Python - fake_useragent

目录

楔子

在爬虫中进行request请求，很多时候，都需要添加请求头，不然服务器会认为是非法的请求，从而拒绝你的访问。

```
1  import requests
2  url = 'https://www.zhihu.com/question/315387406/answer/812734512'
3  response = requests.get(url=url)
4  print(response.status_code)  # 400
```

在添加请求头中最常用的就是添加 `user-agent` 来讲本次请求伪装成浏览器。

> User Agent 中文名为用户代理，简称 UA，它是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。

那么个人怎么搞这个 `user-agent` 呢，没错？八仙过海各显神通，但一般都是用手来解决个人问题！

```
import requests

url = 'https://www.zhihu.com/question/315387406/answer/812734512'
headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36"
}
response = requests.get(url=url, headers=headers)
print(response.status_code)  # 200
```

但，自从有了 `fake_useragent` ，妈妈再也不用担心……

```
from fake_useragent import UserAgent
# 实例化 user-agent 对象
ua = UserAgent()

url = 'https://www.zhihu.com/question/315387406/answer/812734512'
headers = {"user-agent": ua.chrome}  # 指定浏览器 user-agent
# 或者可以这样写
# headers = {"user-agent": UserAgent().random}  # 一步到位，随机生成一个 user-agent
response = requests.get(url=url, headers=headers)
print(response.status_code)  # 200
```

About

什么是fake_useragent?

简单来说， `fake_useragent` 就像你的女朋友，能灵活的帮助我们生成 `user-agent` ，从而解放双手。

install

```
pip install fake_useragent
```

update

```
pip install -U fake-useragent
```

查看版本

```
import fake_useragent
print(fake_useragent.VERSION)  # 0.1.11
```

Usage

生成指定浏览器的user-agent

```
import fake_useragent

# 实例化 user-agent 对象
ua = fake_useragent.UserAgent()

# ua.ie
print(ua.ie)  # Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; chromeframe/13.0.782.215)

# ua.msie
```

```
print(ua['Internet Explorer'])  # Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; InfoPath.2; SLCC1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 2.0.50727)

# ua.opera
print(ua.opera)  # Opera/9.80 (Windows NT 6.1; U; en-US) Presto/2.7.62 Version/11.01

# ua.chrome
print(ua.chrome)  # Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.16 Safari/537.36

# ua.google
print(ua['google chrome'])  # Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2227.0 Safari/537.36

# ua.firefox
print(ua.firefox)  # Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:16.0.1) Gecko/20121011 Firefox/21.0.1

# ua.ff
print(ua.ff)  # Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101 Firefox/29.0

# ua.safari
print(ua.safari)  # Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW) AppleWebKit/533.19.4 (KHTML, like Gecko) Version/5.0.2 Safari/533.18.5

```

随机生成user-agent

```
import fake_useragent

# 实例化 user-agent 对象
ua = fake_useragent.UserAgent()
print(ua.random)  # Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.67 Safari/537.36
print(ua.random)  # Mozilla/5.0 (compatible; MSIE 10.0; Macintosh; Intel Mac OS X 10_7_3; Trident/6.0)
print(ua.random)  # Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36
```

每次都能随机生成一个UA表示，大大增强了爬虫的真实性。

其他用法

将远程user agent json文件下载到本地

由于 `fake_useragent` 库维护的user-agent json文件是在线的：

```
import fake_useragent
print(fake_useragent.settings.CACHE_SERVER)
'''
# 网址，其实是个json文件
https://fake-useragent.herokuapp.com/browsers/0.1.11
'''
```

既然是在线的json文件，那么我们就可将之下载到本地：

```
from fake_useragent import UserAgent, VERSION


```

```
    location = './fake_useragent%s.json' % fake_useragent.VERSION
4   ua = UserAgent(path=location)
```

如果报错 `fake_useragent.errors.FakeUserAgentError: Maximum amount of retries reached`，重新运行代码就好。

完事你就会发现在与脚本文件的同级目录有了一个json文件。

如果仅是想更新本地已保存的json文件

```
1   from fake_useragent import UserAgent
2   ua = UserAgent()
3   ua.update()
```

如果你不想缓存数据库或没有可写文件系统

```
1   from fake_useragent import UserAgent
2   ua = UserAgent(cache=False)
```

如果不想使用托管的缓存服务器

```
1   from fake_useragent import UserAgent
2   ua = UserAgent(use_cache_server=False)
```

处理异常

fake_useragent.errors.FakeUserAgentError: Maximum amount of retries

```
1   from fake_useragent import UserAgent
2   # 禁用服务器缓存：  use_cache_server=False
3   headers = {"User-Agent": UserAgent(use_cache_server=False).chrome}
```

```
response = requests.get(url=url, headers=headers)
print(response.status_code)  # 200
```

FakeUserAgentError('Maximum amount of retries reached'

```
from fake_useragent import UserAgent
# 法1 禁用服务器缓存： use_cache_server=False
headers = {"User-Agent": UserAgent(use_cache_server=False).chrome}
# 法2 忽略ssl验证
headers = {"User-Agent": UserAgent(verify_ssl=False).chrome}
# 法3 不缓存数据
headers = {"User-Agent": UserAgent(cache=False).chrome}
response = requests.get(url=url, headers=headers)
print(response.status_code)  # 200
```

fake_useragent.errors.FakeUserAgentError: Maximum amount of retries reached

```
from fake_useragent import UserAgent, VERSION

location = './fake_useragent%s.json' % fake_useragent.VERSION
ua = UserAgent(path=location)
```

我在将在线的json文件写入到本地时，由 `urllib.error.URLError: <urlopen error timed out>` 引起的报错，重新运行就好，本地文件也下载完成。

欢迎斧正，that's all see also：[fake_useragent-github](https://github.com/hellysmile/fake-useragent) | [fake_user_agent-pypi]() | [Python爬虫小技巧之伪造随机的User-Agent]() | [Python 使用 fake-useragent 库时报错的解决方法]()

分类: 爬虫 , Python

3 0

posted @ 2019-09-16 09:29 听雨危楼 阅读(4328) 评论(0) 编辑 收藏 举报

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页