iOS (http://lib.csdn.net/base/ios)

iOS (http://lib.csdn.net/base/ios) - 设计模式 (http://lib.csdn.net/ios/node/673) - 设计模式 (http://lib.csdn.net/ios/knowledge/1462)

iOS 页面间几种传值方式（属性，代理，block，单例，通知）

作者：AckyiOSDeveloper (http://my.csdn.net/AckyiOSDeveloper)

第二个视图控制器如何获取第一个视图控制器的部分信息

例如 ：第二个界面中的lable显示第一个界面textField中的文本

这就需要用到属性传值、block传值

那么第一个视图控制器如何获的第二个视图控制器的部分信息

例如：第一个界面中的lable显示第二个界面textField中的文本

这就需要使用代理传值

页面间传值有八大传值方式，下面我们就简单介绍下页面间常用的五种传值方式：

（一）属性传值

第二个界面中的lable显示第一个界面textField中的文本

首先我们建立一个RootViewControllers和一个DetailViewControllers，在DetailViewControllers中声明一个textString属性，用于接收传过来的字符串，

```
8
9  #import <UIKit/UIKit.h>
10
11 @interface DetailViewController : UIViewController
12
13 @property (nonatomic, copy) NSString *textString;//属性传值
14
15
16 @end
```

同时创建一个Lable用来显示传过的字符串

```objc
#import "DetailViewController.h"

@interface DetailViewController ()

@end
@implementation DetailViewController

- (void)dealloc {
    [_textString release];
    [super dealloc];
}

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    UILabel *aLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 100,
        CGRectGetWidth(self.view.bounds) - 40, 100)];
    aLabel.backgroundColor = [UIColor whiteColor];
    aLabel.font = [UIFont systemFontOfSize:20];
    aLabel.numberOfLines = 0;
    aLabel.text = self.textString;//使用属性传递得到的文本
    [self.view addSubview:aLabel];
    [aLabel release];
    self.view.backgroundColor = [UIColor greenColor];

}

```

在RootViewControllers上引入DetailViewControllers同时声明一个textField属性用来输入字符串

```objc

#import "RootViewController.h"
#import "DetailViewController.h"

@interface RootViewController ()

@property (nonatomic, retain) UITextField *textField;

@end

@implementation RootViewController

- (void)dealloc {
    [_textField release];
    [super dealloc];
}
//重写textField的getter方法，并且为textField的相关属性赋值
- (UITextField *)textField {
    if (!_textField) {
        self.textField = [[[UITextField alloc] initWithFrame:CGRectMake(20, 100
            , CGRectGetWidth(self.view.bounds) - 40, 40)] autorelease];
        _textField.borderStyle = UITextBorderStyleRoundedRect;
    }
    return _textField;
}

```

然后在RootViewControllers上我们创建并添加一个button，当点击button时响应相应方法进行视图间的切换完成视图间的传值

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    self.view.backgroundColor = [UIColor redColor];
    [self.view addSubview:self.textField];
//    创建一个轻拍手势当点击视图的任何位置就可以取消键盘的第一响应
    UITapGestureRecognizer *tap = [[UITapGestureRecognizer alloc]
        initWithTarget:self action:@selector(handleTap:)];
    [self.view addGestureRecognizer:tap];
    [tap release];

    UIButton *pushButton = [UIButton buttonWithType:UIButtonTypeCustom];
    pushButton.frame = CGRectMake(0, 0, 80, 40);
    [pushButton setTitle:@"入栈显示" forState:UIControlStateNormal];
    pushButton.titleLabel.font = [UIFont systemFontOfSize:20];
    pushButton.center = self.view.center;
    [pushButton addTarget:self action:@selector(handlePush:) forControlEvents:
        UIControlEventTouchUpInside];
    [self.view addSubview:pushButton];
}

- (void)handlePush:(UIButton *)sender {
    DetailViewController *detailVC = [[DetailViewController alloc] init];
    //创建完对象，就为对应的属性textString赋值
    detailVC.textString = self.textField.text;
//    通过导航控制器push进入下一个界面
    [self.navigationController pushViewController:detailVC animated:YES];
    [detailVC release];
}
```

（二）Block传值

block传值也是从第二个界面给第一个界面传值

首先我们在DetailViewcontrollers的.h文件中，属性

