概要是一种机器可读的文档，用于描述可用的API，其URLS，以及它们支持的操作。

概要可用于自动生成文档，也可以用于驱动可与API进行交互的动态客户端库。

一、 Core API

为了提供概要支持，REST框架引用了Core API规范。

Core API是用于描述API的文档规范。它用于提供API的内部表示形式和交互方式。可同时用于服务器端或客户端。

当在服务器端使用时，Core API支持以各种模式或超媒体格式呈现。

当在客户端使用时，Core API允许动态驱动的客户端库与任何公开支持的模式或超媒体格式的API交互。

二、添加概要

REST框架支持明确定义的概要视图，也可以自动生成概要。由于教程走到这里，我们使用的是视图集和路由器，所以可以简单地使用自动生成概要的方式。

我们需要安装`coreapi` python包才能生成API概要，还需要安装pyyaml库，渲染概要，使之成为通用的基于YAML格式的OpenAPI 。

```
$ pip install coreapi pyyaml
```

现在我们可以通过在URL配置中包含一个自动生成的概要视图来为API添加概要。

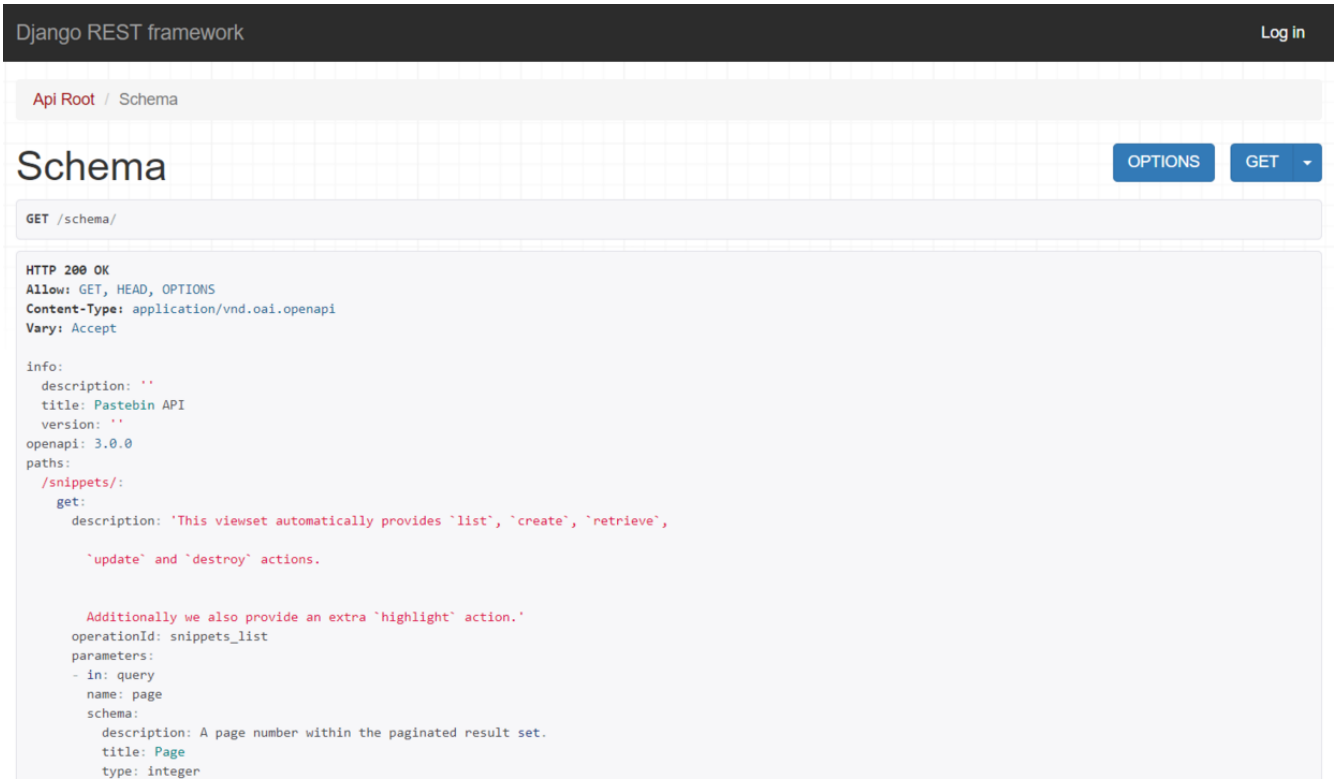
在根路由urls.py下，灵活地插入下面的代码：

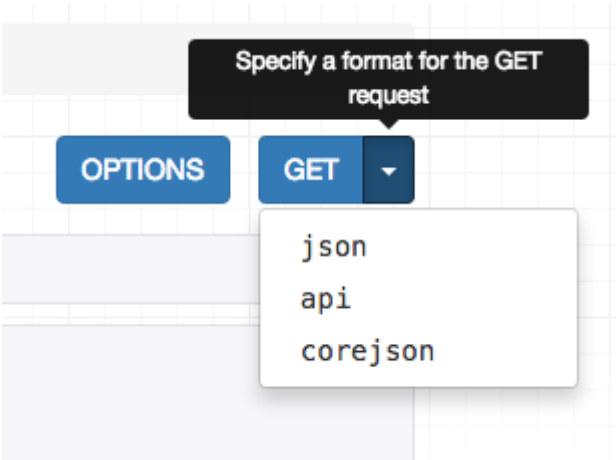
```
from rest_framework.schemas import get_schema_view

schema_view = get_schema_view(title='Pastebin API')

urlpatterns = [
    path('schema/', schema_view),
    ...
]
```

