iOS 5 Programming Cookbook

第十一章
照相机和图片库

版本 1.0

翻译时间：2012-07-23

DevDiv 翻译： kyelup cloudhsu 耐心摩卡
wangli2003j3 xiebaochun dymx101
Jimmylianf BeyondVincent

DevDiv 校对： laigb kyelup

DevDiv 编辑： BeyondVincent

写在前面

目前，移动开发被广大的开发者们看好，并大量的加入移动领域的开发。

鉴于以下原因：

- 国内的相关中文资料缺乏
- 许多开发者对 E 文很是感冒
- 电子版的文档利于技术传播和交流

DevDiv.com 移动开发论坛特此成立了翻译组，翻译组成员具有丰富的移动开发经验和英语翻译水平。组员们利用业余时间，把一些好的相关英文资料翻译成中文，为广大移动开发者尽一点绵薄之力，希望能对读者有些许作用，在此也感谢组员们的辛勤付出。

关于 DevDiv

DevDiv 已成长为国内最具人气的综合性移动开发社区

更多相关信息请访问 DevDiv 移动开发论坛。

技术支持

首先 DevDiv 翻译组对您能够阅读本文以及关注 DevDiv 表示由衷的感谢。

在您学习和开发过程中，或多或少会遇到一些问题。DevDiv 论坛集结了一流的移动专家，我们很乐意与您一起探讨移动开发。如果您有什么问题和技术需要支持的话，请访问 DevDiv 移动开发论坛 或者发送邮件到 BeyondVincent@DevDiv.com，我们将尽力所能及的帮助你。

关于本文的翻译

感谢 kyelup、cloudhsu、耐心摩卡、wangli2003j3、xiebaochun、dymx101、jimmylianf 和 BeyondVincent 对本文的翻译，同时非常感谢 laigb 和 kyelup 在百忙中抽出时间对翻译初稿的认真校验，指出了文章中的错误。才使本文与读者尽快见面。由于书稿内容多，我们的知识有限，尽管我们进行了细心的检查，但是还是会存在错误，这里恳请广大读者批评指正，并发送邮件至 BeyondVincent@devdiv.com，在此我们表示衷心的感谢。

读者查看下面的帖子可以持续关注章节翻译更新情况

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译各章节汇总

另外，我们也为大家准备了 iOS6 新特征的相关内容，请参考下面的帖子：

iOS6 新特征：参考资料和示例汇总------持续更新中

目录

前言

参考帖子

[DevDiv翻译]iOS 5 Programming Cookbook翻译_前言

第 1 章 基础入门

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第一章_基础入门

第 2 章 使用控制器和视图

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第二章_使用控制器和视图(上)

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第二章_使用控制器和视图(下)

第 3 章　构造和使用 Table View

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第三章_构造和使用 Table View

第 4 章 Storyboards

参考帖子

[DevDiv 翻译]iOS 5 Programming Cookbook 翻译_第四章_Storyboards

第 5 章　并发

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第五章_并发

第 6 章 定位核心与地图

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第六章_核心定位与地图

第 7 章 实现手势识别的功能

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第七章_实现手势识别

第 8 章 网络，JSON，XML 以及 Twitter

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 翻译_第八章_网络，JSON，XML 以及 Twitter

第 9 章 音频和视频

参考帖子

[DEVDIV 翻译]iOS 5 Programming Cookbook 中文翻译_第九章_音频和视频

第 10 章　通讯录

参考帖子

[DEVDIV 翻译] iOS 5 Programming Cookbook 中文翻译_第十章_通讯录

第 11 章 照相机和图片库

11.0. 介绍

目前一些 iOS 的设备都是带有摄像头的，例如 iPhone4S 有前后两个摄像头，iPhone3G 和 iPhone3GS 是有一个摄像头，而有些是没有带摄像头的，比如说 iPhone 第一代。

UIImagePickerController 这个类可以为大家提供照相的功能，以及图片，视频浏览的功能。

这个章节中我们首先要了解，如何判断我们的设备是否支持照相的功能，然后我们会接着讲一下如何利用相关的类库添加照相，录像，图片浏览的功能。

iOS 模拟器是没有摄像的功能的，因此如果我们需要测试这一类的程序的时候我们需要连接我们的真本进行真机测试。

由于我们要使用 UIImagePickerController 这个类，因此我们需要为我们的程序添加相对应得框架包。在本章节中我们需要导入 MobileCoreServices.framework 这个框架包。具体添加过程请参考第一章节的部分。

11.1. 摄像头设备存在与否的检测

11.1.1. 问题

你想知道运行你的程序的设备是否支持摄像或者拍照的功能，那么你就需要先进行一些硬件设备的检测，这个过程是必须的，当你的程序中添加了需要使用摄像头的功能。

11.1.2. 方案

通过 UIImagePickerController 这个类的 isSourceTypeAvailable 这个方法进行检测。代码如下

```
- (BOOL) isCameraAvailable{

   return [UIImagePickerController isSourceTypeAvailable:
              UIImagePickerControllerSourceTypeCamera];

}
- (BOOL)                    application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{

   if ([self isCameraAvailable]){
      NSLog(@"Camera is available.");
   } else {
```

```
    NSLog(@"Camera is not available.");
  }

  self.window = [[UIWindow alloc] initWithFrame:
                 [[UIScreen mainScreen] bounds]];


  self.window.backgroundColor = [UIColor whiteColor];
  [self.window makeKeyAndVisible];
  return YES;
}
```

11.1.3. 讨论

当使用 UIImagePickerController 个类来进行类似拍照，录像的功能的时候，你就必须要对摄像头的存在与否进行检测。通过 isSourceTypeAvailable 这个方法，能够对三种数据进行检查。

