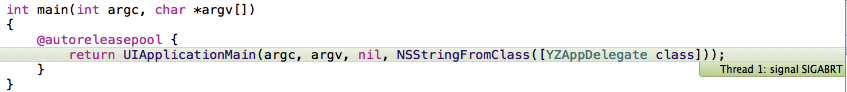
# 11.21常见问题

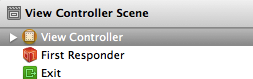
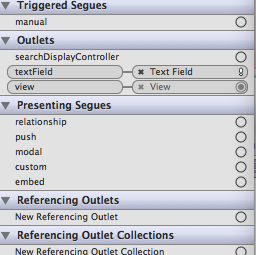
## 一storyboard连线问题





产生原因：将与storyboard关联的属性删除了，但是storyboard中还保持之前所关联的属性。

解决：

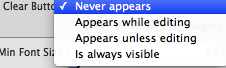
1. 点击view controller 
2. 点击这排最后一个按钮
3. 会出现
4. 发现感叹号。点击感叹号的左边的x，取消关联就不会报错了。

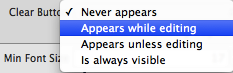
## 二 文本框中怎么输入显示类似输入密码时候的东东。

解决：勾选这个

## 三．输入文本的时候，怎么显示右边的X按钮

1.点击

2.会显示

3.选择，表示当编辑的时候，会出出现X。

效果：

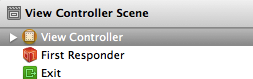
# 11.22 常见问题

## 一．Storyboard连线问题。

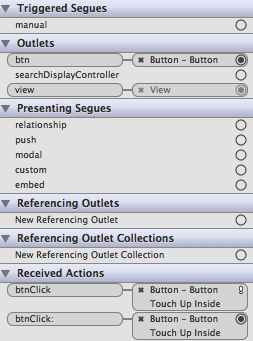
报错原因：1.没有实现btnClick这个方法。

解决方式一：添加这个btnClick这个方法

解决方式二：

1.点击view controller 

2.点击这排最后一个按钮

3.会出现

4.发现感叹号没，和之前一样，x了它，就哦了。

注意点；OC中冒号也算做方法名的一部分喔，记住！

## 二．结构体问题



报错原因：OC语法规定：不允许直接修改某个对象的结构体属性的成员

\_btn 是个对象

frame是个结构体。

对象和结构体是不一样的，结构体是C语言中的，里面可以定义许多属性，但是不能定义方法，而对象是即可以定义属性又可以定义方法的，是典型的面向对象语法。

如何改变对象中结构体属性的成员：

解决方法一：

// 既然不能直接修改对象中的结构体属性成员

// 先取出结构体

CGRect frame = \_btn.frame;

// 修改结构体

frame.origin.y -= 10;

// 将修改后的结构体重新赋值回去

\_btn.frame = frame;

解决方法二：

// 先取出y值

CGFloat y = \_btn.frame.origin.y;

// 修改y值

y -= 10;

// 重新设置\_btn的y值，其他属性和\_btn保持不变

\_btn.frame = CGRectMake(\_btn.frame.origin.x, y, \_btn.frame.size.width,\_btn.frame.size.height);

## 三．Id问题



报错原因：id类型不能使用点语法

解决方式一：

// 利用get方法获取tag值

NSInteger i = [sender tag];

解决方式二：

// 将id强转为UIButton

UIButton \*button = (UIButton \*)sender;

// 就能使用点语法获取tag，编译器很笨的，他只会根据当前类型，去判断是否能使用这个语法。一般强转为对应类型，就能使用对应类型的方法了。

NSInteger i = button.tag;

## 四．如果发现给控件设置transform属性，控件没有任何反应，或者反应了，但是效果不对。

解决方法：将这个选项取消勾选。

## 五．如果想让同一个控件同时即改变位置的移动，又放大。这样设置是无效果的。

\_btn.transform = CGAffineTransformMakeTranslation(0, 100);

\_btn.transform = CGAffineTransformMakeScale(1.2, 1.2);

这样操作是创建新的transform然后赋值，给按钮的transform，第二次赋值的会把之前赋值的给覆盖，所以会达不到想要的效果。

解决方法：

\_btn.transform = CGAffineTransformMakeTranslation(0, 100);

// 在之前的transform情况下，继续添加缩放的形变。

\_btn.transform = CGAffineTransformScale(\_btn.transform, 1.2, 1.2);

## 六．四舍五入问题。

float i = 1.7;

// 会自动四舍五入，不保留小数

NSLog(@"%0.f",i); // 打印结果2

// 强转类型不会四舍五入

int j = (int)i;

NSLog(@"%d",j); // 打印结果1

## 七．优先级问题

// int b = 2;

// int a = 4 \* (b == 2? 1:2);

