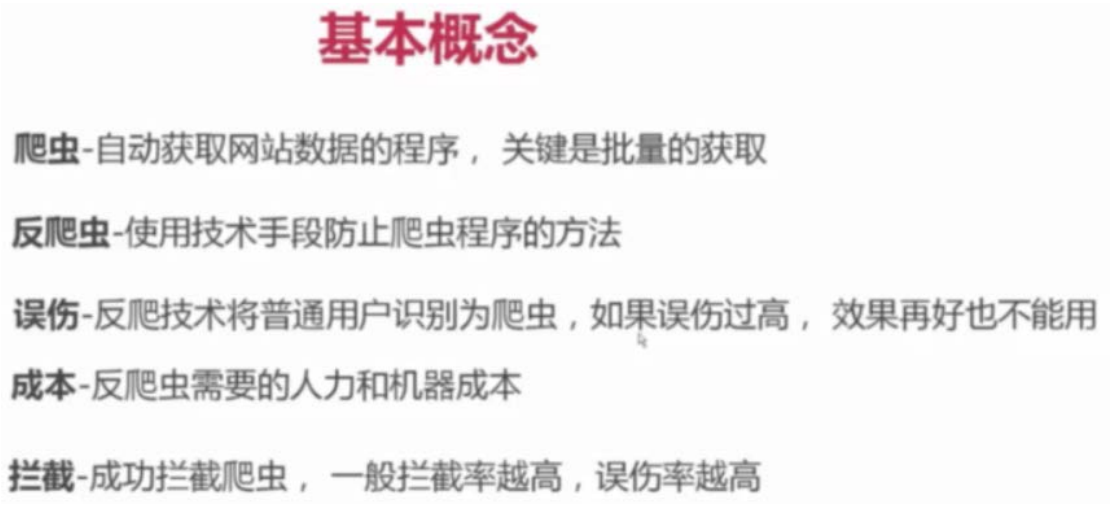
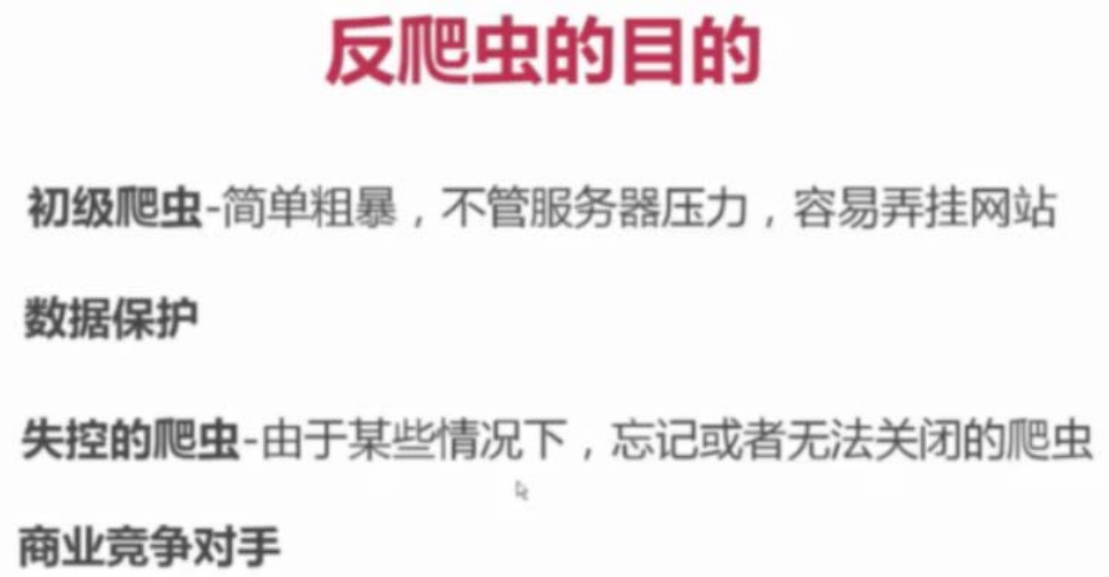
scrapy 突破反爬虫策略解析

