

通过UIActivityViewController实现更多分享服务

ios开发app

shaobo 1月4日发布

前言

我在[通过UIDocumentInteractionController预览和分享"史蒂夫·乔布斯传"](#)这篇文章中，详细讲了UIDocumentInteractionController的用途和使用方法。而在iOS 6 SDK中,苹果提供了UIActivityViewController来让我们可以使用更多地服务。这篇文章，我就来介绍一下怎么通过UIActivityViewController实现更多地服务。

简介

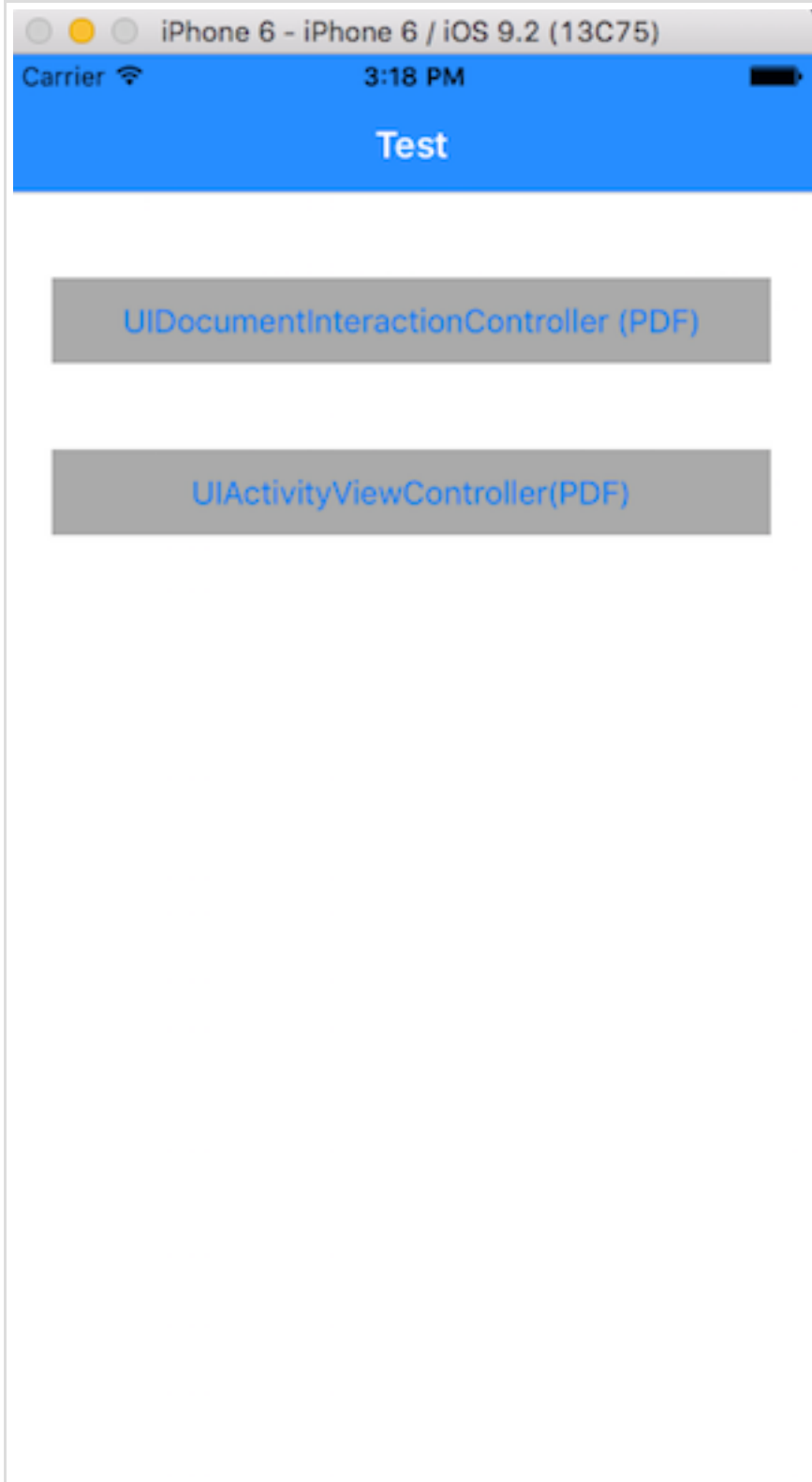
打开UIActivityViewController的API文档，我们可以看到UIActivityViewController的声明。

```
NS_CLASS_AVAILABLE_IOS(6_0) __TVOS_PROHIBITED @interface UIActivityViewController :  
    UIViewController
```

