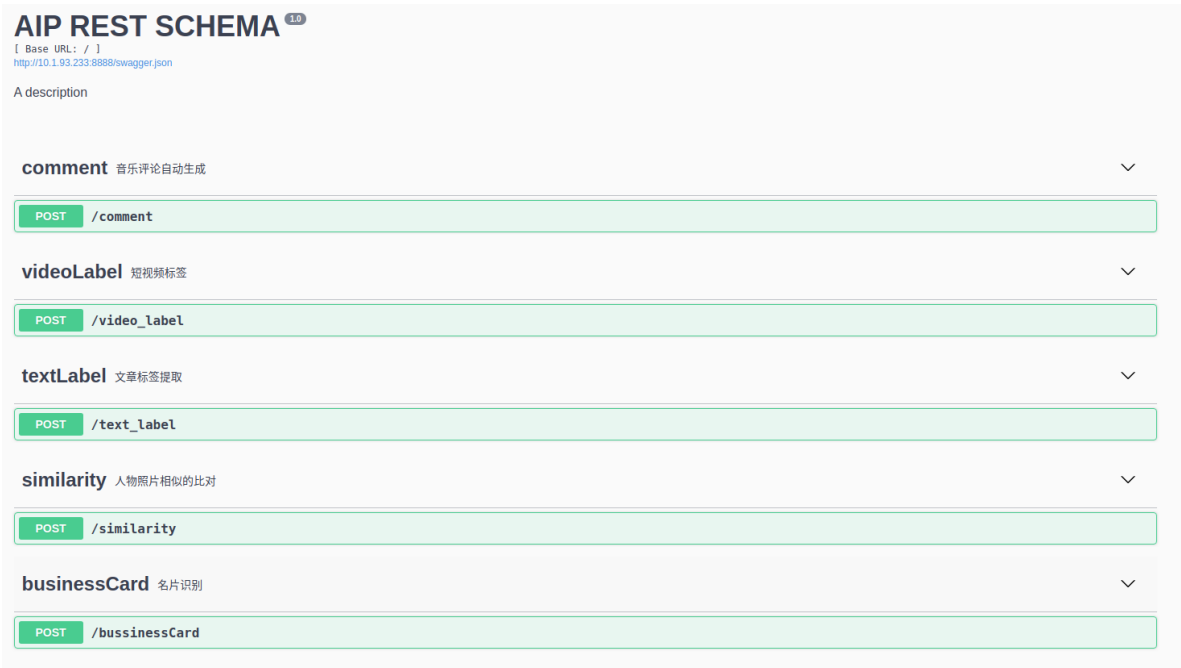
Flask-RESTPlus 使用手册

本手册主要指导原基于 `Flask` 的 Python 应用升级到 `Flask-RESTPlus`，更多内容可查看官方文档。

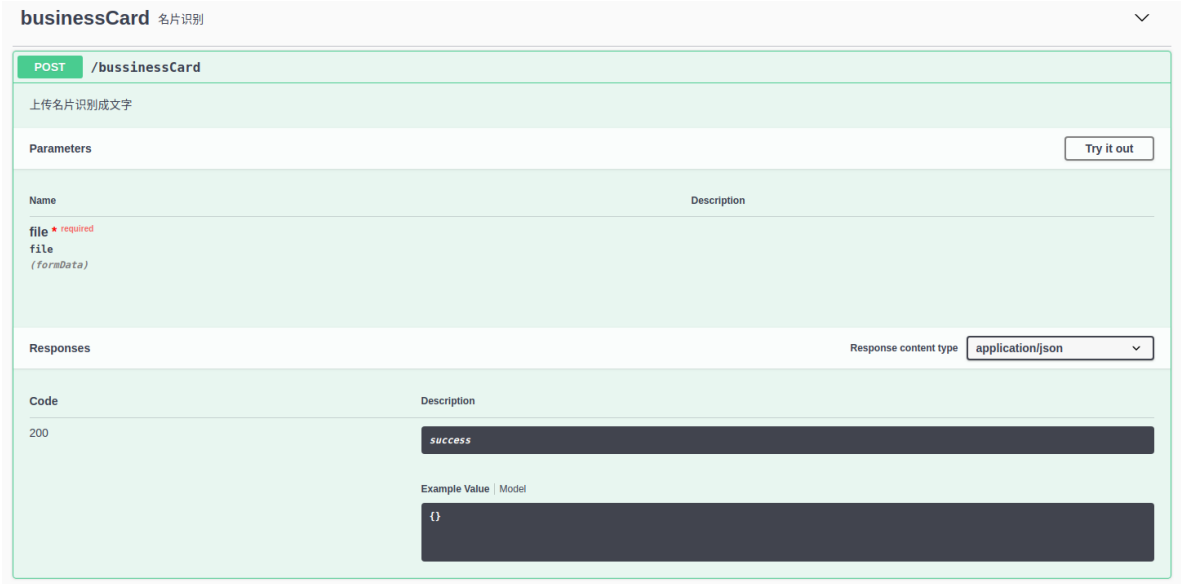
1. Flask-RestPlus 介绍

`Flask-RESTPlus` 基于 `Flask` 增加了快速构建 `REST API` 的支持。只需要少量的设置就能实现最佳实践。如果熟悉 `Flask` 那么很容易就能上手。最终将会生成 `swagger` 规范的API文档。

生成结果



API详情



2. 使用指南

2.1. 示例

按照如下配置，即可平滑的从 `Flask` 迁移至 `Flask-RESTPlus`

```
from flask import Flask
from flask_restplus import Api

app = Flask(__name__)
api = Api(
    title='AIP REST SCHEMA',
    version='1.0',
    description='A description',
    doc="/"
)
api.init_app(app)


@api.route('/hello')
class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

if __name__ == '__main__':
