UIKit基础(5)

果核科技
CORE TECH.

这节课我会学到什么？

UINavigationController的使用

UITabBarController的使用

多视图之间传值

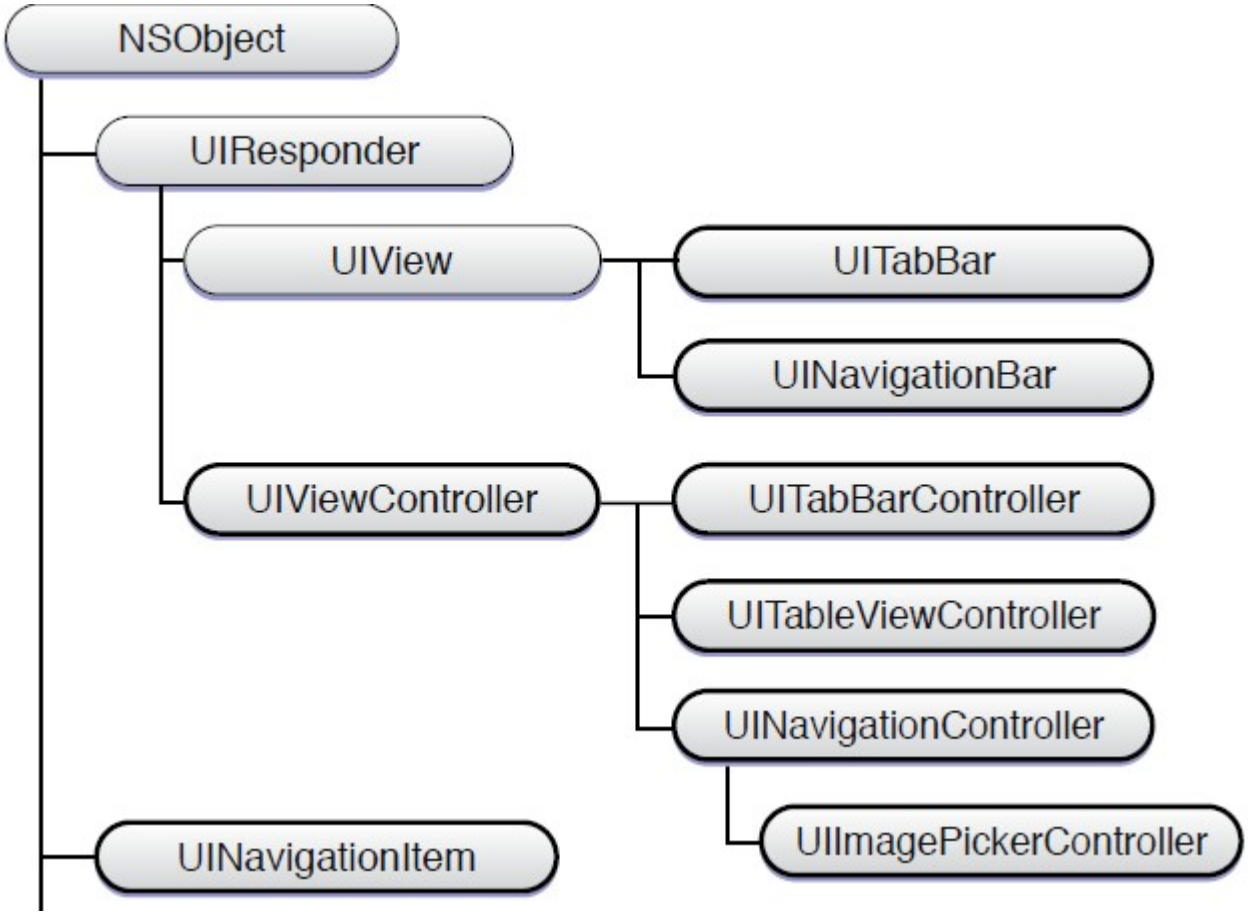
多视图app

大部分的iOS应用程序都是采用多视图设计,多视图的展现需要多个控制器来组织app.

