

## UIViewController(뷰를 관리)

- 앱의 기초가 되는 내부 구조
- 모든 앱은 적어도 한개 이상의 뷰 컨트롤러를 가지고 있으며 대부분의 앱은 많은 뷰컨트롤러로 이뤄져있다.
- 뷰컨트롤러는 사용자의 인터렉션(터치, 드래그동작)과 앱의 데이터 사이에서 컨트롤의 역할을 한다.
- 뷰컨트롤러는 뷰 관리, 이벤트 핸들링, 다른 뷰컨트롤러로의 전환 등의 메소드와 프로퍼티를 가지고 있다.

## UIViewController - Root View

- 뷰의 계층을 관리하는 기능
- 모든 뷰컨트롤러는 한개의 root view를 가지고 있다.
- 화면에 표시하기 위해서는 모든 뷰는 root view의 계층안에 있어야 한다.

## UserInteraction

- 뷰컨트롤러는 UIResponder를 상속받은 객체로 이벤트 체인으로부터 오는 이벤트는 효과적으로 처리한다.
- 즉 사용자의 모든 이벤트는 ViewController가 받아서 각 뷰에서 처리되는 Action Method와 Delegate에서 처리된다.

## Child UIView Controller

- 뷰컨트롤러는 다른 뷰컨트롤러를 child 뷰컨트롤러로 관리 할수있다.
- child 뷰컨트롤러의 root view를 자신의 root view에 addSubview하여 화면에 표시 가능하다.

## Data Marshaling

- 뷰 컨트롤러는 자신이 관리하는 view들과 앱 내부의 데이터와의 중계자 역할을 한다.

## Resource Management

- 뷰컨트롤러 안에 있는 모든 뷰나 객체는 모두 뷰컨트롤러의 책임이다.
- 앱 사용중 메모리가 부족할때 didReceiveMemoryWarning 메서드가 불리며, 오래동안 사용하지 않은 객체와 다시 쉽게 만들수 있는 객체를 제거할수있다.

## 다양한 ViewController

- General View Controller
  - UIViewController
  - UITableViewController
  - UICollectionViewController
- Container View Controller
  - UINavigationController
  - UITabBarController
  - UISplitViewController

## General View Controller

- 일반적인 View Controller형태
- 각 View Controller가 Root View를 가지고 있다.
  - UIViewController Root View = UIView
  - UITableViewController Root View = UITableView
  - UICollectionViewController = UICollectionView

## UIViewController Object Load

- Using Storyboard

```
//Main이라는 스토리보드를 가져옴
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"Main" bundle:nil];

//스토리보드 id가 viewController인 스토리보드를 vc1으로 가져옴
UIViewController *vc1 = [storyboard instantiateViewControllerWithIdentifier:@"ViewController"];
```

- Using xib

```
UIViewControllerWithXib *vcxib = [[UIViewControllerWithXib alloc] initWithNibName:@"ViewControllerWithXib" bundle:nil];
```

## 화면전환 - Present Modally

- Present Modally ViewController간의 화면 전환

## 화면전환순서 - 코드

- NextViewController 인스턴스화 / 첫번째
- PresentViewController / 두번째
- dismissViewController / 세번째

## Animation - transition

- animation에 따라 다른 형태의 느낌을 주기도 한다.
- 4가지의 화면전환 애니메이션이 있다.

```
//화면전환 애니메이션
[vc2 setModalTransitionStyle:UIModalTransitionStyleCrossDissolve];
```

## 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
  - UINavigationController
  - UITabBarController
  - UISplitViewController

## NavigationBar

- 네비게이션 인터페이스를 관리 하는 뷰
- navigation bar의 외관은 customize할수있다. 하지만
- frame, bounds, or alpha values는 절대 직접 바꿀 수 없다.
- Left / backbarbuttonitem, leftbarbuttonitem
- Center / titleview
- Right / rightbarbuttonitem

## ToolBar

- 뷰컨트롤러의 부가적 기능을 위해 ToolBar를 활용할수있다.
- 기본 옵션은 Hidden이다(toolbarHidden = NO)

## TabBar

- 각각의 ViewController를 각 TabBar item에 매칭 되어 있다.
- TabBarItem은 이미지와 텍스트를 넣을 수있다.
- 각 TabBarItem은 해당 viewController의 tabbaritem 프로퍼티로 접근 가능하다.

## Alert Controller

- Alert Controller종류로는 ActionSheet와 Alert가 있다.
- Action / cancel은 1개만 사용할수있다.
- UIAlertController







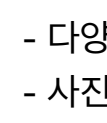
```
//앨럿액션 타이틀:취소, 스타일:cancel, 핸들러 블록사용
UIAlertAction *cancle = [UIAlertAction actionWithTitle:@"취소"
                                                    style:UIAlertActionStyleCancel
                                                    handler:handlerBlock];

//앨럿액션 타이틀:확인, 스타일:default, 핸들러 블록사용
UIAlertAction *ok = [UIAlertAction actionWithTitle:@"확인"
                                                    style:UIAlertActionStyleDefault
                                                    handler:handlerBlock];

//앨럿액션 타이틀:파괴, 스타일:destructive, 핸들러 블록사용
UIAlertAction *delete = [UIAlertAction actionWithTitle:@"파괴"
                                                    style:UIAlertActionStyleDestructive
                                                    handler:handlerBlock];
```

