**scrapy-redis学习**

# 简介

scrapy-redis是scrapy框架基于redis数据库的组件，用于scrapy项目的分布式开发和部署

特征：

* 分布式爬取

您可以启动多个spider工程，相互之间共享单个redis的requests队列。最适合广泛的多个域名网站的内容爬取。

* 分布式数据处理

爬取到的scrapy的item数据可以推入到redis队列中，这意味着你可以根据需求启动尽可能多的处理程序来共享item的队列，进行item数据持久化处理

* Scrapy即插即用组件

Scheduler调度器 + Duplication复制 过滤器，Item Pipeline，基本spider

# 安装

通过pip 安装： pip install scrapy-redis

本文档使用版本： 0.6.8

依赖环境：

python 3.6.3

redis 2.10.6

scrapy 1.5.0

# 文档

官方文档：<https://scrapy-redis.readthedocs.io/en/stable/>

源码位置：<https://github.com/rmax/scrapy-redis>

# 使用

## 配置文件

配置文件是在 scrapy 的 settings 中进行修改的：

|  |
| --- |
| *#启用Redis调度存储请求队列* SCHEDULER = **"scrapy\_redis.scheduler.Scheduler"** *#确保所有的爬虫通过Redis去重* DUPEFILTER\_CLASS = **"scrapy\_redis.dupefilter.RFPDupeFilter"** *#默认请求序列化使用的是pickle 但是我们可以更改为其他类似的。该配置2.X的可以用。3.X的不能用*  *用于序列化的两个模块 　　json：用于字符串和Python数据类型间进行转换 　　pickle: 用于python特有的类型和python的数据类型间进行转换 　　json提供四个功能：dumps,dump,loads,load 　　pickle提供四个功能：dumps,dump,loads,load #SCHEDULER\_SERIALIZER = "scrapy\_redis.picklecompat"  #不清除Redis队列、这样可以暂停/恢复 爬取 #SCHEDULER\_PERSIST = True  #使用优先级调度请求队列 （默认使用） #SCHEDULER\_QUEUE\_CLASS = 'scrapy\_redis.queue.PriorityQueue' #可选用的其它队列 #SCHEDULER\_QUEUE\_CLASS = 'scrapy\_redis.queue.FifoQueue' #SCHEDULER\_QUEUE\_CLASS = 'scrapy\_redis.queue.LifoQueue'  #最大空闲时间防止分布式爬虫因为等待而关闭 #这只有当上面设置的队列类是SpiderQueue或SpiderStack时才有效 #并且当您的蜘蛛首次启动时，也可能会阻止同一时间启动（由于队列为空） #SCHEDULER\_IDLE\_BEFORE\_CLOSE = 10  #将数据项目在redis进行处理* ITEM\_PIPELINES = {  **'scrapy\_redis.pipelines.RedisPipeline'**: 300 }  *#序列化项目管道作为redis Key存储，假如爬虫类的name='first'，name这个KEY就是 'first:items' #REDIS\_ITEMS\_KEY = '%(spider)s:items'  #默认使用ScrapyJSONEncoder进行项目序列化 #You can use any importable path to a callable object. #REDIS\_ITEMS\_SERIALIZER = 'json.dumps'  #指定连接到redis时使用的端口和地址（可选） #REDIS\_HOST = 'localhost' #REDIS\_PORT = 6379  #指定用于连接redis的URL（可选） #如果设置此项，则此项优先级高于设置的REDIS\_HOST 和 REDIS\_PORT #REDIS\_URL = 'redis://user:pass@hostname:9001'  #自定义的redis参数（连接超时之类的） #REDIS\_PARAMS = {}  #自定义redis客户端类 #REDIS\_PARAMS['redis\_cls'] = 'myproject.RedisClient'  #如果为True，则使用redis的'spop'进行操作。 #如果需要避免起始网址列表出现重复，这个选项非常有用。开启此选项urls必须通过sadd添加，否则会出现类型错误。 #REDIS\_START\_URLS\_AS\_SET = False  #RedisSpider和RedisCrawlSpider默认 start\_urls 键 #REDIS\_START\_URLS\_KEY = '%(name)s:**start\_urls'  #设置redis使用utf-8之外的编码 #REDIS\_ENCODING = 'latin1'* |

最常用的配置修改为：

|  |
| --- |
| SCHEDULER = **"scrapy\_redis.scheduler.Scheduler"** DUPEFILTER\_CLASS = **"scrapy\_redis.dupefilter.RFPDupeFilter"** REDIS\_URL = **'redis://root:密码@主机ＩＰ:端口'**  ITEM\_PIPELINES = {  **'scrapy\_redis.pipelines.RedisPipeline'**: 300 }  SCHEDULER\_PERSIST = **True** |

以上常用配置下，会在 redis 中，增加4个key：

1. 项目名:items

list 类型，保存爬虫获取到的 数据item

内容是 json 字符串

1. 项目名:dupefilter

set类型，用于爬虫访问的URL去重

内容是 40个字符的 url 的hash 字符串zme

1. 项目名: start\_urls

List 类型，用于获取spider启动时爬取的初始的urls

1. 项目名:requests

zset类型，用于scheduler调度处理 requests

内容是 request 对象的序列化 字符串

scrapy-redis对scrapy项目的改造，只需要增加以上几个配置就可以完成了

## 原理

scrapy-redis的原理非常简单，分为以下几步：

1. 将scrapy项目的 scheduler、items处理、url去重，都放入在redis中处理，其中start\_urls也可以放入到redis中
2. 对部署在不同机器的爬虫程序，都连接第一步中的redis数据库，获取需要爬取的requests，并且将爬取到的数据结构化成item之后，持久化到redis中
3. 通过另外单独的数据持久化处理程序，将redis中的item数据保存到其他普通的数据库中，譬如mongodb、mysql等