# **MainTabBarController**

## **Class MainTabBarController được kế thừa từ class UITabBarController.**

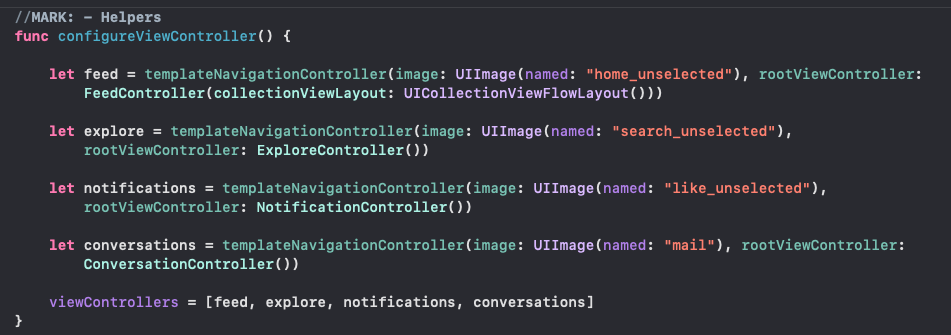


Figure 1: Khởi tạo TabBarController

## **Khởi tạo TabBarController: Thu được kết quả**

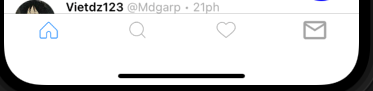


Figure 2: Kết quả khởi tạo TabBarController

## **Lấy viewController đầu tiên trong TabBar**

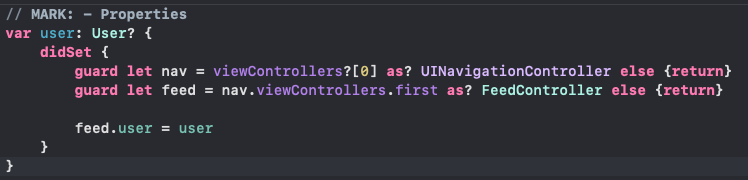


Figure 3: Lấy Controller đầu tiên trong TabBarController

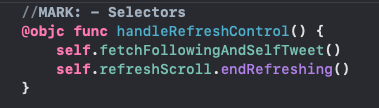
Tại viewController[0] khả năng cao là lấy controller đầu tiên trong TabBarController, bởi vì mọi Contoller trong TabBarController này đều được nhúng vào 1 UINavigationController nên ta được Code như trên.

# **Feed Controller**

## **Configure cho refreshController**

No description available.

No description available.



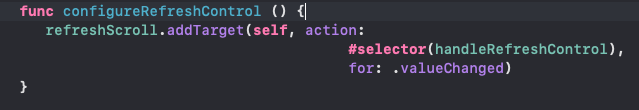


Figure 4: Configure RefreshController

## **Các thuộc tính của ContentMode ImageView**

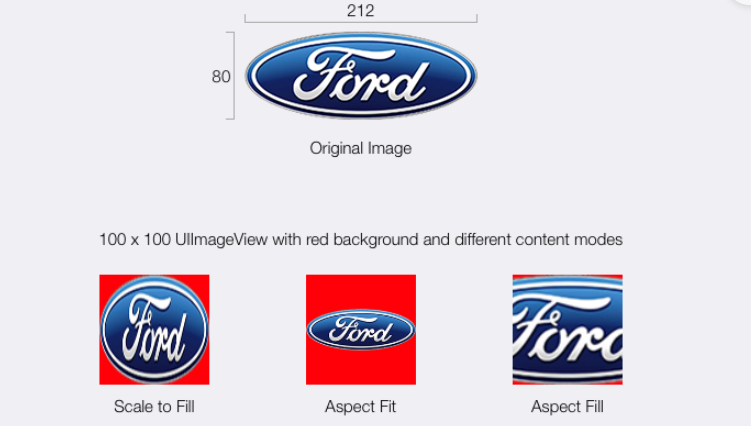


Figure 5: ContentMode ImageView

* **UIViewContentModeScaleToFill**: Scales the content to fit the size of itself by changing the aspect ratio of the content if necessary(Kéo chiều dài với chiều rộng để chạm các cạnh)
* **UIViewContentModeScaleAspectFit**: Scales the content to fit the size of the view by maintaining the aspect ratio. Any remaining area of the view’s bounds is transparent(trong suốt).
* **UIViewContentModeScaleAspectFill**: Scales the content to fill the size of the view. Some portion of the content may be clipped to fill the view’s bounds.