为了方便管理多个控制器,iOS提供了两个特殊的控制器:

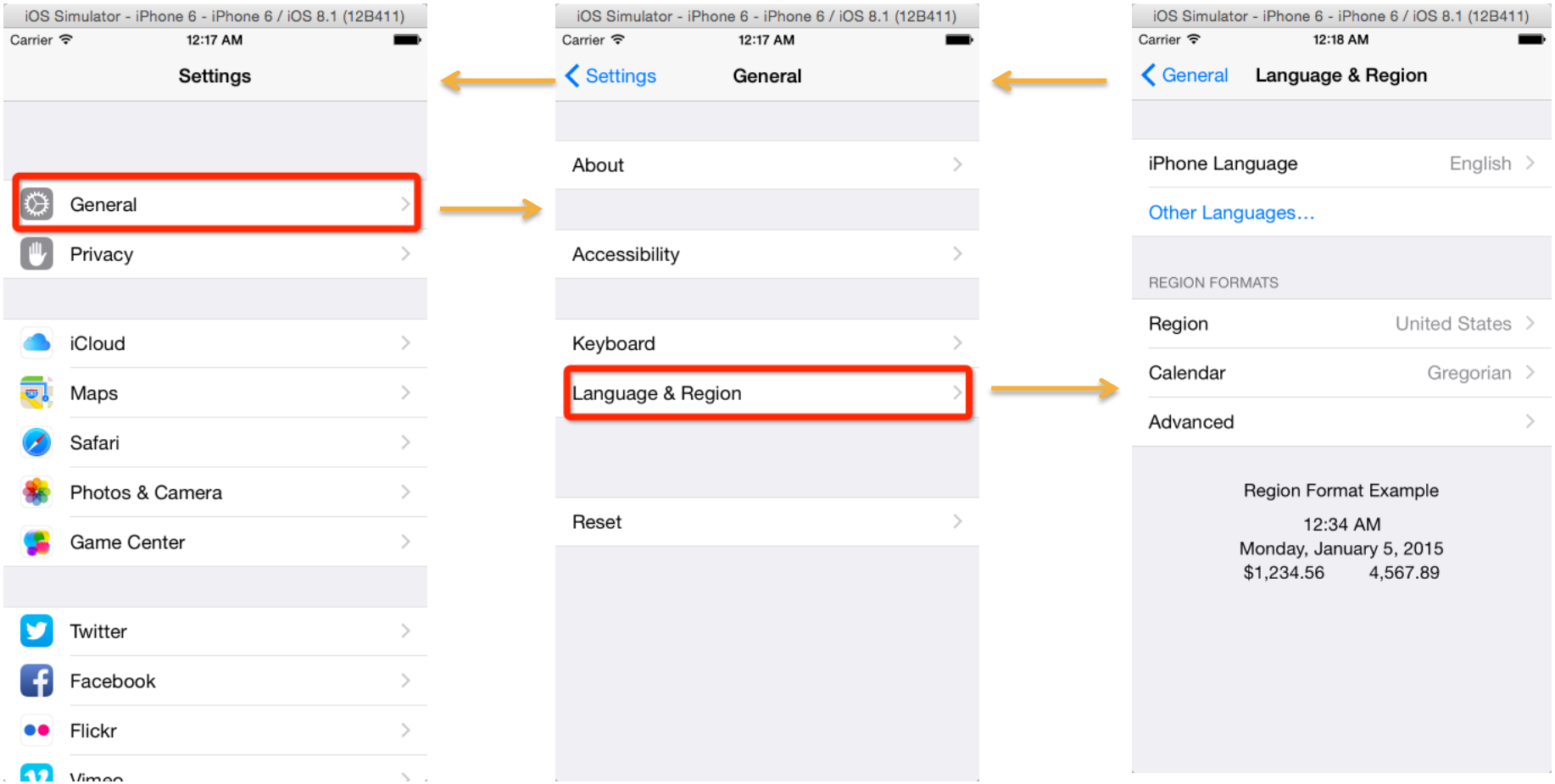
- 导航控制器（ UINavigationController ）
- 标签栏控制器（ UITabBarController ）

视图和控制器类图


NSObject
UIResponder
UIView
UITabBar
UINavigationBar
UIViewController
UITabBarController
UITableViewController
UINavigationController
UIImagePickerController
UINavigationItem

UINavigationController

导航控制器UINavigationController管理控制一系列的UIViewController.



生成导航控制器

黑核科技 CORE TECH.

```
@property (strong, nonatomic) UINavigationController *navController;

-(BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
  self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];

   FirstViewController *firstVC = [[FirstViewController alloc] init];
// 初始化UINavigationController
   self.navController = [[UINavigationController alloc] initWithRootViewController:firstVC];
//  设置这个navigationController为UIWindow的根视图
   self.window.rootViewController = self.navController;
   self.window.backgroundColor = [UIColor whiteColor];
   [self.window makeKeyAndVisible];
   return YES;
}
```

导航控制器视图结构组成

- UINavigationController主要采用栈的形式来组织管理视图控制器。
- 在创建导航控制器时需要指定根视图即用户看到的第一个视图。
- 根视图控制器是被导航控制器推入到栈中的第一个视图控制器。当用户查看下一个视图时，栈中将加入一个新的视图控制器，把它所控制的视图将展示给用户。

```
@property(nonatomic,copy) NSArray *viewControllers;
```

