1.request对象

Flask中获取请求数据，与django中不同，

django视图中第一个参数必须是HttpRequest对象，

Flask中需要导入从flask中导入request对象，request对象中已经封装所有的请求参数。

属性	功能	类型
form	一个包含解析过的从 POST 或 PUT 请求发送的表单对象的 MultiDict 。请注意上传的文件会在这里，而是在 files 属性中。	MultiDict
args	一个包含解析过的查询字符串（URL 中问号后的部分）内容的	MultiDict
values	一个包含 form 和 args 全部内容	CombinedMultiDict
cookies	一个包含请求中传送的所有 cookie 内容	dict
headers	请求头存为一个类似字典的对象	dict
method	当前请求的 HTTP 方法 (POST， GET 等等)	string
files	一个包含 POST 和 PUT 请求中上传的文件的 MultiDict 。每个文件存储为 FileStorage 对象。其基本的行为类似你在 Python 中见到的标准文件对象，差异在于这个对象有一个 save() 方法可以把文件存储到文件系统上。	

django中查询字典QueryDict
flask是 一键多值字典(multidict)

OrderedMultiDict --- 但保持了字典秩序 有序字典 ---一般来说 OrderedMultiDict 是一个数量级慢于 MultiDict .

CombinedMultiDict ---- 不变的multidict

```
>>> d = MultiDict([('a', 'b'), ('a', 'c')])
>>> d
MultiDict([('a', 'b'), ('a', 'c')])
>>> d['a']
'b'
>>> d.getlist('a')
['b', 'c']
>>> 'a' in d
True
```

path 路径
script_root 根路由

```
url url地址
base_url  根地址
url_root 根路由地址

http://www.example.com/myapplication/page.html?x=y
```

属性	获取路径
path	/page.html
script_root	/myapplication
base_url	http://www.example.com/myapplication/page.html
url	http://www.example.com/myapplication/page.html?x=y
url_root	http://www.example.com/myapplication/

