.百度ai官网，下载sdk包，连接：http://ai.baidu.com/docs#/ASR-Online-Java-SDK/top

2.官网有demo,可以参考，下面是在下的util类，  
import org.apache.log4j.Logger;  
import org.json.JSONObject;  
  
  
import com.baidu.aip.speech.AipSpeech;  
import com.baidu.aip.util.Util;  
  
  
public class BaiduAipUtils {  
  
protected static Logger logger = Logger.getLogger(BaiduAipUtils.class);  
  
//设置APPID/AK/SK  
    public static final String APP\_ID = "";  
    public static final String API\_KEY = "";  
    public static final String SECRET\_KEY = "";  
  
      
    public static String getBaiduAipResult(String saveUrt,String type){  
    logger.info("百度aip------------->BaiduAipUtils：url="+saveUrt);  
    // 初始化一个FaceClient  
        AipSpeech client = new AipSpeech(APP\_ID, API\_KEY, SECRET\_KEY);  
  
  
        // 可选：设置网络连接参数  
        client.setConnectionTimeoutInMillis(2000);  
        client.setSocketTimeoutInMillis(60000);  
          
        if(type == null)  
        type = "wav";  
          
        if(saveUrt == null)  
        return null;  
   
        JSONObject res = client.asr(saveUrt, type, 16000, null);  
        String works = "";  
        if("0".equals(res.get("err\_no").toString()) && "success.".equals(res.get("err\_msg").toString())){  
        works = res.getJSONArray("result").getString(0);  
        }  
        logger.info(res.toString(2));  
        logger.info(works);  
        return works;  
      
    }  
    /\*\*  
     \* @param client  
     \* @throws Throwable  
     \*/  
      
    public void synthesis(AipSpeech client) throws Throwable  
    {  
        // 对本地语音文件进行识别  
        String path = "D:\\code\\java-sdk\\speech\_sdk\\src\\test\\resources\\16k\_test.pcm";  
        JSONObject asrRes = client.asr(path, "pcm", 16000, null);  
        System.out.println(asrRes);  
  
  
        // 对语音二进制数据进行识别  
        byte[] data = Util.readFileByBytes(path);     //readFileByBytes仅为获取二进制数据示例  
        JSONObject asrRes2 = client.asr(data, "pcm", 16000, null);  
        System.out.println(asrRes);  
  
  
        // 对网络上音频进行识别  
        String url = "http://somehost/res/16k\_test.pcm";  
        String callback = "http://callbackhost/aip/dump";  
        JSONObject res = client.asr(url, callback, "pcm", 16000, null);  
        System.out.println(res);  
    }  
  
  
}

3.视频转码插件 ffmpeg

import java.util.ArrayList;  
import java.util.List;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class FFMpegUtil implements IStringGetter {  
  
  
private int runtime = 0;  
  
  
private String ffmpegUri;//ffmpeg的全路径 如e:/ffmpeg/ffmpeg.exe 或 /root/ffmpeg/bin/ffmpeg  
  
  
private String originFileUri; // 视频源文件地址  
  
  
private enum FFMpegUtilStatus {  
Empty, CheckingFile, GettingRuntime  
};  
  
  
private FFMpegUtilStatus status = FFMpegUtilStatus.Empty;  
  
  
/\*\*  
\*  执行命令组合  
\*/  
private List<String> cmd = new ArrayList<String>();  
  
  
/\*\*  
\* 构造函数  
\*   
\* @param ffmpegUri  
\*            ffmpeg的全路径 如e:/ffmpeg/ffmpeg.exe 或 /root/ffmpeg/bin/ffmpeg  
\* @param originFileUri  
\*            所操作视频文件的全路径 如e:/upload/temp/test.wmv  
\*/  
public FFMpegUtil(String ffmpegUri,String originFileUri) {  
this.ffmpegUri = ffmpegUri;  
this.originFileUri = originFileUri;  
}  
  
  
/\*\*  
\* 获取视频时长  
\*   
\* @return  
\*/  
public int getRuntime() {  
runtime = 0;  
status = FFMpegUtilStatus.GettingRuntime;  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");  
cmd.add(originFileUri);  
CmdExecuter.exec(cmd, this);  
return runtime;  
}  
  
