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

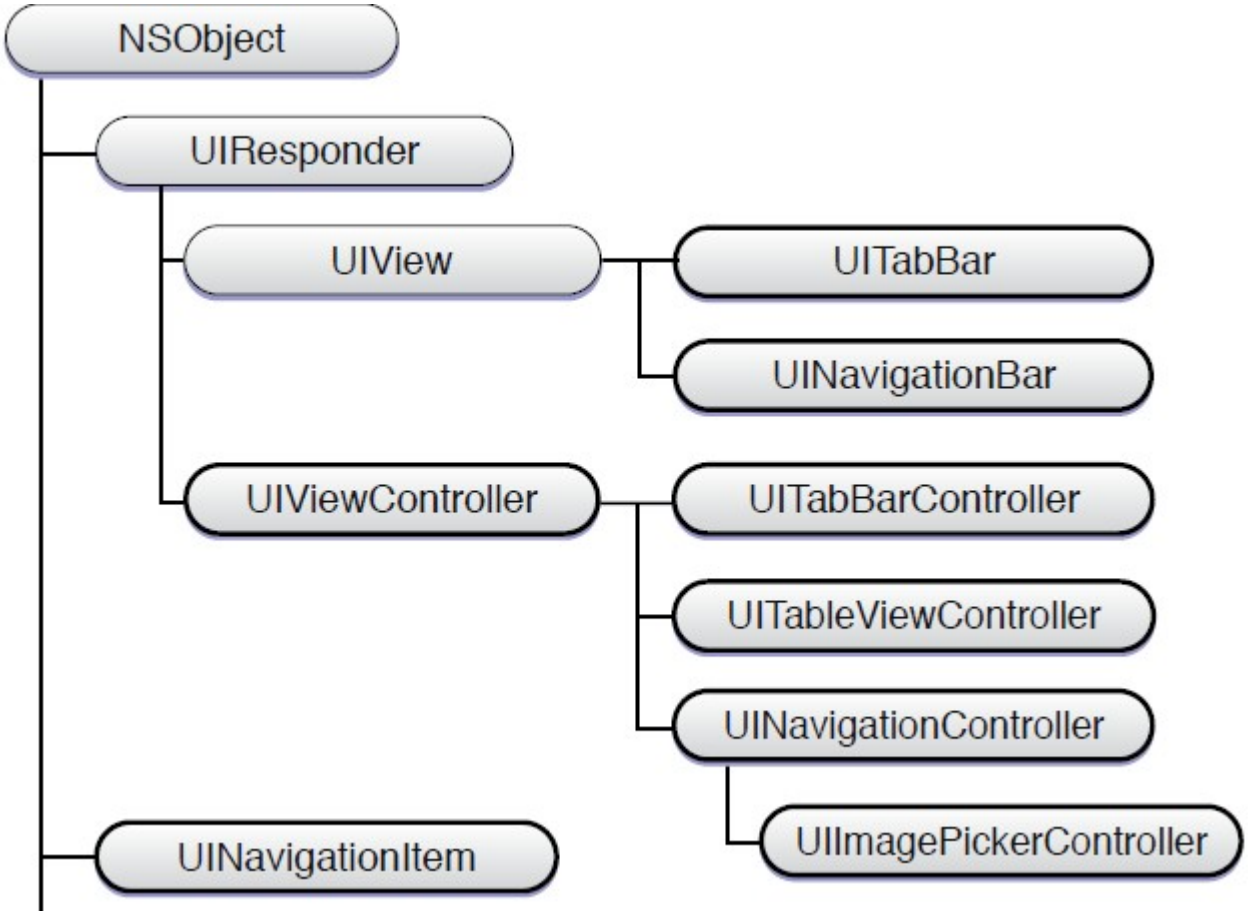
多视图app

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:

