:

Truyền dữ liệu qua TextView qua viewController B

### Ưu điểm của sử dụng Segue

* **Tích hợp sâu trong Storyboard**: Segue cho phép bạn thiết kế và quản lý chuyển tiếp giữa các ViewController trực tiếp trong Storyboard, giúp cho việc phát triển giao diện người dùng trở nên trực quan và dễ dàng hơn.
* **Đơn giản hóa mã nguồn**: Segue tự động quản lý việc tạo và chuyển sang ViewController đích, giúp giảm bớt mã nguồn cần thiết trong ViewController gốc.
* **Dễ dàng quản lý và duy trì**: Với Segue, việc theo dõi và quản lý các chuyển tiếp giữa ViewController trở nên dễ dàng hơn, đặc biệt khi làm việc với các dự án lớn.

1. ***Truyền dữ liệu trực tiếp***

Chuyển dữ liệu trực tiếp giữa các UIViewController trong iOS là một kỹ thuật quan trọng khi bạn muốn gửi thông tin từ một màn hình này sang màn hình khác. Trong trường hợp này, chúng ta sẽ xem xét cách chuyển dữ liệu từ ViewControllerA sang ViewControllerC mà không sử dụng Segue.

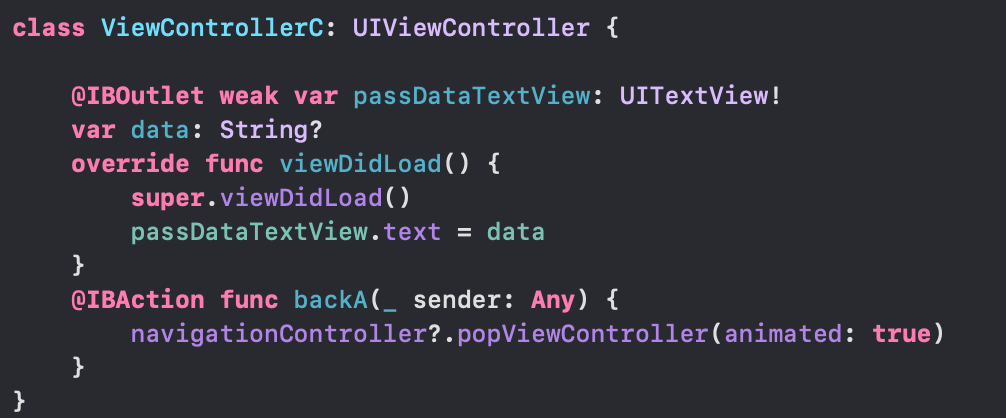
### Các bước thực hiện

1. **Khởi tạo ViewControllerC từ ViewControllerA:**
   * Bạn cần khởi tạo ViewControllerC từ ViewControllerA bằng cách tạo một instance của nó và đặt các thuộc tính cần thiết.
2. **Truyền dữ liệu:**
   * Bạn có thể truyền dữ liệu trực tiếp bằng cách thiết lập các thuộc tính của ViewControllerC.
3. **Hiển thị ViewControllerC:**
   * Bạn có thể sử dụng phương thức present(\_:animated:completion:) để trình bày ViewControllerC hoặc navigationController?.pushViewController(\_:animated:) nếu bạn sử dụng Navigation Controller.

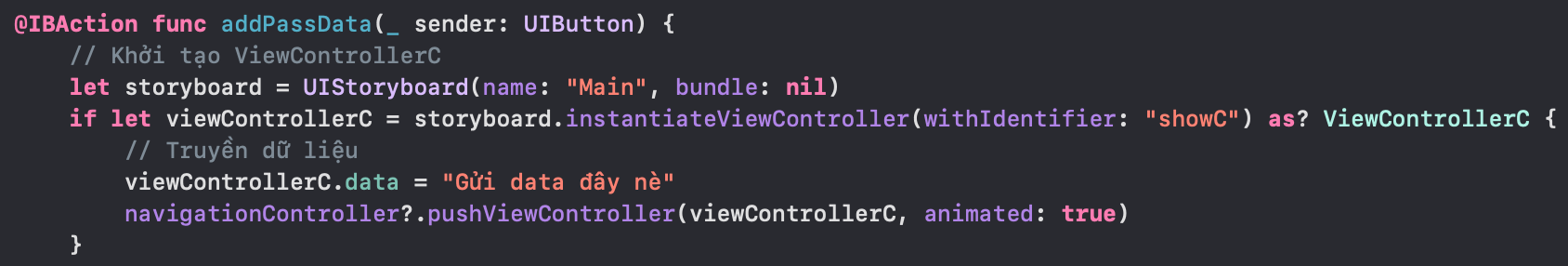
### Ví dụ cụ thể

Giả sử chúng ta có hai view controllers, ViewControllerA và ViewControllerC. Chúng ta sẽ truyền một chuỗi từ ViewControllerA sang ViewControllerC.

