让不懂编程的人爱上iPhone开发(2013球iOS7版)-第6篇

欢迎继续回来一起学习iPhone开发。

在之前的学习中，我们已经学会了如何把滑动条的数值保持在变量中，然后将其显示在提示对话框中。这个算是不小的进步了，不过我们还可以做的更好。想想看，如果你想把xib界面里面的滑动条初始数值设置为1或者100，又该怎么办呢？currentValue这个时候的数值就是错的，因为在viewDidLoad方法中始终假定它的值是50。
怎么办？最简单的方法似乎是回到viewDidLoad方法，然后给currentValue赋予一个新的初始值。这样似乎是理所当然的。不过当你的项目越来越大时，里面会有几个，十几个甚至更多的视图控制器，你不可能一个个去手动修改。更可怕的是，你很可能忘了还有这个事情要去处理。
所以我们最好用下面的方法来处理：
回到Xcode，点击CrazyDragViewController.m，找到viewDidLoad方法，修改其中的代码如下：

```
- (void)viewDidLoad
{
   [super viewDidLoad];
   currentValue = slider.value;
}
```

为什么这样来处理呢？因为这里我们会直接获取xib界面中滑动条的初始值（不管是50，1还是100），然后把它作为currentValue的初始值。
你可能已经看到红色的错误提示了，Use of undeclared identifier ‘slider’

肿么会这样？因为viewDidLoad方法对slider这个东西一无所知。
且慢，之前在sliderMoved方法里面似乎用到过吧？不信你来看：

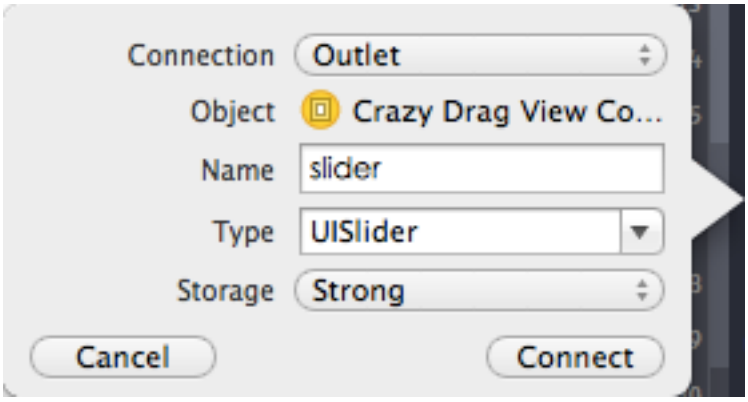
```
- (IBAction)sliderMoved:(id)sender {
-
   UISlider *slider = (UISlider*)sender;

   currentValue = (int)lroundf(slider.value);

}
```

这里我们同样用到了slider.value，不同的是这里的slider指的是sliderMoved这个动作的参数。当我们创建完一个动作方法，然后在Interface Builder里面把它关联到一个界面控件

时，可以指定我们希望获取一个到该控件对象的引用，从而和这个动作方法一起作为消息来发送。如果我们要在方法中添加到该对象的引用，或者说我们可能会对多个界面控件对象使用同一个动作方法的时候，这一点就非常有用了。
在我们这个例子里，当执行sliderMoved动作时，slider包含了到正在被我们移动的真实滑动条对象的引用。这个UISlider滑动条对象会说，“视图控制器你好~我是一个滑动条对象，刚才我被人动了。顺便说一下，这是我的电话号码，你可以通过它来联系我。”此时的slider变量包含了“电话号码”，可惜的是它只在这个特定的方法中存活。换句话说，它是一个本地变量。一旦方法执行完毕，它的生命也就终结了。
真正有效的方法是将滑动条的引用保存到一个实例变量中，正如我们对currentValue所做的那样。只不过这一次的变量数据类型不是int（整数），而是UISlider(滑动条对象）。而且我们不是用的常规实例变量，而是一个特殊的实例变量，在Objective-C语言里面又叫property(属性）。

回到Xcode，打开Main.storyboard，按照和之前一样的操作打开Assistant Editor，然后选中滑动条，按住Control键，同时用鼠标左键拖出一根线到辅助编辑器的花花绿绿的代码中，只不过终点要在-(IBAction)showAlert:(id)sender;这行代码之后和@end之前。这时会弹出一个小提示框，按照下图来填写内容：

