Displaying Content Using

UICollectionView

 使用collectionView来进行页面展示

Now that you understand how collection views fit within an iOS app using the Model-

View-Control (MVC) paradigm, it’s time to get to the good stuff: code. This chapter starts off easy and shows how you can use storyboards or .xibs to set up collection views, and then it shows you how to set them up in code. Collection views extend their UIScrollView superclass,so the chapter takes a brief det our to show how to use that to your advantage with UIScrollViewDelegate. You begin customizing actual content to show to your users using cell reuse before finishing off with a case study on performance.

现在你明白如何手机视图来适应一个iOS应用程序使用模型-视图控制器(MVC)模式，是时候来研究collectionView这个好东西了:代码。本章的开始展示了如何简单的使用故事板或xib文件设立集合视图,然后向您展示如何在代码中设置它们。集合视图是UIScrollView的子类,所以这一章做一个简要的介绍，我们展示如何使用collectionView与如何使用UIScrollViewDelegate对你有利。你开始定制实际内容展示给用户利用item重用机制在完成对性能的案例研究之前。

Setting Up Using Code and Storyboards

Traditionally, .xib files were used to lay out interface code for OS X and iOS apps. These files are “freeze-dried” versions of your interface that are thawed at runtime. The benefit of .xibs is that they’re easy to use to create basic interfaces; you usually have one instance of UIViewController per .xib.

Storyboards, first introduced in iOS 5 in 2011, enable developers to visually lay out the interaction between view controllers. Not only can developers visualize the connections between view controllers, but they can also define how their entire application transitions from one view controller to another. The key thing about storyboards is their efficiency; a huge .xib file, which has to be completely loaded into memory, can delay the time it takes for your app to launch. Storyboards efficiently lazy-load only the view controllers necessary.

使用代码和传统故事板

设置xib文件被用来制定为OS X和iOS应用程序界面代码。这些文件是“冷冻”版本的界面在运行时解冻。xib的好处是很容易使用它来创建基本界面;你通常为每个xib实例化一个UIViewController。

故事板,于2011年首次引入iOS 5,使开发人员能够直观地展示视图控制器之间的互动。开发人员不仅可以可视化视图控制器之间的连接,但他们还可以定义整个应用程序如何转换从一个视图控制器到另一个。故事板的关键之处在于他们的效率,一个巨大的,xib文件,必须是完全加载到内存,可以推迟你的应用程序启动的时间。故事板有效地延迟加载必要的视图控制器。

Of course, anything you can do in a .xib file or storyboard can be done using cold, hard code. If you are integrating collection views into your existing application, which uses .xib files or storyboards, it might be convenient to continue to use them. However, because collection views require the use of code for layout, it’s often easier to avoid using .xibs and

storyboards altogether. Nevertheless, this chapter explains how to set up the collection view from the last chapter using a storyboard and then set it up again using only code.

Create a new Xcode project with the Single View template. Make sure that Use Storyboards is checked. Open the MainStorboard.storyboard file and delete the view controller that’s already there. Drag a Collection view controller from the object library in the right pane onto the empty canvas, as shown in Figure 2.1.

当然,任何你可以通过xib文件或故事板文件完成的，使用冷,硬编码依然可以完成。如果你集成CollectionView到您的现有应用程序中,它使用的是xib文件或故事板,继续使用它们将会更加方便。然而,由于collectionView需要使用代码来设置layout布局,xib和故事板文件通常更容易避免使用。然而,这一章解释了如何设置集合视图使用故事板从最后一章,然后再设置它仅使用代码。创建一个新的Xcode项目与单一视图模板。确保使用故事板检查。打开MainStorboard。故事板文件和删除已有的视图控制器。拖动对象库的集合视图控制器在右窗格中空白的画布上,如图2.1所示。

igure 2.1 
Basic collection view using storyboards 

You could run the app right now and it would work, but it would be pretty boring. The storyboard has set up the delegate and data source outlets of the collection view to point to your collection view controller. The next step is to customize what that view controller actually does. This part is easy, because you’re just going to copy the existing code from Chapter 1, "Understanding Model-View-Controller on iOS."