1. 摄像头的功能，通过给这个方法传递 UIImagePickerControllerSourceTypeCamera 这个值
2. 图片库，通过传递 UIImagePickerControllerSourceTypePhotoLibrary 这个对象。这个时候是打开了照相目录的顶层目录
3. 列表形式的浏览目录，通过传递 UIImagePickerControllerSourceTypeSavedPhotosAlbum 这个值。

如果你想检测如上的任何一个功能是否能够使用，那么你就必须要传递不同的参数。

下面让我们看一下具体的代码改如何实现。

```
#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>
#import <MobileCoreServices/MobileCoreServices.h>
@interface Detecting_and_Probing_the_CameraAppDelegate
                : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@end
```

下面是我们需要在.m 文件中需要添加的检测方法。

```
- (BOOL) cameraSupportsMedia:(NSString *)paramMediaType
                sourceType:(UIImagePickerControllerSourceType)paramSourceType{

  __block BOOL result = NO;

  if ([paramMediaType length] == 0){
    NSLog(@"Media type is empty.");
    return NO;
  }

  NSArray *availableMediaTypes =
  [UIImagePickerController availableMediaTypesForSourceType:paramSourceType];
[availableMediaTypes enumerateObjectsUsingBlock:
   ^(id obj, NSUInteger idx, BOOL *stop) {

     NSString *mediaType = (NSString *)obj;
```

```
        if ([mediaType isEqualToString:paramMediaType]){
            result = YES;
            *stop= YES;
        }

    }];

    return result;

}
- (BOOL) doesCameraSupportShootingVideos{

    return [self cameraSupportsMedia:(__bridge NSString *)kUTTypeMovie
                                sourceType:UIImagePickerControllerSourceTypeCamera];

}
- (BOOL) doesCameraSupportTakingPhotos{

    return [self cameraSupportsMedia:(__bridge NSString *)kUTTypeImage
                                sourceType:UIImagePickerControllerSourceTypeCamera];

}
- (BOOL)                    application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{

    if ([self doesCameraSupportTakingPhotos]){
        NSLog(@"The camera supports taking photos.");
    } else {
        NSLog(@"The camera does not support taking photos");
    }

    if ([self doesCameraSupportShootingVideos]){
        NSLog(@"The camera supports shooting videos.");
    } else {
        NSLog(@"The camera does not support shooting videos.");
    }

    self.window = [[UIWindow alloc] initWithFrame:
                            [[UIScreen mainScreen] bounds]];

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

但是有些是有，有些 iOS 设备是有超过一个摄像头的，为了检测所有的摄像头是否都能正常功能做，那么你就需要使用 isCameraDeviceAvailable 这个方法了，代码如下。

```
- (BOOL) isFrontCameraAvailable{

    return [UIImagePickerController
                isCameraDeviceAvailable:UIImagePickerControllerCameraDeviceFront];

}
- (BOOL) isRearCameraAvailable{

    return [UIImagePickerController
```

```
            isCameraDeviceAvailable:UIImagePickerControllerCameraDeviceRear];

}
```

当在比较老的没有前置摄像头的设备中，你将会看到 isFrontCameraAvailable，和 isRearCameraAvailable 的值分别是 YES 和 NO。当你在 iphone4S 中测试的时候，发觉前后摄像头都是 OK 的。现在我们把一些常用的方法使用罗列如下。

```
- (BOOL)                    application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{

   if ([self isFrontCameraAvailable]){
      NSLog(@"The front camera is available.");
      if ([self isFlashAvailableOnFrontCamera]){
         NSLog(@"The front camera is equipped with a flash");
      } else {
         NSLog(@"The front camera is not equipped with a flash");
      }
   } else {
      NSLog(@"The front camera is not available.");
   }

   if ([self isRearCameraAvailable]){
      NSLog(@"The rear camera is available.");
      if ([self isFlashAvailableOnRearCamera]){
         NSLog(@"The rear camera is equipped with a flash");
      } else {
         NSLog(@"The rear camera is not equipped with a flash");
      }
   } else {
      NSLog(@"The rear camera is not available.");
   }

   if ([self doesCameraSupportTakingPhotos]){
      NSLog(@"The camera supports taking photos.");
   } else {
      NSLog(@"The camera does not support taking photos");
   }

   if ([self doesCameraSupportShootingVideos]){
      NSLog(@"The camera supports shooting videos.");
   } else {
      NSLog(@"The camera does not support shooting videos.");
   }

   self.window = [[UIWindow alloc] initWithFrame:
                  [[UIScreen mainScreen] bounds]];

   self.window.backgroundColor = [UIColor whiteColor];
   [self.window makeKeyAndVisible];
   return YES;
}
```

如上代码执行之后，将会打印如下效果。（iPhone4）

```
The front camera is available.
The front camera is not equipped with a flash
```

```
The rear camera is available.
The rear camera is equipped with a flash
The camera supports taking photos.
The camera supports shooting videos.
```

如上代码执行之后，将会打印如下效果。（3GS）

```
The front camera is not available.
The rear camera is available.
The rear camera is not equipped with a flash
The camera supports taking photos.
The camera supports shooting videos.
```

如上代码执行之后，将会打印如下效果。（第一代 iPad）

```
The front camera is not available.
The rear camera is not available.
The camera does not support taking photos
The camera does not support shooting videos
```

如上代码执行之后，将会打印如下效果。（第二代 iPad）

```
The front camera is available.
The front camera is not equipped with a flash
The rear camera is available.
The rear camera is not equipped with a flash
The camera supports taking photos.
The camera supports shooting videos.
```

11.2. 利用摄像头进行拍照的功能

11.2.1. 问题

你需要为你的应用程序添加拍照的功能。

11.2.2. 方案

利用 UIImagePickerController 这个类，然后把他传递到一个新的 viewcontroller 控制器中。请参考如下头文件需要修改的代码。

