



Datawhale AI夏令营 Task3

Task3：星火 API & Gradio 学习指南



如何从一段代码行对话，变成我们日常使用的**APP、小程序、网页**等工具？

如果我们想快速添加常用的Prompt模板、或是想要拥有一个历史会话功能，应该怎么办？

此时，我们就需要通过代码级开发来自定义前端了，前面大家已经在Task1快速入门了代码级开发，复刻了自己的应用demo，相信下面的内容对你来说也绝对**不在话下**——

配套魔搭示例代码项目可见：https://modelscope.cn/studios/Datawhale/datawhale_spark_2024

零、预备：了解基础概念



关于开发类的内容，可能会涉及到很多新概念，

为了尽量压缩篇幅，且让大家不用有那么多认知负担，

我们仅会对Gradio框架以及相关术语进行简单介绍，

主要目标是引导大家进入实操，且开始掌握如何解决问题的思路，

大家如果对相关信息有更多的学习需求，可以通过官方文档、浏览器搜索等方式研习。

Gradio 官方文档详见：<https://www.gradio.app/>

要想拥有更符合心意的应用，需要自己进行开发，即：我们需要把星火大模型的 API **作为一个功能组件**，加入到我们自己的应用中去，做出属于我们创意的内容~

创建应用有很多多种方式，常见如：**网页、浏览器插件、小程序、移动应用程序(App)、桌面应用软件**等。

一般来说，小程序、App、企业微信、订阅号服务等还有一定的**开发及审核门槛**，且需要学习较多的前端甚至UI设计相关的知识，需要具有一定的专业度和人力投入。

因此我们在快速验证功能和设计demo时，往往较优的选择是直接通过 **Gradio/Streamlit 快捷开发可交互的demo**。

使用 **飞桨AI Studio**、**魔搭Notebook** 等云开发平台，可以减少很多本地环境配置的麻烦。
dwspark、Gradio、也只需要简单通过pip安装对应的Python库即可。

在学习之前，这里还有几个基础概念你需要知道：

- **Python**：一种简洁、易读、功能强大的高级编程语言，常言道“人生苦短，我用Python”。**Gradio** 是基于Python开发并开源的库。
- **pip**：Python的包管理工具，主要用于安装、管理和卸载Python的第三方库。它是Python开发中不可或缺的工具之一，常用命令：
 - `pip install xxx`、`pip install --upgrade xxx`、`pip uninstall xxx`、`pip show xxx`、`pip install xxx==1.1.1`、`pip list`
安装指定python库、升级指定python库、卸载指定python库、查看指定Python库的安装情况、安装指定Python库的指定版本、列出所有通过pip安装的Python库
 - 在使用gradio、streamlit、以及appbuilder-SDK时，记得安装这些库哟~
- **SDK**：即“软件开发工具包”（Software Development Kit），一般会包含一系列工具和文档的集合，旨在帮助开发者更有效地开发特定的软件应用程序。工具可能包括编程语言的库、APIs、开发环境（IDE）、编译器、调试&性能分析工具、相关文档等。
 - **dwspark** 就是我们为了方便大家调用星火大模型API而封装开发的SDK
- **飞桨 AI Studio**：基于百度深度学习平台飞桨（PaddlePaddle）的人工智能学习与实训社区。它提供了一站式的模型在线开发与应用环境，支持多种编程语言和深度学习框架。
 - 我们可以在AI Studio上进行python代码编写与运行，且可以免费使用其CPU环境（GPU环境需要付费）
 - 另外我们的 Gradio 项目完成开发后，还可以通过其进行部署，分享给他人体验

- **魔搭ModelScope**：

💡 如果有本地开发的需求，欢迎自己探索环境配置，带着实操经验和问题在学习群进行提问和交流

一、dwspark 进阶指南

1. 为什么会有 dwspark？

考虑到官方提供很多模型能力，但各自有自己的SDK且调用起来较为麻烦，所以我们对基础的模型进行了代码封装，以方便大家以较为简洁的方式调用这些模型能力。

具体的模型官方接口文档见下表：

模型	文档地址
认知大模型	https://www.xfyun.cn/doc/spark/Web.html
文本合成语音	https://www.xfyun.cn/doc/tts/online_tts/API.html
语音识别文本	https://www.xfyun.cn/doc/asr/voicedictation/API.html
文生图	https://www.xfyun.cn/doc/spark/ImageGeneration.html
图片理解	https://www.xfyun.cn/doc/spark/ImageUnderstanding.html
文本向量化	https://www.xfyun.cn/doc/spark/Embedding_api.html

2. 如何使用dwspark？

2.1 安装SDK

pypi包地址：<https://pypi.org/project/dwspark>

```
1 pip install dwspark
```

2.2 加载配置

```
1 from dwspark.config import Config
2 # 加载系统环境变量: SPARKAI_APP_ID、SPARKAI_API_KEY、SPARKAI_API_SECRET
3 config = Config()
4 # 自定义key写入
5 config = Config('14****93', 'eb28b****b82', 'MWM1MzBkOD****Mzk0')
```