    app.run(debug=True)
```

2.2. 生成REST API

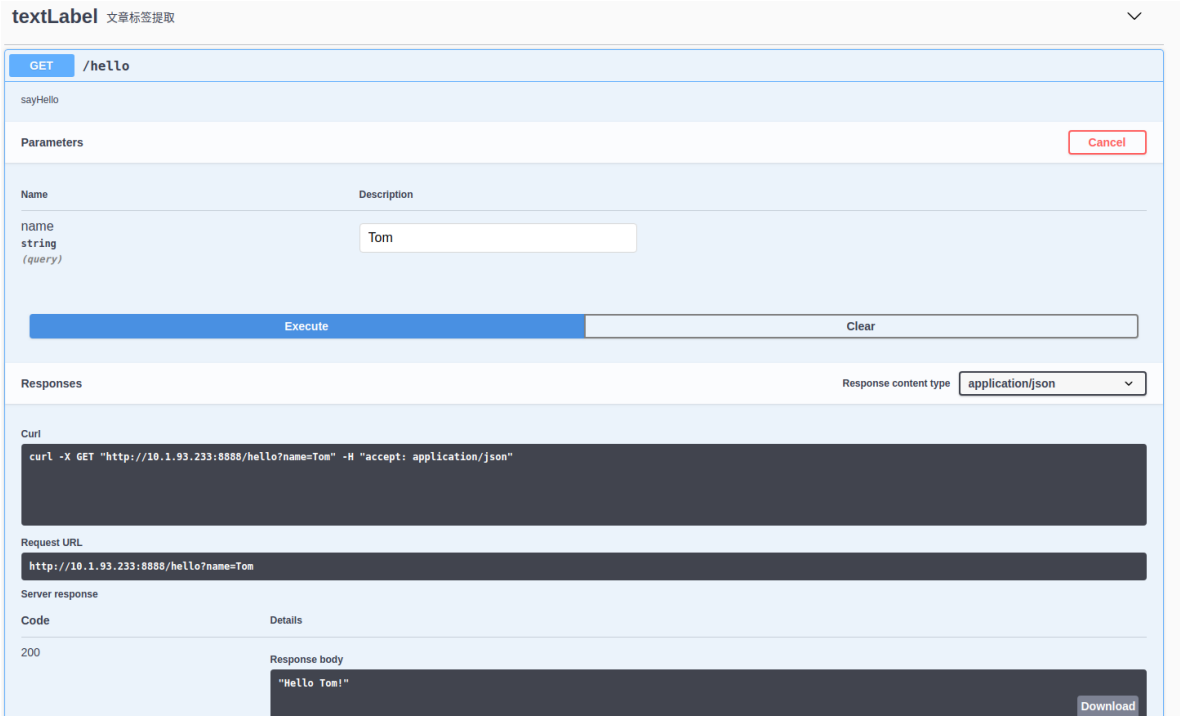
```
from flask import Flask
from flask_restplus import Api

app = Flask(__name__)
api = Api(
    title='AIP REST SCHEMA',
    version='1.0',
```

```
    description='A description',
    doc="/"
)
api.init_app(app)


parser = api.parser()
parser.add_argument("name", help="用户名", location='args')

@api.route('/hello')
class HelloWorld(Resource):
    @api.doc(description="sayHello")
    @api.expect(parser)
    @api.response(200, description="success", model='object')
    def get(self):
        args = parser.parse_args()
        return 'Hello ' + args.get('name') + " !", 200

if __name__ == '__main__':
    app.run(debug=True)
```