```
#import <UIKit/UIKit.h>

typedef void (^PassingValueBlock)(UILabel *);

@interface DetailViewController : UIViewController

@property (nonatomic, copy) PassingValueBlock passingValue;//使用block传值

@end
```

在RootViewControllers的.m文件中，其他不变，在button的响应方法里我们为block属性赋值完成block传值

```
52
53 - (void)handlePush:(UIButton *)sender {
54     DetailViewController *detailVC = [[DetailViewController alloc] init];
55 //    为detailVC的block属性赋值，完成block传值
56     detailVC.passingValue = ^(UILabel *aLabel){
57         aLabel.text = self.textField.text;
58     };
59     [self.navigationController pushViewController:detailVC animated:YES];
60     [detailVC release];
61 }
62
```

（三）代理传值

RootViewControllers页面push到DetailViewControllers页面，如果DetailViewControllers页面的信息想回传（回调）到RootViewControllers页面，用代理传值，其中DetailViewControllers定义协议和声明代理，RootViewControllers确认并实现代理，RootViewControllers作为DetailViewControllers的代理

首先在DetailViewControllers.h文件中我们创建协议方法

```
8
9  #import <UIKit/UIKit.h>
10 //第三种传值方式，代理
11 @class DetailViewController;
12
13 @protocol PassingValueDelegate <NSObject>
14
15 @optional
16
17 - (void)viewController:(DetailViewController *)viewcontroller
       didPassingValueWithInfo:(id)info;
18
19 @end
20
21 @interface DetailViewController : UIViewController
22
23 @property (nonatomic, assign) id<PassingValueDelegate>
       delegate;//通过代理对象传值
24
25 @end
26
```

在DetailViewControllers的.m中我们判定代理对象存在时，为其绑定相应方法

```
34
35 - (void)viewWillDisappear:(BOOL)animated {
36     [super viewWillDisappear:animated];
37     //视图将要消失时，通过代理传值
38     //首次判定代理是否存在，并且代理能够响应代理方法时，才执行代理方法
39     if (self.delegate && [self.delegate respondsToSelector:
           @selector(viewController:didPassingValueWithInfo:)]) {
40         //传递当前视图的背景颜色
41         [self.delegate viewController:self
               didPassingValueWithInfo:self.view.backgroundColor];
42     }
43 }
44
```

RootViewControllers的.m文件中我们指定代理并让其执行代理的方法

```
52
53 - (void)handlePush:(UIButton *)sender {
54     DetailViewController *detailVC = [[DetailViewController
           alloc] init];
55 //    为detailVC指定代理对象
56     detailVC.delegate = self;
57     [self.navigationController pushViewController:detailVC
           animated:YES];
58     [detailVC release];
59 }
60
61 - (void)viewController:(DetailViewController *)viewcontroller
       didPassingValueWithInfo:(id)info {
62 //     把第二个视图的颜色传给第一个视图
63     self.view.backgroundColor = info;
64 }
65
```

（四）单例传值

单例传值（实现共享）

AppStatus.h 创建一个单例类 AppStatus


```
#import <Foundation/Foundation.h>

@interface AppStatus : NSObject
{
    NSString *_contextStr;
}

@property(nonatomic,retain)NSString *contextStr;

+(AppStatus *)shareInstance;

@end
```


AppStatus.m


```
#import "AppStatus.h"

@implementation AppStatus

@synthesize contextStr = _contextStr;

static AppStatus *_instance = nil;

+(AppStatus *)shareInstance
{
    if (_instance == nil)
    {
        _instance = [[super alloc]init];
    }
    return _instance;
}

-(id)init
{
    if (self = [super init])
    {

    }
    return self;
}

-(void)dealloc
{
    [super dealloc];
}

@end
```


RootViewController.h


```
#import "RootViewController.h"
#import "DetailViewController.h"
#import "AppStatus.h"

@interface RootViewController ()

