

毕业设计（论文）

题目名称：“72便”郑州市便民服务网站设计与实现

---爬虫模块

学院名称：软件学院

班 级：RB软工网151

学 号：20156040129

学生姓名：王贺

指导教师：邢颖

2019年 6月

**论文编号：201560040129**

**“72便”郑州市便民服务网站设计与实现**

**---爬虫模块**

**Design and implementation of "72 Bian" zhengzhou convenience service website**

**--- Crawler Module**

学院名称：软件学院

班 级：RB软工网151

学 号：201560040129

学生姓名：王贺

指导教师：邢颖

2019年 6月

**摘 要**

从20世纪90年代互联网引入中国，中国互联网发展经历了近30年；随着中国互联网日新月异的快速发展、互联网技术的百花齐放，人们的生活也更加丰富多彩。

近两年“互联网+”的话题让人们津津乐道，互联网+就是让人的生活和互联网结合起来让我们的生活更加便利，如今成熟的产品有美团、摩拜单车、滴滴打车等。针对这个互联网+的话题我们设计了一个郑州市的便民服务的网站；其中这个包含多个便民服务模块，如二手市场、新闻资讯、问答资讯等模块，而我主要为这些模块提供数据支持，并且对这些数据获取筛选等数据处理操作。

当前的网络技术和信息已经发展的非常成熟, 随之而来的即是信息的获取途径和获取效率问题, 传统的媒体,犹如电视、报纸、广播等由于传播的时效性和地域性的限制，并不能很好的在以快速高效为特点的互联网环境中让人接受，在这样的大环境下，网络信息的传播就需要有更好的载体，新闻资讯网站应运而生，弥补了传统媒体平台的缺点，更加适合当下互联网环境，人们对信息的获取途径逐渐转向了以网站为主的信息化平台，依赖大规模的网络基础设施，极速且精确的传播着无数信息，获取到生活资讯、新闻、娱乐、租房等与人息息相关的大量信息。

整个项目都是模块开发，采用微服务架构来实现，针对登录这一块采用redis分布式session和单点登录技术来达到一次的登录多出同步的效果；数据抽取主要是都对各大网站成熟的数据编写脚本进行程序获取（爬虫），脚本爬取的过程中需要对爬取策略进行规划，防止脚本被目标网站发现从而进入黑名单，不同网站有不同的爬取规则，这个策略就要根据一些发爬虫措施来制定反反爬虫策略；数据获取过程中需要对数据进行筛选和清洗的操作，防止质量不高的数据影响数据展示；最后把这些过滤后的数据放到对应的数据库中提供给网站使用；爬虫过程中需要对整个爬虫进行监控，随时汇报爬虫情况；整个项目中所用的变成语言是Python,数据库有Redis、Mongodb、Mysql,成熟框架主要用到Requests和Scrapy等。

**关键词**：redis；爬虫；Session；单点登录；互联网；

Abstract

Since the Internet was introduced into China in the 1990s, China's Internet development has experienced nearly 30 years. With the rapid development of China's Internet and the blooming of Internet technologies, people's lives have become more colorful.In the past two years, the topic of "Internet +" has aroused people's interest. Internet + is to combine people's life with the Internet to make our life more convenient. Nowadays, mature products include Meituan, mobike and didi taxi. Aiming at this Internet + topic, we designed a website for convenience services in zhengzhou. Among them, this module contains multiple convenient service modules, such as second-hand market, news information, question and answer information, etc., for which I mainly provide data support and conduct data processing operations such as data acquisition and screening.

The whole project is a module development, which is realized by micro service architecture. For login, redis distributed session and single sign-on technology are adopted to achieve the effect of multiple synchronization of one login. Mature to each big web site data extraction is mainly of data and obtain (spider) procedure to write the script, the script crawl in the process of the need to crawl strategy planning, prevent scripts from being discovered by the target site to blacklist, different sites that have different rules, this strategy will be based on some hair crawler measures to reflect the crawler strategy; In the process of data acquisition, it is necessary to filter and clean the data to prevent the low-quality data from affecting the data display. Finally, these filtered data are put into the corresponding database for the website to use. It is necessary to monitor the whole crawler in the process of crawler and report the crawler situation at any time; The language used in the whole project is Python, and the database includes Redis, Mongodb, Mysql, and mature framework is mainly used for Requests and Scrapy, etc..  
Keywords: Requests; The crawler. The Session; Single sign-on; Internet;

目录

[第1章 引言](#_Toc13376)

[1.1 背景](#_Toc14846)

[1.2 研究内容和意义](#_Toc21715)

[第2章 系统需求](#_Toc19036)

[2.1 系统概述](#_Toc30995)

[2.2 系统功能](#_Toc23206)

[2.2.1 爬虫](#_Toc22970)

[2.2.2 搜索引擎的设计和实现模块](#_Toc27206)

