UIViewController Class Reference

https://developer.apple.com/library//ios/documentation/UIKit/Reference/UIViewController_Class/index.html
https://developer.apple.com/documentation/uikit/uiviewcontroller

Overview

1.View Controller main responsibilities:

- Updating contents of the views, changing underlying data.
- Responding to user interactions with views.
- Resizing views & managing layout of overall interface.

2.View Controller are UIResponder objects

Subclassing Notes

View Management

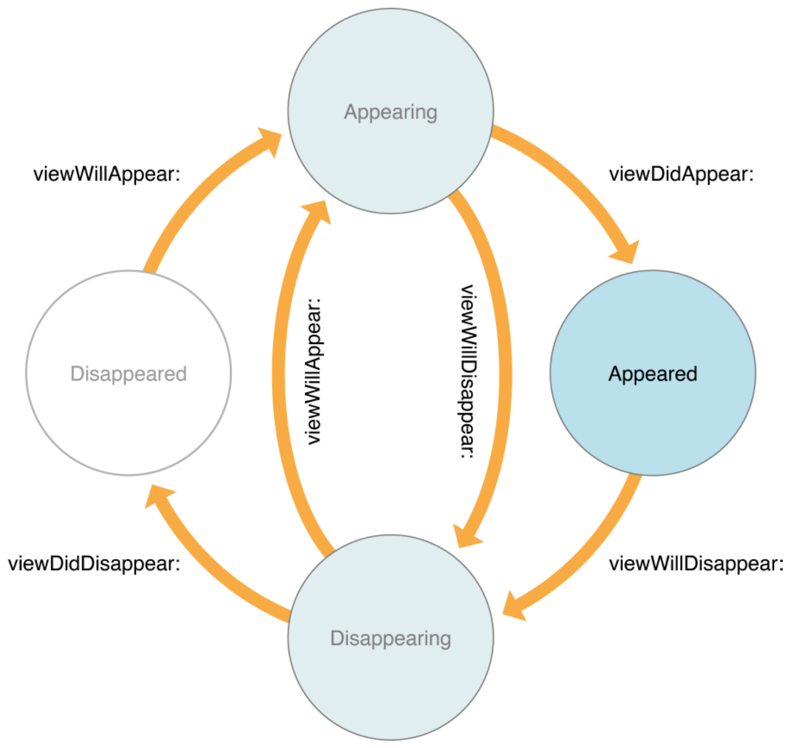
1. Root view is stored in view property of this class.
2. View controllers load their views lazily.

- storboard. To load a view controller from a storyboard, call (UIStoryboard) the instantiateViewController(withIdentifier:)
- using nib file
- loadView()

Note: storyboard & nib automatically gets its own copy of these views. However, manually each view controller must have its own unique set of views. could not share views between view controllers.

3. Auto Layout Guide.

Handling View-Related Notifications



Handling View Rotations

In iOS 8, all rotation-related methods are deprecated. Instead, rotations are treated as a change in the size of the view controller's view and are therefore reported using the viewWillTransitionToSize:withTransitionCoordinator: method.
UIKit call this method on window's root view controller, and view controller then notifies its child view controller.

In iOS 6, 7, define interface orientations in your app's info.plist. view controller can override supportedInterfaceOrientations

Implementing Container View Controller

Here are the essential methods you might need to call:

```
addChildViewController:
removeFromParentViewController
willMoveToParentViewController:
didMoveToParentViewController:
```

Memory Management

didReceiveMemoryWarning()

State Preservation & Restoration

view controller's restorationIdentifier property

Old Version

where necessary, a view controller:

resizes and lays out its views

adjusts the contents of the views

acts on behalf of the views when the user interacts with them

A key advantage of storyboards is that they can express relationships between different view controllers in your app.
do not need to directly allocate and instantiate the view controller. Instead, when you need to programmatically instantiate the view controller, you do so by calling the `instantiateViewControllerWithIdentifier:` method on a `UIStoryboard` object.
set the autoresizing properties of your views.