2.3 调用模型

```
1 # SDK引入模型
2 from dwspark.models import ChatModel, Text2Img, ImageUnderstanding, Text2Audio, Audio2Text,
   EmbeddingModel
3 # 讯飞消息对象
4 from sparkai.core.messages import ChatMessage
5 # 日志
6 from loguru import logger
7 '''
8 对话
9 '''
10 # 模拟问题
11 question = '你好呀'
12 logger.info('-----批式调用对话-----')
13 model = ChatModel(config, stream=False)
14 logger.info(model.generate([ChatMessage(role="user", content=question)]))
15 logger.info('-----流式调用对话-----')
16 model = ChatModel(config, stream=True)
17 [ logger.info(r) for r in model.generate_stream(question)]
18 logger.info('done.')
19 '''
20 文字生成语音
21 '''
22 text = '2023年5月，讯飞星火大模型正式发布，迅速成为千万用户获取知识、学习知识的“超级助手”，成为解放生产力、释放想象力的“超级杠杆”。2024年4月，讯飞星火V3.5春季升级长文本、长图文、长语音三大能力。一年时间内，讯飞星火从1.0到3.5，每一次迭代都是里程碑式飞跃。'
23 audio_path = './demo.mp3'
24 t2a = Text2Audio(config)
25 # 对生成上锁，预防公有变量出现事务问题，但会降低程序并发性能。
26 t2a.gen_audio(text, audio_path)
27 '''
28 语音识别文字
29 '''
30 a2t = Audio2Text(config)
31 # 对生成上锁，预防公有变量出现事务问题，但会降低程序并发性能。
32 audio_text = a2t.gen_text(audio_path)
```

```
33 logger.info(audio_text)
34 '''
35 生成图片
36 '''
37 logger.info('-----生成图片-----')
38 prompt = '一只鲸鱼在快乐游泳的卡通头像'
39 t2i = Text2Img(config)
40 t2i.gen_image(prompt, './demo.jpg')
41 '''
42 图片解释
43 '''
44 logger.info('-----图片解释-----')
45 prompt = '请理解一下图片'
46 iu = ImageUnderstanding(config)
47 logger.info(iu.understanding(prompt, './demo.jpg'))
48 '''
49 获取文本向量
50 '''
51 logger.info('-----获取文本向量-----')
52 em = EmbeddingModel(config)
53 vector = em.get_embedding("我们是datawhale")
54 logger.info(vector)
```


模型列表

模型名称	调用方式
批式调用 对话	<div>1 model = ChatModel(config, stream=False) 2 model.generate([ChatMessage(role="user", content='对话内容'])])</div>
流式调用 对话	<div>1 model = ChatModel(config, stream=True) 2 model = ChatModel(config, stream=True) 3 [logger.info(r) for r in model.generate_stream('对话内容')]</div>
文字生成 语音	<div>1 t2a = Text2Audio(config) 2 model = ChatModel(config, stream=True) 3 t2a.gen_audio('你好啊', '生成音频地址')</div>
生成图片	<div>1 t2i = Text2Img(config) 2 t2i.gen_image('生成图片需求', '图片地址')</div>
图片解释	<div>1 iu = ImageUnderstanding(config) 2 iu.understanding('图片理解方向描述', '图片地址')</div>

获取文本
向量

```
1 em = EmebdddingModel(config)
2 vector = em.get_embedding("需要向量化的文本")
```

3. dwspark 代码详解

 dwspark主要包含以下几个文件：

- **config.py**：统一的配置文件
- **utils**：一些工具类
- **models**：存放模型的地方
 - **models.ChatModel.py**：对话相关模型
 - **models.AudioModel.py**：音频相关模型
 - **models.ImageModel.py**：图片相关模型
 - **models.EmbeddingModel.py**：向量化模型

下面是其中两个代码的源码：

- **config.py**

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # @File : config.py
4 # @Author: Richard Chiming Xu
5 # @Date : 2024/6/24
6 # @Desc :
7
8 import os
9
10
11 class Config():
12     def __init__(self, appid: str = None, apikey: str = None, apisecret: str = None):
13         '''
14         讯飞API统一的环境配置
15         :param appid: appid
16         :param apikey: api key
17         :param apisecret: api secret
18
19         调用方式：
20         # 加载系统环境变量SPARKAI_APP_ID、SPARKAI_API_KEY、SPARKAI_API_SECRET
21         # 系统环境变量需要进行赋值
22         config = Config()
23         # 自定义key写入
24         config = Config('14***93', 'eb28b***b82', 'MWM1MzBkOD***Mzk0')
25         '''
26         if appid is None:
27             self.XF_APPID = os.environ["SPARKAI_APP_ID"]
28         else:
29             self.XF_APPID = appid
30         if apikey is None:
31             self.XF_APIKEY = os.environ["SPARKAI_API_KEY"]
32         else:
33             self.XF_APIKEY = apikey
34         if apisecret is None:
```

```

35         self.XF_APISECRET = os.environ["SPARKAI_API_SECRET"]
36     else:
37         self.XF_APISECRET = apisecret

```

- **ChatModel.py**