  
/\*\*  
\* 检测文件是否是支持的格式 将检测视频文件本身，而不是扩展名  
\*   
\* @return  
\*/  
public boolean isSupported() {  
isSupported = true;  
status = FFMpegUtilStatus.CheckingFile;  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");  
cmd.add(originFileUri);  
CmdExecuter.exec(cmd, this);  
return isSupported;  
}  
  
  
private boolean isSupported;  
  
  
/\*\*  
\* 生成视频截图  
\*   
\* @param imageSavePath  
\*            截图文件保存全路径  
\* @param screenSize  
\*            截图大小 如640x480  
\*/  
public void makeScreenCut(String imageSavePath) {  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");//设定输入流  
cmd.add(originFileUri);  
cmd.add("-q:v");//质量为基础的VBR  
cmd.add("2");  
cmd.add("-y");//覆盖输出文件  
cmd.add("-f");//设定输出格式  
cmd.add("image2");//格式  
//cmd.add("-ss");//开始时间  
//cmd.add("8");  
//cmd.add("-t");//设置纪录时间  
//cmd.add("0.001");  
//cmd.add("-s");//设定画面的宽与高  
//cmd.add(screenSize);//暂时不要截屏大小  
cmd.add(imageSavePath);  
CmdExecuter.exec(cmd, null);  
}  
  
  
/\*\*  
\* 视频转换  
\*   
\* @param fileSavePath  
\*            文件保存全路径（包括扩展名）如 e:/abc/test.flv  
\* @param screenSize  
\*            视频分辨率 如640x480  
\* @param audioByte  
\*            音频比特率  
\* @param audioCollection  
\*            音频采样率  
\* @param quality  
\*            视频质量(0.01-255)越低质量越好  
\* @param fps  
\*            每秒帧数（15或29.97）  
\*/  
public void videoTransfer(String fileSavePath, String screenSize, int audioByte, int audioCollection,  
double quality, double fps) {  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");  
cmd.add(originFileUri);  
cmd.add("-y");  
cmd.add("-ab");  
cmd.add(Integer.toString(audioByte));  
cmd.add("-ar");  
cmd.add(Integer.toString(audioCollection));  
cmd.add("-qscale");  
cmd.add(Double.toString(quality));  
cmd.add("-r");  
cmd.add(Double.toString(fps));  
cmd.add("-s");  
cmd.add(screenSize);  
cmd.add(fileSavePath);  
CmdExecuter.exec(cmd, null);  
}  
  
  
/\*\*  
\* 视频转换  
\*   
\* @param fileSavePath  
\*            文件保存全路径（包括扩展名）如 e:/abc/test.flv  
\* @param screenSize  
\*            视频分辨率 如640x480  
\* @param audioByte  
\*            音频比特率  
\* @param audioCollection  
\*            音频采样率  
\* @param quality  
\*            视频质量(0.01-255)越低质量越好  
\* @param fps  
\*            每秒帧数（15或29.97）  
\*/  
public void videoTransfer(String fileSavePath) {  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");  
cmd.add(originFileUri);  
cmd.add("-y");  
cmd.add(fileSavePath);  
CmdExecuter.exec(cmd, null);  
}  
  
  
@Override  
public void dealString(String str) {  
  
  
switch (status) {  
case Empty:  
break;  
case CheckingFile: {  
Matcher m = Pattern.compile("Unknown format").matcher(str);  
if (m.find())  
this.isSupported = false;  
break;  
}  
case GettingRuntime: {  
Matcher m = Pattern.compile("Duration: //w+://w+://w+").matcher(str);  
while (m.find()) {  
String msg = m.group();  
msg = msg.replace("Duration: ", "");  
runtime = msg.length();  
}  
break;  
}  
}// switch  
}  
  
/\*\*  
\* 转成pcm  
\* @param fileSavePath  
\*/  
public void videoTransferPcm(String fileSavePath) {  
cmd.clear();  
cmd.add(ffmpegUri);  
cmd.add("-i");  
cmd.add(originFileUri);  
cmd.add("-f");  
cmd.add("s16le");  //16 位有符号数  
cmd.add("-ar");  
cmd.add("16000");    //16000 Hz 采样率  
cmd.add("-ac");  
cmd.add("1");    // 单声道  
cmd.add(fileSavePath);  
CmdExecuter.exec(cmd, null);  
}  
}