在视图之间进行切换


```
// 把要展示的控制器压入栈
-(void)selectRightAction
{
   SecondViewController *secondVC = [[SecondViewController alloc] init];
   [self.navigationController pushViewController:secondVC animated:YES];
}

// 把视图弹出栈
-(void)goBack:(id)sender
{
//  移除栈顶的视图控制器
   [self.navigationController popViewControllerAnimated:YES];
}
```

此外还有两种方法可以返回之前的控制器:

```
//回到指定的控制器
- popToViewController:animated:
//回到根视图控制器
- popToRootViewControllerAnimated:
```

UINavigationItem

每一个加到navigationController的viewController都会有一个对应的navigationItem，navigationItem决定了导航栏上显示的内容.

UINavgationItem包含以下主要属性：

```
// 中间标题
@property(nonatomic,copy)   NSString       *title;
//  返回按钮
@property(nonatomic,retain) UIBarButtonItem *backBarButtonItem;
//    标题视图（可自定义）
@property(nonatomic,retain) UIView         *titleView;
//  左边item
@property(nonatomic,copy) NSArray *leftBarButtonItems ;
//  右边item
@property(nonatomic,copy) NSArray *rightBarButtonItems;
```

为导航控制器添加控制按钮


```
- (void)viewDidLoad
{
    [super viewDidLoad];

    UIBarButtonItem *leftButton = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:UI
BarButtonSystemItemAction target:self action:@selector(selectLeftAction:)];
    self.navigationItem.leftBarButtonItem = leftButton;

    UIBarButtonItem *rightButton = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:U
IBarButtonSystemItemAdd  target:self action:@selector(selectRightAction:)];
    self.navigationItem.rightBarButtonItem = rightButton;
}
```

navigationBar的显示原则

通过上面介绍的内容，我们知道navigationBar中包含了这几个重要组成部分：leftBarButtonItem, rightBarButtonItem, backBarButtonItem, title。当一个view controller添加到navigationController以后，navigationBar的显示遵循一下几个原则：

1）navigationBar左侧显示原则

a）如果当前的viewController设置了leftBarButtonItem，则显示当前VC自带的leftBarButtonItem。

b）如果当前的VC没有设置leftBarButtonItem，且当前VC不是rootVC的时候，则显示前一层VC的backBarButtonItem。如果前一层的VC没有显示的指定backBarButtonItem的话，系统将会根据前一层VC的title属性自动生成一个back按钮，并显示出来。

c）如果当前的VC没有设置leftBarButtonItem，且当前VC已是rootVC的时候，左边将不显示任何东西。

此处注意：5.0中新增加了一个属性leftItemsSupplementBackButton，通过指定该属性为YES，可以让leftBarButtonItem和backBarButtonItem同时显示，其中leftBarButtonItem显示在backBarButtonItem的右边。

2） title部分的显示原则

a）如果当前VC通过 .navigationItem.titleView指定了自定义的titleView，系统将会显示指定的titleView，此处要注意自定义titleView的高度不要超过navigationBar的高度，否则会显示出界。

b）如果当前VC没有指定titleView，系统则会根据当前VC的title或者当前VC的navigationItem.title的内容创建一个UILabel并显示。

3）navigationBar右侧显示原则

a）如果当前VC指定了rightBarButtonItem的话，则显示指定的内容。

b）如果当前VC没有指定rightBarButtonItem的话，则不显示任何东西。

修改导航栏背景图片

- 修改导航栏背景图片

```
[self.navigationController.navigationBar setBackgroundImage:
[UIImage imageNamed:@"topNav"]
forBarMetrics:UIBarMetricsDefault];
```

UINavigationController的Toolbar

黑核科技
CORE TECH.

navigationController自带了一个工具栏，与UINavigationBar类似，导航控制器只拥有一个UIToolBar实例

- 设置工具栏的显示

  self.navigationController.toolbarHidden = NO;

- self.navigationController.toolbar 取到的toolBar是只读的，所以不能直接给它添加item，而是要通过调用下面的方法：（其中self指的是视图控制器）

  [self setToolbarItems:@[one, flexItem, two, flexItem, three, flexItem, four, flexItem]];