```

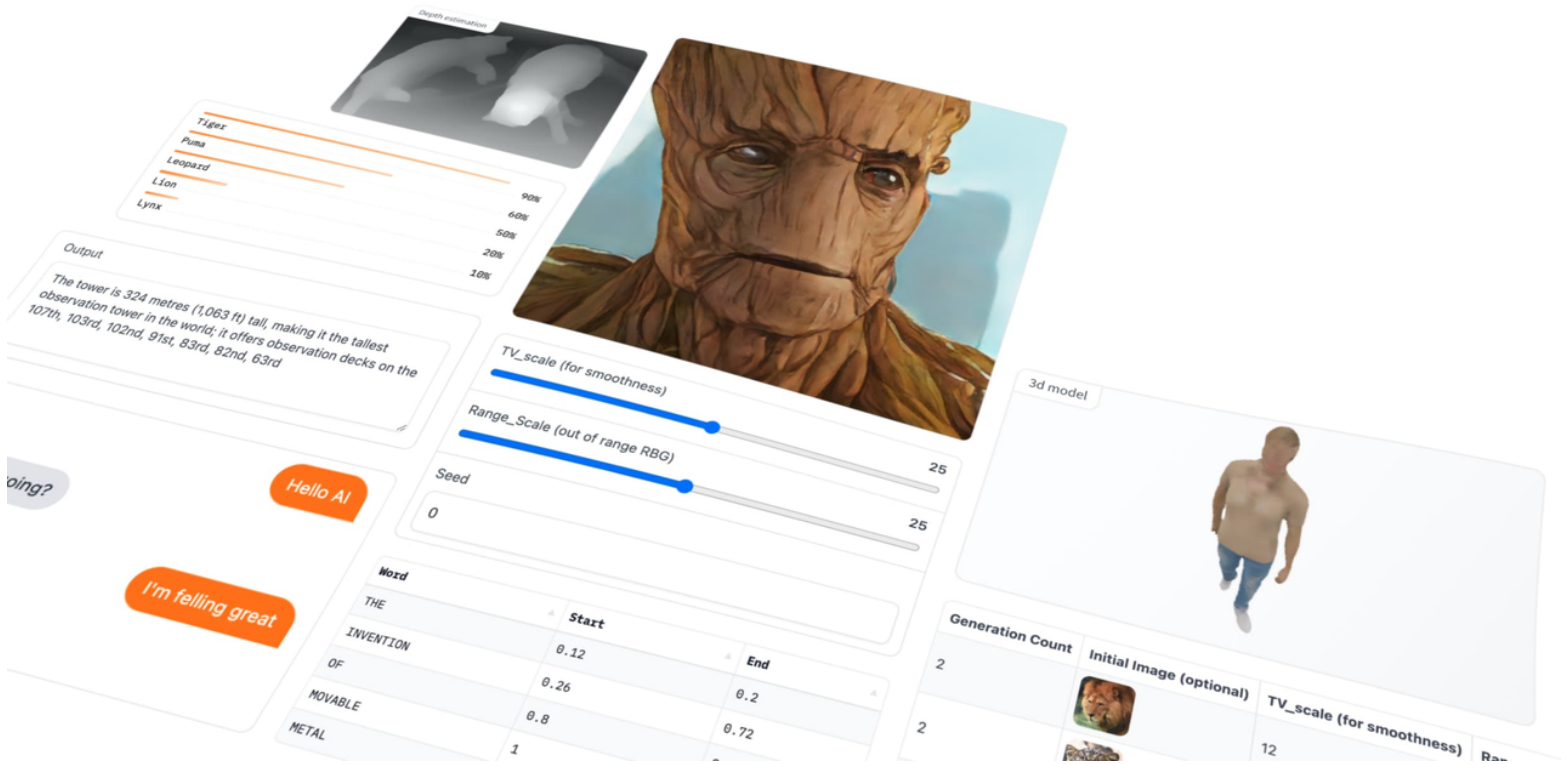
1  from typing import List, Iterable
2
3  from sparkai.llm.llm import ChatSparkLLM
4  from sparkai.core.messages import ChatMessage
5  from dwspark.config import Config
6  from loguru import logger
7
8
9  class ChatModel():
10     def __init__(self, config: Config, domain: str = 'generalv3.5', model_url: str = 'wss://spark-
        api.xf-yun.com/v3.5/chat', stream:bool=False):
11         '''
12         初始化模型
13         :param config: 项目配置文件
14         :param domain: 调用模型
15         :param llm_url: 模型地址
16         :param stream: 是否启用流式调用
17         '''
18         self.spark = ChatSparkLLM(
19             spark_api_url=model_url,
20             spark_app_id=config.XF_APPID,
21             spark_api_key=config.XF_APIKEY,
22             spark_api_secret=config.XF_APISECRET,
23             spark_llm_domain=domain,
24             streaming=stream,
25         )
26         self.stream = stream
27
28     def generate(self, msgs: str | List[ChatMessage]) -> str:
29         '''
30         批式调用
31         :param msgs: 发送消息，接收字符串或列表形式的消息
32         :return:
33         '''
34         if self.stream is True:
35             raise Exception('模型初始化为流式输出，请调用generate_stream方法')
36
37         messages = self.__trans_msgs(msgs)
38         resp = self.spark.generate([messages])
39         return resp.generations[0][0].text
40
41     def generate_stream(self, msgs: str | List[ChatMessage]) -> Iterable[str]:
42         '''
43         流式调用
44         :param msgs: 发送消息，接收字符串或列表形式的消息
45         :return:
46         '''
47         if self.stream is False:
48             raise Exception('模型初始化为流式输出，请调用generate方法')
49         messages = self.__trans_msgs(msgs)
50         resp_iterable = self.spark.stream(messages)
51         for resp in resp_iterable:
52

```