```
#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>
#import <MobileCoreServices/MobileCoreServices.h>
@interface Taking_Photos_with_the_CameraViewController
        : UIViewController <UINavigationControllerDelegate,
                            UIImagePickerControllerDelegate>
@end
```

由于 UIImagePickerController 这个的协议类必须要遵循 UINavigationControllerDelegate 和 UIImagePickerControllerDelegate 这两个协议类。如果你忘记在你的程序中添加这个协议对象。那么你将会在编译的时候有一堆的警告，提醒你需要给你的 image picker 协议属性进行赋值。下面我们将看一下具体在 body 文件中式怎么来操作的。

```
- (void)viewDidLoad{
   [super viewDidLoad];

   if ([self isCameraAvailable] &&
         [self doesCameraSupportTakingPhotos]){

      UIImagePickerController *controller =
        [[UIImagePickerController alloc] init];

      controller.sourceType = UIImagePickerControllerSourceTypeCamera;

      NSString *requiredMediaType = (__bridge NSString *)kUTTypeImage;
      controller.mediaTypes = [[NSArray alloc]
                                    initWithObjects:requiredMediaType, nil];

      controller.allowsEditing = YES;
      controller.delegate = self;

      [self.navigationController presentModalViewController:controller
                                                    animated:YES];

   } else {
      NSLog(@"Camera is not available.");
   }

}
```

我们在如上的代码中使用了 isCameraAvailable 和 doesCameraSupportTaking 这两个属性，如果不明白的情参考本章节的第一小节。

通过上面的这些代码，我们可以让用户在他的应用程序中添加是否能使用照相机的功能。

其中有一点，你是需要注意的，我们给 image picker 这个属性设置了相对性的值。因此为了能够完成的使用照相的功能我们有还需要实现相关协议类的方法。代码如下。

```
- (void)    imagePickerController:(UIImagePickerController *)picker
   didFinishPickingMediaWithInfo:(NSDictionary *)info{

   NSLog(@"Picker returned successfully.");

   NSString        *mediaType = [info objectForKey:
                                    UIImagePickerControllerMediaType];

   if ([mediaType isEqualToString:(__bridge NSString *)kUTTypeMovie]){

      NSURL *urlOfVideo =
      [info objectForKey:UIImagePickerControllerMediaURL];

      NSLog(@"Video URL = %@", urlOfVideo);
```

```
    }

    else if ([mediaType isEqualToString:(__bridge NSString *)kUTTypeImage]){

        /* Let's get the metadata. This is only for
          images. Not videos */

        NSDictionary *metadata =
        [info objectForKey:
          UIImagePickerControllerMediaMetadata];

        UIImage *theImage =
        [info objectForKey:
          UIImagePickerControllerOriginalImage];

        NSLog(@"Image Metadata = %@", metadata);
        NSLog(@"Image = %@", theImage);


    }

    [picker dismissModalViewControllerAnimated:YES];

}
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker{

    NSLog(@"Picker was cancelled");
    [picker dismissModalViewControllerAnimated:YES];

}
```

11.2.3. 讨论

如上的代码中我们需要注意几个问题，首先，我们要完全的实现我们继承协议的方法。imagePickerController:didFinishPickingMediaWithInfo,这个方法是当我们完成拍照之后会返回调用的方法。imagePickerControllerDidCancel，这个方法是当我们进入拍照程序之后，点击取消之后进入的方法。

当然，当我们实现 didFinishPickingMediaWithInfo 这个方法之后，如果我们有一些拍照，或者录像，是有相关的数据进行返回的。他可能是一段录像，或者一张拍照图片。为了能够充分的获取这些数据，我们需要了解如下两个参数类型的值。首先，我们需要知道 UIImagePickerControllerMediaType 的值是什么，然后根据类型来进行获取相关的数据。

kUTTypeImage

这个是一个图片类型

kUTTypeMovie

视频类型

The kUTTypeImage 和 kUTTypeMovie 是 MobileCore Services 框架包中的内容，由于使用了桥连模式，你可以直接使用

当时做了硬件检测，和资源类型的辨别，那么你就可以在

didFinishPickingMediaWithInfo 这个方法中获取相关的数据。你可以通过如下几种类型获取数据。

UIImagePickerControllerMediaMetadata

这些数据时以 NSDictionary 形式保存起来，这个字典中包含了很多有用的信息，不过本小节不做过多介绍。

UIImagePickerControllerOriginalImage

这个数据其实是一个 UIImage 类型的数据，也就是用户所拍摄的那张图片数据。

UIImagePickerControllerCropRect

当打开编辑模式的时候，这个对象，将会包含裁剪剩余的图像部分。

UIImagePickerControllerEditedImage

当打开编辑模式的时候，这个对象，将会包含裁剪的图像部分。

当时 Video 类型的时候，你可以通过 UIImagePickerControllerMediaURL 这个值来进行数据的获取。这个就是用户所拍摄的录像的路径信息。

当你的到了用户拍照的图像引用的时候，你就可以继续你的功能，可以是展示啊，提交啊等等。

 请注意一下，当用户在这种情况下拍摄的图像的时候只是暂时的存储，并没有保存在你的手机中。

11.2.4. 参考

无

11.3. 利用摄像头进行视频的录制。

11.3.1. 问题

你想在你的应用中添加视频录制的功能。

11.3.2. 方案

通过使用 kUTTypeMovie 这个类型来进行视频的录制。

参考代码如下。

```
- (void)viewDidLoad{
   [super viewDidLoad];

   if ([self isCameraAvailable] &&
         [self doesCameraSupportTakingPhotos]){

      UIImagePickerController *controller =
         [[UIImagePickerController alloc] init];
```

