Listing 1.1

Basic UICollectionViewController Header File

@interface AFViewController : UICollectionViewController

@end

Replace AFViewController with the name of your view controller. My initials are AF, so

I prefix my class names with them to avoid namespace collisions.

Next, head over to your .xib file and delete the view. Drag a collection view onto the blank canvas and connect the collection view’s delegate and dataSource outlets to the File’s Owner, the view controller. It should look like Figure 1.4 when you’re done.

 AFViewController换成你的视图控制器的名称。我的首字母是AF,所以我类名与他们避免命名空间前缀碰撞。接下来,去你的。xib文件和删除视图。将一个集合视图拖到空白画布并连接集合视图的委托和数据源文件的所有者,视图控制器。它的外观应该类似于图1.4当你完成。

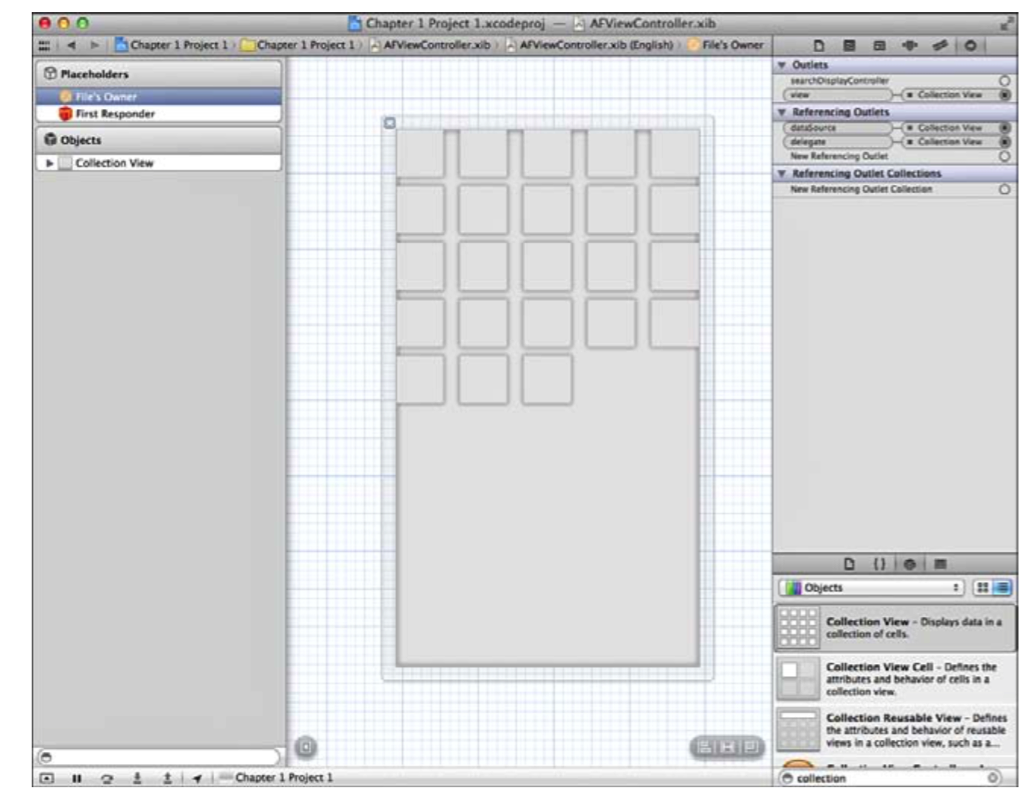


Figure 1.4 Basic UICollectionView setup using a .xib

Now comes the fun part: the code! UICollectionViewDataSource has two required methods. One returns the number of items in a section, and another configures a cell for a given index path.

If you’re not familiar with these terms, don’t worry. Chapter 2 explains everything in great detail. This quick example just gets your feet wet.

Following MVC, you need a model. Use a basic array that you'll populate with a bunch of randomly generated colors. The top of your implementation file should look something like Listing 1.2.

