Scrapy实战案例——爬取豆瓣读书

需求

爬取豆瓣读书，并提取后续链接加入待爬取队列

链接分析

```
第一页 https://book.douban.com/tag/%E7%BC%96%E7%A8%8B?start=0&type=T
第二页 https://book.douban.com/tag/%E7%BC%96%E7%A8%8B?start=20&type=T
```

start的值一直在变化

项目开发

创建项目

```
$ scrapy startproject mageduspider . # 在已有python项目中构建爬虫

USER_AGENT = "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/55.0.2883.75 Safari/537.36"
```

配置settings.py

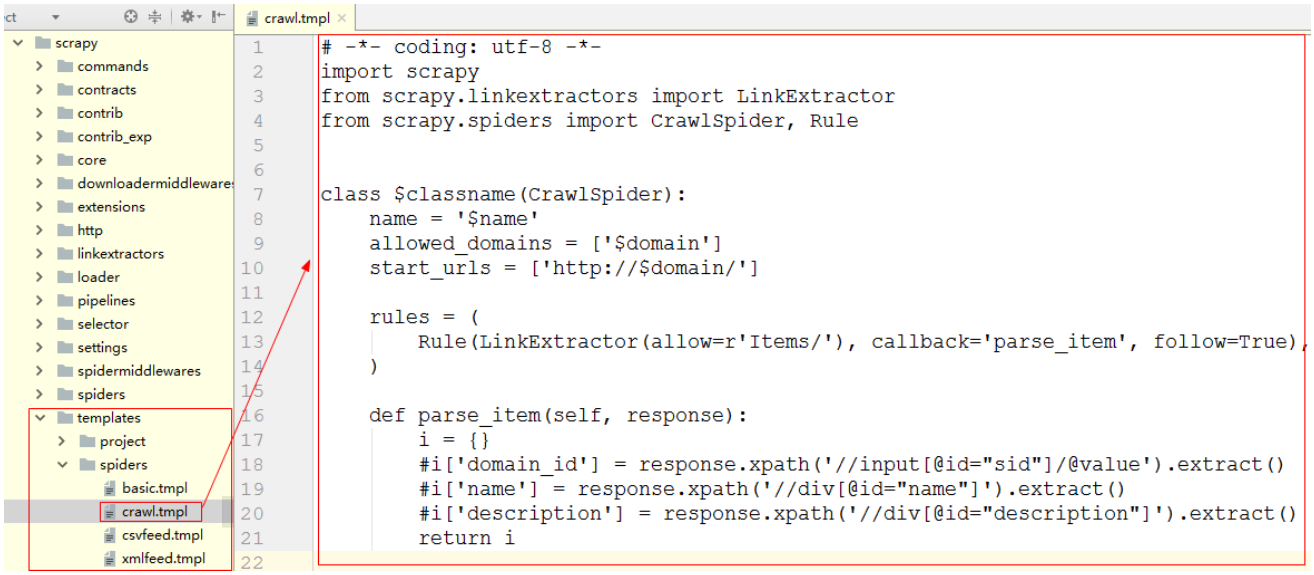
```
BOT_NAME = 'mageduspider'

SPIDER_MODULES = ['mageduspider.spiders']
NEWSPIDER_MODULE = 'mageduspider.spiders'

USER_AGENT = "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/55.0.2883.75 Safari/537.36"

ROBOTSTXT_OBEY = False
COOKIES_ENABLED = False
```

创建爬虫

`scrapy genspider [-t template] <name> <domain>`

模板

-t 模板，这个选项可以使用一个模板来创建爬虫类，常用模板有basic、crawl。



使用模板创建爬虫`scrapy genspider -t crawl book douban.com`，得到如下代码

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule


class BookSpider(CrawlSpider):
    name = 'book'
    allowed_domains = ['douban.com']
    start_urls = ['http://douban.com/']

    rules = (
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item', follow=True),
    )

    def parse_item(self, response):
        i = {}
        #i['domain_id'] = response.xpath('//input[@id="sid"]/@value').extract()
        #i['name'] = response.xpath('//div[@id="name"]').extract()
        #i['description'] = response.xpath('//div[@id="description"]').extract()
        return i
```

scrapy.spiders.crawl.CrawlSpider 是 scrapy.spiders.Spider的子类，增强了功能。在其中可以使用LinkExtractor、Rule。

规则Rule定义

- rules元组里面定义多条规则Rule，用规则来方便的跟进链接
- LinkExtractor从response中提取链接
  - allow需要一个对象或可迭代对象，其中配置正则表达式，表示匹配什么链接，即只关心`<a>`标签
- callback 定义匹配链接后执行的回调，特别注意不要使用parse这个名称。返回一个包含Item或Request对象的列表
  - 参考`scrapy.spiders.crawl.CrawlSpider#_parse_response`

- follow是否跟进链接

由此得到一个本例程的规则，如下

```
class BookSpider(CrawlSpider):
    name = 'book'
    allowed_domains = ['douban.com']
    # 起点
    start_urls = ['https://book.douban.com/tag/%E7%BC%96%E7%A8%8B']

    rules = (
        Rule(LinkExtractor(allow=r'start=\d+'), callback='parse_pagination', follow=False),
    ) # follow为False将不跟进

    def parse_pagination(self, response:HtmlResponse):
        print(response.url)  # scrapy crawl book > u.txt
        for subject in response.xpath('//li[@class="subject-item"]'):
            item = BookItem()
            item['title'] = subject.xpath('.//h2/a/text()')[0].extract().strip()
            rate = subject.xpath('.//span[@class="rating_nums"]/text()').extract()
            if rate: # 有可能没有评分
                item['rate'] = rate[0]
            else:
                item['rate'] = '0'
            yield item
```

上面的代码，爬虫在运行时，会分析页面内的链接，提取到匹配链接就会执行回调函数parse_pagination。回调中response就是提取到的链接的页面请求返回的HTML，直接对这个HTML使用xpath或css分析即可。

编写item