## Gesture Recognizer

- 뷰위에 올라간다.
- UIGestureRecognizer
  - 사용자의 입력을 전달받을 수 있는 방법을 제공
  - tap, pinch, rotation, swipe, pan(drag), edge pan, long press 등을 인지하는 각각의 서브클래스 존재
  - view위에 얹어 액션을 핸들링

	<b>Tap Gesture Recognizer</b> - Recognizes tap gestures, including double-tap or multiple-touch.
	<b>Pinch Gesture Recognizer</b> - Recognizes pinch gestures.
	<b>Rotation Gesture Recognizer</b> - Recognizes rotation gestures.
	<b>Swipe Gesture Recognizer</b> - Recognizes swipe gestures.
	<b>Pan Gesture Recognizer</b> - Recognizes pan (dragging) gestures.
	<b>Screen Edge Pan Gesture Recognizer</b> - Recognizes pan (dragging) gestures that start ne...
	<b>Long Press Gesture Recognizer</b> - Recognizes long press gestures, based on the nu...
	<b>Custom Gesture Recognizer</b> - Recognizes custom gestures. Set a custom subclass in the Identity...

## Image Picker

- UIImagePickerController
  - 다양한 소스로부터 사진을 가져오는 기능이 구현되어있는 클래스
  - 사진소스에는 라이브러리, 사진앨범, 카메라 등이 있다.
  - 옵션을 주어 사진을 가져올때 사용자에게 정사각형으로 편집할 수 있도록 할 수있다.

```
- image picker + delegate
4 //이미지피커를 띄우기 위한 메서드
5 -(void)showImagePicker:(UIImagePickerControllerSourceType)sourceType
6 {
7     UIImagePickerController *picker = [[UIImagePickerController alloc]init];
8     [picker setDelegate:self]; //델리게이트 연결
9     [picker setSourceType:sourceType];
0     [picker setAllowsEditing:YES]; //편집사용
1     //presentviewController 화면을 올려준다.
2     [self presentViewController:picker animated:YES completion:nil];
3 }
```

## User Defaults

- NSUserDefaults
  - 관리할 수 있는 데이터의 종류
    - float, double, NSInteger, BOOL
    - NSURL, NSData, NSNumber, NSDate
    - NSString, NSArray, NSDictionary
  - 사용자의 환경설정 내용을 반영구적으로 저장하기위해 사용
  - 주로 standardUserDefaults라는 프로퍼티를 사용하여 싱글턴 객체로 사용
  - suiteName을 통한 생성자 메서드를 통해 별도의 분리된 User Default 생성 가능
  - synchronize라는 메소드를 통해 기존의 데이터와 병합
    - 카카오톡의 배경화면이나, 글씨체, 채팅방스타일등 기본정보

```
//이진 데이터
//jpeg데이터를 그냥랑 그대로 가져온다, (1.0, 용량 그래로 가져온다는말)
NSData *imageData = UIImageJPEGRepresentation(pickedImage, 1.0);
```

```
//싱글턴, 한번지정해주면 어디에서든 똑같이 쓸수가 있다.
NSUserDefaults *userDefault = [NSUserDefaults standardUserDefaults];
```

```
//NSUserDefaults가 딕셔너리 형식으로 되어있기 때문에 setObject와 forKey로 불러온다.
[userDefault setObject:imageData forKey:@"imageData"];
```

```
//동기화
[userDefault synchronize];
```

## Singleton

- 애플리케이션 전 영역의 걸쳐 **하나의 클래스가 단 하나의 인스턴스만(객체)**을 생성하는 디자인 패턴
- 애플리케이션 내부에서 유일하게 하나만 필요한 객체에서 사용 (셋팅, 데이터등)
- 클래스 메서드로 객체를 만들며 static을 이용하여 단 1개의 인스턴스만 생성
- 애플리케이션 전역에서 공유하는 객체 생성 가능

```
//스크린 정보를 가지고 있는 객체
UIScreen *screen = [UIScreen mainScreen];
```

```
//사용자 정보를 저장하는 객체
NSUserDefaults *data = [NSUserDefaults standardUserDefaults];
```

```
//애플리케이션 객체
UIApplication *app = [UIApplication sharedApplication];
```

```
//파일 시스템 정보를 가지고 있는 객체
NSFileManager *filemanager = [NSFileManager defaultManager];
```

## State Restoration

- app delegate

```
-(BOOL)application:(UIApplication *)application shouldSaveApplicationState:(NSCoder *)coder
{
    return YES;
}

-(BOOL)application:(UIApplication *)application shouldRestoreApplicationState:(NSCoder *)coder
{
    return YES;
}
```
- 각각의 view controller

```
//인코딩
-(void)encodeRestorableStateWithCoder:(NSCoder *)coder
{
    [super encodeRestorableStateWithCoder:coder];

    [coder encodeObject:self.OrangeTextField,text forKey:UserInputText];
}

//디코딩
-(void)decodeRestorableStateWithCoder:(NSCoder *)coder
{
    [super decodeRestorableStateWithCoder:coder];

    NSString *text = [coder decodeObjectForKey:UserInputText];
    self.userInputText = text;
}
```

- 스토리보드

- Restoration Idetifier 필요

<b>Identity</b>	
Storyboard ID	<input type="text" value="aqua"/>
Restoration ID	<input type="text" value="aqua"/>
<input checked="" type="checkbox"/> Use Storyboard ID	