[2.3 系统性能](#_Toc25524)

[2.4 运行环境](#_Toc14569)

[2.5 用例规约](#_Toc27842)

[2.5.1 数据爬虫模块](#_Toc3837)

[2.5.2 反反爬虫模块](#_Toc9829)

[2.5.3 数据存储模块](#_Toc32653)

[2.5.4 爬虫部署模块](#_Toc2278)

[第3章 系统分析](#_Toc5533)

[3.1 爬虫工作模型](#_Toc32679)

[3.2 架构设计](#_Toc12713)

[3.3 系统关键技术](#_Toc10551)

[第4章 系统实现](#_Toc16385)

[4.1 新闻资讯爬取](#_Toc20661)

[4.2 租房信息爬取](#_Toc30481)

[4.3 图片资源爬取](#_Toc13185)

[4.4 反反爬虫措施设计](#_Toc1240)

[4.5 爬虫部署](#_Toc32082)

[第5章 系统测试](#_Toc3349)

[5.1 数据爬虫](#_Toc30251)

[5.1.1 数据爬虫](#_Toc9723)

[5.1.2 新闻资讯爬虫](#_Toc30916)

[5.1.3 租房爬取](#_Toc12139)

[5.1.4 问答平台爬取](#_Toc15711)

[5.1.5 图片爬取](#_Toc254)

[5.2 反反爬虫](#_Toc9249)

[5.2.1 伪装浏览器](#_Toc3433)

[5.2.2 改变请请求时间间隔](#_Toc17259)

[5.2.3 设置Cookie模拟登录](#_Toc19972)

[5.2.4 IP代理池](#_Toc25033)

[5.3 数据入库](#_Toc8055)

[5.4 分布式爬虫](#_Toc103)

[第6章 结束语](#_Toc16600)

[致谢](#_Toc564)

[参考文献](#_Toc28869)

[附录](#_Toc7448)

[附录A 郑州市便民服务网站设计与实现爬虫模块的核心代码](#_Toc16442)

# 第1章 引言

## 1.1 背景

随着国内引入互联网近三十年，互联网行业的高速发展，中国互联网行业在国际也达到了不俗的成就，国内互联网产品层出不穷；便民服务类的互联网产品也爆发时的增长，比如餐饮类、居住类、出行类等；这些互联网产品很大程度的方便了我们的日常生活，让我们感受到了互联网的便捷；种种迹象显示出了，伴随着互联网的高速发展，人们在生活中依赖互联网的程度节节攀升，人们的生活逐步趋向于智能化；但是对日常民生生活的传统服务行业，无法完全覆盖一些目标功能，民生点到点的服务要求更为细节的需求，这些需求在一些人文文化发达地区尤其的突出。我希望通过我们郑州市的便民服务网站可以有效的解决这些问题，提供较好的解决方案。这个项目主要为郑州市的一些便民服务行业的数据进行整合，方便居住在郑州市的人民生活。

此项目是根据调查一些生活在郑州市的居民设计的，从调查的数据显示四分之三的居民感觉一些便民服务网站种类繁杂五花八门，好多人都不知道如何选择，并且这些网站的数据都市独立的，如果想要找多种信息就要查询多个网站。为了解决这个问题，从而设计了这个郑州市便民服务的网站，整合了日常人们生活所需要的数据信息，并且其中还添加问答服务和废品回收的功能，更加贴切人们日常生活的需求。项目本身的目的就是为郑州市生活的人们提供快速的便捷服务，从而让生活在郑州的人们享受更快的便利。

## 1.2 研究内容和意义

传统的互联网便民服务产品都是针对性单个行业进行服务，这样的情况导致了我们在生活着需要下载很多的app或者记录很多网站，所以针对这个情况我们开发了一个综合日常经常用到的几个功能的便民服务网站。与其他网站不同之处就是我们的项目整体采用的市微服务架构，一个功能模块采用单独服务进行拆分；这样的好处就是相互之间功能模块互不影响，易于开发和维护，单个应用某个模块或需求需要修改，就将不得不重新部署整个应用，而微服务正是解决这样问题的一种方式。微服务应用是将服务进行拆分不同的服务，但它们还能彼此连接，就算某个微服务需求有变化需要重新修改，只需要重新部署这个服务即可。生活中，我们每天都会关注新闻资讯、查询问题、租房、出售二手物品等一些小事件。往往我们做以上事件的时候需要到多个网站进行逐个进行。这样既浪费我很多时间也纠结到底选哪个平台的数据。为此我们开发了这个平台，通过网络爬虫对多个平台的信息整合到我们的平台，这样既能节省用户很多事件还能过滤很多无用的广告信息，增加了用户的体验。从而提供的是用户真实需要的信息。

