iPhone与iPad应用开发课程
精通iOS开发

第十二讲 多媒体API

主讲人：关东升

eorient@sina.com

主要知识点

- 播放视频
- 播放音频
- 录制音频

视频文件介绍

- 视频格式可以分为适合本地播放的本地影像视频和适合在网络中播放的网络流媒体影像视频两大类。尽管后者在播放的稳定性和播放画面质量上可能没有前者优秀，但网络流媒体影像视频的广泛传播性使之正被广泛应用于视频点播、网络演示、远程教育、网络视频广告等等互联网信息服务领域。

适合移动设备的视频文件

- 3GP，3GP是一种3G流媒体的视频编码格式，主要是为了配合3G网络的高传输速度而开发的，也是目前手机中最为常见的一种视频格式。
- 视频MP4格式，除了支持MP3所具有的音乐播放功能外，还具备强大的MPEG-4视频播放能力。
- iPhone中还支持mov格式文件。

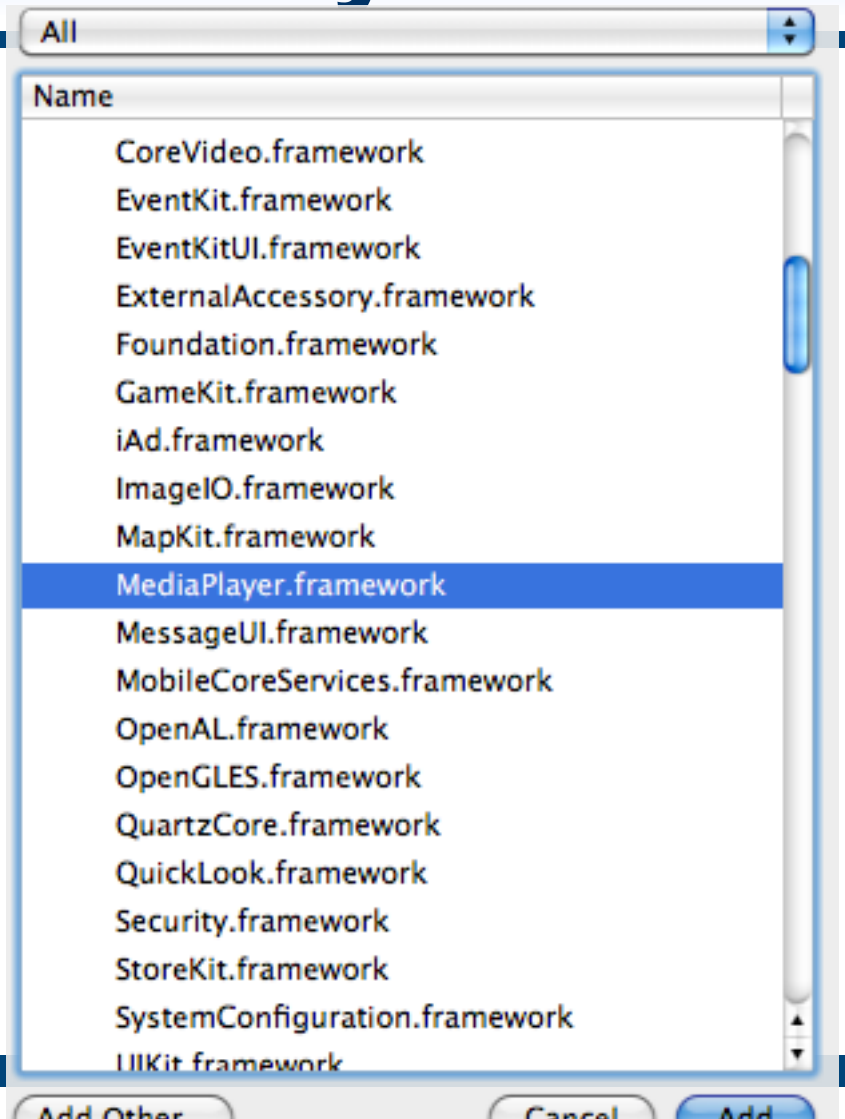
iOS播放视频

- iOS sdk为播放视频提供了非常简便方法，提供的MPMoviePlayerViewController类作为开发使用，在iOS4以前的版本是MPMoviePlayerController。
- 在iPhone开发规范中禁止使用私有API播放视频，因此播放画面的控制的控件都是有iPhone提供好的，我们没有别的选择。我们能做的：
  - 加载URL中视频
  - 播放、暂停视频
  - 用户控制行为和缩放模式
  - 产生通知

