分布式网络爬虫实验

——域名信息需求爬取

[1. 分工明细 2](#_Toc11271828)

[2. 需求分析 2](#_Toc11271829)

[3. 实验流程及架构原理 3](#_Toc11271830)

[2.1 实验流程 3](#_Toc11271831)

[2.2 实验原理 3](#_Toc11271832)

[2.2.1 Scrapy 3](#_Toc11271833)

[2.2.2 IP池 5](#_Toc11271834)

[4. 模块设计及代码实现 5](#_Toc11271835)

[4.1 爬虫框架设计 5](#_Toc11271836)

[4.1.1 数据模型设计 5](#_Toc11271837)

[4.1.2 HTML代码分析与解析 5](#_Toc11271838)

[4.1.3 制作爬虫 7](#_Toc11271839)

[4.1.4 信息传输 11](#_Toc11271840)

[4.1.5 信息存储 11](#_Toc11271841)

[4.2 爬虫性能提升 12](#_Toc11271842)

[4.2.1 IP代理 12](#_Toc11271843)

[4.2.2 分布式与并行 13](#_Toc11271844)

[5. 心得体会 14](#_Toc11271845)

# 分工明细

|  |  |
| --- | --- |
| 李盛树 | PPT演讲、Scrapy框架搭建、网页token获取及分析、IP池框架搭建、测试、文档补充 |
| 刘彩荣 | PPT制作、 Scrapy框架搭建、代码整理、网页Url获取、测试、文档补充 |
| 阮少峰 | 文档撰写、Scrapy模型搭建、页面解析、爬取网页 |
| 郑立旺 | 文档撰写、Scrapy模型搭建、页面解析、爬取网页 |

# 需求分析

随着国际互联网的迅速发展，网上的信息越来越多，全球网页数量超过20亿，每天新增加730万网页。要在如此浩瀚的信息海洋里寻找信息，就像“大海捞针”一样困难。在实际生活中我们经常会使用像百度、Google这些搜索引擎检索各种信息，搜索引擎正是为了解决这个问题而出现的技术，而网络爬虫正是搜索引擎所需要的关键部分

既然百度、Google这些搜索引擎巨头已经帮我们抓取了互联网的大部分信息，为什么还要自己写爬虫呢？因为深入整合信息的需求是广泛存在的，在企业中，爬虫抓取下来的信息可以作为数据仓库多维展现的数据源，也可以作为数据挖掘的来源，甚至有人为了炒股，专门抓取股票信息。这些实际问题的解决所需要的根本技术就是分布网络爬虫。

