LinkExtractor

```
from scrapy.linkextractors import linkExtractor as LE
le=LE(restrict_xpaths='//div[@class="pg"]',tags='a',attrs='href',allow='bbs\.vivo\.com\.cn/thread-\d+-\d+-\d+\.html')
for _ in le.extract_links(response):
    yield Request(_.url,callback=self.parse)
# 这样写更健壮,不容易断
```

scrapy_redis

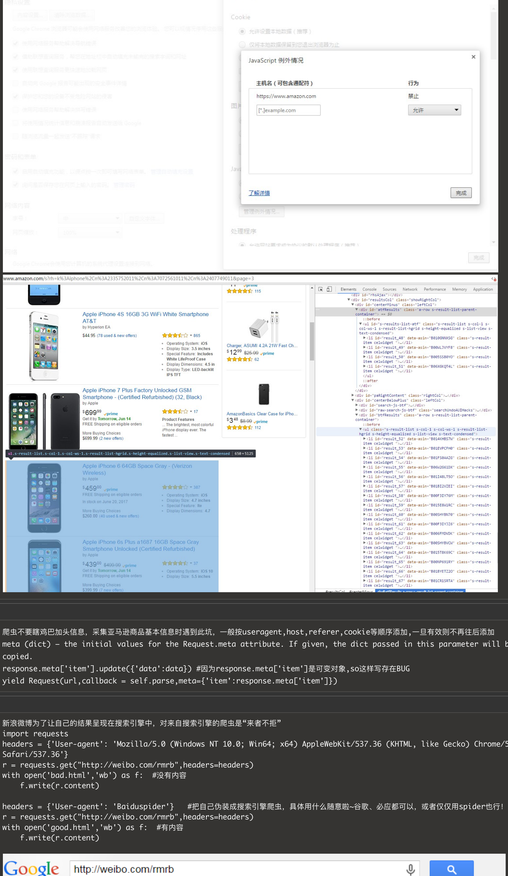
```
spiders.py和connection.py实现分布式（手动推送去抽取）
pipelines.py和connection.py实现把数据存储到redis数据库
scheduler.py，dupefilter.py，queue.py和connection.py实现断点续爬,分布式

要想实现分布式，只需将该spider复制一份，然后再运行即可
item_key(self, item, spider)此处可以自定义item_key
SCHEDULER_PERSIST = True，中断程序，dmoz:dupefilter和dmoz:requests会存在，程序执行完后还会保存dmoz:dupefilter（此时已不存在
dmoz:requests）
SCHEDULER_PERSIST = False，中断程序，dmoz:dupefilter和dmoz:requests会被清空，程序执行完后会删除dmoz:dupefilter（此时已不存在
dmoz:requests）
type dmoz:dupefilter返回set
type dmoz:requests返回zset
```

采集纸:

```
import requests
r=requests.get(
  url='https://tsvideo.oss-cn-hangzhou.aliyuncs.com/崔庆才Python三大案例/405003.mp4',
  headers={
    'Accept-Encoding':'identity;q=1, *;q=0',
    'Range':'bytes=0-',
    'Referer':'https://edu.hellobi.com/course/157/play/lesson/2575',
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115
Safari/537.36',
  },
  stream=True,
)
with open('16.mp4','wb') as f:      # 注意头信息Range,必须要从0开始,对于大文件下载一般要stream=True
  for chunk in r.iter_content(1024 * 100):
    f.write(chunk)
print(r.status_code)
```

```
import requests
import re
import json
r=requests.get('http://commoncgi.wepiao.com/data/v5/movies/cities/227/movies_city_227.json')
_r=r.content.decode('unicode_escape').replace('\"','"')   #注意
_=re.search(r'"info":(.+?),"announcement":',_r.S).group(1)
print(_)
```

亚马逊基本信息页采集ASIN时xpath('//*[@id="atfResults"]/ul/li/@data-asin')会出现采集不全（只有前4个）的情况
是应为某些标签被JS更改了，而爬虫只能获取未被JS处理的网页源代码
解决办法是Chrome设置->禁用亚马逊执行JS，然后再F12观察其源码结构



