

---

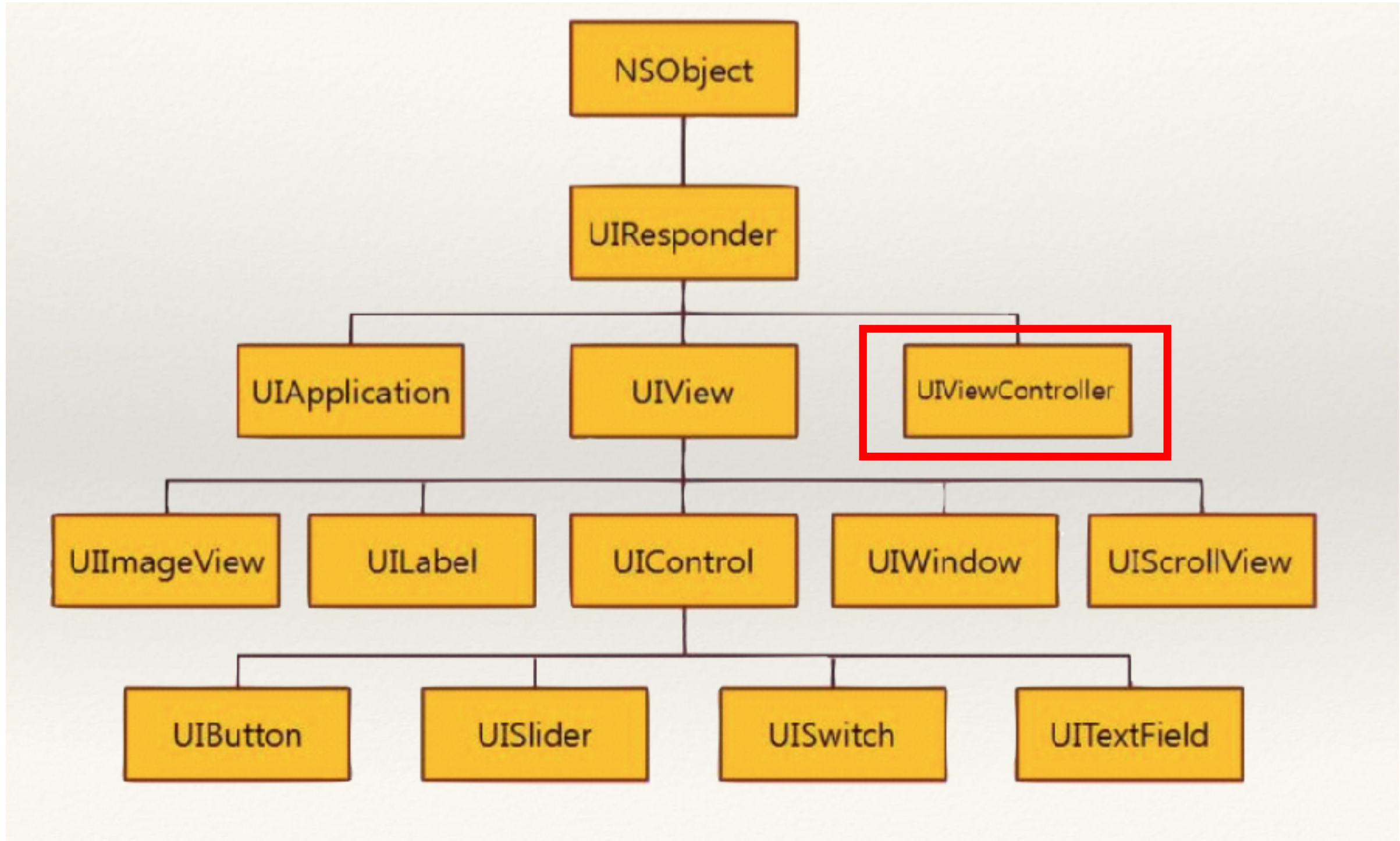
# UIViewController

---

강사 주영민

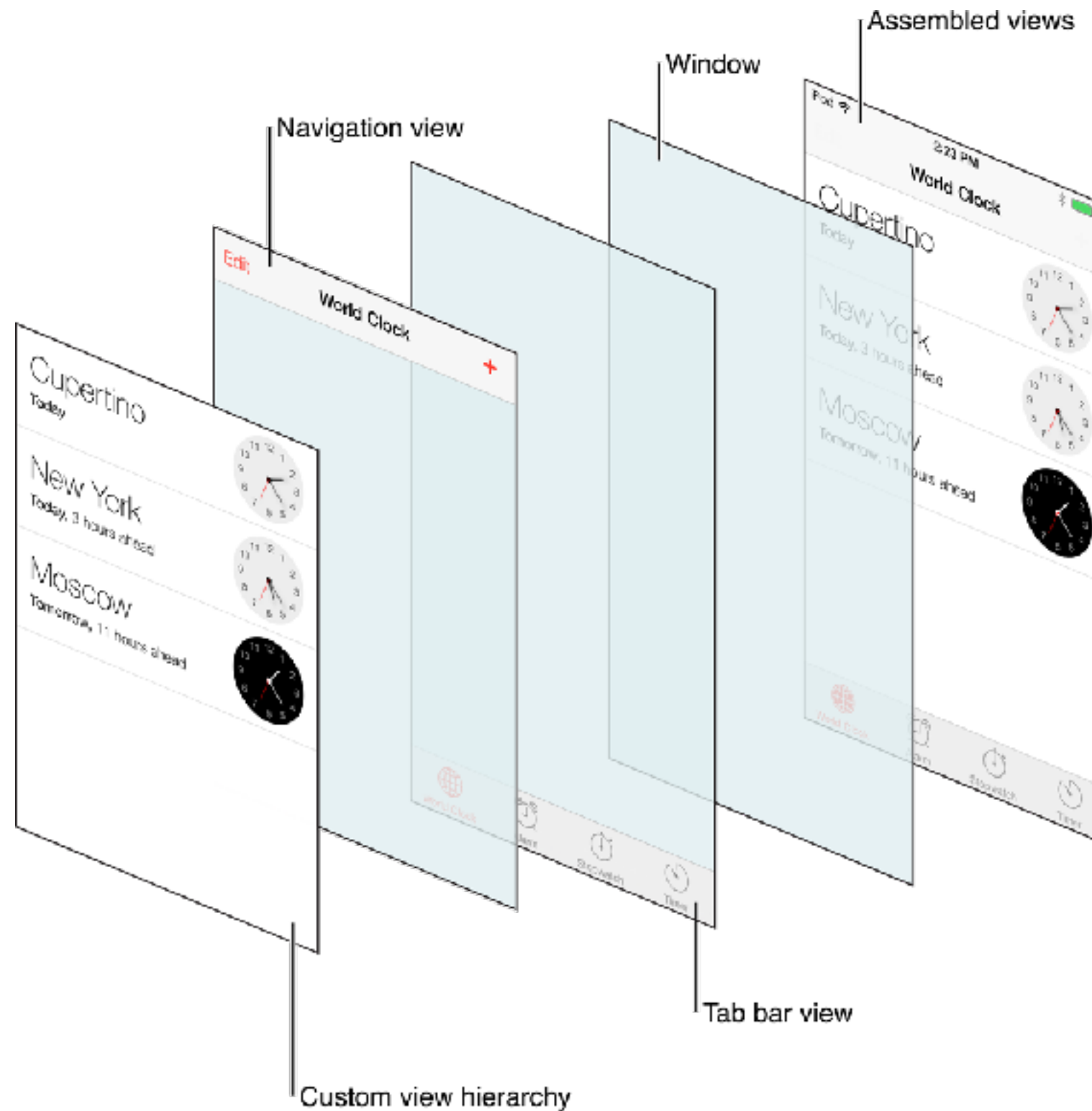
# UI Class Hierarchy

---



# UIViewController Interface

---



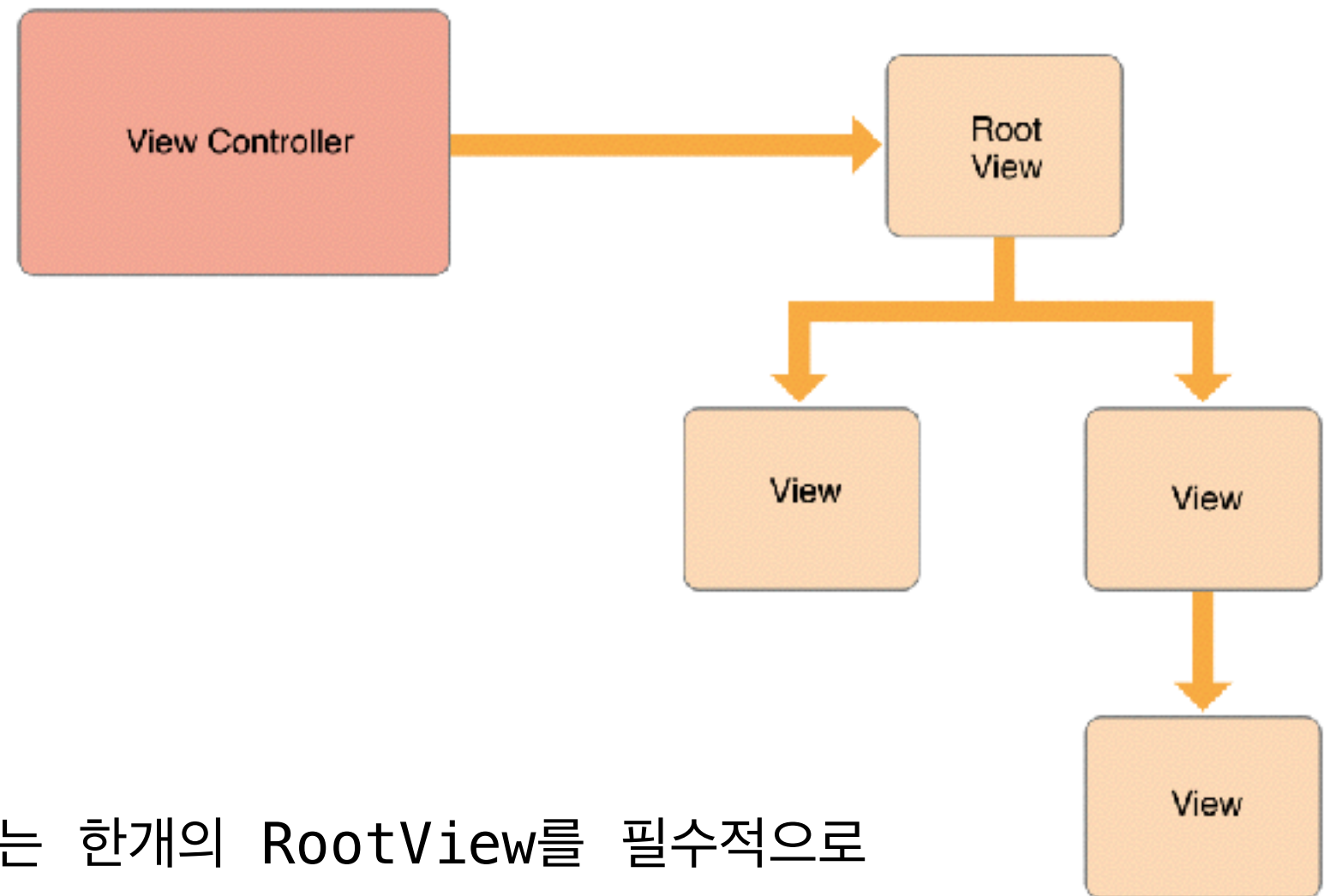
# UIViewController

---

- 앱의 기초가 되는 내부 구조(일반적으로 한 화면은 하나의 UIViewController로 이루어 진다.)
- 모든 앱은 적어도 한개 이상의 UIViewController를 가지고 있으며, 대부분의 앱은 여러개의 UIViewController로 이뤄져 있다.
- UIViewController는 사용자의 인터렉션과 앱의 데이터 사이에서 컨트롤의 역할을 한다. (MVC디자인 패턴)
- UIViewController는 모든 View의 관리, 사용자 이벤트 핸들링, UIViewController간의 전환 등의 역할을 수행한다.

# UIViewController - Root View

---



- 모든 UIViewController는 한개의 RootView를 필수적으로 가지고 있다.
- 화면에 표시되는 모든 View는 RootView의 SubView로 존재한다.