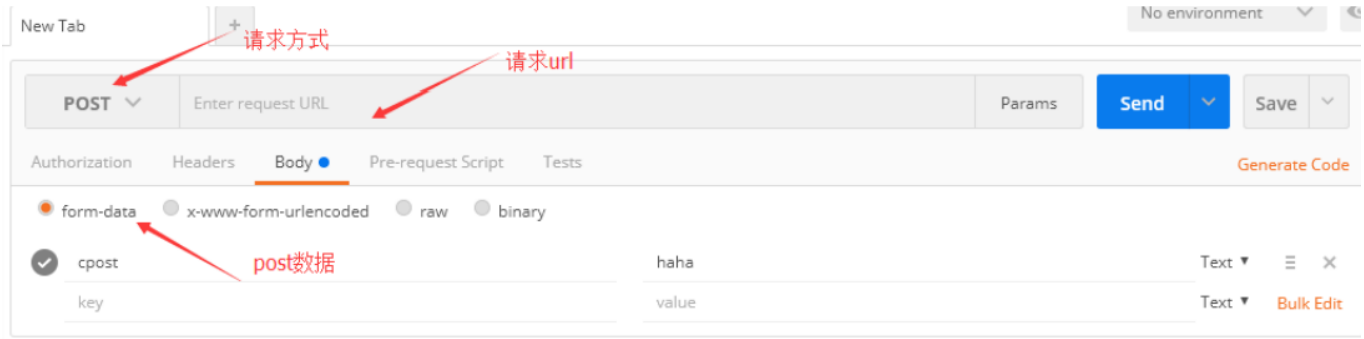
2.postman插件

建议使用postman软件，

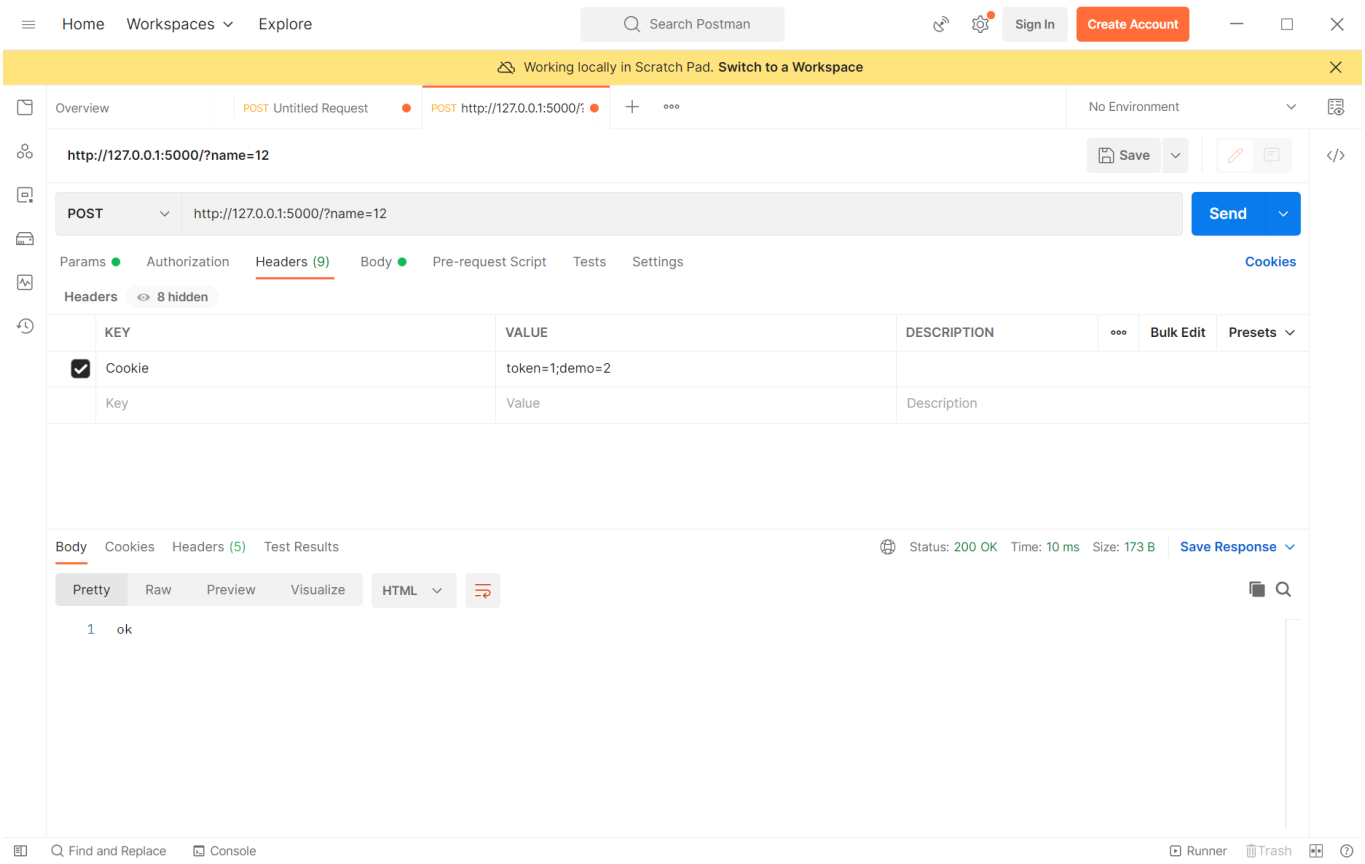
谷歌浏览器加载扩展程序

将解压好的扩展程序加载进去启动，第一次启动会要求填写一些内容，直接将窗口关闭，第二次启动就不需要填写了。

利用这个插件了以模拟浏览器以不同的请求方式向服务器提交参数。




```
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask, request

# Flask 接收一个参数 name ,
# 导入模块的目录， flask以这个目录为基础，寻找静态文件目录static和模板目录templates
app = Flask(__name__)


@app.route('/',methods=['GET','POST'])
def index():
    # form 获取post提交的数据   from flask import request
    post_dict = request.form
    print(post_dict)

    # args 获取查询字符串，即url ？ 后面的参数
    query_dict = request.args
    print(query_dict)

    # 获取所有查询参数，跟post参数
    all_dict = request.values
    print(all_dict)

    # 获取cookie
    cookie = request.cookies
    print(cookie)