```
import scrapy

class BookItem(scrapy.Item):
    title = scrapy.Field()  # 书名
    rate = scrapy.Field()  # 分数

    def __repr__(self):
        return '<{} {}>'.format(self.__class__.__name__, dict(self))
```

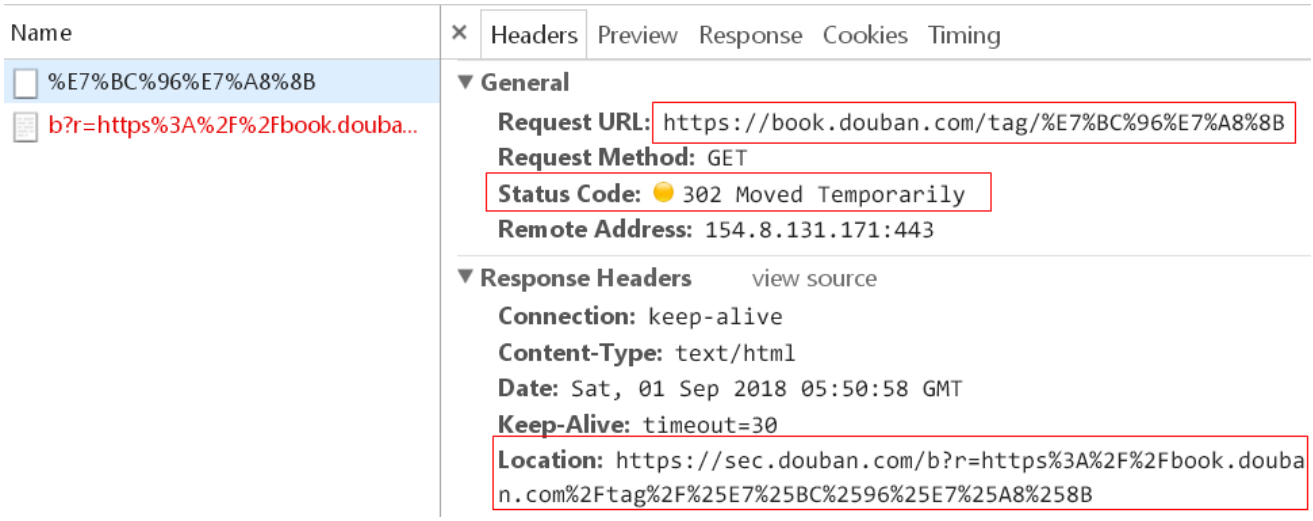
爬取

`scrapy crawl book`

将follow改为True试一试

代理

在爬取过程中，豆瓣使用了反爬策略，可能会出现以下现象



这相当于封了IP，所以，可以在爬取时使用代理来解决。

思路：在发起HTTP请求之前，会经过下载中间件，自定义一个下载中间件，在其中临时获取一个代理地址，然后再发起HTTP请求

从`http://www.xicidaili.com/`代理上找到免费代理，测试通过后，可以加入到代码中。

1、下载中间件

仿照middlewares.py中的下载中间件写，编写process_request，返回None。

```
from scrapy.http.request import Request
import random

class ProxyDownloaderMiddleware(object):
    # 增加代理 IP池，可以从网络搜罗，可以从配置文件中读取
    proxy_ip = "36.59.203.93"  # 代理ip，通过http://h.wandouip.com/get获得
    proxy_port = "894"  # 代理端口号
    proxies = [
        'http://{}:{}'.format(proxy_ip, proxy_port)
    ]

    def process_request(self, request:Request, spider):
        request.meta['proxy'] = random.choice(self.proxies) # 增加proxy
        print(request.url, request.meta['proxy'])
        print('-'*30)
```

2、配置

在settings.py中

```
# 3秒
DOWNLOAD_DELAY = 3

DOWNLOADER_MIDDLEWARES = {
    'mageduspider.middlewares.ProxyDownloaderMiddleware': 125,
}
```

增加一个测试用spider类

如果使用代理设置成功，该爬虫返回的内容就会是当前代理的信息

```
class IPTestSpider(Spider):
    name = 'test'
    allowed_domains = ['ipip.net']
    # 起点
    start_urls = ['http://myip.ipip.net/']

    def parse(self, response:HtmlResponse):
        text = response.text
        print(text, '======================')
        return text
```

测试运行

`scrapy crawl test`

查看打印的内容，是不是代理的地址信息。如果代理测试成功，就可以继续下面的操作

pipeline

在settings.py中开启pipeline

```
ITEM_PIPELINES = {
   'mageduspider.pipelines.MageduspiderPipeline': 300,
}
```

将数据存入json文件

```
import json

class MageduspiderPipeline(object):
    def process_item(self, item, spider):
        # item 获取的item；spider 获取该item的spider
        self.jsonfile.write(json.dumps(dict(item)) + ',\n')
        return item  # 向后处理

    def open_spider(self, spider):
        filename = 'o:/books.json'
        self.jsonfile = open(filename, 'w')
        self.jsonfile.write('[\n')

    def close_spider(self, spider):
        if self.jsonfile:
            self.jsonfile.write(']')
            self.jsonfile.close()
```