爬虫与反爬虫基本概念



误伤：由于学校、网吧等等用的是同一个公网 ip，而内部使用局域网，所以如果封禁了此 ip，会导致大量用户的流失，同时还有动态 ip 分配的存在，所以在反爬虫中，封禁 ip 的策略一般网站不会使用，最多是封禁 ip 一小段时间。

反爬目的



突破反爬取的策略

1、随机切换用户代理 User-Agent

简单实现

维护一个包含很多 User-Agent 的列表，每次在 Request yield 时，使用 random 函数随机选一个 User-Agent 传入 Request header。但是这种方法冗余度高，不利于复用。

Middleware 实现

自定义一个 Downloader Middleware，可以做到每次请求时，拦截一下，给请求头自动随机更换 User-Agent。

fake-useragent

维护的 user-agent 列表存放在在线网页上，随机生成 UA

安装

pip install fake-useragent

用法

```
from fake_useragent import UserAgent
ua = UserAgent()

ua.ie
# Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US);
ua.msie
# Mozilla/5.0 (compatible; MSIE 10.0; Macintosh; Intel Mac OS X 10_7_3; Trident/6.0)'
ua['Internet Explorer']
# Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; GTB7.4; InfoPath.2; SV1; .NET CLR 3.3.69573; WOW64; en-US)
ua.opera
# Opera/9.80 (X11; Linux i686; U; ru) Presto/2.8.131 Version/11.11
ua.chrome
# Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like Gecko) Chrome/22.0.1216.0 Safari/537.2'
ua.google
# Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/537.13 (KHTML, like Gecko) Chrome/24.0.1290.1 Safari/537.13
ua['google chrome']
# Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11
ua.firefox
# Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:16.0.1) Gecko/20121011 Firefox/16.0.1
ua.ff
# Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:15.0) Gecko/20100101 Firefox/15.0.1
ua.safari
# Mozilla/5.0 (iPad; CPU OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5355d Safari/8536.25

# and the best one, random via real world browser usage statistic
ua.random
ua.update()
```

参考一下源码自带的用户代理 Downloader Middleware：

site-package/scrapy/downloadermiddlewares/useragent.py

```
"""Set User-Agent header per spider or use a default value from settings"""

from scrapy import signals
```

```
class UserAgentMiddleware(object):
    """This middleware allows spiders to override the user_agent"""

    def __init__(self, user_agent='Scrapy'):
        self.user_agent = user_agent

    @classmethod
    def from_crawler(cls, crawler):
        o = cls(crawler.settings['USER_AGENT'])
        crawler.signals.connect(o.spider_opened, signal=signals.spider_opened)
        return o

    def spider_opened(self, spider):
        self.user_agent = getattr(spider, 'user_agent', self.user_agent)

    def process_request(self, request, spider):
        if self.user_agent:
            request.headers.setdefault(b'User-Agent', self.user_agent)
```