我们的项目分了两大功能模块：其一是服务注册模块，服务注册模块包括租房查询服务、新闻资讯服务、问答服务、预约回收服务、二手市场服务；这些服务主要为用户提供搜索和数据展示。其二是数据获取模块，此模块主要为服务模块提供数据支持；数据模块会对多个目标网站进行大量的数据爬取并且需要对数据进行解析、数据清洗、数据分类、数据操作等步骤。在数据获取过程中要对数据进行去重增量爬取；以防止重复数据。对于爬取的目标网站每天的数据都会及时更新，这样我们就要设置爬虫每天定时爬取；多个平台数据非常大，这个时候我们为了加快数据的获取效率需要采用分布式爬虫来提交效率；这样数据的解析、去重、过滤、操作等操作都会很多程度的提升。

# 第2章 系统需求

## 2.1 系统概述

郑州市便民服务网站设计与实现整个项目是完全基于python编程语言来完成的要给web项目；功能模块之间采用微服务架构，模块和模块直接不相互影响。整个项目主要提供了二手市场的交易平台、新闻资讯平台、租房信息查询平台、预约回收平台、问答平台多个功能。这些展示的功能数据主要是从当前比较热门的互联网平台上进行爬虫爬取然后进行筛选；然后整合数据库中提供给自己的项目来使用 。用户通过真个平台展现的书可以做多个网站的事情。

## 2.2 系统功能

郑州市便民服务网站设计与实现这个平台类似一个局部的58同城便民服务网站，；针对的用户是生活在郑州市的居民。项目主要分为两大模块，数据模块和数据展示模块；数据模块主要是对数据的获取和整理；数据展示模块主要是根据数据模块提供的数据进项展示。而这两个大模块之间有分为多个子模块。系统的总结抠图如下：

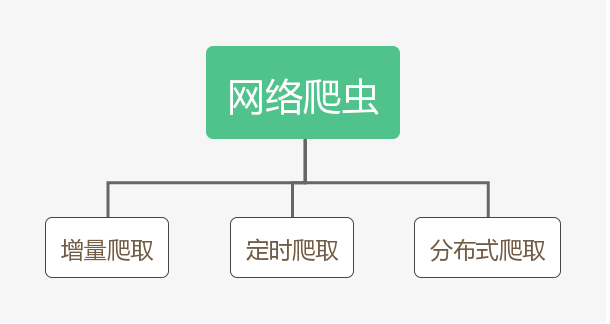


图2.1系统总结构图

## 2.2.1 网络爬虫

当前项目中的数据都是根据python编程语言中的pyspider框架进行获取的。目前主要准对的爬取网站有五八同城的租房信息，安居客的租房信息，头条的新闻资讯，知乎上的问答数据。这些爬虫需要根据不同的网站而制定不同的解析数据规则。通过解析获取数据，然后对数据进行清洗和存储的关系型数据和非关系型数据库中来供平台使用。

数据爬虫爬取的过程中分了多个步骤，不同的步骤有着不同个数据操作。首先我们要对爬虫进行页面解析，来制定规定规则数据爬虫；然后需要对数据的下载，紧接着就是对数据清洗过滤的操作，确保数据的准确性，最后把整理好数据入库。针对爬虫的完成度不确定性，我们对爬虫都加入了爬取完成后邮件通知操作，这样我们就可以及时获取爬虫信息。数据爬虫机构图如下：

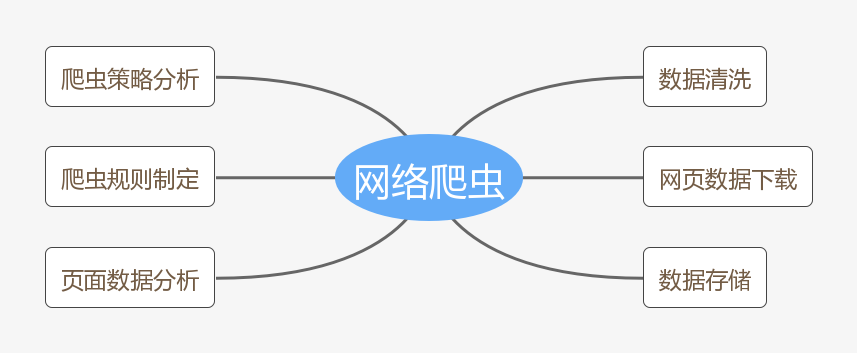


图2-3 数据抽取模块图

## 2.2.2 单点登录系统

传统的服务一个登录点就可以了，但是微服务架构需要达到一点登路多点同步的要求，当服务要求性能高的时候需要整个微服务架构分布式部署；这样的部署就需要redis非关系型数据库来做分布式session记录登录状态。单点的服务访问服务的时候需要到登录中心去确认是否已经登录，如果已经登录记录整个单点服务的session登录状态；当退出登录的时候都会通知服务站点次session已经推出登录；这样就做到了单一登录系统。