本次实验主要的内容

唯一索引：域名

需要爬取的数据项有：

（1）网站名称、首页网址、主办单位、单位性质、网站备案/许可证号；

（2）服务器IP、IP所在地、协议类型、服务器类型、页面类型；

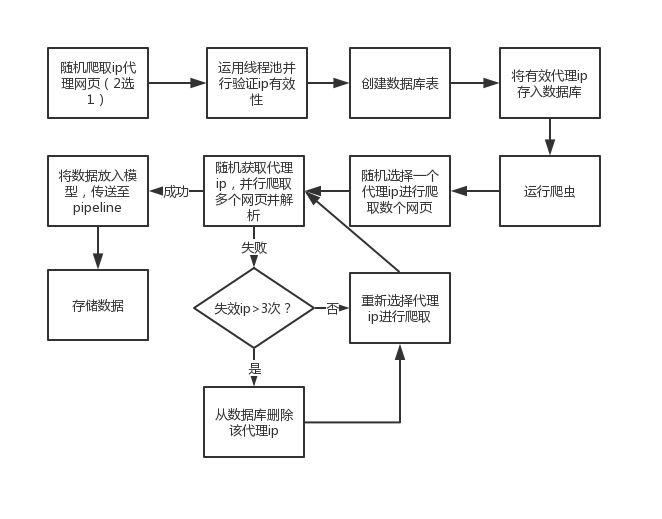
（3）域名、注册商、WHOIS服务器、DNF服务器、创建时间；

（4）全球排名（PV Rank）、访客排名（UV Rank）、国家/地区、国家/地区排名；

（5）排名走势图链接、搜索流量占比图链接

# 实验流程及架构原理

## 2.1 实验流程

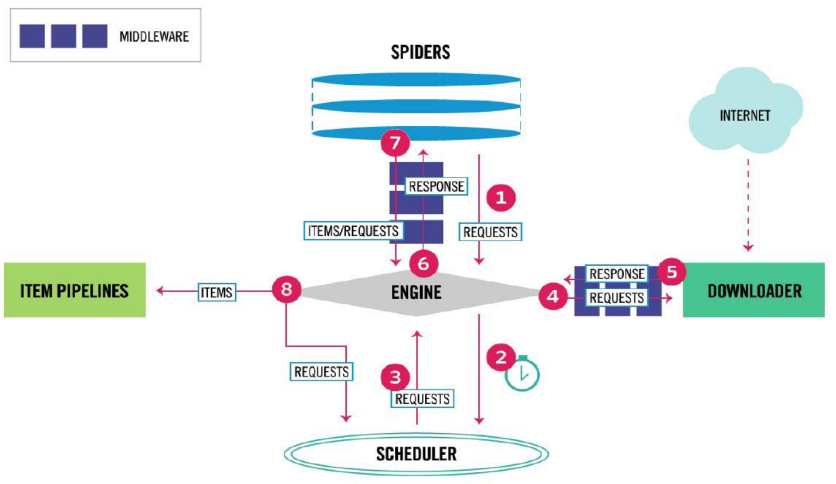


## 2.2 实验原理

2.2.1 Scrapy

这次实验我们主要使用了Scrapy的爬虫框架来爬取网站上的信息，通过Spider去爬取网页上的信息，然后将信息传到item中，再由pipeline对item进行信息的保存。

Scrapy是一个基于python，为了爬取网站数据，提取结构性数据而编写的应用框架。 可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。Scrapy的爬虫结构具体如下：

****

1. Spiders

Spider是有Scrapy用户自己定义用来解析网页并抓取制定URL返回的内容的类，每个Spider都能处理一个域名或一组域名。换句话说就是用来定义特定网站的抓取和解析规则。

Spider的整个抓取流程（周期）是这样的：

i. 首先获取第一个URL的初始请求，当请求返回后调取一个回调函数。第一个请求是通过调用start\_requests()方法。该方法默认从start\_urls中的Url中生成请求，并执行解析来调用回调函数。

ii. 在回调函数中，你可以解析网页响应并返回项目对象和请求对象或两者的迭代。这些请求也将包含一个回调，然后被Scrapy下载，然后有指定的回调处理。

iii. 在回调函数中，解析网站的内容，并生成解析的数据项。

iv. 最后，从Spider返回的项目通常会进驻到项目管道。

1. Item Pipeline（项目管道）

项目管道的主要责任是负责处理有蜘蛛从网页中抽取的项目，主要任务是清洗、验证和存储数据。当页面被蜘蛛解析后，将被发送到项目管道，并经过几个特定的次序处理数据。每个项目管道的组件都是有一个简单的方法组成的Python类。他们获取了项目并执行他们的方法，同时他们还需要确定的是是否需要在项目管道中继续执行下一步或是直接丢弃掉不处理。

项目管道通常执行的过程有：

1. 清洗HTML数据
2. 验证解析到的数据（检查项目是否包含必要的字段）
3. 检查是否是重复数据（如果重复就删除）
4. 将解析到的数据存储到数据库或文件之中

2.2.2 IP代理池

本次项目IP代理池使用了GitHub上的开源框架，因为在爬取网页的过程之中，发现其会对每日的爬取进行限制，而绕过这个限制的方便，便是使用高匿名代理。

该框架收集的代理网站有无忧代理、ip181代理、快代理和西刺代理，因为无忧代理和ip181代理的免费服务已经停止，因此最后只选择使用了快代理和西刺代理。

框架通过随机爬取多个免费代理地址，然后过滤掉无效 IP 代理后存放到 Mysql 数据库。另外，它会每 10 分钟轮询数据库中的 IP 代理数量。如果代理地址因为连接失败次数超过 3 次被删除，从而导致代理不够，则会后台重新爬取新的 IP 代理。

