**presentViewController：**

**使用场景**

* 模态的展现另一个UIViewController

**基本用法(OC)**

[viewControllerA presentViewController:viewcontrollerB animated:YES completion:^{

NSLog(@"从viewControllerA切换到viewControllerB");

];

注意，这个completion block是当viewControllerB的viewDidAppear方法执行完毕之后，执行的。

**多说点**

1. viewController有两个属性跟这个方法关联，一个是presentedViewController，一个是presentingViewController.就拿上面的例子来说，  
   viewControllerA.presentedViewController = viewControllerB,  
   viewControllerB.presentingViewController = viewControllerA  
   这个关系执行presentViewController时就确定了。
2. 当你需要从B跳转回A时，在B调用以下方法即可  
   [viewControllerB dismissViewControllerAnimation:YES completion:^{  
   NSLog(@"从viewControllerB切回到viewControllerA");  
   ];
3. 如果你接连presentViewController多次呢？比如，A->B->C->D,这个时候会有一个栈，A处于栈底，D处于栈顶。当你需要从D跳转到A时，需要在A中调用dismissViewControllerAnimation:,这样D会执行dismiss animation，而B，C，D都会从占中移除。

**transitionFromViewController**

**使用背景**

* 用于container viewController的两个子view controller之间的切换

**基本用法**

若viewControllerA有两个childViewController， viewControllerB跟viewControllerD，当前viewControllerB的view在父ViewController的View层级中。

[viewControllerA transitionFromViewController:viewControllerB

toViewController:viewControllerA

duration:0.25

opitions:UIViewAnimationOpitionsNone

animations:NO

completion:^{

NSLog(@"viewControllerB从A中移除，viewControllerD在A中");

];

**多说点**

1. 在A中，依次从B转到了C，再转到了D。那么销毁的顺序则是，D，C，B，A，很像栈的顺序。
2. 其实这个方法的就是把当前ViewController的view从父viewController的view层级中移除，添加上to viewController的view。
3. 跟这个方法关联的属性说两个吧。一个是parentViewController，如viewControllerB的parentViewController是viewControllerA。一个是childViewControllers，这个属性是一个数组，包含所有viewControllerA的child viewController。
4. 还有其他两个方法也很重要。一个是addChildViewController，添加child ViewController，如  
   [viewControllerA addChildViewController:viewControllerB];  
   一个是removeFromParentViewController，如：  
   [childViewControllerB removeFromParentViewController];

**UINavigationController**

**使用背景**

UINavigationController是一个很重要的ViewController容器。完全可以单写一篇文章来解析。但是今天只说viewController之间的切换吧。UINavigationController管理了ViewController之间的层级关系，依照栈的模式来工作。可以参照下iPhone中的设置应用，一层一层往下跳转。

**基本用法**

假设有UINavigationController A， UIViewController B，C，D，A初始化的时候指定rootViewController为 B。

* 从B跳转到C ，C跳转到D
* [B.navigationController pushViewController:C animated:YES];

C入栈

[C.navigationController pushViewController:D animated:YES];

D入栈

* 弹出D，D出栈  
  [D.navigationController popViewControllerAnimated:YES];
* 直接跳转到rootViewController  
  [D.navigationController popToRootViewControllerAnimated:YES];
* 跳转到栈中已存在的某个ViewController
* [D.navigationController popToViewController:B animated:YES];

**多说点**

1. UINavigationController继承自UiViewController，扮演导航控制器的角色。你呈现的是他的rootViewController，并维护一个stack，可以访问viewControllers属性获取其栈中的viewController。
2. UIViewController有一个navigationController的属性，指向其所属的导航控制器。
3. 导航控制器还有两个有意思的组件，一个是导航栏，一个是底部工具栏，这个以后专门写一篇文章来说吧。

**UITabBarController**

**使用背景**

像微信那种，应用底部会有<=5个的按钮的使用场景。这种应用也很常见。一般情况下我们只需要初始化UITabBarController的每个tab对应的root view controller，那么点击底部tab的时候，就会自动跳转到每个tab对应的root view。UITabBarController也是继承自UIViewController，他的root view 可以分为两个模块，一个是底部的tab bar，一个是底部上层的content view，点击tab时，就会看到content view 中的view之间的切换。

**基本用法**

这里的基本用法就是关于UITabBarController的一些初始化了。UITabBarController有一个viewControllers的属性，是每个tab对应的UIViewController的数组，每个viewController包含一个UITabBarItem的属性，定义每个tab的title，image，badgeValue等属性。badgeValue就像微信消息那个tab对应的未读消息数。

viewController1.tabBarItem = [[UITabBarItem alloc] initWithTitle:@"One"

image:nil

tag:1];

viewController2.tabBarItem = [[UITabBarItem alloc] initWithTitle:@"Second"

image:nil

tag:2];

tabBarController.viewControllers = [NSArray arrayWithObjects:viewController1,viewController2,nil];

**多说点**

1. UITabBarController的viewControllers是copy属性，UINavigationController中的viewControllers是readonly。毕竟在UITabBarController中，viewControllers是暴露给外界添加child view controller的唯一接口。后期不知道苹果会不会对UITabBarController的公共接口迭代更新。
2. 在UITabBarController中，可以通过selectedViewController，selectedIndex两个属性获取当前选中的viewController。
3. 在项目中，UITabBarController跟UINavigationContorller经常会搭配使用，比如UITabBarController每个tab的rootViewController就是一个UINavigationController。