UIViewController的navigationController属性

每个viewController都有这个属性，如果当前的viewController是被加入到NavigationController里面，那么这个属性就是这个NavigationController的指针，否则就是nil。通过这个属性可以取到viewController所在的NavigationController。处于同一个navigationController栈内的不同viewController取到的navigationController是同一个（内存地址相同）。

navigationController的代理

这个代理真的很简单，就是当一个viewController要显示的时候通知一下外面，给你一个机会进行设置，当你需要对某些将要显示的viewController进行修改的话，可实现该代理。包含如下两个方法：

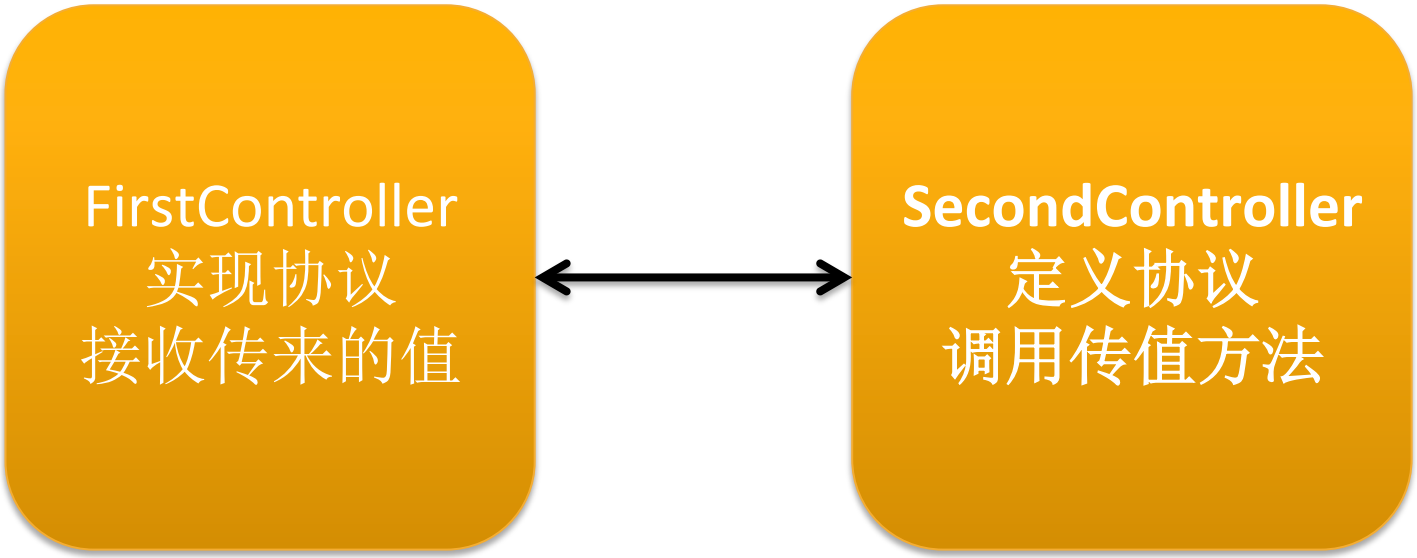
```
-(void)navigationController:(UINavigationController *)navigationController
willShowViewController:(UIViewController *)viewController animated:
(BOOL)animated;
-(void)navigationController:(UINavigationController *)navigationController
didShowViewController:(UIViewController *)viewController animated:
(BOOL)animated;
```

视图之间传值

两个视图间传值其实就是两个ViewController之间进行传值，有很多方法，这里推荐使用协议来实现。

方法1： 类间传值，利用类的属性等特性进行传值。

方法2： 协议传值。



使用storyBoard
来创建UINavigationController

模态信息视图

模态视图控制器在屏幕上显示，无需成为标准视图控制器栈的一部分，用于“中断”你当前的工作流程，对于选取数据或者演示信息非常有用。模态展示的视图可能是一个视图控制器，也可能是一个导航控制器。两个视图之间是“父子”关系。

```
// 显示模态视图
 SecondViewController *secondVC = [[SecondViewController alloc] init];
  secondVC.modalTransitionStyle = UIModalTransitionStyleFlipHorizontal;
    UINavigationController *navigation=[[UINavigationController
alloc]initWithRootViewController:secondVC];// 没这个下个页面就没导航栏了
    navigation.navigationBar.barStyle=UIBarStyleDefault;
    [self presentViewController:navigation animated:YES completion:^{[self
animationCompleted];}];

// 退出模态视图
[self dismissViewControllerAnimated:YES completion:nil];
```