 4.执行linux命令，util类

import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.util.List;  
  
public class CmdExecuter {  
/\*\*  
\* 执行指令  
\*   
\* @param cmd  
\*            执行指令  
\* @param getter  
\*            指令返回处理接口，若为null则不处理输出  
\*/  
static public void exec(List<String> cmd, IStringGetter getter) {  
try {  
ProcessBuilder builder = new ProcessBuilder();  
builder.command(cmd);  
builder.redirectErrorStream(true);  
Process proc = builder.start();  
BufferedReader stdout = new BufferedReader(new InputStreamReader(proc.getInputStream()));  
String line;  
while ((line = stdout.readLine()) != null) {  
if (getter != null)  
getter.dealString(line);  
}  
proc.waitFor();  
stdout.close();  
} catch (Exception e) {  
e.printStackTrace();  
}  
}  
}

5.

package com.sinaif.common.utils;  
  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.util.List;  
public class CmdExecuter {  
/\*\*  
\* 执行指令  
\*   
\* @param cmd  
\*            执行指令  
\* @param getter  
\*            指令返回处理接口，若为null则不处理输出  
\*/  
static public void exec(List<String> cmd, IStringGetter getter) {  
try {  
ProcessBuilder builder = new ProcessBuilder();  
builder.command(cmd);  
builder.redirectErrorStream(true);  
Process proc = builder.start();  
BufferedReader stdout = new BufferedReader(new InputStreamReader(proc.getInputStream()));  
String line;  
while ((line = stdout.readLine()) != null) {  
if (getter != null)  
getter.dealString(line);  
}  
proc.waitFor();  
stdout.close();  
} catch (Exception e) {  
e.printStackTrace();  
}  
}  
}

6.ffmpeg  相关操作命令，参考官网：

https://ffmpeg.org/ffmpeg.html，也可以百度别人总结，这里就不赘述了

7.解决ffmpeg   comman not found 问题

[hoover@110 bin]# ffmpeg  -i /usr/local/ffmpeg/15023354983067356.mp4  -y /usr/local/ffmpeg/aa.wav  
-bash: ffmpeg: command not found

写全路径，从根目录开始

[hoover@110 bin]# /usr/local/ffmpeg/bin/ffmpeg -i /usr/local/ffmpeg/15023352284504711.mp4  -f s16le -ar 8000 -ac 1 /usr/local/ffmpeg/dd.pcm

**第二篇**

简单说来，就是无须在开发者的项目中写入代码，或引入 jar 包 。 REST API 就是将音频文件转换成某种特定格式，通过 http 请求发送给百度语音识别的服务器，由百度语音识别的服务器进行语音识别，最后返回识别出的文本。

在我看来，她好在可以很方便的调用，我们自己不用去维护语音识别部分的代码，接入也十分简单，**关键是她是免费的**！

使用的方式简单说来是   
1、根据百度语音识别官方网站提供的   
App ID 和 API Key 获取 accessToken。   
2、根据上一步的 accessToken 连同其它请求参数一起向百度语音识别网关发出请求，获得识别的文本。

是不是觉得和微信公众平台的开发有点像？的确是这样的。微信公众平台的开发的确也是先获取 token，再通过 token 去请求其它数据。

**集成步骤**

本集成步骤参考了《Baidu\_Voice\_REST\_API\_Manual》。建议各位朋友先下载该资料学习。

**第 1 步：注册成为百度开发者，创建应用，得到 API Key 和 Secret Key。**

这一步非常简单，在官网上也有操作提示，这里就不多介绍了。

**第 2 步：开通语音识别服务**

开通语音识别服务的步骤也非常简单，大家可以自行操作或者参考官方文档。这里要注意：“ 语音识别” 服务初次开通成功后即可获得 50000 次/ 日 的在线调用次数配额。

如果我们每天调用的次数大于 50000 次，可以向百度申请提高次数，据说也是免费的，大赞。

**以下的步骤就很关键了，因为我们要开始写代码了。**

**第 3 步：获取 Access Token**

简而言之，就是向百度 OAuth2.0 授权服务的网关发出请求，将返回的数据（一般是字符串）进行解析，解析出我们想要的 Access Token。下面的图片截取自官网文档，写得非常详细了。

说明：其实就是使用 **API Key** 和 **Secret Key** 和一个固定值的参数向百度 OAuth2.0 授权服务的网关发出 POST 请求，如果请求成功，则解析返回的字符串，从中解析出 access token 待用。

