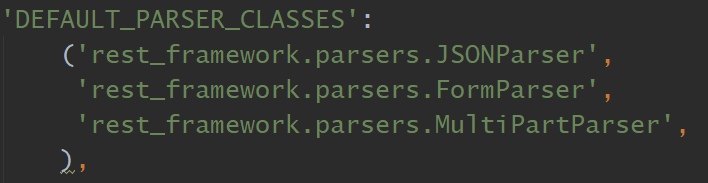
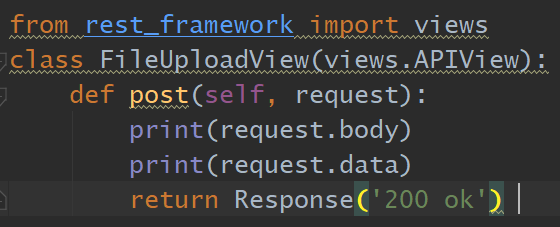
发送from-data：

Settings:

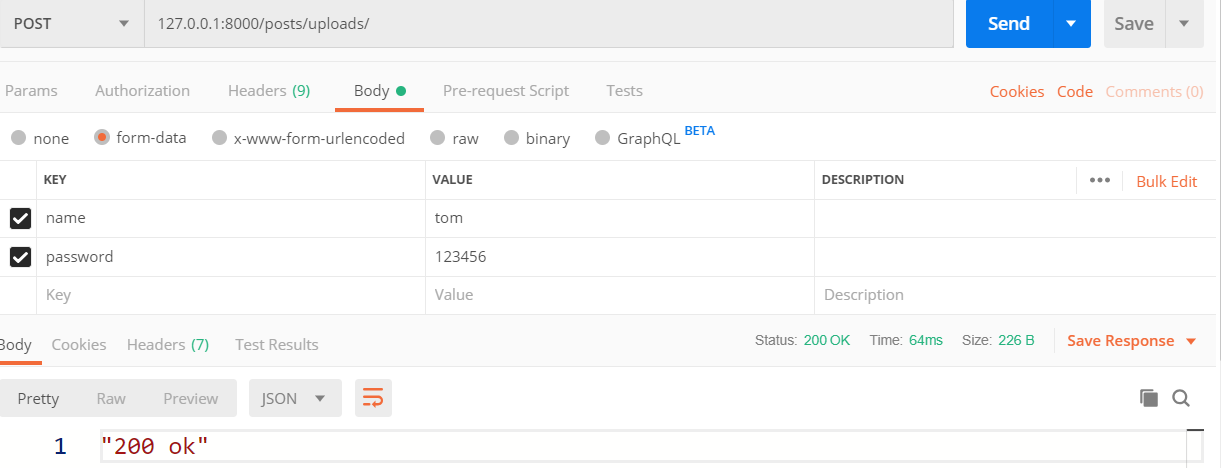
注意：在seetings下的解析器优先级一样，会依次查找



Views:



Urls:



结果

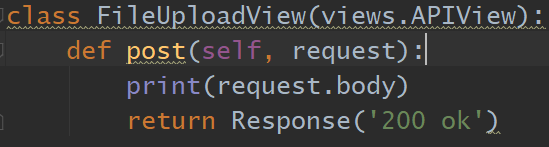
b'----------------------------210428822457045831572470\r\nContent-Disposition:form-data; name="name"\r\n\r\ntom\r\n----------------------------210428822457045831572470\r\nContent-Disposition:form-data; name="password"\r\n\r\n123456\r\n----------------------------210428822457045831572470--\r\n'