我们可以看出UIActivityViewController是在iOS 6开始支持的，同样是不能在Apple TV的开发中使用。而且UIActivityViewController是直接继承UIViewController的，这意味着我们需要自己来展示和解散视图。

准备

我使用在UIDocumentInteractionController测试中使用的Demo,GitHub地址是：[ZSDocumentInteractionTest](#)。然后添加一个新的Button作为UIActivityViewController的触发事件，运行程序，就可以看到下面的界面啦(过程自行想象，哈哈)



初始化

接着我们在Button的触发方法里面开始操作 `UIActivityViewController` 来提供服务。首先，我们需要初始化一个 `UIActivityViewController` 的实例， `UIActivityViewController` 提供了一个初始化方法：

```
- (instancetype)initWithActivityItems:(NSArray *)activityItems applicationActivities:
(nullable NSArray<__kindof UIActivity *> *)applicationActivities
NS_DESIGNATED_INITIALIZER;
```

官方文档对这两个参数有详细的解释：

参数	描述

activityItems	The array of data objects on which to perform the activity. The type of objects in the array is variable and dependent on the data your application manages. For example, the data might consist of one or more string or image objects representing the currently selected content. Instead of actual data objects, the objects in this array can be objects that adopt the UIActivityItemSource protocol, such as UIActivityItemProvider objects. Source and provider objects act as proxies for the corresponding data in situations where you do not want to provide that data until it is needed. Note that you should not reuse an activity view controller object that includes a UIActivityItemProvider object in its activityItems array. This array must not be nil and must contain at least one object.
applicationActivities	An array of UIActivity objects representing the custom services that your application supports. This parameter may be nil.

大概意思是这个方法接收俩个数组类型的参数，第一个数组内的对象代表的是我们想要操作的数据的一些表征，而且这个数组至少需要一个值，比如我们PDF文档的名称，URL；第二个数组指定了 **泛型**，数组内的对象必须是 **UIActivity** 类型的对象，代表的是iOS系统支持的我们自定义的服务，关于这点我在后面 **自定义UIActivity服务** 的内容中会讲解，现在我们暂时置为 **nil**。代码如下：

```
- (IBAction)presentPDFActivityView:(id)sender {
    UIActivityViewController *activity = [[UIActivityViewController alloc] initWithActivityI
```

视图展示

UIActivityViewController 是直接继承 **UIViewController** 的，看到这，你想象可以通过自己的需求来使用不同的方式展示 **UIActivityViewController** 啦，然而事与愿违。

官方文档中是这么说的: “When presenting the view controller, you must do so using the appropriate means for the current device. On iPad, you must present the view controller in a popover. On iPhone and iPod touch, you must present it modally”。大概意思是说，展示 **UIActivityViewController** 的时候需要根据当前的设备类型选择合适的展示方式，在iPad设备上就必须在'popover'视图里面展示，在其他设备上，必须以模态视图展示。

个人认为开发必须持怀疑和验证的态度，所以我尝试在一个 **UINavigationController** 中push一个 **UIActivityController** ,代码如下：

```
- (IBAction)presentPDFActivityView:(id)sender {

    UIActivityViewController *activity = [[UIActivityViewController alloc] initWithActivityI
    [self.navigationController pushViewController:activity animated:YES];
}
```

