iOS 5 Programming Cookbook

第九章
音频和视频

版本 1.0

翻译时间：2012-07-09

DevDiv 翻译： kyelup cloudhsu 耐心摩卡 wangli2003j3 xiebaochun dymx101 jimmylianf

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 和 jimmylianf 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

9.0 简介

iOS SDK 中的 AV 框架可以帮助开发者轻松的播放/录制音频和视频。除此之外 Media Player 框架还允许开发者播放音频和视频文件。 在你运行本章的代码之前，必须将 AVFoundation.frame-work 和 MediaPlayer.framework 框架添加到 Xcode 项目中。可以通过以下几步添加：

1. 在 Xcode 中，点击项目上的图标。
2. 选择要把框架添加到哪个 target 。
3. 选择界面上方的 Build Phases。
4. 选择 Link Binaries 左下方的 + 按钮。
5. 按住 Command 键并且在列表中选择 AVFoundation.framework 和 Media-Player.framework。
6. 选择添加。

9.1. 播放音频文件

9.1.1. 问题

你想在你的应用中播放音频文件。

9.1.2. 方案

使用 AV 框架（Audio 和 Video 框架）里的 AVAudioPlayer 类。

9.1.3. 讨论

AV 框架（Audio 和 Video 框架）里的 AVAudioPlayer 类播放 iOS 支持的所有音频格式。AVAudioPlayer 实例的 delegate 属性允许我们通过事件获得通知，例如当音频播放被打断或者播放音频文件出错时。我们来看个例子，演示如何播放我们程序束中的音频文件：

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

    self.view.backgroundColor = [UIColor whiteColor];

    dispatch_queue_t dispatchQueue =
        dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
```

```
    dispatch_async(dispatchQueue, ^(void) {
        NSBundle *mainBundle = [NSBundle mainBundle];

        NSString *filePath = [mainBundle pathForResource:@"MySong"
                                                  ofType:@"mp3"];

        NSData     *fileData = [NSData dataWithContentsOfFile:filePath];

        NSError    *error = nil;

        /* 开启音频控制器 */
        self.audioPlayer = [[AVAudioPlayer alloc] initWithData:fileData
                                                         error:&error];

        /* 得到的 AVAudioPlayer 实例是否非空*/
        if (self.audioPlayer != nil){
            /* 设置 delegate 开始播放*/
            self.audioPlayer.delegate = self;
            if ([self.audioPlayer prepareToPlay] &&
                [self.audioPlayer play]){
                /* 播放成功 */
            } else {
                /* 播放失败 */
            }
        } else {
            /* 初始化 AVAudioPlayer 失败 */
        }
    });

}
```

可以看到文件中的数据首先被加载到一个 NSData 实例，然后被传递到 AVAudioPlayer 类的 initWithData:error:方法。因为我们需要 MP3 文件的绝对路径来获取文件数据，我们调用 NSBundle 类的 mainBundle 方法来从程序的配置中获取信息。然后像代码中写的那样，用 NSBundle 类的 pathForResource:ofType:方法来得到指定类型资源的绝对路径。

在 viewDidLoad 方法中，我们用 GCD 来异步从歌曲数据中加载数据到 NSData 实例中，而且把数据提供给音频播放器。这么做是因为加载不同长度的音频文件中的数据需要很长时间，如果我们在主线程中做的话会有影响 UI 体验的风险。

因此，我们利用一个全局的并发队列来确保代码不在主线程中运行。因为我们将 AVAudioPlayer 实例赋值给了 audioPlayer,属性，我们需要指定该属性如何定义：

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
@interface Playing_Audio_FilesViewController
                : UIViewController <AVAudioPlayerDelegate>
@property (nonatomic, strong) AVAudioPlayer *audioPlayer;
@end
```

可以看到，我们把试图控制器作为了音频播放器的 delegate 。这样就可以每当播放歌曲被打断或结束时从系统中得到消息。掌握了这些信息后，我们就可以在程序中做出正确操作，例如播放另一个音频文件时。

9.1.4. 参考

9.2 节; 9.5 节

9.2. 处理播放音频时的中断

9.2.1. 问题

你想让你的 AVAudioPlayer 实例在被打断后恢复播放，例如来电时。

9.2.2. 方案

在你的 AVAudioPlayer 实例的 delegate 对象中实现 AVAudioPlayerDelegate 协议的 audioPlayerBeginInterruption: 和 audioPlayerEndInterruption:withFlags:方法：

```
- (void)audioPlayerBeginInterruption:(AVAudioPlayer *)player{

   /* Audio Session is interrupted. The player will be paused here */

}
- (void)audioPlayerEndInterruption:(AVAudioPlayer *)player
                         withFlags:(NSUInteger)flags{

   if (flags == AVAudioSessionInterruptionFlags_ShouldResume &&
         player != nil){
      [player play];
   }

}
```

9.2.3. 讨论

在 iOS 设备例如 iPhone 上，来电会打断一个正在运行的程序。与正在运行的程序相关的音频 session 就会被终止，音频文件就会停止播放直到中断结束。在中断的开始和结束，我们都会收到来自 AVAudioPlayer 的代理消息通知我们音频 session 当前的不同状态。

中断结束后，我们可以方便的恢复音频的播放。iPhone 模拟器不能模拟来电。必须在真机上测试程序。

当中断发生时，AVAudioPlayer 实例的 audioPlayerBeginInterruption: delegate 方法会被调用。这时，你的音频 session 已经被打断了。

如果是来电，用户只能听到铃声。当通话结束或用户拒绝来电后，AVAudioPlayer 类的 udioPlayerEndInterruption:withFlags: delegate 方法会被调用。如果参数 withFlags 包含 AVAudioSessionInterruptionFlags_ShouldResume 标识，你就可以用 AVAudioPlayer 的播放实

例来立即恢复播放音频。

用 AVAudioPlayer 恢复播放音频文件在 iPhone 模拟器运行时 Instruments 会出现内存泄漏。在 iOS 设备上测试能证明内存泄漏致出现在模拟器上。我强烈建议你在你的应用放到 App Store 之前在真机上运行、调试、优化一下。