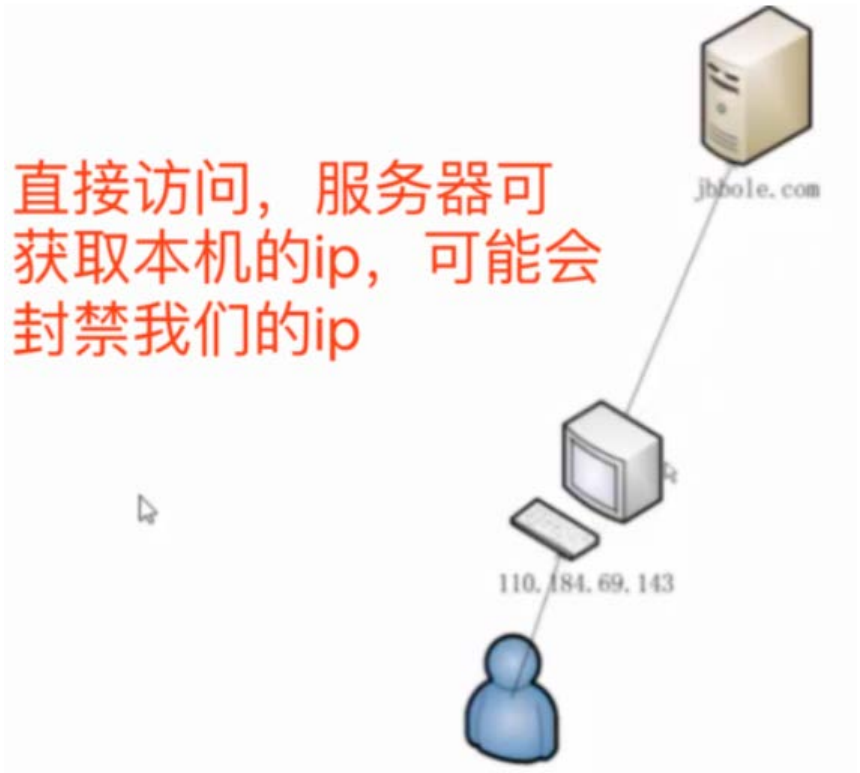
模仿写一个我们自己的随机更换的 Downloader Middleware，middlewares.py 中加入：

```
from fake_useragent import UserAgent

class UserAgentMiddleware(object):
    def __init__(self, user_agent=''):
        self.ua = UserAgent(verify_ssl=False)

    def process_request(self, request, spider):
        print('===UserAgentMiddleware process_request==')
        if self.ua:
            # 显示当前使用的 useragent
            print("********Current UserAgent:%s************")
            print(self.ua.random)
            request.headers.setdefault('User-Agent', self.ua.random)
```

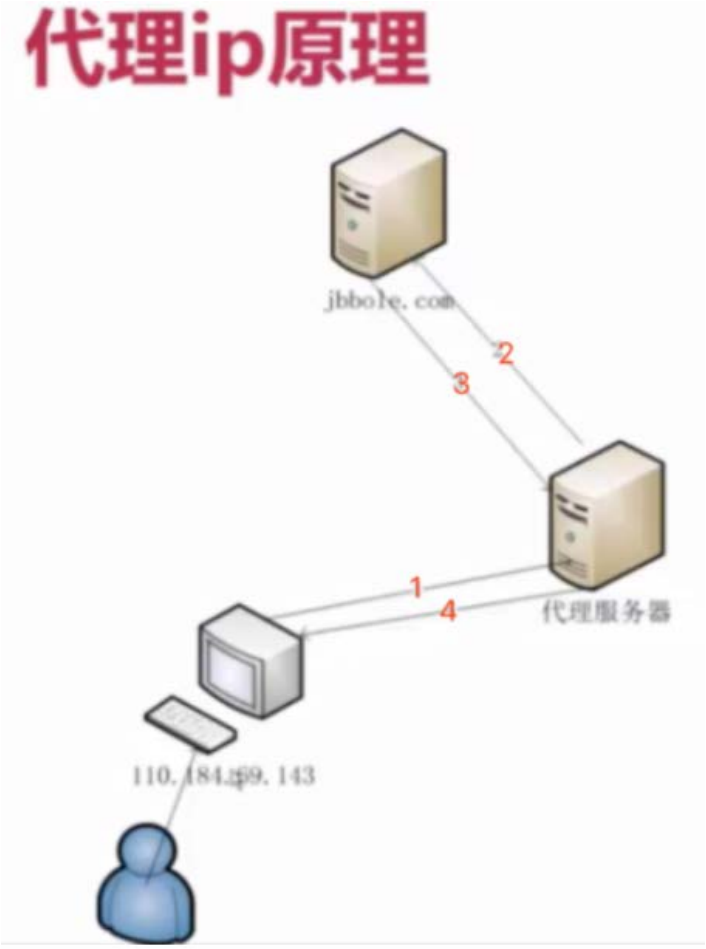
2 、随机更换代理 ip 策略

目前，很多小区、公共 wifi 的 ip 是动态分配的，当重启路由器、光猫后，本机的对外 ip 可能会发生改变。所以，若本机 ip 被封，可重启以求换 ip。对外 ip 可直接百度查看：