点击connect，就会在刚才拖曳的终点处自动添加一行代码：



回到Xcode，点击CrazyDragViewController.h，在@interface和@end之间的某个位置添加下面的这行代码：
@property (strong, nonatomic) IBOutlet UISlider *slider;

这样就可以了吗？还不行。切换回CrazyDragViewController.m，你会看到红色的代码错误提示依然存在，只不过现在是,'Use of undeclared identifier 'slider';did you mean '_slider'?
这又是什么意思呢？
让我们在@implementation这段代码的下面添加一行代码：

```
@synthesize slider;
```

这个时候CrazyDragViewController.m这个文件顶部的代码应该是下面这样的：

```
#import "CrazyDragViewController.h"

@interface CrazyDragViewController (){

    int currentValue;

}
- (IBAction)sliderMoved:(id)sender;
@property (strong, nonatomic) IBOutlet UISlider *slider;

- (IBAction)showAlert:(id)sender;
@end
```

此时错误提示就消失了。
让我们来回顾下添加一个和Storyboard界面对象相关联的属性变量的步骤：
1.在Interface Builder里面创建outlet连接
2.在对应viewcontroller的.m文件里面添加@synthesize语句。

如果使用xib，那么就需要三个步骤，这里暂且不管。

@synthesize这行代码将会帮我们在视图控制器自动添加一些代码，从而让我们可以正常使用属性。如果不添加@synthesize代码，就会出问题。

现在我们已经完成了所需的设置工作，在视图控制器的任何一个地方都可以使用slider属性来获取对滑动条对象的引用。
更改viewDidLoad方法的代码如下：

```
- (void)viewDidLoad
{
   [super viewDidLoad];
      // Do any additional setup after loading the view, typically from a nib.
```

```
    currentValue = self.slider.value;
}
```

这里的唯一的变化是在slider的前面添加了关键词self。这就是我们使用属性变量的方式。self 关键词允许视图控制器获取对自己的引用。其实这个概念一点也不奇怪。当我们说，“我想见苍老师”的时候，这里的“我”指的就是你自己。同样的，对象也可以用self来表示自己。
这里的self指的是视图控制器。self.slider指的是视图控制器里面的slider属性。而self.slider.value则指的是滑动条的数值，它恰好也是一个属性（UISlider对象的属性）。
好吧，这就是术语狂口中的点表示法，使用点来区分两个或多个名称。每当你看到something.somethingElse的时候，就表示我们用到了属性。
现在，无论你在Interface Builder中为滑动条设置的初始值是多少，只要应用被开启，currentValue就会自动对应这个设置值。

ok了，点击Run按钮运行应用，直接触碰按钮，它会告诉你滑动条的当前数值是50。关闭应用，切换到Interface Builder，更改滑动条的初始值，改成你喜欢的某个数值。再次运行应用，触碰按钮，你会看到提示对话框中的数值就是你更改后的初始值。

作业：你可以不添加self,然后运行应用看看实际的效果

不过这还没完，你会在Xcode的代码编辑区看到黄色的警告。’Local declaration of ‘slider’ hides instance variable’.

之所以会看到这样的警告信息，是因为当我们添加一个名为slider的属性时，系统会自动为我们创建一个同样名为slider的实例变量。而这一切要归功于@synthesize这行代码。
在Objective-C编码中，苹果认为某个本地变量的名称和实例变量的名称相同是很可怕的一件事情。因为本地变量的优先级要高于实例变量，从而会‘hide’实例变量。如果你对这种命名叠加现象不管不顾，很可能就给自己的产品留下了隐患。而这样的bug是很难被发现的，所以Xcode第一时间给了你黄牌警告。

科普：关于错误和警告
在开发iOS应用时，Xcode有时候会给你发黄牌警告，严重的则会直接红牌驱逐出场。黄色的警告信息出现后，你还是可以运行自己的应用，只不过在未来的某个时候会发现自己