9.3. 录制音频

9.3.1. 问题

你想在 iOS 设备上录制音频。

9.3.2. 方案

确保你已经将 CoreAudio.framework 库添加到目标文件中。使用 AV 库中的 AVAudioRecorder 类:

```
NSError     *error = nil;
NSString *pathAsString = [self audioRecordingPath];
NSURL *audioRecordingURL = [NSURL fileURLWithPath:pathAsString];
self.audioRecorder = [[AVAudioRecorder alloc]
                      initWithURL:audioRecordingURL
                      settings:[self audioRecordingSettings]
                      error:&error];
```

更多关于例子中 audioRecordingSettings 和 audioRecordingPath 方法的信息请参考讨论章节。

9.3.3. 讨论

AV 框架中的 AVAudioRecorder 类使得在 iOS 中录制音频变得很简单。开始录制音频需要提供一些参数给 AVAudioRecorder 实例的 initWithURL:settings:error:方法。文件的 URL 应该被保存。文件的 URL 是一个本地 URL.AV 框架会根据 URL 的扩展名来决定录制文件的音频格式。索引要仔细选择扩展名。

在采样之前和过程中使用的 settings 包括采样率、频道以及其他音频录制器开始录音的信息。Setting 是一个 dictionary 对象。

初始化错误发生时保存到 error 变量中的 NSError 实例的地址应该是很有用的。你可以在出现异常的情况下得到这个实例中的值。

initWithURL:settings:error: 方法的 setting 参数很有意思。很多值都可以保存在这个 setting 字典里，但是在本节中我们只讨论一些最重要的：

AVFormatIDKey

录音的格式。可能的值有：

- kAudioFormatLinearPCM

- kAudioFormatAppleLossless

AVSampleRateKey
录制音频的采样率。
AVNumberOfChannelsKey
录制音频的频道编号。
AVEncoderAudioQualityKey
录制音频的质量，可能的值有：

- AVAudioQualityMin
- AVAudioQualityLow
- AVAudioQualityMedium
- AVAudioQualityHigh
- AVAudioQualityMax

掌握了所有这些信息后我们可以开始写一个可以录制音频文件然后用 AVAudioPlayer 播放的程序。我们要做的具体事情是：

1. 用 Apple Lossless 格式录制音频。
2. 把录制的音频文件用 Recording.m4a 文件名保存到程序的 Documents 字典目录中。
3. 在录音开始 5 秒后停止录制并且立刻开始播放录制的音频。

我们将从在简单的试图控制器的头文件中声明一下需要的属性开始：

```
#import <UIKit/UIKit.h>
#import <CoreAudio/CoreAudioTypes.h>
#import <AVFoundation/AVFoundation.h>
@interface Recording_AudioViewController : UIViewController
                <AVAudioPlayerDelegate, AVAudioRecorderDelegate>
@property (nonatomic, strong) AVAudioRecorder *audioRecorder;
@property (nonatomic, strong) AVAudioPlayer *audioPlayer;
- (NSString *)          audioRecordingPath;
- (NSDictionary *)    audioRecordingSettings;
@end
```

当我们的视图控制器中的视图第一次加载时，我们尝试开始录音然后如果成功的话在 5 秒后停止。

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

   NSError     *error = nil;

   NSString *pathAsString = [self audioRecordingPath];

   NSURL *audioRecordingURL = [NSURL fileURLWithPath:pathAsString];

   self.audioRecorder = [[AVAudioRecorder alloc]
                                 initWithURL:audioRecordingURL
                                 settings:[self audioRecordingSettings]
                                 error:&error];

   if (self.audioRecorder != nil){

      self.audioRecorder.delegate = self;
      /*准备开始录制音频*/
```

```
    if ([self.audioRecorder prepareToRecord] &&
        [self.audioRecorder record]){
      NSLog(@"Successfully started to record.");

      /* 5 秒后，我们终止录制过程 */
      [self performSelector:@selector(stopRecordingOnAudioRecorder:)
                 withObject:self.audioRecorder
                 afterDelay:5.0f];

    } else {
      NSLog(@"Failed to record.");
      self.audioRecorder = nil;
    }

  } else {
    NSLog(@"Failed to create an instance of the audio recorder.");
  }

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

  if ([self.audioRecorder isRecording]){
    [self.audioRecorder stop];
  }
  self.audioRecorder = nil;


  if ([self.audioPlayer isPlaying]){
    [self.audioPlayer stop];
  }
  self.audioPlayer = nil;

}
```

在我们的视图控制器的 viewDidLoad 方法中，我们尝试实例化一个 AVAudioRecorder 类型的对象。而且我们把这个对象赋值给了我们在同一个视图控制器的.h 文件中声明的一个属性。

我们用一个 audioRecordingPath 方法来决定我们存储录制的文件的 URL 的 NSString 类型的路径。这个方法这样实现：

```
- (NSString *) audioRecordingPath{

  NSString *result = nil;

  NSArray *folders =
  NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                      NSUserDomainMask,
                                      YES);

  NSString *documentsFolder = [folders objectAtIndex:0];

  result = [documentsFolder
            stringByAppendingPathComponent:@"Recording.m4a"];

  return result;
```

```
}
```

这个方法的返回值是程序的文档目录后面加上目标文件名。例如你程序的文档路径是：/var/mobile/Applications/ApplicationID/Documents/

目标音频文件的目录就是：
/var/mobile/Applications/ApplicationID/Documents/Recording.m4a

像前面介绍的那样，初始化 AVAudioRecorder 时，我们使用了一个 dictionary 作为音频录制器的初始化方法中的 setting 参数。这个 dictionary 用 audioRecordingSettings 方法创建。实现如下：

```
- (NSDictionary *) audioRecordingSettings{

   NSDictionary *result = nil;

   /* 我们在 dictionary 中初始化录制音频的选项。稍后我们会用这个 dictionary 来初始化一个 AVAudioRecorder 类型的音频录制器*/

   NSMutableDictionary *settings = [[NSMutableDictionary alloc] init];

   [settings
    setValue:[NSNumber numberWithInteger:kAudioFormatAppleLossless]
    forKey:AVFormatIDKey];

   [settings
    setValue:[NSNumber numberWithFloat:44100.0f]
    forKey:AVSampleRateKey];

   [settings
    setValue:[NSNumber numberWithInteger:1]
    forKey:AVNumberOfChannelsKey];

   [settings
    setValue:[NSNumber numberWithInteger:AVAudioQualityLow]
    forKey:AVEncoderAudioQualityKey];

   result = [NSDictionary dictionaryWithDictionary:settings];

   return result;