## 2.3 环境支持

1. 数据库：Mongodb非关系型数据库；MySql关系型数据库。
2. 编程语言：Python3
3. 服务器：Ubuntu16.4

## 2.4 网站性能

1. 爬虫的数据的速度必须流畅，效率高。
2. 页面操作没有卡顿现象，能够基本的完场用户操作。
3. 用户的并发量要达到千人以上。

## 2.5 用例规约

### 2.5.1 数据爬虫模块

数据爬虫主要用的框架是python编程语言的Pyspider框架，pyspider的优点是简单，立刻就能上手，脚本编写规则；并且这是由本国人编写的开源框架。数据爬虫主要爬取的网站有趣头条、今日头条、58同城租房、爱租房等互联网平台。然后数据经过一些特定清洗筛选操作后放入mongodb数据库中，其他的数据展示功能模块从数据库中获取数据来展示。接下里进行数据爬虫的具体流程和操作；数据爬虫的操作主要分为页面解析、制定爬虫策略、数据定位、数据清洗和过滤、反反爬虫措施、爬虫部署、数据存储。数据爬虫的用例规约如下：

表2-1 数据分析与解析

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS1-1 | 功能名称 | 数据分析与解析 |
| 前提条件 | 分别获取不同平台URL链接地址 | | |
| 基本事件流 | 1. 页面解析：根据xpath解析规则对网页的结构进行分析，或者想要的数据规律。 2. 制定爬虫策略：根据获取到url要制定不同的数据抽取策略。 3. 目标数据查找：因为目标网站中的每个页面查找的目标信息不同，所以要根据不同页面的结构获取下个要的数据。 4. 数据清洗和筛选：有些网站网页的数据并非一直按照一个结构展示，这样导致获取数据失败，所以要对获取到的数据进行取人筛选过滤。 5. 反反爬虫措施：网站会对一下恶意爬虫进行抵制，比如会限制ip、添加多个请求头操作；而我们将会使用IP代理池和请求头池来应对反爬。 6. 爬虫部署：当制定好爬虫的规则后需要对爬虫放到服务器上进行部署，部署过程中要对爬虫进行脚本启动管理，部署过程中还会开启多线程进行高速率爬取。 7. 数据存储：爬虫对数据一系列操作后按照数据库字段入库，对数据进行二次利用。 | | |
| 后置条件 | 首先要先获得目标网站首页的链接 | | |
| 执行者 | 超级用户 | | |

1. 新闻数据的爬虫执行人是超级用户；主要爬取近一段时间的热门资讯；能够提供数据平台最新的新闻信息，及时获取最新的新闻信息。爬取新闻资讯主要爬取新闻的Title、Content、Author、发布时间，版权等关键内容。用例规约如下：

表2-2 新闻数据获取

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS1-2 | 功能名称 | 新闻数据获取 |
| 前提条件 | 获得目标网站链接地址 | | | |
| 基本事件流 | 1. 对新闻的分类爬取和数据请求接口分析 2. 解析网页然后根据解析的网页获取新闻的标题和新闻内容。 3. 数据定位，根据查看网页的数据规律，进行xpath获取数据，爬取的平台主要是趣头条和今日头条。 4. 通过分析网页的数据结构和规定要对新闻的标题、浏览数量评论、版权、作者、详情、图片地址等数据的获取。 5. 对爬取的新闻资讯的数据进行筛选，主要判断是否是空数据和乱码的数据，获取的url必须完整，不然将会对以后数据产生影响。 6. 获取新闻数据下一页接口，进行参数分析来达到接口的请求要求。 | | | |
| 执行者 | 超级用户 | | | |

1. 对一些问答平台的数据爬取，根据不同的分类获取不同类别的问答url;然后根据获取的url来进行问答种类的列表页进行分析。从而获得多个问答详情页的url；这些url组成了一个很大的数组。再逐个分析详情页的元素结构层次，从详情页中获取问答平台的详情信息。用例规约如下图:

表2-3 QA数据获取

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS1-3 | 功能名称 | QA数据获取 |
| 前提条件 | 获得目标网站URL | | |
| 基本事件流 | 1. 查看问答平台的类型分类，根据不同的类别制定不同的爬虫规则。 2. 通过问答分类URL进入每个分类的列表页从而获得每一个分类列表中的每一条数据的详情页地址。 3. 根据每一条详情页的页面结构层次来定位数据的位置和获取方式。 4. 从定位到数据来获取数据，然后进行过滤清洗，最后入库。 | | |
| 执行者 | 超级用户 | | |

1. 下载页面中的图片，主要根据xpath的方式找到图片的src属性，从而能获得图片的服务器地址，然后根据图片地址进行下载图片，重新对图片进行命名并且在数据库中保存新的图片地址。用例规约如下：