```
    controller.sourceType = UIImagePickerControllerSourceTypeCamera;

    NSString *requiredMediaType = (__bridge NSString *)kUTTypeMovie;
    controller.mediaTypes = [[NSArray alloc]
                             initWithObjects:requiredMediaType, nil];

    controller.allowsEditing = YES;
    controller.delegate = self;

    [self.navigationController presentModalViewController:controller
                                                 animated:YES];

  } else {
    NSLog(@"Camera is not available.");
  }

}
```

同理我们添加如上代码之后需要添加想关的协议方法。代码如下

```
- (void)   imagePickerController:(UIImagePickerController *)picker
  didFinishPickingMediaWithInfo:(NSDictionary *)info{

  NSLog(@"Picker returned successfully.");

  NSLog(@"%@", info);

  NSString        *mediaType = [info objectForKey:
                                UIImagePickerControllerMediaType];

  if ([mediaType isEqualToString:(__bridge NSString *)kUTTypeMovie]){

    NSURL *urlOfVideo =
    [info objectForKey:UIImagePickerControllerMediaURL];

    NSLog(@"Video URL = %@", urlOfVideo);

    NSError *dataReadingError = nil;

    NSData *videoData =
    [NSData dataWithContentsOfURL:urlOfVideo
                          options:NSDataReadingMapped
                            error:&dataReadingError];

    if (videoData != nil){
      /* We were able to read the data */
      NSLog(@"Successfully loaded the data.");
    } else {
      /* We failed to read the data. Use the dataReadingError
        variable to determine what the error is */
      NSLog(@"Failed to load the data with error = %@",
            dataReadingError);
    }

  }

  [picker dismissModalViewControllerAnimated:YES];
}
```

```
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker

   NSLog(@"Picker was cancelled");
   [picker dismissModalViewControllerAnimated:YES];

}
```

11.3.3. 讨论

如上代码，首先我们还是需要先检测设备是否能够支持拍照的功能，然后我们在设置成录像的模式，最后我们可以在回调方法 didFinishPickingMediaWithInfo 中获取到我们录制的视频。

当用户在进行视频录制的时候，这些视频会保存在一些临时的目录当中，并不是在多媒体的目录当中，一个通常的路径形式如下。

file://localhost/private/var/mobile/Applications /<APPID>/tmp/capture-T0x104e20.tmp.TQ9UTr/capturedvideo.MOV

作为一个开发人员，我们为用户提供的并不是简单的录像的功能，我们还需要提供视频格式的问题。那么我们就需要使用如下两个参数进行设置。

videoQuality

这个属性，我们有两个设置的值，UIImagePickerControllerQualityTypeHigh（高清）UIImagePickerControllerQualityTypeMedium（普清）

videoMaximumDuration

这个属性让我们用来设置视频录制的最小单元，这个属性是用秒来计算的。

例如我们需要录制高清的视频，每个文件的单元为 30 秒，那么我们就需要修改代码，代码如下。

```
- (void)viewDidLoad{
   [super viewDidLoad];

   if ([self isCameraAvailable] &&
         [self doesCameraSupportTakingPhotos]){

      UIImagePickerController *controller =
         [[UIImagePickerController alloc] init];
      controller.sourceType = UIImagePickerControllerSourceTypeCamera;

      NSString *requiredMediaType = (__bridge NSString *)kUTTypeMovie;
      controller.mediaTypes = [[NSArray alloc]
                                    initWithObjects:requiredMediaType, nil];

      controller.allowsEditing = YES;
      controller.delegate = self;

      /* Record in high quality */
      controller.videoQuality = UIImagePickerControllerQualityTypeHigh;

      /* Only allow 30 seconds of recording */
      controller.videoMaximumDuration = 30.0f;

      [self.navigationController presentModalViewController:controller
                                                   animated:YES];
```

```
    } else {
        NSLog(@"Camera is not available.");
    }

}
```

11.3.4. 参考

暂无

11.4. 把图片存储在多媒体库中

11.4.1. 问题

你需要来把一些图片存储在用户的图片库中

11.4.2. 方案

通过使用 UIImageWriteToSavedPhotosAlbum 这个方法来实现。

比如说，你现在拍摄了一张照片，你需要保存在你的手机中，而不是临时的使用，那么你就需要添加你的代码类似如下。

```
- (void) imageWasSavedSuccessfully:(UIImage *)paramImage
                 didFinishSavingWithError:(NSError *)paramError
                                contextInfo:(void *)paramContextInfo{

    if (paramError == nil){
        NSLog(@"Image was saved successfully.");
    } else {
        NSLog(@"An error happened while saving the image.");
        NSLog(@"Error = %@", paramError);
    }

}
(void)    imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info{

    NSLog(@"Picker returned successfully.");

    NSLog(@"%@", info);

    NSString        *mediaType = [info objectForKey:
                                  UIImagePickerControllerMediaType];

    if ([mediaType isEqualToString:(__bridge NSString *)kUTTypeImage]){
        UIImage *theImage = nil;

        if ([picker allowsEditing]){
            theImage = [info objectForKey:UIImagePickerControllerEditedImage];
        } else {
            theImage = [info objectForKey:UIImagePickerControllerOriginalImage];
```

```
        }

        SEL selectorToCall =
        @selector(imageWasSavedSuccessfully:didFinishSavingWithError:contextInfo:);

        UIImageWriteToSavedPhotosAlbum(theImage,
                                       self,
                                       selectorToCall,
                                       NULL);

    }

    [picker dismissModalViewControllerAnimated:YES];

}
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker{

    NSLog(@"Picker was cancelled");
    [picker dismissModalViewControllerAnimated:YES];

}
- (void)viewDidLoad{
    [super viewDidLoad];

    if ([self isCameraAvailable] &&
            [self doesCameraSupportTakingPhotos]){