// NSLog(@"%d",a); 打印出4

// int a = 4 \* b == 2? 1:2;

// NSLog(@"%d",a); 打印出2

// 由此得出 \* 比 == 优先级高，先算\*,在算==

## 八．模拟器黑屏

解决方法：

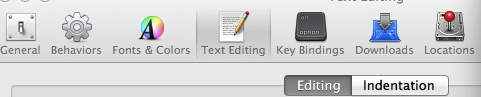


## 九．打代码时，Xcode没提示

解决方法：

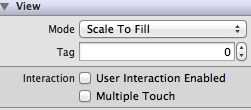
1. 点击Preferences



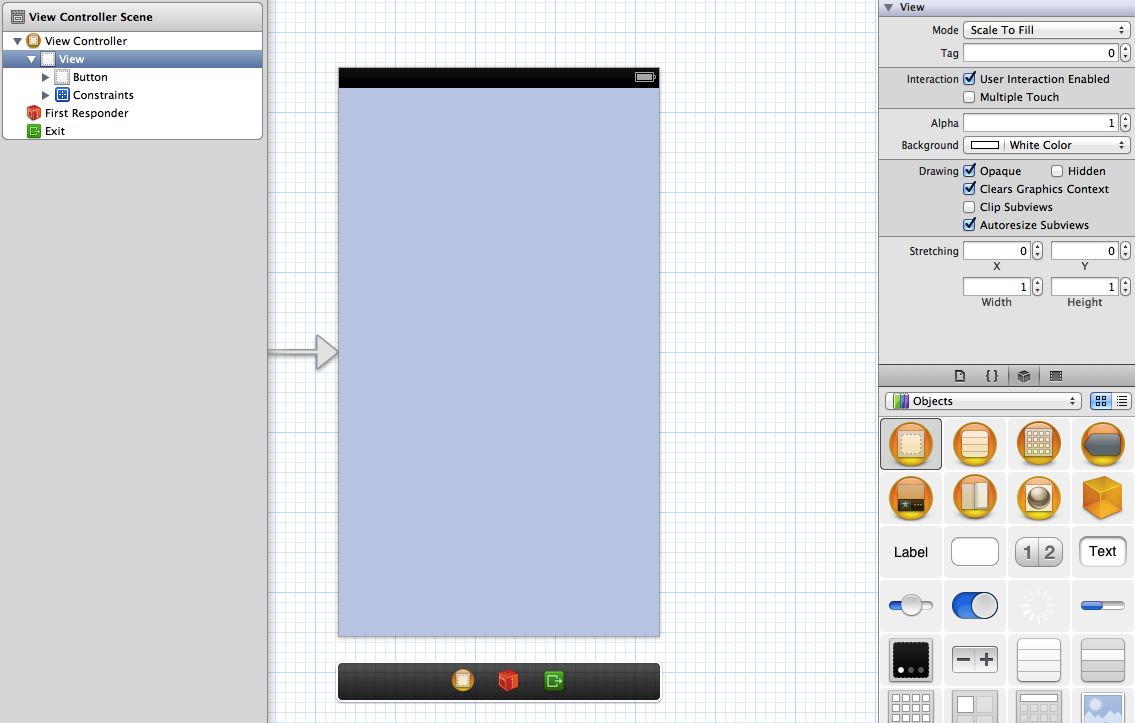
1. 进入Text Editing

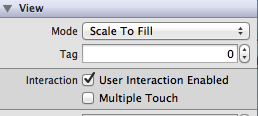
2. 勾选

## 十．有的同学可能在勾选 Autolayout的时候，搞错了对象，误把控制器的View的User Interaction Enabled勾选掉了。



解决办法：User Interaction Enabled 必须勾选，否则控制器根视图中所有子控件无法进行任何操作。





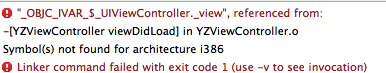
# 11.23 常见问题

## 一．访问权限

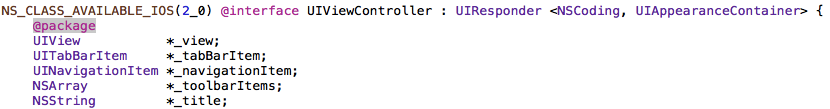
错误：



会报链接错误。



报错原因，利用下划线访问了@package这个权限里的东西。



被@package 修饰的成员属性只能在同一个框架内部才允许访问。否则会引发link erro。

@private 实例变量只能被声明它的类访问

@protected 实例变量能被声明它的类和子类访问。

@public 实例变量可以被任何类访问。

# 11.24 常见问题

## 一．运行程序时，显示运行完成，但是模拟器没反应。



问题原因： 有时候应用程序的标示符一样，会导致程序不能成功运行。

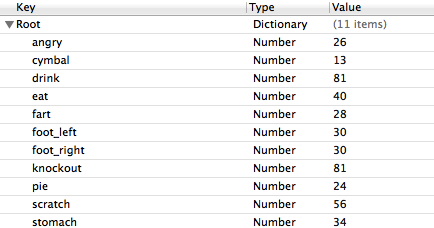
解决方式一：将模拟器之前的程序删除。

解决方式二: 将模拟器还原

## 二．2d表示保留两位 02d 表示不够了 用0来补齐。

NSLog(@"%02d",cols);