```
53         yield resp.content
54
55     def __trans_msgs(self, msg: str):
56         '''
57         内部方法，将字符串转换为消息
58         :param msgs: 字符串
59         :return:
60         '''
61         if isinstance(msg, str):
62             messages = [ChatMessage(role="user", content=msg)]
63         else:
64             messages = msg
65         return messages
```

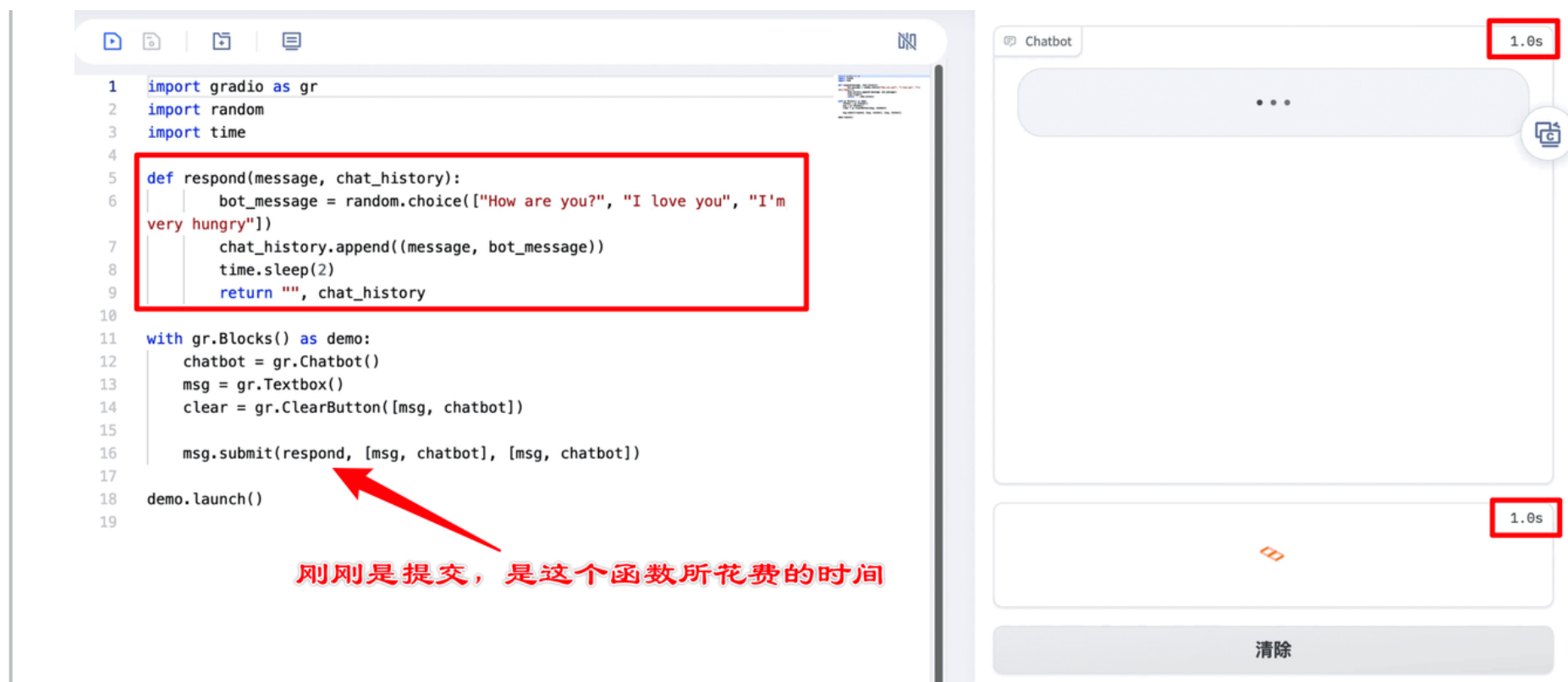
二、Gradio基础概念入门



Gradio 可以包装几乎任何 Python 函数为易于使用的用户界面，在 飞桨AI Studio 上可直接渲染效果，如下图所示：



- 还可以查看运行相关函数从输入到输出所花费的时间



要基于Gradio开发应用，必须了解 Gradio 有**输入输出组件、控制组件、布局组件**几个基础模块，其中

- **输入输出组件**用于展示内容和获取内容，如：Textbox 文本、Image 图像
- **布局组件**用于更好地规划组件的布局，如：Column（把组件放成一列）、Row（把组件放成一行）
 - 推荐使用 `gradio.Blocks()` 做更多丰富交互的界面，`gradio.Interface()` 只支持单个函数交互
- **控制组件**用于直接调用函数，无法作为输入输出使用，如：Button（按钮）、ClearButton（清除按钮）

Gradio的设计哲学是将输入和输出组件与布局组件分开。输入组件（如Textbox、Slider等）用于接收用户输入，输出组件（如Label、Image等）用于显示函数的输出结果。而布局组件（如Tabs、Columns、Row等）则用于组织和排列这些输入和输出组件，以创建结构化的用户界面。

大部分**输入输出组件**都有以下三个参数：

- `fn`：绑定的函数，输入参数需与 `inputs` 列表类型对应
- `inputs`：输入组件变量名列表，（例如：`[msg, chatbot]`）
- `outputs`：输出组件变量名列表，（例如：`[msg, chatbot]`）
- 另外不同的 **输入输出组件、控制组件** 有不同动作可响应（对应一些方法，如下面的 `msg.submit()`）

💡 **需要注意，Gradio的组件更新，实现交互，都只能通过绑定的 `fn` 进行实现：**

- `fn` 中需要使用的用户指定的数据来源，均需要放入 `inputs` 中，
- 需要更新的组件均需要通过 `fn return` 回来，并与 `outputs` 中的组件类型意义对应

对应到 **【AI Studio】速通星火API开发** 中的“体验随机回复应用”的示例代码中即：

```
1 # 导入gradio、random、time库，他们的功能大致如名字所示
2 import gradio as gr # 通过as指定gradio库的别名为gr
3 import random
4 import time
5
6 # 自定义函数，功能是随机选返回指定语句，并与用户输入的 chat_query 一起组织为聊天记录的模式返回
7 def chat(chat_query, chat_history):
8     # 在How are you 等语句里随机挑一个返回，放到 bot_message 变量里
9     bot_message = random.choice(["How are you?", "I love you", "I'm very hungry"])
```



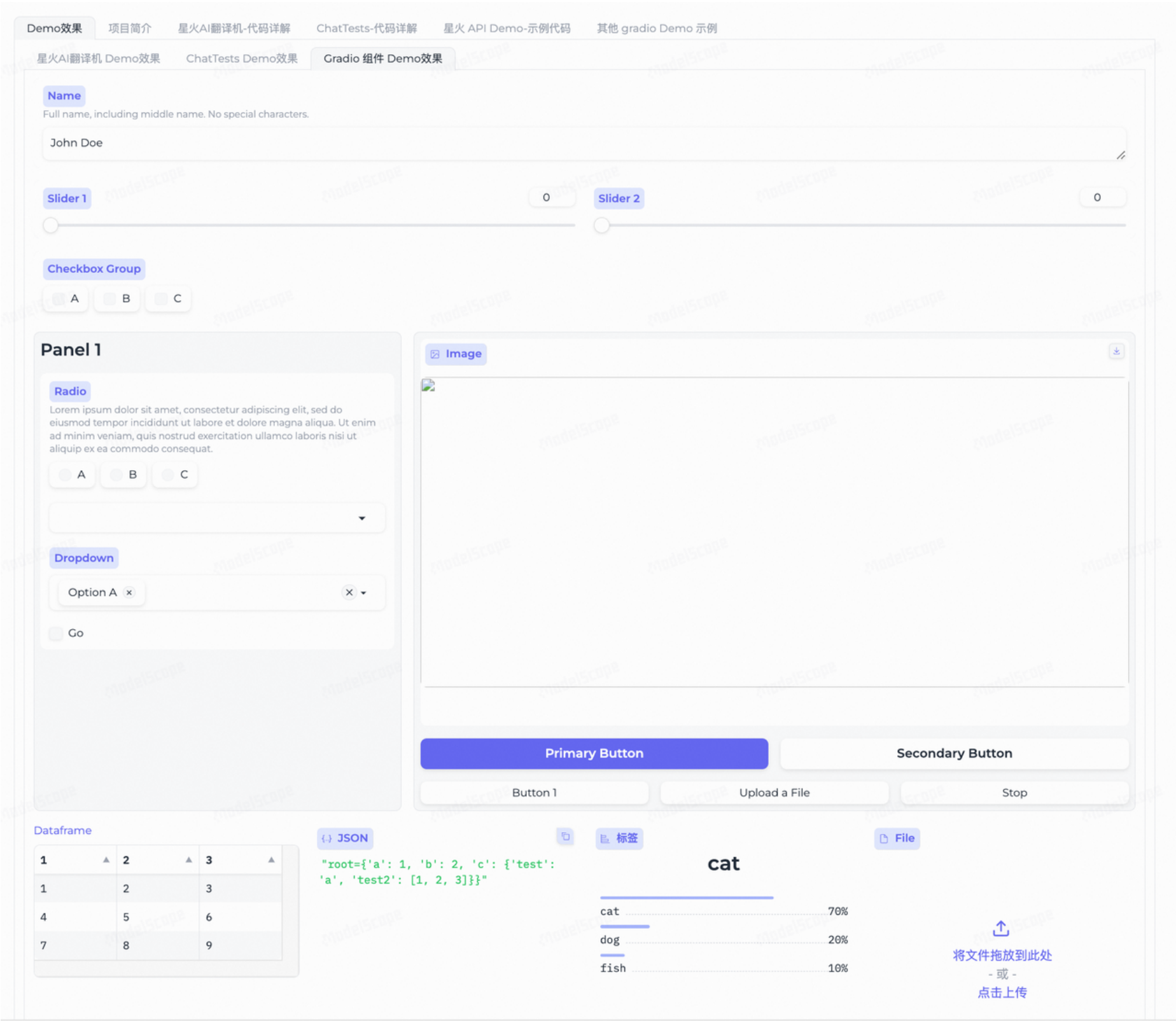
```