然后运行程序，点击Button，意料之中程序崩溃掉了，给出我们的错误反馈是：

```
2015-12-31 15:03:03.733 ZSDocumentInteractionTest[9307:971136] *** Terminating app due to
*** First throw call stack:
(
  0  CoreFoundation                0x0000000103197e65 __exceptionPreprocess + 165
  1  libobjc.A.dylib               0x0000000102c10deb objc_exception_throw + 48
  2  CoreFoundation                0x0000000103197d9d +[NSException raise:format:] + 12
  3  UIKit                         0x0000000103e68e55 -[UIActivityViewController presentPDFActivityView:] + 12
  4  UIKit                         0x00000001036e0949 -[UIViewController _setView controllersToPresent:] + 12
  5  UIKit                         0x00000001036e12cc -[UIViewController _endAppearanceTransition] + 12
  6  UIKit                         0x000000010371bf63 -[UINavigationController navgationController:pushViewController:animated:] + 12
  7  UIKit                         0x0000000103711d24 __49-[UINavigationController pushViewController:animated:] block_invoke + 12
  8  UIKit                         0x0000000103f4ad20 -[_UIViewControllerAnimatedTransitioning _animate] + 12
  9  UIKit                         0x000000010352cfff __53-[_UINavigationController pushViewController:animated:] block_invoke + 12
  10 UIKit                        0x00000001035f1076 -[UIViewAnimationBlockDelegate _start] + 12
  11 UIKit                        0x00000001035ce2af -[UIViewAnimationState sendActionToItems:] + 12
  12 UIKit                        0x00000001035ce65e -[UIViewAnimationState animateWithDuration:delay:options:animations:completion:] + 12
  13 QuartzCore                   0x00000001070c2fa0 _ZN2CA5Layer23run_animation_block_ at 1070c2fa0 + 12
  14 libdispatch.dylib            0x000000010589f49b _dispatch_client_callout + 12
  15 libdispatch.dylib            0x00000001058872af _dispatch_main_queue_callback_4bb + 12
  16 CoreFoundation              0x00000001030f7d09 __CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__ + 12
  17 CoreFoundation              0x00000001030b92c9 __CFRunLoopRun + 2073
  18 CoreFoundation              0x00000001030b8828 CFRunLoopRunSpecific + 488
  19 GraphicsServices             0x0000000106954ad2 GSEventRunModal + 161
  20 UIKit                        0x0000000103544610 UIApplicationMain + 171
  21 ZSDocumentInteractionTest    0x000000010270b6af main + 111
  22 libdyld.dylib                0x00000001058d392d start + 1
)
```

我们看出错误说明是 "UIActivityViewController can only be used modally or as contentViewController in popover on iPad." 因此我们需要更换一下展示方法，在手机上已一个模态视图的方式展示，而在iPad上则作为 **popover** 的内容视图展示。代码如下：

```

- (IBAction)presentPDFActivityView:(id)sender {

    UIActivityViewController *activity = [[UIActivityViewController alloc] initWithActivityI
    activity.excludedActivityTypes = @[UIActivityTypeAirDrop];

    // incorrect usage
    // [self.navigationController pushViewController:activity animated:YES];

    UIPopoverPresentationController *popover = activity.popoverPresentationController;
    if (popover) {
        popover.sourceView = self.activityButton;
        popover.permittedArrowDirections = UIPopoverArrowDirectionUp;
    }

    [self presentViewController:activity animated:YES completion:NULL];
}

```

再次运行代码，点击Button，就可以看到下面的界面啦(完美展示)



然后我们就可以选择服务来操作和分享 [史蒂夫·乔布斯传](#) 啦。

excludedActivityTypes

`UIActivityViewController` 相比于 `UIDocumentInteractionController` 优势除了可以添加额外的自定义服务，它还提供了非常好的原生服务的定制化功能。我们可以完全根据自己的需求，控制 `UIActivityViewController` 提供的系统服务的显示，比如我不想展示 `AirDrop` 这个功能，而这点在 `UIDocumentInteractionController` 是做不到的。想做到这一点，就需要使用到 `UIActivityViewController` 提供的一个属性：

```
@property(nullable, nonatomic, copy) NSArray<NSString *> *excludedActivityTypes; //
default is nil. activity types listed will not be displayed
```