## 三．Plist文件读取。



错误原因：不要看到有很多元素的，就认为是数组。

这样解析是错误的。

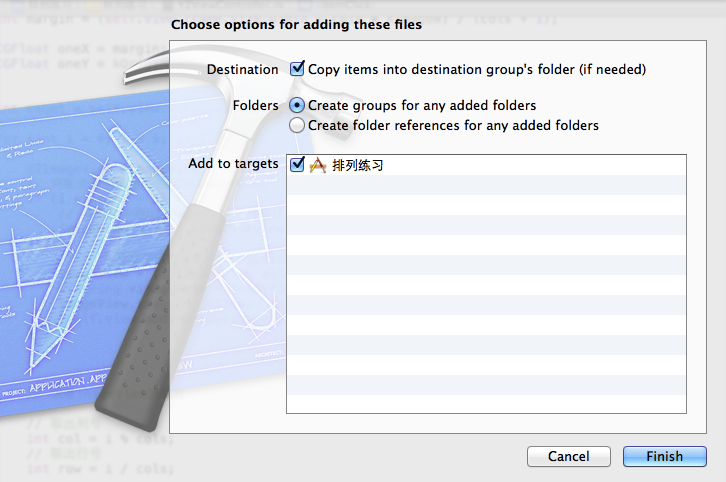
\_dict = [NSArray arrayWithContentsOfFile:path];

这个pist文件是一个字典，正确的解析此plist文件。

// 2.根据文件路径加载字典

\_dict = [NSDictionary dictionaryWithContentsOfFile:path];

## 四，给程序拖图片的时候，一定要注意，如下图勾选的，都要勾选中。



## 五．禁用UITextField的双击出现复制粘贴剪切等操作

解决方式：

有时候我们需要禁用UITextField的双击出现copy paste的功能，然而UITextField本身没有直接设置禁止用户复制粘贴剪切操作等方法，但是可以重载canPerformAction方法来实现。

新建一个类继承UITextField，然后实现canPerformAction方法：  
只需覆盖canPerformAction:withSender方法就可以，canPerformAction:withSender属于UIResponder类的。

如下：

-(BOOL)canPerformAction:(SEL)action withSender:(id)sender {

UIMenuController \*menuController = [UIMenuController sharedMenuController];

if (menuController) {

[UIMenuController sharedMenuController].menuVisible = NO;

}

return NO;

}

# 11.25 常见问题

## 1.将UIButton添加到UILabel，UIButton是不能点击的。

原因： 1.UILabel是继承UIView的，默认不能监听点击事件。UIButton是继承UIControl能够监听点击事件。

2.将UIButton添加到UILabel，他们之间的关系是UILabel是UIButton的父视图，父视图都不能监听点击事件，自然而然不会将事件传递给子视图，因此UIButton也不能监听点击事件了。

思维指导：有些人会认为UIControl不是继承UIView的吗，都是继承UIView，为什么单独继承UIControl可以监听点击事件，继承UIControl的父类UIView不能监听事件了，这是因为我们一般在父类里实现的都是一些共用的属性和方法，而在子类中具体实现子类特有的方法。因此在UIControl实现了监听点击的特有方法，即继承UIControl才能监听点击事件。

## 2.发现不少人在给成员变量初始化的时候，容易进错一个方法去初始化。

注意这个方法只有在内存发生警告的时候才会调用。

- (void)didReceiveMemoryWarning

{

[super didReceiveMemoryWarning];

// Dispose of any resources that can be recreated.

}

大部分成员属性的初始化应该在viewDidLoad里面进行。

- (void)viewDidLoad

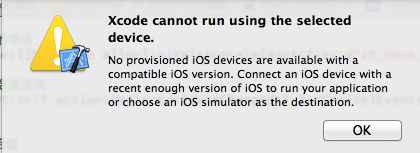
{

[super viewDidLoad];

}

最牛解决方法：在一开始就将didReceiveMemoryWarning这个方法删掉。删掉不会影响程序运行。

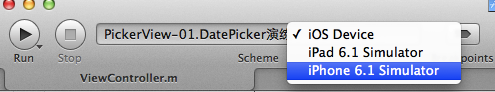
## 3.模拟器问题



当出现这个问题的时候，原因：没有选择模拟器。



解决办法：



# 11.26常见问题

## 1.对象方法和类方法问题(此问题基础好的可以看看，基础稍微差的，可以先放着，以后回顾的时候，看看。)

// 创建视图的工厂方法

+ (UIView \*)rowViewInitWithicon:(UIImage \*)icon shuju:(NSString \*)shuju

{

rowView \*viewtext = [[NSBundle mainBundle]loadNibNamed:@"rowView" owner:nil options:nil][0];

