Scrapy爬虫研究

1. Scrapy框架介绍：



**数据流(Data flow)**

Scrapy中的数据流由执行引擎控制，其过程如下:

1. engine从Spiders获取初始requests
2. engine将获取的requests传递给scheduler调度，并请求要进行crawl的requests。
3. scheduler按照调度策略，返回engine要执行crawl的requests
4. engine获取要执行crawl的requests后，将requests下发给downloader下载，通过downloadermiddlewares并执行process\_request方法
5. downloader执行完下载后，将response返回给engine，通过downloadermiddlewares并执行process\_response方法
6. engine接收response后，通过spidermiddleware执行[process\_spider\_input](https://doc.scrapy.org/en/master/topics/spider-middleware.html#scrapy.spidermiddlewares.SpiderMiddleware.process_spider_input)方法传递给spider处理
7. spider处理接收到的response并返回爬取到的items或者新的requests给engine，通过spidermiddleware执行[process\_spider\_output](https://doc.scrapy.org/en/master/topics/spider-middleware.html#scrapy.spidermiddlewares.SpiderMiddleware.process_spider_input)
8. engine将spider返回的item传给pipline处理，requests传递给scheduler调度。
9. 重复上述步骤直至scheduler中requests为空

Scrapy基于事件驱动网络框架 [Twisted](http://twistedmatrix.com/trac/) 编写。因此，Scrapy基于并发性考虑由非阻塞(即异步)的方式实现。

Scrapy工程（tutorial）主要包括以下文件：

scrapy.cfg *# deploy configuration file*

tutorial/ *# project's Python module, you'll import your code fromhere*

\_\_init\_\_.py

items.py *# project items definition file*

middlewares.py *# project middlewares file*

pipelines.py *# project pipelines file*

settings.py *# project settings file*

spiders/ *# a directory where you'll later put your spiders*

\_\_init\_\_.py

Item：定义了爬取的结果，一般为自定义字段

Middleware：为自定义中间，主要是downloadermiddleware、spidermiddleware两类，download主要是在下载之前，主要是反反爬虫，spidermiddleware在获取下载结果后很少

Spiders：各种爬虫实现，可继承spider(start\_url,pares等方法)、crawlspider（rules、callback方法等）

Settings：配置文件，详情参考官方文档

1. 整体设计：

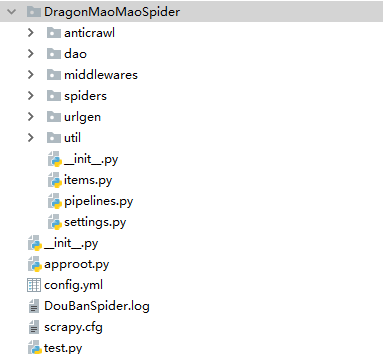
原生scrapy.spider组件将url存储在内存中，不同的spider是不能访问同一队列

基于scrapy-redis组件实现分布式，该组件重写了sipder类，start\_url存放在redis数据库中，并重写了scheduler、filter等组件。

Scrapy-cluster分布式框架kafka+redis+scrapy的一套框架，待研究

Spider业务若要支持分布式爬取，需继承RedisSpider或RedisCrawl-Spider，实现redis\_key、以及自定义的callback函数。

测试程序结构：



Anticraw：主要是反爬虫技术，当前实现有user-agent，ip-proxy（获取免费代理，本身也是一个爬虫），提供给各种middleware组件使用，测试可用

Dao：数据库操作封装，读取配置文件生成redis、mongodb client实例

Middleware：自定义scrapy的中间件，当前有user-agent动态更换模拟浏览器代理，ProxyMiddleware根据自定义策略决定是否使用代理

Spiders：具体爬虫业务实现

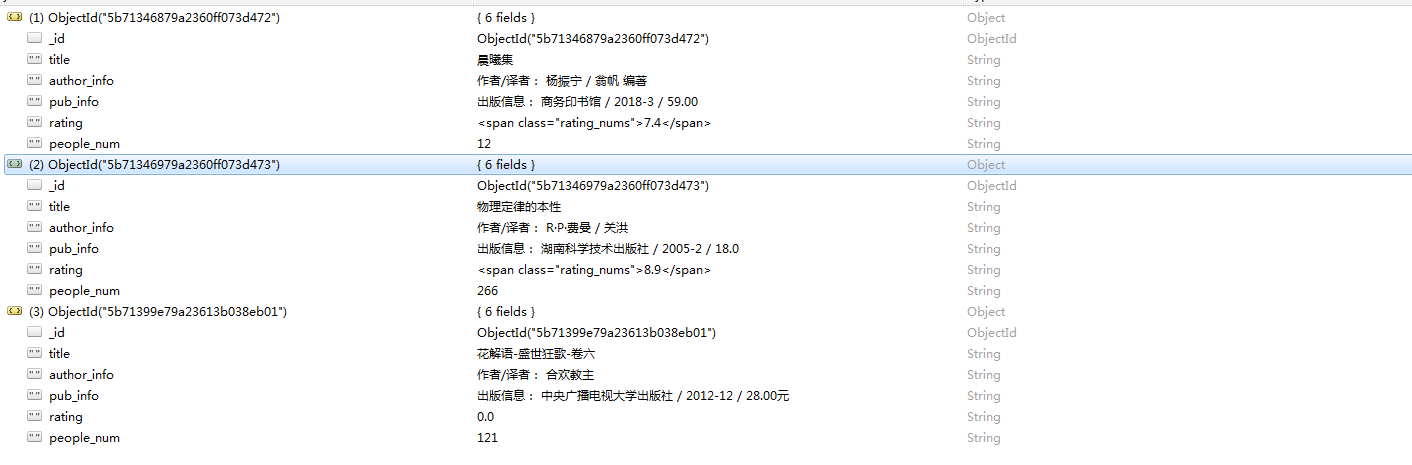
Urlgen：暂定url生成器，生成策略由需求定（定时、一次性、事件驱动）

Util：通用操作，当前实现有conf读取封装

Config.yml：自定义配置文件，redis(集群)、mongodb(集群)等配置

其他，为spider原生文件，可扩展。

单机测试：不用登陆网站爬取，可用，豆瓣读书，保存在mongodb中数据如下



1. 问题：

1、免费代理不稳定，容易造成不停的爬取IP、删除IP

2、未实际部署测试，部署脚本、方式等未确定，scrapy爬虫启动命令scrapy crawl spidername args

3、爬取业务还不精通，模拟浏览器+人操作需要研究，登陆+验证码等等

4、分布式的爬虫待研究深入，以及环境部署，运行成本等，开源的分布式框架 frontera、[*scrapy-cluster*](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Fistresearch%2Fscrapy-cluster)和[*scrapy-redis*](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Frolando%2Fscrapy-redis)

5、总结，简单的爬取可以跑，想要稳定持续的爬取有待深入，有点难度

1. 目标：

先弄个单机稳定版本，以微博为目标，需要账号