标贝科技声音复刻项目

Android SDK 使用说明文档（1.0）

Name	Date	Reason For Changes	Version
	2020.03.12	创建文档，编写使用说明。	1.0
	2020.08.05	增加试听功能	1.1

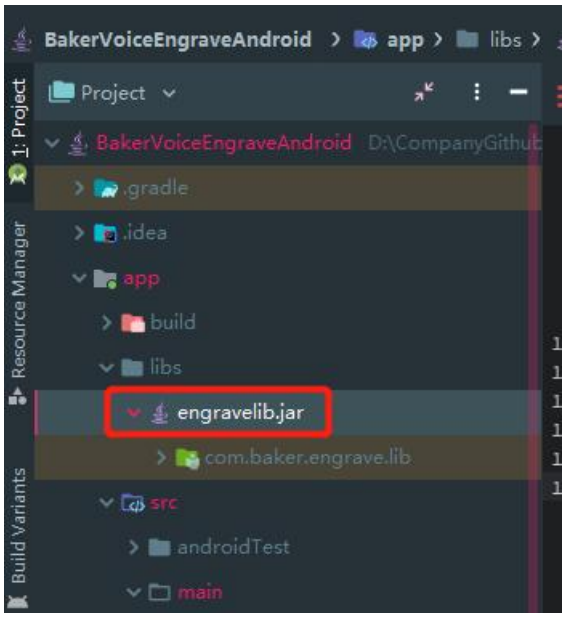
标贝（北京）科技有限公司 DataBaker(Beijing)technology co.,LTD

北京市海淀区西小口路 66 号中关村东升科技园 B-2 号楼 B303 室，010-58465943

目录

1.Android Studio 集成 lib（参考 demo）

1.1 将 jar 包添加至工程主 module 下，lib 文件夹里。同步运行一下 grale 文件，加载该 jar 包。



1.2 在主 module 的 build.gradle 文件里，添加以下代码。

```
dependencies {
    implementation 'com.squareup.okhttp3:okhttp:4.2.2'
    implementation 'com.google.code.gson:gson:2.8.6'
    implementation 'com.kailashdabhi:om-recorder:1.1.5'
}
```

1.3 在主 Module 的 AndroidManifest.xml 文件中添加网络权限。安卓 6.0 及以上系统版本在使用声音复刻 SDK 的时候，必须要申请 RECORD_AUDIO 和 WRITE_EXTERNAL_STORAGE 权限。

