CrawlSpider--自动爬取网页

使用 *CrawlSpider* 类，使用简单方式实现多网页自动爬取。

CrawlSpider 是 Spider 的派生类，首先在说下 Spider，它是所有爬虫的基类

Spider 设计原则是只爬取 start_url 列表中的网页

CrawlSpider 从爬取的网页中获取 link 并继续爬取的工作

rules 属性

它与 Spider 类的最大不同是多了一个 rules 参数，其作用是定义提取动作。

在 rules 中包含一个或多个 Rule 对象

Rule 类与 CrawlSpider 类都位于 scrapy.contrib.spiders 模块中。

```
class scrapy.spiders.Rule (
link_extractor, callback=None,cb_kwargs=None,follow=None,process_links=None,process_request=None )
```

其中：

link_extractor 参数：LinkExtractor，用于定义需要提取的链接。

callback 参数：回调函数，当 link_extractor 获取到链接时调用。

> **注意：**
> 由于 CrawlSpider 使用 parse 方法来实现其逻辑，如果覆盖了 parse 方法，crawlspider 将会运行失败。
> 当编写爬虫规则时，请避免使用 parse 作为回调函数。

follow 参数：指定了根据该规则从 response 提取的链接是否需要跟进。当 callback 为 None,默认值为 true。

process_links 参数：回调，用来过滤由 link_extractor 获取到的链接。

```
def deal_links(self, links):
    for link in links:
```

```
        print('link: ',link.url)
    return links
```

process_request 参数：主要用来过滤在 rule 中提取到的 request。

LinkExtractor

概念

顾名思义，链接提取器。

作用

response 对象中获取链接，并且该链接接下来会被爬取。

使用

提取希望获取的链接。

```
class scrapy.linkextractors.sgml.LinkExtractor(
allow=(),deny=(),allow_domains=(),deny_domains=(),deny_extensions=None,restrict_xpaths=(),tags=('a','area'),attrs=('href'),canonicalize=True,unique=True,process_value=None)
```

主要参数

allow：接收一个正则表达式或一个正则表达式列表，提取绝对 url 与正则表达式匹配的链接。如果该参数为空，默认全部提取。

deny：与正则表达式*(*或正则表达式列表*)*匹配的 *URL 被排除，*一定不提取。它优先于 allow 参数。如果没有给出（或为空），它不会排除任何链接。

allow_domains：会被提取的链接的 *domains*。

deny_domains：一定不会被提取链接的 *domains*。

restrict_xpaths：使用 *xpath* 表达式，和 *allow* 共同作用过滤链接。

restrict_css：一个 CSS 选择器（或选择器列表），用于定义响应中应提取链接的区域。

tags：接收一个标签（字符串）或一个标签列表，提取指定标签内的链接，默认为 tags=（‘a’，‘area’）

attrs：接收一个属性（字符串）或者一个属性列表，提取有指定的属性内的链接，默认为 attrs=（‘href’，）

范例：sina 博客爬取

1、创建 scrapy 项目

scrapy startproject blogCrawlSpider

2、items.py

```
class BlogcrawlspiderItem(scrapy.Item):
    blog_name = scrapy.Field()
    blog_url = scrapy.Field()
```