的应用莫名崩溃。当你看到红色的错误提示时，就表示你犯的错是致命的，应用根本就没办法运行。
通常情况下，作为程序猿，我们要确保代码中没有一行错误。但很多程序猿会选中忽略代码中的黄色警告，因为它们看起来不会让程序直接崩溃。不过我个人的原则是，尽可能消灭一切的黄色警告。当然，这个也不是绝对的，在某些情况下我们也需要容忍黄色的警告。具体来说，当代码中用到了第三方的类库时，因为第三方类库未必兼容最新版本的iOS，就会导致大量的警告信息。最典型的如cocos2d，每次iOS版本更新后，cocos2d的旧版本并不能保证第一时间支持最新的iOS类库。此外，当我们使用github，google code上的一些开源项目时，也会出现类似的情况。对于此类的警告，我们有两种处理方式。一种是选择使用较低版本的iOS，以消除警告；一种是等待最新版本的第三方类库，以消除警告。还有第三种方式就是自己手动更改，但如果是cocos2d这样的规模较大的第三方类库，就必须慎重了。
但即便是此类情况所造成的警告，程序猿也应该在出现警告的地方说明原因，以及未来可能的解决方法，以及可能造成的bug，以便测试的时候来确认和改进。

如果你是产品经理，在程序猿提交的源代码中发现了大量的黄色警告，而且没有在出现警告的地方注释说明是什么原因造成的，就一定要打回去让他们仔细再看看。不要被一大堆专业术语忽悠，不行就是不行，没有什么废话。
0错误，0警告，是程序猿所必须追求的事情。即便当前版本不能实现，也一定要在后续的版本中纠正，而不能放任不管，最终酿成大错。

回到刚才的代码，对这个命名重复的警告该如何处理呢？很简单，只需要更改本地变量slider为其它名称就好了。注意，通常我们选中更改本地变量名称，而不是去更改实例变量或者属性变量的名称。
让我们修改sliderMoved方法的代码如下：

```
- (IBAction)sliderMoved:(UISlider*)sender {

   currentValue = (int)lroundf(sender.value);

}
```

需要注意的是，这里我们改动了两个地方，将方法名里面的id更改为UISlider*，从而告诉程序对象是UISlider类型的，然后将方法体里面的slider改为sender，从而避免了变量名的冲突。

到了这一步，所有的警告信息都应该消失了。

在Xcode中点击CrazyDragViewController.m，把sliderMoved这个动作方法的定义（在@interface和@end之间）修改为：

```
- - (IBAction)sliderMoved:(UISlider*)sender;
```

这里我用的是sender,这个实际上是动作方法常用的标准参数名称。你也可以选择使用其它的名称，具体用哪个就看你自己的爱好了，只要不和实例变量或属性变量的名称冲突就好。随着编程经验的增加，你会发现，每个程序猿都会有自己的风格。只是，你最好让别人，或是2个月后的自己能够看懂自己的代码。
我们的每一点进步都不容易啊。重要的不是写了几行代码，做了几步操作，而是在这个过程中真正学会程序猿的思维方式。

又见科普：属性vs实例变量

现在来看看这两个概念吧。属性和实例变量实际上有很多类似之处。实际上，当我们使用@synthesize这行语句来创建属性的时候，也默认创建了一个实例变量(ivar)。这就意味着我们的slider属性实际上是保存在名为slider的实例变量中，而这个实例变量是由Xcode中的Objective-C编译器自动添加到视图控制器中的。
区别属性和实例变量的方法很简单，因为在用到属性的地方几乎都要用到self关键词：