为了方便说明问题，下面代码没有应用良好格式，仅仅只是测试的方法，不建议直接应用于生产环境。

本例使用 HttpClient 框架发送 post 请求，HttpClient 的 Gradle 依赖为：

compile "org.apache.httpcomponents:httpclient:4.5.2"

* 1

示例代码：

/\*\*

\* 获取 token，推荐用 POST 方法

\*/

@Test

public void test01(){

try {

CloseableHttpClient httpClient = HttpClients.createDefault();

HttpPost httpPost = new HttpPost("https://openapi.baidu.com/oauth/2.0/token");

List<NameValuePair> nvps = new ArrayList<>();

nvps.add(new BasicNameValuePair("grant\_type","client\_credentials"));

nvps.add(new BasicNameValuePair("client\_id",apiKey));

nvps.add(new BasicNameValuePair("client\_secret",secretKey));

httpPost.setEntity(new UrlEncodedFormEntity(nvps));

ResponseHandler<String> responseHandler = new ResponseHandler(){

@Override

public String handleResponse(HttpResponse response) throws ClientProtocolException, IOException {

int status = response.getStatusLine().getStatusCode();

if (status >= 200 && status < 300) {

HttpEntity entity = response.getEntity();

try {

return entity != null ? EntityUtils.toString(entity) : null;

} catch (ParseException ex) {

throw new ClientProtocolException(ex);

}

} else {

throw new ClientProtocolException("Unexpected response status: " + status);

}

}

};

String responseBody = httpClient.execute(httpPost,responseHandler);

System.out.println(responseBody);

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40

返回数据：

{"access\_token":"24.463f2a9f7ce6721fe4d15568f812c086.2592000.1469627568.282335-7038695","session\_key":"9mzdDxLM148MA1QmcNNrxGfLYBU9VokNbUY\/8WsJ1r4rUSev1bjP9GTKP6L6SVDnjx4BZxE5ZpJoqztA2K7O0MM9l0Z4","scope":"public audio\_voice\_assistant\_get wise\_adapt lebo\_resource\_base lightservice\_public hetu\_basic lightcms\_map\_poi kaidian\_kaidian","refresh\_token":"25.f77abdb8f638404747dd969615c7b557.315360000.1782395568.282335-7038695","session\_secret":"3efb3872a362beacab28879eed85497b","expires\_in":2592000}

* 1
* 2

格式化以后：

我们须要从中解析出 access\_token ， json 字符串解析的框架有很多 fastjson、Jackson、json-lib、gson，这里就不多做介绍了。

**另外还是要说明一下，因为 access\_token 的有效时间是 2592000 （秒），即 30 × 24 × 60 × 60 （秒）， 30 天，所以没有必要每一次请求都去获取 access\_token ，建议把 access\_token 放在应用的缓存里，如果失效了，再去获取，可以提高应用的效率，这一点和微信公众平台开发是一样的。**

**第 4 步：根据 Access Token 调用语音识别接口（隐式发送）**

说明：我们这里采用的是隐式发送，即不发送真实的音频文件，而是发送音频文件转换而成的字节数组。这里一定要看官网说明，严格调用，才会识别出理想的结果。难点是音频格式的转换。

我是看了官网说明文档和示例代码，经过反复调试才得以调用成功的。鉴于这里篇幅的限制，请大家先看官网说明文档，我这里就不复制了。

这里为了方便说明，先上示例代码，同样地该代码只是为了便于说明问题，不建议在生产环境中直接使用：

/\*\*

\* 识别英文

\*/

@Test

public void test02(){

recognize("voice\_en.wav","en");

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7

上面的测试方法调用了语音识别的方法，该方法传递两个参数，一个是文件的全路径，另一个是中文或者英文的参数。

下面

/\*\*

\* 请求语音识别的时候使用

\*/

private static final String speech\_recognition\_url = "http://vop.baidu.com/server\_api";

* 1
* 2
* 3
* 4

private void recognize(String wavName,String language){

File wavFile = new File(wavName);

HttpPost httpPost = null;

CloseableHttpResponse response = null;

CloseableHttpClient httpClient = HttpClients.createDefault();

httpPost = new HttpPost(speech\_recognition\_url);

SpeechRecognitionRequestEntity requestEntity = new SpeechRecognitionRequestEntity();