}
```

可以看到在 view controller 的 viewDidLoad 方法中成功录音 5 秒后，我们调用了 stopRecordingOnAudioRecorder 方法，实现如下：

```
- (void) stopRecordingOnAudioRecorder
            :(AVAudioRecorder *)paramRecorder{

   /* 在这儿停止录制 */
   [paramRecorder stop];

}
```

既然我们已经停止了录制，我们要等待 delegate 消息通知我们录制确实停止了。最好不

要认为是 AVAudioRecorder 的停止方法迅速终止了录制。相反的，我建议你在 audioRecorderDidFinishRecording:successfully: delegate 方法(在 AVAudioRecorderDelegate 协议中声明的) 后再继续操作。当录制停止后，我们就可以播放录制的音频：

```
- (void)audioRecorderDidFinishRecording:(AVAudioRecorder *)recorder
                                successfully:(BOOL)flag{

   if (flag){

      NSLog(@"Successfully stopped the audio recording process.");

      /* 我们尝试从一个录制的文件中得到数据 */
      NSError *playbackError = nil;

      NSError *readingError = nil;
      NSData    *fileData =
      [NSData dataWithContentsOfFile:[self audioRecordingPath]
                                options:NSDataReadingMapped
                                  error:&readingError];

      /* 创建一个音频播放器并播放录制的数据*/
      self.audioPlayer = [[AVAudioPlayer alloc] initWithData:fileData
                                                       error:&playbackError];

      /* 能初始化播放器吗? */
      if (self.audioPlayer != nil){
         self.audioPlayer.delegate = self;

         /* 准备并开始播放 */
         if ([self.audioPlayer prepareToPlay] &&
               [self.audioPlayer play]){
            NSLog(@"Started playing the recorded audio.");
         } else {
            NSLog(@"Could not play the audio.");
         }

      } else {
         NSLog(@"Failed to create an audio player.");
      }

   } else {
      NSLog(@"Stopping the audio recording failed.");
   }

   /* 这儿我们就不在需要音频录制器 */
   self.audioRecorder = nil;

}
```

在音频播放器播放完（如果成功的话）一首歌曲后，在音频播放器的 delegate 对象中 audioPlayerDidFinishPlaying:successfully: delegate 方法会被调用。该方法（在 AVAudioPlayerDelegate 协议中声明）实现如下：

```
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player
                            successfully:(BOOL)flag{
```

```
    if (flag){
        NSLog(@"Audio player stopped correctly.");
    } else {
        NSLog(@"Audio player did not stop correctly.");
    }

    if ([player isEqual:self.audioPlayer]){
        self.audioPlayer = nil;
    } else {
        /* 这不是我们的播放器*/
    }

}
```

在 9.2 节中解释过，我们用 AVAudioPlayer 播放音频文件，在将应用发布到 iOS 设备和 App Store 之前，我们也需要出来例如来电这样的终端。

```
- (void)audioPlayerBeginInterruption:(AVAudioPlayer *)player{

    /* 音频 session 在这儿被停止*/

}
- (void)audioPlayerEndInterruption:(AVAudioPlayer *)player
                         withFlags:(NSUInteger)flags{

    if (flags == AVAudioSessionInterruptionFlags_ShouldResume){
        [player play];
    }

}
```

AVAudioRecorder 实例和 AVAudioPlayer 实例一样需要处理中断。可以向 9.4 节那样处理终端。

9.3.4. 参考

9.2 节; 9.4 节

9.4. 处理录制音频过程中的中断

9.4.1. 问题

你想在 AVAudioRecorder 实例被例如来电打断后能够恢复录制。

9.4.2. 方案

在音频录制器的 delegate 方法中实现 AVAudioRecorderDelegate 协议中的 udioRecorderBeginInterruption: and audioRecorderEndInterruption:withFlags:方法。并且在中断结束后调用 AVAudioRecorder 方法来恢复录制过程。如下：

```
- (void)audioRecorderBeginInterruption:(AVAudioRecorder *)recorder{

   NSLog(@"Recording process is interrupted");

}
- (void)audioRecorderEndInterruption:(AVAudioRecorder *)recorder
                             withFlags:(NSUInteger)flags{

   if (flags == AVAudioSessionInterruptionFlags_ShouldResume){
      NSLog(@"Resuming the recording...");
      [recorder record];
   }

}
```

9.4.3. 讨论

就像 AVAudioPlayer 实例的音频播放器一样，AVAudioRecorder 类型的录制器也能在音频 session 因为终端停止后收到代理消息。在本节的方案里提到的两个方法是处理这种终端的最佳方法。

当录制音频被打断时，你可以在中断结束后调用 AVAudioRecorder 的录音方法来继续录制进程。然而新的录音会覆盖之前的而且所有在打断前录制的数据都会丢失。

请记住：当你在录制音频过程中收到 audioRecorderBeginInterruption 方法时，音频 session 已经被停止了，这时调用恢复方法是没用的。当中断结束后，必须调用 AVAudioRecorder 的录音方法来恢复录音。

9.4.4. 参考

9.5. 在其他活动声音上面播放音频

9.5.1. 问题

你可能想在播放音频时让其他程序静音，或者在其他程序的音频之上播放音频。

9.5.2. 解决

使用音频会话设置你的应用程序所使用的音频类别。

9.5.3. 讨论

AVAudioSession 类由 AVFoundation 框架引入。每个 iOS 应用都有一个音频会话。这个

会话可以被 AVAudioSession 类的 sharedInstance 类方法访问，如下：