2.2.1. 代码说明

- Api：定义Api，指定标题、版本、描述及访问RestApi的路径，整个应用唯一。
- parser：定义请求解析器
- parser.add_argument： 定义请求参数
  - "name"： 参数名
  - help：参数描述，如果传入参数与type不符合，则返回该提示
  - location： 参数位置。可选项为：[args,form,headers,json,values,files]
  - type：参数类型。可选项为：[int,str,bool,float,None] 和 werkzeug.datastructures.FileStorage
  - required：参数是否必须。默认false
  - default：指定默认值
- @api.route：定义请求地址
- @api.doc：定义接口描述
- @api.expect(parser)：绑定请求参数
- @api.response(200, description="success", model='object')： 指定返回值类型
- args = parser.parse_args()： 解析参数
- args.get('name')：获取参数

2.2.2. 访问&测试



3. 升级 Flask-RESTPlus

为支持复杂场景，将接口按层次或类型分为各个模块可以使用 `NameSpace` 。

3.1. 源码

以 `comment.py` 为例

```python
from flask import jsonify, request

from app import app

@app.route('/comment', methods=['POST'])
def comment():
    result = 'success'
    match_flag = 1
    content = []
    upload_file = request.files['file']
    songname = request.form.get('songname')
    singer = request.form.get('singer')
    platform = request.form.get('platform', default='咪咕音乐')
    numbers = request.form.get('numbers')
    motion = request.form.get('positive_ratio', default='0.5')

    if upload_file and songname and singer and platform and numbers and motion:
        # do something
        pass
    else:
        print("There's an error !")
        result = 'error'

    d = {
        'result': result,
```

```
        'content': content
    }
    return jsonify(d)
```

3.2. 基础使用

⚠ 注意： 基础使用是为了尽可能的少改动原来的代码和适配SME，但是丧失了Flask-RESTPlus 提供的参数校验、swagger UI所提供的接口测试及API参考价值等众多优良特性。 新开发的接口不建议这么使用 。

3.2.1. 修改

1. 创建Apis包（将所有基于Flask实现的Python文件放在此包内。）
2. 编辑 `__init__.py`

```
from flask_restplus import Api

api = Api(
    title='AIP REST SCHEMA',
    version='1.0',
    description='AIP 接口文档',
    doc="/"
)
```

3. 复制 `comment.py` 到 `Apis` 包下
4. 修改 `comment.py`
   - 创建namespace
   - 创建类
   - 指定路径
   - 指定请求返回值
   - 指定HTTP方法

```
from flask import jsonify, request
from flask_restplus import Namespace, Resource

# 创建namespace
api = Namespace("comment", path="/", description="音乐评论自动生成")


# 指定路径
@api.route('/comment')
class Comment(Resource):

    # 指定请求返回值
    @api.response(200, description="success", model='object')
    # 指定HTTP方法
    def post(self):
        result = 'success'
        match_flag = 1
        content = []
        upload_file = request.files['file']
        songname = request.form.get('songname')
```

```
            singer = request.form.get('singer')
            platform = request.form.get('platform', default='咪咕音乐')
            numbers = request.form.get('numbers')
            motion = request.form.get('positive_ratio', default='0.5')

            if upload_file and songname and singer and platform and numbers
    and motion:
                # do something
                pass
            else:
                print("There's an error !")
                result = 'error'

            d = {
                'result': result,
                'content': content
            }
            return jsonify(d)