// 语音压缩的格式：请按照官网文档填写 pcm（不压缩）、wav、opus、speex、amr、x-flac 之一，不区分大小写

requestEntity.setFormat("wav");

// 声道数，仅支持单声道，请填写 1

requestEntity.setChannel("1");

// 采样率，支持 8000 或者 16000 （这个类型是 int ，不能设置为 String 类型，关于采样率如何转换，请见下文）

requestEntity.setRate(16000);

// todo 这里应判断 AccessToken 是否过期，处理异常，如果过期了，应该重新获取 accessToken

requestEntity.setToken("24.463f2a9f7ce6721fe4d15568f812c086.2592000.1469627568.282335-7038695");

// Cuid 貌似可以随意填写

requestEntity.setCuid("goodluck");

requestEntity.setLen(wavFile.length());

// 官网说： speech 要传递真实的语音数据，需要进行 base64 编码

// 重点关注：请见后面封装的方法，就是把一个文件转换成为指定格式的字节数组

requestEntity.setSpeech(handlerWavFile(wavFile));

// 语种选择，中文=zh、粤语=ct、英文=en，不区分大小写，默认中文

requestEntity.setLan(language);

// 关键点 1 ：将请求参数转换为 json 格式

String requestEntityJson = JSON.toJSONString(requestEntity);

// 关键点 2 ：封装 StringEntity ，为解决中文乱码问题，应该设置编码

StringEntity entity = new StringEntity(requestEntityJson.toString(), "UTF-8");

entity.setContentEncoding("UTF-8");

// 关键点 3 ：设置 StringEntity 的 ContentType

entity.setContentType("application/json");

httpPost.setEntity(entity);

ResponseHandler<String> responseHandler = new ResponseHandler<String>() {

@Override

public String handleResponse(HttpResponse response) throws ClientProtocolException, IOException {

String resData = null;

int statusCode = response.getStatusLine().getStatusCode();

if (statusCode >= 200 && statusCode < 300) {

HttpEntity httpEntity = response.getEntity();

resData = EntityUtils.toString(httpEntity,"utf-8");

EntityUtils.consume(httpEntity);

}

return resData;

}

};

try {

String responseStr = httpClient.execute(httpPost,responseHandler);

System.out.println(responseStr);

} catch (IOException e) {

e.printStackTrace();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52

SpeechRecognitionRequestEntity 类（省略了 get 和 set 方法）：

public class SpeechRecognitionRequestEntity {

// 语音压缩的格式

private String format;

/\*\*

\* 注意，采样率的数据类型一定是 int，不能是 String

\*/

// 采样率，支持 8000 或者 16000，在我们的项目中，写 16000

private int rate;

// 声道数，仅支持单声道，请填写 1

private String channel;

// 开发者身份验证密钥

private String token;

// 用户 ID，推荐使用设备 mac 地址 手机 IMEI 等设备唯一性参数

// todo 貌似可以随意填写，唯一即可

private String cuid;

/\*\*

\* 注意：这里填写的是原始语音的长度，不是使用 base64 编码的语音长度

\*/

// 原始语音长度，单位字节

private long len;

// 真实的语音数据，需要进行 base64 编码

private String speech;

// 语种选择，中文=zh、粤语=ct、英文=en，不区分大小写，默认中文

private String lan;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28

这部分代码摘抄自官网示例代码：

private byte[] loadFile(File file) throws IOException {

InputStream is = new FileInputStream(file);

long length = file.length();

byte[] bytes = new byte[(int) length];

int offset = 0;

int numRead = 0;

while (offset < bytes.length && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {

offset += numRead;

}

if (offset < bytes.length) {

is.close();

throw new IOException("Could not completely read file " + file.getName());

}

is.close();

return bytes;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

返回结果：

{"corpus\_no":"6300874524819907792","err\_msg":"success.","err\_no":0,"result":["one day in the cage club got bad news, ","one day in the case club got bad news, ","one day in that case club got bad news, ","one day in the cage club got the bad news, ","one day in the case club got the bad news, "],"sn":"843115237281467036671"}