    # 获取某个参数值get方法,如果不存在这个key会报错，为了不报错可以传一个默认值，如果不存在key，将使用默认值。
    # 如果是多值使用getlist方法,如果不存在，则返回空列表。
    a = query_dict.get('a', '')
    b = query_dict.getlist('b')
    print(a)
    return 'ok'


if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```



3.文件上传

已上传的文件存储在内存或是文件系统中一个临时的位置。

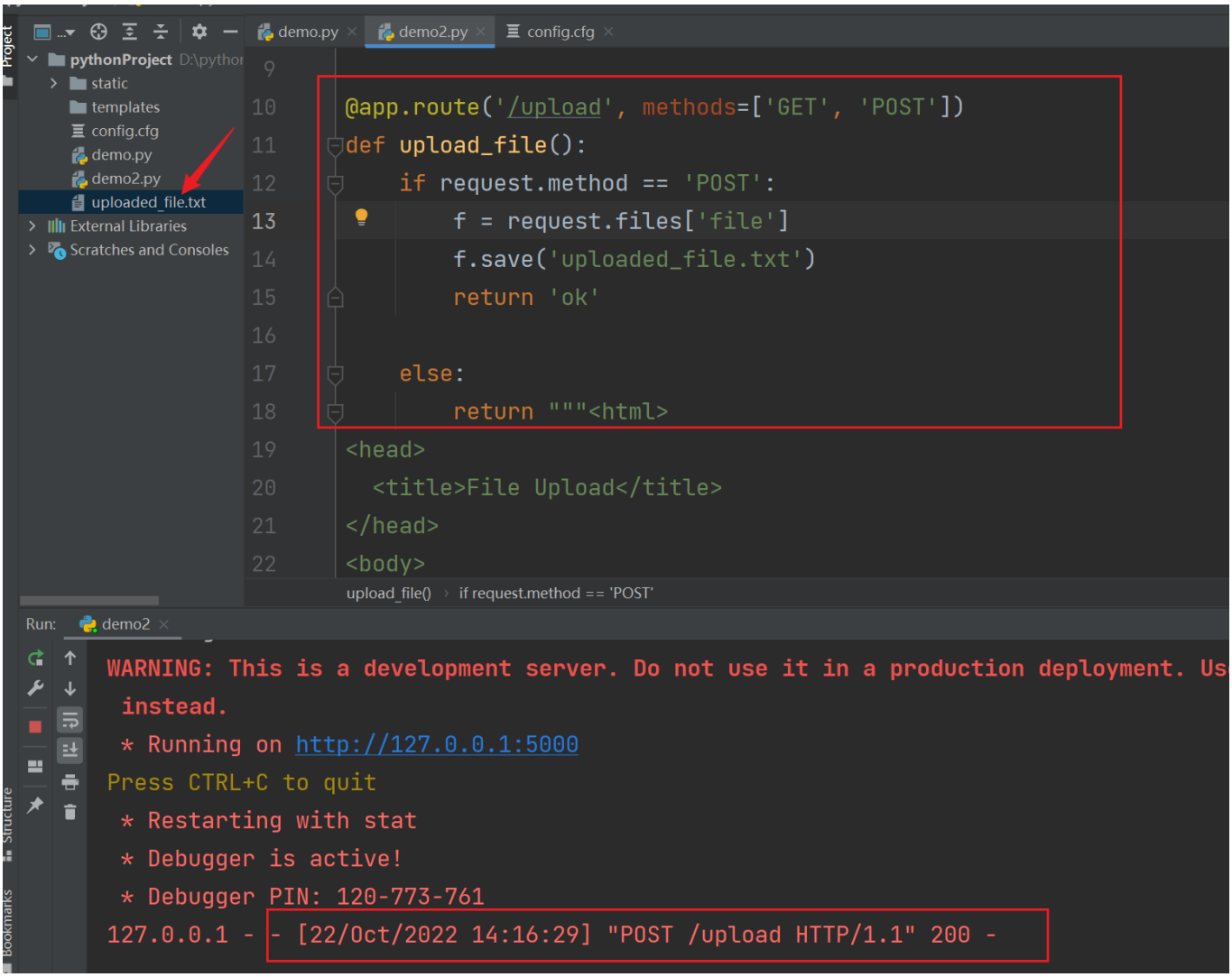
你可以通过请求对象的 `files` 属性访问它们。

每个上传的文件都会存储在这个字典里。

它表现近乎为一个标准的 `Python file` 对象，但它还有一个 `save()` 方法，这个方法允许你把文件保存到服务器的文件系统上。

```
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask, request

# Flask 接收一个参数 name ，
# 导入模块的目录， flask以这个目录为基础，寻找静态文件目录static和模板目录templates
app = Flask(__name__)


@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['file']
        f.save('uploaded_file.txt')
        return 'ok'