```

5. 添加NameSpace到Api

   编辑 `__init__.py`

```
    from flask_restplus import Api
    #import
    from .comment import api as ns1

    api = Api(
        title='AIP REST SCHEMA',
        version='1.0',
        description='A description',
        doc="/"
    )
    #添加NameSpace到api
    api.add_namespace(ns1)
```

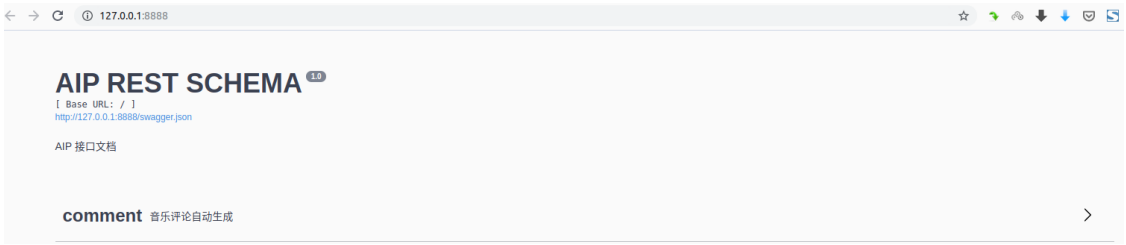
6. 初始化 `api` 对象

   编辑app.py

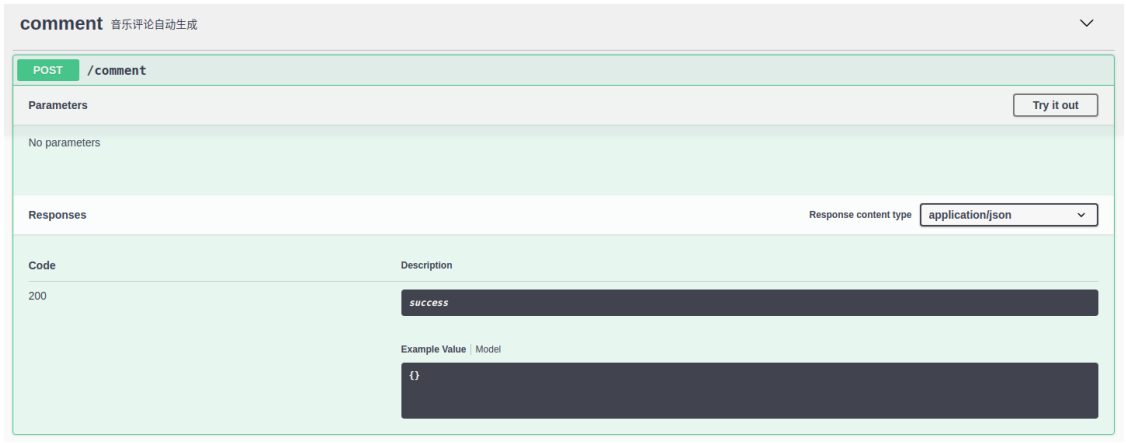
```
    from flask import Flask
    #import api
    from apis import api

    app = Flask(__name__)
    #init api
    api.init_app(app)

    if __name__ == '__main__':
        app.run(debug=True)
```