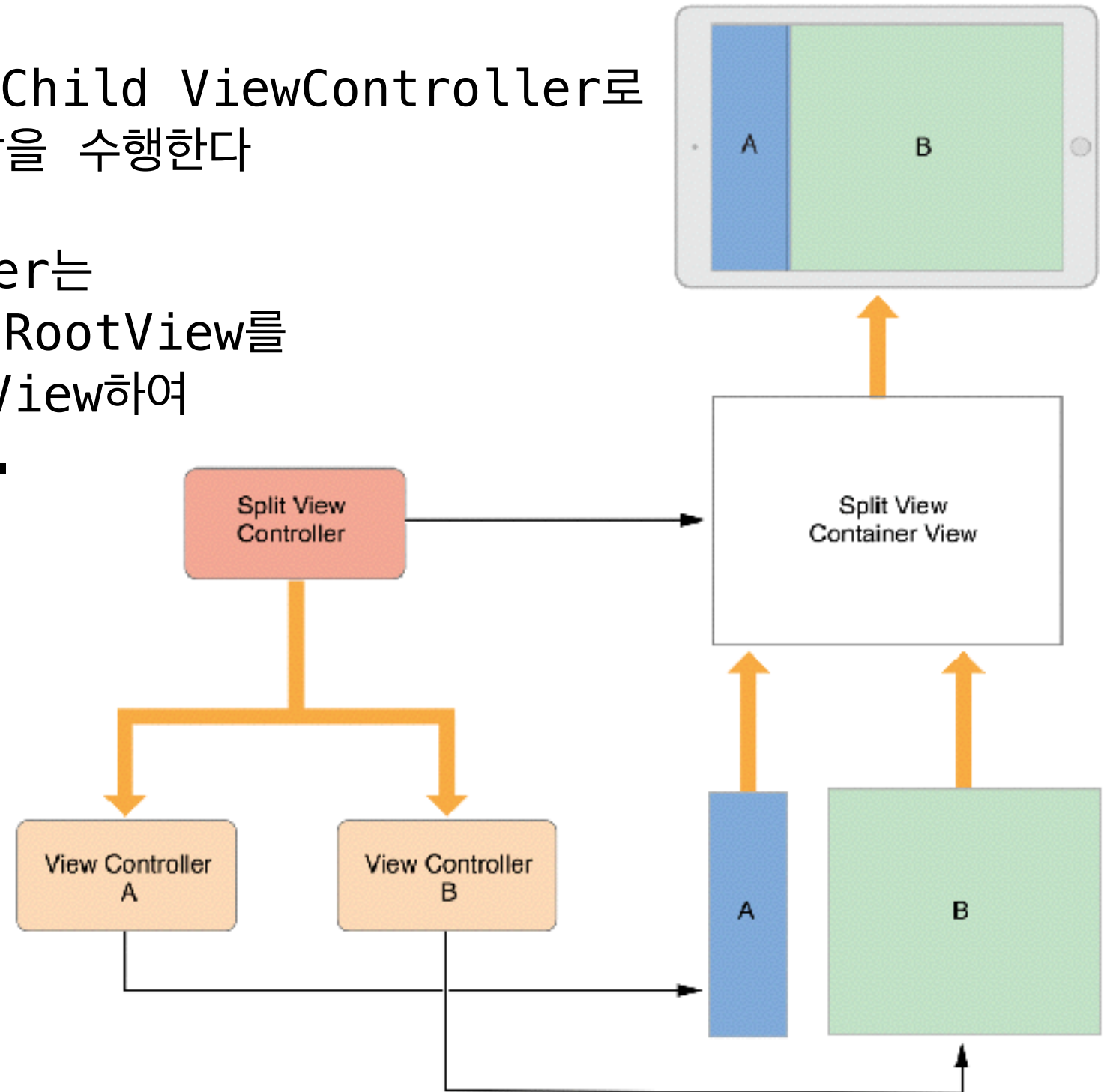
# ViewController 특징

---

- Child ViewController(자식 ViewController)
- UserInteraction(사용자 인터렉션처리)
- Data Marshaling(중계자)
- Resource Management(리소스 자원관리)

# Child ViewController

- 한 UINavigationController는 다른 UINavigationController를 Child ViewController로 여기므로서 Container의 역할을 수행한다
- Container ViewController는 Child ViewController의 RootView를 자신의 RootView에 addSubview하여 SubView로서 화면에 표시한다.



# UserInteraction

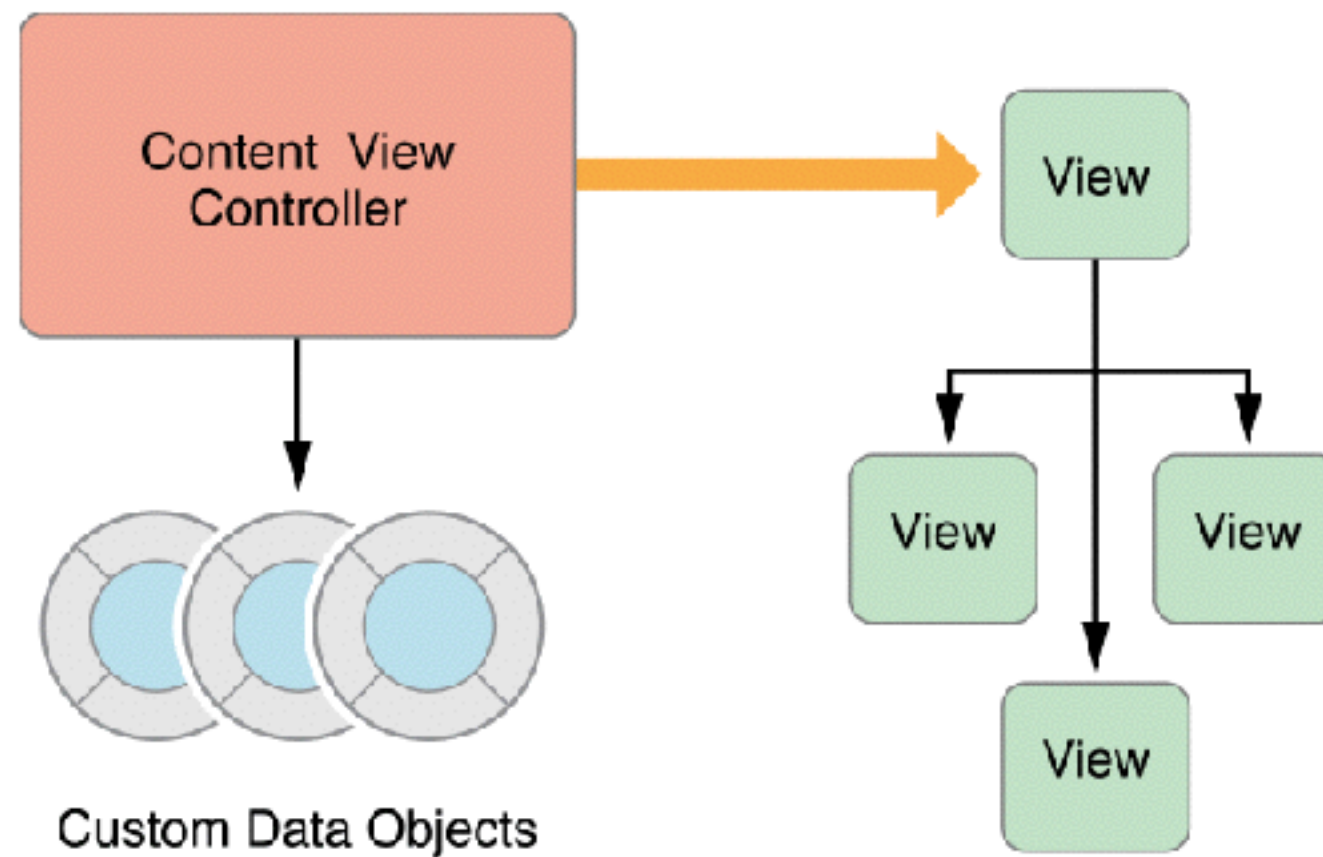
---

- UINavigationController는 UIResponder를 상속받은 클래스로 이벤트 체인으로부터 오는 모든 이벤트를 효과적으로 처리해야한다.
- 즉 사용자의 모든 이벤트는 ViewController가 받아서 각 View에 해당되는 Method와 Delegate로 처리한다.