```
AVAudioSession *audioSession = [AVAudioSession sharedInstance];
```

在获得一个 AVAudioSession 类的实例后，你就能通过调用音频会话对象的 setCategory:error:实例方法，来从 iOS 应用可用的不同类别中作出选择。下面列出了可供使用的音频会话类别：

AVAudioSessionCategorySoloAmbient
这个非常像 AVAudioSessionCategoryAmbient 类别，除了会停止其他程序的音频回放，比如 iPod 程序。当设备被设置为静音模式，你的音频回放将会停止。

AVAudioSessionCategoryRecord
这会停止其他应用的声音（比如 iPod）并让你的应用也不能初始化音频回放（比如 AVAudioPlayer）。在这种模式下，你只能进行录音。使用这个类别，调用 AVAudioPlayer 的 prepareToPlay 会返回 YES，但是调用 play 方法将返回 NO。主 UI 界面会照常工作。这时，即使你的设备屏幕被用户锁定了，应用的录音仍会继续。

AVAudioSessionCategoryPlayback
这个类别会禁止其他应用的音频回放（比如 iPod 应用的音频回放）。你可以使用 AVAudioPlayer 的 prepareToPlay 和 play 方法，在你的应用中播放声音。主 UI 界面会照常工作。这时，即使屏幕被锁定或者设备为静音模式，音频回放都会继续。

AVAudioSessionCategoryPlayAndRecord
这个类别允许你的应用中同时进行声音的播放和录制。当你的声音录制或播放开始后，其他应用的声音播放将会停止。主 UI 界面会照常工作。这时，即使屏幕被锁定或者设备为静音模式，音频回放和录制都会继续。

AVAudioSessionCategoryAudioProcessing
这个类别用于应用中进行音频处理的情形，而不是音频回放或录制。设置了这种模式，你在应用中就不能播放和录制任何声音。调用 AVAPlayer 的 prepareToPlay 和 play 方法都将返回 NO。其他应用的音频回放，比如 iPod，也会在此模式下停止。

AVAudioSessionCategoryAmbient
这个类别不会停止其他应用的声音，相反，它允许你的音频播放于其他应用的声音之上，比如 iPod。你的应用的主 UI 县城会工作正常。调用 AVAPlayer 的 prepareToPlay 和 play 方法都将返回 YES。当用户锁屏时，你的应用将停止所有正在回放的音频。仅当你的应用是唯一播放该音频文件的应用时，静音模式将停止你程序的音频回放。如果正当 iPod 播放一首歌时，你开始播放音频，将设备设为静音模式并不能停止你的音频回放。

为了给你一个例子说明 AVAudioSession 的用法，让我们开始写一个音频播放器，它能在其他应用的音频回放之上播放自己的音频文件。我们会从一个视图控制器的.h 文件开始：

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
@interface Playing_Audio_over_Other_Active_SoundsViewController : UIViewController <AVAudioPlayerDelegate>
@property (nonatomic, strong) AVAudioPlayer *audioPlayer;
@end
```

下面的代码演示了如何改变我们的音频会话，然后如何将音乐加载到内存，并载入音频播放器来播放。我们会在视图控制器的 viewDidLoad 方法中做这件事。

```
- (void)viewDidLoad {
[super viewDidLoad];
NSError *audioSessionError = nil;
AVAudioSession *audioSession = [AVAudioSession sharedInstance];
if ([audioSession setCategory:AVAudioSessionCategoryAmbient
error:&audioSessionError]){
NSLog(@"Successfully set the audio session.");
} else {
NSLog(@"Could not set the audio session");
}
dispatch_queue_t dispatchQueue =
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
dispatch_async(dispatchQueue, ^(void) {
NSBundle *mainBundle = [NSBundle mainBundle];
NSString *filePath = [mainBundle pathForResource:@"MySong"
ofType:@"mp3"];
NSData *fileData = [NSData dataWithContentsOfFile:filePath];
NSError *audioPlayerError = nil;
self.audioPlayer = [[AVAudioPlayer alloc] initWithData:fileData
error:&audioPlayerError];
if (self.audioPlayer != nil){
self.audioPlayer.delegate = self;
if ([self.audioPlayer prepareToPlay] &&
[self.audioPlayer play]){
NSLog(@"Successfully started playing.");
} else {
NSLog(@"Failed to play the audio file.");
self.audioPlayer = nil;
}
} else {
NSLog(@"Could not instantiate the audio player.");
}
});
}
- (void) viewDidUnload{
[super viewDidUnload];
if ([self.audioPlayer isPlaying]){
[self.audioPlayer stop];
}
self.audioPlayer = nil;
}
```

接下来，我们继续处理 AVAudioPlayerDelegate 协议方法：

```
- (void)audioPlayerBeginInterruption:(AVAudioPlayer *)player{
/* The audio session has been deactivated here */
}
- (void)audioPlayerEndInterruption:(AVAudioPlayer *)player
withFlags:(NSUInteger)flags{
if (flags == AVAudioSessionInterruptionFlags_ShouldResume){
[player play];
}
}
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player
```

```
successfully:(BOOL)flag{
if (flag){
NSLog(@"Audio player stopped correctly.");
} else {
NSLog(@"Audio player did not stop correctly.");
}
if ([player isEqual:self.audioPlayer]){
self.audioPlayer = nil;
} else {
/* This is not our audio player */
}
}
```

你可以看到，我们在视图控制器的 ViewDidLoad 实例方法中使用了 AVAudioSession 的共享实例，将我们的应用的音频分类设置为 AVAudioSessionCategoryAmbient，这样就允许了我们的应用可以在其他应用的音频回放上面播放声音。

9.5.4. 参考

9.6. 播放视频文件

9.6.1. 问题

你大概想在你的 IOS 应用中播放视频文件

9.6.2. 解决

使用 MPMoviePlayerController 的实例。

如果你只是想显示一个全屏的视频播放器，你可以使用 MPMoviePlayerViewController 类，（举例来说）你可以将你的影片播放视图控制器推入到一个导航控制器的视图控制器栈上，或者简单的使用 UIViewController 的 presentMoviePlayerViewControllerAnimated:实例方法，在别的控制器上展示你的影片播放视图控制器。但在这里，我们不使用 MPMoviePlayerViewController 类，而是使用 MPMoviePlayerController 类，目的是为了访问控制器没有提供的各种设置，比如说窗口模式的视频回放（不是全屏幕）。

9.6.3. 讨论

iOS SDK 中的媒体播放器框架允许程序员们播放声音和视频文件。为了能够播放一个视频文件，我们会实例化一个 MPMoviePlayerController 类对象，如下：

self.moviePlayer = [[MPMoviePlayerController alloc] initWithContentURL:url];

在以上代码中，mMoviePlayer 是 MPMoviePlayerController 类型的属性，它在当前视图控制器中定义和同步。在早期的 iOS SDK 中，程序员在使用 Media Player 框架时，对于视频如何播放只有极少的控制。随着 iPad 设备的出现，这个框架有了很大的改变，赋予程序