ViewControlleC.swift: ViewController nhận dữ liệu và đẩy vài textView



ViewControlleA.swift: Truyền dữ liệu của data qua viewController đích



Kết quả giữa hai màn hình như sau:

Việc truyền dữ liệu trực tiếp giữa các ViewController trong iOS rất linh hoạt và có thể được thực hiện dễ dàng bằng cách khởi tạo ViewController đích, thiết lập các thuộc tính của nó, và sau đó hiển thị nó. Điều này giúp bạn quản lý dữ liệu một cách rõ ràng và hiệu quả khi chuyển đổi giữa các màn hình trong ứng dụng của bạn.

1. ***Sử dụng Delegate***

Chuyển dữ liệu giữa các ViewController bằng cách sử dụng Delegate là một phương pháp phổ biến trong phát triển ứng dụng iOS. Nó giúp tạo ra một giao tiếp lỏng lẻo giữa hai ViewController, đảm bảo tính mô-đun và dễ bảo trì. Để minh họa cách sử dụng Delegate để chuyển dữ liệu giữa **ViewControllerA** và **ViewControllerDelegate**, chúng ta sẽ làm theo các bước sau:

### Các bước thực hiện

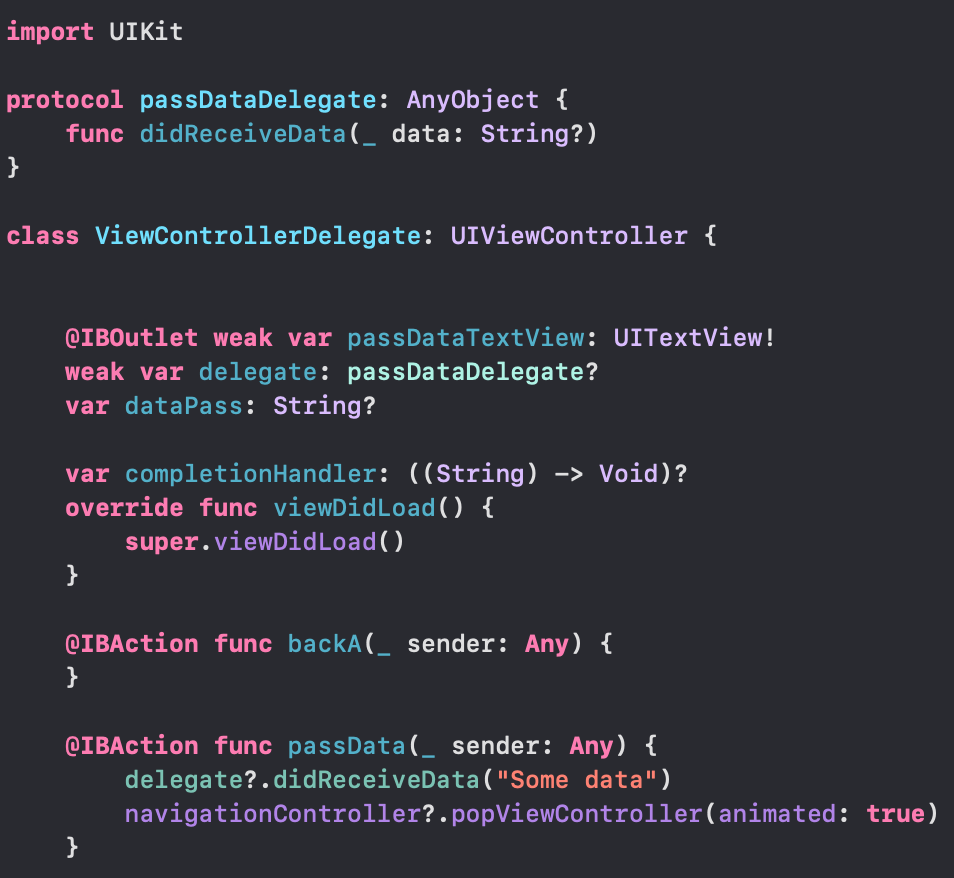
* **Định nghĩa giao thức Delegate:**
  1. Tạo một giao thức để định nghĩa các phương thức mà ViewControllerA sẽ triển khai để nhận dữ liệu từ **ViewControllerDelegate**.
* **Khai báo thuộc tính Delegate trong ViewControllerDelegate:**
  1. Thêm một thuộc tính delegate trong **ViewControllerDelegate** để lưu tham chiếu đến ViewControllerA.
* **Triển khai giao thức trong ViewControllerA:**
  1. Đảm bảo **ViewControllerA** tuân thủ giao thức và thực hiện các phương thức được định nghĩa trong giao thức đó.
* **Thiết lập Delegate:**
  1. Khi khởi tạo **ViewControllerDelegate** từ ViewControllerA, thiết lập ViewControllerA làm delegate của ViewControllerDelegate.
* **Gọi phương thức Delegate để truyền dữ liệu:**
  1. Khi dữ liệu cần được truyền, **ViewControllerDelegate** sẽ gọi phương thức delegate để gửi dữ liệu tới **ViewControllerA**.