3.2.2. 启动测试

1. 启动程序，打开浏览器即可访问swagger UI查看契约



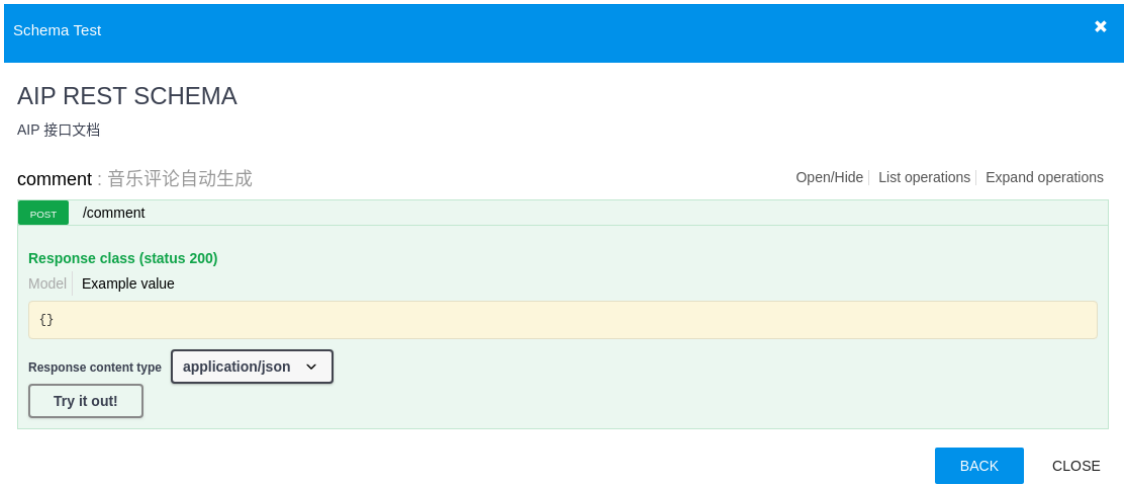
2. 展开具体接口



3. 点击页面上方链接即可获取到契约文件

至此契约文件已经生成，并可以便捷的使用swaggerUI对接口进行查看、测试。SME-Mesher会在启动时主动调用接口获取并更新契约文件。无需手动操作。

4. 启动 `mesher`，在ServiceCenter查看契约文件。



3.3. 高级使用

⚠ 注意：高级使用完全按照Flask-RESTPlus的用法，能够很方便的对参数进行校验，能够生成详细的接口文档，还具备在线测试的功能。

3.3.1. 修改

1. 创建Apis包（将所有基于Flask实现的Python文件放在此包内。）
2. 编辑 `__init__.py`

```python
from flask_restplus import Api

api = Api(
    title='AIP REST SCHEMA',
    version='1.0',
    description='AIP 接口文档',
    doc="/"
)
```

3. 复制 `comment.py` 到 `Apis` 包下
4. 修改 `comment.py`
   - 创建namespace
   - 创建请求解析器
   - 添加参数
   - 创建类
   - 指定路径
   - 添加接口描述（可选）
   - 指定请求参数
   - 指定请求返回值
   - 指定HTTP方法
   - 获取参数

```python
from flask import jsonify
from flask_restplus import Namespace, Resource
from werkzeug.datastructures import FileStorage

#创建namespace
api = Namespace("comment", path="/", description="音乐评论自动生成")

#创建请求解析器
parser = api.parser()

#添加参数
parser.add_argument('file', type=FileStorage, required=True, location='files')
parser.add_argument('songname', required=True, location='form')
parser.add_argument('singer', required=True, location='form')
parser.add_argument('platform', default='咪咕音乐', location='form')
parser.add_argument('numbers', required=True, type=int, help="请输入数字类型", location='form')
parser.add_argument('motion', type=float, default=0.5, help="请输入小数类型", location='form')

#指定路径
@api.route('/comment')
class Comment(Resource):

    #添加接口描述
    @api.doc(description="生成音乐评论")
    #指定请求参数
    @api.expect(parser)
    #指定请求返回值
    @api.response(200, description="success", model='object')
    #指定HTTP方法
```

```
30        def post(self):
31            args = parser.parse_args()
32            result = 'success'
33            match_flag = 1
34            content = []
35            #获取参数
36            # upload_file = request.files['file']
37            upload_file = args.get('file')
38
39            # songname = request.form.get('songname')
40            songname = args.get('songname')
41
42            # singer = request.form.get('singer')
43            singer = args.get('singer')
44
45            # platform = request.form.get('platform', default='咪咕音乐')
46            platform = args.get('platform')
47
48            # numbers = request.form.get('numbers')
49            numbers = args.get('numbers')
50
51            # motion = request.form.get('positive_ratio', default='0.5')
52            motion = args.get('positive_ratio')
53
54            if upload_file and songname and singer and platform and numbers
    and motion:
55                # do something
56                pass
57            else:
58                print("There's an error !")
59                result = 'error'
60
61            d = {
62                'result': result,
63                'content': content
64            }
65            return jsonify(d)
```