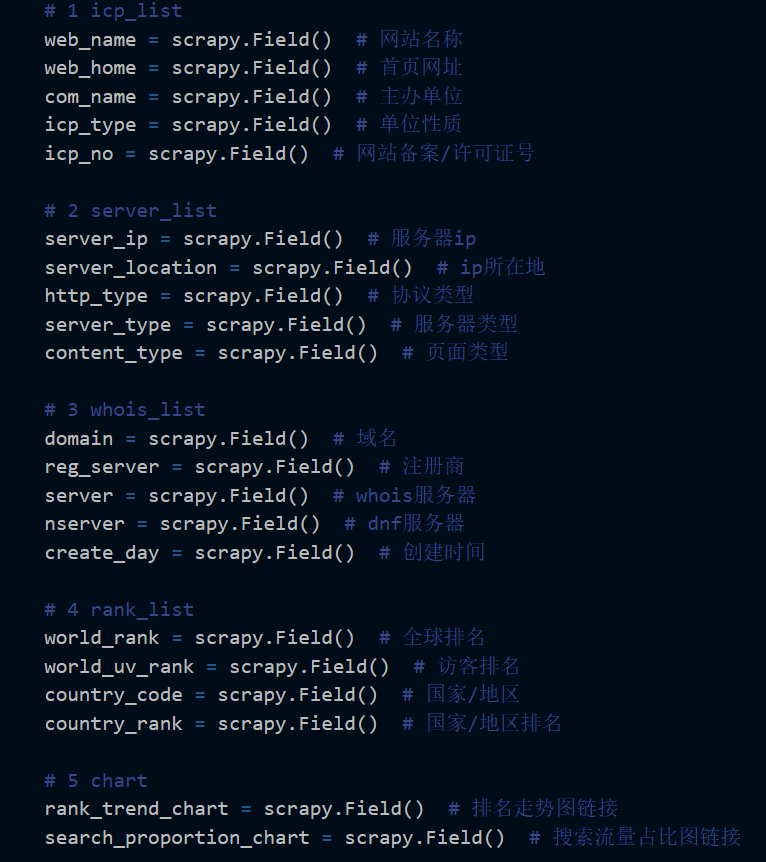
框架的基本架构也是Scrapy，通过Spider获取代理ip，再将ip的list返回，验证后写入数据库，之后启动项目的爬虫程序，在每个请求上随机添加ip代理。

