API 客户端

API客户端为开发人员提供了一个方便的API测试接口，而不需要使用浏览器。

客户端通常有两种，基于命令行的和图形化界面的

一、命令行客户端

命令行客户端常用的有curl、wget、HTTPie和coreapi。

这里介绍coreapi。

注意coreapi的命令行客户端，不等同于python客户端模块，需要单独安装：

```
$ pip install coreapi-cli
```

下面是一个使用的例子：

```
$ coreapi get http://api.example.org/
<Pastebin API "http://127.0.0.1:8000/">
snippets: {
    create(code, [title], [linenos], [language], [style])
    destroy(pk)
    highlight(pk)
    list([page])
    partial_update(pk, [title], [code], [linenos], [language], [style])
    retrieve(pk)
    update(pk, code, [title], [linenos], [language], [style])
}
users: {
    list([page])
    retrieve(pk)
}
```

使用coreapi action命令和API进行交互：

```
$ coreapi action users list
[
    {
        "url": "http://127.0.0.1:8000/users/2/",
        "id": 2,
        "username": "aziz",
        "snippets": []
    },
    ...
]
```

使用 `--debug` 标识，查看具体的交互信息：

```
$ coreapi action users list --debug
> GET /users/ HTTP/1.1
> Accept: application/vnd.coreapi+json, */*
> Authorization: Basic bWF4Om1heA==
> Host: 127.0.0.1
> User-Agent: coreapi
< 200 OK
< Allow: GET, HEAD, OPTIONS
< Content-Type: application/json
< Date: Thu, 30 Jun 2016 10:51:46 GMT
< Server: WSGIServer/0.1 Python/2.7.10
< Vary: Accept, Cookie
<
< [{"url":"http://127.0.0.1/users/2/","id":2,"username":"aziz","snippets":
[]},{"url":"http://127.0.0.1/users/3/","id":3,"username":"amy","snippets":
["http://127.0.0.1/snippets/3/"]},
{"url":"http://127.0.0.1/users/4/","id":4,"username":"max","snippets":
["http://127.0.0.1/snippets/4/","http://127.0.0.1/snippets/5/","http://127.
0.0.1/snippets/6/","http://127.0.0.1/snippets/7/"]},
{"url":"http://127.0.0.1/users/5/","id":5,"username":"jose","snippets":[]},
{"url":"http://127.0.0.1/users/6/","id":6,"username":"admin","snippets":
["http://127.0.0.1/snippets/1/","http://127.0.0.1/snippets/2/"]}]

[
    ...
]
```

使用选项或必填的参数：

```
$ coreapi action users create --param username=example
```

如果你想让命令的意思更加明确，使用 `--data` 标识用于空、数字、布尔、列表或者对象的输入，使用 `--string` 标识用于字符串输入：

```
$ coreapi action users edit --string username=tomchristie --data
is_admin=true
```

认证和头部属性

加入证书：

```
$ coreapi credentials add <domain> <credentials string>
```

例如：

```
$ coreapi credentials add api.example.org "Token
9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b"
```

`--auth` 标识用于选择认证类型：

```
$ coreapi credentials add api.example.org tomchristie:foobar --auth basic
```

使用 `headers` 命令添加头部属性：

```
$ coreapi headers add api.example.org x-api-version 2
```

使用 `coreapi help` 查看更多的帮助。

编码解码器Codecs

默认情况下，coreapi只支持JSON格式，但是可以安装别的插件来增加支持的类型：

```
$ pip install openapi-codec jsonhyperschema-codec hal-codec
$ coreapi codecs show
Codecs
corejson        application/vnd.coreapi+json encoding, decoding
hal             application/hal+json         encoding, decoding
openapi         application/openapi+json     encoding, decoding
jsonhyperschema application/schema+json      decoding
json            application/json             data
text            text/*                       data
```

Utilities

书签功能:

```
$ coreapi bookmarks add accountmanagement
```

查看历史记录:

```
$ coreapi history show
$ coreapi history back
```

更多帮助查看 `coreapi bookmarks --help` 或者 `coreapi history --help`.

其它命令

显示当前的 `Document`:

```
$ coreapi show
```

重载当前的 `Document` :

```
$ coreapi reload
```

从本地硬盘加载概要:

```
$ coreapi load my-api-schema.json --format corejson
```

dump当前文档:

```
$ coreapi dump --format openapi
```

删除当前文档:

```
$ coreapi clear
```

二、Python的coreapi模块

`coreapi` 这个Python模块允许你以编程的方式与API交互。这不同于前面的coreapi命令行使用环境。

安装：

```
$ pip install coreapi
```

导入模块后，首先需要实例化一个 `Client` 对象，如下所示：

```
import coreapi
client = coreapi.Client()
```

然后可以通过这个client对象访问API了：

```
schema = client.get('https://api.example.org/')
```

返回的是一个 `Document` 实例，也就是API的概要。

认证方式

使用 `TokenAuthentication` 类进行认证：

```
auth = coreapi.auth.TokenAuthentication(
    scheme='JWT',
    token='<token>'
)
client = coreapi.Client(auth=auth)
```

或者使用 "Django REST framework JWT" 模块

```
client = coreapi.Client()
schema = client.get('https://api.example.org/')

action = ['api-token-auth', 'create']
params = {"username": "example", "password": "secret"}
result = client.action(schema, action, params)

auth = coreapi.auth.TokenAuthentication(
    scheme='JWT',
    token=result['token']
)
client = coreapi.Client(auth=auth)
```

或者使用最基础的 `BasicAuthentication` 方式，使用用户名和密码进行认证：

```
auth = coreapi.auth.BasicAuthentication(
    username='<username>',
    password='<password>'
)
client = coreapi.Client(auth=auth)
```

与API交互

使用action方法进行交互：

```
users = client.action(schema, ['users', 'list'])
```

添加参数：

```
new_user = client.action(schema, ['users', 'create'], params={"username": "max"})
```

Codecs

配置Codecs：

```
from coreapi import codecs, Client

decoders = [codecs.CoreJSONCodec(), codecs.JSONCodec()]
client = Client(decoders=decoders)
```

可以直接使用：

```
input_file = open('my-api-schema.json', 'rb')
schema_definition = input_file.read()
codec = codecs.CoreJSONCodec()
schema = codec.load(schema_definition)
```

或者这样：

```
schema_definition = codec.dump(schema)
output_file = open('my-api-schema.json', 'rb')
output_file.write(schema_definition)
```

三、图形化API工具Postman

Postman是google开发的一款功能强大的API测试工具，可以作为Chrome浏览器的插件使用。

其主要功能包括：

- 允许用户发送任何类型的 HTTP 请求，例如 GET，POST，HEAD，PUT、DELETE等
- 允许任意的参数和 Headers
- 支持不同的认证机制，包括 Basic Auth，Digest Auth，OAuth 1.0，OAuth 2.0等
- 可以响应数据是自动按照语法格式高亮的，包括 HTML，JSON和XML

官方主页：**https://www.getpostman.com/**

下载链接：**https://www.getpostman.com/downloads/**

本来最简单的方法是在Chrome中以浏览器插件的方式安装postman，但是你懂的，所以只能下载app，并安装。

（在Linux命令行下老老实实用wget、curl、httpie和coreapi吧！别想图形界面的事了）