**一、内容协商Content negotiation**

内容协商：客户端与服务器协商使用哪种数据格式返回给客户端的过程。

一般使用默认的



在DRF的settings里面的默认配置如下，自带的一般不配置

REST\_FRAMEWORK = {

'DEFAULT\_CONTENT\_NEGOTIATION\_CLASS':'myapp.negotiation.IgnoreClientContentNegotiation',

}

**二．元数据Metadata**

REST框架包含一个可配置的机制，用于确定API如何响应 OPTIONS 方式的HTTP请求。即可以在浏览器中点击OPTIONS ，将返回API模式或其他资源信息。

REST\_FRAMEWORK = {

'DEFAULT\_CONTENT\_NEGOTIATION\_CLASS':'rest\_framework.metadata.SimpleMetadata'

,

}

1. **概要Schemas**

API概要是一个有用的工具，通过它你可以对服务器提供的所有API有一个整体上的浏览或查询，包括生成参考文档，或者驱动与API交互的动态客户端。

1. **安装 Core API 和PyYAML**

我们需要安装 coreapi 工具，为DRF提供概要支持。也有可能需要安装 pyyaml 模块，这样你 就可以将概要渲染成常用的基于YAML的OpenAPI格式。通过pip一起安装它们吧

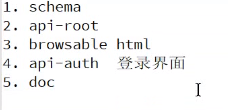
pip install coreapi pyyaml

**2 生成概要（在项目根目录下）**



然后访问<http://127.0.0.1:8000/schema/>查看概要

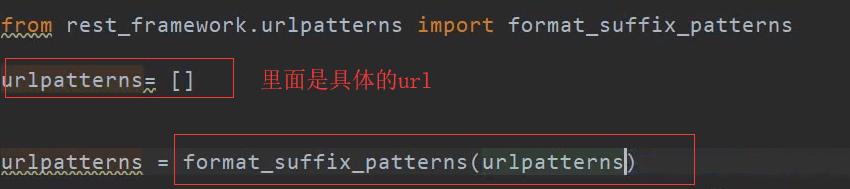
注意区分DRF提供的如下内容：



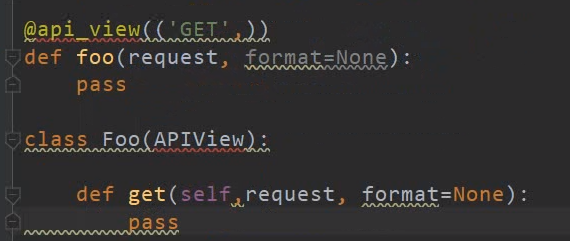
1. **格式后缀Format suwixes**

在Web APIs中，经常使用 http://example.com/api/users**.json** 或者： http://example.com/organizations/**?format=csv**这种，以文件扩展名为后缀的形式或者是请求参数的形式，要求响应返回的内容类型。

1. 使用DRF的DefaultRouter路由和ViewSet视图，将默认支持以上两种请求方式
2. 如果没有使用DefaultRouter路由和ViewSet视图那需要如下配置
3. 需要在你的需要**格式化的app**里面的**urls**配置如下



1. 需要在url对应的视图做如下处理（一个是FBV,一个是CBV）



1. **Reverse（反向解析）**

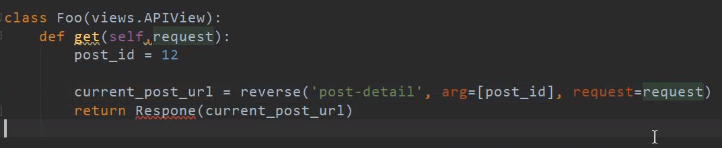
reverse就是解析或生成url的意思，其行为和Django原生的 django.urls.reverse 基本一样，除了它使用request中的信息确定主机和端口信息，然后返回的一个完全限定的URL

比如我们要访问：<http://www.baidu.com/posts/12>

1. urls:



1. views

注意：

1. 我们是在python的代码中，解析或生成url
2. post-detail 模板名
3. arg 参数（本例是为了生产/posts/12这个url）
4. Request

前面用户发过来的请求，它里面包含了域名http://www.baidu.com,然后把两者拼接到一起，生产完整的url

1. 返回一个response对象
2. **异常Exceptions**

**自定义异常处理的方法**

1. 在views里面,返回DRF默认的异常信息，在加上我们自定义的内容，一起返回回去



1. 在settings里面指定自己所编写的异常处理类(比如为app2)

REST\_FRAMEWORK = {

'EXCEPTION\_HANDLER': 'app2.views.my\_exception\_handler'

#注意这个路径

}

1. **状态码**
2. **信息** - 1xx

此类状态代码表示临时响应。默认情况下，REST框架中没有使用1xx状态代码。

1. **成功 - 2xx**

此类状态码表示已成功接收、解析和接受客户端的请求。

1. **重定向 - 3xx**

此类状态码指示用户代理需要采取进一步操作才能完成请求。

1. **客户端错误 - 4xx**

4xx类状态代码适用于客户端似乎有错误的情况。除了在响应HEAD请求时，服务器应该包括一个实体，其中包含错误情况的解释，以及它是临时还是永久

1. **服务器错误 - 5xx**

以数字“5”开头的响应状态码表示服务器知道它已经出错或无法执行请求的情况。除了在响应 HEAD请求时，服务器应该包括一个实体，其中包含错误情况的解释，以及它是临时或永久

1. **辅助函数**

is\_informational() # 1xx

is\_success() # 2xx

is\_redirect() # 3xx

is\_client\_error() # 4xx

is\_server\_error() # 5xx

1. **Settings**

REST framework所有的配置参数都放置在Django的settings文件中的 REST\_FRAMEWORK 变量中。

使用 api\_settings 对象来访问你在settings中配置的参数值，例如：

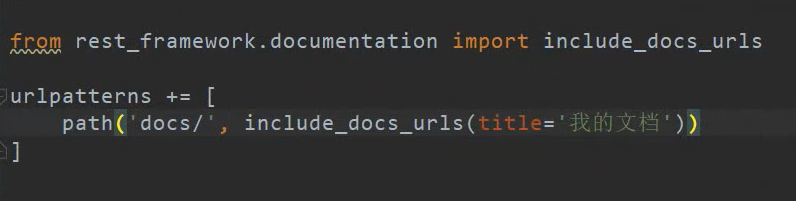
from rest\_framework.settings import api\_settings

print(api\_settings.DEFAULT\_AUTHENTICATION\_CLASSES)

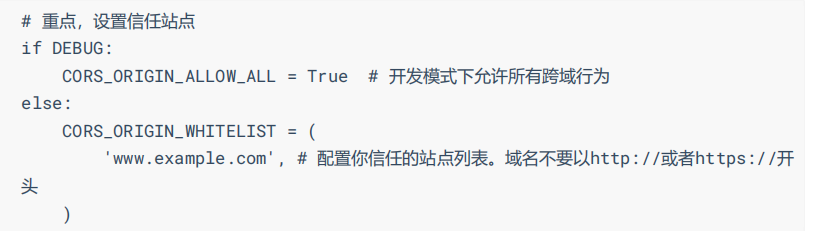
1. **文档**

**安装**

1. 依赖 coreapi ，pygments 代码高亮和 markdown 编辑语法可选安装。
2. DRF自带的只需在项目的根目录下导入即可



1. **跨域同源策略**
2. **在DRF后端，可使用DRF自带的**
3. pip install django-cors-headers
4. 在setting下的app里面注册corsheaders模块
5. 配置中间件
6. 在配置在CommonMiddleware前面
7. 配置参数



最后前后端分离后，由前端的代理服务器等解决