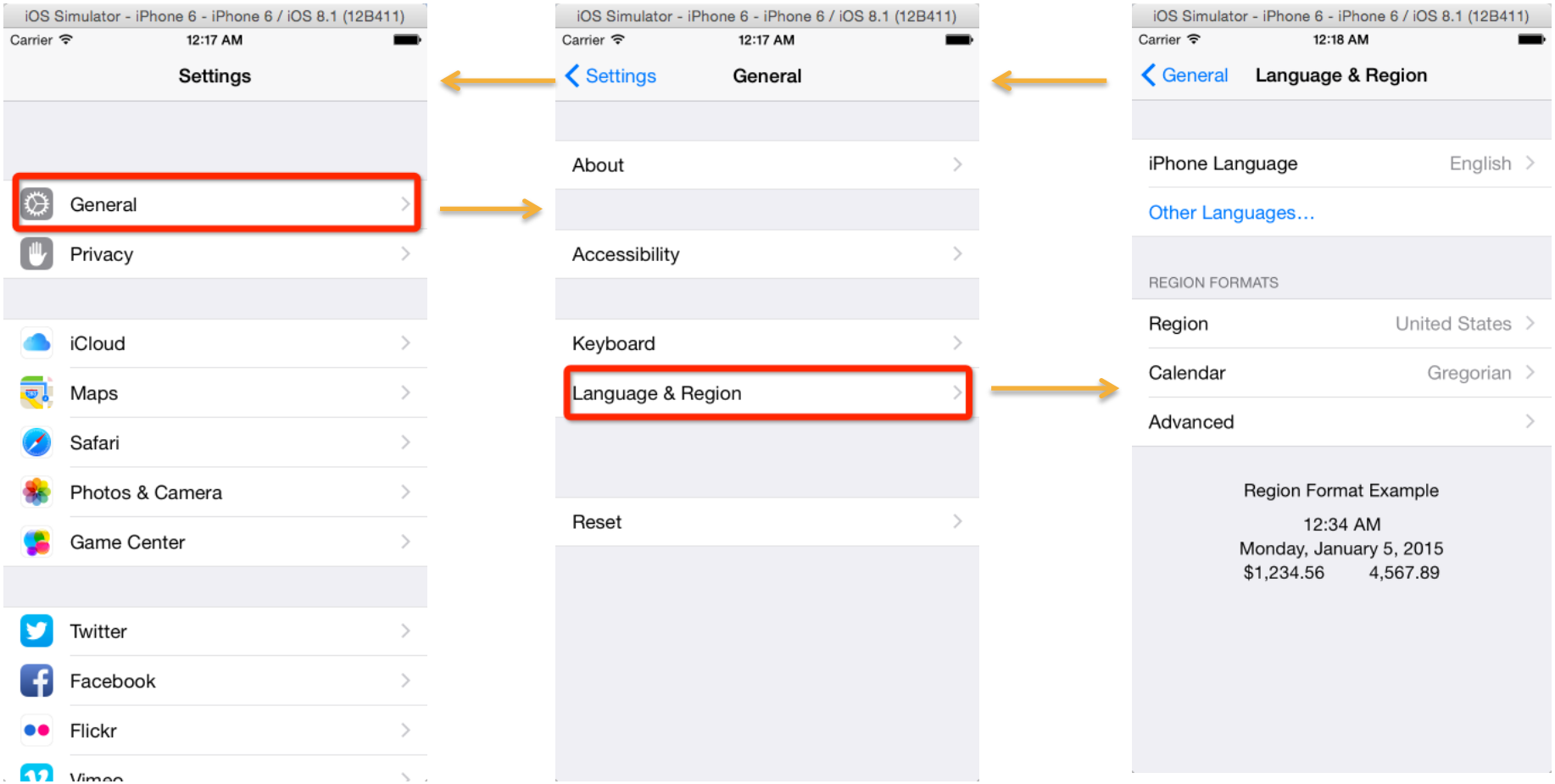
- 导航控制器（ UINavigationController ）
- 标签栏控制器（ UITabBarController ）