# Data Marshaling(중계자)

- ViewController는 자신이 관리하는 View들과 앱 내부의 데이터와의 중계자 역할을 한다. (MVC패턴에 의해)



# Resource Management

---

- UINavigationController안에 있는 모든 View나 Instance는 모두 ViewController의 책임하에 있다.
- 메모리가 부족시 `didReceiveMemoryWarning()` 메소드가 자동으로 불리며, 오래동안 사용하지 않은 인스턴스와 다시 쉽게 만들수 있는 인스턴스를 제거할 수 있어, 메모리를 효율적으로 관리한다.
- `didReceiveMemoryWarning()` 메소드 호출시, 해당 ViewController는 나타날때 `ViewDidLoad()` 메소드를 다시 호출한다.

# ViewController 종류

---

- **General** View Controller
  1. UIViewController
  2. UITableViewController
  3. UICollectionViewController
- **Container** View Controller
  1. UINavigationController
  2. UITabBarController
  3. UISplitViewController
  4. ....

# General View Controller

---

- 일반 적인 ViewController형태
- 각 ViewController가 Root View를 가지고 있다.
  1. UINavigationController의 Root View 는 UIView
  2. UITableViewController의 Root View 는 UITableView
  3. UICollectionViewController의 Root View 는 UICollectionView

# UIViewController Instance 생성

---

- UIViewController의 Instance를 생성하는 방법에는 크게 2가지가 있다.

1. 일반적인 초기화 메소드를 활용한 instance 생성

```
let vc:UIViewController = UIViewController()
```

2. **xib파일**이 Storyboard에 있을때, Storyboard를 통한 인스턴스 생성방법

```
let storyboard = UIStoryboard(name: "Storyboard이름", bundle: nil)
let vc:UIViewController =
storyboard.instantiateViewController(withIdentifier: "StoryboardID")
```

# UIViewController간 화면 전환

---

- 현재 UIViewController에서 새로운 화면으로 전환을 위해 ViewController는 화면전환을 해야한다.
- UIViewController의 화면전환 방법은 크게 3가지가 존재한다.  
(Present Modally, UINavigationController, UITabBarController)
- Storyboard내에서의 화면 전환에는 Segue Instance를 사용해서 직관적으로 ViewController간의 관계를 보여준다.

# Present Modally

---

- 일반적인 화면 전환을 위한 방법
- 기존 ViewController에서 대상 ViewController를 Present한다. (일반적으로 기존 VC가 대상VC Instance를 생성한다.)
- 화면이 전환 되어도 기존VC를 메모리에 존재 하며, 대상VC를 되돌아 갈때 메모리에서 제거된다.
- 다음 두 메소드를 사용해서 화면 전환과 되돌아 오기가 가능하다.

```
present(UIViewController, animated: Bool,  
        completion: (() -> Void)?)
```

```
dismiss(animated: Bool, completion: (() -> Void)?)
```

# Present Modally Sample Code

---

- present ( 기존 ViewController Method 내부)

```
func goToNextVC() {  
    //다음 인스턴스 생성  
    let storyboard = UIStoryboard(name: "Storyboard이름", bundle: nil)  
    let nextVC = storyboard.instantiateViewController(withIdentifier:  
"NextViewController") as! NextViewController  
    //present  
    self.present(nextVC, animated: true) {  
        //컴플리션 클로저 내부  
    }  
}
```

- dismiss (대상 ViewController Method 내부)

```
func backVC() {  
    //되돌아 가기  
    dismiss(animated: true, completion: nil)  
}
```



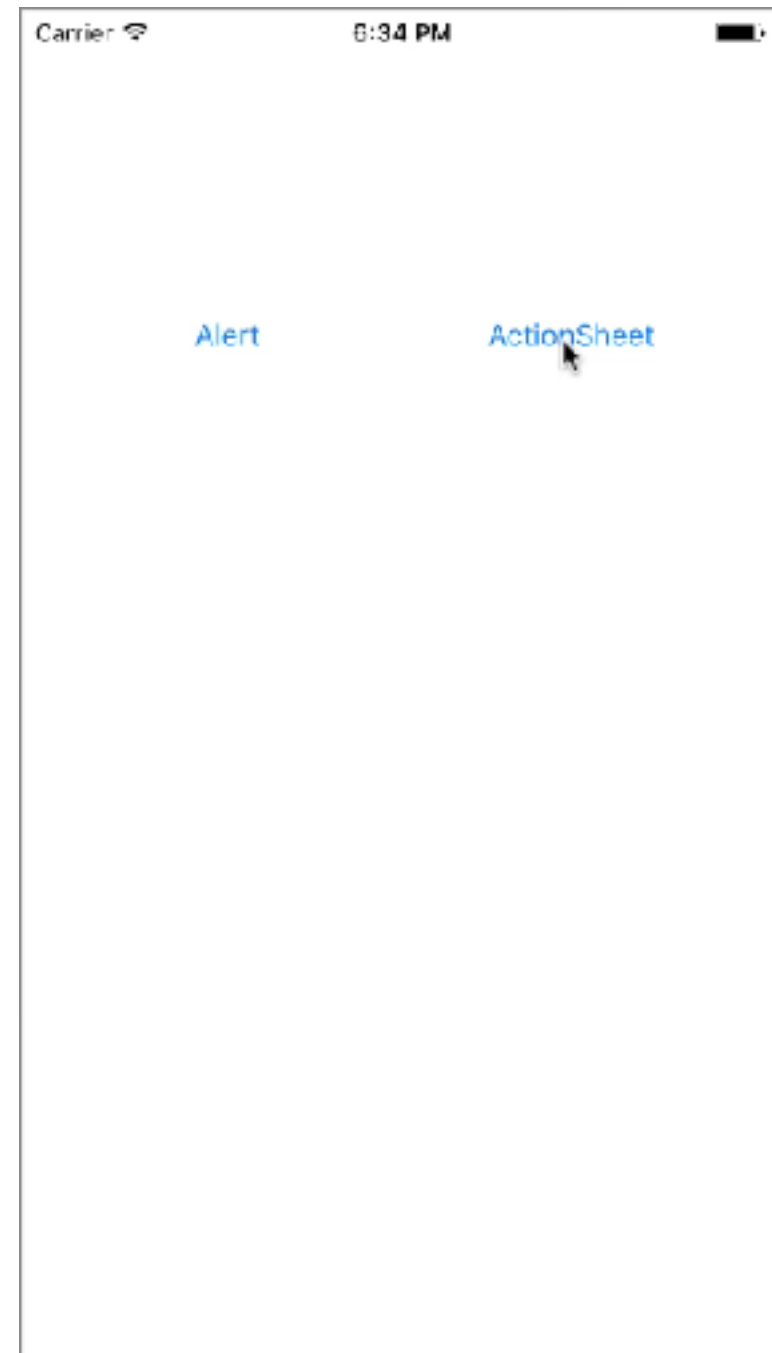
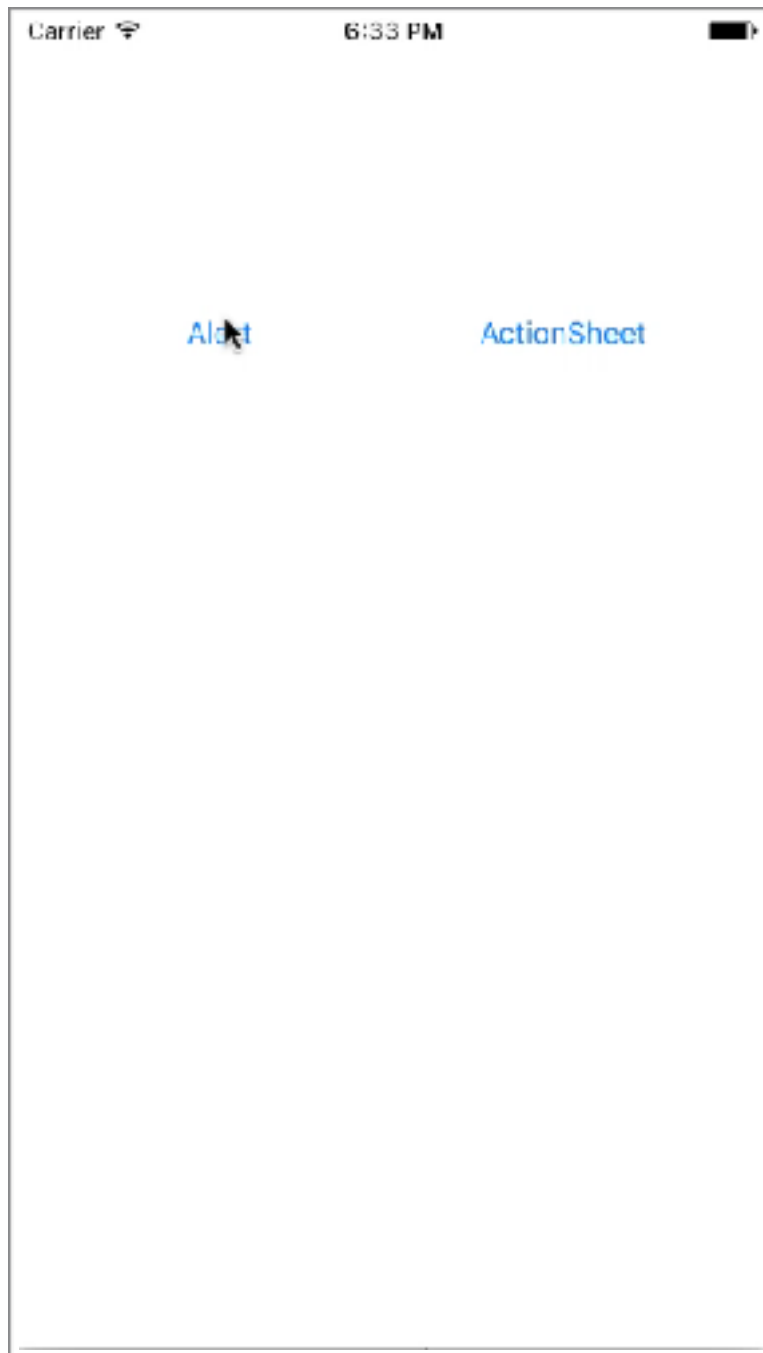
---