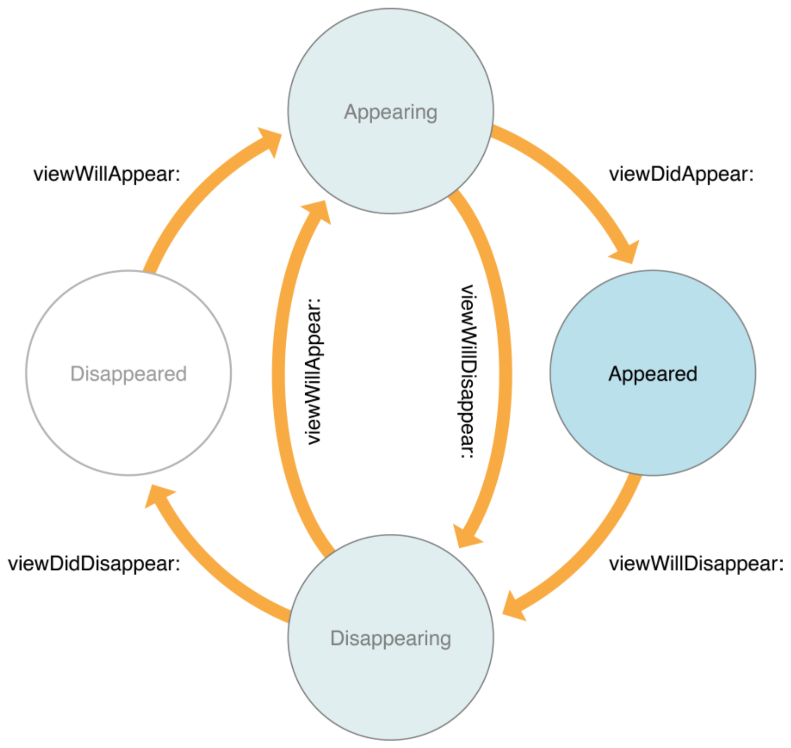
- in Interface Builder using the inspector window or programmatically by modifying the `autoresizesSubviews` and `autoresizingMask` properties of each view.

If your view controller supports auto layout and is a child of another view controller, you should call the view's `setTranslatesAutoresizingMaskIntoConstraints:` method to disable these constraints.

State Preservation and Restoration

If you assign a value to the view controller's `restorationIdentifier` property, the system may ask the view controller to encode itself when the app transitions to the background. When preserved, a view controller preserves the state of any views in its view hierarchy that also have restoration identifiers.

Explaining State Transitions



Difference between loadView and viewDidLoad?

If you are developing applications without using xib LoadView() method is called and if there is an xib then ViewDidLoad method is called So it is better to use LoadView method

```
/*
// Implement loadView to create a view hierarchy programmatically,
without using a nib.
- (void)loadView {
}
*/


/*
// Implement viewDidLoad to do additional setup after loading the view,
typically from a nib.
- (void)viewDidLoad {
[super viewDidLoad];
}
*/
```

Don't read self.view in -loadView. Only *set* it, don't *get* it.

The self.view property accessor *calls* -loadView if the view isn't currently loaded.
There's your infinite recursion.

```
    NSLog(@"---------- follow is alloc and init");
     self.vc = [[HelloViewController alloc] init];    // init, including init and inner
if
    NSLog(@"---------- follow is self.VC.view.frame setting");
     self.vc.view.frame = CGRectZero;     // loadView, viewDidLoad
    NSLog(@"---------- follow is self.window.rootVC = self.vc");
     self.window.rootViewController = self.vc;     // viewWillAppear,
viewDidLayoutSubviews * 2, then viewDidAppear
```

```
    NSLog(@"---------- follow is alloc and init");
     self.vc = [[HelloViewController alloc] init];     // init
    NSLog(@"---------- follow is self.window.rootVC = self.vc");
     self.window.rootViewController = self.vc;     // loadView, viewDidLoad,
viewWillAppear, viewDidLayoutSubviews * 2, then viewDidAppear
```

```
loadView —— without nib/xib 自定义UIViewController的view
            self.view = [UIWebView alloc] initWithFrame:...
viewDidLoad —— addSubview in view, or add data from model/database to view.
               UIButton *button = ;
               [self.view addSubview:button];
```

viewDidUnload —— didReveiveMemoryWarning中，实现：view的superview为nil，释放view，调用viewDidUnload，并且释放相关资源。self.name, self.pwd ...
viewWillAppear —— 可以在此加载一些图片，和一些其他占内存的资源
viewDidLayoutSubviews * 2, then viewDidAppear,
viewWillDisAppear —— 在此将一些占用内存比较大的资源先释放掉，在 viewWillAppear: 中重新加载。如：图片/声音/视频。
http://blog.sina.com.cn/s/blog_76264a170101e5lb.html

二、使用 NavigationController 去 Push 切换显示的View的时候，调用的顺序：

例如 从 A 控制器 Push 显示 B 控制器，

[(A *)self.navigationController pushViewController:B animated:YES]

1. 加载B控制器的View（如果没有的话）；

2. 调用 A 的 - (void)viewWillDisappear:(BOOL)animated；

3. 调用 B 的 - (void)viewWillAppear:(BOOL)animated；

4. 调用 A 的 - (void)viewDidDisappear:(BOOL)animated；

5. 调用 B 的 - (void)viewDidAppear:(BOOL)animated；

总结来说，ViewController 的切换是先调用 隐藏的方法，再调用显示的方法；先调用Will，再调用Did。

三、重新布局View的子View

- (void)viewWillLayoutSubviews

- (void)viewDidLayoutSubviews

看字面意思就知道这两个方法是在View对他的子View进行布局的时候会被调用，包括View 显示/隐藏/屏幕旋转 的时候都会被调用。

如果设计的应用需要支持多方向可以在这里面进行一些UI的横竖屏适配。