10         # 添加到 chat_history 变量里
11         chat_history.append((chat_query, bot_message))
12         # 返回 空字符, chat_history 变量, 空字符用于清空 chat_query 组件, chat_history 用于更新 chatbot组件
13         return "", chat_history
14
15 # gr.Blocks(): 布局组件, 创建并给了他一个名字叫 demo
16 with gr.Blocks() as demo:
17     # gr.Chatbot(): 输入输出组件, 用于展示对话效果
18     chatbot = gr.Chatbot([], elem_id="chat-box", label="聊天历史")
19     # gr.Textbox(): 输入输出组件, 用于展示文字
20     chat_query = gr.Textbox(label="输入问题", placeholder="输入需要咨询的问题")
21     # gr.Button: 控制组件, 用于点击, 可绑定不同的函数触发处理
22     llm_submit_tab = gr.Button("发送", visible=True)
23
24     # gr.Examples(): 输入输出组件, 用于展示组件的样例, 点击即可将内容输入给 chat_query 组件
25     gr.Examples(["请介绍一下Datawhale。", "如何在大模型应用比赛中突围并获奖?", "请介绍一下基于Gradio的应用开发"], chat_query)
26
27     # 定义gr.Textbox()文字组件 chat_query 的 submit 动作(回车提交)效果, 执行函数为 chat, 第一个[chat_query, chatbot]是输入, 第二个 [chat_query, chatbot] 是输出
28     chat_query.submit(fn=chat, inputs=[chat_query, chatbot], outputs=[chat_query, chatbot])
29     # 定义gr.Button()控制组件 llm_submit_tab 的 点击动作 效果, 执行函数为 chat, 第一个[chat_query, chatbot]是输入, 第二个 [chat_query, chatbot] 是输出, 效果与上一行代码同
30     llm_submit_tab.click(fn=chat, inputs=[chat_query, chatbot], outputs=[chat_query, chatbot])
31
32 # 运行demo
33 if __name__ == '__main__':
34     demo.queue().launch()