# UIAlertController

---

# UIAlertController

---



# Step 1. file 보기

---

- UIAlertController 파일 보기

## Step 2. Sample Code

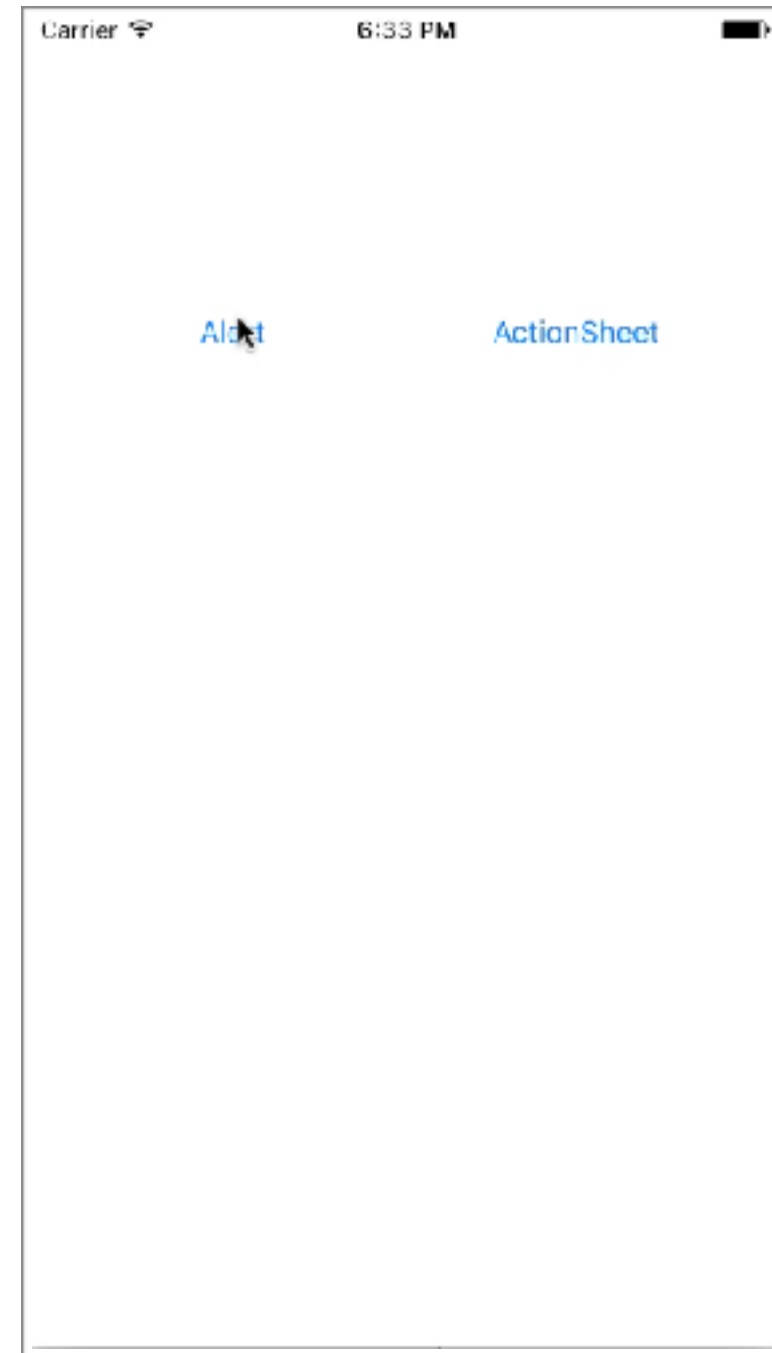
---

```
func btnAction(sender:UIButton) {  
    let alertVC = UIAlertController.init(title: "타이틀",  
                                         message: "알럿 메세지",  
                                         preferredStyle: .alert)  
  
    let okAction = UIAlertAction.init(title: "확인",  
                                       style: .default) { (action) in  
        //버튼 클릭시 실행 코드  
    }  
  
    alertVC.addAction(okAction)  
  
    self.present(alertVC, animated: true) {  
        //알럿 띄운 후 실행할 액션  
    }  
}
```

# Step 3. Exercise

---

- Alert // ActionSheet만들기
- 다양한 Action을 추가해보기



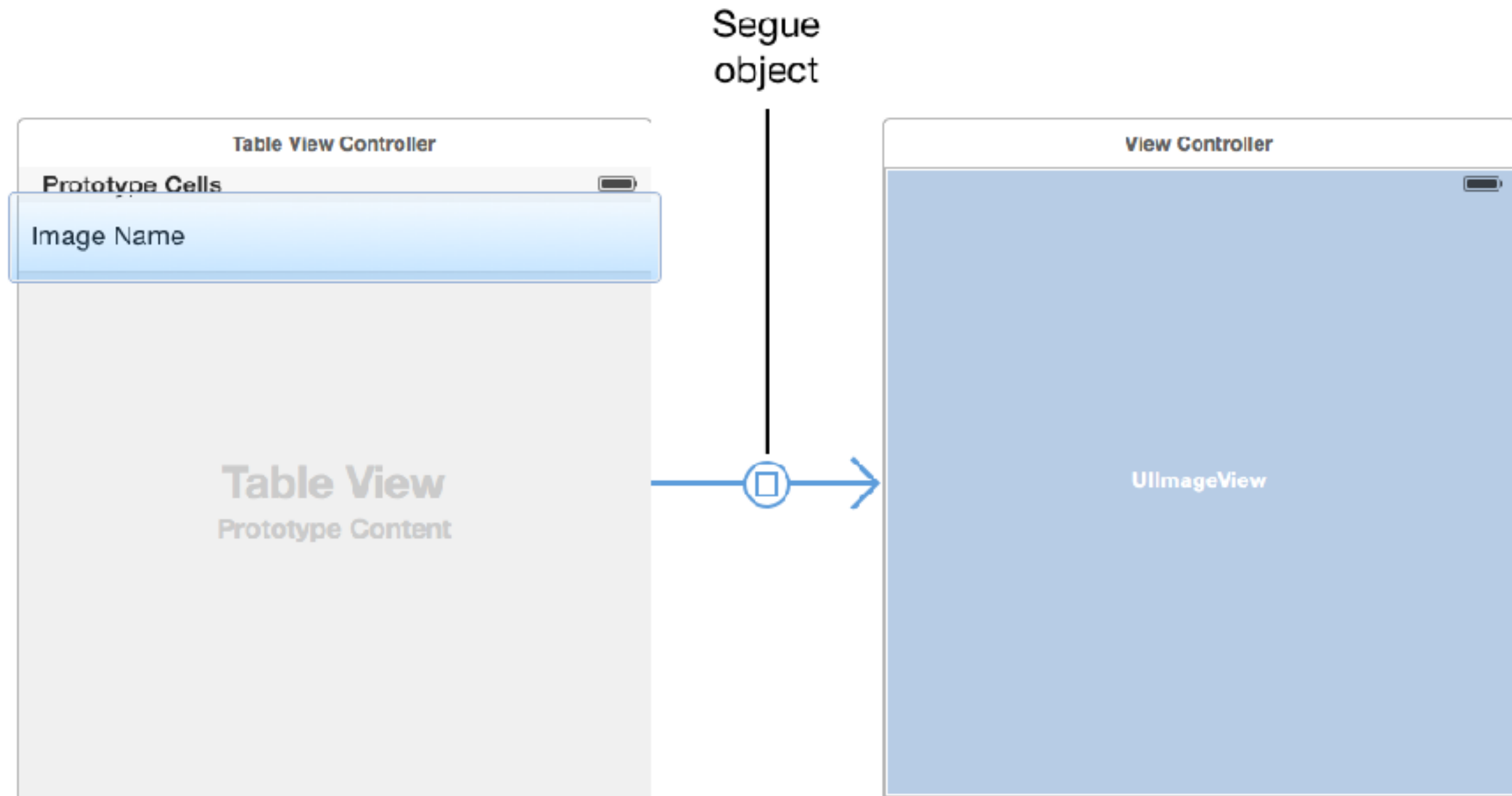
# Segue

---

- Storyboard 파일 내 두 개의 ViewController사이의 화면전환을 정의한 인스턴스
- 앱의 인터페이스 흐름을 확인하는데 도움
- Segue의 시작점은 UIButton, UITableView의 selected row, UIGesture등으로 시작하며, 끝점은 전환되는 다음 UIViewController를 가르킨다.
- segue화면전환이 된 UIViewController는 unwind segue를 통해 되돌릴수 있다.(역 화면 전환)