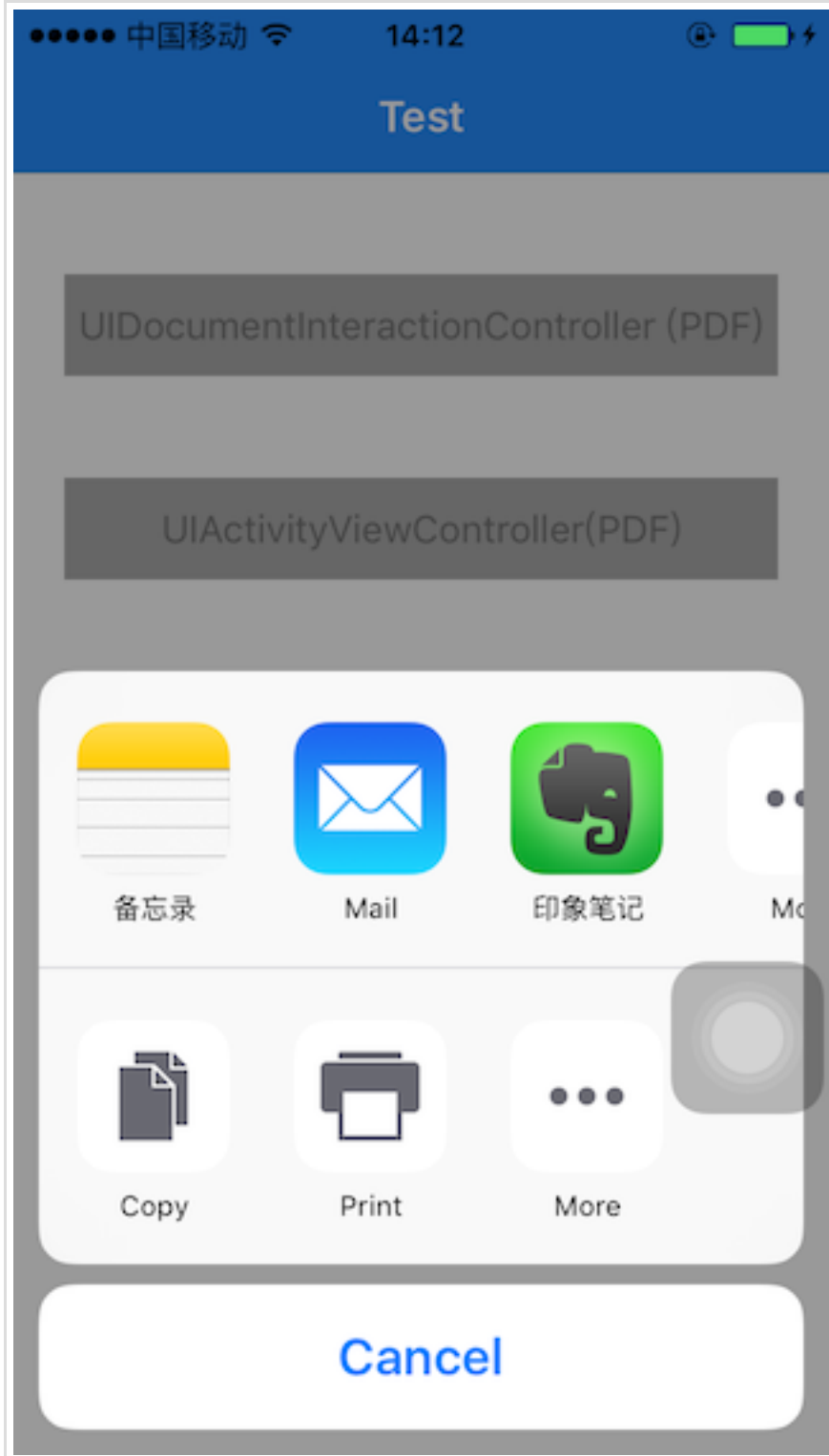
正如注释中提到的，`excludedActivityTypes` 这个属性包含了所有不想在 `UIActivityViewController` 中展示的Item服务。`excludedActivityTypes` 是一个字符串数组，所包含的内容必须是系统提供的 `UIActivity` 的 `activityType` 字符串，而系统提供的字符串如下：

```
NSString *const UIActivityTypePostToFacebook;
NSString *const UIActivityTypePostToTwitter;
NSString *const UIActivityTypePostToWeibo;
NSString *const UIActivityTypeMessage;
NSString *const UIActivityTypeMail;
NSString *const UIActivityTypePrint;
NSString *const UIActivityTypeCopyToPasteboard;
NSString *const UIActivityTypeAssignToContact;
NSString *const UIActivityTypeSaveToCameraRoll;
NSString *const UIActivityTypeAddToReadingList;
NSString *const UIActivityTypePostToFlickr;
NSString *const UIActivityTypePostToVimeo;
NSString *const UIActivityTypePostToTencentWeibo;
NSString *const UIActivityTypeAirDrop;
```