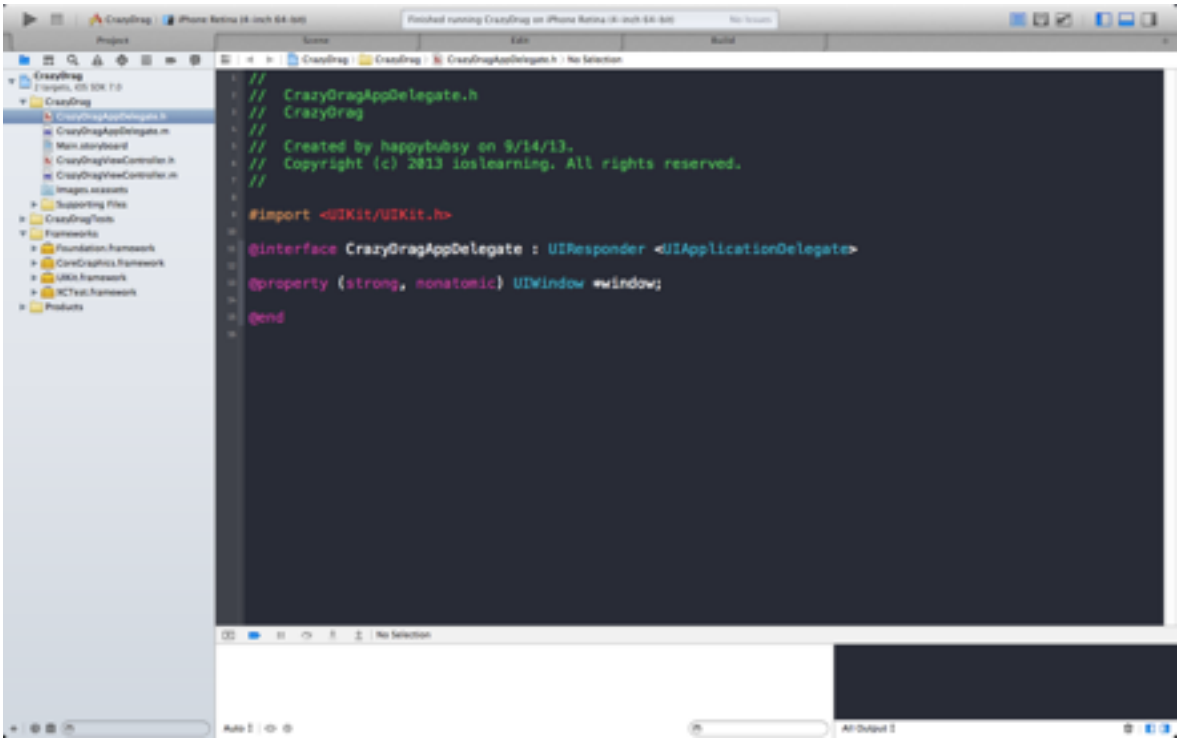
```
// 属性:
self.slider.value = 50;

// 属性“背后”的实例变量:
slider.value = 50;
```

那么属性和实例变量的区别究竟在哪里？好吧，这里又要重温代码世界的生命周期论了。在Objective-C的代码世界里，实例变量依附于创建它的对象，只有对象中的其它生命才能和它交换信息。其它对象如同其它平行宇宙时空中的生命体，无法了解和使用这个实例变量。属性则不同，其它对象可以通过平行宇宙之间的超时空管道来获取它的信息。
这就是我们之前在Interface Builder中所做的事情：我们把slider对象关联到视图控制器的slider属性上，但却无法和视图控制器的实例变量关联起来。因为Storyboard界面中的控件和视图控制器实际上是两个不同的世界。

关于这一点我们在后面的教程中还将反复说明，因为这个概念在面向对象的编程中非常重要。

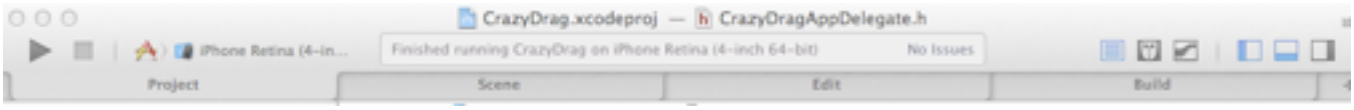
在结束之前，再附上一个使用Xcode的小技巧，相信对新老朋友都会有所帮助。这个小技巧是在WWDC2013上由苹果的某个美女程序媛分享的。
假定你使用的是Macbook Air等屏幕相对比较小的Mac电脑，那么会经常感觉自己的屏幕空间太小，特别是有多个视图控制器的时候。
那么Xcode提供了一个类似Safari 浏览器Tab的概念。也就是可以在一个界面中打开多个子界面，然后可以轻松跳转而互不影响。