        UIImagePickerController *controller =
        [[UIImagePickerController alloc] init];

        controller.sourceType = UIImagePickerControllerSourceTypeCamera;

        NSString *requiredMediaType = (__bridge NSString *)kUTTypeImage;
        controller.mediaTypes = [[NSArray alloc]
                                 initWithObjects:requiredMediaType, nil];
            controller.allowsEditing = YES;
        controller.delegate = self;

        [self.navigationController presentModalViewController:controller
                                                     animated:YES];

    } else {
        NSLog(@"Camera is not available.");
    }

}
```

11.4.3. 讨论

由于用户利用照相机来进行拍照，系统并不知道这张照片是否需要保存在手机中，那么这个保存的与否完全就是根据业务的规则来定的。因此我们在需要保存的时候才进行添加相关的代码来进行保存。其中我们使用 UIImageWriteToSavedPhotosAlbum 这个方法来进行保存，那么这个参数需要传递如下几个值。

1. 需要保存的图像

2. 一个通知对象，这个对象包含图片是否完整保存的信息。
3. 这个参数是根据第二个参数的完整情况来的，当第二个参数返回调用这个参数设置的方法。
4. 同 3.

如上第 2,3,4 个参数是可选的，当你提供第二个，第三个参数的时候，那么第四个参数也是可选的。如下代码我们提供了其中的一个方法案例。

```
- (void) imageWasSavedSuccessfully:(UIImage *)paramImage
              didFinishSavingWithError:(NSError *)paramError
                           contextInfo:(void *)paramContextInfo{

   if (paramError == nil){
      NSLog(@"Image was saved successfully.");
   } else {
      NSLog(@"An error happened while saving the image.");
      NSLog(@"Error = %@", paramError);
   }

}
```

如果这个 error 参数，你收到的是 nil 的话，这就意味着你的图片已经正常保存完毕。否则，就说明有错误。

11.5. 把视频录像文件保存在多媒体库中

11.5.1. 问题

你需要通过一个录制好的临时视频的链接，把这个视频保存在你的多媒体库中。

11.5.2. 方案

通过使用 writeVideoAtPathToSavedPhotosAlbum:completionBlock:这个方法来实现。具体参考如下代码。

```
- (BOOL)                    application:(UIApplication *)application
   didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{

   self.assetsLibrary = [[ALAssetsLibrary alloc] init];

   NSURL *videoURL = [[NSBundle mainBundle] URLForResource:@"MyVideo"
                                                withExtension:@"MOV"];

   if (videoURL != nil){
      [self.assetsLibrary
        writeVideoAtPathToSavedPhotosAlbum:videoURL
        completionBlock:^(NSURL *assetURL, NSError *error) {
```

```
            if (error == nil){
                NSLog(@"no errors happened");
            } else {
                NSLog(@"Error happened while saving the video.");
                NSLog(@"The error is = %@", error);
            }

        }];
    } else {
        NSLog(@"Could not find the video in the app bundle.");
    }

    self.window = [[UIWindow alloc] initWithFrame:
                   [[UIScreen mainScreen] bounds]];

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

在如上的示例中，我们通过创建一个 ALAssetsLibrar 的对象来进行视频的保存动作。

11.5.3. 讨论

Assets Library 框架包是一个很方便的桥梁，在用户与多媒体库打交道的时候。在 iOS 类库中，有可疑创建自己的视图组建界面来进行多媒体数据的交互，但是，这样的话你就需要花费很多时间来解决问题。所以 Asserts Library 的出现，帮你解决了很多为题。

当我们创建了一个 ALAssetsLibrary 类型的对象之后，我们就可以使用 writeVideoAtPathToSavedPhotosAlbum:completionBlock:这个方法来进行视频文件的保存动作。这个方法中你需要传递几个参数，你需要提供一个 NSURL 类型的视频连接地址，其次，你需要添加一个 NSError 类型的错误对象。

如果这个 error 对象是空的话，那么你的保存动作就成功完成，通过，我们可能碰到一个比较常见的问题就可能如下。

Error Domain=ALAssetsLibraryErrorDomain Code=-3302 "Invalid data"
UserInfo=0x7923590 {NSLocalizedFailureReason=
There was a problem writing this asset because
the data is invalid and cannot be viewed or played.,
NSLocalizedRecoverySuggestion=Try with different data,
NSLocalizedDescription=Invalid data}

如果你是在你的模拟器上运行这些代码的话，你肯能会碰到如下问题。

Error Domain=ALAssetsLibraryErrorDomain Code=-3310 "Data unavailable"
UserInfo=0x6456810 {NSLocalizedRecoverySuggestion=
Launch the Photos application, NSLocalizedDescription=Data unavailable}

如果碰到这个问题，请进入到你的程序主界面，然后再打开 Photos 程序，然后再重新运行你的程序。

writeVideoAtPathToSavedPhotosAlbum:completionBlock:这个方法中的第一个参数将会指定这个视频保存的路径，可能格式如下。