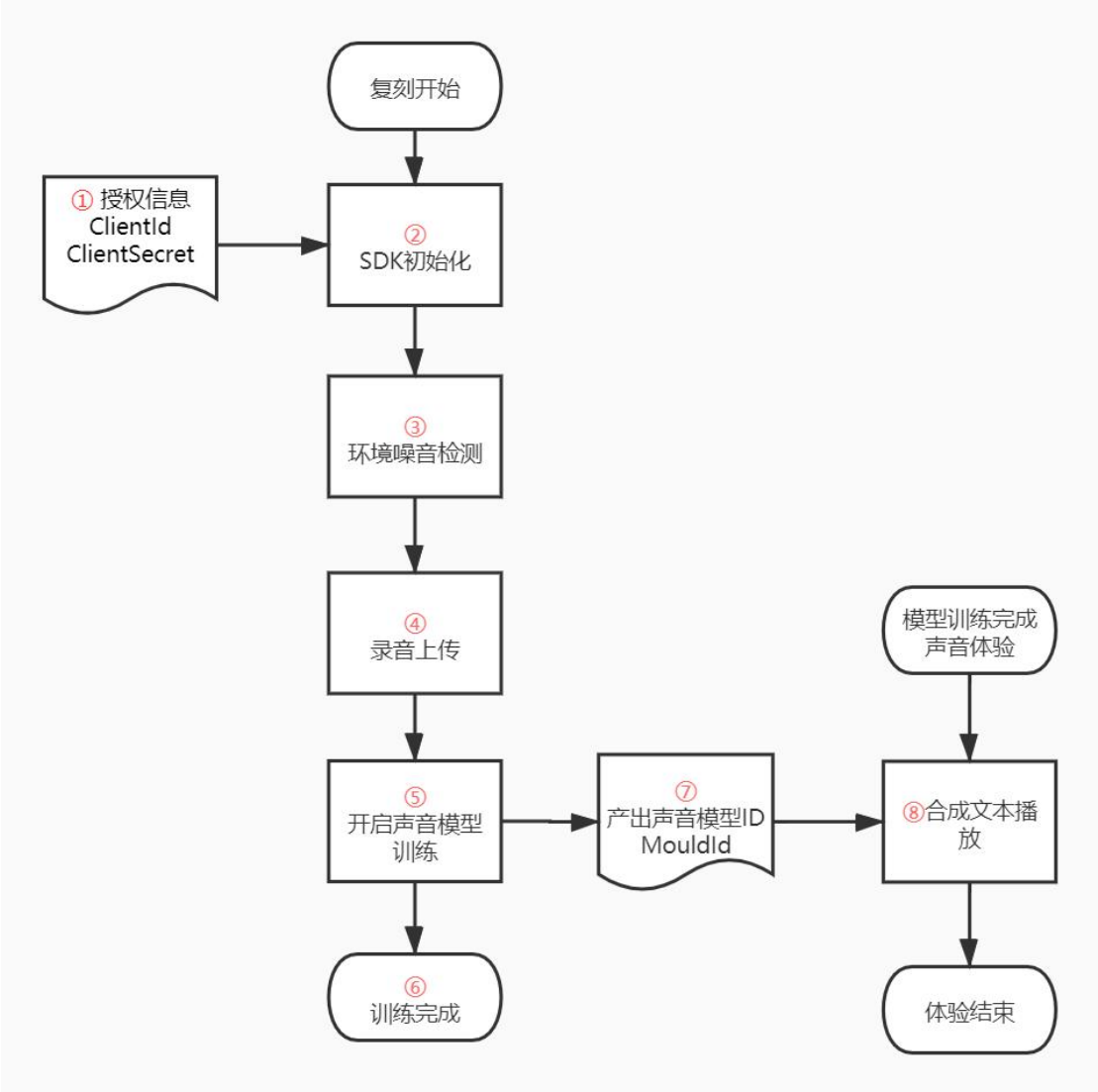
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

1.4 在主 Module 的 AndroidManifest.xml 文件中的 application 节点添加以下属性。

```
android:usesCleartextTraffic="true"
android:requestLegacyExternalStorage="true"
```

Eclipse 环境也遵循相关集成 jar 包的方式即可。

2. 背景及核心流程介绍



2.1 整个复刻体验闭环包含两个模块，一个模块是录音上传服务器进行声音模型训练，产出声音模型 ID，另一个模块是根据声音模型 ID 去合成服务器合成，产出声音文件，播放声音文件完成体验。

2.2 此（复刻）SDK 仅支持第一个模块的功能，即②③④⑤⑥⑦等功能。第二个体验的模块⑧，我们提供 2 种集成方式供选择，以便实现实际项目中的需求。

第一种集成方式如 Demo 中所示，用声音模型 ID + RestAPI 的形式，合成 MP3 声音文件，进行播放。这种合成方式适用于单次不超过 250 字文本长度的文本合成。

另一种方式则是声音模型 ID + TTS 合成 SDK，具体集成方式可参考我们 TTS 合成 SDK 的接入文档。这种方式无文本长度限制，实时合成流式返回，TTS 合成 SDK 中也有播放器功能，集成使用很方便。

3. SDK 关键类

3.1 BakerVoiceEngraver：声音复刻 SDK 关键业务处理类，SDK 中单例存在。可通过 BakerVoiceEngraver.getInstance()获得该实例。
3.2 DetectCallback：环境噪音检测回调类。开启环境检测后，检测结果会实时通过回调方法返回，最终结果以及错误信息都会通过此类的回调方法返回。
3.3 ContentTextCallback：录音文本内容信息获取回调类。文本内容以及错误信息会通过回调方法返回。
3.4 RecordCallback：录音及录音文件上传相关业务回调类。录音状态、文件上传识别、识别结果、错误信息等都会通过回调方法返回。录音过程中会实时将声音分贝值返回。
3.5 UploadRecordsCallback：录音完成后开启声音模型训练的回调类。提交结果、声音模型 ID、错误信息等都会通过回调方法返回。
3.6 MouldCallback：查询声音模型信息回调类。可以通过模型 ID 查询单个模型的训练信息，可以通过 queryId 分页查询该 Id 下所有模型的训练信息。
3.7 Mould：声音模型的实体类。属性如下：