视频播放案例

MoviePlayer

添加 MediaPlayer.framework



MoviePlayerViewController.h

```
#import <UIKit/UIKit.h>
#import <MediaPlayer/MediaPlayer.h>

@interface MoviePlayerViewController : UIViewController {

    MPMoviePlayerViewController * moviePlayerView;
}

@property (nonatomic, retain) MPMoviePlayerViewController * moviePlayerView;

-(IBAction) playMovie: (id) sender;
- (void) playingDone;

@end
```

m文件的加载和卸载方法

```
- (void) viewDidLoad {
    [[NSNotificationCenter defaultCenter] addObserver:self
        selector: @selector (playingDone)
        name:MPMoviePlayerPlaybackDidFinishNotification object:nil];
}
- (void)dealloc {
    [[NSNotificationCenter defaultCenter] removeObserver:self];
    [moviePlayerView release];
    [super dealloc];
}
```

说明

- MPMoviePlayerViewController提供了在播放过程中的状态改变和其它事件的通知。在viewDidLoad注册了一个播放完成的通知，常用的通知有：
  - MPMoviePlayerPlaybackDidFinishNotification通知接收者播放结束。
  - MPMoviePlayerScalingModeDidChangeNotification改变影片的尺寸。
  - MPMoviePlayerContentPreloadDidFinishNotification表示预处理以及完成，准备开始播放影片。
- dealloc方法中的[[NSNotificationCenter defaultCenter] removeObserver:self];影片播放完成要注销通知。

播放事件

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

    moviePlayerView = [[MPMoviePlayerViewController alloc]
                       initWithContentURL:[NSURL fileURLWithPath:[[NSBundle mainBundle]
                                           pathForResource:@"short" ofType:@"3gp"]]];

    moviePlayerView.moviePlayer.controlStyle = MPMovieControlStyleFullscreen;
    // MPMovieControlStyleNone
    //MPMovieControlStyleEmbedded
    //MPMovieControlStyleDefault

    //[movieplayer play];
    //在当前view上添加视频的视图
  [[[UIApplication sharedApplication] keyWindow] addSubview:moviePlayerView.view];

}
```

说明

- 视频文件可以播放资源目录、沙箱目录和网络播放。本例中我们采用资源目录。
- moviePlayerView.moviePlayer属性是MPMoviePlayerController类型，它有的controlStyle属性可以控制播放行为，它的取值有：
  - MPMovieControlStyleFullscreen
  - MPMovieControlStyleNone没有播放控件
  - MPMovieControlStyleEmbedded
  - MPMovieControlStyleDefault

说明

- MPMoviePlayerController类还有scalingMode属性用于控制影片的尺寸，它的取值有：

MPMovieScalingModeNone原始尺寸

MPMovieScalingModeAspectFit缩放到一个填充方向

MPMovieScalingModeAspectFill填充两边可能会切除一部分

MPMovieScalingModeFill填充两边可能会改变比例

播放完成