现在有趣的部分:代码!UICollectionViewDataSource有两个必需的方法。一个返回的物品数量部分,和另一个配置cell对于一个给定的索引路径。

如果你不熟悉这些术语,别担心。第二章详细地解释了一切。这个简单的例子只是让你的脚湿了。MVC之后,你需要一个模型。使用一个基本的数组,您将和一群随机生成的颜色填充。实现文件的顶部应该如清单1.2所示。

Listing 1.2 Setting Up the Model

static NSString \*kCellIdentifier = @"Cell Identifier";

@implementation AFViewController

{

NSArray \*colorArray;

}

- (void)viewDidLoad {

[super viewDidLoad];

[self.collectionView registerClass:[UICollectionViewCell class] forCellWithReuseIdentifier:kCellIdentifier];

const NSInteger numberOfColors = 100;

NSMutableArray \*tempArray = [NSMutableArray arrayWithCapacity:numberOfColors];

for (NSInteger i = 0; i < numberOfColors; i++)

{

CGFloat redValue = (arc4random() % 255) / 255.0f; CGFloat blueValue = (arc4random() % 255) / 255.0f;

CGFloat greenValue = (arc4random() % 255) / 255.0f;

[tempArray addObject:[UIColor colorWithRed:redValue green:greenValue blue:blueValue alpha:1.0f]];

}

colorArray = [NSArray arrayWithArray:tempArray]; }

Notice the copy of the array; we’re doing so to avoid a mutable instance as our color array, which would be unnecessarily slower.

The kCellIdentifier string is used to register a plain UICollectionViewCell as the cell for the collection view to use, so don’t pay much attention to it. The part that involves the model is the instance variable called colorArray. In viewDidLoad, you use a for loop to populate this array with random colors.

Now that you have the model set up, you need to configure your view to represent it. For this, use the two UICollectionViewDataSource methods mentioned earlier (see Listing 1.3).

注意数组的副本,我们这样做,以避免一个可变的实例作为我们的颜色数组,将不必要的慢。kCellIdentifier字符串用于注册一个普通UICollectionViewCell作为集合视图使用的细胞,所以不要太关注它。这部分涉及到模型实例变量被称为colorArray。在viewDidLoad,您使用一个for循环和随机颜色填充该数组。现在您已经建立的模型,您需要配置您的视图来表示它。为此,使用前面提到的两个UICollectionViewDataSource方法(参见清单1.3)。

Listing 1.3 Configuring the View

-(NSInteger)collectionView:(UICollectionView \*)collectionView numberOfItemsInSection:(NSInteger)section

{

return colorArray.count;

}

- (UICollectionViewCell \*)collectionView:(UICollectionView \*)collectionView cellForItemAtIndexPath:(NSIndexPath \*)indexPath

{

UICollectionViewCell \*cell = [collectionView dequeueReusableCellWithReuseIdentifier:kCellIdentifier forIndexPath:indexPath]; //Discussed in Chapter 2 - pay no attention

cell.backgroundColor = colorArray[indexPath.item];

Macintosh HD:Users:huanglaifeng:Library:Group Containers:UBF8T346G9.Office:msoclip1:01:A256FC65-365A-C846-83BD-397AB7243162.png

The first method—collectionView:numberOfItemsInSection:—lets the collection view know how many cells it’s going to display. You rely on the model to let the controller know what number to return. Next is collectionView:cellForItemAtIndexPath:, which returns a cell that you are responsible for configuring in a way that represents your model. To do this, you grab the model at the given index and use that color as the background color for the cell. If you run the app, you get something like what you see in Figure 1.5. Because the colors are randomly generated, of course, your app will look different.

第一个method-collectionView:number Of Items In Section:经常collectionView知道有多少cell就会显示。你依赖于模型让控制器知道返回数量。接下来是collectionView:cellForItemAtIndexPath:返回一个cell,你负责配置的方式表示你的模型。要做到这一点,你抓住指数给出的模型和使用颜色的单元格的背景颜色。如果你运行这个应用程序,你得到类似于图1.5你所看到的。因为颜色是随机生成的,当然,你的应用将会不同。



Figure 1.5 First run of the basic app

Note that we’re not using this collection view within a UINavigationController, so the status bar is transparent. In production code on iOS 7+, you’ll usually encapsulate your collection view within a navigation controller, whose navigation bar is extended behind the status bar.

图1.5第一次运行的基本应用请注意,我们不使用这个UICollectionView内UINavigationController,状态栏是透明的。在iOS 7 +生产代码,你通常封装UICollectionView在导航控制器里,导航栏的状态栏后面的扩展会在状态栏之后。

So, this simple example demonstrates how a model can represent a view and how you can configure a view to represent that model without either being aware of the other. This example demonstrates the platonic ideal of what you should strive for: clear separation between model, view, and controller.

因此,这个简单的例子演示了如何一个模型可以代表一个视图,以及如何配置一个视图来表示该模型没有意识到另一个。这个例子演示了你应该争取的柏拉图式的理想:清晰的分离模型,视图和控制器。