语音合成

最佳实践

腾讯云

文档目录

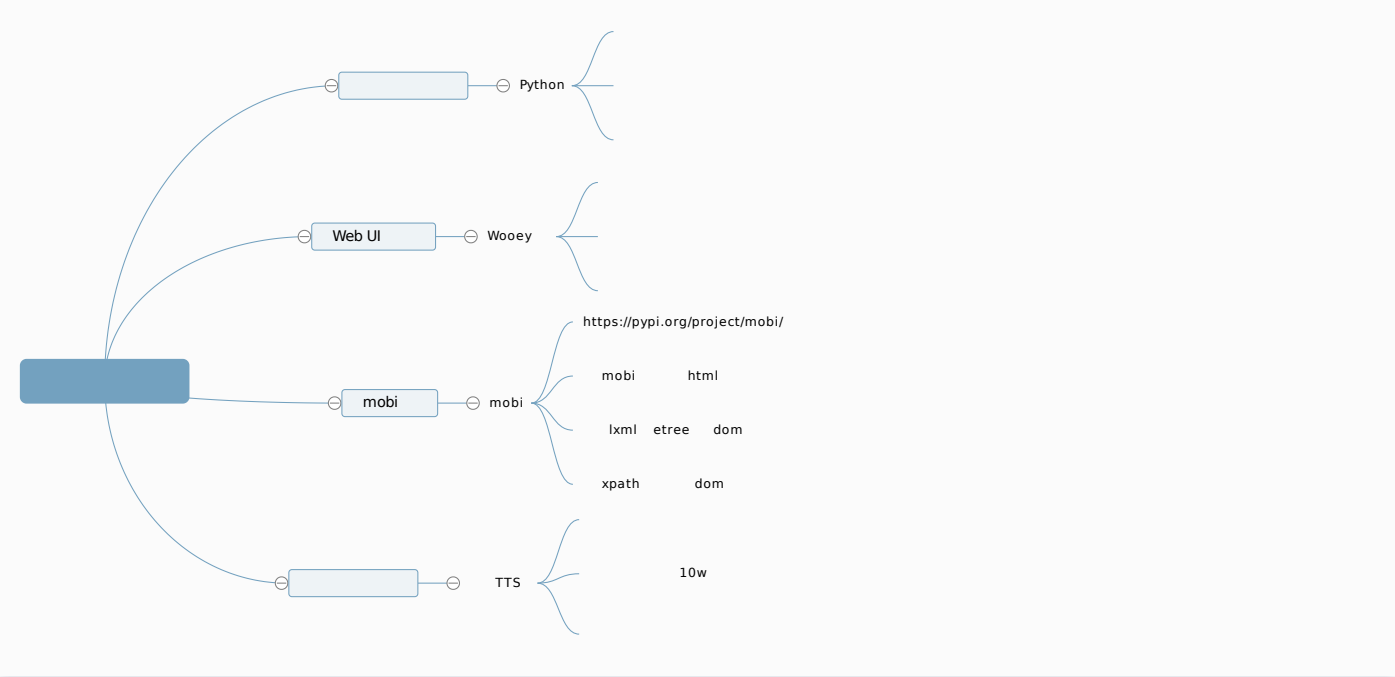
最佳实践
用腾讯云 AI 语音合成打造有声书制作工具

最近更新时间：2023-07-13 17:52:02

腾讯云 AI 语音合成服务已经非常成熟，基于开源工具整合 TTS PaaS 服务，可以非常方便地打造一款个人定制的有声书制作工具，充分利用生活中的碎片时间。
本文档介绍如何通过腾讯云 AI 语音合成技术打造有声书制作工具。

背景分析

有声书需求是指把电子书制作成有声音频，并提供下载链接。
本次实践需要的技术支持：
1. 电子书资源（注意商用时务必确保已获授权）。
2. Web 交互库，上传指定的 mobi 电子书。
3. mobi 解析库，用于获取文本内容。
4. 云计算语音合成 PaaS 服务，基于文本内容，调用语音合成服务，获取有声书音频内容。
5. 提供有声书音频下载。
准备使用工具栈如下：



代码开发

第一步：电子书文件解析

解析模块，先引入外部库 mobi，通过 mobi.extract 函数读取电子书文件，解析为 html 格式的文件 tmp_html。
mobi 库使用可以参见文档 mobi - library for unpacking mobi files 。

```
import mobi
def load_file(self, file_name):
        logging.info('begin to parse file')
        start_t = time.time()
        tmp_dir, tmp_html = mobi.extract(file_name) # 解析 mobi 文件
        end_t = time.time()
        logging.info('extract {} to {}. cost {}ms'.format(file_name, tmp_html, int((end_t-start_t)*1000)))

        with open(tmp_html, 'r') as fp:
```

```
        lines = fp.readlines()
    self.html_content = ''.join(lines) # 读取 html
    logging.info('load file total {} chars'.format(len(self.html_content)))

    shutil.rmtree(tmp_dir)
    logging.info('clean temp dir {}'.format(tmp_dir))
```

