

scrapy框架梳理

scrapy五大组件和工作流程

- 1 【1】引擎Engine
- 2 【2】爬虫程序Spider
- 3 【3】调度器Scheduler
- 4 【4】下载器Downloader
- 5 【5】项目管道Pipeline
- 6
- 7 爬虫项目启动时,引擎找到爬虫文件所要第一批要抓取的URL地址,交给调度器入队列,调度器出队列,交给下载器下载,下载完成后将response交给爬虫文件做数据解析提取,提取出来的数据交给项目管道去处理,如果有需要继续跟进的URL地址,则再次交给调度器入队列,如此循环

Scrapy写爬虫项目流程

- 1 【1】创建项目和爬虫文件
- 2 scrapy startproject 项目名
- 3 cd 项目名
- 4 scrapy genspider 爬虫文件名 允许抓取的域名
- 5 【2】定义要抓取的数据结构(items.py)
- 6 【3】爬虫文件解析提取数据(爬虫文件名.py)
- 7 【4】项目管道负责数据处理(pipelines.py)
- 8 【5】全局配置(settings.py)
- 9 【6】运行爬虫(run.py) scrapy crawl 爬虫名

Scrapy项目的启动流程

- 1 【1】方式一: 通过start_urls变量启动
- 2 【2】方式二: 通过重写start_requests()
- 3 def start_requests(self):
- 4 生成所有要抓取的URL地址,交给调度器入队列

Scrapy多级页面数据抓取思路

```
1 class TencentSpider(scrapy.Spider):
2     name = 'tencent'
3     allowed_domains = ['tencent.com']
4     start_urls = ['http://tencent.com/']
```

```

5
6     def parse(self, response):
7         """一级页面解析函数"""
8         li_list = response.xpath('xxxx')
9         for li in li_list:
10             item = TencentItem()
11             # 提取具体数据
12             yield scrapy.Request(url=xxx, meta={'item':item}, callback=parse_two)
13
14     def parse_two(self, response):
15         """二级页面解析函数"""
16         item = response.meta['item']
17
18         yield item

```

settings.py中常用变量

```

1  【1】 USER_AGENT = ''
2  【2】 ROBOTSTXT_OBEY = False
3  【3】 CONCURRENT_REQUEST = 32
4  【4】 DOWNLOAD_DELAY = 1
5  【5】 DEFAULT_REQUEST_HEADERS = {}
6  【6】 COOKIES_ENABLED = False
7      False: 找 DEFAULT_REQUEST_HEADERS = {'Cookie':''}
8      True: 找爬虫文件中 yield scrapy.Request(url=xx, callback=xx, cookies={})
9  【7】 ITEM_PIPELINES = {}
10 【8】 FEED_EXPORT_ENCODING = 'utf-8'

```

Scrapy数据持久化梳理

```

1  【1】 csv json
2      scrapy crawl 爬虫名 -o xxx.csv
3      scrapy crawl 爬虫名 -o xxx.json
4  【2】 MySQL、MongoDB
5      2.1》 settings.py中定义常用变量
6      2.2》 pipelines.py中新建管道
7          from .settings import *
8          class XxxPipeline(object):
9              def open_spider(self, spider):
10                 pass
11
12             def process_item(self, item, spider):
13                 return item
14
15             def close_spider(self, spider):
16                 pass
17      2.3》 settings.py中开启管道
18          ITEM_PIPELINES = {'项目名.pipelines.类名':优先级}
19  【3】 redis

```

Scrapy分布式爬虫梳理

```
1  【1】 分布式爬虫原理
2      多台主机共享一个爬取队列,利用redis中的集合实现
3  【2】 具体实现(settings.py)
4      2.1》先写正常的Scrapy爬虫项目
5      2.2》配置settings.py为分布式
6          重新指定调度器    : SCHEDULER = 'scrapy_redis.scheduler.Scheduler'
7          重新指定去重机制    : DUPEFILTER_CLASS = 'scrapy_redis.dupefilter.RFPDupeFilter'
8          不清除请求指纹      : SCHEDULER_PERSIST = True
9          Redis主机地址      : REDIS_HOST = 'IP'
10         Redis端口号        : REDIS_POST = 6379
11         Redis的管道         : ITEM_PIPELINES = {'scrapy_redis.pipelines.RedisPipeline':200}
```

Scrapy如何发送Post请求

```
1  【1】 scrapy发送get()请求
2      yield scrapy.Request(url=xxx, callback=xxx)
3
4  【2】 scrapy发送post()请求
5      yield scrapy.FormRequest(url=xxx, formdata={}, callback=xxx)
6
7      # post请求不能放到start_urls变量中,必须重写start_requests()方法
```