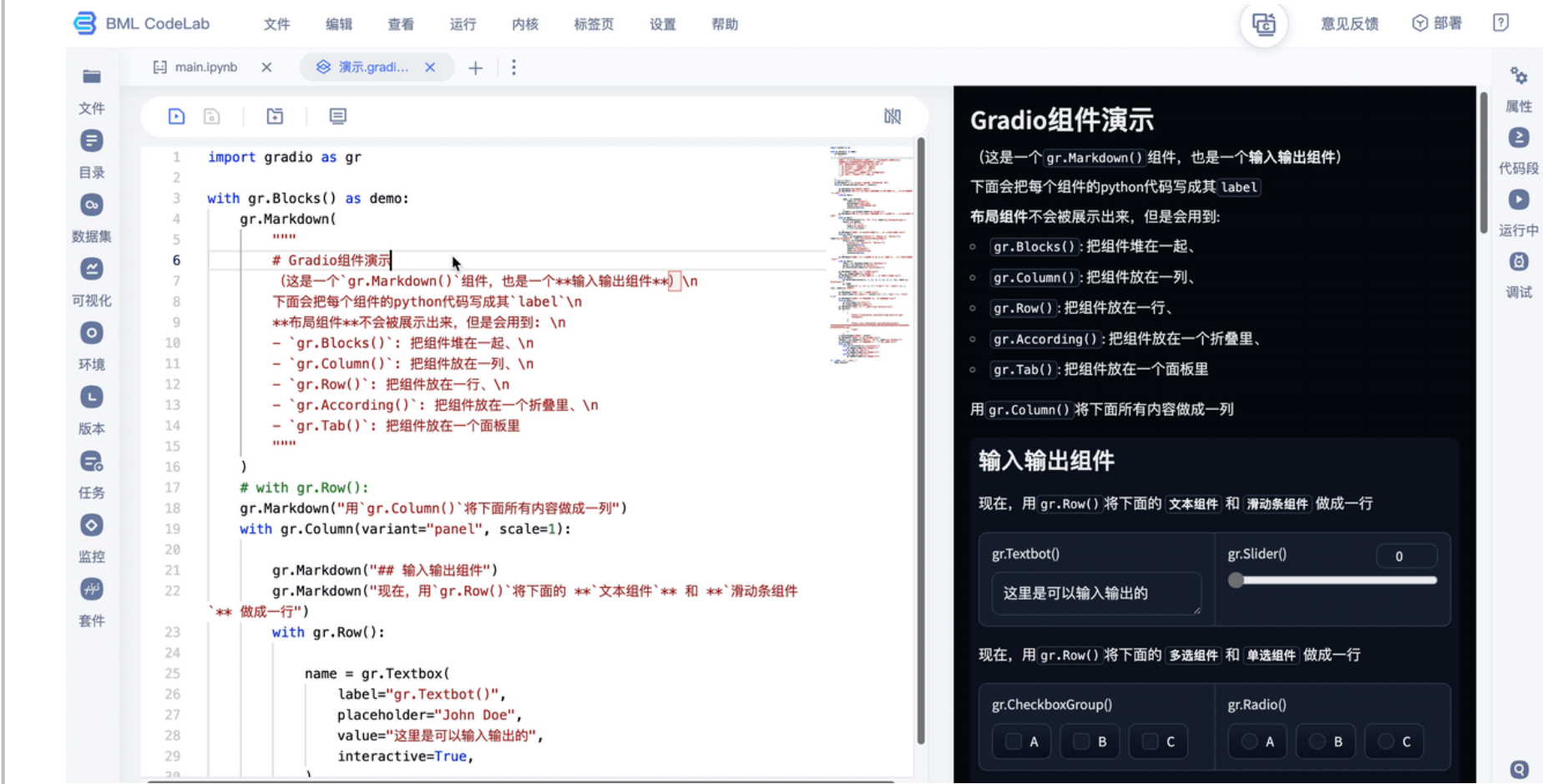
```

三、更多 Demo 效果体验 &学习

更多的组件效果查看, 可见 [【魔搭】零基础开始大模型应用开发-配套项目](#)（并不是全部），全量组件相关信息可在[官方文档](#)中查看



- 但需要注意，在 飞桨AI Studio 上完成了代码修改后，需要再次点击左上角的【运行】，他不会自动更新



- 当文件重命名，或者调整了文件位置后，需要重新打开代码运行

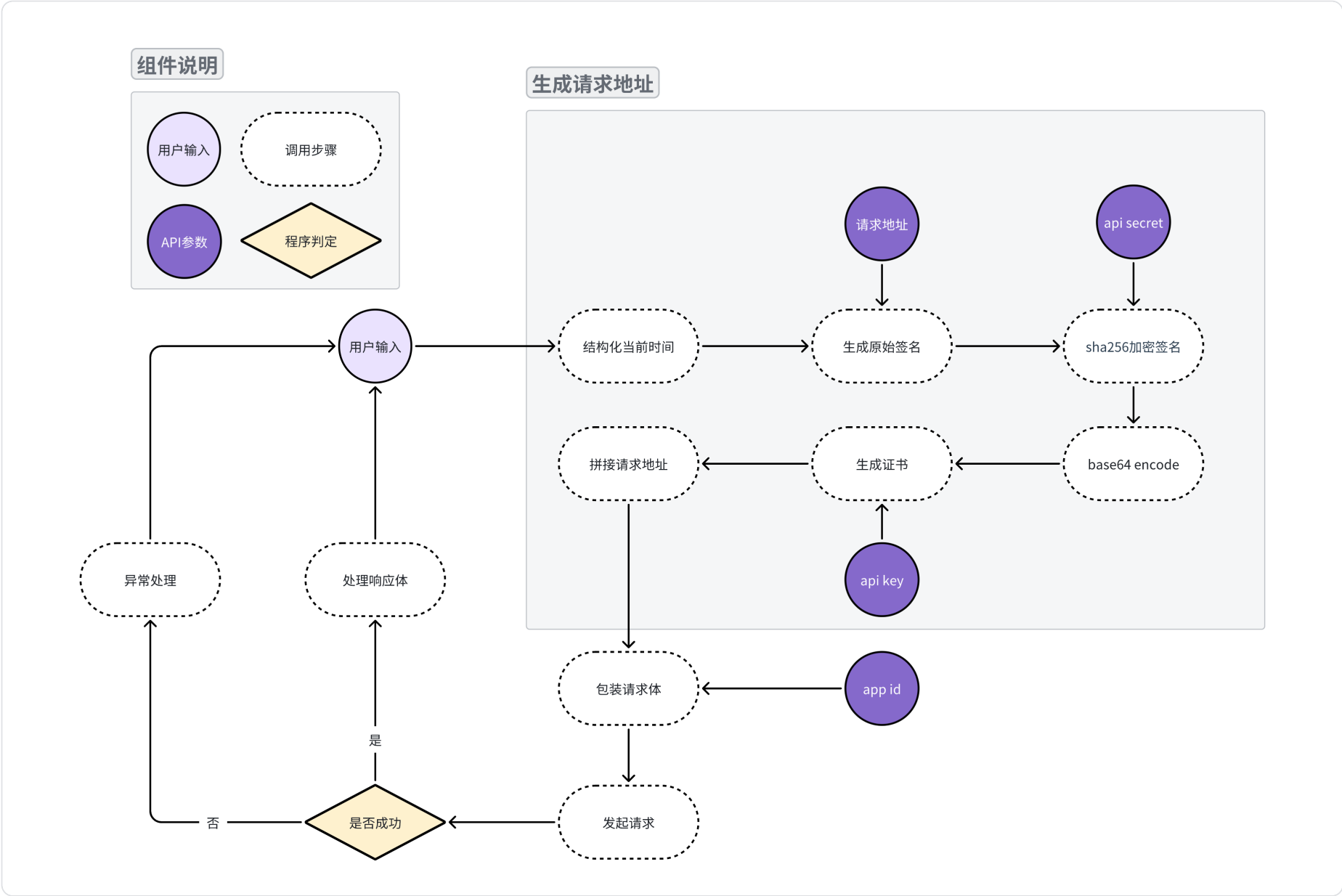
如果想了解更多组件详情，可查看 [官方文档](#)；另外，如果想设计更复杂的界面风格，还可以查看学习 [官方文档：主题](#)

(较难，选学)四、星火大模型 API进阶指南

💡 报名后，官方提供多款星火大模型API的token。其中包括：**星火大模型**、**语音合成**、**语音识别**、**图片生成**、**图片理解**、**文本向量化**以及**智能PPT生成**。这些模型API让我们能够把更多的注意力放在idea实现和应用开发当中。

下面将介绍大模型的API调用原理及学习使用的思路

API流程（图像生成为例）



API调用流程

如何查看API文档？（图像生成为例）

根据模型需求，进入[API接口文档](#)。根据下列步骤查看API关键信息（其他API类似）

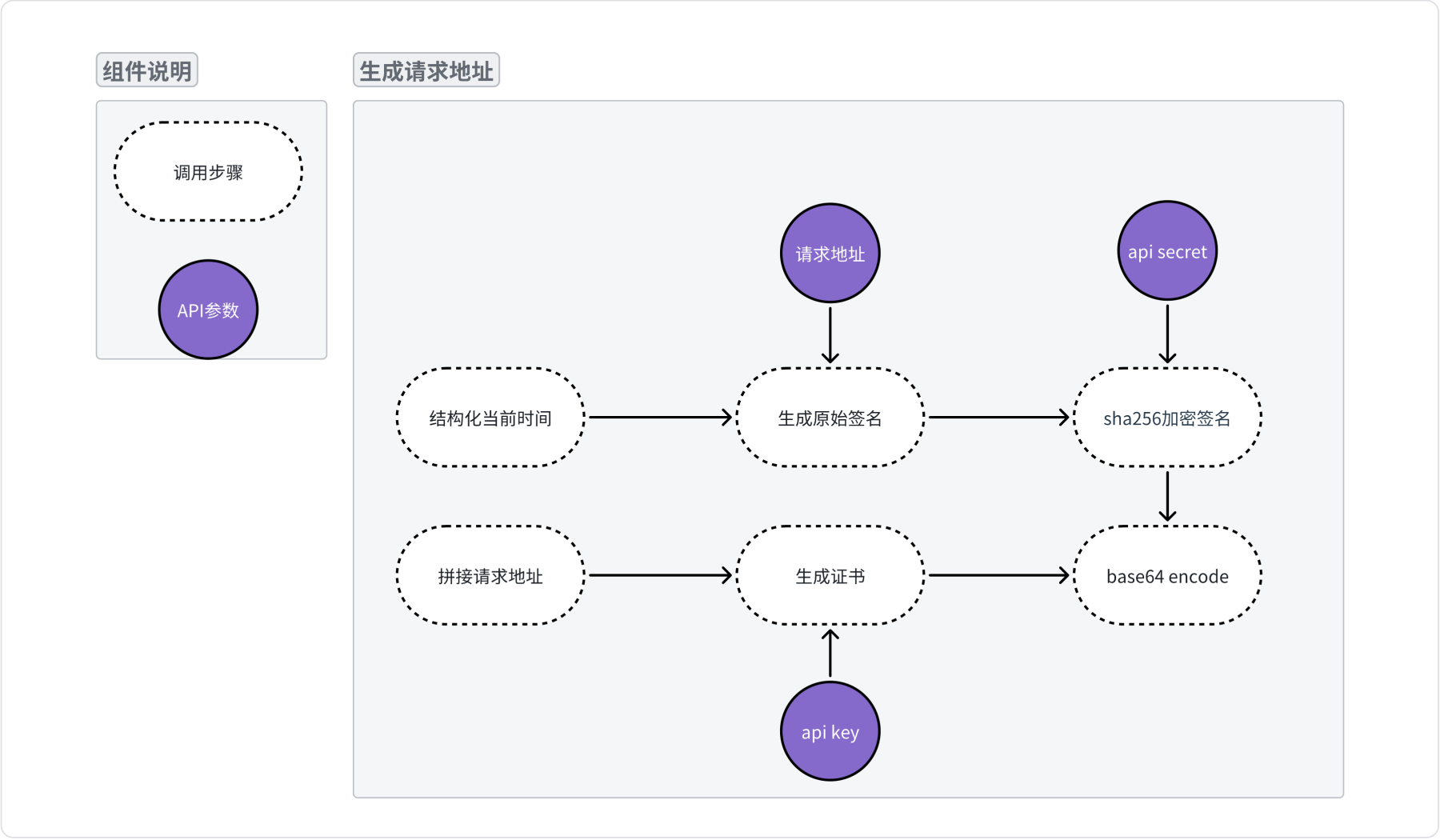
- 首先需要关注API的接口信息。

这一步主要目的是了解API的网络协议，特别是**传输方式**、**请求地址**、**Content-Type**和**接口鉴权**4部分。因为讯飞星火不同的模型有不同的调用方式，所以我们需要根据要求调整我们的网络调用协议。

内容	说明
传输方式	http[s] (为提高安全性, 强烈推荐https)
请求地址	https://spark-api.cn-huabei-1.xf-yun.com/v2.1/tti 注: 服务器IP不固定, 为保证您的接口稳定, 请勿通过指定IP的方式调用接口, 使用域名方式调用
请求行	POST /v2.1/tti HTTP/1.1
Content-Type	application/json;charset=UTF-8
接口鉴权	签名机制, 详情请参照 签名生成
字符编码	UTF-8
响应格式	统一采用JSON格式
开发语言	任意, 只要可以向讯飞云服务发起HTTP请求的均可
适用范围	任意操作系统, 但因不支持跨域不适用于浏览器

2. 查看鉴权方式

讯飞API有一个特点, 就是需要先使用我们的key进行加密鉴权, 然后填充新的url进行实际调用。所以我们需要根据[鉴权文档](#)步骤, 实时调整我们的url。具体方式如图中所示:



鉴权流程

3. 查询调用参数

返回API文档, 根据文档查看模型的可用调用参数。具体如下图所示, 分别为参数实例与参数说明。为了让我们的应用更加的具有特色, 通常更简易针对需求调整API的参数。相同的模型相同的API, 在不同参数的调整下可能会有不同的效果。

请求参数

在调用业务接口时，都需要在 Http Request Body 中配置以下参数，请求数据均为json字符串。
请求参数示例：

json