String modelId; //模型 ID
int modelStatus;//模型状态 1=默认状态，2=录制中，3=启动训练失败，4=训练中，5=训练失败，6=训练成功。
String statusName;//模型状态中文(值)

4. 调用说明

4.1 本 SDK 调用过程可能会发生在多个页面，所以我们将各功能实现拆细了，同时伴随着细分的多个回调。这样做的目的是方便灵活调用。

4.2 第一步需要初始化。

可通过调用 BakerVoiceEngraver.getInstance().initSDK(Context context, String clientId, String clientSecret, String queryID)实现初始化。前三个参数是必传参数，clientId 和 clientSecret 是授权信息。第四个参数 queryID 可传空。这个 queryID 的作用是与当前训练的声音模型 ID 关联，存储备份在标贝服务器。以备后期可通过 queryId 查询到与此管理的所有声音模型信息。

如果在初始化时未上传 queryId，也可以调用 BakerVoiceEngraver.getInstance().setQueryId(String queryID) 方法设置 queryID，但设置 queryID 一定要在调用 getVoiceMouldId()方法之前调用，即 4.4 步之前设置。在产出声音模型 ID 时，就需要将 queryId 与声音模型 ID 关联存储。

4.3 环境噪音检测。

在此步之前，请务必申请到 Manifest.permission.RECORD_AUDIO, Manifest.permission.WRITE_EXTERNAL_STORAGE 这 2 个权限。SDK 中也做了权限检测，无权限检测会上报错误信息。

BakerVoiceEngraver.getInstance().setDetectCallback(DetectCallback

callback);设置噪音检测的回调。检测过程中会将实时分贝数据、检测最终结果、错误信息等通过回调接口中的方法返回。DetectCallback 接口的具体信息请参考 xxx。

调用 BakerVoiceEngraver.getInstance().startDBDetection();方法开启噪音检测。检测时长是 3 秒。

噪音检测通过固定算法得出环境声音分贝值作为检测终值，如果终值大于 70 分贝，是不允许进行后续步骤的，因为环境太嘈杂，会直接影响训练出来的声音的品质。可以参考 demo 中对于返回结果的提示以及逻辑处理，如果噪音检测不通过，可以换环境在相对安静的空间内重新检测。重新检测也是调用 BakerVoiceEngraver.getInstance().startDBDetection();方法。

理论上是环境越安静，录音效果越好，训练出来的声音品质会越好。所以对这个噪音的限制，SDK 只做不能超过 70 分贝的最大值限制。使用 SDK 的亲们可以在自己的应用中灵活做二次限制。

4.4 获取录音文本，并录音上传。

在此步也请一定确保申请到 Manifest.permission.RECORD_AUDIO, Manifest.permission.WRITE_EXTERNAL_STORAGE 这 2 个权限。SDK 中也做了权限检测，无权限检测会上报错误信息。

BakerVoiceEngraver.getInstance().setContentTextCallback(ContentTextCallback callback);设置获取录音文本的回调。文本信息或错误信息都将通过回调方法返回。

BakerVoiceEngraver.getInstance().setRecordCallback(RecordCallback callback);设置录音、上传、识别检测、错误信息等接口方法的回调。

BakerVoiceEngraver.getInstance().getTextList();调用此方法获取录音文本信息。参考 demo，将返回的文本信息列表分条展示，供用户完成录音流程。

BakerVoiceEngraver.getInstance().getVoiceMouldId();调用此方法申请此次声音训练的相关资源。在开始录音上传之前，必须调用此方法。

BakerVoiceEngraver.getInstance().startRecord(String contentText);调用此方法开启录音。此方法会返回一个 int 值，该值的意义分别是：0=mouldId 为空，1=无权限，2=开启成功。

BakerVoiceEngraver.getInstance().uploadRecords(String contentText); 本小段朗读完成后，调用此方法停止录音，并开始上传识别。此方法会返回一个 int 值，该值的意义分别是：0=mouldId 为空， 1=结束成功，开始上传识别。