重启服务器，在浏览器中访问 `http://127.0.0.1:8000/schema/` ，可以看到 `corejson` 成为可用选项之一。





我们也可以通过在 `Accept` 标头中指定所需的内容类型从命令行请求概要。

```
$ http http://127.0.0.1:8000/schema/ Accept:application/coreapi+json



HTTP/1.1 200 OK
Allow: GET, HEAD, OPTIONS
Content-Length: 1919
Content-Type: application/coreapi+json
```

```
Date: Sun, 28 Apr 2019 15:01:04 GMT
Server: WSGIServer/0.2 CPython/3.7.3
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
    "_meta": {
        "title": "Pastebin API",
        "url": "http://127.0.0.1:8000/schema/"
    },
    "_type": "document",
    "snippets": {
    ...
```

默认输出样式是使用Core JSON编码。

还支持其他概要格式，如Open API（以前叫Swagger）。

二、 使用命令行客户端与API进行交互

现在我们的API暴露了一个概要url，我们可以使用一个动态的客户端库与API进行交互。为了演示这个，我们来使用Core API命令行客户端。

先安装需要的 `coreapi-cli` 包：

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

检查一下安装是否成功：

```
$ coreapi


Usage: coreapi [OPTIONS] COMMAND [ARGS]...

  Command line client for interacting with CoreAPI services.

  Visit http://www.coreapi.org for more information.

Options:
  --version  Display the package version number.
  --help     Show this message and exit.

Commands:
```

```
  action       Interact with the active document.
  bookmarks    Add, remove and show bookmarks.
  clear        Clear the active document and other state.
  codecs       Manage the installed codecs.
  credentials  Configure request credentials.
  describe     Display description for link at given PATH.
  dump         Dump a document to console.
  get          Fetch a document from the given URL.
  headers      Configure custom request headers.
  history      Navigate the browser history.
  load         Load a document from disk.
  reload       Reload the current document.
  show         Display the current document.
```

首先，使用命令行客户端加载API概要：

```
$ coreapi get http://127.0.0.1:8000/schema/



<Pastebin API "http://127.0.0.1:8000/schema/">
    snippets: {
        list([page])
        read(id)
        highlight(id)
    }
    users: {
        list([page])
        read(id)
    }
```

我们还没有认证，所以现在只能看到只读的API，这与我们设置的API权限是一致的。

使用命令行客户端，尝试列出现有的代码片段：

```
$ coreapi action snippets list

{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "url": "http://127.0.0.1:8000/snippets/1/",
```

```
            "id": 1,
            "highlight": "http://127.0.0.1:8000/snippets/1/highlight/",
            "owner": "admin",
            "title": "test",
            "code": "print('hello')",
            "linenos": false,
            "language": "abap",
            "style": "abap"
        },
        {
            "url": "http://127.0.0.1:8000/snippets/2/",
            "id": 2,
            "highlight": "http://127.0.0.1:8000/snippets/2/highlight/",
            "owner": "admin",
            "title": "haha",
            "code": "import this",
            "linenos": false,
            "language": "abap",
            "style": "abap"
        },
        {
            "url": "http://127.0.0.1:8000/snippets/3/",
            "id": 3,
            "highlight": "http://127.0.0.1:8000/snippets/3/highlight/",
            "owner": "admin",
            "title": "what is new",
            "code": "print('i dont know')",
            "linenos": false,
            "language": "abap",
            "style": "abap"
        }
    ]
}
```

访问一些API需要提供关键字参数。例如，要获取特定代码片段的高亮HTML表示，我们需要提供一个id：

```
$ coreapi action snippets highlight --param id=1

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">

<html>
<head>
```

```
  <title>test</title>
  <meta http-equiv="content-type" content="text/html; charset=None">
  <style type="text/css">
td.linenos { background-color: #f0f0f0; padding-right: 10px; }
span.lineno { background-color: #f0f0f0; padding: 0 5px 0 5px; }
pre { line-height: 125%; }
...
...
body .il { color: #33aaff } /* Literal.Number.Integer.Long */

  </style>
</head>
<body>
<h2>test</h2>

<div class="highlight"><pre><span></span><span class="nv">print</span><span
class="p">(</span><span class="s1">&#39;hello&#39;</span><span cla
ss="p">)</span>
</pre></div>
</body>
</html>

```

带认证参数进行API访问

如果我们想要创建，编辑和删除代码片段，我们需要进行合法的用户身份验证。在教程中，我们只需使用基本的auth。

请确保使用实际的用户名和密码替换下面的 `<username>` 和 `<password>` 。

```
$ coreapi credentials add 127.0.0.1 <username>:<password> --auth basic



Added credentials
127.0.0.1 "Basic YWRtaW46MTIzNGFzZGY="

```

现在，如果我们再次访问概要API，我们将获得完整可用的操作权限。

```
$ coreapi reload

```

```
<Pastebin API "http://127.0.0.1:8000/schema/">
    snippets: {
        list([page])
        create(code, [title], [linenos], [language], [style])
        read(id)
        update(id, code, [title], [linenos], [language], [style])
        partial_update(id, [title], [code], [linenos], [language], [style])
        delete(id)
        highlight(id)
    }
    users: {
        list([page])
        read(id)
    }
```

我们现在能够与这些API行交互。例如，要创建一个新的代码片段:

```
$ coreapi action snippets create --param title="Example" --param
code="print('hello, world')"


{
    "url": "http://127.0.0.1:8000/snippets/4/",
    "id": 4,
    "highlight": "http://127.0.0.1:8000/snippets/4/highlight/",
    "owner": "admin",
    "title": "Example",
    "code": "print('hello, world')",
    "linenos": false,
    "language": "python",
    "style": "friendly"
}
```

或者删除一个代码片段:

```
$ coreapi action snippets delete --param id=7
```

似乎和HTTPie差不多。

最后

DRF的官方快速入门教程到这里就结束了。