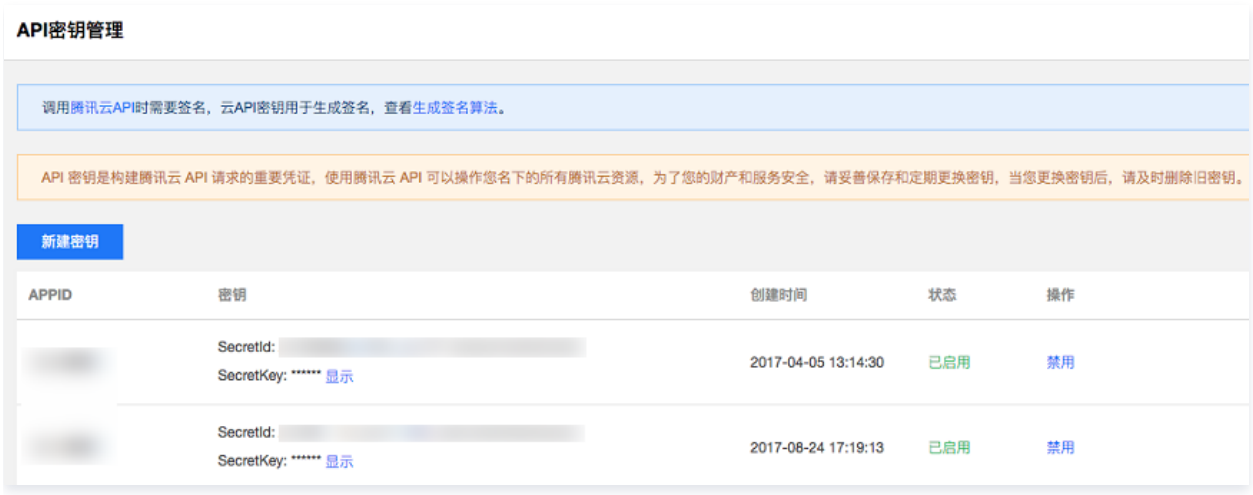
得到 html 文件后，通过 lxml.etree 将其解析为一棵 DOM 树，然后可以通过 xpath，获得其中的任意内容。例如特定属性的元素、特定位置的段落、标题等，您可参考 XPath 教程 。

```
from lxml import etree
def parse_html(self):
    logging.info('parse html')
    # pre process
    self.html_content = self.pre_process(self.html_content)

    # parse dom
    dom = etree.fromstring(self.html_content)
    plist = dom.xpath('//p/text()')
    audio_texts = []

    # 示例，比如从 1010 段开始，获取后面 10 个段落
    idx_start = 1010
    for p in plist[idx_start:idx_start+10]:
        #logging.info('{}'.format(p))
        audio_texts.append(p)

    self.text = ''.join(audio_texts)
    logging.info('content length {}'.format(len(self.text)))
```

以上为电子书解析模块，封装在 AudioBookGenerator 类，详情请参见 src/audio_book_generator.py 。

第二步：有声语音合成

有声语音合成需要基于腾讯云语音合成 TTS 服务。语音合成服务的注册与开通等操作请参见 快速入门 。
服务开通后，登录访问管理控制台，在 API 密钥管理 页面获取密钥，配置到 config 文件中即可。



配置文件 src/config.py 。

```
class Config(object):
    SECRET_ID = 'XXXX' # 对应上面的 SecretId
    SECRET_KEY = 'XXXX' # 对应上面的 SecretKey
```

下面介绍如何使用官网提供的 SDK ，调用语音合成服务。具体参见 长文本语音合成 SDK，这里我们用 Python SDK，集成 SDK 到本示例的工程中。
长文本合成是个异步服务，提供两个接口用于服务调用：