若结束录音后上传过程中出现错误，可以通过调用以下方法 BakerVoiceEngraver.getInstance().reUploadRecords(String contentText);重新上传本段录音。

若因各种原因，在录音过程中需要退出录制，或者放弃当前任务，请调用 BakerVoiceEngraver.getInstance().recordInterrupt();这个方法通知 SDK 中止录音。若在异常退出录制的过程中调用此方法，可以即时释放声音复刻名额以及停止录音。否则未中止录音可能会产生非常大的录音文件，占用用户设备存储资源。

此步中我们将录音的顺序及录音总数的维护交给了 SDK 接入方来维护。理

论上应当一段一段录制，当检测到都录制完成后，就开启声音模型训练了。

4.5 开启声音模型训练。

BakerVoiceEngraver.getInstance().setUploadRecordsCallback(UploadRecordsCallback callback);通过此方法设置开启模型训练结果的回调。开启训练的结果、声音模型 ID、错误信息等都将通过此回调接口方法返回。

BakerVoiceEngraver.getInstance().finishRecords(String phone, String notifyUrl);调用此方法开启模型训练。这个方法中的两个参数都是选填的。第一个参数是手机号，如果上传后，可以通过手机短信收到训练进度通知，但短信中带有标贝公司签名信息，所以不建议在正式项目中上传该手机号。第二个参数是提供一个接收异步训练进度回调的服务器接口地址，建议 SDK 接入方设置该回调地址。标贝科技的异步训练服务器将训练结果以推送方式通知接入方，该地址必须为外网可访问的 url，不能携带参数。（文档最后附了示例回调接口的代码，提供参考）目前主要包括开启训练通知和训练完成时通知。

开启模型训练成功后，此步回调方法中返回的 mouldId 建议 SDK 接入方自行将其存储维护起来，此 mouldId 应当与用户一一对应。在声音合成体验时，要用到这个 mouldId。

4.6 如果需要再次发起新声音模型训练，可以重复第 4.3-4.5 步。

4.7 查询模型信息。

BakerVoiceEngraver.getInstance().setMouldCallback(MouldCallback callback)设置查询结果回调。声音模型相关信息、错误信息等会通过回调方法返回。

BakerVoiceEngraver.getInstance().getMouldList(int page, int limit, String queryId)根据 queryId 分页查询此 queryId 下的所有声音模型信息。Page 从 1 开始。

BakerVoiceEngraver.getInstance().getMouldInfo(String mouldId);根据模型 Id 查询模型信息。

4.8 体验合成声音。

第一种方式类似 Demo 所示，通过 restAPI 形式，拼接 MP3 链接，然后请求播放。这个地方需要 token 信息，这个 token 信息可以通过 BakerVoiceEngraver.getInstance().getToken();实时获取。Token 其实是用 sdk 初始化时提供的 clientId 和 clientSecret 获取的，且有过期时效，所以不建议保存，使用时实时获取即可。相关介绍可以参考标贝公司官网：https://www.data-baker.com/tts_api_rest.html

第二种方式是集成声音合成的 SDK，这种方式接入的相关链接：https://www.data-baker.com/tts_api_androidsdk.html SDK 采用实时合成流式返回，响应速度很快。SDK 也集成了播放器功能，集成使用很方便。

