

Swift Keywords



IBOutlet(아울렛)

아울렛은 IBOutlet 키워드를 사용하여 선언하는 인스턴스 변수들

역할

컨트롤러 헤더 파일에 선언한 객체를 인터페이스 빌더가 알아볼 수 있도록 해준다.

*xib파일 안의 객체와 연결을 하고자 하는 모든 인스턴스 변수들은 IBOutlet 키워드로 다음과 같은 형식으로 선언 되어야 한다.

*@property (nonatomic, retain) IBOutlet UILabel *newLabel;*

아울렛으로 선언된 인스턴스 변수는 프로젝트가 빌드되면 전처리기 번역 > 컴파일러에게 전달 (xib파일과 연결된 객체라는 사실)

프로젝트 실행과 관련 컴파일에는 아무런 영향을 미치지 않지만, 이것이 없다면 인터페이스 빌더는 어떻게 객체들과 연결해야 할지 모른 채 빌드를 실행하게 됨

IBAction(액션)

액션은 IBOutlet과 마찬가지로 컨트롤러 헤더파일에서 하나의 메소드 형태로 선언되어 역할을 하는 변수
IBAction이 선언되면 이 메소드가 액션 메소드라는 것을 인터페이스 빌더에게 알려주고, 컨트롤러를 통해서 호출이 가능해진다.

(IBAction)newAction:(id)sender;

메소드의 형식을 갖는 IBAction은 void를 리턴 타입으로 가진다. (*액션 메소드는 변수값을 리턴하지 않음)
액션 메소드는 sender라는 하나의 인자값을 가지게 되는데, 이것은 sender라는 이름의 id 타입으로 정의되고, 포인터 값이 전달된다.
동일한 액션 메소드를 호출하는 데 있어 어떤 액션을 통해 메소드를 호출하였는 지 구분하는 구분자의 역할을 하게 된다.

(만약 버튼이 하나밖에 없는 것 처럼, 액션의 구분이 필요하지 않다면 뒷부분의 'id(sender)' 를 제거하고 작성하면 가능,
But 키워드를 제거하게 된다면 모든 버튼이 동일한 역할을 수행)

Storyboard

스토리보드는 그래픽 유저 인터페이스를 위한 메뉴구성 툴 ‘인터페이스 빌더’에서 시작되었다.

nib(nextstep interface builder), xib(xcode interface builder)와 같은 포맷 위에서 iOS SDK 5부터 새롭게 도입된 개념

스토리보드와 xib의 가장 큰 차이점은, xib와 달리 스토리보드에서는 ‘실행 흐름’을 제어할 수 있다.

