项

目

设

计

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 版本描述 | 完成日期 |
| 1.0 | 安志斌 | 基本实现对目标网站核心数据抓取 | 2016.06.12 |

目录

[第一节 项目目的及方法 3](#_Toc453838287)

[第二节 总体设计 3](#_Toc453838288)

[2.1 爬虫架构概览 3](#_Toc453838289)

[2.2 流程设计 4](#_Toc453838290)

[2.3 条目设计（item.py） 5](#_Toc453838291)

[2.4 网页结构分析（spiders.py） 5](#_Toc453838292)

[2.5 中间件配置（middleware） 6](#_Toc453838293)

[2.6 数据库存储（pipelines.py） 8](#_Toc453838294)

# 第一节 项目目的及方法

本项目主要目的是设计面向特定网站（<http://www.qincai.net/>）的网络爬虫程序，将网页中北京地区所有企业信息抓取下来，并经过一定处理后存放到数据库中，以备下一阶段数据分析使用。

采用的工具为开源网页信息抓取框架Scrapy，scrapy是一个快速,高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据，其用途广泛，可以用于数据挖掘、监测和自动化测试。

# 第二节 总体设计

## 2.1 爬虫架构概览

设计scrapy爬虫之前首先熟悉下它的总体架构（见图2.1）及各组件的功能。

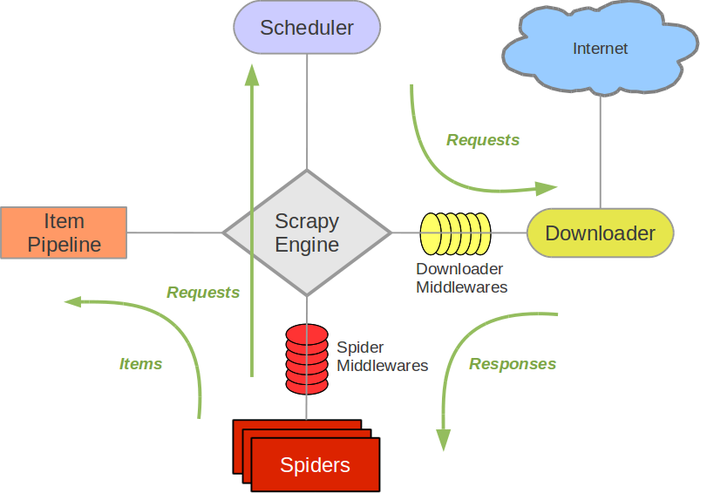


图 2.1 scrapy架构

① Engine-引擎

主要负责控制各个系统组件之间的数据流动

② Scheduler-调度

从引擎接收Request请求数据，并把这些数据排成队列，当Engine请求需要这些数据的时候再按序返给它。

③ Spider-爬虫

用户根据自己需要编写的一个类，从数据流的角度看它有三个作用，第一，给Engine传送url请求；第二，解析Downloader下载器返回的网页responses数据；第三，从responses中提取items数据并传递给pipeline类。

④ Downloader-下载器

负责从互联网获取指定网页html文件。

⑤ Downloader Middlewares-下载器中间件

位于引擎和下载器之间的钩子组件，一方面用来处理requests请求对象（从引擎到下载器），另一方面处理responses响应对象（从下载器到引擎）。常见的中间件有：浏览器user-agent中间件，代理proxy中间件，retry中间件等，通过这些中间件的存在增强了scrapy的功能。

⑥ Spider middlewares

Spider中间件是在引擎及Spider之间的特定钩子(specific hook)，处理spider的输入(response)和输出(items及requests)

