Android音频开发之音频采集

未见哥哥 已关注

0.657 2018.03.31 20:34:44 字数 936 阅读 12,102

Android音频开发之音频采集

在 Android 系统中，一般使用 `AudioRecord` 或者 `MediaRecord` 来采集音频。

AudioRecord 是一个比较偏底层的API,它可以获取到一帧帧 PCM 数据，之后可以对这些数据进行处理。

而 MediaRecorder 是基于 AudioRecorder 的 API(最终还是会创建AudioRecord用来与AudioFlinger进行交互)，它可以直接将采集到的音频数据转化为执行的编码格式，并保存。

直播技术采用的就是 AudioRecorder 采集音频数据。

本文主要介绍例如 `AudioRecord` 进行音频的采集。

基本API

- 获取最小的缓冲区大小，用于存放 AudioRecord 采集到的音频数据。

```
static public int getMinBufferSize(int sampleRateInHz, int channelConfig, int audioFor
```

- AudioRecord构造方法

> 根据具体的参数配置，请求硬件资源创建一个可以用于采集音频的 AudioRecord 对象。

参数描述：参考Android音频开发之音频基本概念

- audioResource

  音频采集的来源

- audioSampleRate

  音频采样率

- channelConfig

  声道

- audioFormat

  音频采样精度，指定采样的数据的格式和每次采样的大小。

- bufferSizeInBytes

  AudioRecord 采集到的音频数据所存放的缓冲区大小。

```
//设置采集来源为麦克风
private static final int AUDIO_RESOURCE = MediaRecorder.AudioSource.MIC;
```

```
//设置采样率为44100，目前为常用的采样率，官方文档表示这个值可以兼容所有的设置
private final static int AUDIO_SAMPLE_RATE = 44100;
//设置声道声道数量为双声道
private final static int CHANNEL_CONFIG = AudioFormat.CHANNEL_IN_STEREO;
//设置采样精度，将采样的数据以PCM进行编码，每次采集的数据位宽为16bit。
private final static int AUDIO_FORMAT = AudioFormat.ENCODING_PCM_16BIT;

public AudioRecord(int audioSource, int sampleRateInHz, int channelConfig, int audioFo
```

- 开始采集

> 开始采集之后，状态变为RECORDSTATE_RECORDING 。

```
public void startRecording ()
```

- 读取录制内容，将采集到的数据读取到缓冲区

> 方法调用的返回值的状态码：
> 情况异常：
> 1.ERROR_INVALID_OPERATION if the object wasn't properly initialized
> 2.ERROR_BAD_VALUE if the parameters don't resolve to valid data and indexes.
> 情况正常：the number of bytes that were read

```
public int read (ByteBuffer audioBuffer, int sizeInBytes)
public int read (byte[] audioData, int offsetInBytes, int sizeInBytes)
public int read (short[] audioData, int offsetInShorts, int sizeInShorts)
```

- 停止采集

> 停止采集之后，状态变为 RECORDSTATE_STOPPED 。

```
public void stop ()
```

- 获取AudioRecord的状态

> 用于检测AudioRecord是否确保了获得适当的硬件资源。在AudioRecord对象实例化之后调用。

STATE_INITIALIZED 初始完毕
STATE_UNINITIALIZED 未初始化

```
public int getState ()
```

- 返回当前AudioRecord的采集状态

> public static final int RECORDSTATE_STOPPED = 1; 停止状态
> 调用 void stop() 之后的状态
> public static final int RECORDSTATE_RECORDING = 3;正在采集
> 调用 startRecording () 之后的状态

```
public int getRecordingState()
```

AudioRecord 采集音频的基本流程

- 权限

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- 构造一个 `AudioRecord` 对象。
- 开始采集。
- 读取采集的数据。
- 停止采集。

构造一个 `AudioRecord` 对象

```
AudioRecord audioRecord = new AudioRecord(audioResource, audioSampleRate, channelConfi
```