直接访问网站



使用 ip 代理之后访问网站，可避免本机的 ip 暴露：



设置 ip 代理

request.meta['proxy'] = 'xxx.xx.xx.xxx:xx'

为了获取代理 ip，我们可以前往西刺网获取免费的来试用或者付费购买一批可用的私密代理 IP：

```
PROXIES = [
    {'ip_port': '111.8.60.9:8123', 'user_passwd': 'user1:pass1'},
    {'ip_port': '101.71.27.120:80', 'user_passwd': 'user2:pass2'},
    {'ip_port': '122.96.59.104:80', 'user_passwd': 'user3:pass3'},
    {'ip_port': '122.224.249.122:8088', 'user_passwd': 'user4:pass4'},
]
```

高匿代理：能将我们的本机 ip 完全隐藏，普通代理可能还是会将本机 ip 带给服务器

```
class RandomProxy(object):
    def process_request(self, request, spider):
        proxy = random.choice(PROXIES)

        if proxy['user_passwd'] is None:
            # 没有代理账户验证的代理使用方式
            request.meta['proxy'] = "http://" + proxy['ip_port']
        else:
            # 对账户密码进行 base64 编码转换
            base64_userpasswd = base64.b64encode(proxy['user_passwd'])
            # 对应到代理服务器的信令格式里
            request.headers['Proxy-Authorization'] = 'Basic ' +
base64_userpasswd
            request.meta['proxy'] = "http://" + proxy['ip_port']
```

3、动态网页的爬取

```
class WebDriverMiddleware(object):

    @classmethod
    def from_crawler(cls, crawler):
        s = cls()
        crawler.signals.connect(s.spider_opened, signal=signals.spider_opened)
        return s

    def process_request(self, request, spider):
        # 加载驱动
        print('================process_request================')
        browser = webdriver.PhantomJS()
        browser.get(request.url)  # 加载网页
        data = browser.page_source  # 获取网页文本
        data = data.encode('utf-8')
        browser.quit()
        return HtmlResponse(request.url, body=data, encoding='utf-8', request=request)

    def process_response(self, request, response, spider):
        return response
```

```
    def process_exception(self, request, exception, spider):
        pass

    def spider_opened(self, spider):
        spider.logger.info('Spider opened: %s' % spider.name)
```

DOWNLOADER_MIDDLEWARES 设置

最后设置 setting.py 里的 DOWNLOADER_MIDDLEWARES，添加自己编写的下载中间件类。

```
DOWNLOADER_MIDDLEWARES = {
    'mySpider.middlewares. UserAgentMiddleware: 1,
    'mySpider.middlewares. RandomProxy: 100
}
```

4、禁用 cookies

特殊情况下防止某些网站根据 Cookie 来封锁爬虫。

```
COOKIES_ENABLED = False
```

5、设置下载延迟

1. *# 下载延迟单位秒，下载器在下载同一个网站下一个页面前需要等待的时间。该选项可以*
   用来限制爬取速度， 减轻服务器压力。同时也支持小数:
2. *# 该设定影响(默认启用的) RANDOMIZE_DOWNLOAD_DELAY 设定。 默认情况下，S*
   *crapy 在两个请求间不等待一个固定的值， 而是使用 0.5 到 1.5 之间的一个随机值 * DO*
   WNLOAD_DELAY 的结果作为等待间隔。
3.

```
DOWNLOAD_DELAY = 3
```