员更多的控制权，允许他们使用比以前更灵活的方式展示内容。

MPMoviePlayerController 实例有一个称为 view 的属性。这个视图是一个 UIView 类型，而且他是诸如视频之类的媒体播放的那个视图。作为程序员，你有责任将这个视图插入到你的应用中的视图结构中，为你的用户展示播放的内容。因为你得到了一个 UIView 对象的引用，你可以按照自己的想法修改这个视图。例如，你可以将视图的背景色修改为自定义颜色。

很多的多媒体操作依赖通知系统。例如，MPMoviePlayerController 并不和委托一起工作，作为替代，它依靠通知。这允许在系统库和 iOS 开发者写的程序代码间，实现非常灵活的解耦。对于像 MPMoviePlayerController 的类，我们只要监听那个类所发出的通知就好了。我们使用默认的通知中心，并将我们自己加为某个通知的观察者。

为了进行测试，我们需要一个样例.mov 文件，使用影片播放器对其进行播放。你可以从 http://support.apple.com/kb/HT1425 下载 APPLE 提供的 sample 文件。确保你下载的是 H.264 文件格式。如果文件是压缩过的，解压缩并将其重命名为 Sample.m4v。将这个文件拖放到 XCODE 里面你程序的目录下。

完成后，我们就可以写一个尝试播放这个视频文件的简单例子了。这里是我们的.h 文件：

```
#import <UIKit/UIKit.h>
#import <MediaPlayer/MediaPlayer.h>
@interface Playing_Video_FilesViewController : UIViewController
@property (nonatomic, strong) MPMoviePlayerController *moviePlayer;
@property (nonatomic, strong) UIButton *playButton;
@end
```

下面是头文件中定义的 startPlayingVideo:方法的实现：

```
- (void) startPlayingVideo:(id)paramSender{
/* 首先，让我们构造影片播放器所需的程序目录下的文件的 URL*/
NSBundle *mainBundle = [NSBundle mainBundle];
NSString *urlAsString = [mainBundle pathForResource:@"Sample"
ofType:@"m4v"];
NSURL *url = [NSURL fileURLWithPath:urlAsString];
/* If we have already created a movie player before,
let's try to stop it */
if (self.moviePlayer != nil){
[self stopPlayingVideo:nil];
}
/*现在使用这个 URL 创建一个新的 movie player*/
self.moviePlayer = [[MPMoviePlayerController alloc] initWithContentURL:url];
if (self.moviePlayer != nil){
/* 不论何时结束播放文件，播放器都会发送通知，让我们监听这个通知 */

[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(videoHasFinishedPlaying:)
name:MPMoviePlayerPlaybackDidFinishNotification
object:self.moviePlayer];
NSLog(@"Successfully instantiated the movie player.");
```

```
/* 缩放影片播放器以适应宽高 */
self.moviePlayer.scalingMode = MPMovieScalingModeAspectFit;
/* 让我们开始以全屏模式播放视频*/
[self.moviePlayer play];
[self.view addSubview:self.moviePlayer.view];
[self.moviePlayer setFullscreen:YES
animated:YES];
} else {
NSLog(@"Failed to instantiate the movie player.");
}
}
```

如你所见，我们自己管理播放器的视图。如果我们将影片播放器的视图添加到视图控制器的视图上面，之后我们需要手动的移除它。即使我们释放了影片播放器，它的视图也不会被视图控制器释放。下面的方法停止播放视频并将其相关的视图移除。

```
- (void) stopPlayingVideo:(id)paramSender {
if (self.moviePlayer != nil){
[[NSNotificationCenter defaultCenter]
removeObserver:self
name:MPMoviePlayerPlaybackDidFinishNotification
object:self.moviePlayer];
[self.moviePlayer stop];
if ([self.moviePlayer.view.superview isEqual:self.view]){
[self.moviePlayer.view removeFromSuperview];
}
}
}
```

在使用你的影片播放器视图的 superView 属性时要格外小心。在因为内存警告使你的视图被卸载的情况下，视图控制器在将其 view 属性设置为 nil 之后，你就不要访问它了。否则，iOS SDK 将在控制台窗口抛出警告。

这里是我们的 viewDidUnload 方法实现：

```
- (void) viewDidUnload{
self.playButton = nil;
[self stopPlayingVideo:nil];
self.moviePlayer = nil;
[super viewDidUnload];
}
```

在视图控制器的 startPlayingVideo: 实例方法中，我们注册了 MKMoviePlayerViewController 发送到默认通知中心的 MPMoviePlayerPlaybackDidFinishNotification 通知。我们在视图控制器的 videoHasFinishedPlaying:实例方法中监听这个通知。在这里，我们可以得知影片已经完成播放，然后可能处理影片播放对象：

```
- (void) videoHasFinishedPlaying:(NSNotification *)paramNotification{
/* 找出播放器停止的原因 */
NSNumber *reason =
[paramNotification.userInfo
valueForKey:MPMoviePlayerPlaybackDidFinishReasonUserInfoKey];
```

```
if (reason != nil){
NSInteger reasonAsInteger = [reason integerValue];
switch (reasonAsInteger){
case MPMovieFinishReasonPlaybackEnded:{
/* 影片正常结束 */
break;
}
case MPMovieFinishReasonPlaybackError:{
/* 发生了一个错误导致影片结束 */
break;
}
case MPMovieFinishReasonUserExited:{
/* 用户退出了播放器*/
break;
}
}
NSLog(@"Finish Reason = %ld", (long)reasonAsInteger);
[self stopPlayingVideo:nil];
} /* if (reason != nil){ */
}
```

你可能已经注意到，我们调用了 stopPlayingVideo: 实例方法，它在 videoHasFinishedPlaying:通知处理者中实现。这样做是因为，videoHasFinishedPlaying:实例方法处理诸如：从影片播放器收到的通知上反注册我们的对象，以及将媒体播放器从其父视图上移除这类事情。换句话说，当视频停止播放时，并不意味着该播放器分配的资源也被释放了。我们需要小心的手动处理。记住 MPMoviePlayerController 类在 IPHONE 模拟器上不能工作。你必须在真机上跑这些代码，并由自己检查结果。