```
- (void) playingDone {
    NSLog(@"播放完成");
    [moviePlayerView.view removeFromSuperview];
    [moviePlayerView release];
    moviePlayerView = nil;
}
```

- playingDone 方法是在影片播放完成时候调用，这是因为我们在通知中心注册的方法。
- 播放完成需要把播放视图remove这样才可以获得上一个屏幕。

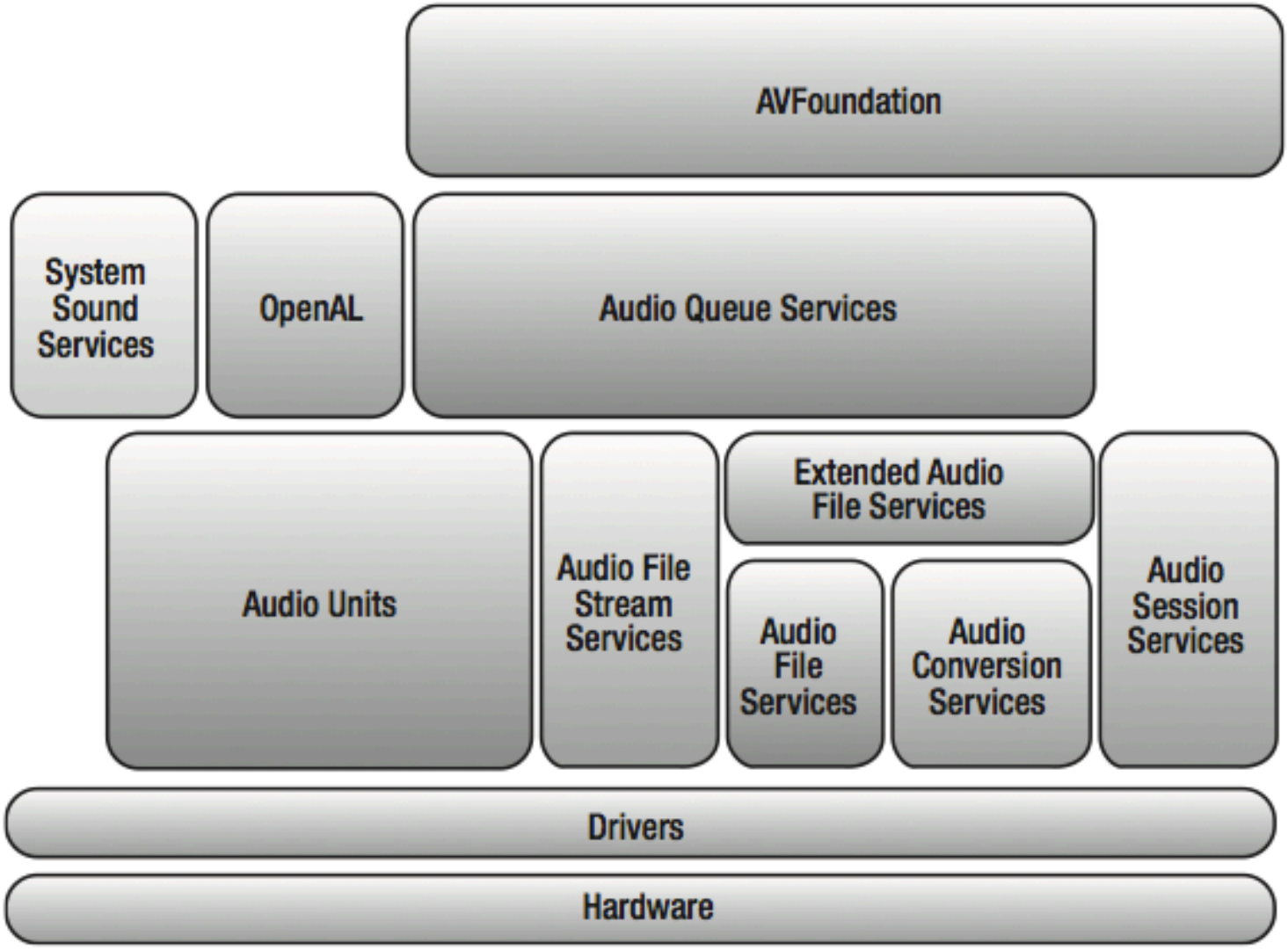
音频文件介绍

- 有两类主要的音频文件格式：
  - 无损格式，例如WAV，PCM，TTA，FLAC，AU，APE，TAK，WavPack(WV)，CAF
  - 有损格式，例如MP3，Windows Media Audio（WMA），Ogg Vorbis（OGG），AAC

移动音频文件

- 作为移动设备音频文件应该原则上比较小，一般的格式：
  - WAV、由于无损压缩效果最好。
  - MP3、有损压缩，文件比较小，由于去除的是人类无法感应到的声音，效果也很好。这是目前常用格式。
  - AAC、压缩比例更大，比MP3文件还要小。
  - CAF（Core Audio Format）是Apple专用的无损压缩格式。