# 模块设计及代码实现

## 4.1 爬虫框架设计

### 4.1.1 数据模型s设计

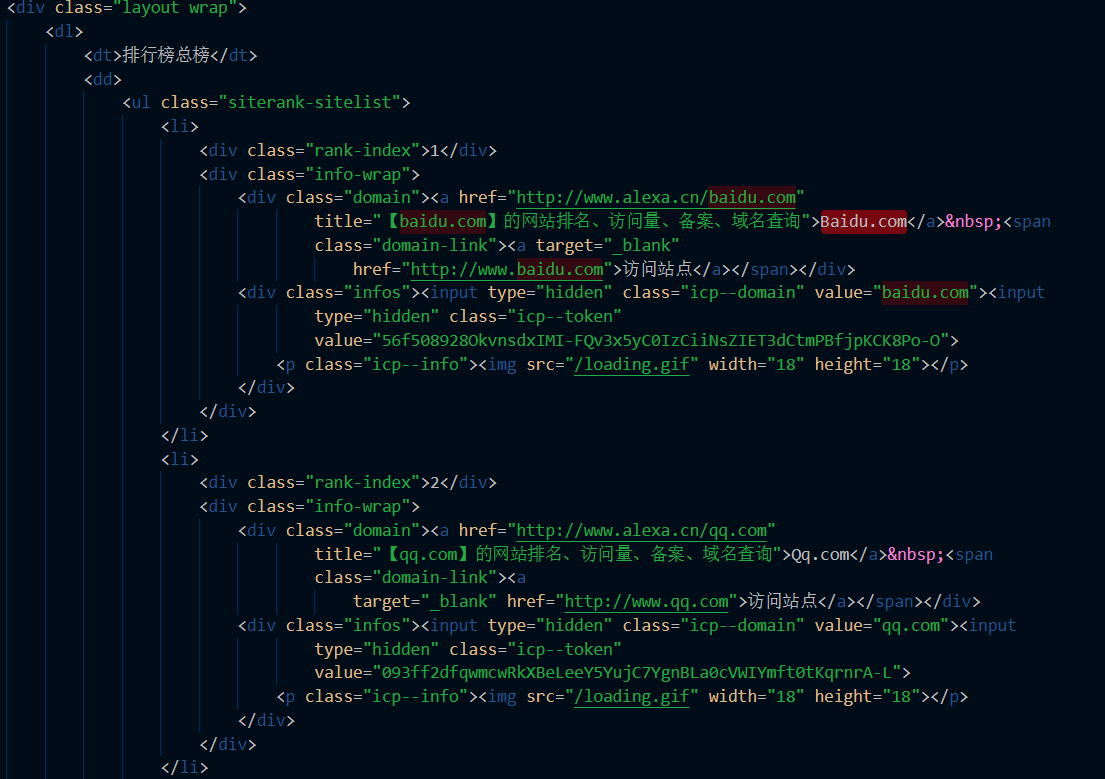
本项目抓取的数据如下信息模型表示（使用python数据模型类代码进行表示）：



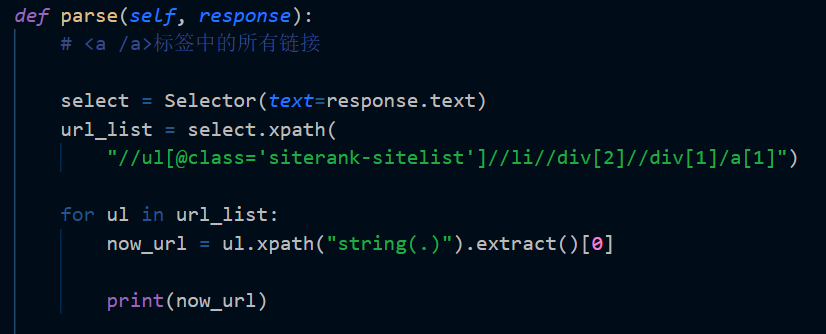
该信息模型会存储我们爬虫所得到的数据，最终会上传到Item。

### 4.1.2 HTML代码分析与解析

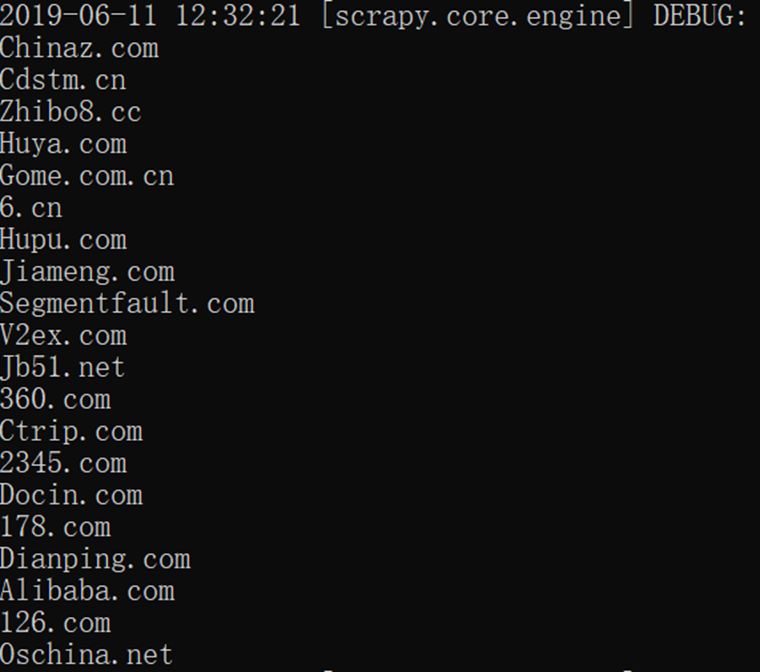
我们先使用浏览器去查看网页的源代码，可以得到以下的代码，从代码中我们可以看到网页的结构，然后通过使用Xpath技术对我们需要的有用信息筛选出来。



通过对排行榜网站的html代码的解析，提取出我们所需要的网页的URL信息。这一步的操作主要是通过直接在网页上的排行榜页面去获取上面URL，获取我们需要的URL信息，为后面获取这些网页上的详细信息做准备。