爬虫不要瞎鸡巴加头信息，采集亚马逊商品基本信息时遇到此坑，一般按useragent,host,referer,cookie等顺序添加,一旦有效则不再往后添加
meta (dict) – the initial values for the Request.meta attribute. If given, the dict passed in this parameter will be shallow copied.
response.meta['item'].update({'data':data}) #因为response.meta['item']是可变对象,so这样写存在BUG
yield Request(url,callback = self.parse,meta={'item':response.meta['item']})

新浪微博为了让自己的结果呈现在搜索引擎中，对来自搜索引擎的爬虫是"来者不拒"

```
import requests
headers = {'User-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36'}
r = requests.get("http://weibo.com/rmrb",headers=headers)
with open('bad.html','wb') as f:  #没有内容
    f.write(r.content)

headers = {'User-agent': 'Baiduspider'}  #把自己伪装成搜索引擎爬虫，具体用什么随意啦~谷歌、必应都可以，或者仅仅用spider也行!
r = requests.get("http://weibo.com/rmrb",headers=headers)
with open('good.html','wb') as f:  #有内容
    f.write(r.content)
```


UserAgent
有些网站必须把头信息设置成手机头信息,eg: https://www.amazon.com/gp/aw/d/B00YD53YQU

Noscript标签的使用
<noscript>标签是在浏览器（或者用户浏览标识），没有启动脚本支持的情况下触发的标签，在低级爬虫中，基本都没有配置js引擎，通常这种方式和Ajax异步加载同时使用。用于保护自己不想让爬虫接触的信息。

天眼查绕过登录拿页面数据

- 访问https://m.tianyancha.com/company/2323965264
- 访问https://m.baidu.com/s?word=广州奥利门窗有限公司 site:www.tianyancha.com，获取到公司在企查查对应地址url及Referer信息，再访问

```
import requests,re
from lxml.html import fromstring
headers = {
    'Referer':'https://m.baidu.com/from=0/bd_page_type=1/ssid=0/uid=0',
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.170
Safari/537.36',
}

r=requests.get('https://m.baidu.com/s?word=广州奥利门窗有限公司 site:www.tianyancha.com')
html=fromstring(r.content)
data=html.xpath('//*[@id="results"]/div[1]/@data-log')[0]
url=re.findall(r'"https://www\.tianyancha\.com/company/\d+',data)[0]

r=requests.get(url,headers=headers)  # 头信息中的Referer非常重要
print(r.text)
```

Lazada
必须要加cookie,不然每次请求都只能拿每个品类的第一页数据
不能并行查看品类信息，不然每次请求都只能拿每个品类的第一页数据，只能一个品类一个品类查看
验证如下：

```
import requests,re,pymysql,time,random
headers = {
    'accept': '*/*',
    'content-type': 'application/json',
    'accept-encoding': 'gzip, deflate, br',
    'accept-language': 'en-US,en;q=0.9',
    'cookie':'t_uid=c4929da6-ca29-4143-83e9-e996fcac4189;',
    'user-agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117
Safari/537.36',
}
for page in range(1,3): # OK
    url='https://www.lazada.co.id/beli-handphone/?ajax=true&page={page}'
    r=requests.get(url,headers=headers,timeout=18)
    print(r.json()['mods']['listItems'])
for page in range(1,3):
    url='https://www.lazada.co.id/beli-wanita-emas-murni/?ajax=true&page={page}'
    r=requests.get(url,headers=headers,timeout=18)
    print(r.json()['mods']['listItems'])

for page in range(1,3): # Error
    url=f'https://www.lazada.co.id/beli-handphone/?ajax=true&page={page}'
    r=requests.get(url,headers=headers,timeout=18)
    print(r.json()['mods']['listItems'])
    url=f'https://www.lazada.co.id/beli-wanita-emas-murni/?ajax=true&page={page}'
    r=requests.get(url,headers=headers,timeout=18)
    print(r.json()['mods']['listItems'])
```

也可以用浏览器分别打开https://www.lazada.co.id/beli-handphone/?page=1和https://www.lazada.co.id/beli-wanita-emas-murni/?page=1进行验证

```
import requests
res = requests.get("https://114.1688.com/sitemap.html?")
res.encoding = 'gb2312'
print(res.text)
```