表2-4 图片资源数据获取

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS1-4 | 功能名称 | 图片资源数据获取 |
| 前提条件 | 目标网站URL | | |
| 基本事件流 | （1）分析图片的位置。  （2）利用获得到的src中的值来下载图片  （3）爬虫通过图片的链接地址获得图片二进制数据，进行写入文件中，图片命中需要根据uuid来进行命名操作。  （4）根据新下载过来的图片名称和路径写入数据库中来供其他模块使用。 | | |
| 执行者 | 超级用户 | | |

1. 58同城租房信息的爬取和写入数据库；租房信息模块的需求包括租房信息的获取，如房子的租金、位置、租房整套出租还是单间出租、价钱、房间情况等必须的信息。租房信息面向的参与者是所有的人，租房信息的爬虫用例参与者是开发者。房源信息目前主要针对五八同城和爱租房进行爬取。用例规约如下：

表2-5 租房数据获取

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS1-5 | 功能名称 | 租房数据获取 |
| 前提条件 | 目标网站URL | | | |
| 基本事件流 | 1. 根据页面的树形结构，分别分析租房中的标题、价格、位置等重要信息的xpath 进行查找。 2. 根据关键字中的xpath来获取每一个页面中关键字段的信息。对于这些关键字的信息进行自己想要的数据结构进行整合。 3. 对爬取的租房资讯的数据进行筛选，主要判断是否是空数据和乱码的数据，获取的url必须完整，不然将会对以后数据产生影响。 4. 根据自己的数据库设计整理相应的数据结构，然后把数据存储到数据库中。 5. 爬虫结束后进行邮件通知超级用户。 | | | |
| 执行者 | 超级用户 | | | |

### 2.5.2 反爬虫反制

有些公司针对一些无规则爬虫需要进行防护；因为过多的爬虫会消耗服务器资源导致服务器资源匮乏甚至自己公司的产品收到影响。一些公司会会使用Robots爬虫协议声明本产品的数据是不可爬取的，这样就会对制定爬虫的人启到警告提型作用。但是有些公司也会对爬虫进行必要的反爬虫措施；比如对需要保护的数据进行对称加密、出现验证码策略、访问频率限制加入黑名单策略等一些反爬操作。针对不同的反爬措施我们要制定不同的反反爬措施来应对。

反反爬虫是针对一些网站制定的反爬虫措施的应对，要从网站访问中查看目标网站的反爬规律，分析主要的反爬规则，根据分析出来的规则来制定相应的措施。反爬虫反制的用例规约如下：

表2-6 反制措施

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS2-1 | 功能名称 | 反制错误 |
| 前提条件 | 网站是否遵守爬虫协议 | | |
| 基本事件流 | (1)网站是否遵守爬虫协议，然后注意网站爬取的数据有那些版权问题和那些合法用途。  (2)根据请求的动作分析请求中的headers头部信息，进行脚本发启请求。  (3)分析去请求中cookies信息进行模拟登录，请求那些需要登录才能看到的数据。  (4)针对有访问频率限制的网站则采用间隔爬虫来实现，每个几秒后请求一次，从而尽量来模仿人的行为来访问网站。避开系统的访问限制。  (5)有些网站会对IP来进行限制，那么我们采用IP代理池的方法来爬取数据，每经过一段时间来切换IP请求网站数据。I | | |
| 执行者 | 超级用户 | | |

### 2.5.3 写入数据库

爬虫的目的就是获取想要的数据，来对平台的数据支持；爬取的数据要进行按照自己的数据结构表来存储到数据库中，然后拿到这些数据库中的数据展示到前台的功能模块。用例规约如下：

表2-7 写入数据库

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS3-1 | 功能名称 | 写入数据库 |
| 前提条件 | 获得的数据是否合格 | | | |
| 基本事件流 | （1）设计表结构，结合前台接口和爬虫的数据制定不同数据库表。  （2）为了需要达到快速存储数据的目的，数据存储采用了异步存储的方式提高存储的效率。  （3）确定数据的确定性。  （4）将数据批量导入数据库mysql中，设计表结构，字段名称和数据类型等。 | | | |
| 后置条件 | 数据无误 | | | |
| 执行者 | 超级用户 | | | |

### 2.5.4 爬虫部署

爬虫部署主要是把写好的爬虫程序放到服务器上去执行，这样爬虫就能时时刻刻都是爬数据。这样不占用自己电脑时间，而且还能和数据库进行无缝连接。用例规约如下所示:

表2-8 爬虫部署

|  |  |  |  |
| --- | --- | --- | --- |
| 功能编号 | FTS4-1 | 功能名称 | 爬虫部署 |
| 前提条件 | Linux服务器一台 | | |
| 基本事件流 | 1.把爬虫脚本程序放到服务器上。  2.启动数据库服务。  3.把服务器上的爬虫都运行起来。  4.记录爬虫脚本日志。  5.将服务器上的爬虫爬取的数据存入数据库中。 | | |
| 后置条件 | 所有爬虫开始运行，并且服务器上爬虫能获得数据 | | |
| 执行者 | 超级用户 | | |