- 创建合成任务接口：CreateTtsTask。
- 查询任务状态及结果接口：DescribeTtsTaskStatus。

下面分别针对 **create_task** 和 **query_task** 这两个函数进行了封装。

> ⚠ 注意
> 查询任务状态时，任务可能并未执行完成，所以需要间隔一段时间后循环查询，直到任务完成（成功或失败）。

1. 创建任务：CreateTtsTask
   调用时，需注意两个参数：
   - VoiceType：音色 id，用于选择不同的发音人，这里使用的是智逍遥（100510000），适用于武侠或玄幻小说的场景。
   - VoiceoverDialogueSplit：旁对白支持选项，需要设置为 True，可以将文本中的对话和旁白分割，并分别用对应的音色进行合成
   请求成功后，返回该任务的唯一 ID：TaskId。

```
def create_task(self) -> str:
  task_id = ''

  req = models.CreateTtsTaskRequest()
  req.Text = self.text # 合成文本
  req.VoiceType = self.voice_type # 设置音色id，此处选用 智逍遥100510000
  req.VoiceoverDialogueSplit = self.voiceover_dialogue_split # 打开旁对白支持
  req.Codec = self.codec
  req.SampleRate = self.sample_rate
  req.ModelType = self.model_type
  try:
      resp = self.client.CreateTtsTask(req)
      task_id = resp.Data.TaskId
      req_id = resp.RequestId
      print('call CreateTtsTask succeed, task_id: {} request_id: {}'.format(task_id, req_id))
  except TencentCloudSDKException as err:
      print('call CreateTtsTask failed, err: {}'.format(str(err)))

  return task_id
```

2. 查询任务状态及结果：DescribeTtsTaskStatus
   调用时，将上面得到的 TaskId 作为参数传进去，请求会实时返回任务的相关信息，主要包含：
   - Status：任务状态。
   - ErrorMsg：任务错误信息（任务失败时）。
   - ResultUrl：合成音频地址。

```
def query_task(self, task_id):
  req = models.DescribeTtsTaskStatusRequest()
  req.TaskId = task_id
  try:
      resp = self.client.DescribeTtsTaskStatus(req)
      data = resp.Data
      req_id = resp.RequestId
      print('call DescribeTtsTaskStatus succeed, data: {} request_id: {}'.format(str(data), req_id))
  except TencentCloudSDKException as err:
      print('call DescribeTtsTaskStatus failed, err: {}'.format(str(err)))

  if data:
      return data.Status, data.ErrorMsg, data.ResultUrl # 任务状态、错误信息、音频文件地址
  else:
      return 3, 'internal error', ''
```

以上是有声书语音合成模块，封装在 TencentSDK 类，详情请参见 src/tencent_sdk.py 。

第三步：完成有声书制作脚本

通过 main 脚本，将以上两步的电子书解析模块、语音合成模块集成到一起，再增加文件下载功能，即可完成有声书制作脚本。
腾讯云 TTS 服务返回的合成音频 url，新增 HttpAgent 类，将音频二进制文件下载到本地。

```
from audio_book_generator import AudioBookGenerator
from http_agent import HttpAgent

def main():
    file_name = sys.argv[1]
    logging.info('upload file: {}'.format(file_name))

    # gen audio
    generator = AudioBookGenerator()
    generator.process(file_name)
    audio_url = generator.get_audio_url()
    logging.info('get audo url: {}'.format(audio_url))

    # download audio
    session_path = os.environ.get('SESSION_PATH', './')
    audio_name = os.path.join(session_path, 'result.mp3')
    agent = HttpAgent()
    agent.download(audio_url, audio_name)
    logging.info('download audio to: {}'.format(audio_name))
```