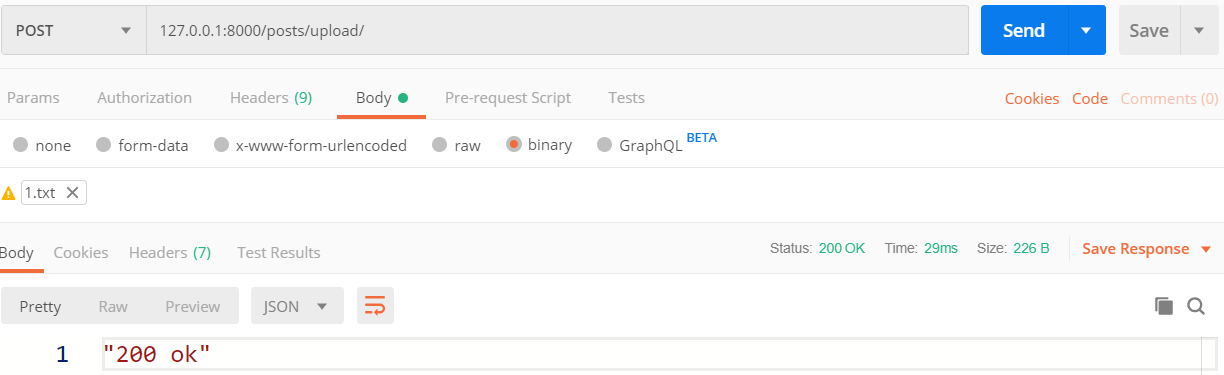
<QueryDict: {'name': ['tom'], 'password': ['123456']}>

request.data为解析完的数据

发送binary(图片或者文本):

Views:





request.body为图片或者文本的内容如：1.txt里面的

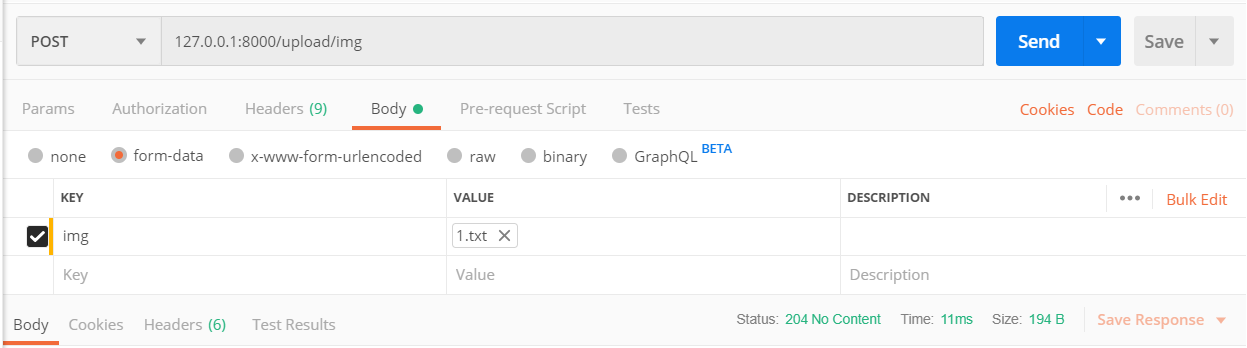


数据并不在request.data中，因此上传文件类型最好继承parasers.FileUploadParser

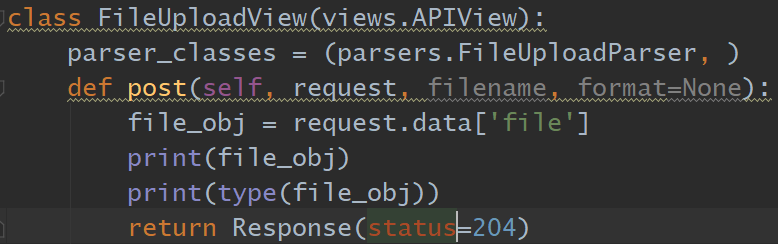
**FileUploadParser：**

解析原始文件上传内容。此时， request.data 属性将是一个字典，并且只包含一个键，这个键叫做 **'file'** ，对应的值包含上传的文件内容。

如果使用FileUploadParser解析器的视图，在被调用的时候URL中携带一个 **filename 关**键字参数，则该参数将被用作文件名。如果在没有这个关键字参数的情况下调用它，则客户端必须在 HTTP头部的 Content-Disposition 中设置文件名。例如 Content-Disposition: attachment; filename=upload.jpg 。



Views：



Urls：



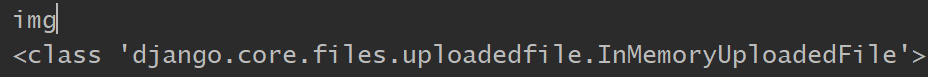
其中必须指明filename即：



打印结果：

Print(file\_obj)

Print(type(file\_obj))



img 为文件名（在路由中指定）

InMemoryUploadedFile是Django原生的文件上传类型可直接使用ORM与models.ImageFiled交联

**第三方模块：yaml与xml与自定义解析器(自行了解)**