```
{
  "header": {
    "app_id": "your_appid"
  },
  "parameter": {
    "chat": {
      "domain": "general",
      "width": 512,
      "height": 512
    }
  },
  "payload": {
    "message": {
      "text": [
        {
          "role": "user",
          "content": "帮我画一座山"
        }
      ]
    }
  }
}
```

请求参数说明：

注：文生图目前仅开放单轮交互，单轮交互只需要传递一个user角色的数据

参数名	类型	必传	描述
header.app_id	string	是	应用的app_id
header.uid	string	否	每个用户的id，非必传字段，用于后续扩展，“maxLength”:32
parameter.chat.width	int	图片的宽度	参考下方分辨率说明, 不同的分辨率计费不同，请选择合适的使用
parameter.chat.height	int	图片的宽度	参考下方分辨率说明, 不同的分辨率计费不同，请选择合适的使用
payload.message.text	json/object/array	是	文本数据
payload.message.text.role	string	是	角色，user：表示是用户的问题
payload.message.text.content	string	是	文本内容，该角色的对话内容，不得超过1000个字符

4. 查询返回结果参数与异常信息码

返回结果参数便于我们针对式的开发我们的模型后处理的代码，而异常信息码则便于我们实现异常处理问题，提高应用高可用。

返回参数说明:

参数	类型	含义
header.code	int	服务错误码， 0表示正常，非0表示出错
header.sid	string	会话的sid
header.status	int	会话的状态， 文生图场景下为2
header.message	string	返回消息描述， 错误码的描述信息
payload.choices.status	int	数据状态， 0:开始, 1:开始, 2:结束（表示文本响应结束）
payload.choices.seq	int	数据序号， 最小值:0, 最大值:9999999
payload.choices.text	json object array	文本结果， 是一个json 数组

text字段参数说明

参数	类型	含义
content	string	返回的base64图片结果，默认分辨率512*512
index	int	结果序号，在多候选中使用
role	string	角色，assistant说明这是AI的回复

错误码描述

错误码	错误信息
0	成功
10003	用户的消息格式有错误
10004	用户数据的schema错误
10005	用户参数值有错误
10008	服务容量不足
10021	输入审核不通过
10022	模型生产的图片涉及敏感信息，审核不通过

API文档很复杂，实在看不懂怎么办？

为了降低参赛选手的门槛，我们基于讯飞星火大模型常用的API封装了一个较为简单的SDK包。[\[点我查看\]](#)

安装方法