[viewtext.btntouxiang setBackgroundImage:icon forState:UIControlStateNormal];

viewtext.mingzilablexiao.text = shuju;

// 重点是这句

// 这是往通知中心添加一条通知 指定通知名称为 back 当观察者self监听到 back 通知是 就调用 callback

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(callback) name:@"back" object:nil];

/\* 以下是报错信息：

+[rowView callback]: unrecognized selector sent to class 0x79d8

2013-11-26 15:31:02.581 lianxirenlianxi[1266:c07] \*\*\* Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '+[rowView callback]: unrecognized selector sent to class 0x79d8'

\*/

// reason: '+[rowView callback]: 看到报错原因里的+就想到没有实现callback这个类方法。

// 由于self这个观察者是在类方法中添加的，指的是一个类，所以在调用方法的时候，他会去类方法中找有没有这个方法，不会去对象方法中找。因此我们也应该实现类方法。因此这里的self也只能调用类方法

// 由于实现的callback为对象方法 所以会报错

// 解决方法 把callback 写成类方法 供观察者调用

return viewtext;

}

//callback方法

- (void)callback

{

NSLog(@"11111111111");

}

解决方法，将callback 写成类方法 供观察者调用

//callback方法

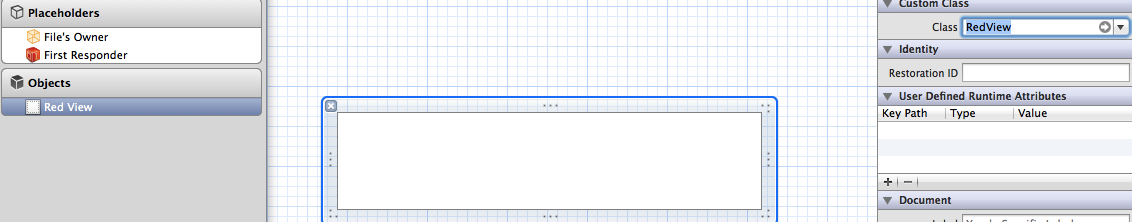
+ (void)callback

{

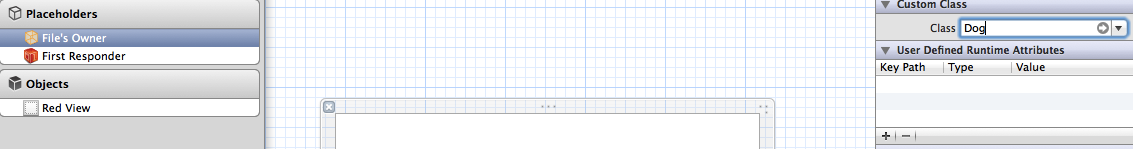
NSLog(@"11111111111");

}

## 2.Xib是用来描述视图长什么样子，一个项目中允许有很多xib，因此我们需要给xib绑定一个标识，即他View中对应的class是谁，就代表描述哪个class。



Xib中owner的class是用来告诉xib中的View需要调用哪个对象的方法，就填谁。比如需要调用dog类中的方法，就填dog。



注意：在连线选择上别连错了，

步骤一：先考虑自己是想给视图添加控件了还是想给视图添加事件

步骤二：添加控件就跟xib中的view连线。添加一些事件就给xib中的File’s Owner 连线。

## 3.代码顺序问题

[UIView animateWithDuration:0.5 animations:^{

CGRect tmpFrame = sender.superview.frame;

tmpFrame.origin.x = self.view.frame.size.width;

sender.superview.frame = tmpFrame;

sender.superview.alpha = 0;

} completion:^(BOOL finished) {

int index = [self.view.subviews indexOfObject:sender.superview];

[sender.superview removeFromSuperview];

[UIView animateWithDuration:0.2 animations:^{

for (int i = index; i<self.view.subviews.count; i++)

{

UIView \*chlid = self.view.subviews[i];

CGRect tmp = chlid.frame;

tmp.origin.y -=kViewH+1;

chlid.frame =tmp;

}

}];

// 在这判断删除按钮是否允许点中，会在动画执行完毕的时候，判断。

\_removeIteam.enabled = self.view.subviews.count>1;

}];

// 而在执行代码块之外，判断删除按钮是否允许点中是不对的，因为动画是在后台运行的，所以在执行动画的时候，就已经执行完判断语句了，而这时最后一个视图还没销毁掉，因此删除按钮永远不会不允许选中，也就不能在判断删除按钮是否允许点中。