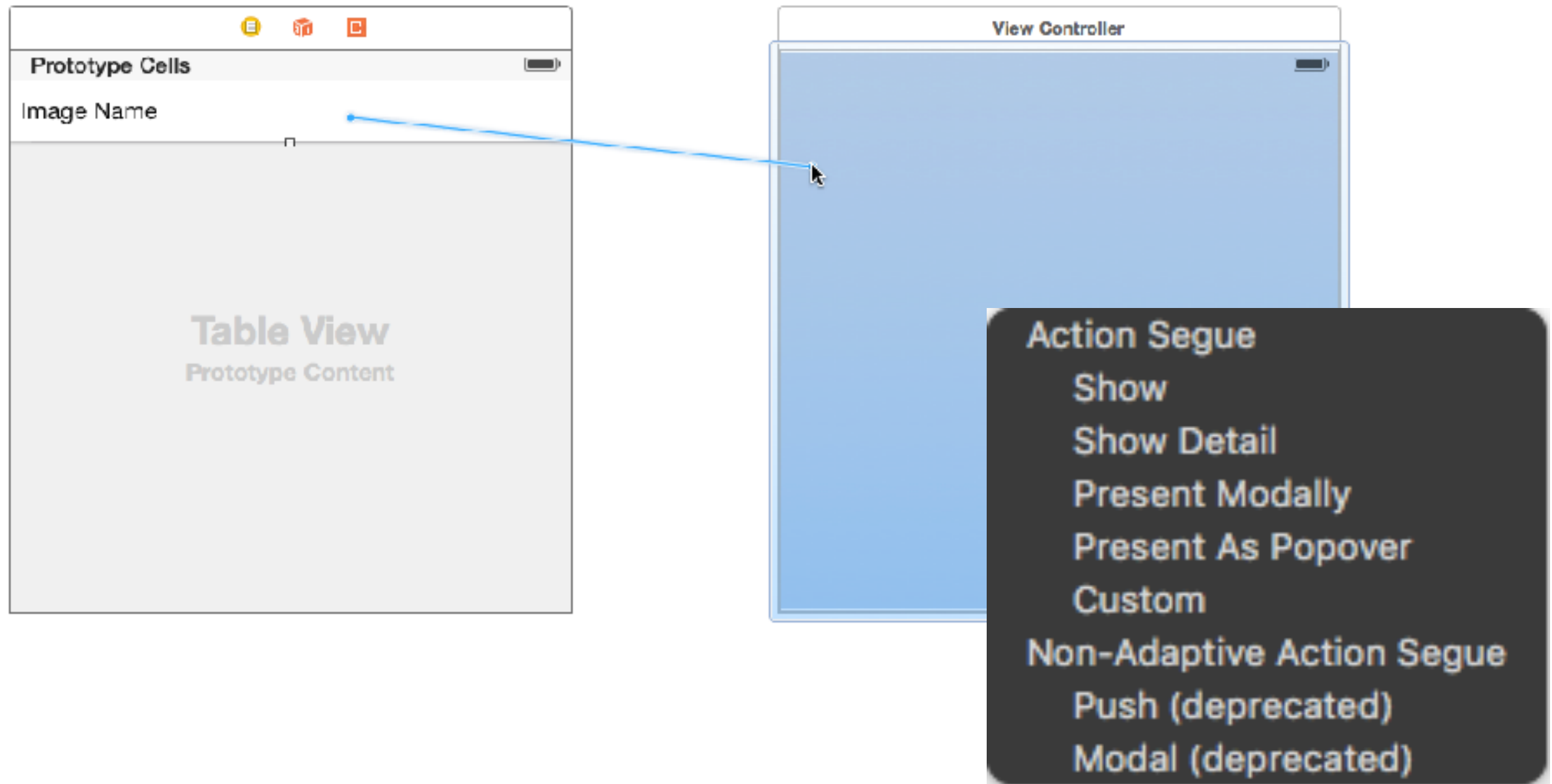
# Segue

---



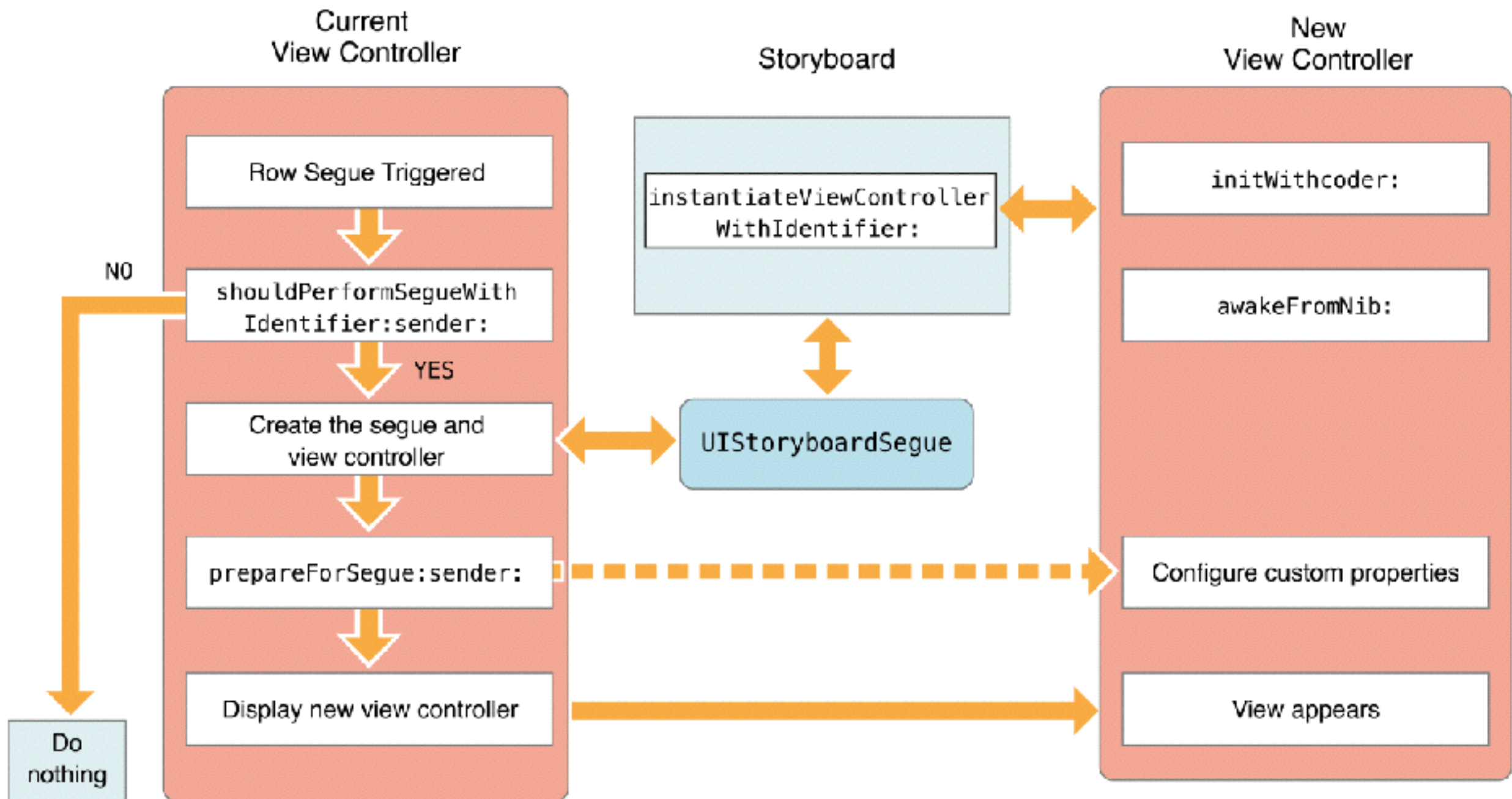
# Create Segue

---





# Segue 작동 순서



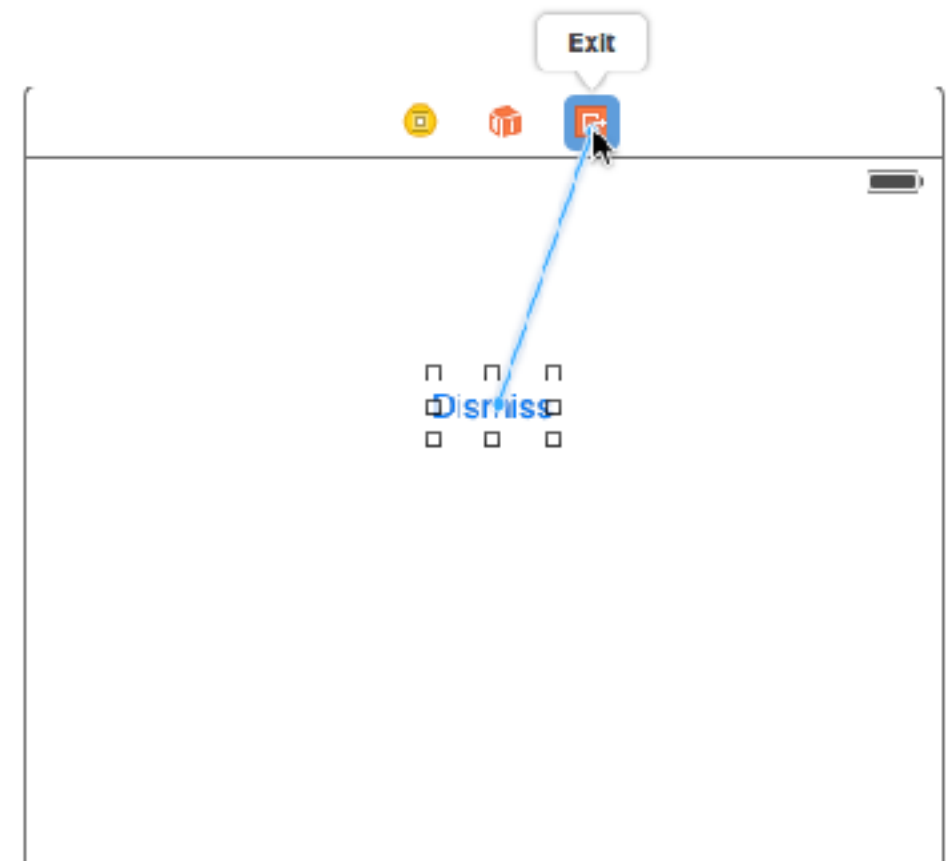
# Unwind Segue (돌아가기)

- Unwind Segue의 사용방법은 다음과 같다.

1. 기존 ViewController file을 선택한다.
2. 파일내부에 Unwind Segue작성을 위한 Method생성

– (IBAction)myUnwindAction:(UIStoryboardSegue\*)unwindSegue

3. Storyboard파일로 이동 후 Unwind Segue Event를 실행할 대상 UIViewController 선택 후, 이벤트 액션과 Exit버튼을 연결한다.
4. 2번에 작성한 Method를 선택한다.

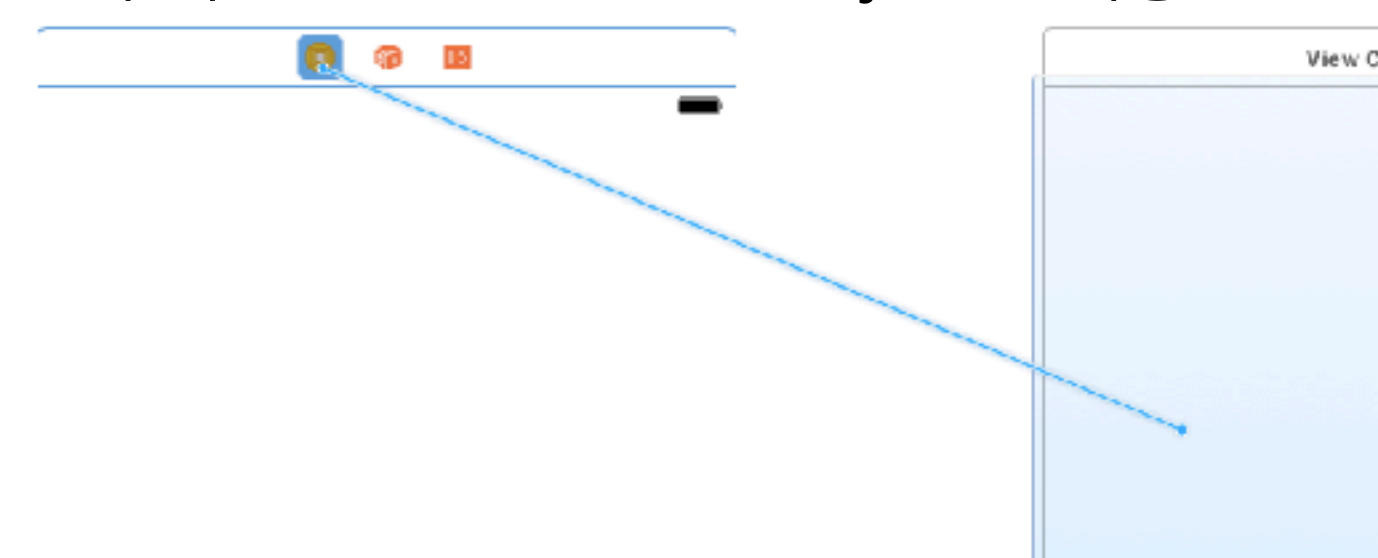


# Manual Segue (수동 Segue)

---

- Segue 생성 후 실행을 코드로 실행하기 위해 사용하는 Segue

1. Storyboard의 기존 ViewController object를 이동될 ViewController에 연결한다.



2. Segue Instance 선택 후 Identifier를 설정한다.
3. 기존 ViewController file로 이동해서 Event Method에서 `performSegue(withIdentifier: String, sender: Any?)` 메소드를 실행한다.

```
performSegue(withIdentifier: "id", sender: self)
```