### Ví dụ cụ thể

Giả sử chúng ta có **ViewControllerA** và **ViewControllerDelegate**. Chúng ta sẽ truyền dữ liệu từ **ViewControllerDelegate** trở về **ViewControllerA**.

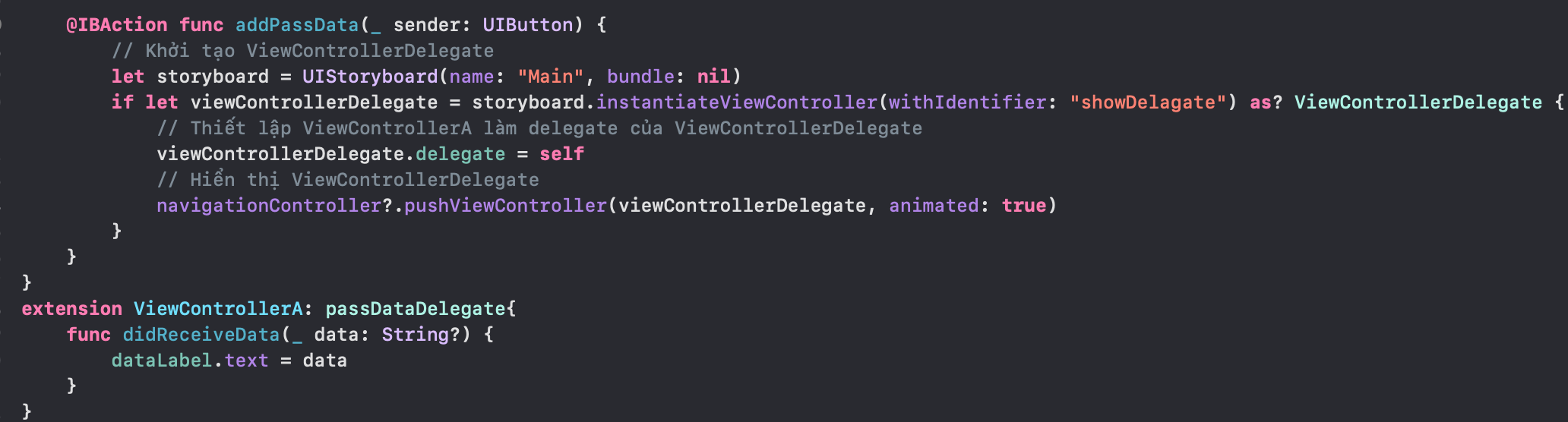
* **Định nghĩa giao thức Delegate:**

**ViewControllerDelegate.swift:**



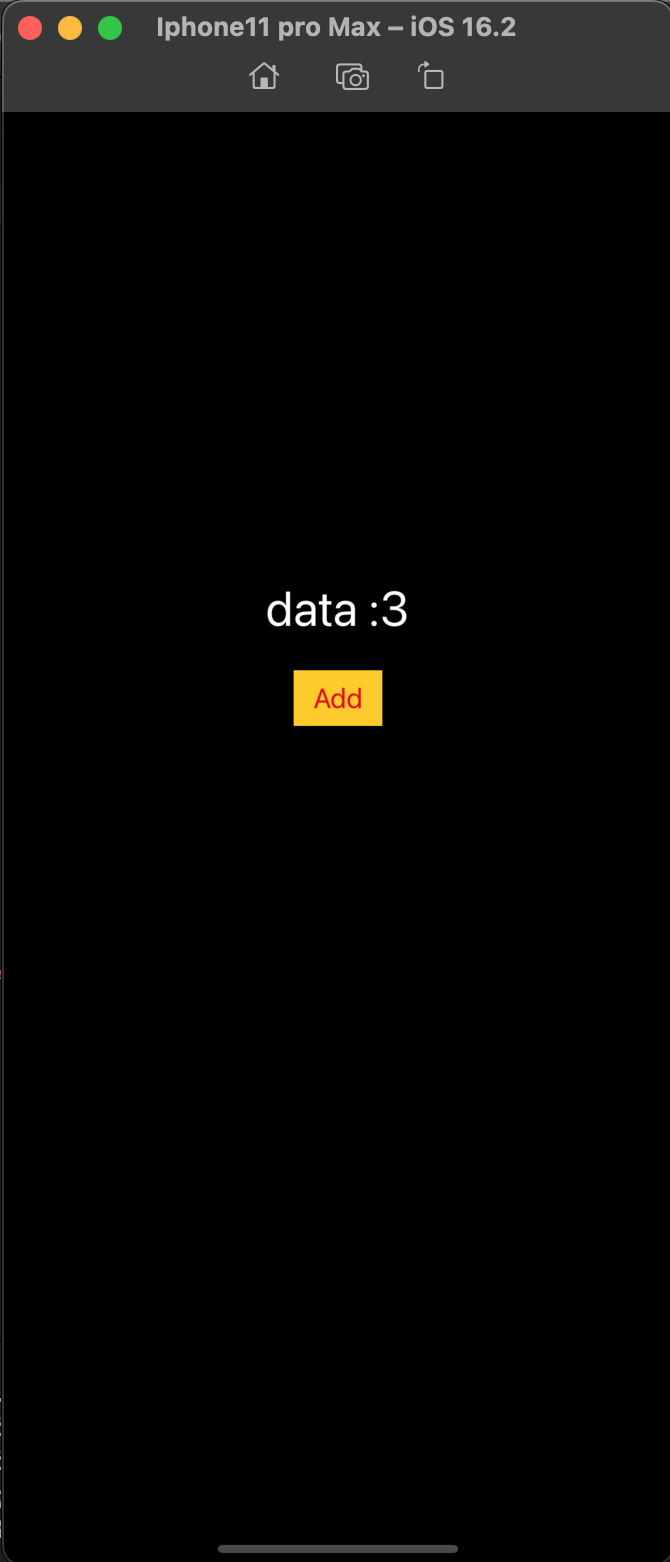