# 第3章 系统分析

## 3.1 爬虫工作模型

便民服务网站设计与实现的数据支持主要是由爬虫模块完成，爬虫模块主要采用的是python编程语言的PySider框架来实现的。PySpider爬虫框架的用途非常广泛，如数据挖掘和自动化测试；但是此框架主要的操作是进行数据挖掘的工作。PySpider爬虫框架主要是使用redis消息队列来进行通信的。这种通信方式很大程度上增加了爬虫的爬取数据的效率以及数据的加载量。其PySpider爬虫框架主要包含了调度器、引擎、爬虫、下载器、项目管道、爬虫中间件、下载中间件和调度中间件等组成；这些功能部分各司其职，为爬虫提供了各种方法的基类以供开发者对方法进行重定义和二次开发。而且这种多个模块协同模式下，模块和模块之间的耦合度相对较低并且减少了爬虫的冗余度；降低了爬虫的后期维护成本。PySpider的这种数据的处理模式，使得爬虫能够实现相关数据流的接续传递。

1. Engine

PySpider框架的引擎主要用来管理整个项目的全部的数据流，它是PySpider框架的核心所在。

1. Scheduler

PySpider框架的调度器，用于处理调度相关的业务流程。其主要作用是来接收引擎传递过来的请求，并将请求压入到下载队列中，然后在引擎再次发出请求时返回数据，从而形成了一个下载优先队列；并由这个队列决定接下来需要下载的对象要进行哪一个。在这个过程中还需要对数据进行去重和筛选。

1. Downloader

PySpider框架的下载器主要用于页面的数据内容；同时将网页中的数据继续返还给爬虫，爬虫拿到数据后对网页中的结构化数据进行再解析；然后将解析过的数据交给爬虫的管道。即Pipeline进行数据流处理；同时爬虫解析完分析的数据后，如数据中仍旧存在二级网页中，那么爬虫将再次向引擎发送数据请求的操作。如此循环下去；项目一边下载数据一边解析数据一边继续请求数据，形成一个循环的过程；因此达到遍历目标网站所有数据的目的。

4.Spiders

PySpier框架中的爬虫在整个框架中担非常重要的角色。爬虫的主要任务就是爬取目标数据；并且完成相应数据解析、过滤和数据的存储入库。爬虫将从网页结构中提炼出自己需要的数据信息；然后爬虫对数据处理完毕后会将数据存储到事先定义好的实体（Item）中。如果从中提取出相关的URL地址，它会告诉引擎去判断是否继续抓取下一个页面。

5.Pipeline

PySpier框架中的管道，管道主要作用是爬虫对网页数据抽取后对实体信息的处理；例如数据的持久化存储和验证实体数据的有效性以及清除不必要的信息等作用。当网页被爬虫系统解析后，爬虫会将爬取结果发送至项目管道，然后由项目管道进行一定的数据处理。

## 3.2 架构设计

郑州市便民服务网站设计和实现，网络爬虫模块主要由爬虫部署和数据爬虫两个模块组成；其中爬虫部署需要借助于Linux服务器和使用redis消息队列非关系型数据库进行特定部署。redis的消息队列会先对URL种子数据进行加载存储到redis消息队列中；然后对URL种子进行相关的调度，调度到不同的服务器上进行数据的加载和存储。网络爬虫主要所使用的PySpider框架的模型图如图3-1所示。从这里可以看出PySpider框架主要由爬虫、引擎、爬虫中间件、调度器中间件、调度器、下载器、下载器中间件和项目管道等八个部分组成。爬虫具体的数据流程是：

1. PySpider引擎当拿到首次的URL的时候会将URL先交给调度器，然后调度器拿到引擎的分配后会向下载器发送数据请求。
2. 下载器向Linux服务器请求数据，并且在网络上下载网页的指定数据；
3. 下载器将加载的数据交给爬虫；
4. 爬虫对加载后的数据进行数据筛选和数据清洗，并把处理过的正确数据交给爬虫管道进行数据的持久化存储。最后将数据中存在的链接，交给PySpider引擎；
5. 引擎(Engine)拿到后续数据后，会接着交给调度器(Scheduler)；
6. 调度器(Scheduler)会继续进行数据的请求；
7. 下载器(Downloader)收到调度器(Scheduler)的后续数据请求后，就会接着下载数据；
8. 数据爬虫部分得到数据反馈后。继续对数据进行分析、筛选等操作。
9. 走到这一步爬虫、调度器和下载器形成一个循环的过程，这个循环不断的对数据进行解析请求响应工作。最终完成对目标网站数据的全部解析。