---

# UINavigationController

---

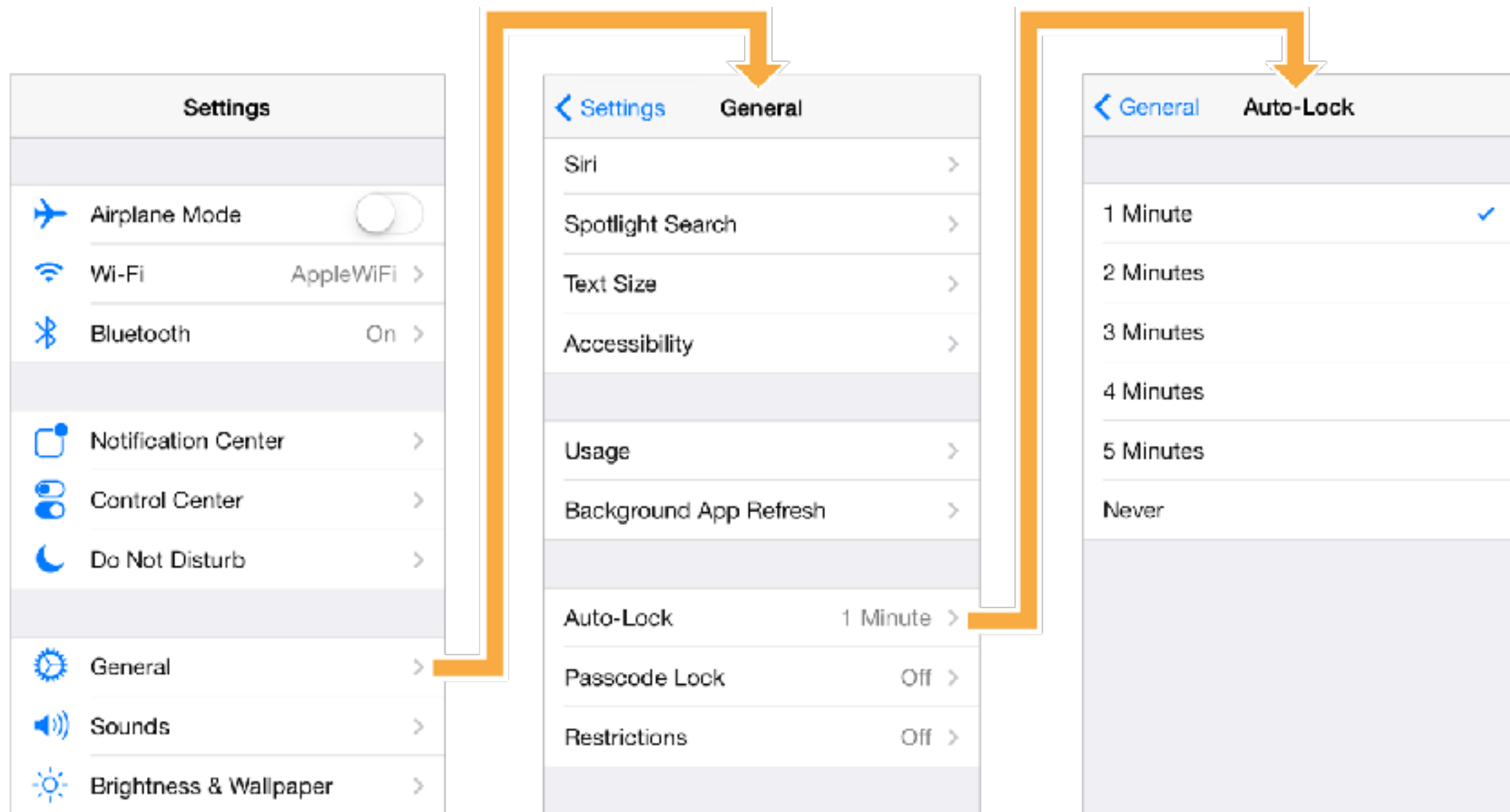
# Container View Controller

---

- View Controller의 Container역할을 하는 View Controller
- **View Controller 간의 구조를 잡는 역할을 한다.**
- 일반적으로 Root View를 가지고 있지 않고, View Controller를 Sub View Controller로 가지고 있다.
- 종류
  1. UINavigationController
  2. UITabBarController
  3. UISplitViewController