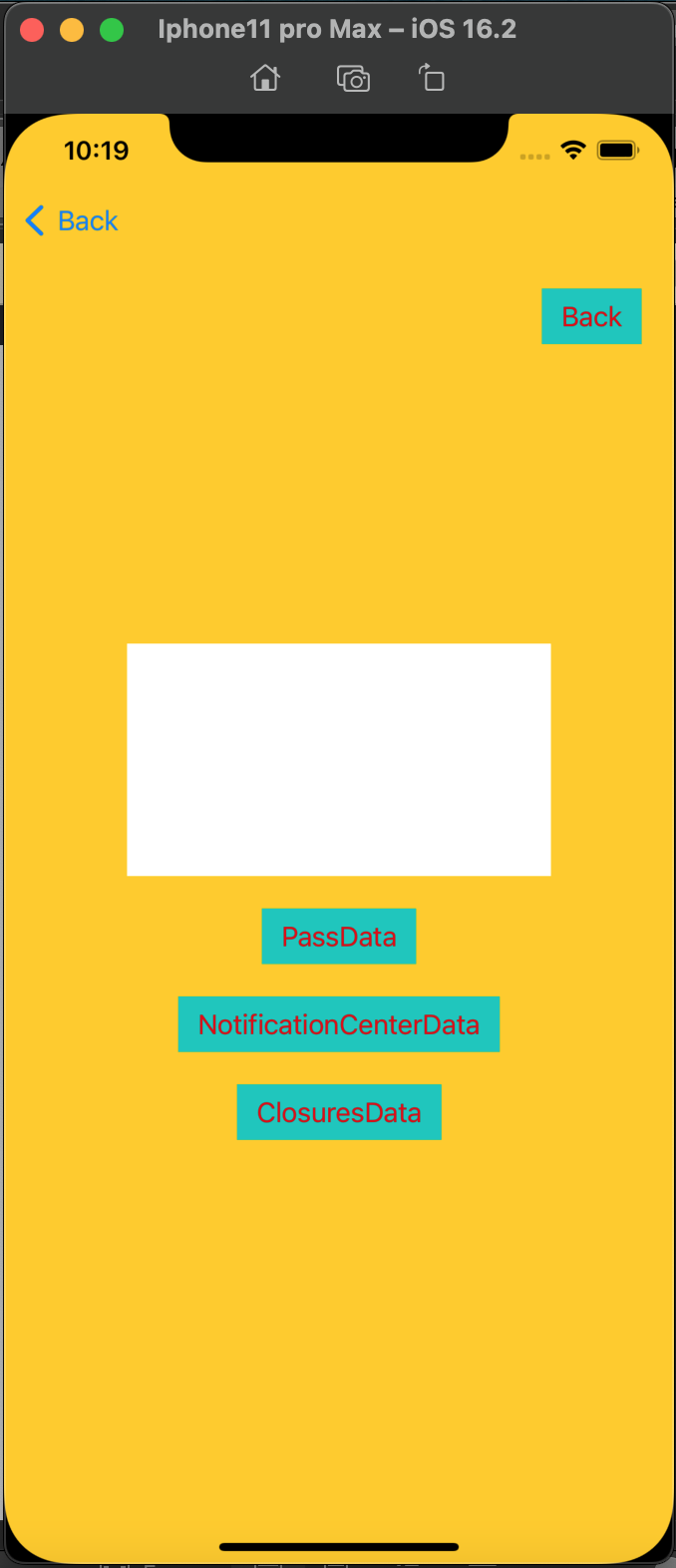
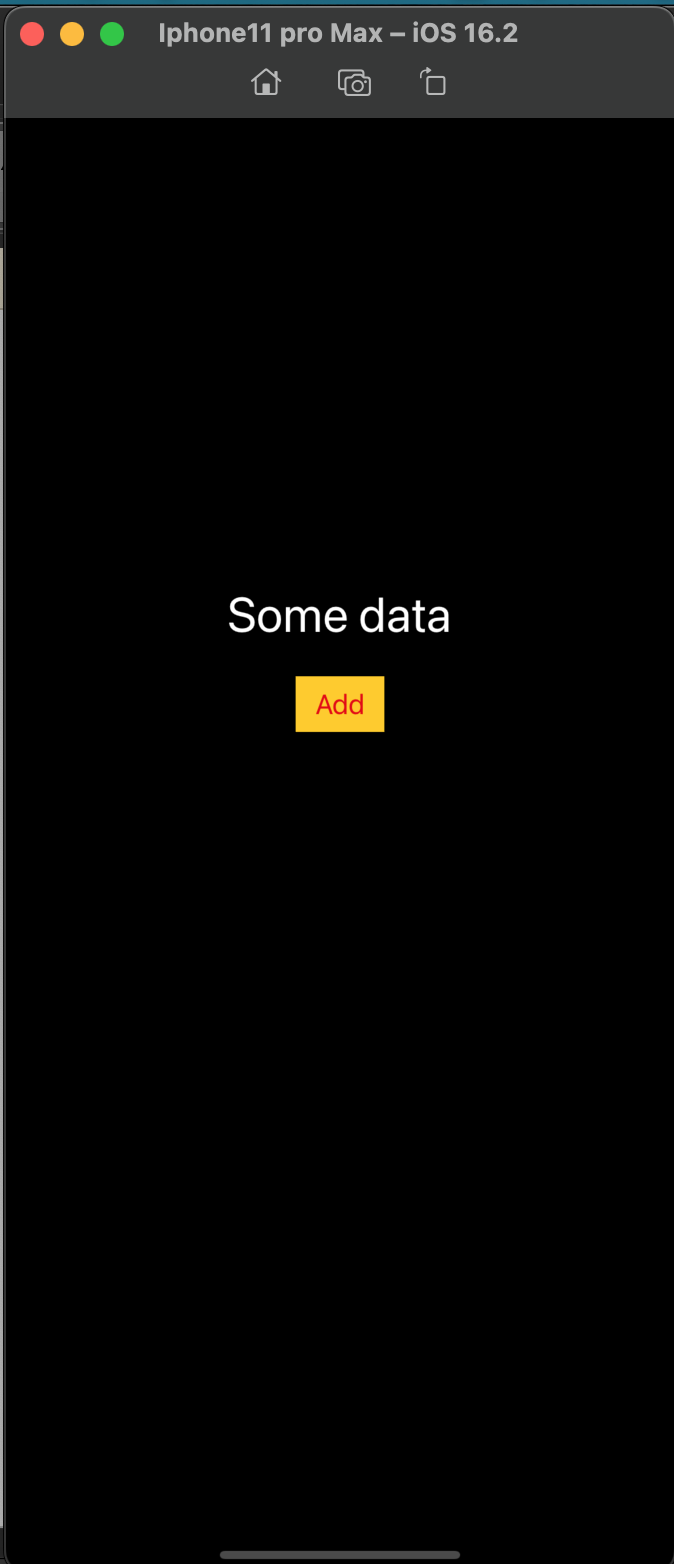
* **Triển khai giao thức trong ViewControllerA:**