如果我们不想展示 `AirDrop` 功能，我们把 `UIActivityTypeAirDrop` 添加到 `excludedActivityTypes` 里面：

```
activity.excludedActivityTypes = @[UIActivityTypeAirDrop];
```

运行程序，点击Button，我们可以看到下面的界面发生的变化。



自定义UIActivity服务

`UIActivityViewController` 相比于 `UIDocumentInteractionController` 的最大优势就是 `UIActivityViewController` 所提供的自定义服务，我们可以通过 `UIActivity` 在 `UIActivityViewController` 上添加我们自定义的服务。

官方文档上对UIActivity有一段解释，"This class must be subclassed before it can be used. The job of an activity object is to act on the data provided to it and to provide some meta information that iOS can display to the user. For more complex services, an activity object can also display a custom user interface and use it to gather additional information from the user."。其大概意思是，UIActivity必须通过继承来使用，它主要是操作给用户展示的信息，而且还可以操作展示定制化的界面来获取更多地数据信息。

现在我们打算自定义一个叫 `ZS Custom` 的服务，所以我们创建一个 `ZSCustomActivity` 得类来继承 `UIActivity`，除此之外，我们必须重写下面的几个方法：

- `activityType`

```
- (nullable NSString *)activityType; // default returns nil. subclass may override to return custom activity type that is reported to completion handler
```

这是用来标识自定义服务的一个字符串,而系统提供的服务的标识在上面我们已经提到了；为了迎合 iOS SDK 中的规范，我给它返回一个 `UIActivityTypeZSCustomMine`，定义如下：

```
NSString *const UIActivityTypeZSCustomMine = @"ZSCustomActivityMine";

- (NSString *)activityType
{
    return UIActivityTypeZSCustomMine;
}
```

- `activityTitle`

```
- (nullable NSString *)activityTitle; // default returns nil. subclass must override and must return non-nil value
```

在 `UIActivityViewController` 中给用户展示的服务的名称，比如上面图片中的 `"Copy"`，`"Print"`，我们自定义的服务名称为 `ZS Custom`：

```
- (NSString *)activityTitle
{
    //国际化
    return NSLocalizedString(@"ZS Custom", @"");
}
```

- `activityImage`

```
- (nullable UIImage *)activityImage; // default returns nil. subclass must override and must return non-nil value
```

在 `UIActivityViewController` 中给用户展示的服务的图标。关于这里的图标，有非常严格的限制：

- 首先是图标的背景色，这里推荐最好的完全透明的背景色。

官方文档中是这么解释的，"The alpha channel of the image is used as a mask to generate the final image that is presented to the user. Any color data in the image itself is ignored. Opaque pixels have a gradient applied to them and this gradient is then laid on top of a

standard background. Thus, a completely opaque image would yield a gradient filled rectangle",意思大概是，在这里颜色数据会被忽略，而透明图层会被当做mask(蒙版图层)，不透明的图片会显示成渐进色填充。