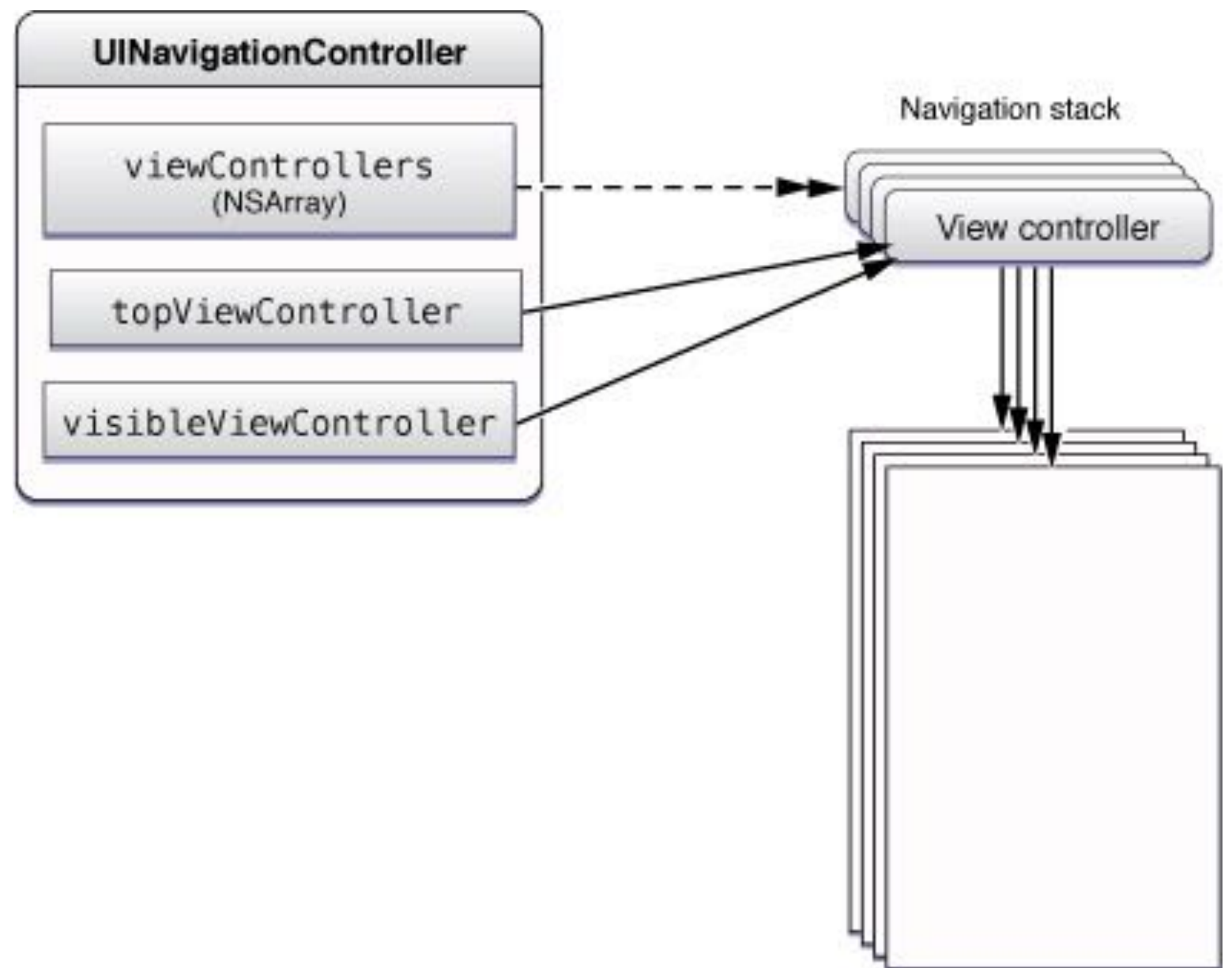
# UINavigationController

---



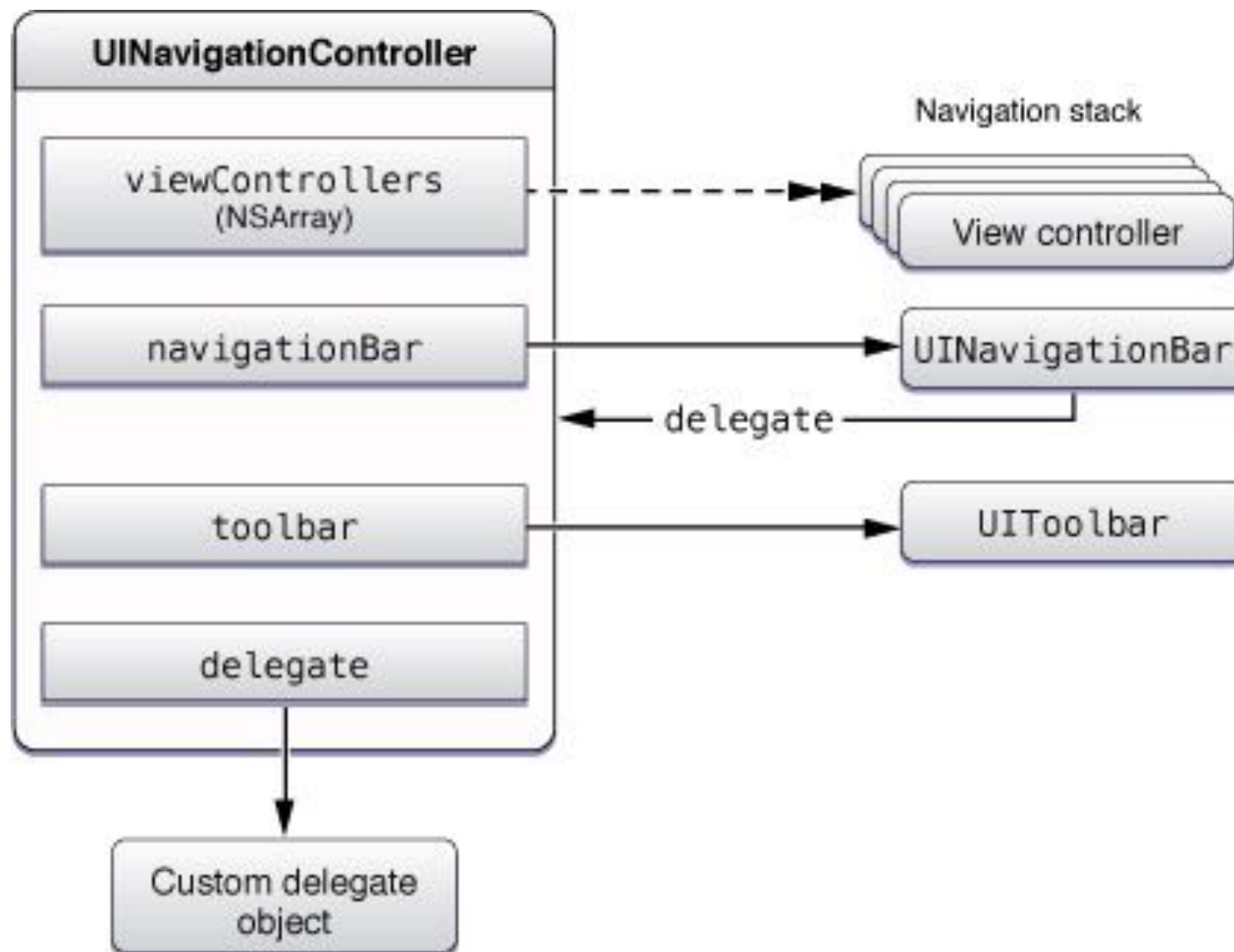
# UINavigationController

- UINavigationController Class
- 네비게이션 컨트롤러는 ViewController들을 스택구조로 관리하는 컨테이너 ViewController이다.



# UINavigationController - 구조


- viewControllers(Array)와 navigationBar toolbar로 이루어져 있다.
- navigationBar를 통해 ViewController의 상태를 관리 한다.
- UIToolbar가 제공되며 기본 hidden상태로 존재 한다.






Carrier 2:58 PM


# NaviagtionBar

 fast campus

■ ■ ■

 아이디를 입력하세요.

\_\_\_\_\_

 비밀번호를 입력하세요.

\_\_\_\_\_

Login

Not yet registered? [Create an account](#)

## RootViewController

# NavigationController 생성

---

- Using Storyboard(Emded)
  1. ViewController선택
  2. Editor -> Embed in -> Navigation Controller
- Using Storyboard(Object Selecte)
  1. UINavigationController Drag and Drop
  2. RootViewController 설정

# NavigationController 생성

---

- Code

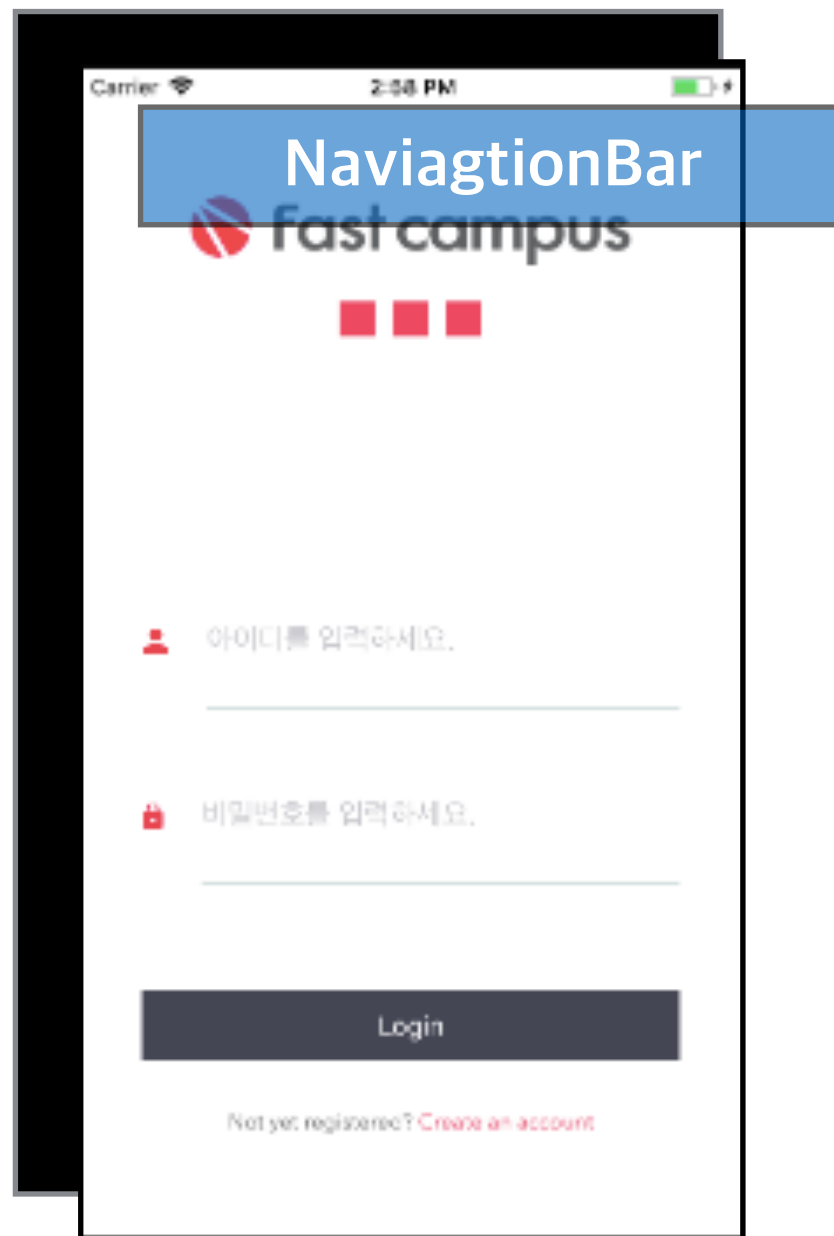
```
let vc = UIViewController()
```

```
let naviVC = UINavigationController(rootViewController: vc)
```

- ViewController instance만드는 코드는 생성과정에 따라 달라집니다.
- rootViewController는 꼭 초기화시 선택해야 하며, 그 후 변경이 불가능합니다. (rootViewController 프로퍼티 존재 X)
- UINavigationController 구조는 보통 AppDelegate에서 생성하나, 앱의 구조에 따라 중간에도 진입할수 있습니다.
- UINavigationController 구조를 중첩해서 만들수 없습니다. (UX가이드에 따라)

# NavigationController 화면 전환

UINavigationController



RootViewController

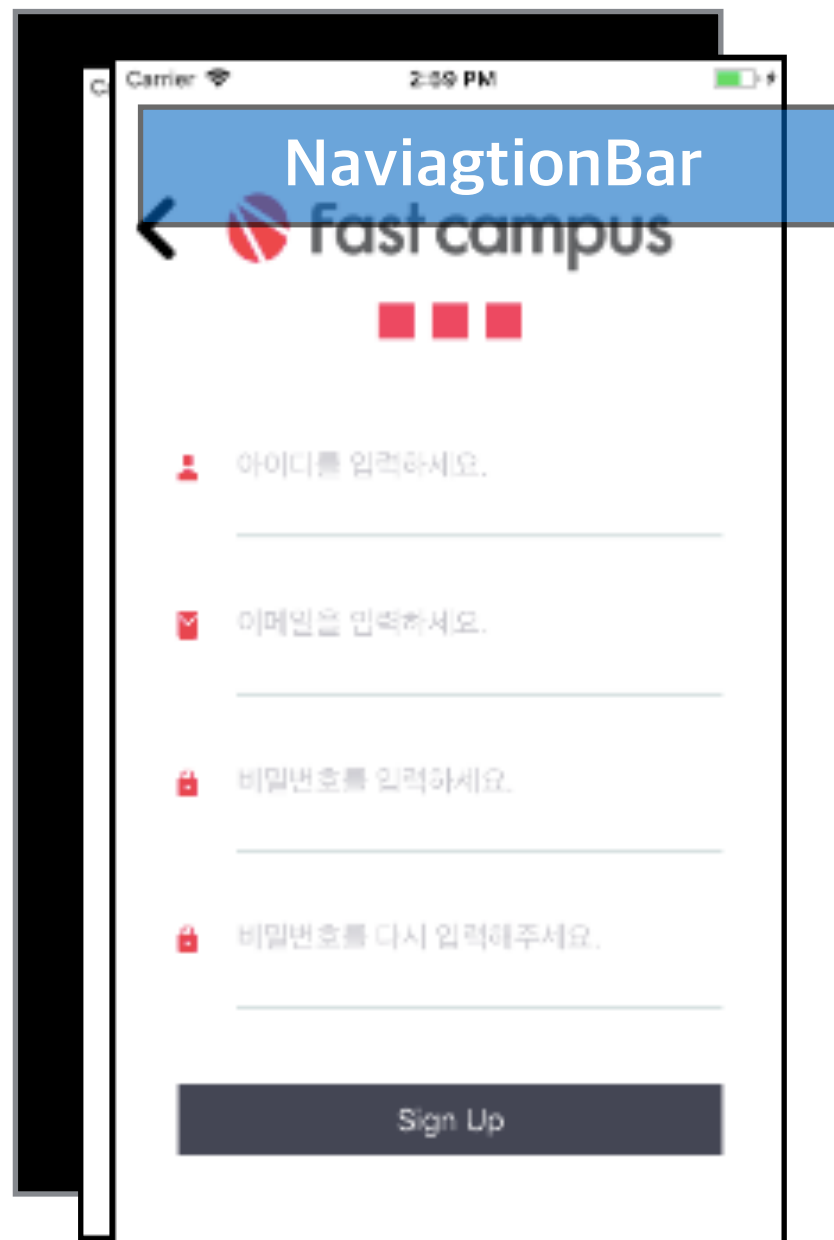
Push



# NavigationController 화면전환

---

UINavigationController



RootViewController



# NavigationController 화면전환

---

- Segue

1. Storyboard의 Segue를 통해 이동가능 (Show = Push)
2. pop의 경우 unwind Segue를 사용해서 돌아간다.

- Coed

1. UINavigationController의 navigationController 프로퍼티를 사용해서 속해있는 navigationController에 접근 가능하다.

```
open var navigationController: UINavigationController? { get }
```