@end

@implementation RootViewController

-(void)loadView
{
    //核心代码
    UIButton *btn = [UIButton buttonWithType:UIButtonTypeRoundedRect];
;
    btn.frame = CGRectMake(0, 0, 100, 30);
    [btn setTitle:@"Push" forState:0];
    [btn addTarget:self action:@selector(pushAction:) forControlEvent
s:UIControlEventTouchUpInside];
    [self.view addSubview:btn];
}

-(void)pushAction:(id)sender
{
    tf = (UITextField *)[self.view viewWithTag:1000];

 //单例传值  将要传递的信息存入单例中（共享中）
  //  [[AppStatus shareInstance]setContextStr:tf.text]; 跟下面这种写法是
等价的
    [AppStatus shareInstance].contextStr = tf.text;
    //导航push到下一个页面
    //pushViewController 入栈引用计数+1，且控制权归系统
    DetailViewController *detailViewController = [[DetailViewControll
er alloc]init];

    //导航push到下一个页面
    [self.navigationController pushViewController:detailViewControlle
r animated:YES];
    [detailViewController release];
}

@end
```


DetailViewController.h


```
#import <UIKit/UIKit.h>
@protocol ChangeDelegate;//通知编译器有此代理

@interface DetailViewController : UIViewController
{
    UITextField *textField;
}

@end
```


DetailViewController.m


```
#import "DetailViewController.h"
#import "AppStatus.h"

@interface DetailViewController ()

@end

@implementation DetailViewController

@synthesize naviTitle = _naviTitle;

-(void)loadView
{
    self.view = [[[UIView alloc]initWithFrame:CGRectMake(0, 0, 320, 480)]autorelease];

    //单例
    self.title = [AppStatus shareInstance].contextStr;
    textField = [[UITextField alloc]initWithFrame:CGRectMake(100, 100, 150, 30)];
    textField.borderStyle = UITextBorderStyleLine;
    [self.view addSubview:textField];
    [textField release];

    UIBarButtonItem *doneItem = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemDonetarget:self action:@selector(doneAction:)];
    self.navigationItem.rightBarButtonItem = doneItem;
    [doneItem release];
}

//这个方法是执行多遍的  相当于刷新view
-(void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    tf = (UITextField *)[self.view viewWithTag:1000];
    tf.text = [AppStatus shareInstance].contextStr;
}

//pop回前一个页面
-(void)doneAction:(id)sender
{
    //单例传值
    [AppStatus shareInstance].contextStr = textField.text;
    [self.navigationController popToRootViewControllerAnimated:YES];
}
```

（五）通知传值

谁要监听值的变化，谁就注册通知 特别要注意，通知的接受者必须存在这一先决条件

1、注册通知

2、通知中心发送一条消息通知，其中name前后一定要一样

3、实现通知中心内部的方法，并实现传值

4、消息发送完，要移除掉。（页面将要消失的时候）

A页面RootViewController.m

```
#import "RootViewController.h"
#import "SecondViewController.h"

@interface RootViewController ()

@property (nonatomic, retain) UILabel *label; //通过label属性显示传递的内容

@end

#define kScreenWidth [UIScreen mainScreen].bounds.size.width
#define kScreenHeight [UIScreen mainScreen].bounds.size.height
#define kNotificationNameChangeLabelText @"changLabelText"

@implementation RootViewController
- (void)dealloc {
    [_label release];
    //移除rootVC上所有监听的通知
    [[NSNotificationCenter defaultCenter] removeObserver:self];
    [super dealloc];
}


- (UILabel *)label {
    if (!_label) {
        self.label = [[UILabel alloc] initWithFrame:CGRectMake(10, 100, kScreenWidth - 20,
            200)];
        self.label.textAlignment = NSTextAlignmentCenter;
        self.label.font = [UIFont boldSystemFontOfSize:30];
        self.label.backgroundColor = [UIColor redColor];
        self.label.text = @"烧烤?烧烤?烧烤!";
        self.label.textColor = [UIColor whiteColor];
    }
    return _label;
}
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"ROOT";
    [self.view addSubview:self.label];