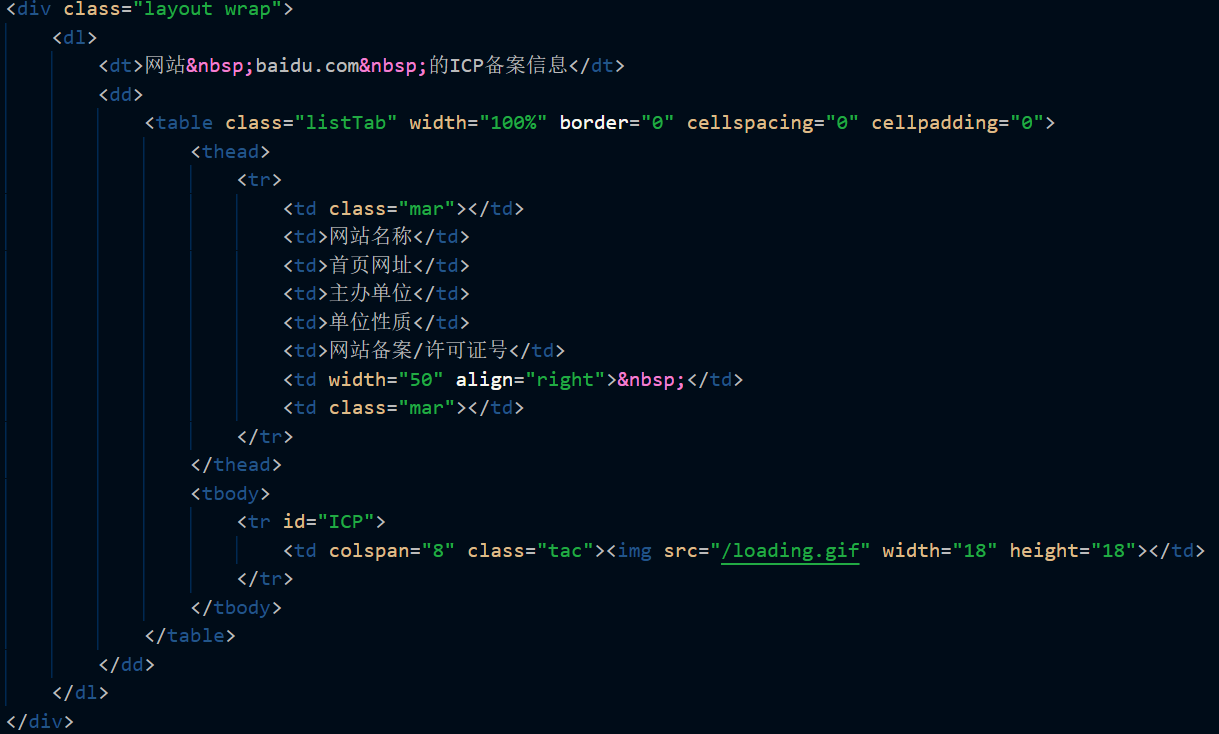


通过上面代码对HTML页面内容的解析，我们得到了以下返回的各种网站的ＵＲＬ：



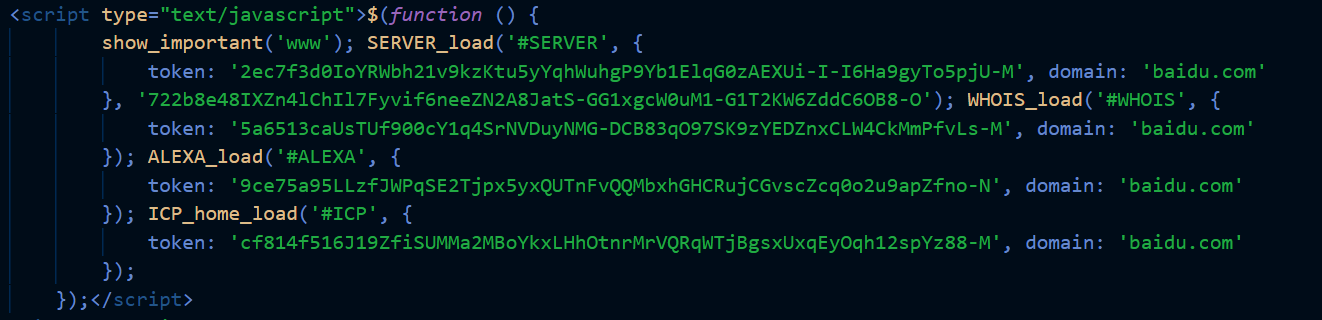
### 4.1.3 制作爬虫

通过上面的步骤，我们获取到了我们需要抓取信息的页面的URL，接下来开始制作爬虫。一开始我们希望从网页返回的响应HTML页面中去直接提取到我们需要的信息，但是我们通过对返回来的HTML响应页面分析发现，网页中并没有我们的需要的信息。



需要的具体信息在HTML页面中没有显示出来

通过分析，我们发现我们需要查找的信息是动态的，其具体信息时通过JavaScript动态获取的，所以我们通过获取该具体信息的token请求后，从而再通过token去获取我们需要的具体信息：