HttpAgent 详情请参见文件 src/http_agent.py 。
本地工具已完成，可以通过下列命令调用查看效果：

```
(venv) justin@VM_centos:[~/audio_book/src]: python main.py ../dou.mobi
2022-06-21 10:36:44,959 - main.py[line:13] - INFO: upload file: ../dou.mobi
2022-06-21 10:36:44,959 - /home/justin/audio_book/src/audio_book_generator.py[line:26] - INFO: begin to parse file
2022-06-21 10:36:47,253 - /home/justin/audio_book/src/audio_book_generator.py[line:30] - INFO: extract ../dou.mobi to
/tmp/mobiexk287bwzw/mobi7/book.html. cost 2294ms
2022-06-21 10:36:47,293 - /home/justin/audio_book/src/audio_book_generator.py[line:35] - INFO: load file total 4988080 chars
2022-06-21 10:36:47,295 - /home/justin/audio_book/src/audio_book_generator.py[line:38] - INFO: clean temp dir
/tmp/mobiexk287bwzw
2022-06-21 10:36:47,295 - /home/justin/audio_book/src/audio_book_generator.py[line:45] - INFO: parse html
2022-06-21 10:36:47,506 - /home/justin/audio_book/src/audio_book_generator.py[line:60] - INFO: content length 625
2022-06-21 10:36:47,549 - /home/justin/audio_book/venv/lib64/python3.6/site-packages/urllib3/connectionpool.py[line:1005] -
DEBUG: Starting new HTTPS connection (1): tts.tencentcloudapi.com:443
2022-06-21 10:36:47,699 - /home/justin/audio_book/venv/lib64/python3.6/site-packages/urllib3/connectionpool.py[line:465] -
DEBUG: https://tts.tencentcloudapi.com:443 "POST / HTTP/1.1" 200 125
2022-06-21 10:36:47,701 - /home/justin/audio_book/venv/lib64/python3.6/site-
packages/tencentcloud/common/http/request.py[line:112] - DEBUG: GetResponse Status: 200
Header: Server: nginx
Date: Tue, 21 Jun 2022 02:36:41 GMT
Content-Type: application/json
Content-Length: 125
Connection: keep-alive
Data: {"Response":{"RequestId":"ffb6f632-bd56-427d-ae21-xxxx","Data":{"TaskId":"gz-27ac44ab-c21e-4e58-b0b3-xxxx"}}}

call CreateTtsTask succeed, task_id: gz-27ac44ab-c21e-4e58-b0b3-xxxx request_id: ffb6f632-bd56-427d-ae21-xxxx

2022-06-21 10:37:27,964 - /home/justin/audio_book/venv/lib64/python3.6/site-packages/urllib3/connectionpool.py[line:1005] -
DEBUG: Starting new HTTPS connection (1): tts.tencentcloudapi.com:443
2022-06-21 10:37:28,016 - /home/justin/audio_book/venv/lib64/python3.6/site-packages/urllib3/connectionpool.py[line:465] -
DEBUG: https://tts.tencentcloudapi.com:443 "POST / HTTP/1.1" 200 576
2022-06-21 10:37:28,017 - /home/justin/audio_book/venv/lib64/python3.6/site-
packages/tencentcloud/common/http/request.py[line:112] - DEBUG: GetResponse Status: 200
Header: Server: nginx
Date: Tue, 21 Jun 2022 02:37:21 GMT
Content-Type: application/json
Content-Length: 576
Connection: keep-alive
```

```
Data: {"Response":{"RequestId":"7c4c20d3-ad79-47ea-86a8-xxxx","Data":{"TaskId":"gz-27ac44ab-c21e-4e58-b0b3-xxxx","Status":2,"StatusStr":"success","ResultUrl":"https://xxxx","ErrorMsg":""}}}

call DescribeTtsTaskStatus succeed, data: {"TaskId": "gz-27ac44ab-c21e-4e58-b0b3-xxxx", "Status": 2, "StatusStr": "success", "ResultUrl": "https://xxxx", "ErrorMsg": ""} request_id: 7c4c20d3-ad79-47ea-86a8-xxxx
2022-06-21 10:37:28,580 - /home/justin/audio_book/venv/lib64/python3.6/site-packages/urllib3/connectionpool.py[line:465] - DEBUG: https://xxxx:443 "GET /xxxx HTTP/1.1" 200 535248
http download succ: https://xxxx -> ./result.mp3
2022-06-21 10:37:29,001 - main.py[line:26] - INFO: download audio to: ./result.mp3
```

可以正常生成音频文件 result.mp3。附录中有一个 demo 音频，供您试听，体验效果。

第四步：脚本可视化

有声书制作脚本已完成，但脚本用起来还是不方便，且无法给他人使用。此时需要对脚本进行可视化，将其部署为一个 Web 工具。
这里采用 Wooey 开源库，有如下优点：

- 通过编译一个适配类，将脚本工具非常方便地转化为 Web 交互页面。
- 支持常见UI交互组件，如下拉框、文件上传等，通过代码配置的方式展示到页面上，无需任何前端知识。
- 支持任务启动、回显执行过程，结果文件下载等功能。

适配类如下，通过 parser 增加了文件上传组件：

```
import os
import sys
import argparse

parser = argparse.ArgumentParser(description="convert mobi file to audio")
parser.add_argument('--audio', help='the mobi file to make audio', type=argparse.FileType('r'), required=True) # 文件上传组件

def audio_book(mobi_file):
    _format = mobi_file.split('.')[-1].lower()
    if _format != 'mobi':
        print('only mobi is supported')
        return
            # TODO: 此处填写业务逻辑

if __name__ == '__main__':

    args = parser.parse_args()
    audio_book(args.audio.name)
```