## **Content Inset, Content OffSet, Content Size**

* Content Offset

contentOffset is the point at which the origin of the content view is offset from the origin of the scroll view. In other words, it is where the user has currently scrolled within the scroll view. This obviously changes as the user scrolls.

contentOffset indicates the current position of the scroll view content, relative to the origin coordinates on the top-left corner. The content size remains the same, but the content offset changes to respond to user interaction.

**Nghĩa là**: TableView hoặc CollectionView sẽ được nhúng 1 scrollView vào. Ta luôn có size của scrollView sẽ luôn lớn hơn size của tableView và CollectionView 1 tí. Theo như đoán thì lớn hơn khoảng 48. Ta có x và y của scroll sẽ được so với view, nghĩa là x và y ban đầu của scrollView sẽ là -48, vì nó sẽ bị thừa ra ngoài 1 tí như đã nhận định. Và contentOffset chính là vị trí của Content so với scrollView. Khi ta kéo xuống thì contentOffset và khi kéo ngược lên thì sẽ giảm dần. Vì vậy contentOffset cho ta biết vị trí của content của tableView or CollectionView so với scrollView.

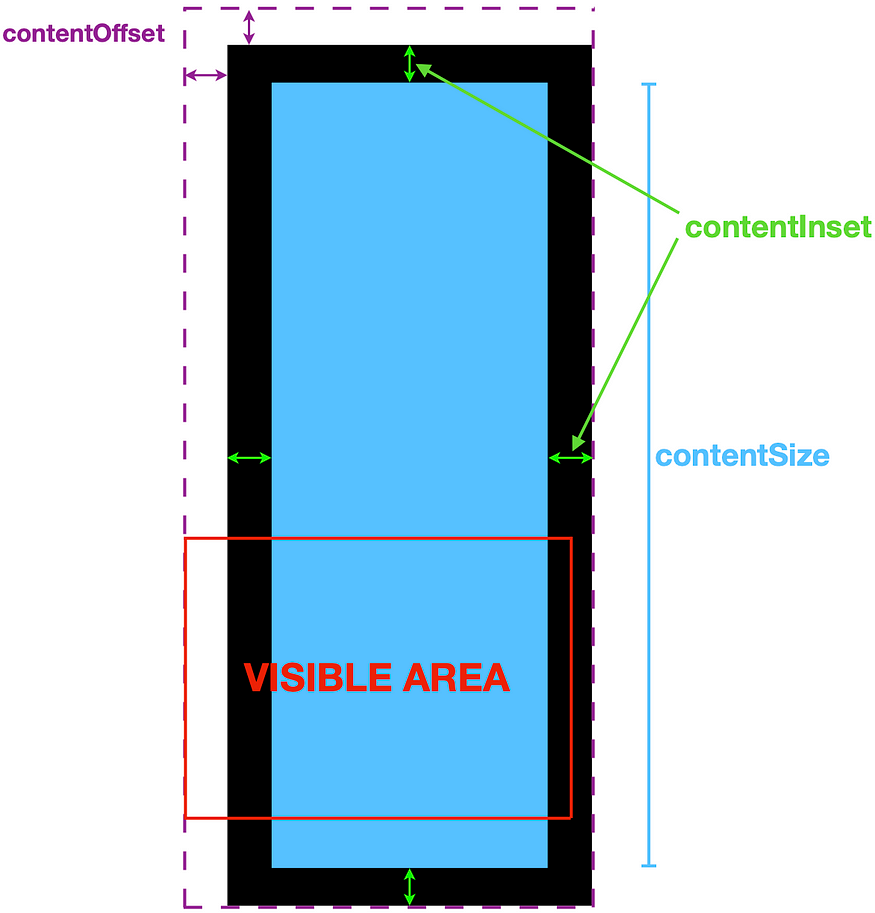