**ViewControllerA.swift:**



Kết quả hiển thị giữa 2 màn hình:

data = “Some data”

Ví dụ trên được hiểu khi ta chuyển từ màn hình vàng về màn hình đen sẽ đồng thời sẽ truyền dữ liệu data về màn hình đen và được gán vào label.

Sử dụng **Delegate** để chuyển dữ liệu giữa các **ViewController** là một cách tiếp cận linh hoạt và mạnh mẽ, đặc biệt hữu ích khi cần truyền dữ liệu ngược trở lại từ một **ViewController** được hiển thị. Cơ chế này giúp tách biệt trách nhiệm giữa các **ViewController**, làm cho mã nguồn của bạn trở nên dễ bảo trì và mở rộng hơn.

1. ***Sử dụng*** ***NotificationCenter***

### Phát hành Thông báo

Để phát hành một thông báo, bạn sử dụng phương thức **post(name:object:userInfo:)** của **NotificationCenter**.

**NotificationCenter.default.post**(**name**: .myNotificationName, **object**: nil, **userInfo**: ["**key**": "**value**"])

* **name**: Tên của thông báo, thường được định nghĩa dưới dạng một thuộc tính mở rộng (extension) của Notification.Name.
* **object**: Đối tượng gửi thông báo, có thể là bất kỳ đối tượng nào hoặc nil.

### **userInfo**: Một từ điển chứa dữ liệu bổ sung bạn muốn gửi kèm thông báo. Có thể là nil nếu không có dữ liệu kèm theo.

### Đăng ký Nhận Thông báo

Để đăng ký nhận thông báo, bạn sử dụng phương thức **addObserver(\_:selector:name:object:)** của **NotificationCenter**.

* **observer**: Đối tượng sẽ nhận thông báo, thường là self.
* **selector**: Phương thức sẽ được gọi khi nhận được thông báo.
* **name**: Tên của thông báo mà bạn muốn lắng nghe.
* **object**: Đối tượng gửi thông báo mà bạn quan tâm, hoặc nil để lắng nghe từ mọi đối tượng.

### Xử lý thông báo

### 

**notification**: Đối tượng **Notification** chứa thông tin về thông báo, bao gồm **userInfo** nếu có.

### Huỷ Đăng ký Thông báo

Để huỷ đăng ký nhận thông báo, bạn sử dụng phương thức removeObserver(\_:name:object:) của NotificationCenter.

**NotificationCenter**.**default**.**removeObserver**(self, **name**: .myNotificationName, **object**: nil)

* + - * **observer**: Đối tượng huỷ đăng ký, thường là self.
      * **name**: Tên của thông báo mà bạn không muốn lắng nghe nữa.
      * **object**: Đối tượng gửi thông báo mà bạn không quan tâm nữa, hoặc nil.