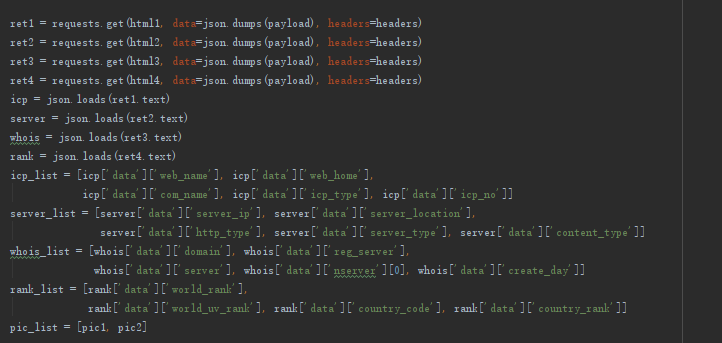
发送token后，我们就会得到一个包，包里面有我们需要的各种信息：



我们可以通过爬虫返回的响应得到所爬取网页的token值，通过编写代码，让爬虫提取出token值后，重新发送token请求，具体代码如下



爬虫发送请求后得到页面信息，我们再把该页面信息转换成JSON格式的信息，于是我们就要处理这些JSON信息，具体代码如下：



### 4.1.4 信息传输

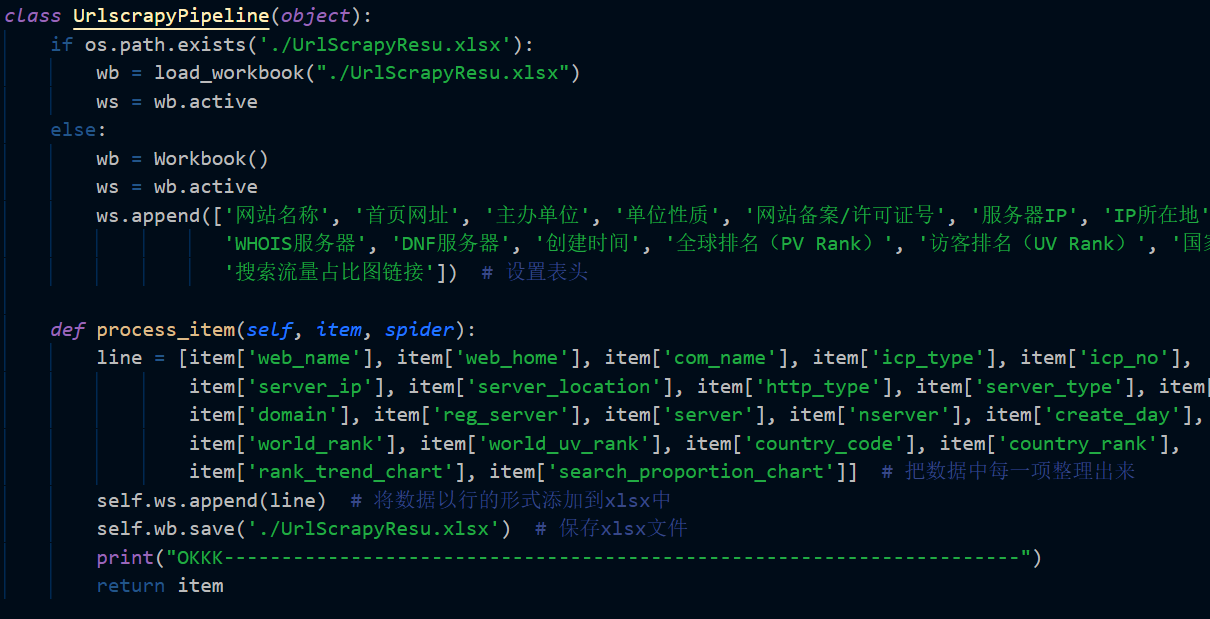
爬虫运行后就可以爬取到我们所需要的信息，于是我们将这些信息存入到我们一开始就建立的数据模型中（参考３.１.１）。具体代码如下：