Figure 6: Tổng quan ContentSize, Offset, Inset

Nhìn vào ảnh ta thấy, size của scrollView luôn lớn hơn size của content bên trong, cụ thể là lớn hơn size tableView và CollectionView

* Content Inset:

The contentInset is how much the content is visually inset inside the scroll view. It is the custom distance that the content view is inset from the safe area or scroll view edges.

contentInset allows you to specify margins or padding around the content in the scrollview. You can specify the margin programmatically as follows:



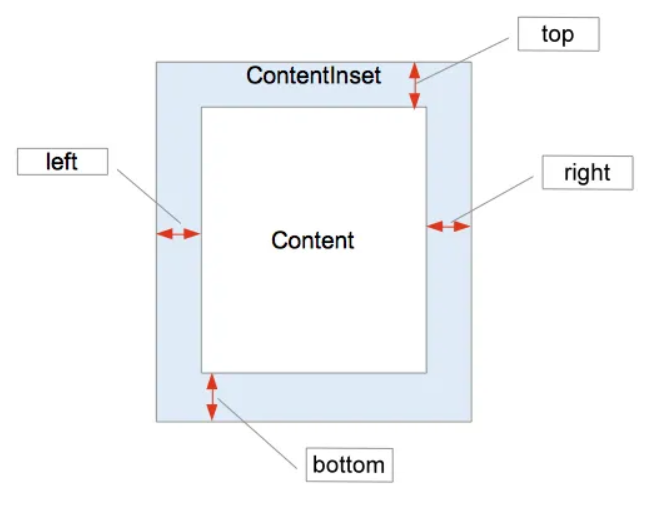


Figure 7: Content Inset

Ta có ví dụ:

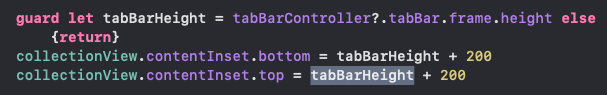


Figure 8: Set Content Inset For ProfileController

Kết quả ta được:

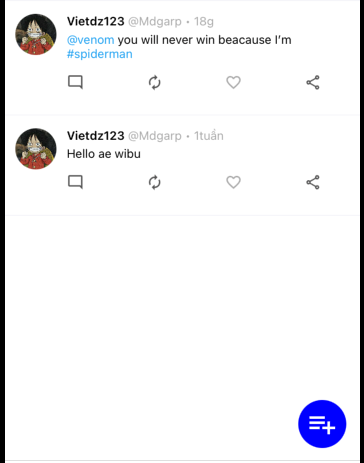
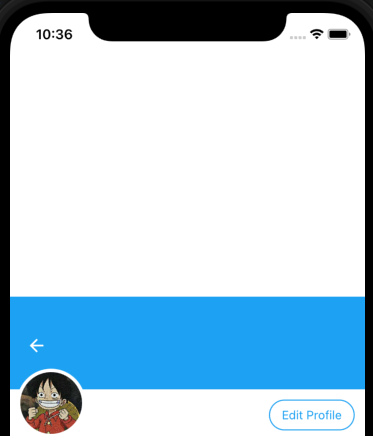


Figure 9: Kết quả thu được khi xét

Ta thấy vị trí ban đầu của collectionView đã bị thụt xuống 1 đoạn khoảng 280(200 + 80), và vị trí cuối cùng của cell trong collectionView còn cách 1 đoạn 280 so với bottom của collectionView

* Content Size

contentSize defines how big the scrollable content is. It is the size of the content within the UIScrollView and how long it can be within the UIScrollView.