6、 scrapy 模拟登录

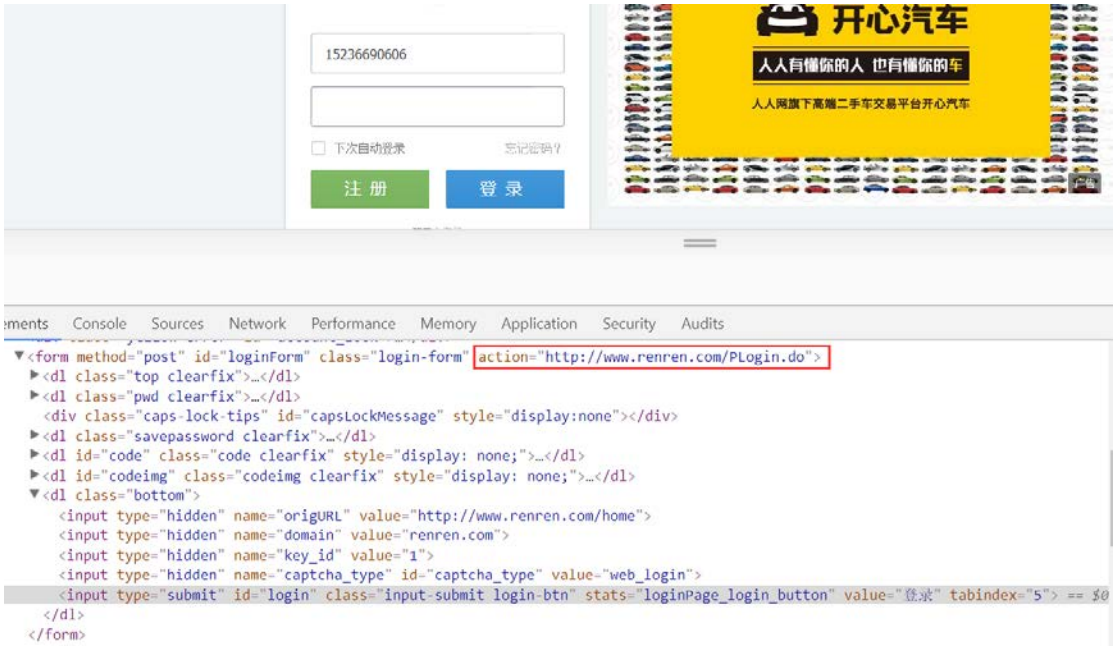
Scrapy.FormRequest 方法

Scrapy 发送 POST 请求的方法

参照 scrapy 官方文档的标准写法是：

```
# header 信息
unicornHeader = {
    'Host': 'www.example.com',
    'Referer': 'http://www.example.com/',
}

# 表单需要提交的数据
myFormData = {'name': 'John Doe', 'age': '27'}

# 自定义信息，向下层响应(response)传递下去
customerData = {'key1': 'value1', 'key2': 'value2'}

yield scrapy.FormRequest(url = "http://www.example.com/post/action",
                         headers = unicornHeader,
                         method = 'POST',                    # GET or POST
                         formdata = myFormData,              # 表单提交的数据
                         meta = customerData,                # 自定义，向 response 传递数据
                         callback = self.after_post,
                         errback = self.error_handle,
                         # 如果需要多次提交表单，且 url 一样，那么就必须加此参数 dont_filter，防止被当成重复网页过滤掉了
                         dont_filter = True
                         )
```

人人网登录网页检查



策略一：直接 POST 数据（需要登陆的账户信息）

1、创建 scrapy 项目

scrapy startproject LoginSpider

2、创建爬虫

scrapy genspider renren1 renren.com

3、编写爬虫

```
import scrapy


class Renren1Spider(scrapy.Spider):
    name = 'renren1'
    allowed_domains = ['renren.com']
```