// \_removeIteam.enabled = self.view.subviews.count>1;

删完最后一行之后，正确的效果。

Macintosh HD:Users:whYzZ:Desktop:经典错误:屏幕快照 2013-11-26 下午05.13.41.png

删完最后一行之后，错误的效果。原因，判断的位置放错了。

Macintosh HD:Users:whYzZ:Desktop:经典错误:屏幕快照 2013-11-26 下午05.12.21.png

## 4.Xib描述视图的时候，已经固定好描述视图的宽高了，外界调用视图的时候，只需要设置x，y值就好了

1.出现的问题，创建xib描述的视图时，将宽度设置为一个按钮的宽度了，导致删除按钮不能点击。



#pragma mark 添加联系人

- (IBAction)AddPerson:(UIBarButtonItem \*)sender {

NSString \*imgName=[NSString stringWithFormat:@"01%d.png",arc4random\_uniform(9)];

NSString \*labelName = arr[arc4random\_uniform(arr.count)];

RowView \*rowView = [RowView rowViewWithIcon:imgName name:labelName];

UIView \*lastView = [self.view.subviews lastObject];

int nextY = lastView.frame.origin.y + kSpace + kItemHW ;

// 设置rowView的位置和尺寸

CGRect cg =CGRectMake(0, nextY, kItemHW, kItemHW);

rowView.frame=cg;

[self.view addSubview:rowView];

}

错误原因：设置rowView的宽度为kItemHW，因此会有以上图片的出现。

错误会导致删除按钮不能点击，原因：父视图的尺寸不够，即父视图能接收事件的尺寸只有一点点，也就导致超出父视图尺寸的子视图不能监听点击事件。还有一点需要注意，将子视图添加到父视图尺寸之外的位置，只要还在屏幕上就会显示子视图，只不过它不能接收任何事件。

解决方法：CGRect cg =CGRectMake(0, nextY,rowView.frame.size.width , kItemHW);

  这样设置就好了，因为xib里面已经设置了rowView的尺寸了，外界不需要更改视图的宽度了，直接获取视图的宽度即可。

正确效果：

## UIToolBar问题

注意UIToolBar中不能使用viewWithTag这个方法，获取UIToolBar里的子视图。因为UIToolBar里的子视图都是UIBarButtonItem,而UIBarButtonItem是继承NSObject的，因此不能使用viewWithTag获取UIToolBar里的子视图，

viewWithTag：实现原理

- (UIView \*)viewWithTag:(NSInteger)tag

{

// 1.如果当前tag和当前视图tag相同，直接返回

if (self.tag == tag) return self;

// 2.如果和当前视图tag不相同，遍历当前视图的所有子控件，查找对应的tag。

for (UIView \*view in self.subviews) {

// 3.如果view不是UIView类或者UIView的子类直接返回nil

if (![view isKindOfClass:[UIView class]]) return nil;

if (tag == view.tag) {

// 4. 返回有相匹配的视图

return view;

}

}

// 5.如果都没有找到，返回nil.

return nil;

}

# 11.27常见问题

## 1.结构体和对象问题

// 这样定义是错的，结构体不是对象，声明变量是不需要加\*

CGRect \*frame = self.view.frame;

结构体变量正确定义：

CGRect frame = self.view.frame;

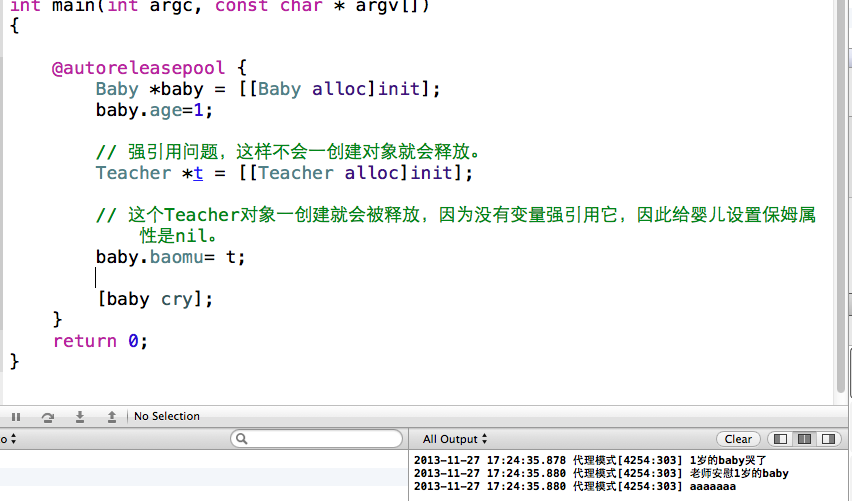
CGPoint center = self.view.center;