调用电子书制作脚本工具，通过 Python venv 方式，隔离 wooey 与 工具脚本的环境变量，方便 wooey 平台集成其他任意脚本。

```
SCRIPT_PATH = '/root/audio_book'

def audio_book(mobi_file):
    # ...
    # TODO: 此处填写业务逻辑
    cmd = []
    cmd.append('export SESSION_PATH={}'.format(os.getcwd())) # 传输本次执行 session 路径到脚本
    cmd.append('cd {}'.format(SCRIPT_PATH))
    cmd.append('source {}/venv/bin/activate'.format(SCRIPT_PATH))
    cmd.append('cd src')
    cmd.append('python main.py {}'.format(mobi_file))
    cmd.append('cd ')
    cmd = '&&'.join(cmd)

    print(cmd)
    os.system(cmd)
```

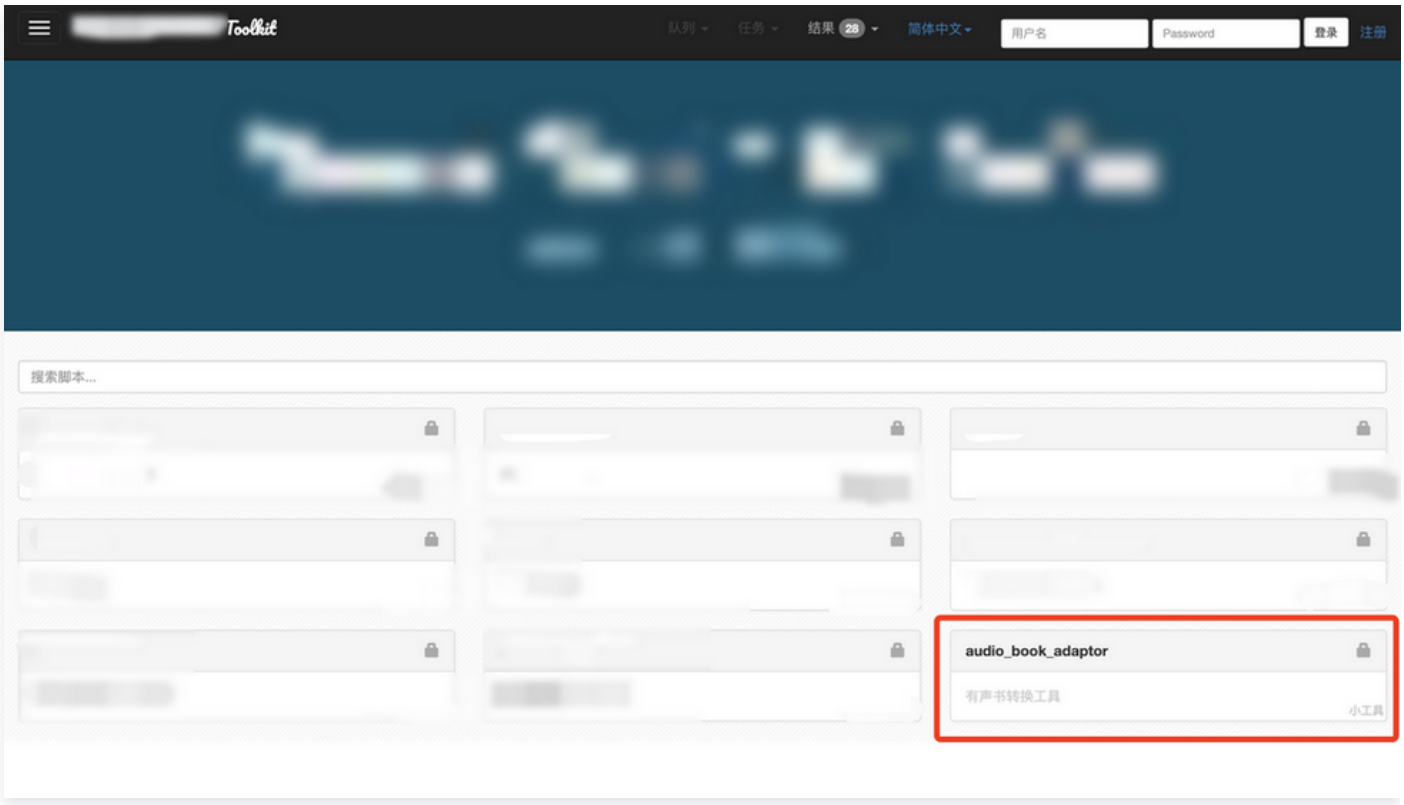
添加脚本到可视化平台：

```
[root@VM-centos ~/TOOLS]# python manage.py addscript ../audio_book/audio_book_adaptor.py --group 小工具
Converting ../audio_book/audio_book_adaptor.py
Converted 0 scripts
```