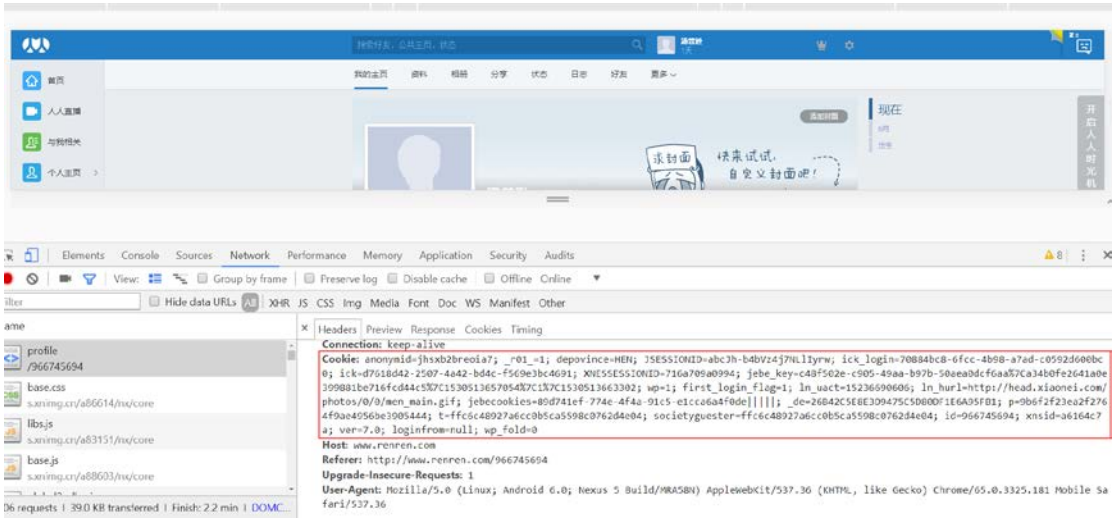
```
    start_urls = ['http://renren.com/']

    def start_requests(self):
        url = 'http://www.renren.com/PLogin.do'
        # FormRequest 是 Scrapy 发送 POST 请求的方法
        yield scrapy.FormRequest(
                url = url,
                formdata = {"email" : "mr_mao_hacker@163.com
", "password" : "axxxxxxxe"},
                callback = self.parse_page)

    def parse_page(self, response):
        print(response.body.decode('utf-8'))
        with open("mao2.html", "w",encoding='utf-8') as filename:
            filename.write(response.body.decode('utf-8'))
```

策略二：直接使用保存登陆状态的 Cookie 模拟登陆

如果实在没办法了，可以用这种方法模拟登录，虽然麻烦一点，但是成功率 100%

网页审查

登录人人网，查找 cookie 信息



爬虫代码

```
import scrapy


class Renren3Spider(scrapy.Spider):
    name = 'renren3'
    allowed_domains = ['renren.com']
    start_urls = ['http://www.renren.com/966745694/profile']

    cookies = {
        "anonymid": "jhsxb2breoia7",
        "_r01_": "1",
        "depovince": "HEN",
        "JSESSIONID": "abcJh-b4bVz4j7NLlIyrw",
        "ick_login": "70884bc8-6fcc-4b98-a7ad-c0592d600bc0",
        "ick": "d7618d42-2507-4a42-bd4c-f569e3bc4691",
        "XNESSESSIONID": "716a709a0994",
        "jebe_key":
"c48f502e-c905-49aa-b97b-50aea0dcf6aa%7Ca34b0fe2641a0e399881be716fcd44c5%7C153051
3657054%7C1%7C1530513663302",
        "wp": "1",
        "first_login_flag": "1",
        "ln_uact": "15236690606",
        "ln_hurl": "http://head.xiaonei.com/photos/0/0/men_main.gif",
        "jebecookies": "89d741ef-774e-4f4a-91c5-e1cca6a4f0de|||||",
        "_de": "26B42C5E8E3D9475C5D80DF1E6A95FB1",
        "p": "9b6f2f23ea2f2764f9ae4956be3905444",
        "t": "ffc6c48927a6cc0b5ca5598c0762d4e04",
        "societyguester": "ffc6c48927a6cc0b5ca5598c0762d4e04",
        "id": "966745694",
        "xnsid": "a6164c7a",
        "ver": "7.0",
        "loginfrom": "null",
        "wp_fold": "0"
    }

    # 可以重写 Spider 类的 start_requests 方法，附带 Cookie 值，发送 POST 请求
    def start_requests(self):
        for url in self.start_urls:
            yield scrapy.FormRequest(url, cookies = self.cookies, callback = self.parse_page)

    # 处理响应内容
```

```
    def parse_page(self, response):
        print("===========" + response.url)
        with open("deng.html", "w",encoding='utf-8') as filename:
            filename.write(response.body.decode('utf-8'))
```