具体是怎样的呢？请看下图。

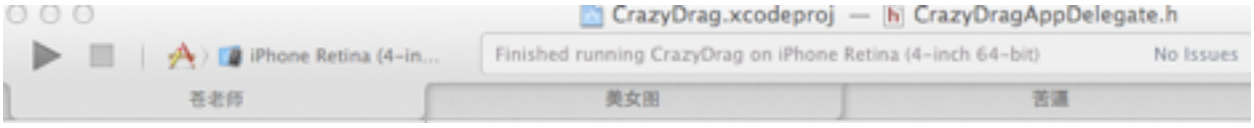


首先，我会让整个Xcode界面全屏显示，从而focus在开发之中，而不受其它程序的影响。

其次，我使用Tab建立了多个子界面，分别用于项目导航，界面编辑和代码编辑。

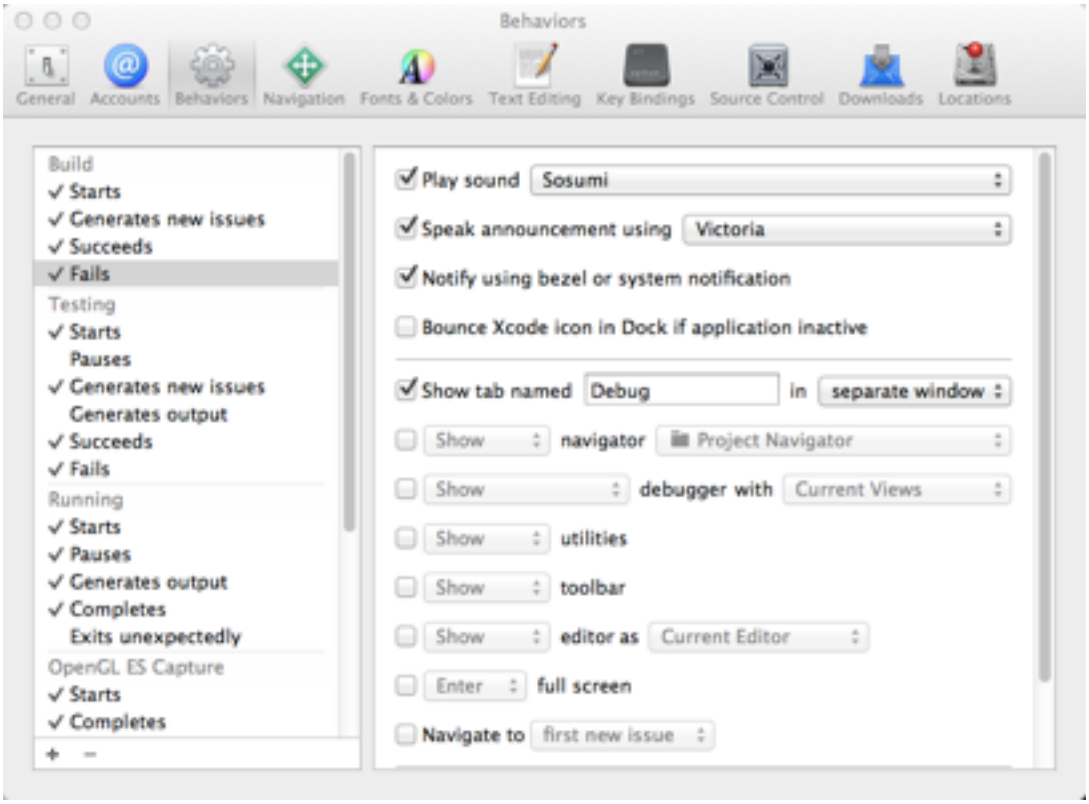


注意到这里的Tab就和浏览器打开的不同网页一样可以随意跳转而不受影响，而且名字是可以自己随便取的。



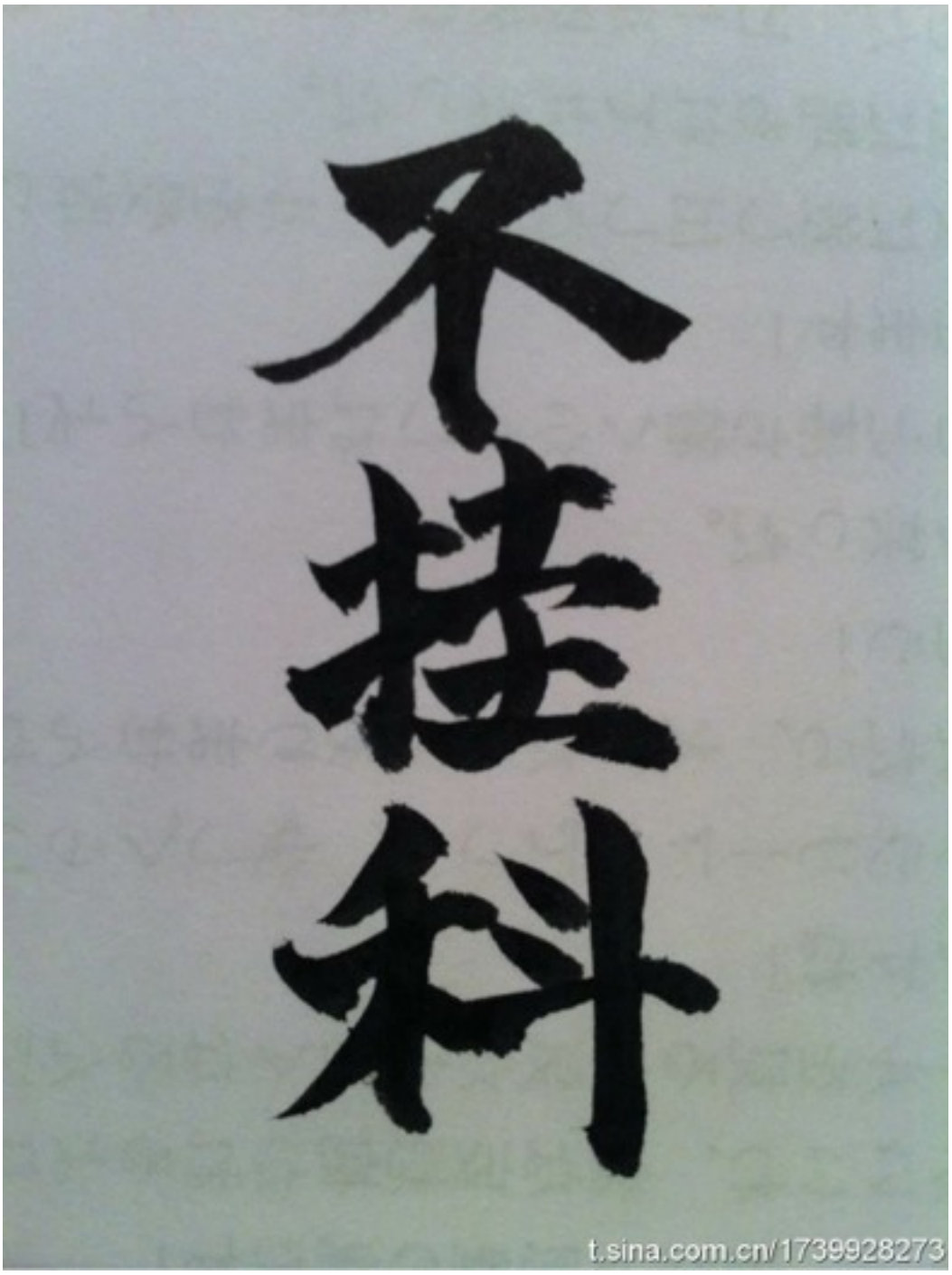
这样一来，其实也会给编程增加了小小的乐趣。

3.通过在Xcode的Preferences-->Behaviors里面设置还可以设置动态的Tab，比如在进行编译的时候单独打开一个Debug的子界面。



工欲善其事必先利其器，使用Tab来尽可能利用屏幕空间，在代码注释区写小说，就和之前设置代码区的背景和字体一样，看似对编程没有任何实际帮助，但实际上这些不起眼的举动至少会让你的编程工作没那么枯燥，甚至多一些意想不到的乐趣。当然，在代码注释区写小说神马的还是算了，毕竟会影响到别人。不过如果是你自娱自乐或者学习的话，完全可以这么做。

ok,在一大堆概念之后，又到了发送福利的时间。


不接科
t.sina.com.cn/1739928273