Core Audio


AVFoundation
System Sound Services
OpenAL
Audio Queue Services
Audio Units
Audio File Stream Services
Extended Audio File Services
Audio File Services
Audio Conversion Services
Audio Session Services
Drivers
Hardware

说明

- 高级API，易用
  - System Sound API –播放短声音、警告音等。
  - AVFoundation 可以播放长时间声音，简单易用。
- 低级API，能够对音频有更多的控制
  - Audio Toolbox – 录制、播放、音频流有全面的控制。
  - OpenAL – 播放立体声，常用于游戏。

System Sound API

- System Sound 可以播放“短的”声音，所谓短声音就是5秒以内。
- 不循环、没有声音控制、立即播放。
- 播放格式限制：
  - 线性PCM 和 IMA4
  - .caf .aif 或 .wav

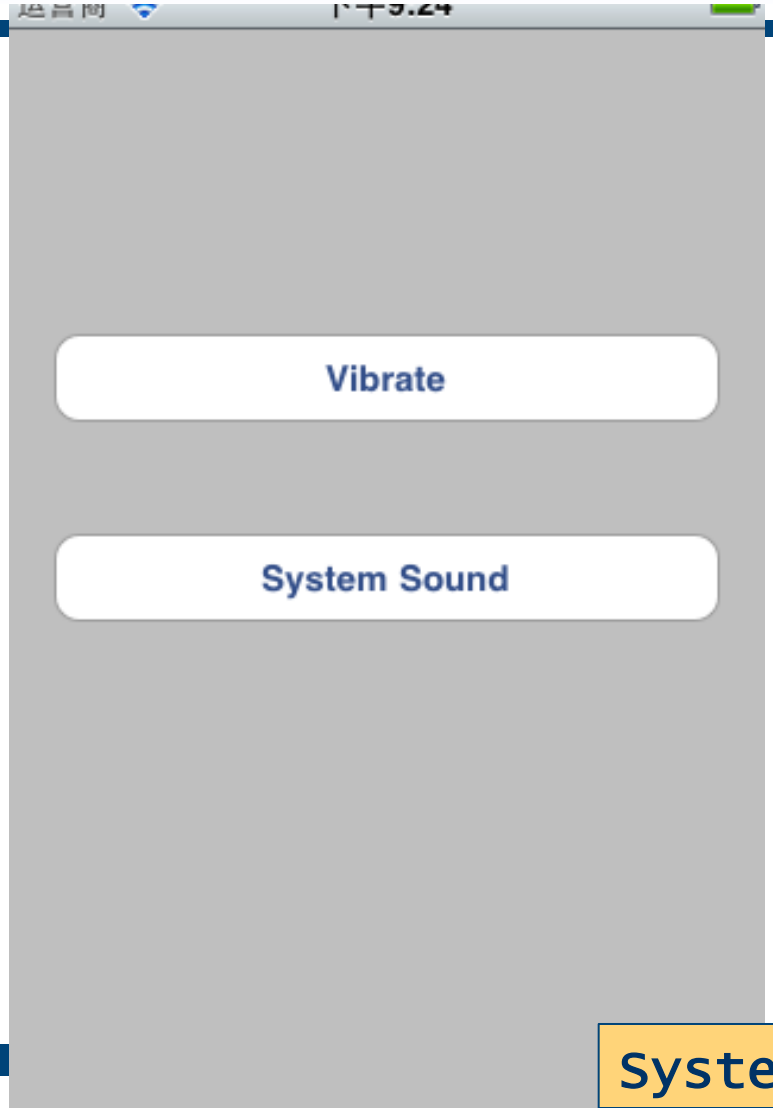
播放“短声音”