    UIButton *rightButton = [UIButton buttonWithType:UIButtonTypeCustom];
    rightButton.frame = CGRectMake(0, 0, 60, 44);
    [rightButton setTitle:@"点你" forState:UIControlStateNormal];
    [rightButton setTitleColor:[UIColor redColor] forState:UIControlStateNormal];
    [rightButton addTarget:self action:@selector(handleTapButtonAction:) forControlEvents:
        UIControlEventTouchUpInside];
    UIBarButtonItem *rightBarButton = [[UIBarButtonItem alloc] initWithCustomView:
        rightButton];
    self.navigationItem.rightBarButtonItem = rightBarButton;

    /*
     addObserver: 注册监听者，一般都是Controller本身去监听一个通知，
     selector ： 当监听到通知的时候执行的方法
     name ： 通知的名字， 要和发送通知的对象的名字一致
     object : nil
     */
    //注册监听者
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector
        (handleChangeLabelText:) name:kNotificationNameChangeLabelText object:nil];
    // Do any additional setup after loading the view.
}

- (void)handleChangeLabelText:(NSNotification *)notification {
    NSDictionary *dic = notification.userInfo;
    self.label.text = dic[@"text"];
}

- (void)handleTapButtonAction:(UIButton *)sender {
    SecondViewController *secondVC = [[SecondViewController alloc] init];
    [self.navigationController pushViewController:secondVC animated:YES];
}
```

SecondViewController.m

```
#import "SecondViewController.h"

@interface SecondViewController ()<UITextFieldDelegate>
@property (nonatomic, retain) UITextField *textField;
@end

#define kScreenWidth [UIScreen mainScreen].bounds.size.width
#define kScreenHeight [UIScreen mainScreen].bounds.size.height
#define kNotificationNameChangeLabelText @"changLabelText"
@implementation SecondViewController
- (void)dealloc {
    [_textField release];
    [super dealloc];
}

//创建一个输入框，通过通知把输入的内容传到前一个界面
- (UITextField *)textField {
    if (!_textField) {
        self.textField = [[[UITextField alloc] initWithFrame:CGRectMake(30, 100,
            kScreenWidth - 60, 44)] autorelease];
        self.textField.placeholder = @"请输入文字";
        self.textField.borderStyle = UITextBorderStyleRoundedRect;
        self.textField.returnKeyType = UIReturnKeyDone;
        self.textField.delegate = self;
    }
    return _textField;
}



- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"Second";
    self.view.backgroundColor = [UIColor colorWithRed:236.0 / 255.0 green:236.0/ 255.0
        blue:73.0 / 255.0 alpha:1.0];
    [self.view addSubview:self.textField];
    // Do any additional setup after loading the view.
}

- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    //每个应用程序只有一个通知中心，通知中心，有一个单例方法defaultCenter，获取唯一的通知中心对象
    /*
     postNotificationName: 给发送的通知起一个名字，
     object : 传递参数，一般可以为nil。如果为self就是把当前Controller传递给监听者
     userInfo :  传递需要的数据，用key：Value的形式把数据封装在字典里边，传递给监听者
     */
    [[NSNotificationCenter defaultCenter] postNotificationName:
        kNotificationNameChangeLabelText object:self userInfo:@{@"text" : self.textField.
        text}];
    [self.navigationController popViewControllerAnimated:YES];
    return YES;
}
```

查看原文>> (http://blog.csdn.net/AckyiOSDeveloper/article/details/50434448)

4

看过本文的人也看了：

- iOS知识结构图 (http://lib.csdn.net/base/ios/structure)
- IOS 设计模式 生成器模式 (http://lib.csdn.net/article/ios/42132)
- 设计模式五 监听器模式(android) & 代... (http://lib.csdn.net/article/ios/42110)
- iOS开发——单例的实现、使用与架构 (http://lib.csdn.net/article/ios/35942)
- iOS页面间传值的方式（Delegate/NSNot... (http://lib.csdn.net/article/ios/42105)
- iOS监听模式之KVO、KVC的高阶应用 (http://lib.csdn.net/article/ios/42122)

发表评论

输入评论内容

发表

0个评论