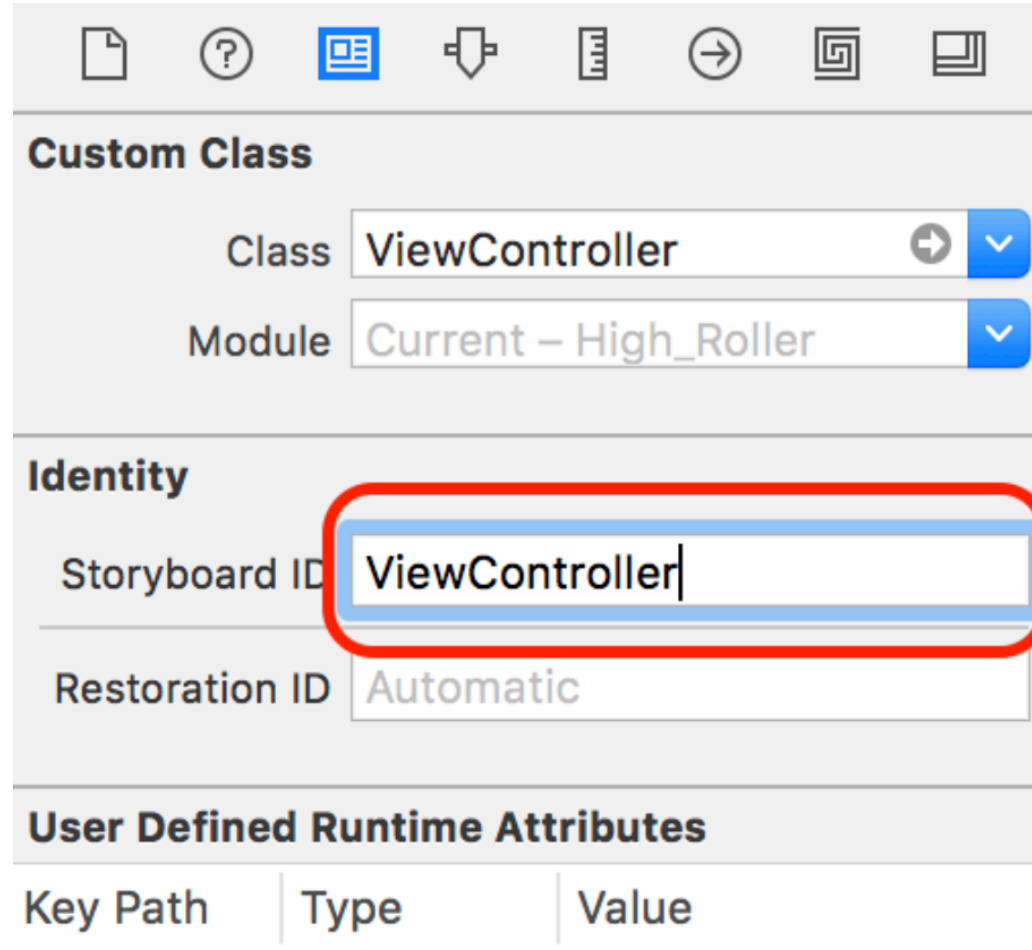
실행 흐름이란, A화면에서 B화면으로 넘어가는 것, 즉 어떤 동작에 따라 화면이 전환되는 것을 말한다.

장점

- 장면에서 모든 보기 컨트롤러를 시작적으로 배치하고 그 사이의 연결을 설명할 수 있다. (*앱의 모든 장면을 더 개념적으로 볼 수 있음)
- 스토리보드는 다양한 장면 간의 전환을 설명할 수 있다. 이러한 전환을 ‘단편(segues)’라고 하며 스토리보드에 바로보기 컨트롤러를 연결하여 단절을 만들어 낸다. (*segues를 통해 UI를 관리하는 데 필요한 코드를 줄일 수 있음)
- 스토리보드를 사용하면 프로토 타입 셀 및 정적 셀 기능을 사용하여 테이블 뷰 작업을 쉽게 수행할 수 있다. (*코드의 양을 줄여 뷰를 거의 완벽히 디자인)
- 스토리보드를 사용하면 위치와 크기를 정의하는 요소간에 수학적 관계를 정의할 수 있는 자동 레이아웃들을 쉽게 사용할 수 있다.
(*다양한 화면 크기 및 크기의 장치를 훨씬 쉽게 처리)

Storyboard ID

각 스토리보드의 명칭



The screenshot shows the Xcode interface for editing a storyboard element. The 'Custom Class' section has 'Class' set to 'ViewController' and 'Module' set to 'Current - High_Roller'. The 'Identity' section has 'Storyboard ID' set to 'ViewController' (highlighted with a red rounded rectangle) and 'Restoration ID' set to 'Automatic'. The 'User Defined Runtime Attributes' section is visible at the bottom with columns for 'Key Path', 'Type', and 'Value'.

Key Path	Type	Value
----------	------	-------

If ~else

If 다음에 오는 조건문의 참, 거짓에 따라 뒤따라오는 코드블록을 실행할지 안 할지 분기를 나눈다.

```
if ([조건문1], [조건문2], ...) {  
    // to do 1  
}  
else if ([조건문3], [조건문4], ...) {  
    // to do 2  
}  
else {  
    // to do 3  
}
```

1, to do 2가 있는 부분에 들어갈 코드가 2줄 이상일 때는 반드시 }로 묶어줘야 한다.