2. .pushViewController()와 .popViewController()를 통해 화면 전환 컨트롤 가능하다.

```
open func pushViewController(_ viewController: UIViewController, animated: Bool)
```

```
open func popViewController(animated: Bool) -> UIViewController?
```

# NavigationController 실습

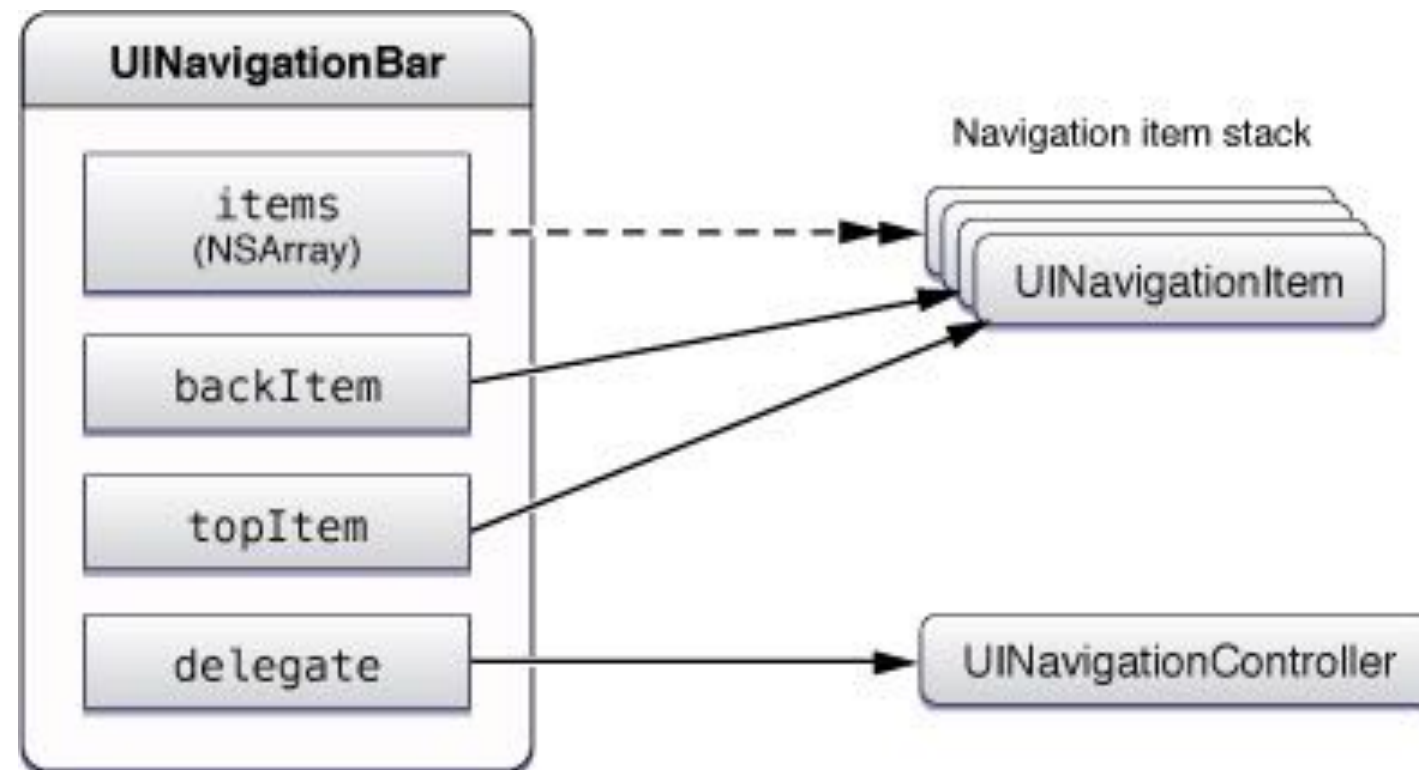
---

- Navigation Controller 기본 프로젝트 만들기
- Navigation Controller 화면 전환
- View Life Cycle 알아보기

# NavigationBar

---

- 네비게이션 인터페이스를 관리 하는 뷰
- navigationBar의 외관은 customize할수 있지만 frame, bounds, or alpha values는 절대 직접 바꿀 수 없다.
- Bar의 높이는  $20(\text{StatusBar height}) + 44(\text{Base NavigationBar Height})$ 이다.
- pop이 가능한 ViewController가 있을시 기본 BackItem이 나타난다.



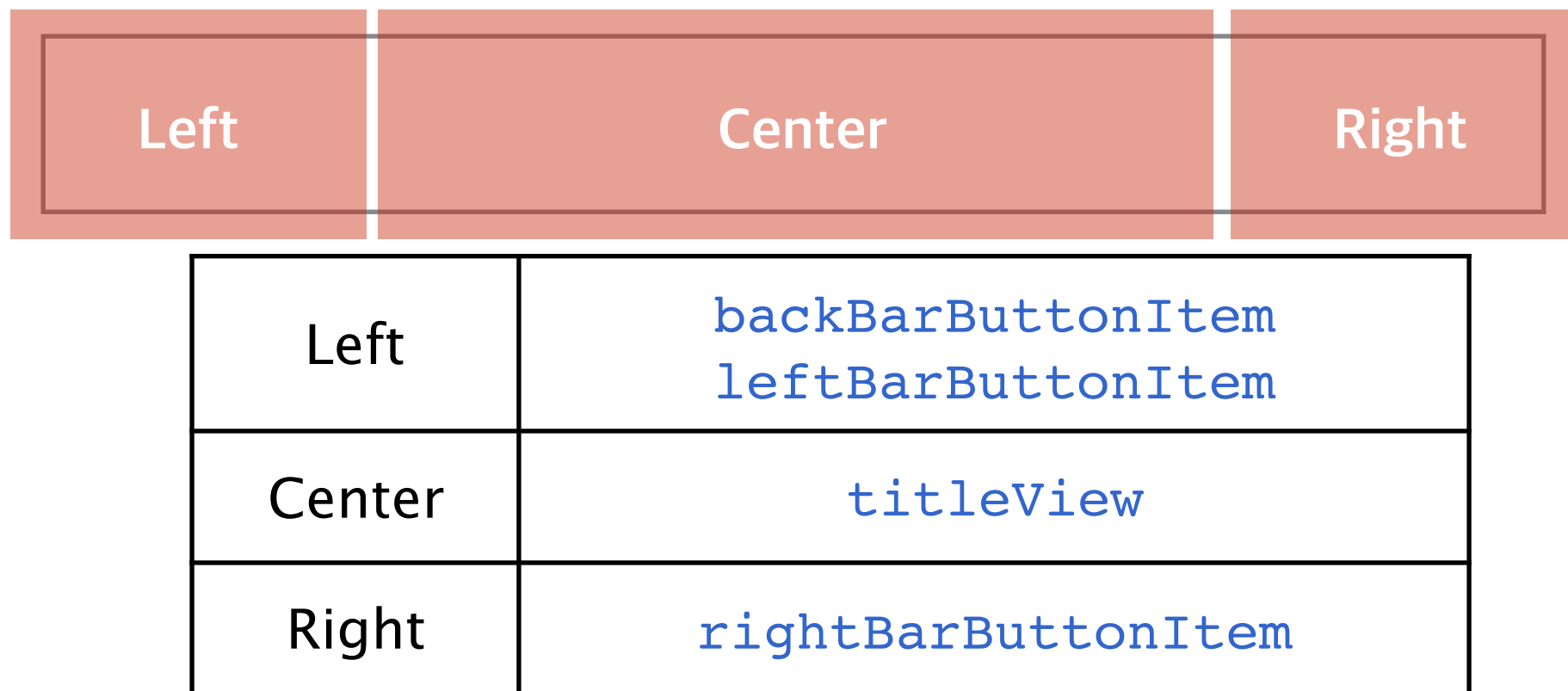


# UINavigationController

---

- UINavigationController의 각 ViewController를 관리하기 위해 만들어지는 인스턴스이다.
- navigation에 push시 자동으로 생성된다.
- 각 아이템은 UIBarButtonItem Type의 인스턴스를 할당 받을수 있다.

```
var navigationItem: UINavigationController { get }
```



# UINavigationController

---

- UINavigationController Instance 생성 후 각 프로퍼티에 할당 한다.
- UINavigationController Instance 생성 방법은 상황에 따라 다를수 있다.
- UINavigationController을 사용하지 않으려면, UINavigationController를 Hidden시켜야 한다.

```
self.navigationController?.isNavigationBarHidden = false
```

```
//Left
self.navigationItem.backBarButtonItem = UIBarButtonItem()
self.navigationItem.leftBarButtonItem = UIBarButtonItem()
self.navigationItem.leftBarButtonItems = [UIBarButtonItem(),UIBarButtonItem()]
//Center
self.navigationItem.titleView = UIView()
//Right
self.navigationItem.rightBarButtonItem = UIBarButtonItem()
self.navigationItem.rightBarButtonItems = [UIBarButtonItem(),UIBarButtonItem()]
```

# UIBarButtonItem

---

//커스텀 이미지를 통해 버튼 생성

```
init(image: UIImage?, style: UIBarButtonItemStyle, target: Any?,  
      action: Selector?)
```

//커스텀 타이틀을 통해 버튼 생성

```
init(title: String?, style: UIBarButtonItemStyle, target: Any?,  
      action: Selector?)
```

//시스템에서 제공하는 기본버튼 생성

```
init(barButtonSystemItem systemItem: UIBarButtonSystemItem, target: Any?,  
      action: Selector?)
```

//커스텀 뷰를 이용해서 버튼 생성 (가장 많이 사용된다)

```
init(customView: UIView)
```

# UIBarButtonItem 예제

---

```
let barBtn1 = UIBarButtonItem(image: UIImage(named:"name"), style: .plain,  
target: self, action: #selector(self.barItemClick(_:)))
```

```
let barBtn2 = UIBarButtonItem(title: "OK", style: .plain, target: self,  
action: #selector(self.barItemClick(_:)))
```

```
let barBtn3 = UIBarButtonItem(barButtonItemSystemItem: .add, target: self,  
action: #selector(self.barItemClick(_:)))
```

```
let btn:UIButton = UIButton()
```

```
//btn 속성 변경
```

```
let barBtn4 = UIBarButtonItem(customView: btn)
```