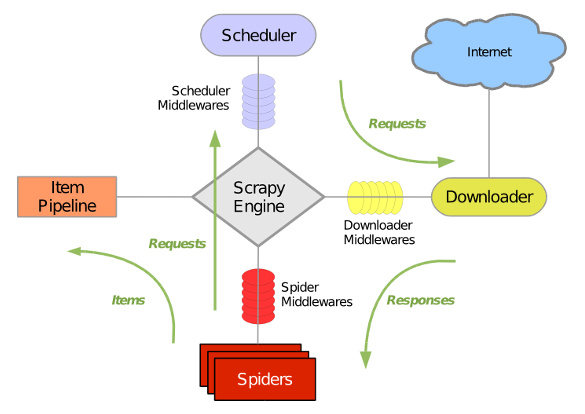


图3-1 scrapy框架的架构模型图

数据爬虫部分主要使用MongoDB非关系型数据库和MySQL关系型数据库作为数据的存储对象。网络爬虫会对下载的数据进行数据分析、数据过滤和数据入库，最终把数据存储在MongoDB和MySql数据库中；后续的数据处理流程会先从MongoDB中查询数据，然后完成之后的的数据处理操作。

## 3.3 系统关键技术

**关键技术1：PySpider框架**

**PySpider**框架，框架是由Python编程语言进行开发的一个快速、高效率的数据抓取和web爬虫框架，这个框架主要作用是抓取网络站点数据，还有的功能是可以从目标页面中分析出结构化有规律的数据。**PySpider**爬虫框架用处十分广泛，在现代互联网中它很大程度上被用于数据挖掘、数据分析和自动化测试等方向中。框架的主要的数据流程大概如下：

1. PySpider引擎当拿到首次的URL的时候会将URL先交给调度器，然后调度器拿到引擎的分配后会向下载器发送数据请求。
2. 下载器向Linux服务器请求数据，并且在网络上下载网页的指定数据；
3. 下载器将加载的数据交给爬虫。
4. 爬虫对加载后的数据进行数据筛选和数据清洗，并把处理过的正确数据交给爬虫管道进行数据的持久化存储。最后将数据中存在的链接，交给PySpider引擎；
5. 引擎(Engine)拿到后续数据后，会接着交给调度器(Scheduler)。
6. 调度器(Scheduler)会继续进行数据的请求。
7. 下载器(Downloader)收到调度器(Scheduler)的后续数据请求后，就会接着下载数据。
8. 数据爬虫部分得到数据反馈后。继续对数据进行分析、筛选等操作。
9. 到了这一步爬虫、调度器和下载器形成一个循环的过程，这个循环不断的对数据进行解析请求响应工作。最终完成对目标网站数据的全部解析。

**关键技术2：反反爬虫措施的规则制定**

随着互联网技术的日新月异，根据python编程语言的网路爬虫逐渐兴起。有些公司针对一些无规则爬虫需要进行防护；因为过多的爬虫会消耗服务器资源导致服务器资源匮乏甚至自己公司的产品收到影响。一些公司会会使用Robots爬虫协议声明本产品的数据是不可爬取的，这样就会对制定爬虫的人启到警告提型作用。但是有些公司也会对爬虫进行必要的反爬虫措施；比如对需要保护的数据进行对称加密、出现验证码策略、访问频率限制加入黑名单策略等一些反爬操作。针对不同的反爬措施我们要制定不同的反反爬措施来应对。具体措施如下：

1. 在框架中的配置文件中更改爬虫是否遵循Robots协议；默认是不遵守相此协议的。
2. 看看浏览器开发工具中的请求头来伪造请求头，模拟服务器请求，从而获取到数据。
3. 有些数据进行加密，通过分析js加密代码进行获取数据。
4. 利用伪造一些提交表单的数据来获得数据。
5. 利用以前的请求获取唯一身份标识像Cookies和Session的Id等。
6. 设置IP代理池来隐藏源IP。

反反爬虫作用是用来应对爬去网站的反爬虫措施；由此可以达到爬取目标网站大量数据的目的。

**关键技术3：数据解析规则的制定**

因为每个网站的网页结构的不确定性和多样性；所以爬虫爬取数据的时候，

页面数据的解析规则也需要针对不同的网页进行制定；为了完成对页面数据的差异化提取。现在主要制定了一下三种分析规则。

1. 纯HTML网页数据，针对这种网页我们只需要对网页进行简单解析就很容易拿到网页数据。
2. API接口返回数据，最常见的都是以JSON格式返回数据；根据数据的结构提取我们需要的信息，整理成自己的数据格式然后入库。
3. 加密数据，为了能够解密加密数据我们需要分析网站的加密规则，根据网站分析到的算法要进行加密算法的编写，有的一些一些网站还会进行文件的混淆操作；针对这样的操作我们还要分析页面中应用的字体文件来映射出正确的字体文件。如果数据没有分析出来的情况，那么要对这种情况需要模仿浏览器Selenium技术来获取数据的访问。

