Android 音频焦点(Audio Focus)

2013-11-08·Android Audio

#引子

说 Audio Focus 前先说个很简单需求：来电时暂停正在播放的音乐，电话结束时恢复播放。

看到这个需求，第一反应肯定是：监听用户来电状态，作相应操作。这里不多做介绍，这样做有个不好的地方就是需要隐私权限！这样做一点也不优雅。

后来搜索时看到一篇分析文章：Android来电时停止音乐播放的流程（顺便说一嘴，这篇转载居然不注明出处！！）。文章里的分析很明确的指出，系统在框架层就很好的帮我们处理了这个需求，问题是如何将音乐交给系统框架来处理呢？

#音频焦点

问题的解决方法就是：请求系统的音频焦点（Request the Audio Focus)。

如果英文还行，强烈建议请看官方的原文：Managing Audio Playback，里面介绍的很清楚。以下为简单概述。

官方文档指出Android 在处理音频播放是分了多个“音频流”的，如音乐流、音效流、电话声音流等，使控制音量时可以互不干涉。多数情况下我们播放音乐都是使用 STREAM_MUSIC 音频流。

另外，系统中可能会有多个应用程序会播放音频，所以需要考虑他们之间该如何协调，为了避免同时播放音乐，Android 系统使用音频焦点来进行统一管理，即只有获得了音频焦点的应用程序才可以播放音乐。

那么，播放音频应该这样来做：

1. 获取音频焦点 requestAudioFocus
2. 获取成功后，开始播放音频
3. 处理音频焦点的丢失和“DUCK”
4. 播放完毕后取消焦点

如此便可以完美的解决引子里的需求。

#一个简单的示例

MusicService.java · Java

```
1
2   public class MusicService extends Service {
3
4     private AudioManager mAm;
5     private boolean isPlaymusic;
6     private String url;
7     private MediaPlayer mediaPlayer;
8
9     @Override
10    public void onCreate() {
11      super.onCreate();
12      mAm = (AudioManager) getSystemService(AUDIO_SERVICE);
13    }
14
15    @Override
16    public void onStart(Intent intent, int startId) {
17      if (intent != null) {
18        Bundle bundle = intent.getExtras();
19        if (bundle != null) {
20          isPlaymusic = bundle.getBoolean("isPlay", true);
21          url = bundle.getString("url");
22          if (isPlaymusic)
23            play();
24          else
25            stop();
26        }
27      }
28
29    }
30
31    OnAudioFocusChangeListener afChangeListener = new OnAudioFocusChangeListener() {
32      public void onAudioFocusChange(int focusChange) {
33        if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT) {
34          // Pause playback
35          pause();
36        } else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
37          // Resume playback
38          resume();
39        } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
40          // mAm.unregisterMediaButtonEventReceiver(RemoteControlReceiver);
41          mAm.abandonAudioFocus(afChangeListener);
42          // Stop playback
43          stop();
44        }
45
46      }
47    };
48
49    private boolean requestFocus() {
50      // Request audio focus for playback
51      int result = mAm.requestAudioFocus(afChangeListener,
```

```
52          // Use the music stream.
53                  AudioManager.STREAM_MUSIC,
54                  // Request permanent focus.
55                  AudioManager.AUDIOFOCUS_GAIN);
56          return result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED;
57      }
58
59      private void resume() {
60          if (mediaPlayer != null) {
61              mediaPlayer.start();
62          }
63      }
64
65      private void pause() {
66          if (mediaPlayer != null && mediaPlayer.isPlaying()) {
67              mediaPlayer.pause();
68          }
69      }
70      OnCompletionListener completionListener = new OnCompletionListener() {
71
72          @Override
73          public void onCompletion(MediaPlayer player) {
74              if(!player.isLooping()){
75                  mAm.abandonAudioFocus(afChangeListener);
76              }
77          }
78      };
79      private void play() {
80          if (requestFocus()) {
81              if (mediaPlayer == null) {
82                  try {
83                      mediaPlayer = new MediaPlayer();
84                      mediaPlayer.setDataSource(url);
85                      mediaPlayer.prepare();
86                      mediaPlayer.setOnCompletionListener(completionListener);
87                  } catch (IOException e) {
88                      e.printStackTrace();
89                  }
90              }
91
92              if (!mediaPlayer.isPlaying()) {
93                  mediaPlayer.start();
94              }
95          }
96      }
97
98      @Override
99      public void onDestroy() {
100         super.onDestroy();
101         if (mediaPlayer != null)
102             mediaPlayer.release();
103     }
```

```
104
105     private void stop() {
106         if (mediaPlayer != null) {
107             mediaPlayer.stop();
108         }
109     }
110     @Override
111     public IBinder onBind(Intent arg0) {
112         // TODO Auto-generated method stub
113         return null;
114     }
115 }
116
```

经模拟器测试，当来电时音频焦点会给到铃声流，并打出日志：

```
I/AudioService(1235): AudioFocus requestAudioFocus() from AudioFocus_For_Phone_Ring_And_Calls
```

此时 MusicService 中的 afChangeListener 会得到 AUDIOFOCUS_LOSS_TRANSIENT ，于是会暂停播放音频。
当通话结束或者挂掉电话， afChangeListener 会得到 AUDIOFOCUS_GAIN ，于是恢复播放音频。

注意：

1. 播放完毕一定要禁止掉请求的音频焦点 abandonAudioFocus(afChangeListener) ，否则，如果播放完毕后的某个时段刚好有个通话结束，并且此时没有其他的应用占用了焦点，系统会重新通知服务里的 afChangeListener ，导致音频再次的播放。
2. 如果丢失的短暂音频焦点允许DUCK状态 AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK ，在这种情况下，应用程序降低音量继续播放，不需要暂停。再次获取后，恢复原来的音量。

#Android #Audio #Media

« Eclipse打开后挂在Android SDK Loader 0% Volley的几点使用心得 »