- 播放“短声音”主要就是两个步骤：
  - 注册声音
  - AudioServicesCreateSystemSoundID ((CFURLRef)fileURL, &myID);
  - 播放声音
  - AudioServicesPlaySystemSound (myID);
- 监听完成事件方法
  - AudioServicesAddSystemSoundCompletion
- 清除播放sound ID
  - SystemSoundID myID;
  - AudioServicesDisposeSystemSoundID (myID);

震动

- 也可以通过System Sound API让iPhone震动，但是iPod touch不能震动。
- 震动可以通过指定一个特殊的system sound ID——kSystemSoundID_Vibrate实现。
  - AudioServicesPlaySystemSound (kSystemSoundID_Vibrate);

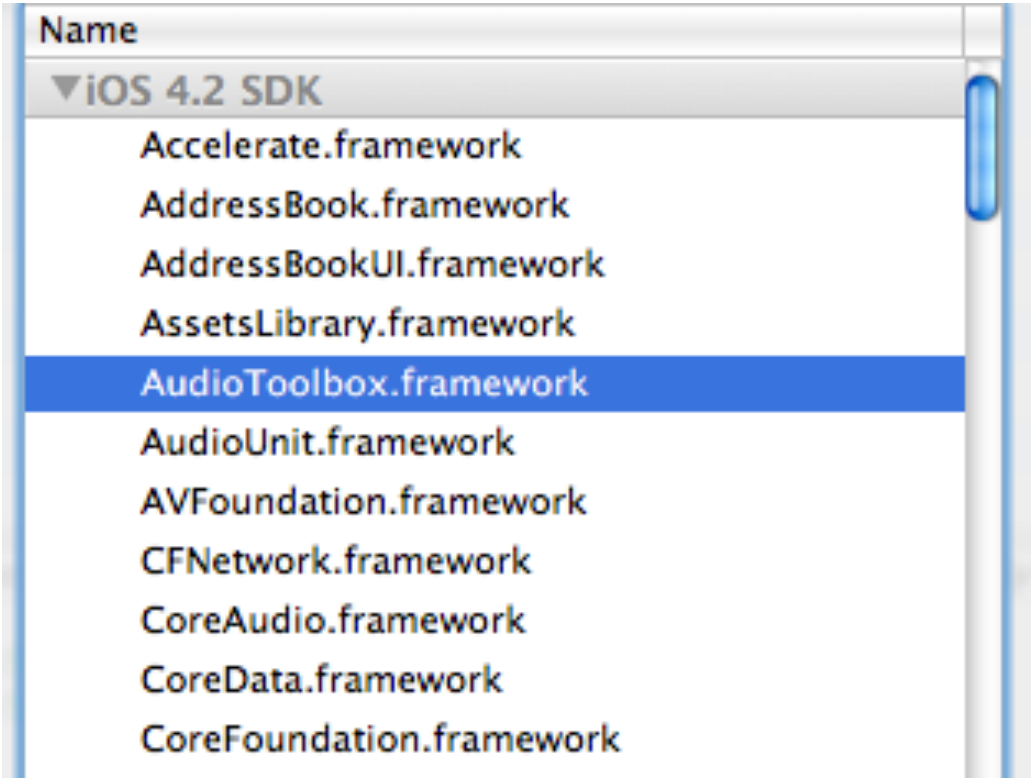
实例



SystemSoundServices

添加框架

- 添加AudioToolbox.framework框架



SystemSoundServicesViewController.h文件