### VD:

### FirstViewController.swift

### Đây là view controller đăng ký nhận thông báo và xử lý dữ liệu nhận được.

### 

### SecondViewController.swift

### Đây là view controller phát hành thông báo khi có dữ liệu cần truyền.

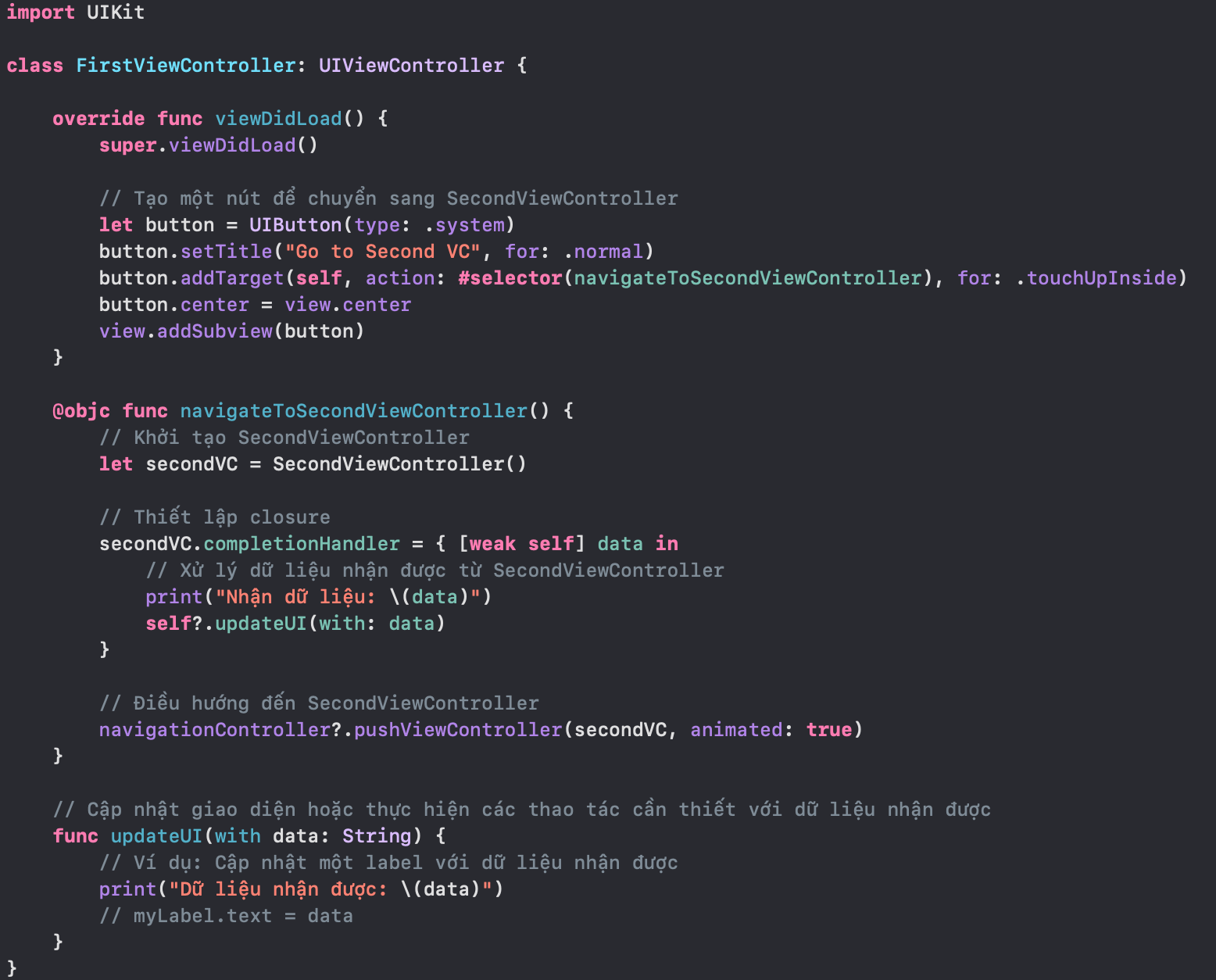
### 

1. ***Sử dụng*** ***Closures (Callbacks)***

### VD

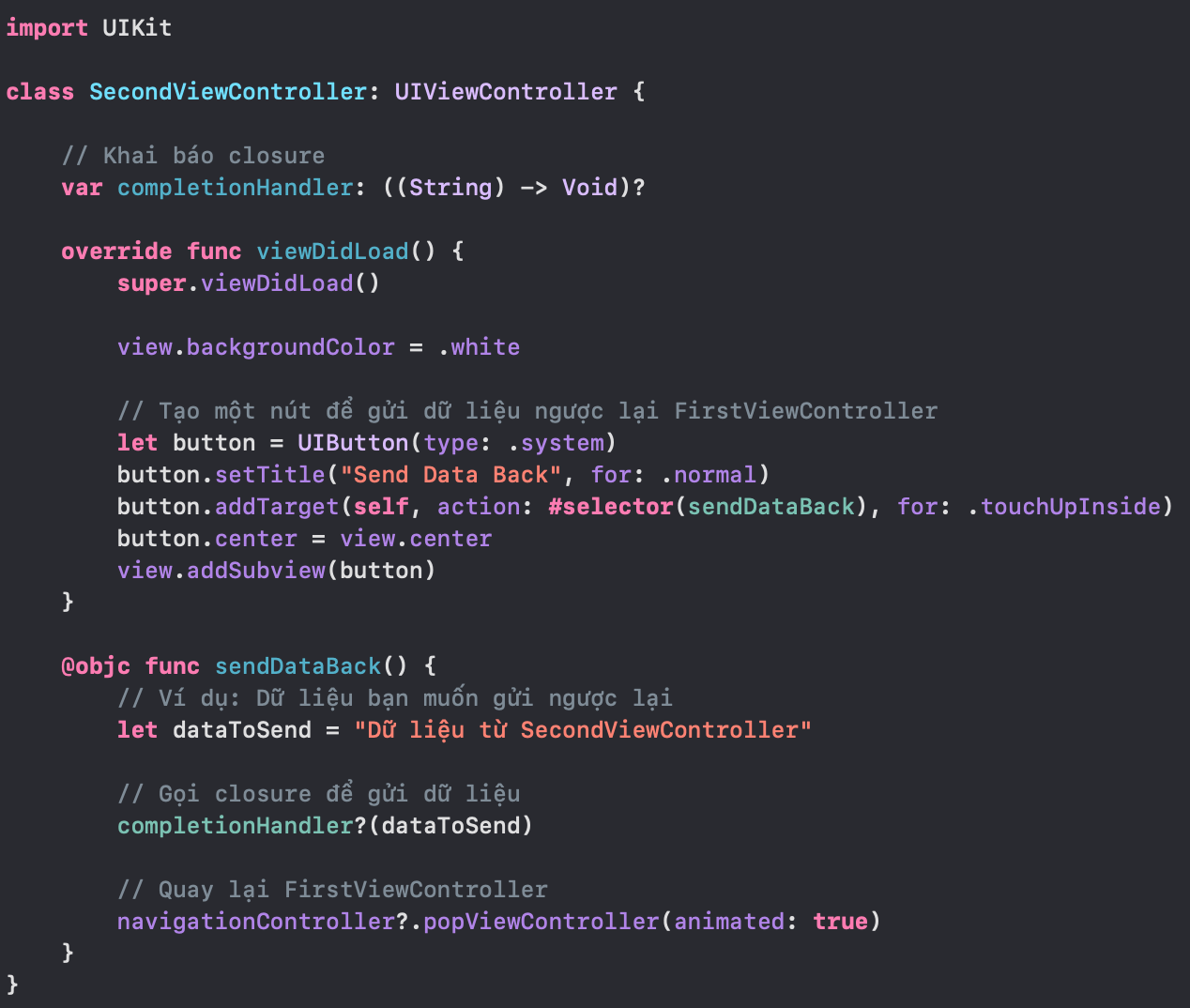
### FirstViewController.swift

Đây là view controller nơi bạn sẽ chuyển đến SecondViewController và nhận dữ liệu ngược lại qua closure.



### SecondViewController.swift

Đây là view controller nơi bạn sẽ gửi dữ liệu ngược lại FirstViewController thông qua closure.



**Kết luận**

Mỗi phương pháp trên đều có ưu và nhược điểm riêng, và sự lựa chọn phương pháp nào phụ thuộc vào kiến trúc và yêu cầu cụ thể của ứng dụng của bạn. Dưới đây là một số lời khuyên chung:

* **Segue và Prepare for Segue**: Tốt cho truyền dữ liệu khi sử dụng Storyboard.
* **Truyền trực tiếp**: Dễ hiểu và đơn giản, tốt cho các ứng dụng nhỏ.
* **Delegate**: Tốt cho việc truyền dữ liệu ngược lại, đặc biệt là khi cần phản hồi.
* **NotificationCenter**: Tốt cho việc truyền dữ liệu không đồng bộ và khi các ViewController không có kết nối trực tiếp.
* **Singleton**: Tốt cho việc chia sẻ dữ liệu toàn cục, nhưng cần cẩn thận để tránh các vấn đề về concurrency.
* **Closures**: Dễ sử dụng và phù hợp cho việc truyền dữ liệu ngược lại mà không cần tạo delegate.