* 1

格式化以后：

以上介绍了代码如何编写。但是在开发中，我遇到了一个难题，要将音频文件转换成百度语音识别能够识别的格式。请看官方文档说明。

于是，为了测试，我使用格式工厂软件进行格式转化。以下是格式转换的参数。   
  
很高兴的是，经过格式工厂软件转换以后的音频文件能够被百度语音识别 REST 服务识别，识别度还不错，这是令人兴奋的。

但是，我又遇到了另一个问题，在服务器上总不能每个从客户端传来的音频文件都用格式工厂转换吧。于是，我找到了 Linux 平台上一款很好用的软件 sox。使用 sox 命令进行格式转换的命令格式：

sox 原始文件名全路径 -r 16000 -c1 生成的文件名全路径

* 1

接下来，我又继续查找资料，在 Linux 上调用 Linux 平台上的服务可以使用Java 中的 Runtime 和 Process 类运行外部程序。   
参考代码：

String[] cmdStrings = new String[] { "/usr/bin/sox", tempWavFileName, "-r", "16000", "-c1", soundFileName\_16000 };

Process psProcess = Runtime.getRuntime().exec(cmdStrings);

psProcess.waitFor();

* 1
* 2
* 3

到这里，语音识别开发的难点都攻克了。现在总结下来，真的是收获了不少。在这里先做个记录，有些知识点的掌握我还不是很透彻，后续还要再完善一下。

**第三篇**

简单说来，就是无须在开发者的项目中写入代码，或引入 jar 包 。 REST API 就是将音频文件转换成某种特定格式，通过 http 请求发送给百度语音识别的服务器，由百度语音识别的服务器进行语音识别，最后返回识别出的文本。

在我看来，她好在可以很方便的调用，我们自己不用去维护语音识别部分的代码，接入也十分简单，**关键是她是免费的**！

使用的方式简单说来是   
1、根据百度语音识别官方网站提供的   
App ID 和 API Key 获取 accessToken。   
2、根据上一步的 accessToken 连同其它请求参数一起向百度语音识别网关发出请求，获得识别的文本。

是不是觉得和微信公众平台的开发有点像？的确是这样的。微信公众平台的开发的确也是先获取 token，再通过 token 去请求其它数据。

## 集成步骤

本集成步骤参考了《Baidu\_Voice\_REST\_API\_Manual》。建议各位朋友先下载该资料学习。

### 第 1 步：注册成为百度开发者，创建应用，得到 ****API Key**** 和 ****Secret Key****。

这一步非常简单，在官网上也有操作提示，这里就不多介绍了。

### 第 2 步：开通语音识别服务

开通语音识别服务的步骤也非常简单，大家可以自行操作或者参考官方文档。这里要注意：“ 语音识别” 服务初次开通成功后即可获得 50000 次/ 日 的在线调用次数配额。

如果我们每天调用的次数大于 50000 次，可以向百度申请提高次数，据说也是免费的，大赞。

**以下的步骤就很关键了，因为我们要开始写代码了。**

### 第 3 步：获取 Access Token

简而言之，就是向百度 OAuth2.0 授权服务的网关发出请求，将返回的数据（一般是字符串）进行解析，解析出我们想要的 Access Token。下面的图片截取自官网文档，写得非常详细了。

说明：其实就是使用 **API Key** 和 **Secret Key** 和一个固定值的参数向百度 OAuth2.0 授权服务的网关发出 POST 请求，如果请求成功，则解析返回的字符串，从中解析出 access token 待用。

为了方便说明问题，下面代码没有应用良好格式，仅仅只是测试的方法，不建议直接应用于生产环境。

本例使用 HttpClient 框架发送 post 请求，HttpClient 的 Gradle 依赖为：

compile "org.apache.httpcomponents:httpclient:4.5.2"

* 1

示例代码：

/\*\*

\* 获取 token，推荐用 POST 方法

\*/

@Test

public void test01(){

try {

CloseableHttpClient httpClient = HttpClients.createDefault();

HttpPost httpPost = new HttpPost("https://openapi.baidu.com/oauth/2.0/token");

List<NameValuePair> nvps = new ArrayList<>();

nvps.add(new BasicNameValuePair("grant\_type","client\_credentials"));

nvps.add(new BasicNameValuePair("client\_id",apiKey));

nvps.add(new BasicNameValuePair("client\_secret",secretKey));

httpPost.setEntity(new UrlEncodedFormEntity(nvps));