assets-library://asset/asset.MOV?id=1000000002&ext=MOV

在本章的 11.7 的时候我们将会学习如何通过这个 URL 来加载这个视频文件。

11.6. 从多媒体库中得到相关的图片和录像程序

11.6.1. 问题

你想在你的程序中添加图片或者视频浏览的功能。

11.6.2. 方案

通过使用 UIImagePickerController 这个类的来实现相关的功能。代码如下。

```
- (BOOL) isPhotoLibraryAvailable{

    return [UIImagePickerController isSourceTypeAvailable:
                UIImagePickerControllerSourceTypePhotoLibrary];

}
- (BOOL) canUserPickVideosFromPhotoLibrary{

    return [self
                cameraSupportsMedia:(__bridge NSString *)kUTTypeMovie
                sourceType:UIImagePickerControllerSourceTypePhotoLibrary];

}
- (BOOL) canUserPickPhotosFromPhotoLibrary{

    return [self
                cameraSupportsMedia:(__bridge NSString *)kUTTypeImage
                sourceType:UIImagePickerControllerSourceTypePhotoLibrary];

}
- (void)viewDidLoad{
    [super viewDidLoad];

    if ([self isPhotoLibraryAvailable]){

        UIImagePickerController *controller =
            [[UIImagePickerController alloc] init];

        controller.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

NSMutableArray *mediaTypes = [[NSMutableArray alloc] init];
      if ([self canUserPickPhotosFromPhotoLibrary]){
            [mediaTypes addObject:(__bridge NSString *)kUTTypeImage];
        }

        if ([self canUserPickVideosFromPhotoLibrary]){
            [mediaTypes addObject:(__bridge NSString *)kUTTypeMovie];
        }

        controller.mediaTypes = mediaTypes;

        controller.delegate = self;
```

```
    [self.navigationController presentModalViewController:controller
                                                  animated:YES];

  }

}
```

11.6.3. 讨论

为了能够让用户选择图片或者是视频，你必须要通过给 sourceType 来设置相关的类型才能区分不同类型的文件。

11.7. 不利用任何 UI 组件来获取手机中的多媒体信息

11.7.1. 问题

你想在你的程序中获取多媒体信息，不是通过任何的 UI 组件来获取的。

11.7.2. 方案

利用 Assets Library framework 这个框架包来帮你实现，不过你需要完成如下几步
下面我们通过代码看一下具体的实现方式。

```
- (void)viewDidLoad{
  [super viewDidLoad];
  self.assetsLibrary = [[ALAssetsLibrary alloc] init];

  [self.assetsLibrary
    enumerateGroupsWithTypes:ALAssetsGroupAll
    usingBlock:^(ALAssetsGroup *group, BOOL *stop) {
      [group enumerateAssetsUsingBlock:^(ALAsset *result,
                                         NSUInteger index,
                                         BOOL *stop) {

        /* Get the asset type */
        NSString *assetType = [result valueForProperty:ALAssetPropertyType];

        if ([assetType isEqualToString:ALAssetTypePhoto]){
          NSLog(@"This is a photo asset");
        }

        else if ([assetType isEqualToString:ALAssetTypeVideo]){
          NSLog(@"This is a video asset");
        }

        else if ([assetType isEqualToString:ALAssetTypeUnknown]){
          NSLog(@"This is an unknown asset");
        }

        /* Get the URLs for the asset */
        NSDictionary *assetURLs = [result valueForProperty:ALAssetPropertyURLs]
```

```
        NSUInteger       assetCounter = 0;
        for (NSString *assetURLKey in assetURLs){
           assetCounter++;
           NSLog(@"Asset URL %lu = %@",
                    (unsigned long)assetCounter,
                    [assetURLs valueForKey:assetURLKey]);
        }

        /* Get the asset's representation object */
        ALAssetRepresentation *assetRepresentation =
         [result defaultRepresentation];

        NSLog(@"Representation Size = %lld", [assetRepresentation size]);

      }];
    }
    failureBlock:^(NSError *error) {
       NSLog(@"Failed to enumerate the asset groups.");
    }];

}
- (void)viewDidUnload{
   [super viewDidUnload];
   self.assetsLibrary = nil;
}
```

11.7.3. 讨论

下面，我们开始写一个小程序，这个小程序从设备中读取一张图片然后创建一个 UIImageView 用来展示这个图片。这样我们就可以了解，如何获取图片，如何来展现他们。

我们首先需要创建一个 ViewController，然后定义一个 imageView 在这个 congroller 中。

代码如下。

头文件大致代码如下。

```
#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>
#import <MobileCoreServices/MobileCoreServices.h>
@interface Retrieving_Assets_from_the_Assets_LibraryViewController
                : UIViewController <UIImagePickerControllerDelegate,
                                    UINavigationControllerDelegate>
@property (nonatomic, strong) ALAssetsLibrary *assetsLibrary;
@property (nonatomic, strong) UIImageView *imageView;
@end
```

包文件代码大致修改如下。

```
#import "Retrieving_Assets_from_the_Assets_LibraryViewController.h"
@implementation Retrieving_Assets_from_the_Assets_LibraryViewController
@synthesize assetsLibrary;
@synthesize imageView;
```

现在我们需要做的就是当这个 viewcontroller 加载的时候，我们需要初始化一个 AssetSLibrary 的对象。然后开始解析数据，获取数据，用来展示。