- 其次是图标的尺寸，在不同的设备需要不同的尺寸，因此需要准备一套图标。

Device	iOS Version	Icon Size(pt)
iPhone、iPod Touch	iOS 6	< 43x43
iPhone、iPod Touch	iOS 7+	60x60
iPad	iOS 6	< 60x60
iPad	iOS 7+	76x76
Retina	All	@2x

- canPerformWithActivityItems:

```
- (BOOL)canPerformWithActivityItems:(NSArray *)activityItems; // override this to return availability of activity based on items. default returns NO
```

指定可以处理的数据类型，如果可以处理则返回`YES`

- prepareWithActivityItems:

```
- (void)prepareWithActivityItems:(NSArray *)activityItems; // override to extract items and set up your HI. default does nothing
```

在用户选择展示在 `UIActivityViewController` 中的自定义服务的图标之后，调用自定义服务处理方法之前的准备工作，都需要在这个方法中指定，比如可以根据数据展示一个界面来获取用户指定的额外数据信息

- activityCategory

```
+ (UIActivityCategory)activityCategory NS_AVAILABLE_IOS(7_0); // default is UIActivityCategoryAction.
```

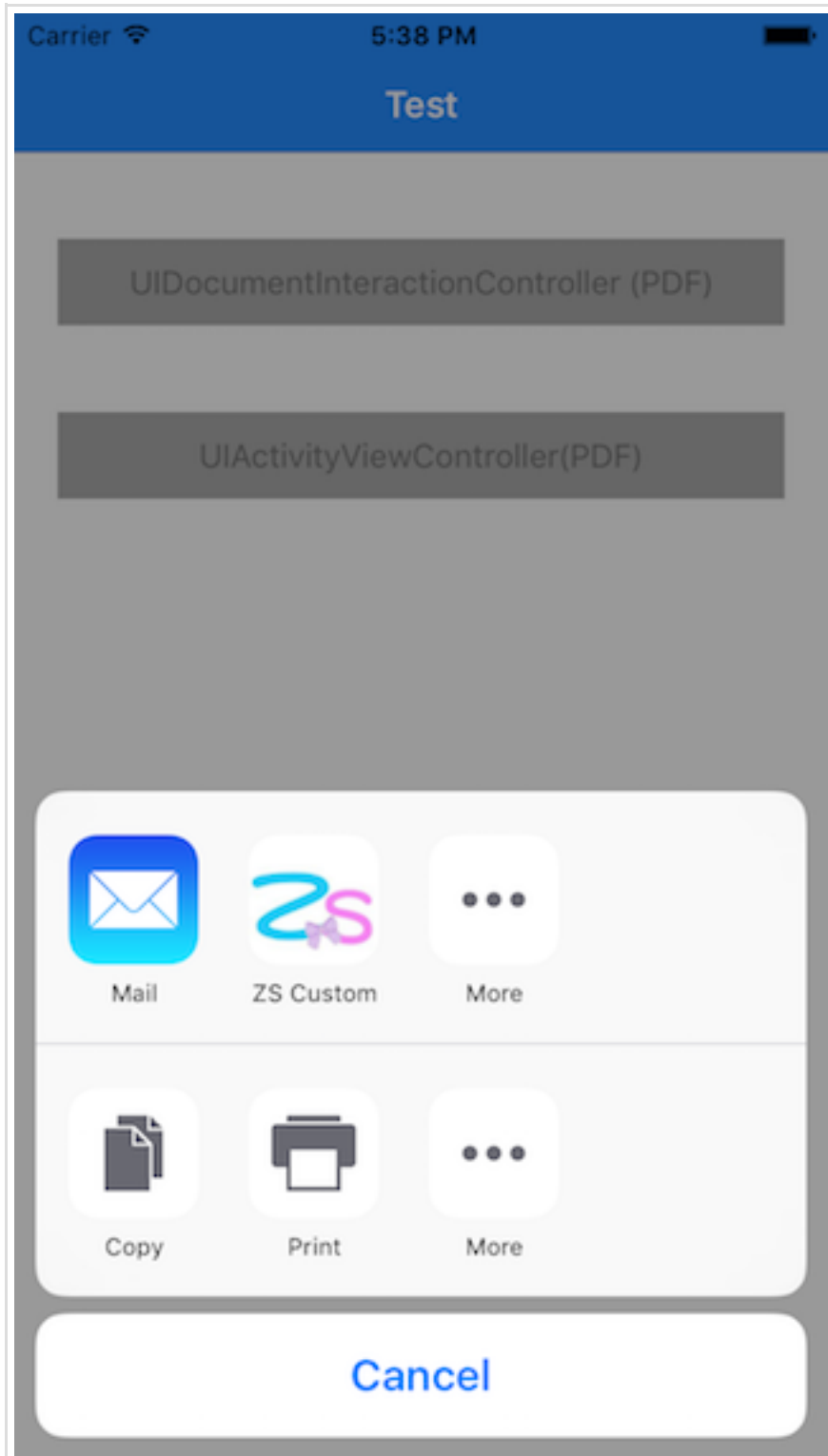
`UIActivityViewController` 中的服务分为了俩种， `UIActivityCategoryAction` 和 `UIActivityCategoryShare`，`UIActivityCategoryAction` 表示在最下面一栏的操作型服务,比如 Copy 、 Print ； `UIActivityCategoryShare``表示在中间一栏的分享型服务， 比如一些社交软件。