图片包含 文字

已生成高可信度的说明

### 4.1.5 信息存储

在Spider模块中爬取出来的信息，返回给items后，通过pipeline模块进行处理，将信息存到xlsx文件中，具体代码如下：



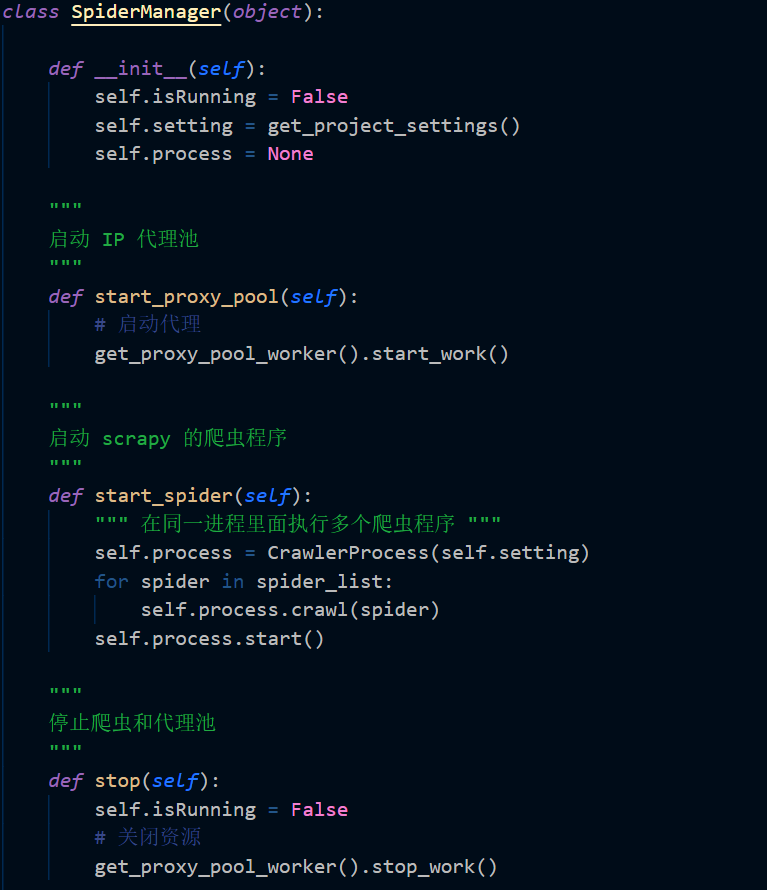
## 4.2 爬虫性能提升

通过实践，我们发现，我们去获取信息的网站对于信息查询有一定的限制，该网页会限制每一个用户每天能够查询信息的数量，并且该网页也有反爬虫机制，这限制了我们爬虫的效率。