```
1 # 本地安装
2 pip install dwspark-2024.0.2-py3-none-any.whl
3 # 线上安装
4 pip install dwspark
```

使用方法

```
1 from dwspark.config import Config
2 '''
3     初始化配置文件（二选一）
4 '''
5 # 加载系统环境变量：SPARKAI_APP_ID、SPARKAI_API_KEY、SPARKAI_API_SECRET
6 config = Config()
7 # 自定义key写入
8 #config = Config('14****93', 'eb28b****b82', 'MWM1MzBkOD****Mzk0')
```



```
9
10 '''
11     SDK使用示例
12 '''
13 # SDK引入模型
14 from dwspark.models import ChatModel, Text2Img, ImageUnderstanding, Text2Audio, Audio2Text,
    EmebddingModel
15 # 讯飞消息对象
16 from sparkai.core.messages import ChatMessage
17 # 日志
18 from loguru import logger
19 '''
20     对话
21 '''
22 # 模拟问题
23 question = '你好呀'
24 logger.info('-----批式调用对话-----')
25 model = ChatModel(config, stream=False)
26 logger.info(model.generate([ChatMessage(role="user", content=question)]))
27 logger.info('-----流式调用对话-----')
28 model = ChatModel(config, stream=True)
29 [logger.info(r) for r in model.generate_stream(question)]
30 '''
31     文字生成语音
32 '''
33 text = '2023年5月，讯飞星火大模型正式发布，迅速成为千万用户获取知识、学习知识的“超级助手”，成为解放生产力、释放想象力的“超级杠杆”。2024年4月，讯飞星火V3.5春季升级长文本、长图文、长语音三大能力。一年时间内，讯飞星火从1.0到3.5，每一次迭代都是里程碑式飞跃。'
34 audio_path = './demo.mp3'
35 t2a = Text2Audio(config)
36 # 对生成上锁，预防公有变量出现事务问题，但会降低程序并发性能。
37 t2a.gen_audio(text, audio_path)
38 '''
39     语音识别文字
40 '''
41 a2t = Audio2Text(config)
42 # 对生成上锁，预防公有变量出现事务问题，但会降低程序并发性能。
43 audio_text = a2t.gen_text(audio_path)
44 logger.info(audio_text)
45 '''
46     生成图片
47 '''
48 logger.info('-----生成图片-----')
49 prompt = '一只鲸鱼在快乐游泳的卡通头像'
50 t2i = Text2Img(config)
51 t2i.gen_image(prompt, './demo.jpg')
52 '''
53     图片解释
54 '''
55 logger.info('-----图片解释-----')
56 prompt = '请理解一下图片'
57 iu = ImageUnderstanding(config)
58 logger.info(iu.understanding(prompt, './demo.jpg'))
59 '''
60     获取文本向量
61 '''
62 logger.info('-----获取文本向量-----')
63 em = EmebddingModel(config)
64 vector = em.get_embedding("我们是datawhale")
65 logger.info(vector)
```

但SDK带来便捷性的同时，它也丧失了个性化的“本领”，同时也难以兼容到讯飞所有的模型产品。所以我们更推荐大家基于API文档开发适合自己的调用方式。

我也想开发自己的调用方式，有没有入门参考？

讯飞为每一个API都打包了一个调用demo，demo通常放在对应API文档的上方。我们可以下载查看demo里面的代码，然后基于demo再进行二次开发。




如有实在不会的，可以在学习群里请教师教。

图片生成 API 文档

接口说明

- 根据用户输入的文字内容，生成符合语义描述的不同风格的图像；
- 部分开发语言demo如下，其他开发语言请参照文档进行开发，也欢迎热心的开发者到[讯飞开放平台社区](#)分享你们的demo。

[图片生成 demo go语言](#)[图片生成 demo python语言](#)[图片生成 demo java语言](#)
- 集成图片生成时，需按照以下要求：

 本教程主要由  周理璇 、  徐炽明 写作贡献，经过多位助教内测和建议修改，期待你的点赞和评论～