9.6.4. 参考

9.7. 从视频文件中捕获缩略图

9.7.1. 问题

你在使用 MPMoviePlayerController 类的实例播放一个视频文件时，可能想要在某个时刻从影片捕获一张截图。

9.7.2. 解决

使用 MPMoviePlayerController 的 requestThumbnailImagesAtTimes:timeOption: 实例方法，如下：

```
/*在影片的第三秒捕获帧图片*/
NSNumber *thirdSecondThumbnail = [NSNumber numberWithFloat:3.0f];
/* 我们可以要求捕获任意多的帧，但现在，我们只请求捕获一帧*/
NSArray *requestedThumbnails = [NSArray arrayWithObject:thirdSecondThumbnail];
/*请求影片播放器为我们捕获这一帧*/
[self.moviePlayer
requestThumbnailImagesAtTimes:requestedThumbnails timeOption:MPMovieTimeOptionExact];
```

9.7.3. 讨论

MPMoviePlayerController 的一个实例能够从最近播放的影片中捕获缩略图，同步和异步都可以。在这里，我们将关注这个类的异步图像捕获。

我们可以使用 MPMoviePlayerController 的 requestThumbnailImagesAtTimes:timeOption:实例方法来异步访问缩略图。当我说“异步”时，我的意思是，在缩略图正在被捕获和被报告给你指派的对象（我们很快能看到）的过程中，视频播放器将继续它的工作，且不会阻塞回放。我们必须观察 MPMoviePlayerThumbnailImage RequestDidFinishNotification 通知消息，这个消息由视频播放器发送给默认消息中心，以便知道我们的缩略图什么时候可用：

```
- (void) startPlayingVideo:(id)paramSender{
/* First let's construct the URL of the file in our application bundle
that needs to get played by the movie player */ NSBundle *mainBundle = [NSBundle mainBundle];
NSString *urlAsString = [mainBundle pathForResource:@"Sample"
ofType:@"m4v"];
NSURL *url = [NSURL fileURLWithPath:urlAsString];
/* If we have already created a movie player before, let's try to stop it */
if (self.moviePlayer != nil){
[self stopPlayingVideo:nil]; }
/* Now create a new movie player using the URL */  self.moviePlayer = [[MPMoviePlayerController alloc] initWithContentURL:url];
if (self.moviePlayer != nil){
/* Listen for the notification that the movie player sends us whenever it finishes playing an audio file */
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(videoHasFinishedPlaying:) name:MPMoviePlayerPlaybackDidFinishNotification object:self.moviePlayer];
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(videoThumbnailIsAvailable:)
name:MPMoviePlayerThumbnailImageRequestDidFinishNotification object:self.moviePlayer];
NSLog(@"Successfully instantiated the movie player.");
/* Scale the movie player to fit the aspect ratio */ self.moviePlayer.scalingMode = MPMovieScalingModeAspectFit;
/* Let's start playing the video in full screen mode */ [self.moviePlayer play];
[self.view addSubview:self.moviePlayer.view];
[self.moviePlayer setFullscreen:YES animated:YES];
/* Capture the frame at the third second into the movie */ NSNumber *thirdSecondThumbnail = [NSNumber numberWithFloat:3.0f];
/* We can ask to capture as many frames as we
want. But for now, we are just asking to capture one frame */
NSArray *requestedThumbnails =  [NSArray arrayWithObject:thirdSecondThumbnail];
/* Ask the movie player to capture this frame for us */ [self.moviePlayer
requestThumbnailImagesAtTimes:requestedThumbnails timeOption:MPMovieTimeOptionExact];
} else {
NSLog(@"Failed to instantiate the movie player."); }
}
```

你可以看到，我们我们要求视频播放器捕获影片中第三秒的那个帧。一旦这个任务完成，我们的视图控制器的 videoThumbnailIsAvailable:实例方法就会被调用，下面的代码说明了我们如何能访问被捕获的图像：

```
- (void) videoThumbnailIsAvailable:(NSNotification *)paramNotification{ MPMoviePlayerController *controller =
[paramNotification object];
if (controller != nil &&
[controller isEqual:self.moviePlayer]){ NSLog(@"Thumbnail is available");
/* Now get the thumbnail out of the user info dictionary */ UIImage *thumbnail =  [paramNotification.userInfo
objectForKey:MPMoviePlayerThumbnailImageKey];
if (thumbnail != nil){  /* We got the thumbnail image. You can now use it here */
```

```
}}
}
```

因为我们在 startPlayingVideo: 中实例化视频播放器时，开始了监听 MPMoviePlayerThumbnailImageRequestDidFinishNotifi cation

通知，所以也必须在我们停止视频播放器时，停止对这个通知的监听（或者任何你在你的应用框架中认为恰当的时间）。

```
- (void) stopPlayingVideo:(id)paramSender { if (self.moviePlayer != nil){
[[NSNotificationCenter defaultCenter] removeObserver:self name:MPMoviePlayerPlaybackDidFinishNotification
object:self.moviePlayer];
[[NSNotificationCenter defaultCenter]  removeObserver:self name:MPMoviePlayerThumbnailImageRequestDidFinishNotification
object:self.moviePlayer];
[self.moviePlayer stop];
if ([self.moviePlayer.view.superview isEqual:self.view]){ [self.moviePlayer.view removeFromSuperview];
}}
}
```

当在调用 MPMoviePlayerController 的 requestThumbnailImagesAtTimes:timeOption:实例方法时，我们可以为 timeOption: 指定 MPMovie TimeOptionExact 或者 MPMovieTimeOptionNearestKeyFrame 其中之一。前者捕获视频时间线上精确的点，后者虽然不那么精确，但使用更少的系统资源，并在捕获缩略图时整体上有更好的表现。MPMovieTimeOptionNearestKey Frame 通常能胜任精确要求，因为它仅仅有那么一两帧的误差。

9.7.4. 参考

9.8. 访问音乐库

9.8.1. 问题

你想要让用户访问从她的音乐库中选取。

9.8.2. 解决

使用 MPMediaPickerController 类：

```
MPMediaPickerController *mediaPicker = [[MPMediaPickerController alloc]
initWithMediaTypes:MPMediaTypeAny];
```

9.8.3. 讨论

MPMediaPickerController 是一个 iPod 程序显示给用户的视图控制器。只要实例化 MPMediaPickerController，你就可以为你的用户展现一个标准的视图控制器，让他们任意从音乐库中进行选择，然后控制重新返回到你的程序中。这点对游戏特别有用，例如，用户玩游戏时，可以让你的程序在背景播放它最喜爱的歌曲。

你可以用成为媒体选择控制器的委托（遵守 MPMediaPickerControllerDelegate 协议）的

方式从它那里获得信息：

```
#import <UIKit/UIKit.h>
#import <MediaPlayer/MediaPlayer.h>
@interface Accessing_the_Music_LibraryViewController
: UIViewController <MPMediaPickerControllerDelegate>
@end
```