### 4.2.1 IP代理

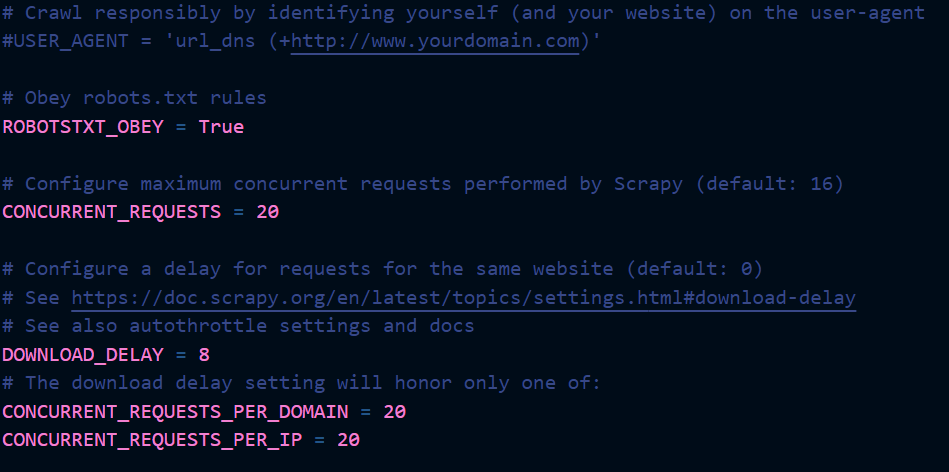
当我们需要大量的获取网站信息时，除了切换User-Agent之外，另外一个重要的方式就是设置IP代理，以防止我们的爬虫被拒绝，就如上面我们需要查询信息的网站限制用户的查询信息数量一样，所以我们通过设置IP代理去提高我们爬虫的性能。



通过设置IP代理我们发现，原本网页限制我们查询信息的数量，但是通过IP代理，我们能够获取多条我们需要的信息而同时不用担心网页的限制。

### 4.2.2 分布式与并行

Scrapy技术是基于twisted异步IO框架，downloader是多线程的。但是，由于python使用GIL（全局解释器锁，保证同时只有一个线程在使用解释器），这极大限制了并行性，在处理运算密集型程序的时候，Python的多线程效果很差，而如果开多个线程进行耗时的IO操作时，Python的多线程才能发挥出更大的作用。简单的说，scrapy是多线程的，我们可以通过设置参数让其并行度提升，但由于目前版本python的特性，多线程地不是很完全，但实际测试scrapy效率还可以。



通过调节参数我们发现，在单独获取一条我们需要的信息时，所需要的时间为2s：





并行情况下，我们爬取20页信息，所需要的时间仅仅需要10s，性能极大的得到了提升。

# 心得体会

在本次的项目之中遇到了很多问题，也学习了很多知识，收获了很多。这一次的项目任务艰巨，需要很多基础知识学习，对网络协议、系统线程、I/O调用都需要熟悉掌握，而且如果是要真正实际运用，还需要投入大量的精力去完成。

通过对相关基础知识的学习，我们对网络协议、线程管理、I/O调用、Socket通信、Scrapy、正则表达式都有了清楚的认识。然后就开始着手设计整个程序框架，并进行各模块的编程，起初因为对于网页不熟悉，且因为网页其实不是静态网页，而是通过JavaScript获取需要的信息，所以一开始并没有头绪，后来通过分析网络包的发送与接收，发现网页时通过token获取对应的内容，因此本次项目的最初目标便是转向了如何去获取token的内容。通过对数据包和网页的分析，发现网页会附上token值，且token在十秒内都是有效的，因此，我们便是学习了正则表达式，通过正则表达式直接获取网页的token，之后尝试着将网页与token连接，发送请求，能够获取正确的json数据，也就代表着我们的工作终于有了一定的进展。

之后便是还需要完成两个基本工作，因为还对Scrapy的框架不熟悉，所以也花费了一部分时间去学习Scrapy的搭建与部署，在完全了解其基本的框架运作方式之后，我们便是很快做好分工，完成了基本模型和运行框架。但是在完成之后便还是会有源源不断的问题，因为网页会对ip的访问次数进行简单的限制，因此我们又花费了大量的时间去研究如何解决这个问题，最后选择使用ip池解决这个问题，但是同时也需要选择如何去搭建ip池，原本简单手动自行搭建了一个ip池，但是效果很差。最终选择在Github上寻找适合的IP池框架，对框架进行理解并将其嵌入到项目之中，之后项目便是能够比较稳定运行了。

当然，我们还有很多的不足，比如没有花费更多的时间精力对代码和算法进行优化。同时在分工操作项目的经验上还稍显不足。另外便是获取的网页还不够多，不能够很好的去测试本次爬虫是否能够真正的稳定爬取数据。