- performActivity

```
- (void)performActivity; // if no view controller, this method is called. call  
activityDidFinish when done. default calls [self activityDidFinish:NO]
```

在用户选择展示在`UIActivityViewController`中的自定义服务的图标之后，而且也调用了`prepareWithActivityItems:`，就会调用这个方法执行具体的服务操作

需要的方法都重写好之后，运行程序，点击Button，就可以看到我们自定义的服务图标显示在了`UIActivityViewController`中。



补充之AirDrop

前面一直提到`AirDrop`,我们在这里额外补充一下`AirDrop`的相关知识点。`AirDrop`是在`iOS 7`中提供的，实现跨设备传输文档的功能。`AirDrop`的实现基于蓝牙创建一种类似WIFI的”点对点网络“，然后实现跨设备

传输功能。

只是 **AirDrop** 的传输是有限制的，我们可以在我们的App中通过 **AirDrop** 传送内容，却不能实现通过 **AirDrop** 接收内容，因为，苹果把设备上通过 **AirDrop** 接收到的内容都放到了自家App上，比如仅仅传送文字时，在接收设备上就会通过 **Notes** 打开；如果传送图片，在接收设备上就会保存到 **Photos** 应用中；通过 URL 传送文件，在接收设备上就会通过 **Safari** 打开。

只要有 **UIDocumentInteractionController** 和 **UIActivityViewController** 展示的地方，都可以展示 **AirDrop** 功能。关于 **AirDrop** 如何连接设备，如何传送，可以到百度经验找完美得教程。

1月4日发布

1 推荐

收藏

你可能感兴趣的文章

- [分享] IOS开发－实现在app中拨打电话 2 收藏，333 浏览
- 3 Step：前端工程师使用 ionic 开发iOS App 12 收藏，7.7k 浏览
- 如图 在 iOS 到处 ipa包的时候 会有四个选项的作用 75 浏览


讨论区

请先 [登录](#) 后评论

本文隶属于专栏

iOS随笔

平时工作和生活中的iOS



shaobo

作者

关注专栏

系列文章

- UIViewController解耦---浅析Three20架构 6 收藏，992 浏览

分享扩散：



Copyright © 2011-2016 SegmentFault. 当前呈现版本 16.06.02

浙ICP备 15005796号-2 浙公网安备 33010602002000号

[移动版](#) [桌面版](#)