CGSize size = self.view.frame.size;

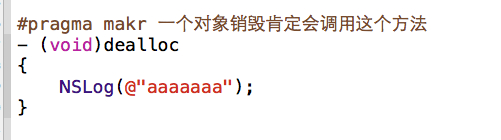
## 2.内存管理问题

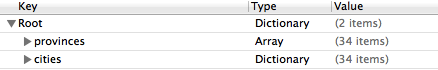
错误打印：



正确打印:  


当对象被销毁，一定会调用的方法，可以用这个方法，判断对象在什么时候销毁,用这个调试。





## 3.创建模型的时候，尽量自定义一个工厂方法供外界调用。

// 工厂方法，简化对象的实例化

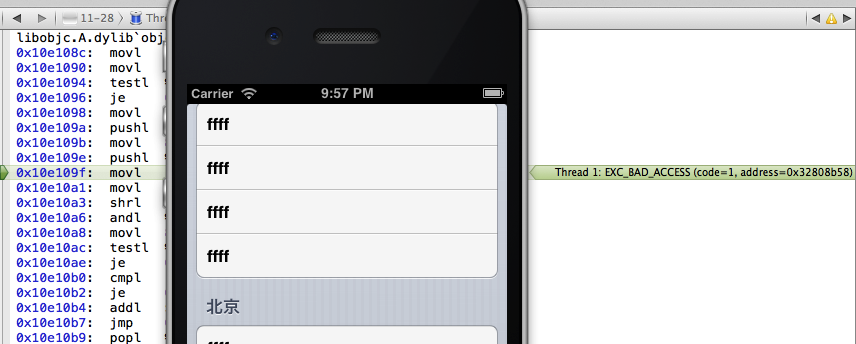
+ (id)provinceWithName:(NSString \*)name;

工厂方法好处：简化对象的实例化，快速创建对象。

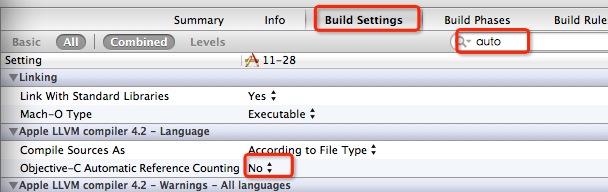
## 4.非ARC内存管理问题。

有些同学在创建项目的时候忘记点ARC了，导致一些成员属性都莫名其妙的释放了。然后出现了一系列莫名其妙的错误。

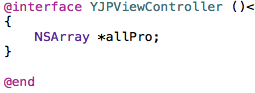
在滚动UITableView的时候出现野指针错误。



一出现这些野指针错误，首先应该想到某些对象被释放了，然后发现代码中，并没有什么造成对象被释放的情况，这时候应该马上想到很可能是非ARC弄的。下图为怎么查看项目是否是非ARC。



在非ARC中没有强引用的概念，因此下图的成员变量是没有被强引用的。



在看看下图，allPro数组没有通过alloc调用，没有调用alloc产生的对象都是自动释放的

allPro=@[

@{

kCities:@[@"浦东",@"杨浦",@"闸北",@"闵行"],

kHeader:@"上海",

kFooter:@"上海不错"