大致我们需要在 viewDidLoad 方法中添加如下代码。

```
- (void)viewDidLoad{
   [super viewDidLoad];
self.view.backgroundColor = [UIColor whiteColor];
self.assetsLibrary = [[ALAssetsLibrary alloc] init];
dispatch_queue_t dispatchQueue =
   dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
dispatch_async(dispatchQueue, ^(void) {

   [self.assetsLibrary
    enumerateGroupsWithTypes:ALAssetsGroupAll
    usingBlock:^(ALAssetsGroup *group, BOOL *stop) {

       [group enumerateAssetsUsingBlock:^(ALAsset *result,
                                          NSUInteger index,
                                          BOOL *stop) {

          __block BOOL foundThePhoto = NO;

          if (foundThePhoto){
             *stop = YES;
          }

          /* Get the asset type */
          NSString *assetType = [result valueForProperty:ALAssetPropertyType];

          if ([assetType isEqualToString:ALAssetTypePhoto]){
             NSLog(@"This is a photo asset");

             foundThePhoto = YES;
             *stop = YES;

             /* Get the asset's representation object */
             ALAssetRepresentation *assetRepresentation =
              [result defaultRepresentation];

             /* We need the scale and orientation to be able to construct a
              properly oriented and scaled UIImage out of the
              representation object */
             CGFloat imageScale = [assetRepresentation scale];

             UIImageOrientation imageOrientation =
              (UIImageOrientation)[assetRepresentation orientation];

             dispatch_async(dispatch_get_main_queue(), ^(void) {

                CGImageRef imageReference =
                 [assetRepresentation fullResolutionImage];

                /* Construct the image now */
                UIImage        *image =
                [[UIImage alloc] initWithCGImage:imageReference
                                           scale:imageScaleorientation:imageOrientation];
                   if (image != nil){
                      self.imageView = [[UIImageView alloc]
                                        initWithFrame:self.view.bounds];
                      self.imageView.contentMode = UIViewContentModeScaleAspectFit;
                      self.imageView.image = image;
```

```
                    [self.view addSubview:self.imageView];

                } else {
                    NSLog(@"Failed to create the image.");
                }
            });

        }

    }];
}
failureBlock:^(NSError *error) {
    NSLog(@"Failed to enumerate the asset groups.");
}];

});

}
- (void)viewDidUnload{
    [super viewDidUnload];
    self.assetsLibrary = nil;
    self.imageView = nil;
}
```

获取图片并用来展示的代码就如上，是不是很简单，下面我们在看一下如何解析一个视频文件并来展示。这个我们需要把 Temp.MOV 放在我们的 Documents 的文件夹中。

```
- (void)viewDidLoad{
    [super viewDidLoad];

    self.view.backgroundColor = [UIColor whiteColor];
    self.assetsLibrary = [[ALAssetsLibrary alloc] init];

    dispatch_queue_t dispatchQueue =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

    dispatch_async(dispatchQueue, ^(void) {
        [self.assetsLibrary
        enumerateGroupsWithTypes:ALAssetsGroupAll
        usingBlock:^(ALAssetsGroup *group, BOOL *stop) {

            __block BOOL foundTheVideo = NO;

            [group enumerateAssetsUsingBlock:^(ALAsset *result,
                                               NSUInteger index,
                                               BOOL *stop) {

                /* Get the asset type */
                NSString *assetType = [result valueForProperty:ALAssetPropertyType];

                if ([assetType isEqualToString:ALAssetTypeVideo]){
                    NSLog(@"This is a video asset");

                    foundTheVideo = YES;
                    *stop = YES;

                    /* Get the asset's representation object */
                    ALAssetRepresentation *assetRepresentation =
                     [result defaultRepresentation];
```

```
            const NSUInteger BufferSize = 1024;
            uint8_t buffer[BufferSize];
            NSUInteger bytesRead = 0;
            long long currentOffset = 0;
            NSError *readingError = nil;

            /* Find the documents folder (an array) */
            NSArray *documents =
            NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                NSUserDomainMask,
                                                YES);

            /* Retrieve the one documents folder that we need */
            NSString *documentsFolder = [documents objectAtIndex:0];

            /* Construct the path where the video has to be saved */
            NSString *videoPath = [documentsFolder
                                   stringByAppendingPathComponent:@"Temp.MOV"];

            NSFileManager *fileManager = [[NSFileManager alloc] init];

            /* Create the file if it doesn't exist already */
            if ([fileManager fileExistsAtPath:videoPath] == NO){
               [fileManager createFileAtPath:videoPath
                                     contents:nil
                                   attributes:nil];


  /* We will use this file handle to write the contents
              of the media assets to the disk */
            NSFileHandle *fileHandle = [NSFileHandle
                                        fileHandleForWritingAtPath:videoPath];
do{

               /* Read as many bytes as we can put in the buffer */
               bytesRead = [assetRepresentation getBytes:(uint8_t *)&buffer
                                              fromOffset:currentOffset
                                                  length:BufferSize
                                                   error:&readingError];

               /* If we couldn't read anything, we will exit this loop */
               if (bytesRead == 0){
                  break;
               }

               /* Keep the offset up to date */
               currentOffset += bytesRead;

               /* Put the buffer into an NSData */
               NSData *readData = [[NSData alloc]
                                   initWithBytes:(const void *)buffer
                                   length:bytesRead];

               /* And write the data to file */
               [fileHandle writeData:readData];

            } while (bytesRead > 0);
```

```
                NSLog(@"Finished reading and storing the \
                        video in the documents folder");

            }

        }];

        if (foundTheVideo){
            *stop = YES;
        }

    }
    failureBlock:^(NSError *error) {
        NSLog(@"Failed to enumerate the asset groups.");
    }];

});

}
- (void)viewDidUnload{
    [super viewDidUnload];
    self.assetsLibrary = nil;
}
```

11.8. 实现带有一般播放功能的应用程序

11.8.1. 问题

你希望在你的应用程序中添加可以直接编辑视频的功能，快进，后退等等

11.8.2. 方案

通过使用 UIVideoEditorController 这个类来帮你解决这个问题，在如下的代码中，我们将会利用这个类和 imagePickercontroleller 来共同解决这个问题，我们会先让用户从多媒体库中选择一个视频文件，然后我们把这个文件加载在 UIVideoEditorController 中让用户来编辑。

11.8.3. 讨论

iOS SDK 的 UIVideoEditorController 这个类可以为用户提供展示视频的相关 UI 接口。你所需要做的就是只需要提供视频的 URL 连接地址。你不应该在这个 View 上面添加其他的什么视图，并且你不能修改这个试图。

presentModalViewController：animated：这方法和 dismissModalViewControllerAnimated 这个方法两关使用将会报错，你必须要等到第一个视图消失完全之后并且准备跳转到第二个试图的时候才能调用第二个方法。你可以充分使用 viewDidAppear 这个方法来进行判断你的视图什么时候展示，当你的这个试图调用这个方法的时候就表示这个 controller 将会消失。