논리적으로 조건문을 통과하지 못했을 때 to do 2, to do 3에서 수행할 일이 없다면 else if/else 블록을 생략해도 된다.

여러 개의 조건문을 구분할 수 있다는 특징이 있다. c++에서 && 연산자로 나열하던 것을 ,로 나열할 수도 있다.

또한 &&로 나열해도 된다. 조건문에서 optional 변수를 바인딩하여 검사할 수 있다.

Optional(옵셔널)

Optional의 뜻은 ‘선택적인’이라는 뜻을 가지고 있다.

코딩 시 어떠한 변수에 값이 있을 수도 없을 수도 없는 경우를 위해 사용되고 있다.

? / ! 은 optional의 기호, 기호를 타입 어노테이션 옆에 붙여줘야 사용할 수 있다.

Swift에서 기본적으로 변수를 선언할 때 non-optional인 값을 주어야 한다.

Int형으로 선언한 변수는 무조건 정수의 타입으로 들어가야 한다.

```
var test : Int?  
test = nil
```

?

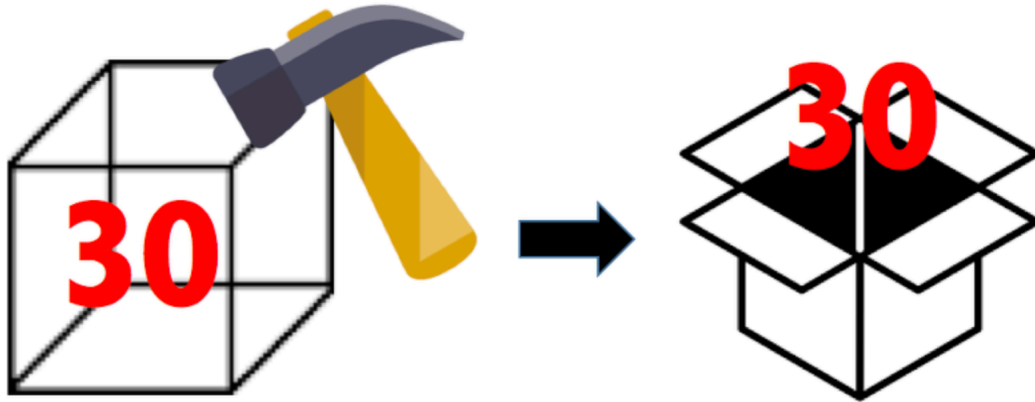
```
var someValue : Int? = 30  
var Value = someValue
```



Value는 optional 타입이고, Int데이터형을 가질 수 있는 변수
(=Int형 값을 가질 수도, 안가질 수도 있어)

! (Unwrapping)

```
var someValue : Int? = 30  
var Value : int = someValue!
```



마치 상자를 깨부시고 값을 꺼내, 그 값(정수)를 Value에 넣어준 코드이기 때문에 Value에서는 아무런 문제도 없다.
상자 안에 값이 없을 경우에는 항상 옵셔널 값이 nil이 아니라는 것을 명시해야 한다.

문자열 템플릿

Swift에 문자열을 상수로 저장하여 재사용하고 변수를 삽입할 수 있다.

let profile = “\ (name)님 \ (birlrthday)년 “ \ (변수나 상수)”를 넣으면 문자열 템플릿

```
var template = "Your name is {0} and your age is {1}."
someLabel.text = string.Format(template, "John", 35)
some2Label.text = string.Format(template, "Jane", 33)
```



```
var template = "Your name is %@ and your age is %d."
someLabel.text = String(format: template, "John", 35)
some2Label.text = String(format: template, "Jane", 33)
```

> % @ : 문자열 (또는 nhgrif가 지적한대로 NSObject의 description /
descriptionWithLocale 속성)
> % d : Int
> % f : 두 배
> % .2f : 정밀도 2 인 Double, 예 : 2.2342 -> 2.23