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：

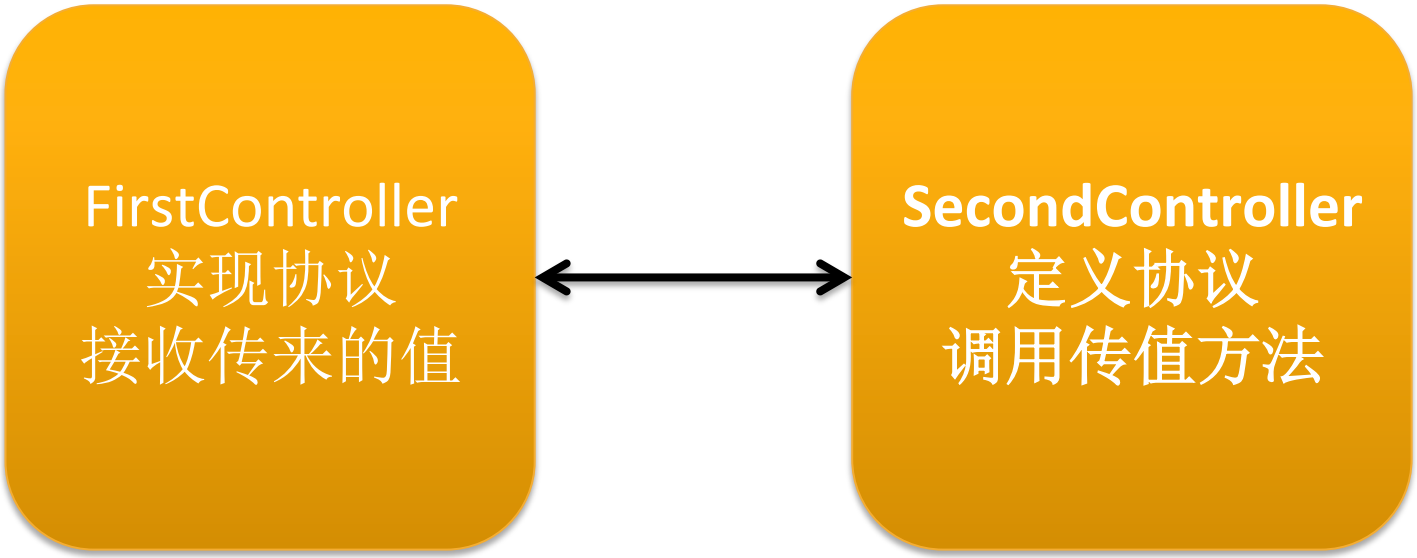
```
-(void)navigationController:(UINavigationController *)navigationController
willShowViewController:(UIViewController *)viewController animated:
(BOOL)animated;
-(void)navigationController:(UINavigationController *)navigationController
didShowViewController:(UIViewController *)viewController animated: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

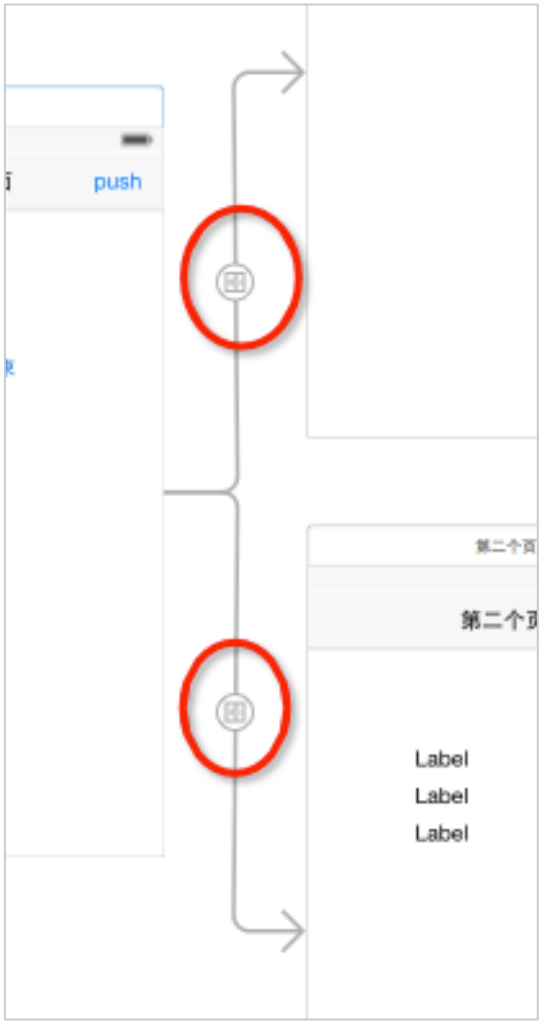
注意：storyBoard里面不能往回连线，如果想要某个点击事件触发pop操作，需要在代码里面手动实现pop方法

Segue：
使用storyBoard进行页面之间传值

Scene和Segue是storyBoard中非常重要的两个概念

Scene： 每个视图控制器都会对应一个Scene，Scene翻译为“场景”，可以理解为应用的一个界面

Segue： 就是用来实现界面跳转的连线，Scene之间通过Segue连接，Segue定义了Scene之间的跳转方式，还体现了Scene之间的关系（源控制器和目标控制器）



segue

Segue的属性

identifier // 唯一标识符，用来区分不同的segue

sourceViewController// 来源控制器

destinationViewController// 目标控制器

Segue的主要方法

- prepareForSegue:sender:

当你从当前 scene中触发一个segue的时候，系统会自动调用这个方法。如果你想从一个界面切换到里另一个界面的时候传递数据,应该重写这个方法，并在这个方法里面进行传值等操作。一般是把具体的按钮和下一个控制器进行连线。

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue.destinationViewController isKindOfClass:[SecondViewController
class]]) {
        Student *stu = [Student studenWithName:@"xiaoming" num:@"123"
andScore:50];
        ((SecondViewController *)segue.destinationViewController).student = stu;
    }
}
```