},

@{

kCities:@[@"海淀",@"昌平",@"天安门"],

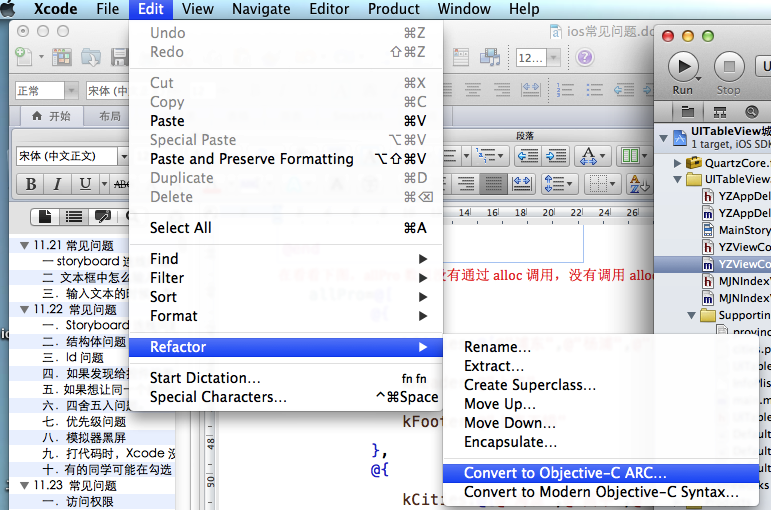
kHeader:@"北京",

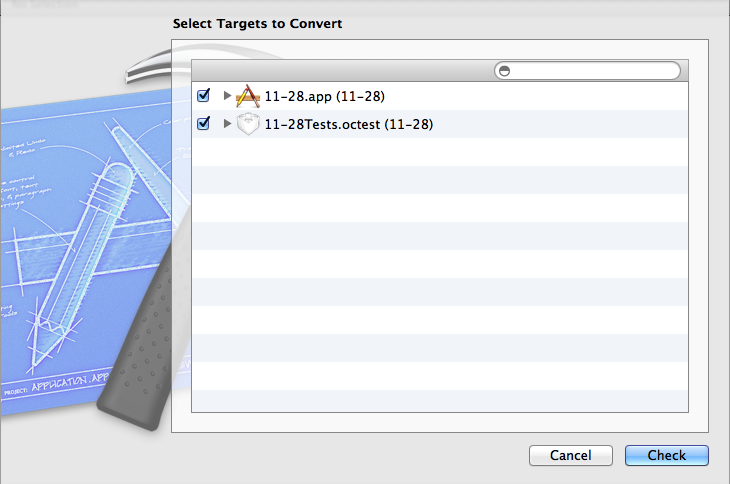
kFooter:@"北京很好"

}

];

所以在滚动的时候会出现野指针错误，因此需要将项目改成ARC。如下图





两个勾都得选中，然后一直点确认就OK了。

## 数据模型属性采用的策略中除了基本类型需要用assign，字符串需要用copy，其他对象类型都需要用strong。而控制器中视图采用的策略需要根据情况而定。

下图就是数据模型中属性用错了策略导致，UITableViewcell重新出现到界面时，会导致数据丢失。



# 11.29 常见问题

## 1.想在点击cell时做些操作，方法选错了。

下面两个方法太相似了，很容易选错。

// 当点击一行cell时，会调用这个方法

* (void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath

// 当取消选择一行cell时，会调用这个方法

* (void)tableView:(UITableView \*)tableView didDeselectRowAtIndexPath:(NSIndexPath \*)indexPath

## 2.字符串小调试技巧

当把一个字符串转换成基本数据类型时，字符串打印有值，但是转换成基本数据类型为0时，这个莫名其妙的问题，首先应该想到字符串中很可能有换行符合等，导致转换不成功。



NSString \*str = @" 123";

NSInteger i = [str integerValue];

NSLog(@"%@",str);

NSLog(@"%d",i);

碰到这种情况，可以在打印字符串的时候在占位符两边各加一个数字.

NSLog(@"1%@1",str);

然后看打印结果就能知道有字符串中有多少空行了



只要将空行去掉就能转换成功了。

字符串去掉空格的方法

// 此方法是通过什么字符集裁剪字符串。

- (NSString \*)stringByTrimmingCharactersInSet:(NSCharacterSet \*)set;

NSCharacterSet对象可以通过类方法创建

// 创建空格和换行字符集

[NSCharacterSet whitespaceAndNewlineCharacterSet];

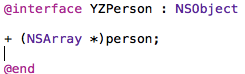
// 创建空格字符集

[NSCharacterSet whitespaceCharacterSet];

## 3.下图的警告意思是:初始化时，类型指向不匹配，应该用NSArray \* 而不是YZPerson \*



找到原因后，然后看person方法是怎么声明。



解决方式：将NSArray \* 改成id。

# 11.30 常见问题

## 1.UITableView数据源问题

1.1



错误原因：说YZViewController 没有实现 tableView:numberOfRowsInSection:

解决方式：实现tableView:numberOfRowsInSection:

1.2



这里返回空，会报错。

UITableView内部实现原理：

数据源实现了这个方法

-(UITableViewCell\*)tableView:(UITableView\*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath

tableView内部自动会调用以下方法添加cell。

[tableView addSubview:cell];

如果返回的cell为空，也就意味着生成下面一行代码。

[tableView addSubview:nil];

而addSubview是将右边参数添加到数组中保存起来，而数组是不能添加空值的。所有集合对象都不能出传空。例如数组，字典，NSSet。

以上错误总结：作为tableView的数据源必须实现两个方法。

返回行数

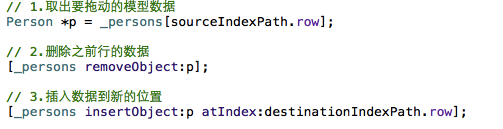
-(NSInteger)tableView:(UITableView\*)tableView numberOfRowsInSection:(NSInteger)section;

返回每一行显示的内容

-(UITableViewCell\*)tableView:(UITableView\*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath

另外返回每一行显示的内容不能返回nil。

## 2.在数组删除一个模型，并不代表把这个模型给释放了。



上面步骤二，仅仅是将模型从数组中销毁，而模型并没有被销。

# 12.1 常见问题

## 1.stroyboard中显示的跟根视图是UIView，而stroyboard的控制器是UITableViewController就会报这个错误

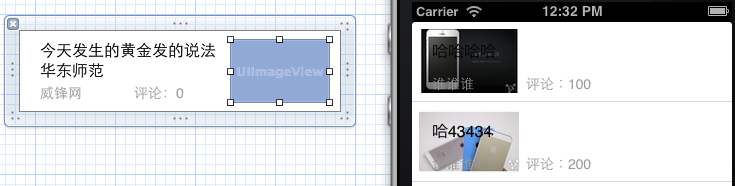


原因是：UITableViewController控制器不能加载UITableView，因为它会去加载stroyboard中的UIView。

解决方式：将stroyboard中的UIView改成UITableView

# 12.2 常见问题

## 1.自定义视图的属性命名冲突问题



当发现自己描述的xib和运行的时候展现出来的不一样的时候，这时候已经想到自己命名的属性名称和系统命名的冲突了。

错误原因

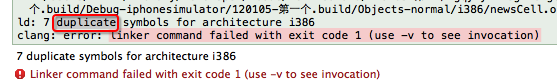


系统自带的UITableViewCell中也有imageView这个属性，因此冲突了。

解决办法：将自定义视图的imageView属性名称改成iconView.

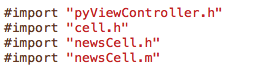
注意：以后自定义属性命名不要和系统自带的属性名称相同。

## 2.链接错误



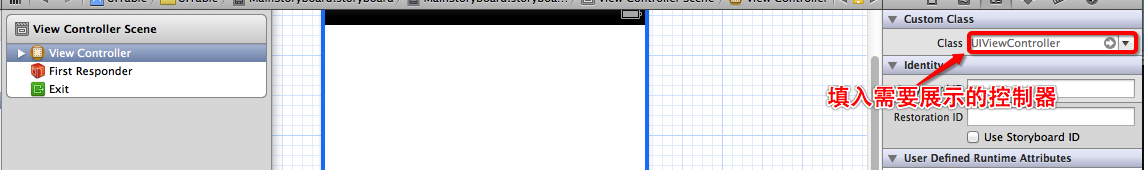
以后看到duplicate这个词语，错误原因就是重复定义了类,函数方法等等。

一般都是因为导入了.m文件

错误：

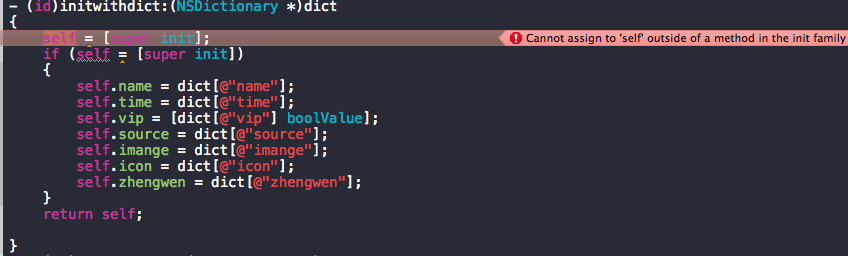
解决方式：将#import "newsCell.m"这一行删掉。

## 3.注意将之前storyboard中控制器删除之后，拖入一个新的控制器的时候，stroyboard中控制器的class也要重新填入自己想要展示的控制器，告诉stroyboard去加载哪个控制器。



# 12.3常见错误

## 1.初始化方法命名规范问题。



看见这个错误，应该要想到初始化方法命名错误的问题。因为self只能在init开头的方法中赋值，init必须是一个独立的单词，因此init后的第一个字母必须大写。

# 12.4常见错误

## 1.重写set方法忘记赋值，以后重写set方法，第一步就先赋值。

## 2.strong和weak乱用

一般情况：代理和控件使用weak

其他对象使用strong

基本数据类型 使用assign

# 12.5常见错误

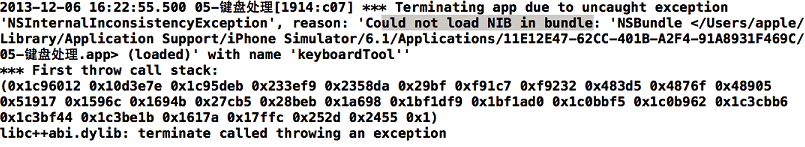
## 1.QQ好友列表中，展开了列表，但是箭头没动画。

原因：由于代理方法中重新刷新了表格，也就意味着把之前的头部视图给替换了，因此没有动画了，不要创建新的头部，才能让旧的头部执行动画

解决方法：用一个数组或者字典保存所有的头部视图，重新刷新的时候，直接取就OK了。

# 12.6 常见错误

## 1.加载xib时，名字是分大小写的，注意名字一定要保持一致。



报错原因：xib是大写的KeyboardTool，而加载的时候名字写成小写了。

解决方式：将加载的xib的名称改成大写。