ResponseHandler<String> responseHandler = new ResponseHandler(){

@Override

public String handleResponse(HttpResponse response) throws ClientProtocolException, IOException {

int status = response.getStatusLine().getStatusCode();

if (status >= 200 && status < 300) {

HttpEntity entity = response.getEntity();

try {

return entity != null ? EntityUtils.toString(entity) : null;

} catch (ParseException ex) {

throw new ClientProtocolException(ex);

}

} else {

throw new ClientProtocolException("Unexpected response status: " + status);

}

}

};

String responseBody = httpClient.execute(httpPost,responseHandler);

System.out.println(responseBody);

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40

返回数据：

{"access\_token":"24.463f2a9f7ce6721fe4d15568f812c086.2592000.1469627568.282335-7038695","session\_key":"9mzdDxLM148MA1QmcNNrxGfLYBU9VokNbUY\/8WsJ1r4rUSev1bjP9GTKP6L6SVDnjx4BZxE5ZpJoqztA2K7O0MM9l0Z4","scope":"public audio\_voice\_assistant\_get wise\_adapt lebo\_resource\_base lightservice\_public hetu\_basic lightcms\_map\_poi kaidian\_kaidian","refresh\_token":"25.f77abdb8f638404747dd969615c7b557.315360000.1782395568.282335-7038695","session\_secret":"3efb3872a362beacab28879eed85497b","expires\_in":2592000}

* 1
* 2

格式化以后：

我们须要从中解析出 access\_token ， json 字符串解析的框架有很多 fastjson、Jackson、json-lib、gson，这里就不多做介绍了。

**另外还是要说明一下，因为 access\_token 的有效时间是 2592000 （秒），即 30 × 24 × 60 × 60 （秒）， 30 天，所以没有必要每一次请求都去获取 access\_token ，建议把 access\_token 放在应用的缓存里，如果失效了，再去获取，可以提高应用的效率，这一点和微信公众平台开发是一样的。**

### 第 4 步：根据 Access Token 调用语音识别接口（隐式发送）

说明：我们这里采用的是隐式发送，即不发送真实的音频文件，而是发送音频文件转换而成的字节数组。这里一定要看官网说明，严格调用，才会识别出理想的结果。难点是音频格式的转换。

我是看了官网说明文档和示例代码，经过反复调试才得以调用成功的。鉴于这里篇幅的限制，请大家先看官网说明文档，我这里就不复制了。

这里为了方便说明，先上示例代码，同样地该代码只是为了便于说明问题，不建议在生产环境中直接使用：

/\*\*

\* 识别英文

\*/

@Test

public void test02(){

recognize("voice\_en.wav","en");

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7

上面的测试方法调用了语音识别的方法，该方法传递两个参数，一个是文件的全路径，另一个是中文或者英文的参数。

下面

/\*\*

\* 请求语音识别的时候使用

\*/

private static final String speech\_recognition\_url = "http://vop.baidu.com/server\_api";

* 1
* 2
* 3
* 4

private void recognize(String wavName,String language){

File wavFile = new File(wavName);

HttpPost httpPost = null;

CloseableHttpResponse response = null;

CloseableHttpClient httpClient = HttpClients.createDefault();

httpPost = new HttpPost(speech\_recognition\_url);

SpeechRecognitionRequestEntity requestEntity = new SpeechRecognitionRequestEntity();

// 语音压缩的格式：请按照官网文档填写 pcm（不压缩）、wav、opus、speex、amr、x-flac 之一，不区分大小写

requestEntity.setFormat("wav");

// 声道数，仅支持单声道，请填写 1

requestEntity.setChannel("1");

// 采样率，支持 8000 或者 16000 （这个类型是 int ，不能设置为 String 类型，关于采样率如何转换，请见下文）

requestEntity.setRate(16000);

// todo 这里应判断 AccessToken 是否过期，处理异常，如果过期了，应该重新获取 accessToken

requestEntity.setToken("24.463f2a9f7ce6721fe4d15568f812c086.2592000.1469627568.282335-7038695");

// Cuid 貌似可以随意填写

requestEntity.setCuid("goodluck");

requestEntity.setLen(wavFile.length());

// 官网说： speech 要传递真实的语音数据，需要进行 base64 编码

// 重点关注：请见后面封装的方法，就是把一个文件转换成为指定格式的字节数组

requestEntity.setSpeech(handlerWavFile(wavFile));

// 语种选择，中文=zh、粤语=ct、英文=en，不区分大小写，默认中文

requestEntity.setLan(language);