在你的 displayMediaPicker:选择器中，实现所需的代码，来显示媒体选择控制器实例并且向用户展现一个模态视图控制器：

```
- (void) displayMediaPicker{
MPMediaPickerController *mediaPicker = [[MPMediaPickerController alloc]
initWithMediaTypes:MPMediaTypeAny];
if (mediaPicker != nil){
NSLog(@"Successfully instantiated a media picker.");
mediaPicker.delegate = self;
mediaPicker.allowsPickingMultipleItems = NO;
[self.navigationController presentModalViewController:mediaPicker
animated:YES];
} else {
NSLog(@"Could not instantiate a media picker.");
}
}
```

媒体选择控制器的 allowsPickingMultipleItems 属性，让你指定用户能否在隐藏控制器之前选择多于一个物品。它接受一个 BOOL 值，目前我们就把它设置为 NO，稍后我们会看到效果。现在，让我们来实现 MPMediaPickerControllerDelegate 协议的各种委托消息：

```
- (void) mediaPicker:(MPMediaPickerController *)mediaPicker
didPickMediaItems:(MPMediaItemCollection *)mediaItemCollection{
NSLog(@"Media Picker returned");
for (MPMediaItem *thisItem in mediaItemCollection.items){
NSURL *itemURL =
[thisItem valueForProperty:MPMediaItemPropertyAssetURL];
NSString *itemTitle =
[thisItem valueForProperty:MPMediaItemPropertyTitle];
NSString *itemArtist =
[thisItem valueForProperty:MPMediaItemPropertyArtist];
MPMediaItemArtwork *itemArtwork =
[thisItem valueForProperty:MPMediaItemPropertyArtwork];
NSLog(@"Item URL = %@", itemURL);
NSLog(@"Item Title = %@", itemTitle);
NSLog(@"Item Artist = %@", itemArtist);
NSLog(@"Item Artwork = %@", itemArtwork);
}
[mediaPicker dismissModalViewControllerAnimated:YES];
}
```

你可以使用 MPMediaItem 的 valueForProperty:实例方法访问不同的属性。这个类的实例通过 mediaPicker:didPick MediaItems:委托消息的 mediaItemCollection 参数返回到你的程序中。

现在让我们写一个带有很简单的 GUI 的一个程序，它允许我们询问用户，来从 iPod 库

中选择一个音乐。在她选择了音乐文件后，我们尝试使用 MPMusicPlayerController 实例来播放它。我们的 GUI 有个简单的按钮：“选择”和“播放”，以及“停止播放”。第一个按钮会要用户从 iPod 库中选择一段音频给我们播放，第二个按钮会停止音频回放（如果我们已经在播放音频了）。我们会从程序的 UI 设计开始。让我们以简单的方式创建，就像图 9-1 那样。

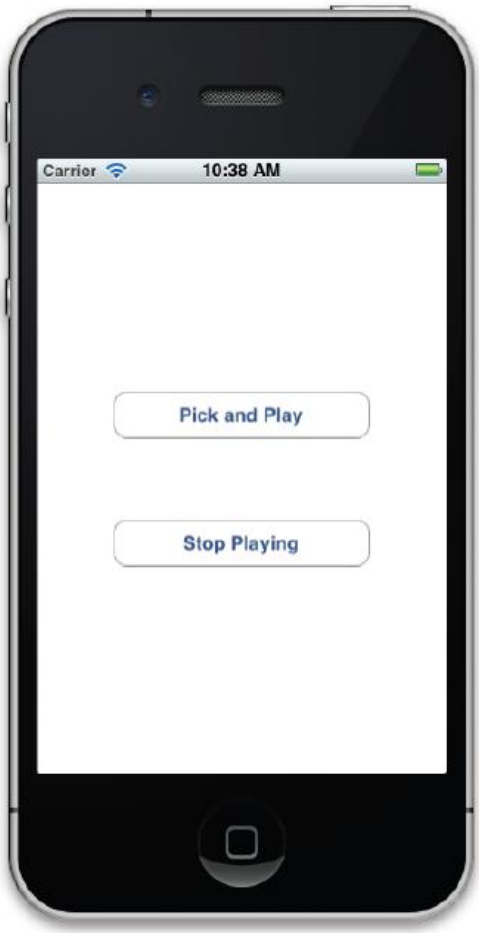

图 9-1 一个极简单的 UI，展示了媒体选择器和 AV 音频播放器

现在让我们继续在视图控制器的.h 中定义这两个按钮：

```
@interface Accessing_the_Music_LibraryViewController : UIViewController
<MPMediaPickerControllerDelegate, AVAudioPlayerDelegate>
@property (nonatomic, strong) MPMusicPlayerController *myMusicPlayer;
@property (nonatomic, strong) UIButton *buttonPickAndPlay;
@property (nonatomic, strong) UIButton *buttonStopPlaying;
@end
```

当我们的视图加载完毕，我们会实例化这两个按钮们将他们放置于视图之上：

```
- (void)viewDidLoad {
[super viewDidLoad];
self.view.backgroundColor = [UIColor whiteColor];
self.buttonPickAndPlay = [UIButton buttonWithType:UIButtonTypeRoundedRect];
self.buttonPickAndPlay.frame = CGRectMake(0.0f,
0.0f,
200,
```

```
37.0f);
self.buttonPickAndPlay.center = CGPointMake(self.view.center.x,
self.view.center.y - 50);
[self.buttonPickAndPlay setTitle:@"Pick and Play"
forState:UIControlStateNormal];
[self.buttonPickAndPlay addTarget:self
action:@selector(displayMediaPickerAndPlayItem)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:self.buttonPickAndPlay];
self.buttonStopPlaying = [UIButton buttonWithType:UIButtonTypeRoundedRect];
self.buttonStopPlaying.frame = CGRectMake(0.0f,
0.0f,
200,
37.0f);
self.buttonStopPlaying.center = CGPointMake(self.view.center.x,
self.view.center.y + 50);
[self.buttonStopPlaying setTitle:@"Stop Playing"
forState:UIControlStateNormal];
[self.buttonStopPlaying addTarget:self
action:@selector(stopPlayingAudio)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:self.buttonStopPlaying];
[self.navigationController setNavigationBarHidden:YES
animated:NO];
}
```