## 2.2 流程设计

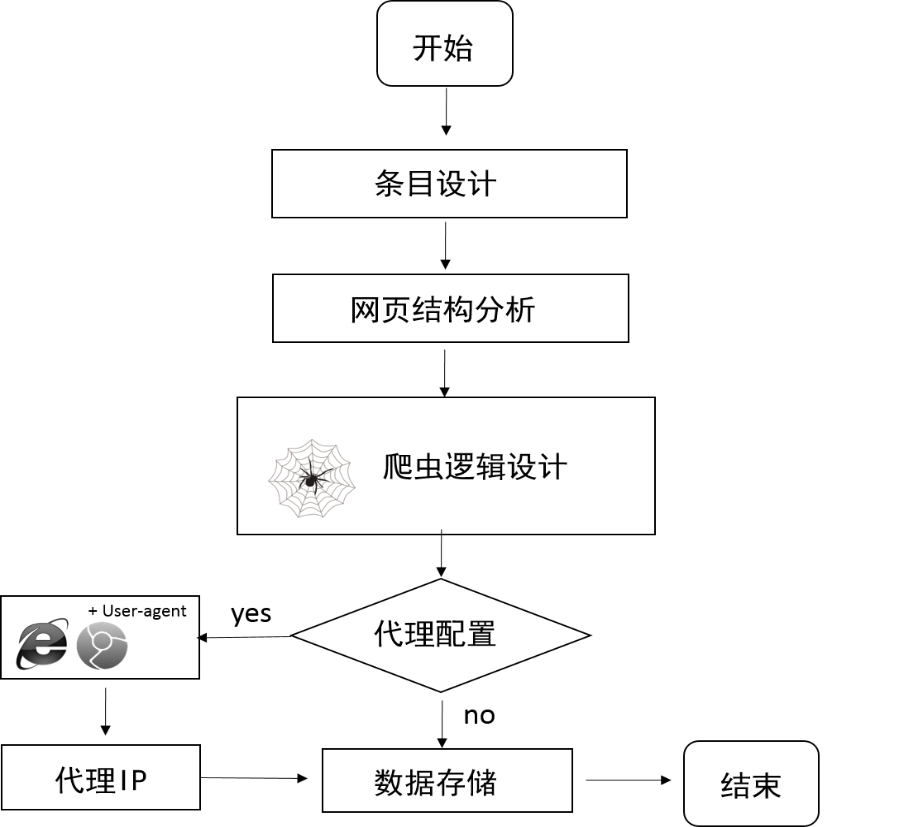


图 2.2 流程图

本爬虫的设计流程如图2.2, 首先是对目标网站中需要抓取的信息进行条目设计，其次在对网页结构详细分析的基础上编写爬虫程序核心逻辑，如果爬虫程序需要代理配置，还需要进行user-agent和ip代理设置，最后将爬取到的数据存储并输出。

## 2.3 条目设计（item.py）

爬虫设计过程中优先要考虑的当属条目设计，即筛选网页中哪些是我们关心的以及需要收集的信息，哪些是需要过滤掉不予收集的信息。针对即将要抓取的目标网站http://www.qincai.net/，我们需要收集的条目如下：企业名、网站地址、法人代表、注册资本、经营模式、员工数量、主要市场、客户类型、所属行业、产品信息、行业信息、区位信息、相关链接、企业标签、最后更新、浏览次数、企业描述、联系人、邮编、地址、电话、传真等，这些内容在网站上的排列格式如下图2.3所示，在爬虫项目文件目录中路径为：

yellowpageSpider/yellowpage/items.py



图2.3 网页信息条目排列

## 2.4 网页结构分析（spiders.py）

网页中需要收集的各类条目信息需要从服务器发送过来的源代码中进行查找，而网页结构分析就是目的就是在纷繁复杂的源代码中快捷地查找到对应的条目信息。常用的分析工具为chome浏览器elements检查选项。

在完成网页结构分析之后，接下来的任务就是设计爬虫项目的核心组件spiders（yellowpageSpider/yellowpage/spiders/spiders.py），它的作用是定义网页信息具体的查找逻辑。

针对本项目目标网站的情况，各公司对应的关键的信息如企业名、网址、更新时间、电话以及地址信息位于第一级深度的公司指数页面上（图2.4）；更加详细的信息位于第二级深度页面上（图2.5），在spiders.py文件中分别对两级页面进行逻辑设计。



图2.4 公司指数页面



图2.5 公司详细页面

## 2.5 中间件配置（middleware）

1)、user-agent中间件

Scrapy框架默认user-agent为Scrapy-version (+http://scrapy.org)，如果直接使用此默认设置访问目标网站，很多时候会遭到服务器拒绝回应，所以需要对user-agent进行一定得伪装。

针对本项目，首先创建一个包含许多待用user-agent的文件useragentlist.txt,所在位置为yellowpageSpider/yellowpage/utils /useragentlist.txt；其次，创建自定义的UserAgentMiddleware浏览器中间件实现从上述文件中随机选取user-agent，所在位置为：yellowpageSpider/yellowpage/downloadermiddlewares /rotate\_useragent.py

2)、proxy代理IP中间件

很多时候，通过简单的user-agent的随机切换还不能达到欺骗服务器的目的，因为每次网页访问所携带的代理IP仍然是一样的，服务器端还是可以比较轻松地通过同一IP较为频繁地访问判断访问者并非真实的用户。所以，这里需要设置代理IP中间件yellowpageSpider/yellowpage/downloadermiddlewares/rotate\_proxy.py

其设置逻辑如下：

首先，代理测试阶段(0-45s)，在这一阶段，Scrapy每条线程随机从代理池中获取各自IP并启动网页请求，如果某一进程成功返回页面信息，就给该代理IP访问成功次数增加1次。当这一阶段结束时候，我们就得到经过测试的代理成功统计数。

第二阶段，代理稳定访问阶段（46-360s），将前一个阶段中获取的代理中选取成功次数最高的某一代理，用该优质代理在这一阶段内高效的访问目标网站。 循环第一、第二阶段

……

需要注意的是，使用这一策略的出发点在于提高网页抓取效率，因为如果每次请求都频繁切换代理的话，必然使得失败代理出现的频次明显增大，抓取耗时大大增加。当然，在第二阶段中也很可能发生某代理使用多次之后失效，这时就要及时切换到另一随机代理上。

3）、代理池

创建IP代理池，可以将其保存到单独的txt文件中以供调用，IP代理由于瞬时性的特点，需要隔一段时间就去抓取新鲜的代理以供使用，所以需要编写脚本频繁地去从网络上收集。代理池和该脚本所在路径分别为：  
yellowpageSpider/yellowpage/utils/validProxy.txt,

yellowpageSpider/yellowpage/utils/crawl-free-proxy-from-gatherproxy.py

## 2.6 数据库存储（pipelines.py）

为了便于数据快速存取，本项目采用sqlite3存放抓取到的每条信息，详细设计见：yellowpageSpider/yellowpage /pipelines.py