5.各类 API 明细

5.1 BakerVoiceEngraver 类说明

方法名	作用	说明
getInstance()	获取单实例	获取 BakerVoiceEngraver 类单实例。
initSDK()	初始化 SDK	initSDK(Context context, String clientId, String clientSecret, String queryID)初始化 SDK。前三个参数是必传参数，clientId 和 clientSecret 是授权信息。第四个参数 queryID 可传空。这个 queryID 的作用是与当前训练的声音模型 ID 关联，存储备份在标贝服务器。以备后期可通过 queryId 查询到与此管理的所有声音模型信息。
setQueryId()	设置 queryId	如果在初始化时未上传 queryId，也可以调用 setQueryId(String queryID)方法设置 queryID，但设置 queryID 一定要在调用 getVoiceMouldId()方法之前调用，即 4.4 步之前设置。
getTextList()	获取录音文本信息	调用此方法获取录音文本信息。参考 demo，将返回的文本信息列表分条展示，供用户完成录音流程。
startDBDetection()	开启噪音检测	噪音检测通过固定算法得出环境声音分贝值作为检测终值，如果终值大于 70 分贝，是不允许进行后续步骤的，因为环境太嘈杂，会直接影响训练出来的声音的品质。可以参考 demo 中对于返回结果的提示以及逻辑处理，如果噪音检测不通过，可以换环境在相对安静的空间内重新检测。重新检测也是调用此方法。
getVoiceMouldId()	申请声音模型训练资源	调用此方法申请此次声音训练的相关资源。在录音上传之前必须调用此方法。
startRecord()	开始录音	调用 startRecord(String contentText)方法开启录音。参数中 contentText 是此条录音的文本信息（必传）。此方法会返回一个 int 值，该值的意义分别是：0=mouldId 为空，1=无权限，2=开启成功。
uploadRecords()	停止录音上传识别	每小段朗读完成后，调用 uploadRecords(String contentText)方法停止录音，并开始上传识别。参数中 contentText 是此条录音的文本信息（必传）。此方法会返回一个 int 值，该值的意义分别是：0=mouldId 为空，1=结束成功，开始上传识别。

reUploadRecords()	重新上传本段录音	若结束单段录音后上传过程中出现错误，可以通过调用以下方法 reUploadRecords(String contentText);重新上传本段录音。参数中 contentText 是此条录音的文本信息（必传）。此方法会返回一个 int 值，该值的意义分别是：0=mouldId 为空， 1=结束成功，开始上传识别。
recordInterrupt()	异常中断(退出)此次录音任务	若因各种原因，在录音过程中需要退出录制，或者放弃当前任务，请调用 recordInterrupt()方法通知 SDK 中止录音。若在异常退出录制的过程中调用此方法，可以即时释放声音复刻名额以及停止录音。否则未中止录音可能会产生非常大的录音文件，占用用户设备存储资源。
finishRecords()	完成录音开启训练	调用此方法开启模型训练。这个方法中的两个参数都是选填的。第一个参数是手机号，如果上传后，可以通过手机短信收到训练进度通知，但短信中带有标贝公司签名信息，所以不建议在正式项目中上传该手机号。第二个参数是训练进度回调的服务器接口地址，建议 SDK 接入方设置该回调地址，这样我们服务器在模型训练过程中的进度变更时通知该地址，以便及时通知用户。目前主要包括开启训练通知和训练完成时通知。
getMouldList()	获取模型信息列表	getMouldList(int page, int limit, String queryId)根据 queryId 分页查询此 queryId 下的所有声音模型信息。Page 从 1 开始。
getMouldInfo	获取单个模型信息	getMouldInfo(String mouldId);根据模型 Id 查询模型信息。
getToken()	获取 token 信息	getToken()实时获取 Token。Token 其实是用 sdk 初始化时提供的 clientId 和 clientSecret 获取的，且有过期时效，所以不建议保存，使用时实时获取即可。
setContentTextCallback()	设置获取录音文本的回调	setContentTextCallback(ContentTextCallback callback)设置获取录音文本的回调。文本信息或错误信息都将通过回调方法返回。
setDetectCallback()	设置噪音检测的回调	setDetectCallback(DetectCallback callback)设置噪音检测的回调。参数需要接入方自己实例化 DetectCallback，可参考 demo 的方式。检测过程中会将实时分贝数据、检测最终结果、错误信息等通过回调接口中的方法返回。DetectCallback 接口的具体信息请参考 xxx。
setRecordCallback()	设置录音上传相关回调	setRecordCallback(RecordCallback callback)设置录音、上传、识别检测、错误信息等接口方法的回调。
setUploadRecordsCallback()	设置开启模型训练结果的回调。	setUploadRecordsCallback(UploadRecordsCallback callback)通过此方法设置开启模型训练结果的回调。开启训练的结果、声音模型 ID、错误信息等都将通过此回调接口方法返回。
setMouldCallback()	设置查询声音模型信息的回调。	setMouldCallback(MouldCallback callback)设置查询结果回调。声音模型相关信息、错误信息等会通过回调方法返回。
startPlay()	播放录音	第一个参数是试听第几个录音，从 0 开始。