Open the header for your view controller and change which class it inherits from (by changing UIViewController to UICollectionViewController). Then copy the implementation file in its entirety from the last chapter. The last, important step is to tell your storyboard which view controller it should use. Click the Collection view controller in the storyboard and open the Identity Inspector. Where it says Class, you see the default placeholder of UICollectionViewController. Boring! Replace that with the name of your view controller—in my case, it’s AFViewController.

This step is crucial; it’s how the storyboard knows what code to execute when laying out the collection view. Run your app, and you see the same output as from Chapter 1.

你现在可以运行这个程序,并且它会工作,但它会很无聊。故事板已设置了collectionView的委托和数据原属性斌并且指向了集合视图控制器。下一步是定制视图控制器实际上做什么。这部分是很容易的,因为你只需要复制现有代码从第一章,“理解模型-视图-控制器在iOS“。

打开你的视图控制器的头文件和改变这类所继承的对象(通过改变UIViewController将其变为UICollectionViewController)。然后复制实现文件的最后一章。最后,重要的一步是告诉你的故事板应该使用哪个视图控制器。在故事板中单击集合视图控制器和属性监视器。说类,你看到UICollectionViewController的默认占位符。无聊!代替你的视图控制器的名称我而言,AFViewController。这一步是至关重要的,它是故事板知道布局集合视图时的执行代码。运行您的应用程序,您将看到从第一章相同的输出。

Using storyboards or .xibs, you have an opportunity to change the visual display of the collection view without any code. Select the collection view in the storyboard and open the Attributes Inspector. Here, you can change the scroll direction of the collection view from Vertical, the default, to Horizontal. You can also change properties of the collection view that belong to its superclass, UIScrollView. Change the Style to white, which makes the scroll indicator visible against the black background.

Open the Size Inspector, and you can change the attributes of the collection view layout, shown in Figure 2.2. (Collection views abstract these properties to their layout objects; read more on that in Chapter 3, “Contextualizing Content.”) Here, you can change the cell size, which is 50 by 50 points by default. Bump the width down to 20 and keep the height set to 50. The header and footer sizes don’t work just yet because you haven’t used headers or footers.

使用故事板或xib,你有机会改变集合视图的视觉显示而不需要使用任何代码。在故事板中选择集合视图并打开属性检查器。在这里,你可以改变CollectionView在默认情况下垂直方向的滚动,将其变成水平方向滚动。您还可以更改集合视图属于它的父类的属性,即UIScrollView。改变原有的白色风格,使滚动指示器在黑色背景下可见。

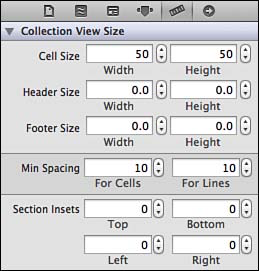


Figure 2.2 Size Inspector of a collection view layout

打开尺寸检查器,您可以更改的属性集合视图布局,如图2.2所示。(这些集合视图抽象布局对象的属性;更多内容将在第3章进行介绍,)。在这里,你可以改变cell的大小,默认是50 \*50像素。cell之间的间距是20像素和保持高度设置为50像素。页眉和页脚默认情况下并不工作,因为你没有使用页眉或页脚。

You can change the distance between cells in the collection view using the Min Spacing section in the Size Inspector. This is only the minimum distance; the default layout, called Flow, makes sure that cells are a minimum distance from one another. The Section Insets area of the Size Inspector enables you to specify the distance surrounding an entire section. (Remember that you only have one section so far.) You take a closer look at section insets in Chapter 3, so don’t worry about the specifics for now. It’s a personal pet peeve of mine to have too small a margin around content, so bump up the section insets to 10 points each. Run the app to see the visual differences in the collection view. It should resemble

Figure 2.3.

