手说 TTS 开发指南

v1.3

2011 年 6 月

版权所有：手说工作室 shoushuo.com

版本	时间	作者	说明
1.0	2010-11-15	张波	
1.1	2010-12-27		在 onDestroy 方法 unbindService
1.2	2010-12-30		去掉 TTS 初始化; ttsService.initialize()方法调用
1.3	2011-6-12		添加接口方法: void nextNoPauseForCall()

目　　录

1 简介

手说 TTS，是 Android 平台下的中文语音引擎，提供了中文文本到语音的转换。

使用手说 TTS 进行中文文本的朗读，包括中文简繁体、阿拉伯数字、英文字母以及一些符号的混读。

开发人员可以使用手说 TTS 来开发需要中文语音的应用程序，本手册介绍如何使用手说 TTS 来进行语音开发。

2 开发准备

2.1 适用平台

手说 TTS 适用于 Android1.5~Android2.2 系统的手机和移动终端。

2.2 下载

在下面位置 http://shoushuo.com/sstts.html 下载应用安装包 shoushuoTTS.apk 和客户端开发包 shoushuotts.jar。

2.3 安装

在 Android 设备上安装下载的 apk 文件。

点击应用应用图标，进入 TTS 设置视图。

进入后聆听范例正常即可开发和部署中文语音应用了。

3 应用开发

3.1 将客户端包导入工程

将 shoushuotts.jar 导入工程 Build class path。

3.2 进行代码编写

以下以在一个 Activity 开发为例进行说明。

3.2.1 在 Activity 声明 tts 接口和绑定标志

```
private ITts ttsService;
private boolean ttsBound;
```

3.2.2 声明 ServiceConnection

```
private ServiceConnection connection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName className, IBinder iservice) {
        ttsService = ITts.Stub.asInterface(iservice);
        ttsBound = true;

        //在应用第一个使用 TTS 的地方，调用下面的 initialize 方法，比如如果有
        //两个 Activity 都使用手说 TTS，则第二个 Activity 在此不需要再调用。
        try {
            ttsService.initialize();
        } catch (RemoteException e) {
        }
    }

    @Override
    public void onServiceDisconnected(ComponentName arg0) {
        ttsService = null;
        ttsBound = false;
    }
};
```

注意 ttsService.initialize()方法只在第一个使用 TTS 的地方调用一次。

3.2.3 处理 bindService 和 unbindService

下面在 onStart()方法 bindService，在 onDestroy ()方法 unbindService。

特别注意 shoushuoTTS 的 Action 为 com.shoushuo.android.tts.intent.action.InvokeTts。

```
@Override
    protected void onStart() {
        super.onStart();
        if (!ttsBound ) {
            String actionName = "com.shoushuo.android.tts.intent.action.InvokeTts";
            Intent intent = new Intent(actionName);
            this.bindService(intent, connection, Context.BIND_AUTO_CREATE);
        }
    }

@Override
    protected void onDestroy () {
        if (ttsBound ) {
            ttsBound    = false;
            this.unbindService(connection);
        }
       super. onDestroy ();
    }
```

3.2.4 服务接口方法调用

服务绑定后，可以使用 ttsService 对服务接口调用来实现所需的语音功能。

方法描述如下：

public void speak(String text, int queueMode);

朗读文本

text，文本内容

queueMode，排队模式，0 或 TextToSpeech.QUEUE_FLUSH 为立即朗读

1 或 TextToSpeech. QUEUE_ADD 为等待前面的读完再读

public boolean isSpeaking();

是否当前正在朗读

public void stop();

停止朗读

public void playSilence(long duration, int queueMode);

朗读一段时间静音

duration，静音时间，单位为毫秒

queueMode，同前面 speak 方法的 queueMode

void nextNoPauseForCall();

下一句朗读时不受来电影响，一般来电等时间会是手说 TTS 的朗读暂停，如果再调用 speak 方法前调用此方法，则下句朗读不受来电影响。

public void speakSyllable(String syllaContent, int queueMode);

朗读音节序列，一个音节的形式为声母_韵母_声调，音节序列为有空格隔开的多个

音节

syllaContent，音节序列字符串

queueMode，同前面 speak 方法的 queueMode

3.2.5 注册和响应事件

ITtsCallback 为 TTS 服务的一个回调接口，回调接口的实例可以注册到 TTS 服务，目前 ITtsCallback 提供了 speakCompleted()回调方法，当朗读完成时该回调方法被调用。如果需要，客户程序应该声明回调接口实例并实现该回调方法。注意回调方法更新界面要通过 Handler 和 Message 来进行，不能直接更新界面。

```
private final ITtsCallback mCallback = new ITtsCallback.Stub() {
    @Override
    public void speakCompleted() throws RemoteException {
        ttsHandler.sendEmptyMessage(0);
    }
};
private Handler ttsHandler = new Handler() {
      @Override public void handleMessage(Message msg) {
        Toast.makeText(SpeechTest.this, "我的话说完了", Toast.LENGTH_LONG).show();
      }
 };
```

TTS 服务接口提供了回调接口实例的注册和注销的方法，只有当注册后，当事件发生后该回调接口实例的回调方法才能被调用。下面修改前面的 ServiceConnection，在 bindService 后马上注册回调接口实例，在 unbindService()前先注销该实例。

```
private ServiceConnection connection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName className, IBinder iservice) {
```

```
            ttsService = ITts.Stub.asInterface(iservice);
            try {
                ttsService.registerCallback(mCallback);
            } catch (RemoteException e) {
            }
            ttsBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            try {
                ttsService.unregisterCallback(mCallback);
            } catch (RemoteException e) {
            }
            ttsService = null;
            ttsBound = false;
        }
    };
```