		第二个参数是个回调，详见 5.7。
stopPlay()	停止播放	停止播放已经开始的录音
isRecord(int index)	判断是否录制成功	index 从 0 开始。x 从 0 开始。 如果录制成功返回 true。 如果录制失败返回 false。

5.2 DetectCallback 环境检测回调接口方法说明

接口方法名	作用	说明
dbDetecting	环境检测中结果反馈	dbDetecting(int value);回调方法的参数是实时返回的声音分贝检测值。使用方式参考 demo。
dbDetectionResult	环境检测最终结果反馈	dbDetectionResult(boolean result, int value); 参数 result=true, 表示检测通过。value 是检测通过的分贝值。result=false，表示检测未通过。value 是检测未通过的分贝值。使用方式参考 demo。
onDetectError	错误信息回调	onDetectError(int errorCode, String message);错误信息回调，errorCode 是错误 code 码，message 是详细错误信息。使用方式参考 demo。

5.3 ContentTextCallback 获取录音文本回调接口方法说明

接口方法名	作用	说明
contentTextList	获取录音文本回调	contentTextList(String[] strList);获取录音文本回调，返回的参数是此次录音需要的录音文本, 以字符串数组形式返回。可参考 demo 维护和使用该数据。
onContentTextError	错误信息回调	onContentTextError(int errorCode, String message);错误信息回调，errorCode 是错误 code 码，message 是详细错误信息。使用方式参考 demo。

5.4 RecordCallback 录音上传识别回调接口方法说明

接口方法名	作用	说明
recordsResult	录音中、识别中、识别结果回调。	recordsResult(int typeCode, int recognizeResult);录音中、识别中、识别结果回调。参数 typeCode 1=录音中， 2=识别中，3=最终结果。recognizeResult 是识别率，取值

		0-100. 使用方式参考 demo，非常建议单段录音识别通过后才开启下一段录制。所有段录音完成录制，识别通过才开启模型训练，否则会影响最终声音的品质。
recordVolume	录音过程中实时返回声音分贝信息。	recordVolume(int volume);录音过程中, 会将声音分贝值实时返回。volume 是分贝值。具体使用方式参考 demo。
onRecordError	错误信息回调	onRecordError(int errorCode, String message);错误信息回调，errorCode 是错误 code 码，message 是详细错误信息。使用方式参考 demo。

5.5 UploadRecordsCallback 开启模型训练回调接口方法说明

接口方法名	作用	说明
uploadRecordsResult	模型开启训练成功回调	uploadRecordsResult(boolean result, String mouldId); 参数 result true=成功，false=失败。参数 mouldId 是声音模型的 ID，此 mouldId 建议 SDK 接入方自行将其存储维护起来，此 mouldId 应当与用户一一对应。在声音合成体验时，要用到这个 mouldId。 使用方式参考 demo。
onUploadError	错误信息回调	onUploadError(int errorCode, String message);错误信息回调，errorCode 是错误 code 码，message 是详细错误信息。使用方式参考 demo。提交不成功，可以再次提交。

5.6 MouldCallback 查询声音模型相关回调接口方法说明

接口方法名	作用	说明
mouldInfo	根据 mouldId 查询 mould 信息回调	mouldInfo(Mould mould);根据 mouldId 查询 mould 信息回调，方法的参数是返回的声音模型信息。若需要此回调方法，需要手动复写该方法。使用方式参考 demo。
mouldList	根据 queryId 分页查询 mould 信息回调	mouldList(List<Mould> list); 根据 queryId 分页查询 mould 信息回调。参数 list 是声音模型信息的列表。使用方式参考 demo。
onMouldError	错误信息回调	onMouldError(int errorCode, String message);错误信息回调，errorCode 是错误 code 码，message 是详细错误信息。使用方式参考 demo。

5.7 PlayListener，录音试听回调

接口方法名	作用	说明
playStart	试听开始	当开始播放的时候回调此方法
playEnd	试听结束	当播放结束的时候回调此方法
playError	试听报错	当播放过程中出现错误的时候回调此方法时候回调此方法