Segue的主要方法

- performSegueWithIdentifier:sender:

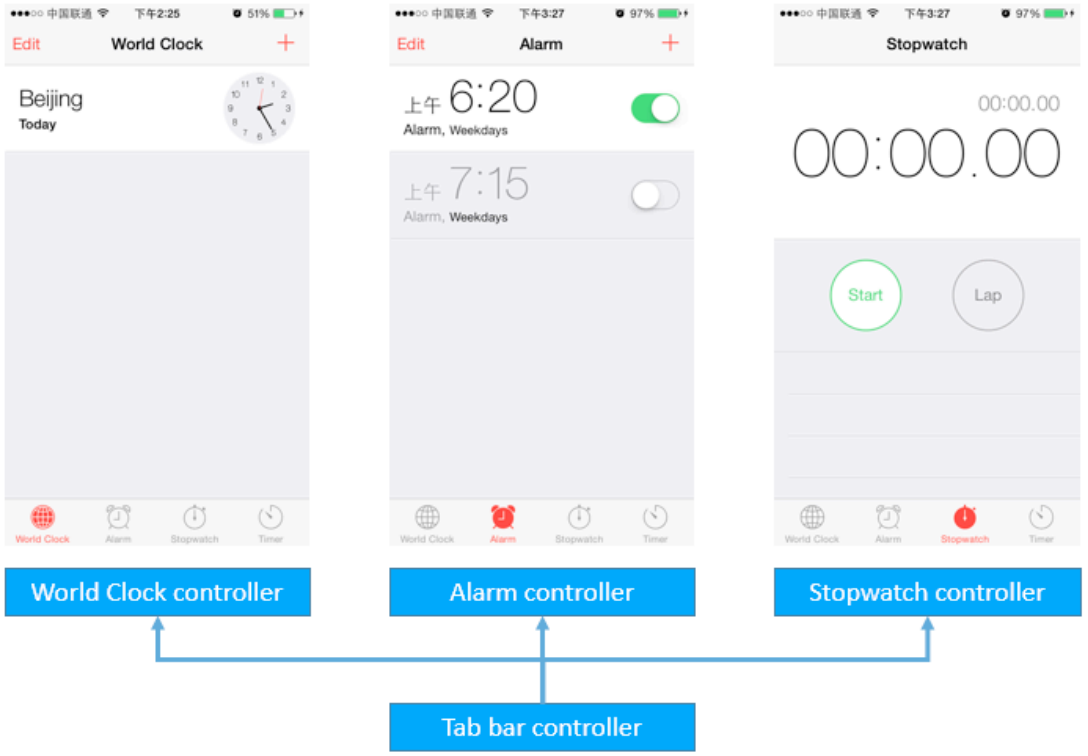
这个方法是在你需要的时候手动触发segue进行主动跳转，调用者为来源控制器。一般是把源控制器和目标控制器进行连线。手动触发segue需要给segue设置一个identifier。

```
// 其中的self 指的是来源控制器
[self performSegueWithIdentifier:@"myConnSegue" sender:nil];
```