Data Pass

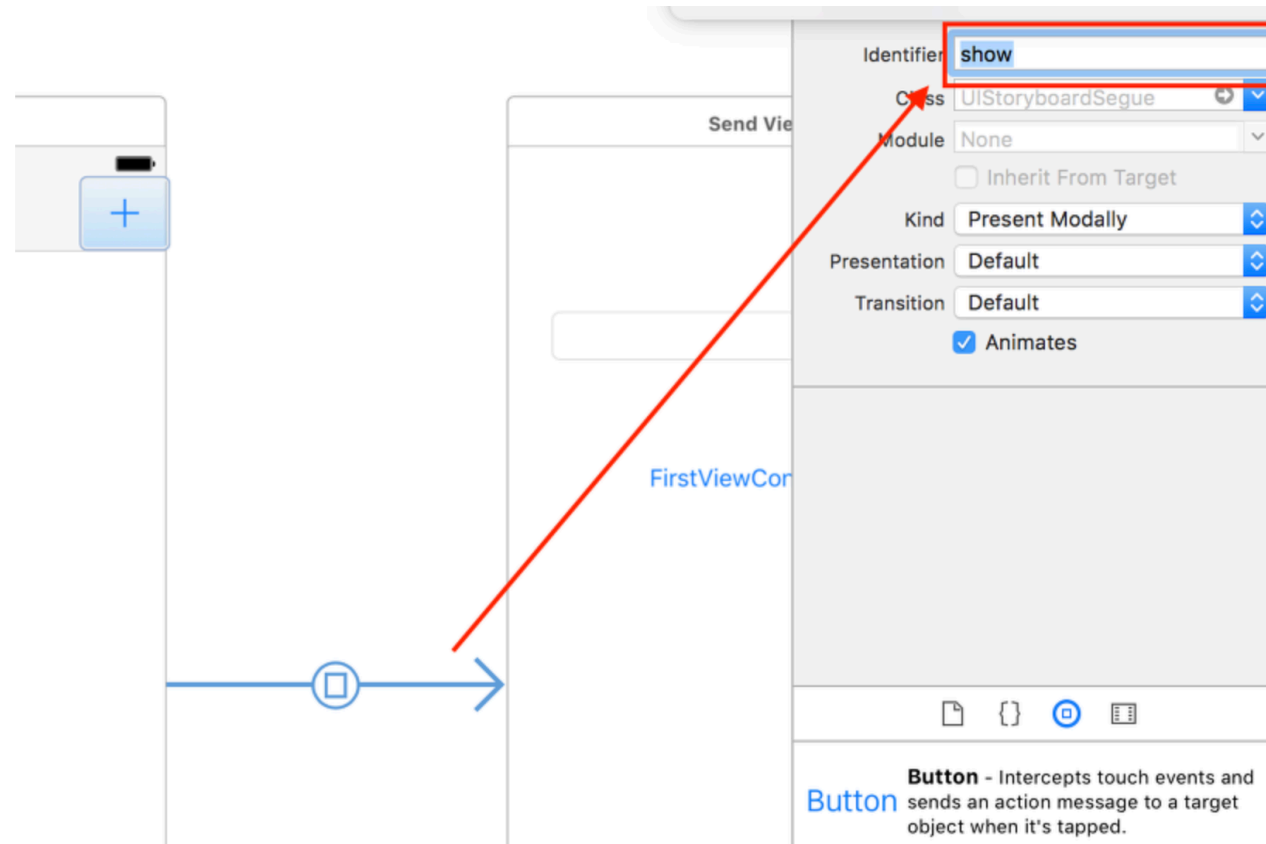
Delegate를 통해 ViewController간 데이터를 전달하는 것을 의미한다.
보통 ViewController간 데이터 전달은 prepare 혹은 IBAction에서 진행한다.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if segue.identifier == "segue name"{  
        let PlaySoundVC = segue.destination as! PlaySoundsViewControll  
        PlaySoundVC.recordedAudioURL = recordedAudioURL  
    }  
}
```

두 개의 ViewController의 상관관계가 깊어지고, 한 ViewController에서의 변화가 다른 ViewController에게 영향을 줄 수 있다.

Data Pass

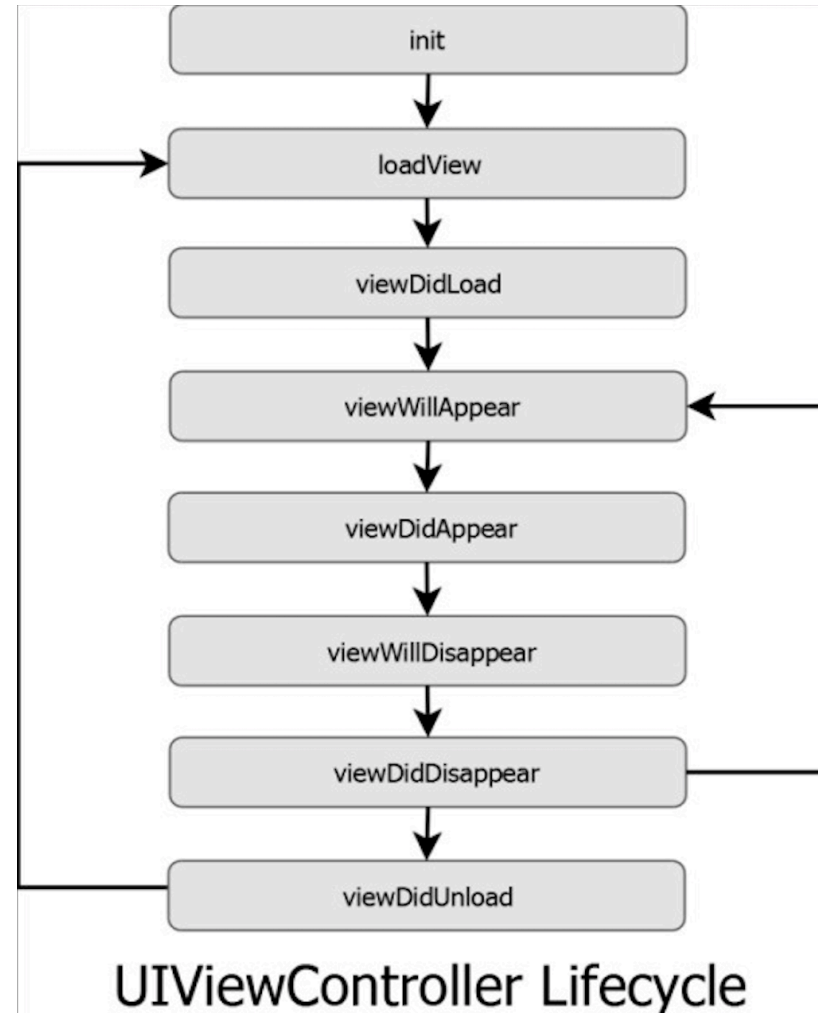
Segue에 identifier를 지정해 Setting작업을 할 수 있다.



Life cycle iOS

ViewController들은 생명주기로 이루어져 있으며,
예시 이미지와 같은 5개의 생명주기로 순환한다.

한 마디로 보여졌다 사라지는 주기를 생명주기라 한다.



present

UIViewController에 정의된 메소드로 기존 뷰 컨트롤러 위에 새로운 뷰 컨트롤러를 덮어 씌워 참조하는 관계를 형성하는 역할을 한다.

ViewController에 다음 view로 넘어가는 버튼을 IBAction으로 연결해 사용할 수 있다.

```
let ViewController인스턴스 =  
self.storyboard?.instantiateViewController(withIdentifier: "Storyboard ID값")  
  
ViewController인스턴스?.modalTransitionStyle =  
UIModalTransitionStyle.coverVertical  
  
self.present(ViewController인스턴스!, animated: true, completion: nil)
```