6. 失败时返回的 code 对应表

错误码	含义
90000	接口正常，正确返回的识别码
90001	请求 token 失败
90002	Token 过期
90003	参数值为空或不正确
90004	网络请求错误
90005	网络请求返回 data 值为空
90006	服务器返回错误的代码
90007	解析 response 出错
90008	当前上传的录音的 mouldId 与录制的录音的 mouldId 不一致
90009	当前上传的录音的 contentText 与录制的录音的 contentText 不一致
90010	停止录音出错
90011	停止检测噪音出错
90012	创建录音文件时异常
90013	因音频焦点丢失或电话等异常中断录音
99999	服务器系统异常
00011	Token 校验失败，请核查！
10003	参数错误
10004	上传文件失败，请选择正确的文件格式
10005	上传文件不能为空
10008	模型 id 不合法，请重试
10009	模型正在过程录制中，其他客户端不能同时录制！
10010	识别语音超时！
40002	提交次数已达到最大限制
40003	接口请在有效期内使用
40004	接口签名不合法
40005	请填写正确的手机号！

7.异步回调接口方法代码示例

@ApiOperation(value = "模型训练结果回调接口，仅作为示例", notes = "该链接通过参数

```
notifyUrl 设置，如果链接无法访问，将无法接收到回调的 push 信息。")
    @PostMapping("/notify")
    public void mouldInformationNotify(HttpServletRequest request,
HttpServletResponse response) {
        LogUtils.printStartTag(request.getRequestURI());
        String resXml = "";
        InputStream inStream;
        try {
            inStream = request.getInputStream();
            ByteArrayOutputStream outSteam = new ByteArrayOutputStream();
            byte[] buffer = new byte[1024];
            int len;
            while ((len = inStream.read(buffer)) != -1) {
                outSteam.write(buffer, 0, len);
            }

            // 获取 模型训练方 调用 notifyUrl 时携带的返回信息
            String result = new String(outSteam.toByteArray(), "utf-8");
            log.info("dataBaker:模型训练返回结果 ----result---- =" + result);
            // 关闭流
            outSteam.close();
            inStream.close();
            // String 转换为 json 对象，然后业务处理
            JSONObject jsonObject = StringUtils.isNotEmpty(result) ?
JSON.parseObject(result) : null;

            //该参数表示模型 id
            String mouldId = jsonObject == null ? "" : jsonObject.getString("mouldId");
            //该参数表示模型当前状态：可能的值为 0、1、2、3
            String mouldStatus = jsonObject == null ? "" :
jsonObject.getString("mouldStatus");
            //该参数表示状态的说明（文案可能会有变动），0：训练任务已启动 1：训
练任务启动失败 2：训练任务启动成功但训练失败 3：训练成功
            String statusName = jsonObject == null ? "" :
jsonObject.getString("statusName");

            //具体业务处理，例如自行通知用户、存储模型 id 和状态信息等（可先返回结
果，然后异步去完成业务逻辑）

            //todo

            //根据情况，向结果中赋值：正常情况下，选择下面的 resSuccess;如果选择
resFail,将会视为推送失败，标贝服务端将重新推送一次相同的内容
            //      * 返回成功 xml
```

```
                //      */
                //                                          String resSuccess =
"<xml><return_code><![CDATA[SUCCESS]]></return_code><return_msg><![CDATA[O
K]]></return_msg></xml>";
                //      /**
                //       * 返回失败 xml
                //       */
                //                                          String resFail =
"<xml><return_code><![CDATA[FAIL]]></return_code><return_msg><![CDATA[ 报 文 为
空]]></return_msg></xml>";

                resXml = Constant.resSuccess;
                //记录日志
                log.info("dataBaker:模型训练回调返回模型  {}  的状态为：--->{}", mouldId,
statusName);
            } catch (Exception e) {
                log.error("dataBaker:模型训练回调异常：", e);
            } finally {
                try {
                    // 处理业务完毕
                    BufferedOutputStream out = new
BufferedOutputStream(response.getOutputStream());
                    out.write(resXml.getBytes());
                    out.flush();
                    out.close();
                } catch (IOException e) {
                    log.error("dataBaker:模型训练回调异常:out：", e);
                }
            }
        }
```