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切换，并可以定制屏幕底部的选项卡栏。

以平行的方式管理视图，各个视图之间往往关系并不大，UITabBarController会一次性初始化所有子控制器。



UITabBarController的创建

黑核科技 CORE TECH.

```
    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];
UINavigationController  *nav=[ [UINavigationContorller
alloc]initWithRootViewController:item1];
    SecondViewController *item2 = [[SecondViewController alloc] init];
    ThirdViewController *item3 = [[ThirdViewController alloc] init];
    ThirdViewController *item4 = [[ThirdViewController alloc] init];
// 2. 加入到tabbar的数组中
    NSArray *controllers = @[nav,item2,item3, item4];
   //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

- 每个视图控制器都有一个tabBarItem属性，tabBar上显示的内容，由对应控制器的tabBarItem决定

- 设置tabBarItem的方式主要有三种:
- // 1> 可以直接设置默认的tabbarItem的title和Image
- vcA.tabBarItem.title = @"AAA";
- vcA.tabBarItem.image = [UIImage imageNamed:@""];
- vcA.tabBarItem.selectedImage = [UIImage imageNamed:@“”];
- // 2> 也可以自己初始化一个系统默认的
- UITabBarItem *itemAA = [[UITabBarItem alloc]initWithTabBarSystemItem:UITabBarSystemItemFavorites tag:100];
- // 3> 也可以自己初始化一个特定的title和image的
- UIImage *image = [UIImage imageNamed:@"circle"];
- image = [image imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
- UITabBarItem *itemAAA = [[UITabBarItem alloc]initWithTitle:@"消息" image:image tag:100];
- vcA.tabBarItem = itemAAA;

UITabBarItem常用属性

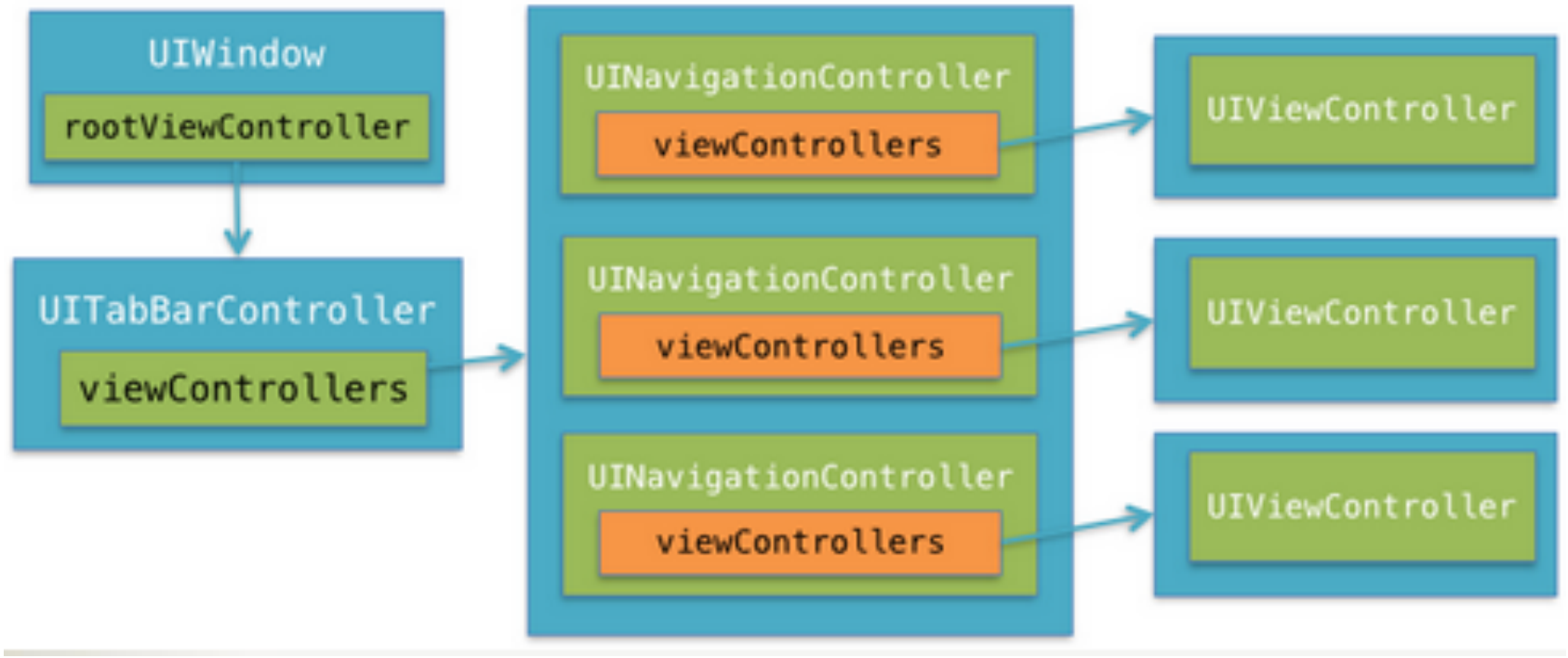

```
// 标题
@property(nonatomic,copy) NSString *title;

// 图像
@property(nonatomic,retain) UIImage *image;

//选中图像
@property(nonatomic,retain) UIImage *selectedImage;

// 小红点内容
@property(nonatomic,copy) NSString *badgeValue;
```

Tab切换事件


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

App主流UI框架结构

黑核科技 CORE TECH.



UITabBarController的StoryBoard实现

模态视图Modal

Modal是另外一种切换控制器的方式。

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。

通常用于显示独立的内容，在模态窗口显示的时其他视图的内容无法进行操作。

```
// 显示模态视图
 SecondViewController *secondVC = [[SecondViewController alloc] init];
// 设置展示动画,默认是从屏幕下方弹出视图
  secondVC.modalTransitionStyle = UIModalTransitionStyleFlipHorizontal;
    UINavigationController *navigation=[[UINavigationController
alloc]initWithRootViewController:secondVC];
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。

Modal的StoryBoard实现

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）