    else:
        return """<html>
```

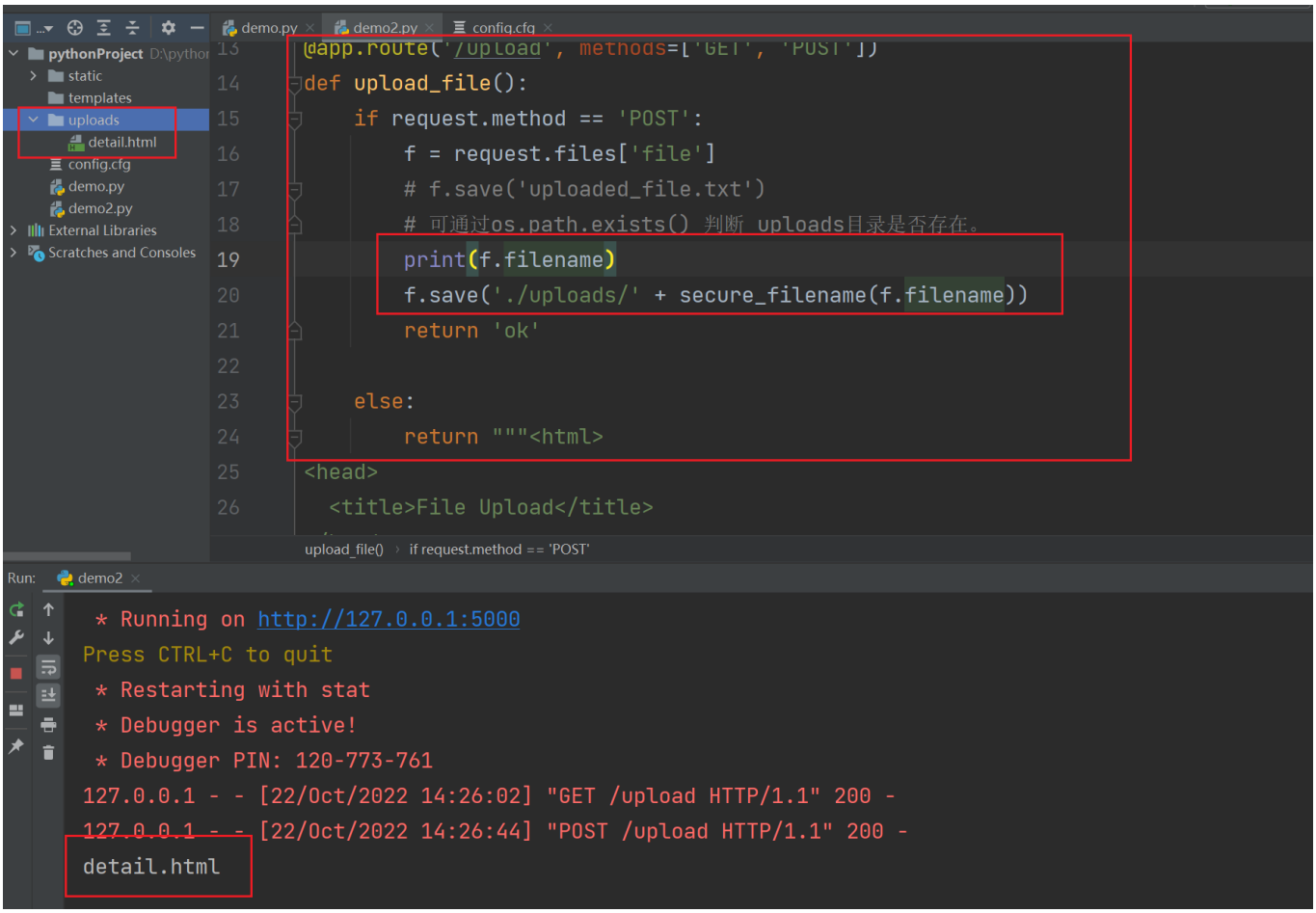
```
<head>
  <title>File Upload</title>
</head>
<body>
    <form action="http://127.0.0.1:5000/upload" method="POST" enctype="multipart/form-data">
        <input type="file" name="file"  />
        <input type="submit" value="提交" />
    </form>
</body>
</html>"""



if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```



如果你想知道上传前文件在客户端的文件名是什么，你可以访问 `filename` 属性。但请记住， 永远不要信任这个值，这个值是可以伪造的。

如果你要把文件按客户端提供的文件名存储在服务器上，那么请把它传递给 `Werkzeug`提供的 `secure_filename()` 函数：传递一个文件名，它会返回一个安全的文件名,防止上传的文件名不符合服务器文件命名要求。

```
# -*- encoding: utf-8 -*-
# 导入Flask类
import os

from flask import Flask, request
from werkzeug.utils import secure_filename

# Flask 接收一个参数 name ，
# 导入模块的目录， flask以这个目录为基础，寻找静态文件目录static和模板目录templates
app = Flask(__name__)


@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['file']
        # f.save('uploaded_file.txt')
        # 可通过os.path.exists() 判断 uploads目录是否存在。
        print(f.filename)
        f.save('./uploads/' + secure_filename(f.filename))
        return 'ok'

    else:
        return """<html>
<head>
  <title>File Upload</title>
</head>
<body>
    <form action="http://127.0.0.1:5000/upload" method="POST" enctype="multipart/form-data">
        <input type="file" name="file"  />
        <input type="submit" value="提交" />
    </form>
</body>
</html>"""



if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```



注意：secure_filename 不能识别中文。 secure_filename仅返回ASCII字符。所以， 非ASCII（比如汉字）会被过
滤掉，空格会被替换为下划线。你也可以自己处理文件名自动生成一个随机文件名，或是在使用这个函数前将中文

提示：
isalpha ( ) 判断是否全是字母
isdigit ( ) 判断是否全是数字
isalnum ( ) 是否全是数字或字母