获取 bufferSizeInBytes 值

> bufferSizeInBytes 是 AudioRecord 采集到的音频数据所存放的缓冲区大小。

注意：这个大小不能随便设置，AudioRecord 提供对应的 API 来获取这个值。

```
this.bufferSizeInBytes = AudioRecord.getMinBufferSize(audioSampleRate, channelConfig,
```

通过 `bufferSizeInBytes` 返回就可以知道传入给 `AudioRecord.getMinBufferSize` 的参数是否支持当前的硬件设备。

```
if (AudioRecord.ERROR_BAD_VALUE == bufferSizeInBytes || AudioRecord.ERROR == bufferSiz
    throw new RuntimeException("Unable to getMinBufferSize");
}

//bufferSizeInBytes is available...
```

开始采集

- 在开始录音之前，首先要判断一下 `AudioRecord` 的状态是否已经初始化完毕了。

```
//判断AudioRecord的状态是否初始化完毕
//在AudioRecord对象构造完毕之后，就处于AudioRecord.STATE_INITIALIZED状态了。
int state = audioRecord.getState();
if (state == AudioRecord.STATE_UNINITIALIZED) {
    throw new RuntimeException("AudioRecord STATE_UNINITIALIZED");
}
```

- 开始采集

```
audioRecord.startRecording();
//开启线程读取数据
new Thread(recordTask).start();
```

读取采集的数据

> 上面提到， `AudioRecord` 在采集数据时会将数据存放到缓冲区中，因此我们只需要创建一个数据流去从缓冲区中将采集的数据读取出来即可。

创建一个 `数据流` ，一边从 `AudioRecord` 中读取音频数据到 `缓冲区` ，一边将 `缓冲区` 中数据写入到 `数据流` 。

因为需要使用IO操作，因此读取数据的过程应该在子线程中执行

```
//创建一个流，存放从AudioRecord读取的数据
File saveFile = new File(Environment.getExternalStorageDirectory(), "audio-record.pcm"
DataOutputStream dataOutputStream = new DataOutputStream(
                new BufferedOutputStream(new FileOutputStream(saveFile)));

private Runnable recordTask = new Runnable() {
    @Override
    public void run() {
        //设置线程的优先级
        android.os.Process.setThreadPriority(android.os.Process.THREAD_PRIOR
        Log.i(TAG, "设置采集音频线程优先级");
        final byte[] data = new byte[bufferSizeInBytes];
        //标记为开始采集状态
        isRecording = true;
        Log.i(TAG, "设置当前当前状态为采集状态");
        //getRecordingState获取当前AudioReroding是否正在采集数据的状态
        while (isRecording && audioRecord.getRecordingState() == AudioRecord
            //读取采集数据到缓冲区中，read就是读取到的数据量
            final int read = audioRecord.read(data, 0, bufferSizeInBytes);
            if (AudioRecord.ERROR_INVALID_OPERATION != read && AudioRecord.E
                //将数据写入到文件中
                dataOutputStream.write(buffer,0,read);
            }
        }
    }
};
```

停止采集

```
/**
 * 停止录音
 */
public void stopRecord() throws IOException {
    Log.i(TAG, "停止录音，回收AudioRecord对象，释放内存");
    isRecording = false;
    if (audioRecord != null) {
        if (audioRecord.getRecordingState() == AudioRecord.RECORDSTATE_RECORDING) {
            audioRecord.stop();
            Log.i(TAG, "audioRecord.stop()");
        }
        if (audioRecord.getState() == AudioRecord.STATE_INITIALIZED) {
            audioRecord.release();
            Log.i(TAG, "audioRecord.release()");
        }
    }
}
```

几个小问题

- 采集数据之后，保存的文件为 `audio-record.pcm` ，这个文件并不能使用普通的播放器播放。它是一个原始的文件，没有任何播放格式，因此就无法被播放器识别并播放。
- 上面的问题可以有两种解决方法
  - 使用 AudioTrack 播放 pcm 格式的音频数据。
  - 将 pcm 数据转化为 wav 格式的数据，这样就可以被播放器识别。

未见哥哥 Android Coder
总资产11 共写了12.4W字 获得638个赞 共297个粉丝

已关注