视图控制器的两个最重要的方法是 displayMediaPickerAnd PlayItem 和 stopPlayingAudio:

```
- (void) stopPlayingAudio{
if (self.myMusicPlayer != nil){
[[NSNotificationCenter defaultCenter]
removeObserver:self
name:MPMusicPlayerControllerPlaybackStateDidChangeNotification
object:self.myMusicPlayer];
[[NSNotificationCenter defaultCenter]
removeObserver:self
name:MPMusicPlayerControllerNowPlayingItemDidChangeNotification
object:self.myMusicPlayer];
[[NSNotificationCenter defaultCenter]
removeObserver:self
name:MPMusicPlayerControllerVolumeDidChangeNotification
object:self.myMusicPlayer];
[self.myMusicPlayer stop];
}
}
- (void) displayMediaPickerAndPlayItem{
MPMediaPickerController *mediaPicker =
[[MPMediaPickerController alloc]
initWithMediaTypes:MPMediaTypeMusic];
if (mediaPicker != nil){
NSLog(@"Successfully instantiated a media picker.");
mediaPicker.delegate = self;
mediaPicker.allowsPickingMultipleItems = YES;
[self.navigationController presentModalViewController:mediaPicker
animated:YES];
} else {
NSLog(@"Could not instantiate a media picker.");
}
```

```
}
```

当我们的媒体选择控制器成功完成，mediaPicker:didPickMediaItems 消息会在委托对象中被调用（这个例子中就是视图控制器）。另一方面，如果用户取消了媒体播放器，我们会得到 mediaPicker:mediaPickerDid Cancel 消息。下面的代码实现了每种情况下将会被调用的方法：

```
- (void) mediaPicker:(MPMediaPickerController *)mediaPicker
didPickMediaItems:(MPMediaItemCollection *)mediaItemCollection{
NSLog(@"Media Picker returned");
/* First, if we have already created a music player, let's
deallocate it */
self.myMusicPlayer = nil;
self.myMusicPlayer = [[MPMusicPlayerController alloc] init];
[self.myMusicPlayer beginGeneratingPlaybackNotifications];
/* Get notified when the state of the playback changes */
[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(musicPlayerStateChanged:)
name:MPMusicPlayerControllerPlaybackStateDidChangeNotification
object:self.myMusicPlayer];
/* Get notified when the playback moves from one item
to the other. In this recipe, we are only going to allow
our user to pick one music file */
[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(nowPlayingItemIsChanged:)
name:MPMusicPlayerControllerNowPlayingItemDidChangeNotification
object:self.myMusicPlayer];
/* And also get notified when the volume of the
music player is changed */
[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(volumeIsChanged:)
name:MPMusicPlayerControllerVolumeDidChangeNotification
object:self.myMusicPlayer];
/* Start playing the items in the collection */
[self.myMusicPlayer setQueueWithItemCollection:mediaItemCollection];
[self.myMusicPlayer play];
/* Finally dismiss the media picker controller */
[mediaPicker dismissModalViewControllerAnimated:YES];
}
- (void) mediaPickerDidCancel:(MPMediaPickerController *)mediaPicker{
/* The media picker was cancelled */
NSLog(@"Media Picker was cancelled");
[mediaPicker dismissModalViewControllerAnimated:YES];
}
```

我么在监听音乐播放器通过通知发送的事件。下面是用于处理被监听的音乐播放器通知消息的三个方法：

```
- (void) musicPlayerStateChanged:(NSNotification *)paramNotification{
NSLog(@"Player State Changed");
/* Let's get the state of the player */
NSNumber *stateAsObject =
[paramNotification.userInfo
```

```
objectForKey:@"MPMusicPlayerControllerPlaybackStateKey"];
NSInteger state = [stateAsObject integerValue];
/* Make your decision based on the state of the player */
switch (state){
case MPMusicPlaybackStateStopped:{
/* Here the media player has stopped playing the queue. */
break;
}
case MPMusicPlaybackStatePlaying:{
/* The media player is playing the queue. Perhaps you
can reduce some processing that your application
that is using to give more processing power
to the media player */
break;
}
case MPMusicPlaybackStatePaused:{
/* The media playback is paused here. You might want
to indicate by showing graphics to the user */
break;
}
case MPMusicPlaybackStateInterrupted:{
/* An interruption stopped the playback of the media queue */
break;
}
case MPMusicPlaybackStateSeekingForward:{
/* The user is seeking forward in the queue */
break;
}
case MPMusicPlaybackStateSeekingBackward:{
/* The user is seeking backward in the queue */
break;
}
} /* switch (State){ */
}
- (void) nowPlayingItemIsChanged:(NSNotification *)paramNotification{
NSLog(@"Playing Item Is Changed");
NSString *persistentID =
[paramNotification.userInfo
objectForKey:@"MPMusicPlayerControllerNowPlayingItemPersistentIDKey"];
/* Do something with Persistent ID */
NSLog(@"Persistent ID = %@", persistentID);
}
- (void) volumeIsChanged:(NSNotification *)paramNotification{
NSLog(@"Volume Is Changed");
/* The userInfo dictionary of this notification is normally empty */
}
```

我们还会实现视图控制器的 viewDidUnload 方法，来确保不会留下任何内存泄漏：

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

接下来，运行我们的程序，点击视图控制器的“点击”和“播放”按钮，我们会看到媒体选择控制器。一旦控制器出现，和 iPod 相同的 UI 将展现给用户。在用户选择了一项之后（或者取消整个对话框），我们会在视图控制器中得到相应的委托消息（当我们的视图控制器是媒体选择器的视图控制器时）。在物品被选择后（在这里我们只允许选一个），我们会

开始我们的音乐播放器，并开始播放整个选择集合。

如果你想允许用户一次选择多于一项，只要设置你的媒体选择器的 allowsPickingMultipleItems 属性为 YES：

```
mediaPicker.allowsPickingMultipleItems = YES;
```

有时，当在使用媒体选择控制器时（MPMedia PickerController），“MPMediaPicker：失去到 iPod 库的连接”消息将会输出到控制台屏幕。这是因为媒体选择器在显示给用户时，因为诸如 iTunes 同步这样的事件而被打断。这时，你的 mediaPickerDidCancel:委托方法立刻会被调用。

9.8.4. 参考

点击这里访问：DevDiv.com 移动开发论坛