```
#import <UIKit/UIKit.h>
#include <AudioToolbox/AudioToolbox.h>

@interface SystemSoundServicesViewController : UIViewController;

-    (IBAction) playSystemSound;
 - (IBAction) vibrate;

@end
```

播放事件

```
- (IBAction) playSystemSound {
        NSURL* system_sound_url = [NSURL fileURLWithPath:[[NSBundle mainBundle]
                    pathForResource:@"BeepGMC500" ofType:@"wav"]];
        SystemSoundID system_sound_id;
        AudioServicesCreateSystemSoundID(
                (CFURLRef)system_sound_url,
                &system_sound_id
        );
    // Register the sound completion callback.
    AudioServicesAddSystemSoundCompletion(
                system_sound_id,
                NULL, // uses the main run loop
                NULL, // uses kCFRunLoopDefaultMode
                MySoundFinishedPlayingCallback, // the name of our custom callback function
                NULL // for user data, but we don't need to do that in this case, so we just pass NULL
        );
        // Play the System Sound
        AudioServicesPlaySystemSound(system_sound_id);
}
```

说明

- AudioServicesAddSystemSoundCompletion方法5个参数，第一参数SystemSoundID，第二参数是是否使用循环，第三个参数是循环模式，第四个参数是回调函数，就是当播放完成时候回调的方法，第五个参数是为回调函数提供参数。
- 这里回调的方法是C语言风格的函数：MySoundFinishedPlayingCallback。

回调函数

```
void MySoundFinishedPlayingCallback(SystemSoundID sound_id, void* user_data)
{
    AudioServicesDisposeSystemSoundID(sound_id);
}
```

震动方法调用

```
// Vibrate on action
- (IBAction) vibrate
{
    AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
}
```

播放和录制音频

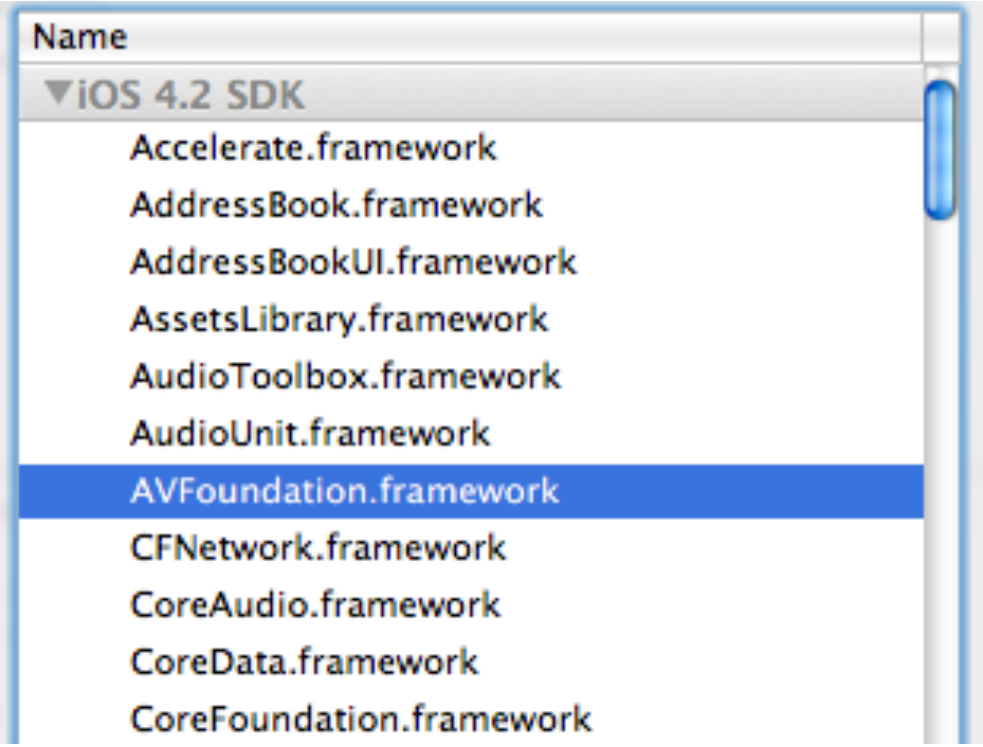
- AVFoundation控件可以实现一般音频播放和录制。
  - AVAudioPlayer音频播放类，用于播放大于5秒钟声音，可以播放本地声音，但是不能播放网络媒体文件。能够播放、暂停、循环和跳过等操作。
  - AVAudioRecorder音频录制类。

AVAudioPlayer

Avplayer

添加框架

- 添加AVFoundation.framework框架



AvplayerViewController.h文件

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>

@interface AvplayerViewController : UIViewController <AVAudioPlayerDelegate> {
        AVAudioPlayer * player;

}

- (IBAction) stopSong: (id) sender;
- (IBAction) playSong: (id) sender;


@end
```