下面让我们来看看具体代码的实现。
头文件的代码大致如下。

```
#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>
#import <MobileCoreServices/MobileCoreServices.h>
@interface Editing_Videos_on_an_iOS_DeviceViewController
                : UIViewController <UINavigationControllerDelegate,
                                    UIVideoEditorControllerDelegate,
                                    UIImagePickerControllerDelegate>
@property (nonatomic, strong) NSURL *videoURLToEdit;
@end
```

下面我们首先需要在包文件中进行如下代码。

```
#import "Editing_Videos_on_an_iOS_DeviceViewController.h"
@implementation Editing_Videos_on_an_iOS_DeviceViewController
@synthesize videoURLToEdit;
```

下面我们需要做的就是捕获在编辑视频时候的不同事件。

```
- (void)videoEditorController:(UIVideoEditorController *)editor
        didSaveEditedVideoToPath:(NSString *)editedVideoPath{
    NSLog(@"The video editor finished saving video");
    NSLog(@"The edited video path is at = %@", editedVideoPath);
    [editor dismissModalViewControllerAnimated:YES];
}
- (void)videoEditorController:(UIVideoEditorController *)editor
                didFailWithError:(NSError *)error{
    NSLog(@"Video editor error occurred = %@", error);
    [editor dismissModalViewControllerAnimated:YES];
}
- (void)videoEditorControllerDidCancel:(UIVideoEditorController *)editor{
    NSLog(@"The video editor was cancelled");
    [editor dismissModalViewControllerAnimated:YES];
}
```

当我们的试图加载的时候，我们需要把这个视频信息展现给用户，这样，用户就能够可以通过自己的选择然后来进行视频的编辑工作。

```
- (BOOL) cameraSupportsMedia:(NSString *)paramMediaType
                sourceType:(UIImagePickerControllerSourceType)paramSourceType{

    __block BOOL result = NO;

    if ([paramMediaType length] == 0){
        NSLog(@"Media type is empty.");
        return NO;
    }

    NSArray *availableMediaTypes =
    [UIImagePickerController availableMediaTypesForSourceType:paramSourceType];
```

```
    [availableMediaTypes enumerateObjectsUsingBlock:
     ^(id obj, NSUInteger idx, BOOL *stop) {

         NSString *mediaType = (NSString *)obj;
         if ([mediaType isEqualToString:paramMediaType]){
            result = YES;
            *stop= YES;
         }

     }];

    return result;

}
- (BOOL) canUserPickVideosFromPhotoLibrary{

    return [self cameraSupportsMedia:(__bridge NSString *)kUTTypeMovie
                          sourceType:UIImagePickerControllerSourceTypePhotoLibrary];

}
- (BOOL) isPhotoLibraryAvailable{

    return [UIImagePickerController
                isSourceTypeAvailable:
                UIImagePickerControllerSourceTypePhotoLibrary];

}
- (void)viewDidLoad {
    [super viewDidLoad];

    if ([self isPhotoLibraryAvailable] &&
          [self canUserPickVideosFromPhotoLibrary]){

        UIImagePickerController *imagePicker =
            [[UIImagePickerController alloc] init];
        /* Set the source type to photo library */
        imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;

        /* And we want our user to be able to pick movies from the library */
        NSArray    *mediaTypes = [[NSArray alloc] initWithObjects:
                                  (__bridge NSString *)kUTTypeMovie, nil];

        imagePicker.mediaTypes = mediaTypes;

        /* Set the delegate to the current view controller */
        imagePicker.delegate = self;

        /* Present our image picker */
        [self.navigationController presentModalViewController:imagePicker
                                                      animated:YES];

    }

}
```

另外我们需要添加用户处理动作的事件监听处理方法。
大致如下。

```
- (void)       imagePickerController:(UIImagePickerController *)picker
      didFinishPickingMediaWithInfo:(NSDictionary *)info{

   NSLog(@"Picker returned successfully.");

   NSString         *mediaType = [info objectForKey:
                                  UIImagePickerControllerMediaType];

   if ([mediaType isEqualToString:(NSString *)kUTTypeMovie]){
      self.videoURLToEdit = [info objectForKey:UIImagePickerControllerMediaURL];
   }

   [picker dismissModalViewControllerAnimated:YES];

}
- (void) imagePickerControllerDidCancel:(UIImagePickerController *)picker{

   NSLog(@"Picker was cancelled");
   self.videoURLToEdit = nil;
   [picker dismissModalViewControllerAnimated:YES];

}
```

当用户选择好视频之后，我们就需要来把视频展现出来。需要添加如下代码。

```
- (void) viewDidAppear:(BOOL)animated{
   [super viewDidAppear:animated];

   if (self.videoURLToEdit != nil){
            NSString *videoPath = [self.videoURLToEdit path];

      /* First let's make sure the video editor is able to edit the
        video at the path in our documents folder */
      if ([UIVideoEditorController canEditVideoAtPath:videoPath]){

         /* Instantiate the video editor */
         UIVideoEditorController *videoEditor =
         [[UIVideoEditorController alloc] init];

         /* We become the delegate of the video editor */
         videoEditor.delegate = self;

         /* Make sure to set the path of the video */
         videoEditor.videoPath = videoPath;

         /* And present the video editor */
         [self.navigationController presentModalViewController:videoEditor
                                                      animated:YES];

         self.videoURLToEdit = nil;

      } else {
         NSLog(@"Cannot edit the video at this path");
      }

   }
```

```
}
```

在我们上段代码中我们允许用户选择任何的视频文件。当他选择完成之后，就会进行一个视频的加载播放程序。然后你可以根据这个 controller 中的各种属性来编辑这个视频的信息，进度条啊，播放事件等信息。