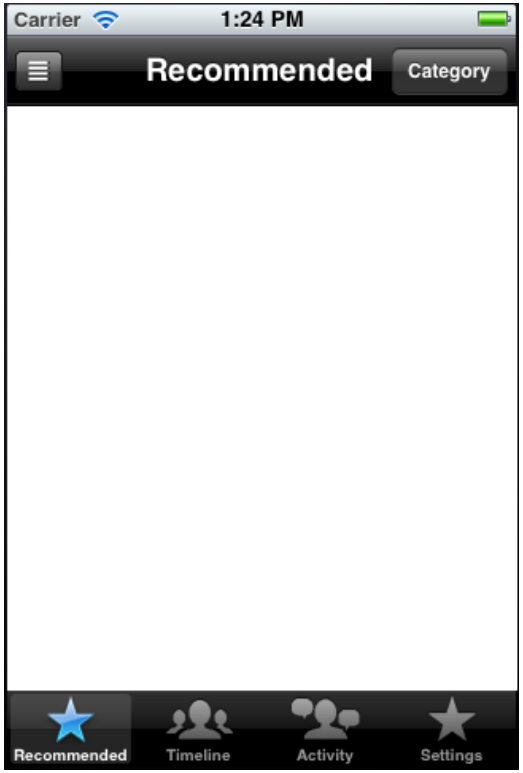
topViewController VS visibleViewController

topViewController代表当前navigation栈中最上层的VC，而visibleViewController代表当前可见的VC，它可能是topViewController，也可能是当前topViewController present出来的VC。因此UINavigationController的这两个属性通常情况下是一样，但也有可能不同。

UITabBarController的功能

和UINavigationController类似，UITabBarController也可以用来控制多个页面导航，用户可以在多个视图控制器之间移动，并可以定制屏幕底部的选项卡栏。

借助屏幕底部的选项卡 栏，UITabBarController不必像UINavigationController那样以栈的方式推入和推出视图，而是组建一系列的控制器（他们各自可以是UIViewController，UINavigationController，UITableViewController或任何 其他种类的视图控制器），并将它们添加到选项卡栏，使每个选项卡对应一个视图控制器。



UITabBarController的组成

UITabBarController的创建


```
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
 self.tbCtrl = [[UITabBarController alloc] init];
 // 1. 生成tab上对应的VC
 FirstViewController *item1 = [[FirstViewController alloc] init];
 SecondViewController *item2 = [[SecondViewController alloc] init];
 ThirdViewController *item3 = [[ThirdViewController alloc] init];
 ThirdViewController *item4 = [[ThirdViewController alloc] init];
 ThirdViewController *item5 = [[ThirdViewController alloc] init];
 // 2. 加入到tabbar的数组中
 NSArray *controllers = @[item1,item2,item3, item4, item5];
//3. 设置属性
 self.tbCtrl.viewControllers= controllers;
 self.tbCtrl. selectedIndex = 2;
self.tbCtrl.delegate = self;

 self.window.backgroundColor = [UIColor whiteColor];
 self.window.rootViewController = self.tbCtrl;
 [self.window makeKeyAndVisible];
 return YES;
```

UITabBarItem


```
// 初始化方法1
UITabBarItem *item = [[UITabBarItem alloc] initWithTitle:@"Music"
image:[UIImage imageNamed:@"3.png"] tag:103];

// 初始化方法2
UITabBarItem *item = [[UITabBarItem alloc]
initWithTabBarSystemItem:UITabBarSystemItemFeatured tag:103];

// 设置图标上的数字
self.tabBarItem.badgeValue = @"2";
```

UITabBarSystemItem

Constants

`UITabBarSystemItemMore`
The more system item. •••
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemFavorites`
The favorites system item. ★
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemFeatured`
The featured system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemTopRated`
The top rated system item. ★
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemRecents`
The recents system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemContacts`
The contacts system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemHistory`
The history system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemBookmarks`
The bookmarks system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemSearch`
The search system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemDownloads`
The downloads system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemMostRecent`
The most recent system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

`UITabBarSystemItemMostViewed`
The most viewed system item.
Available in iOS 2.0 and later.
Declared in `UITabBarItem.h`.

导航控制器和UITabBar控制器结合


```
UINavigationController *nav1=[[UINavigationController alloc]
initWithRootViewController:firstController];

NSArray *controllers = @[nav1,item2,item3, item4, item5];

self.tbCtrl.viewControllers= controllers;
```

Tab切换事件


```
- (void)tabBarController:(UITabBarController *)tabBarController
didSelectViewController:(UIViewController *) viewController
{
    NSLog(@"%d", tabBarController.selectedIndex);
}
```

作业

1、 实现一个通讯簿，两个视图，一个显示当前通讯簿的整体信息，另外一个提供给用户填写新增的通讯簿条目信息。

作业

1. 模拟iOS中电话和时钟的界面（有表格的不用）