5. 添加NameSpace到Api

   编辑 `__init__.py`

```
1    from flask_restplus import Api
2    #import
3    from .comment import api as ns1
4
5    api = Api(
6        title='AIP REST SCHEMA',
7        version='1.0',
8        description='A description',
9        doc="/"
10   )
11   #添加NameSpace到api
12   api.add_namespace(ns1)
```

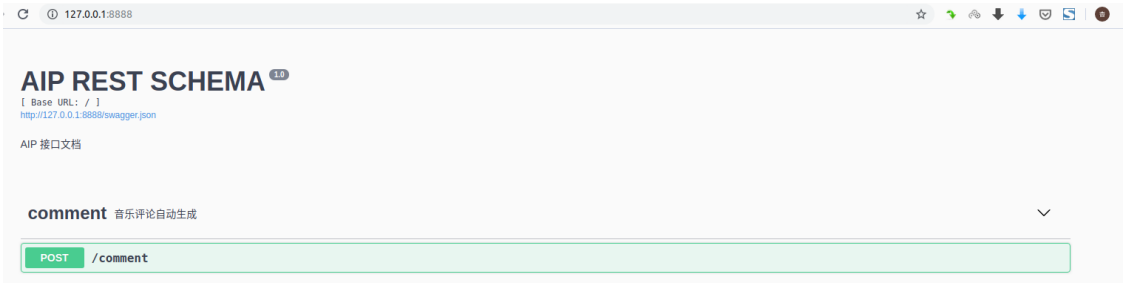
6. 初始化 `api` 对象

   编辑app.py

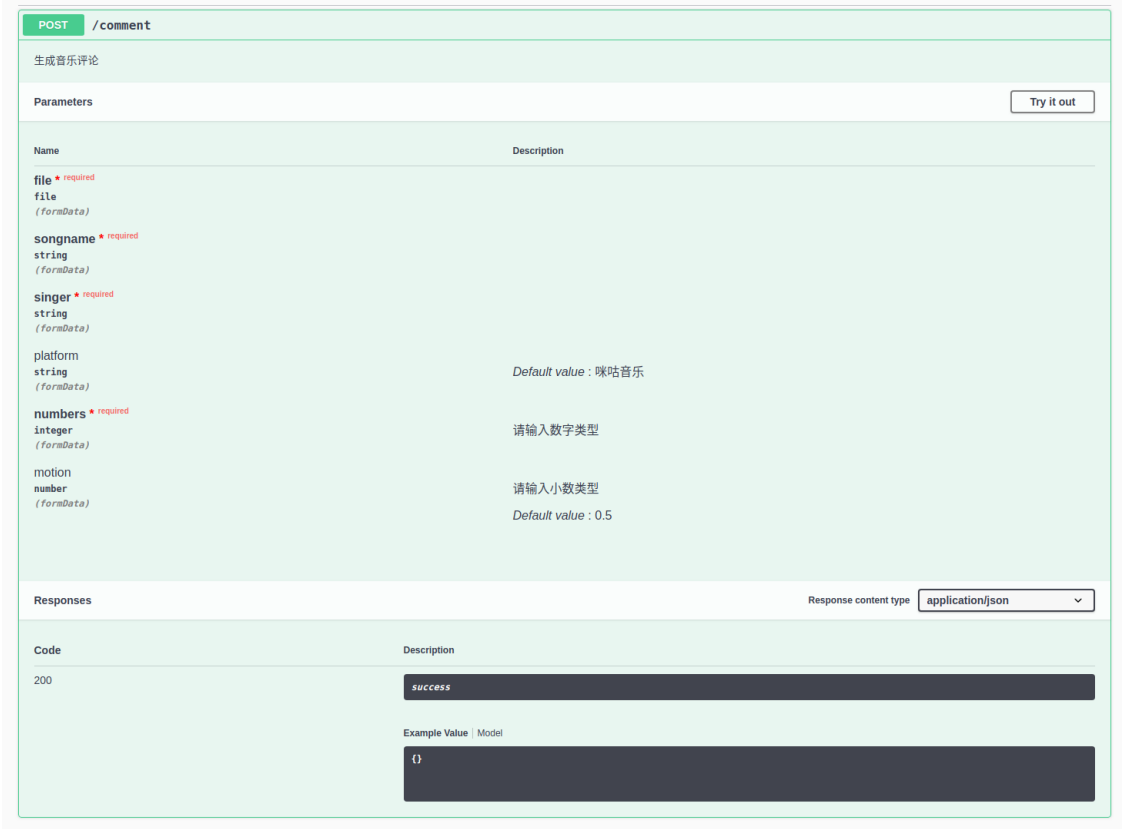
```python
from flask import Flask
#import api
from apis import api

app = Flask(__name__)
#init api
api.init_app(app)

if __name__ == '__main__':
    app.run(debug=True)
```