你可以在尺寸检查器中使用Min Spacing section这个属性改变集合视图中cell之间的距离，这仅仅是最小距离;默认布局,称为流水布局,它为了确保cell之间有一个最小的距离。尺寸检查器的Section Insets使您能够确定cell的整体布局距离周围的整个部分的间距。(记住,你只有一个部分。)你仔细看看部分在第三章的Section Insets你就会非常明白,所以不要担心现在这些细节。我个人的看法在内容中是留一个特表小的间距环绕内容，因此每次将section insets提高到10像素。运行该应用程序的集合视图的视觉差异。它应该类似于图2.3。

 。 ロ 0 ロ ロ 
一 口 ロ 0 
一 口 ロ 0 ロ 0 
一 口 ロ ロ ロ 
一 ロ 00 ロ 
ロ ロ ロ ロ 0 

Figure 2.3 Changes made with storyboards

Not bad at all. Don’t worry that the status bar is visible in front of our content; that is the default on iOS 7. We’ll solve this problem later by placing our Collection view controller inside of a navigation controller. The problem with Figure 2.3 is that only some of the properties of a collection view layout are accessible with storyboards or .xib files. In addition, if you override the properties you’ve set in a storyboard in code, or you forget that you’ve set something in the storyboard, it can lead to a debugging headache. For this reason, I strongly prefer to use a code-only approach with collection views.

很不错不是吗。别担心在我们内容的前面状态栏是可见的;这在iOS 7.0以后是默认的。我们解决这个问题是通过将集合视图控制器放在导航控制器的内部。图2.3的问题是,只有一些属性的集合视图的布局可以使用故事板或.xib文件访问。此外,如果你使用代码覆盖掉那些你在故事版中设置的属性,或者你忘记了在故事板中设置了某些属性,它可能导致你调试的时候非常头痛。出于这个原因,我强烈倾向于使用-代码的方法来使用CollectionView。

Now you can re-create your interface using only code. Create a new Xcode project with the Empty Application template. (For anyone who has never created an app from an empty template, this can be a big step.) Create a new file using File, New, File or ⌘N. Select Objective-C Class and call it something like AFViewController. In the field for Subclass, enter UICollectionViewController. Make sure not to select With XIB for User Interface.

现在你可以重新创建你的界面仅使用代码。创建一个新的Xcode项目与空应用程序模板。(对于那些从来没有创建一个应用程序从一个空的模板,这是一个很大的一步。)使用文件创建一个新文件,新文件或⌘N。选择objective - c类,称之为AFViewController之类。在子类的字段,输入UICollectionViewController。确保不要选择与生成XIB的用户界面。

Open the application delegate implementation file and add an #import statement to import the new view controller’s header file. Change the implementation to look like the code in Listing 2.1.

打开应用程序委托实现文件并添加一个#导入语句导入新视图控制器的头文件。更改实现像清单2.1中的代码。

Listing 2.1 Setting Up the Application

#import "AFAppDelegate.h"

#import "AFViewController.h"

@implementation AFAppDelegate

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

self.window = [[UIWindow alloc] initWithFrame:

[[UIScreen mainScreen] bounds]];

UICollectionViewFlowLayout \*collectionViewLayout = [[UICollectionViewFlowLayout alloc] init];

collectionViewLayout.scrollDirection = UICollectionViewScrollDirectionHorizontal;

collectionViewLayout.sectionInset = UIEdgeInsetsMake(10, 10, 10, 10); collectionViewLayout.itemSize = CGSizeMake(20, 50); self.window.rootViewController = [[AFViewController alloc]

initWithCollectionViewLayout:collectionViewLayout];

self.window.backgroundColor = [UIColor whiteColor]; [self.window makeKeyAndVisible];

return YES;

}

Next, open the view controller’s implementation file and add the following line to the viewDidLoad method (see Listing 2.2).

Listing 2.2 Setting the Scroll Indicator Color

接下来,打开视图控制器的实现文件,并添加以下行viewDidLoad方法(参见清单2.2)。清单2.2颜色设置滚动指标

-(void)viewDidLoad

{

[super viewDidLoad];

//All that other stuff

self.collectionView.indicatorStyle = UIScrollViewIndicatorStyleWhite;

}

Build and run the app, and you see that everything you customized using storyboards has been replicated using just code. High five!

Before you dive deeper into collection views and laying out content, the following section takes you on a quick diversion to discuss UIScrollView.

编译并运行应用程序,你会发现所有你定制使用故事板已经仅使用代码得到一样的结果。前方高能！哈哈哈，在你深入研究集合视图和布局内容之前，以下部分需要你快速转移参与到UIScrollView的讨论中。