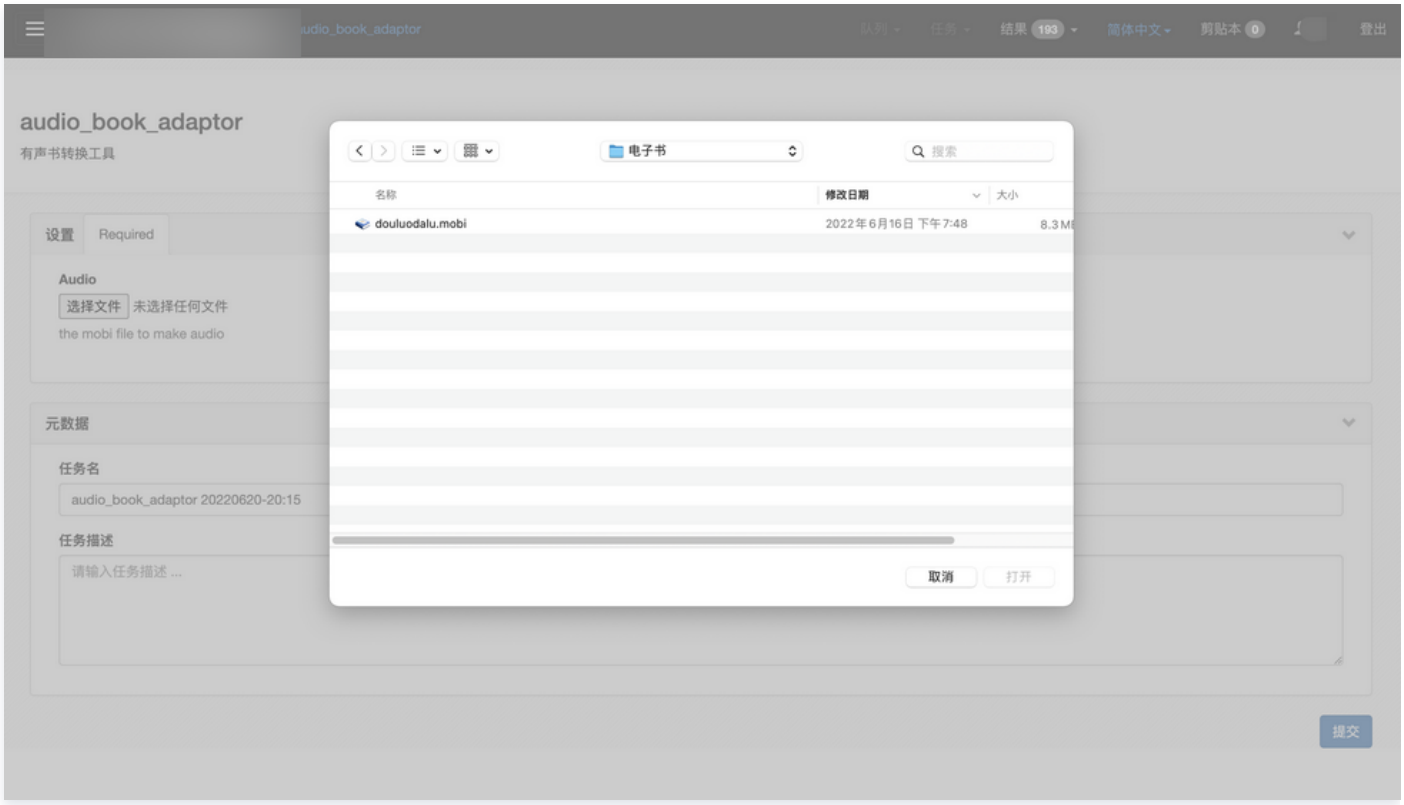
产品体验

工具完成后，可通过以下步骤合成自己的第一本有声书，体验产品效果。

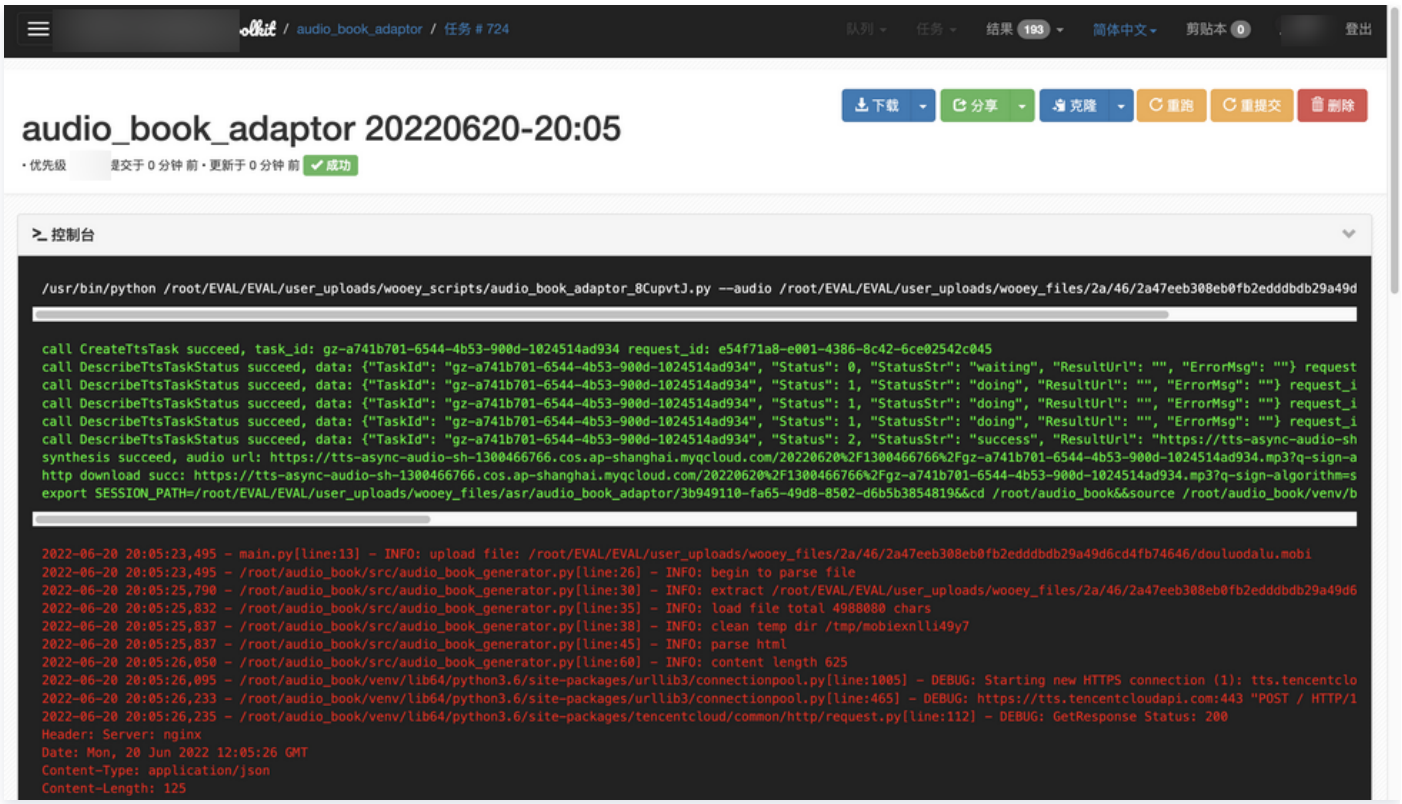
1. 打开工具平台，选择有声书制作工具。



2. 单击**选择文件**按钮，上传需要转换的电子书文件。



3. 启动任务，从页面可以看到脚本执行日志。



4. 任务执行结束后，状态显示成功，可以从页面底部的文件列表中，单击 result.mp3 进行下载。

有声书制作工具已完成，试听音频以及工程代码请参见附录。

附录

- 有声书声音效果试听：result.mp3
- 有声书制作工程代码：audio_book_generator

结合 AI 语音合成和云开发快速上线一款实用工具小程序

最近更新时间：2023-12-07 16:01:23

> **说明：**
> 本文来自 AI 专题用户实践征文，仅供学习和参考。

名词介绍

语音合成（Text To Speech，TTS）满足将文本转化成拟人化语音的需求，打通人机交互闭环。 提供多场景、多语言的音色选择，支持 SSML 标记语言，支持自定义音量、语速等参数，让发音更专业、更符合场景需求。语音合成广泛适用于智能客服、有声阅读、新闻播报、人机交互等业务场景，提升人机交互体验，提高语音类应用构建效率。

云开发（Tencent CloudBase，TCB）是腾讯云提供的云原生一体化开发环境和工具平台，为开发者提供高可用、自动弹性扩缩的后端云服务，包含计算、存储、托管等 serverless 化能力，可用于云端一体化开发多终端应用（小程序、公众号、Web 应用、Flutter 客户端等），帮助开发者统一构建和管理后端服务和云资源，避免了应用开发过程中繁琐的服务器搭建及运维，开发者可以专注于业务逻辑的实现，开发门槛更低，效率更高。