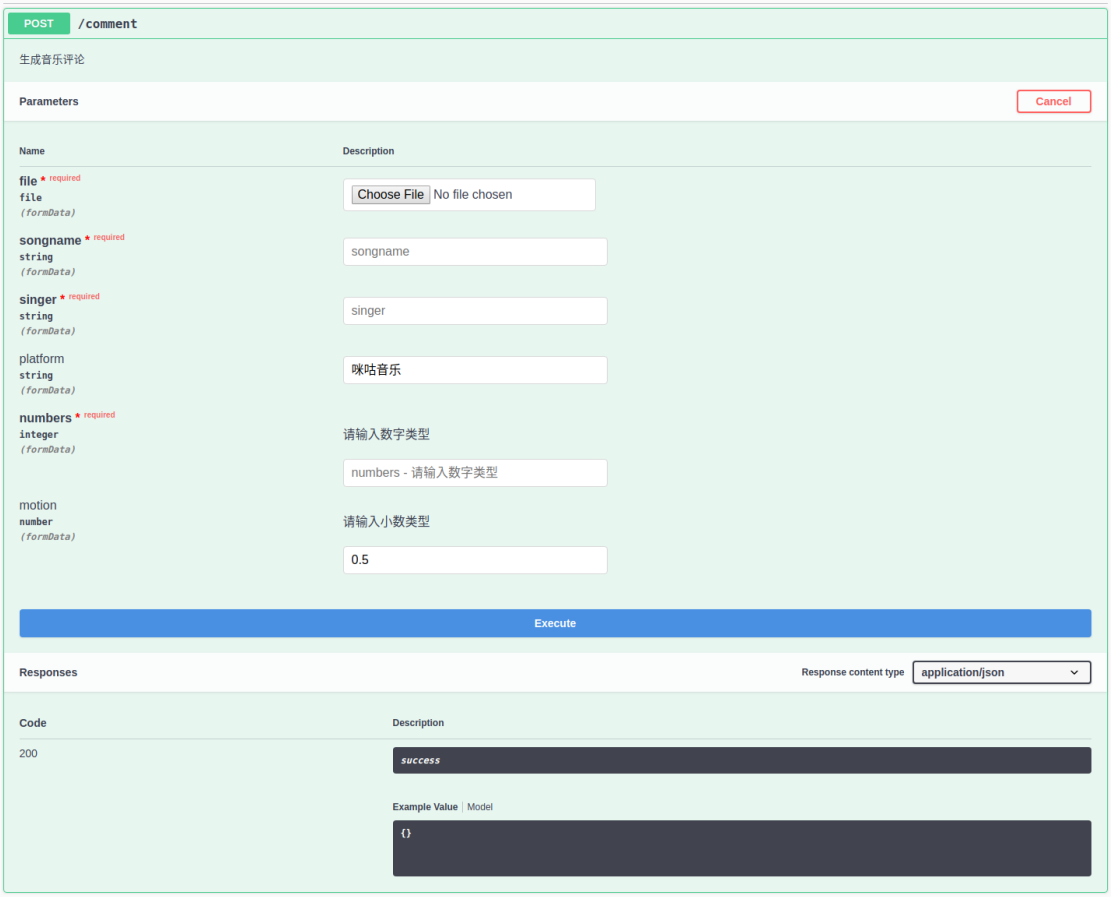
3.3.2. 启动测试

1. 启动程序，打开浏览器即可访问swagger UI查看契约



2. 展开具体接口可以进行测试





3. 点击页面上方链接即可获取到契约文件


```
{"swagger": "2.0", "basePath": "/", "paths": {"/bussinessCard": {"post": {"responses": {"200": {"description": "success", "schema": {"type": "object"}}}, "description":
"\u4e0a\u4f20\u540d\u7247\u8bc6\u522b\u6210\u6587\u5b57", "operationId": "post_business_card", "parameters": [{"name": "file", "in": "formData", "type": "file", "required
["multipart/form-data"], "tags": ["businessCard"]}}, "/comment": {"post": {"responses": {"200": {"description": "success", "schema": {"type": "object"}}}, "description":
"\u751f\u6210\u97f3\u4e50\u8bc4\u8bba", "operationId": "post_comment", "parameters": [{"name": "file", "in": "formData", "type": "file", "required": true}, {"name": "song
"string", "required": true}, {"name": "singer", "in": "formData", "type": "string", "required": true}, {"name": "platform", "in": "formData", "type": "string", "default":
{"name": "numbers", "in": "formData", "type": "integer", "required": true, "description": "\u8bf7\u8f93\u5165\u6570\u5b57\u7c7b\u578b"}, {"name": "motion", "in": "formDat
"description": "\u8bf7\u8f93\u5165\u5c0f\u6570\u7c7b\u578b", "default": 0.5}], "consumes": ["multipart/form-data"], "tags": ["comment"]}}, "/hello": {"get": {"responses":
"success", "schema": {"type": "object"}}}, "description": "sayHello", "operationId": "get_hello_world", "parameters": [{"name": "name", "in": "query", "type": "string",