小结

防止爬虫被反主要有以下几个策略：

动态设置 User-Agent（随机切换 User-Agent，模拟不同用户的浏览器信息）

禁用 Cookies，有些网站通过 cookie 的使用发现爬虫行为

设置延迟下载（防止访问过于频繁，设置为 2 秒 或更高

使用 IP 地址池：VPN 和代理 IP，现在大部分网站都是根据 IP 来 ban 的。

附录：Settings

Scrapy 设置(settings)提供了定制 Scrapy 组件的方法。

内置设置参考手册

- `BOT_NAME`

  默认: 'scrapybot'

  当您使用 startproject 命令创建项目时其也被自动赋值。

- `CONCURRENT_ITEMS`

  默认: 100

  Item Processor(即 Item Pipeline) 同时处理(每个 response 的)item 的最大值。

- `CONCURRENT_REQUESTS`

默认: 16

Scrapy downloader 并发请求(concurrent requests)的最大值。

- **DEFAULT_REQUEST_HEADERS**

  默认: 如下

  {

  'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',

  'Accept-Language': 'en',

  }

  Scrapy HTTP Request 使用的默认 header。

- DEPTH_LIMIT

  默认: 0

  爬取网站最大允许的深度(depth)值。如果为 0，则没有限制。

- **DOWNLOAD_DELAY**

  默认: 0

  下载器在下载同一个网站下一个页面前需要等待的时间。该选项可以用来限制爬取速度，减轻服务器压力。同时也支持小数:

  ```
  DOWNLOAD_DELAY = 0.25 # 250 ms of delay
  ```

  默认情况下，Scrapy 在两个请求间不等待一个固定的值， 而是使用 0.5 到 1.5 之间的一个随机值 * DOWNLOAD_DELAY 的结果作为等待间隔。

- DOWNLOAD_TIMEOUT

默认: 180

下载器超时时间(单位: 秒)。

- `ITEM_PIPELINES`

  默认: {}

  保存项目中启用的 pipeline 及其顺序的字典。该字典默认为空，值(value)任意，不过值(value)习惯设置在 0-1000 范围内，值越小优先级越高。

```
ITEM_PIPELINES = {
'mySpider.pipelines.SomethingPipeline': 300,
'mySpider.pipelines.ItcastJsonPipeline': 800,
}
```

- `LOG_ENABLED`

  默认: True

  是否启用 logging。

- `LOG_ENCODING`

  默认: 'utf-8'

  logging 使用的编码。

- `LOG_LEVEL`

  默认: 'DEBUG'

  log 的最低级别。可选的级别有: CRITICAL、 ERROR、WARNING、INFO、DEBUG 。

- `USER_AGENT`

  默认: "Scrapy/VERSION (+http://scrapy.org)"

爬取的默认 User-Agent，除非被覆盖。

- `COOKIES_ENABLED = False`

  禁用 Cookies