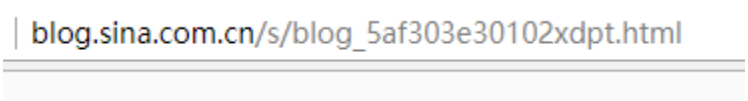
3、创建爬虫

scrapy genspider SinaBlog blog.sina.com

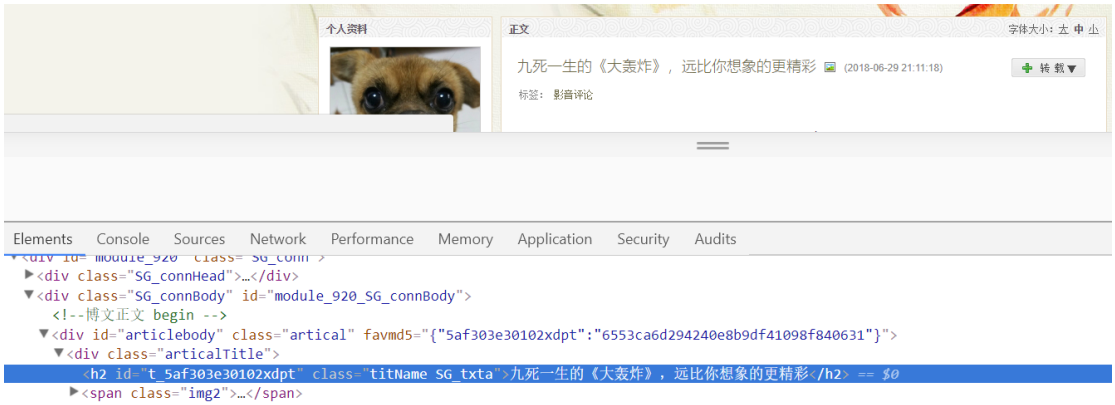
4、元素审查

起始 url

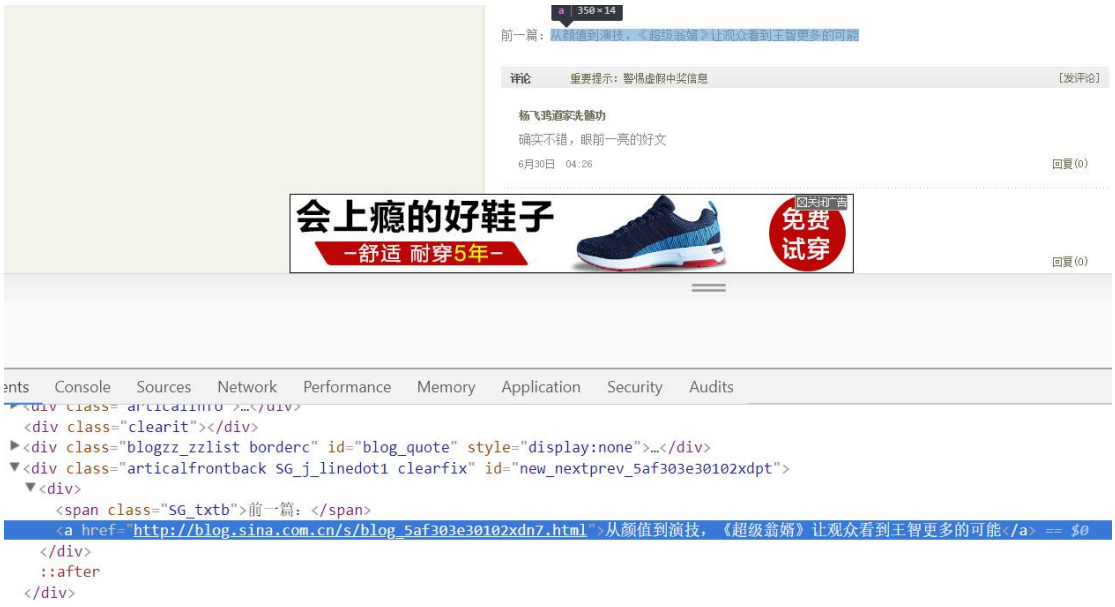
http://blog.sina.com.cn/s/blog_5af303e30102xdpt.html



标题：class="titName SG_txta"



前一篇 class="SG_txtb"



5、编写代码

items.py 和 pipelines.py 以及 settings.py 与 spider 类似，不详细描述。

爬虫编写。

```
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from scrapy.selector import Selector
from blogCrawlSpider.items import BlogcrawlspiderItem


class SinablogSpider(CrawlSpider):
    name = 'SinaBlog'
```

```
    # 设置下载延时
    download_delay = 2
    allowed_domains = ['blog.sina.com.cn']
    # 第一篇文章地址
    start_urls = ['http://blog.sina.com.cn/s/blog_5af303e30102xdpt.html']
    rules = [
        Rule(LinkExtractor(allow=('/s/blog_.*\.html'), restrict_xpaths =
('//div[@class="articalfrontback SG_j_linedot1
clearfix"]/div/a')),callback='parse_item',follow=True)
    ]

    def parse_item(self,response):
        item = BlogcrawlspiderItem()
        sel = Selector(response)
        blog_url = str(response.url)
        blog_name = sel.xpath('//h2[@class="titName SG_txta"]/text()').extract()
        if len(blog_name) ==0:
            blog_name = sel.xpath('//h2[@class="h1_tit"]/text()').extract()

        if len(blog_name)> 0:
            blog_name = blog_name[0]
        else:
            blog_name = '未知'

        print(blog_name)
        print(blog_url)
        item['blog_name'] = blog_name
        item['blog_url'] = blog_url

        yield item
```

基类 CrawlSpider 提供了更完善的自动多网页爬取机制，只需要我们配置的就是 rules，通过 R

ule 对象实现链接的提取与跟进。

作业

阳光热线问政平台爬虫

http://wz.sun0769.com/index.php/question/questionType?type=4

爬取投诉帖子的编号、帖子的 url、帖子的标题，和帖子里的内容。

保存到数据库中