"\u7528\u6237\u540d", "location": "args"}], "tags": ["textLabel"]}}, "/similarity": {"post": {"responses": {"200": {"description": "success", "schema": {"type": "object"}
"\u4f20\u7167\u7247\u8fdb\u884c\u76f8\u4f3c\u5ea6\u5bf9\u6bd4", "operationId": "post_similarity", "parameters": [{"name": "file1", "in": "formData", "type": "file", "requ
"in": "formData", "type": "file", "required": true}], "consumes": ["multipart/form-data"], "tags": ["similarity"]}}, "/video_label": {"post": {"responses": {"200": {"desc
{"type": "object"}}}, "operationId": "post_video_label", "parameters": [{"name": "file", "in": "formData", "type": "file", "required": true}], "consumes": ["multipart/for
["videoLabel"]}}}, "info": {"title": "AIP REST SCHEMA", "version": "1.0", "description": "AIP \u63a5\u53e3\u6587\u6863"}, "produces": ["application/json"], "consumes": ["
[{"name": "comment", "description": "\u97f3\u4e50\u8bc4\u8bba\u81ea\u52a8\u751f\u6210"}, {"name": "videoLabel", "description": "\u77ed\u89c6\u9891\u6807\u7b7e"}, {"name":
"\u6587\u7ae0\u6807\u7b7e\u63d0\u53d6"}, {"name": "similarity", "description": "\u4eba\u7269\u7167\u7247\u76f8\u4f3c\u7684\u6bd4\u5bf9"}, {"name": "businessCard", "descri
"\u540d\u7247\u8bc6\u522b"}], "responses": {"ParseError": {"description": "When a mask can't be parsed"}, "MaskError": {"description": "When any error occurs on mask"}}}
```


至此契约文件已经生成，并可以便捷的使用swaggerUI对接口进行查看、测试。SME-Mesher会在启动时主动调用接口获取并更新契约文件。无需手动操作。

4. 启动 `mesher` ，在ServiceCenter查看契约文件。