// 关键点 1 ：将请求参数转换为 json 格式

String requestEntityJson = JSON.toJSONString(requestEntity);

// 关键点 2 ：封装 StringEntity ，为解决中文乱码问题，应该设置编码

StringEntity entity = new StringEntity(requestEntityJson.toString(), "UTF-8");

entity.setContentEncoding("UTF-8");

// 关键点 3 ：设置 StringEntity 的 ContentType

entity.setContentType("application/json");

httpPost.setEntity(entity);

ResponseHandler<String> responseHandler = new ResponseHandler<String>() {

@Override

public String handleResponse(HttpResponse response) throws ClientProtocolException, IOException {

String resData = null;

int statusCode = response.getStatusLine().getStatusCode();

if (statusCode >= 200 && statusCode < 300) {

HttpEntity httpEntity = response.getEntity();

resData = EntityUtils.toString(httpEntity,"utf-8");

EntityUtils.consume(httpEntity);

}

return resData;

}

};

try {

String responseStr = httpClient.execute(httpPost,responseHandler);

System.out.println(responseStr);

} catch (IOException e) {

e.printStackTrace();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52

SpeechRecognitionRequestEntity 类（省略了 get 和 set 方法）：

public class SpeechRecognitionRequestEntity {

// 语音压缩的格式

private String format;

/\*\*

\* 注意，采样率的数据类型一定是 int，不能是 String

\*/

// 采样率，支持 8000 或者 16000，在我们的项目中，写 16000

private int rate;

// 声道数，仅支持单声道，请填写 1

private String channel;

// 开发者身份验证密钥

private String token;

// 用户 ID，推荐使用设备 mac 地址 手机 IMEI 等设备唯一性参数

// todo 貌似可以随意填写，唯一即可

private String cuid;

/\*\*

\* 注意：这里填写的是原始语音的长度，不是使用 base64 编码的语音长度

\*/

// 原始语音长度，单位字节

private long len;

// 真实的语音数据，需要进行 base64 编码

private String speech;

// 语种选择，中文=zh、粤语=ct、英文=en，不区分大小写，默认中文

private String lan;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28

这部分代码摘抄自官网示例代码：

private byte[] loadFile(File file) throws IOException {

InputStream is = new FileInputStream(file);

long length = file.length();

byte[] bytes = new byte[(int) length];

int offset = 0;

int numRead = 0;

while (offset < bytes.length && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {

offset += numRead;

}

if (offset < bytes.length) {

is.close();

throw new IOException("Could not completely read file " + file.getName());

}

is.close();

return bytes;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

返回结果：

{"corpus\_no":"6300874524819907792","err\_msg":"success.","err\_no":0,"result":["one day in the cage club got bad news, ","one day in the case club got bad news, ","one day in that case club got bad news, ","one day in the cage club got the bad news, ","one day in the case club got the bad news, "],"sn":"843115237281467036671"}

* 1

格式化以后：

以上介绍了代码如何编写。但是在开发中，我遇到了一个难题，要将音频文件转换成百度语音识别能够识别的格式。请看官方文档说明。

于是，为了测试，我使用格式工厂软件进行格式转化。以下是格式转换的参数。   
  
很高兴的是，经过格式工厂软件转换以后的音频文件能够被百度语音识别 REST 服务识别，识别度还不错，这是令人兴奋的。

但是，我又遇到了另一个问题，在服务器上总不能每个从客户端传来的音频文件都用格式工厂转换吧。于是，我找到了 Linux 平台上一款很好用的软件 sox。使用 sox 命令进行格式转换的命令格式：

sox 原始文件名全路径 -r 16000 -c1 生成的文件名全路径

* 1

接下来，我又继续查找资料，在 Linux 上调用 Linux 平台上的服务可以使用Java 中的 Runtime 和 Process 类运行外部程序。   
参考代码：

String[] cmdStrings = new String[] { "/usr/bin/sox", tempWavFileName, "-r", "16000", "-c1", soundFileName\_16000 };

Process psProcess = Runtime.getRuntime().exec(cmdStrings);

psProcess.waitFor();

* 1
* 2
* 3

到这里，语音识别开发的难点都攻克了。现在总结下来，真的是收获了不少。在这里先做个记录，有些知识点的掌握我还不是很透彻，后续还要再完善一下。