AvplayerViewController.m

```
- (IBAction) playSong: (id) sender {
        NSError *error = nil;
        player = [[AVAudioPlayer alloc] initWithContentsOfURL:
                        [NSURL fileURLWithPath:[[NSBundle mainBundle]
                                pathForResource:@"charleston1925_64kb" ofType:@"mp3"]] error:&error];
        player.delegate = self;
        if(error) {
                NSLog(@"%@",[error description]);
                [error release];
        }
        [player play];
}
```

AvplayerViewController.m

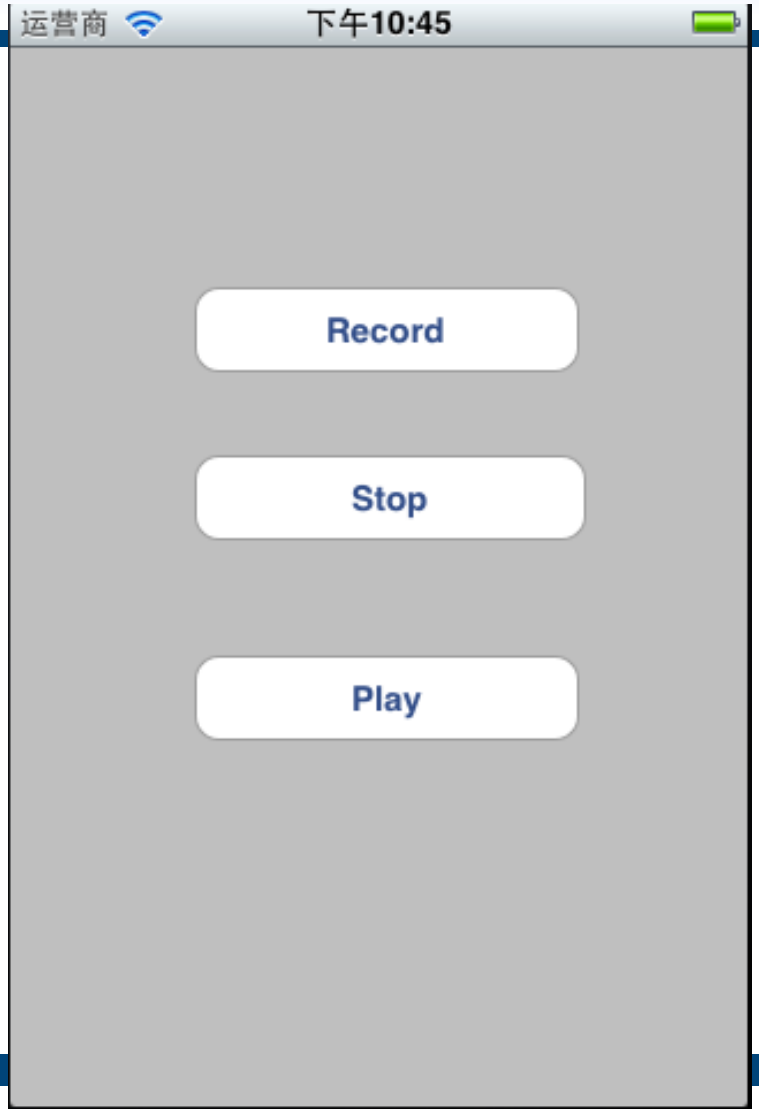
```
- (IBAction) stopSong: (id) sender {
        [player stop];
}
- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player successfully:(BOOL)flag {
        NSLog(@"播放完成。");
}

- (void)audioPlayerDecodeErrorDidOccur:(AVAudioPlayer *)player error:(NSError *)error {
        NSLog(@"播放错误发生: %@", [error localizedDescription]);
}
- (void)dealloc {
        [player release];
    [super dealloc];
}
```

AVAudioPlayer委托

- AVAudioPlayerDelegate委托对象提供了两个主要方法：
  - audioPlayerDidFinishPlaying:successfully:
  - audioPlayerDecodeErrorDidOccur:error:

AVAudioRecorder



Recorder

RecorderViewController.h文件

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>

@interface RecorderViewController : UIViewController <AVAudioPlayerDelegate>
{
    AVAudioRecorder *recorder;
    AVAudioPlayer *player;
        UILabel *label;
}
@property (retain, nonatomic) AVAudioRecorder * recorder;
@property (retain, nonatomic) AVAudioPlayer * player;
@property (retain, nonatomic) IBOutlet UILabel *label;
-(IBAction)recordPushed:(id)sender;
-(IBAction)playPushed:(id)sender;
-(IBAction)stopPushed:(id)sender;
@end
```

音频录制方法

```
-(IBAction)recordPushed:(id)sender{
    label.text = @"recode...";
    if([recorder isRecording])
            return;
    if([player isPlaying])
            [player stop];
    NSError *error = nil;
    [[AVAudioSession sharedInstance]
            setCategory:AVAudioSessionCategoryRecord
            error:&error];
    [[AVAudioSession sharedInstance]
            setActive:YES
            error:&error];
```

说明

- AVAudioSession 是iOS提供音频会话类，音频会话是指定应用程序与音频系统如何交互。AVAudioSession通过指定一个音频类别（Category）实现的，音频类别（Category）描述了应用程序使用音频的方式。下面是语句是设定音频会话类别：
  - [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryRecord error:&error];
- AVAudioSessionCategoryRecord代表只能输入音频，即录制音频了。其效果是停止其它音频播放。
- 使用类别后，音频会话要设置为“活跃的”Active，这会把后台的任何系统声音关闭。
  - [[AVAudioSession sharedInstance] setActive:YES error:&error];

音频录制方法

```
NSMutableDictionary *settings = [NSMutableDictionary dictionary];
  [settings setValue:[NSNumber numberWithInt:kAudioFormatLinearPCM]
          forKey:AVFormatIDKey];
  [settings setValue:[NSNumber numberWithFloat:44100.0]
          forKey:AVSampleRateKey]; //采样率
  [settings setValue:[NSNumber numberWithInt:1]
          forKey:AVNumberOfChannelsKey];//通道的数目
  [settings setValue:[NSNumber numberWithInt:16]
          forKey:AVLinearPCMBitDepthKey];//采样位数 默认 16
  [settings setValue:[NSNumber numberWithBool:NO]
          forKey:AVLinearPCMIsBigEndianKey];//大端还是小端 是内存的组织方式
  [settings setValue:[NSNumber numberWithBool:NO]
          forKey:AVLinearPCMIsFloatKey];//采样信号是整数还是浮点数
```

音频录制方法

```
   NSString *filePath =
    [NSString stringWithFormat:@"%@/rec_audio.caf", [self documentsDirectory]];
    NSURL *fileUrl = [NSURL fileURLWithPath:filePath];
   recorder = [[AVAudioRecorder alloc]
                                initWithURL:fileUrl
                                settings:settings
                                error:&error];
        [recorder record];
}
-(NSString *)documentsDirectory {
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                NSUserDomainMask, YES);
    return [paths objectAtIndex:0];
}
```

音频播放方法

```
-(IBAction)playPushed:(id)sender{
        label.text = @"play...";
    if([recorder isRecording])
        [recorder stop];
    if([player isPlaying])
        [player stop];
    NSString *filePath =
    [NSString stringWithFormat:@"%@/rec_audio.caf", [self documentsDirectory]];
    NSURL *fileUrl = [NSURL fileURLWithPath:filePath];
    NSError *error = nil;
    [[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback
                                        error:&error];
    [[AVAudioSession sharedInstance] setActive:YES error:&error];
    player = [[AVAudioPlayer alloc] initWithContentsOfURL:fileUrl error:&error];
     [player play];
}
```

音频停止方法

```
-(IBAction)stopPushed:(id)sender
{
    label.text = @"stop...";
      if([recorder isRecording])
        [recorder stop];
    if([player isPlaying])
        [player stop];
}
```