The scroll view must know the size of the content-view so it knows when to stop scrolling. its default value is zero, and it must be set to use any scroll view, even if the content-size is smaller than the scroll view’s own size.

# **NotificationCenter**

Để gửi thông báo:

No description available.

Figure 10: Gửi thông báo sử dụng NotificationCenter

Nhận thông báo:

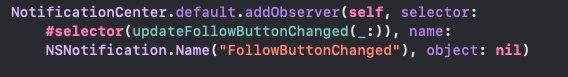


Figure 11: Nhận Thông báo Notification

Xử lý hàm Handle và parse dữ liệu

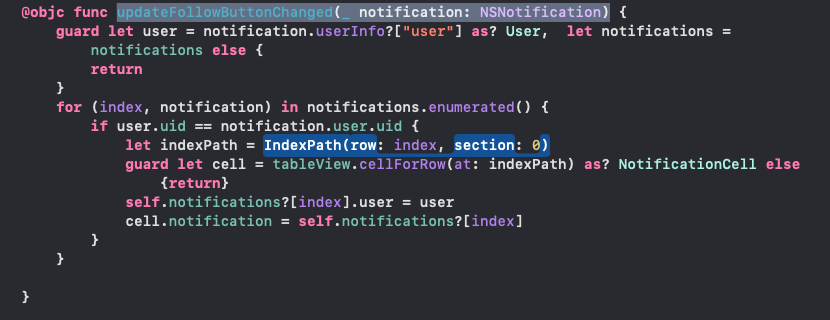


Figure 12: Hàm handle Notifcation và parse dữ liệu

# **UIPickerView**

Khởi tạo imagePicker

No description available.

No description available.

Figure 13: Init imagePicker

Sử dụng present để present imagePickerViewController

No description available.

Sau khi đã chọn xong ảnh:

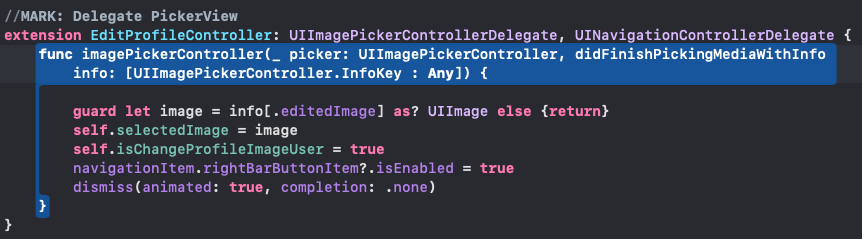


Figure 14: Lấy ảnh sau khi chọn từ ImagePicker

# **TextAttributedString**

Để xét 1 chuỗi String có nhiều Font, nhiều color khác nhau, ta sử dụng TextAtributed.

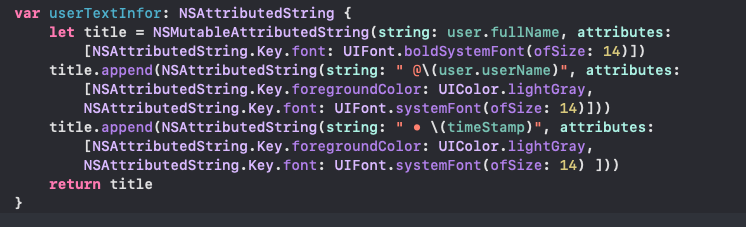


Figure 15: Ví dụ TextAttributed String

Kết quả:

No description available.

Figure 16: Kết quả sử dụng Attributed String

Ta thấy cùng 1 dòng String, nhưng trong đó các đoạn subString lại có các thuộc tính như color, màu khác nhau. Ta cũng có thể sử dụng TextAttributed lên button,…

No description available.

No description available.

Figure 17: Áp dụng textAttributed lên button

# **DateFormatter**