## 使用rules

学习使用rules，首先要学习CrawlSpider和LinkExtractor

### CrawlSpider

应用rules的spider必须继承自scrapy.spiders.CrawlSpider，而不是scrapy.Spider

CrawlSpider继承自Spider，最大区别是多了一个rules属性，这个类用于进行有规则的一网站的整站爬取，通过xpath、css等一些列参数设置，拿到整个网站的你想要的链接，而不是固定的某个标签下的一些链接内容，可以省略很多的步骤

工作流程：

1. 由start\_requests对start\_urls中的每一个url发起请求（make\_requests\_from\_url)，这个请求会被parse接收。
2. 在Spider里面的parse需要我们定义，但CrawlSpider定义parse去解析响应（self.\_parse\_response(response, self.parse\_start\_url, cb\_kwargs={}, follow=True)）

\_parse\_response根据有无callback、follow和self.follow\_links执行不同的操作

特别注意：parse方法不能被占用

### LinkExtractor

|  |
| --- |
| **class** DingdianSpider(CrawlSpider):  name = **'dingdian'** start\_urls = [**'http://www.23us.so/'**]   **def** parse(self, response):  *# link = LinkExtractor(restrict\_xpaths='//div[@class="main m\_menu"]/ul/li/a[contains(@href, "1.html")]')  # links = link.extract\_links(response)* link = LinkExtractor(allow=**'list/\d+\_1.html'**)  links = link.extract\_links(response)    **for** link **in** links:  print(link.url)  print(link.text) |

参数说明：

allow：接收正则表达式或正则表达式列表，提取符合的url，如果为空，默认全部提取

deny：和allow相反，不提取的url的正则表达式

allow\_domains：允许访问的域名或域名列表

deny\_domains：不允许访问的域名或域名列表

restrict\_xpaths：接收xpath表达式或xpath表达式列表，提取符合的url

restrict\_css：接收css表达式或css表达式列表，提取符合的url

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，提取指定的属性内的链接，默认为attrs=（‘href’，）

### rules

包含一个或多个Rule

link\_extractor：对应上面学习的LinkExtractor

callback：回调函数，可以没有，callback='parse\_item'

cb\_kwargs：回调函数的参数，callback(response, \*\*cb\_kwargs) or ()

process\_links：自定义的links处理函数

process\_request：自定义的request处理函数

follow：继续提取url，查看源码可以得知，当callback为空时，自动设置为True

示例：

编写spider：

|  |
| --- |
| **from** scrapy.linkextractors **import** LinkExtractor  **from** scrapy.spiders **import** Rule  **class** DingdianSpider(CrawlSpider):  name = **'dingdian'** *# allowed\_domains = ['23us.so']* start\_urls = [**'http://www.23us.so/'**]   rules = (  Rule(LinkExtractor(allow=**'/list/\d+\_\d+.html'**), follow=**True**),  Rule(LinkExtractor(allow=**'/xiaoshuo/\d+.html'**,), callback=**'parse\_item'**),  )   **def** parse\_item(self, response):  *# auth = response.xpath('//\*[@id="at"]/tbody/tr[1]/td[2]/text()').extract\_first()  # 程序会自动把 tbody 忽略，所以在网页中是上面的xpath匹配到，但是在程序中需要修改为下面这样：* auth = response.xpath(**'//\*[@id="at"]/tr[1]/td[2]/text()'**).extract\_first()  print(auth) |