开发背景

对于媒体从业人员来说，一款简单、易用的文本转语音软件是非常必要的，并且要随时随地无需下载和注册都能使用，所以结合小程序和云开发是最合适的。

开发工具

云开发 CloudBase

代码开发

第一步：用户输入框

```
// WXML
<textarea class="" type="text" value="{{txt}}" bindinput="setTxt" auto-height="true"></textarea>
// JS
data: {
      txt:""
    }
setTxt(e){
   this.setData({txt:e.detail.value})
},
```

第二步：用户提交按钮

```
// WXML
<view class="" bindtap="submit"></view>
//js
submit(){
   wx.cloud.callFunction({
      name:"submitTxt",
      data:{
         txt:this.data.txt
      }
   }).then(res=>{

   })
}
```

第三步：新建云函数

新建云函数，名字为上一步 submit() 中调用的 submitTxt。

```
const tencentcloud = require("tencentcloud-sdk-nodejs");

const TtsClient = tencentcloud.tts.v20190823.Client;

// 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey,此处还需注意密钥对的保密
// 密钥可前往https://console.cloud.tencent.com/cam/capi网站进行获取
// 云开发中的云函数相当于服务端，可以明文填写您的密钥
const clientConfig = {
  credential: {
    secretId: "XXXXXXXXXXXXXXXXXXX",
    secretKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  },
  region: "", // 地域，可以为空
  profile: {
    httpProfile: {
      endpoint: "tts.tencentcloudapi.com",
    },
  },
};
exports.main = async (event, context) => {
  var txt = event.txt
  const client = new TtsClient(clientConfig);
  // 此处仅提供必填参数，更多参数选择请参考官方文档。
  const params = {
    "Text": txt,
    "ModelType": 1,// 模型类型，1-默认模型。
    "VoiceType": 10510000// 10510000-智逍遥，旁对白阅读风格男声
  };
  client.CreateTtsTask(params).then(
    (data) => {
      console.log(data);
      //   {
      //     RequestId: 'dc7708bc-1b8c-412e-9033.....',
      //     Data: { TaskId: 'gz-2e20190c-8f20-4941-ab7e-.....' }
      //   }
      var TaskId = data.Data.TaskId;
      return TaskId
    },
    (err) => {
      console.error("error", err);
      return "FAIL"
    }
  );
}
```

因为长文本的语音合成是需要一定时间的，所以回调是一个任务 ID，需要再根据这个 ID 查询任务状态，如果已完成，会有返回一个 URL。开发者可以根据需要，考虑是否把提交和查询合并在一起。这里采用的是分离的模式。

> **说明：**
> 此处依赖可以不用本地安装，直接右键选择云端安装依赖即可。在云函数文件中 package.json 加入。

```
"dependencies": {
  "tencentcloud-sdk-nodejs": "^4.0.348",// 加入此项
  "wx-server-sdk": "~2.6.1"
}
```

第四步：查询任务状态

新建一个与上一步的环境相同的云函数。

```
const client = new TtsClient(clientConfig);
const params = {
    "TaskId": TaskId // 上一步得到的任务ID
};
client.DescribeTtsTaskStatus(params).then(
    (data) => {
        console.log(data);
    //    {
    //        RequestId: 'effa415f-94b1-46c1-a5fa-.....',
    //        Data: {
    //            TaskId: 'gz-2e20190c-8f20-4941-ab7e.....',
    //            Status: 2, //任务状态码，0：任务等待，1：任务执行中，2：任务成功，3：任务失败。
    //            StatusStr: 'success',
    //            ResultUrl: 'https://tts-async-audio-.....',
    //            ErrorMsg: ''
    //        }
    //    }
        if(data.Data.Status === 2) return data.Data.ResultUrl
        return data.Data.StatusStr
    },
    (err) => {
        console.error("error", err);
        return "FAIL"
    }
);
```

前端得到第三步返回的任务 ID 后，可提供该 ID 继续调用第四步的云函数，获取任务结果。
如果成功，即可返回一段 MP3 的播放地址。

第五步：前端加入播放组件

```
  <video
    id="myVideo"
    src={{ResultUrl}}
  ></video>
```

第六步：前端下载功能

```
wx.downloadFile({
  url: url, //之前获取的地址
  success (res) {
    // 只要服务器有响应数据，就会把响应内容写入文件并进入 success 回调，业务需要自行判断是否下载到了想要的内容
    if (res.statusCode === 200) {
      // TODO
    }
  }
})
```

至此，一款简单的语音合成小程序，已经完成了核心功能。您可以根据需要，美化前端 UI，并增加用户管理功能，结合云开发的数据库可以很好的管理用户的数据。也可以增加用户使用的逻辑，例如新用户可以直接使用 X 次，通过激励用户观看视频或者分享，可以获取使用次数等。