**关键技术4：爬虫部署**

爬虫部署实现的目的在于它能够很大程度的提高爬虫的爬取数据的效率和减少爬虫的资源消耗。另一方面扩大爬虫的数据获取，为完成爬虫部署，我们必须进行爬虫平台的构建和部署。第一需要根据不同的机器修改爬虫的配置。然后将主要的爬虫分别部署到不同的爬虫服务器上；最后利用于redis非关系数据库进行基础的URL的下载和分配操作。然后启动主爬虫程序，紧接着启动从爬虫程序。此时此刻会发现所有爬虫都会对目标网站进行数据的加载、解析和数据的清洗以及数据存储。在数据处理速度和效率上会有一个质的区别。

# 第4章 系统实现

这章的内容是主要介绍基于Python的网络爬虫项目。我们的项目爬取了多个不同平台的数据，网路爬虫主要采用PySpider框架，Redis服务和MySQL数据存储等技术。

## 4.1 新闻资讯爬取

新闻资讯的爬取主要针对今日头条和趣头条互联网平台进行爬取；根据需求爬虫需要对新闻的标题、内容、作者、发表时间、类别、浏览数量、点赞数量、评论。根据以上爬虫需要爬取的信息来分析网站的数据结构；从而设计爬虫规则。首先通过新闻自选页面列表获取每条新闻的详情页的URL，为了获得更多的详情页的URL我要对翻页的接口进行分析，通过添加请求头和Cookie信息来伪装浏览器来获取多页的数据。得到大量详情页的URL然后要对详情页URL进行爬取。分析详情页的页面数据获得以上数据。针对初步获得的数据要进行数据的清洗和筛选；爬取一些空数据和乱码的数据。

为了显示爬虫效果然在终端输入爬虫运行的命令，终端会打印出爬取的信息。然后把爬取的信息存储到数据库中。图4-1是新闻资讯爬取的页面展示爬取的数据内容。图4-2是新闻资讯存储到ElasticSearch中的图片。如图4-1所示、如图4-2：

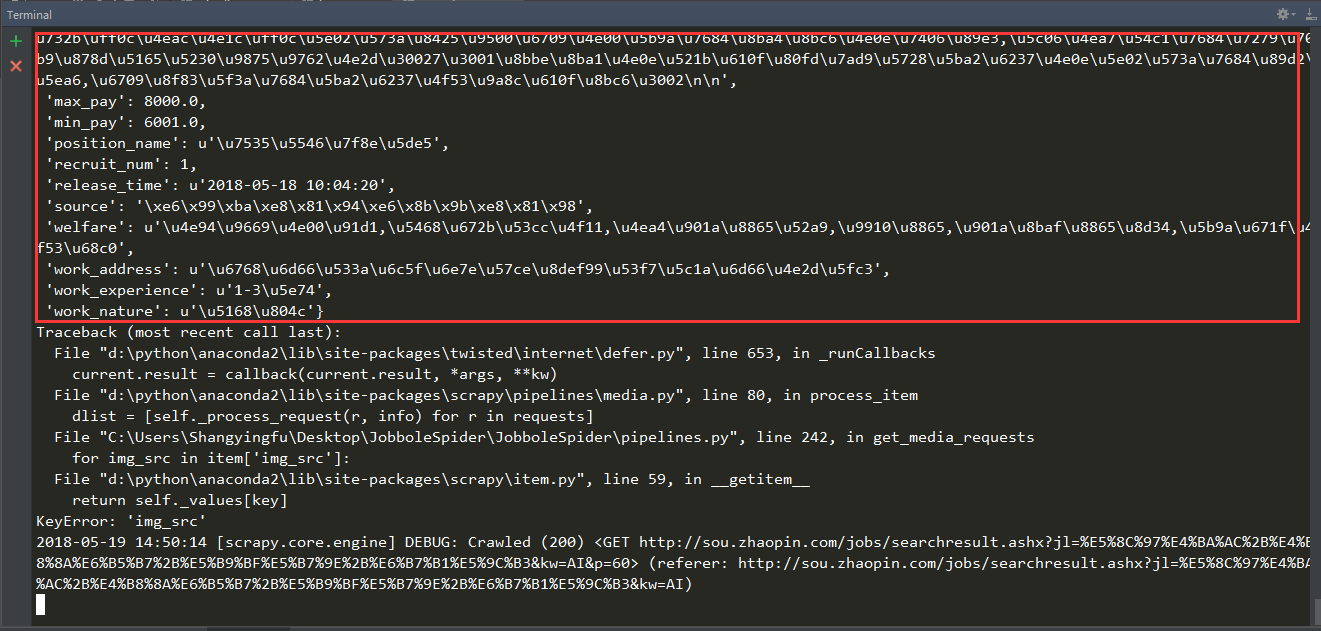


图4-1 终端爬取数据的结果

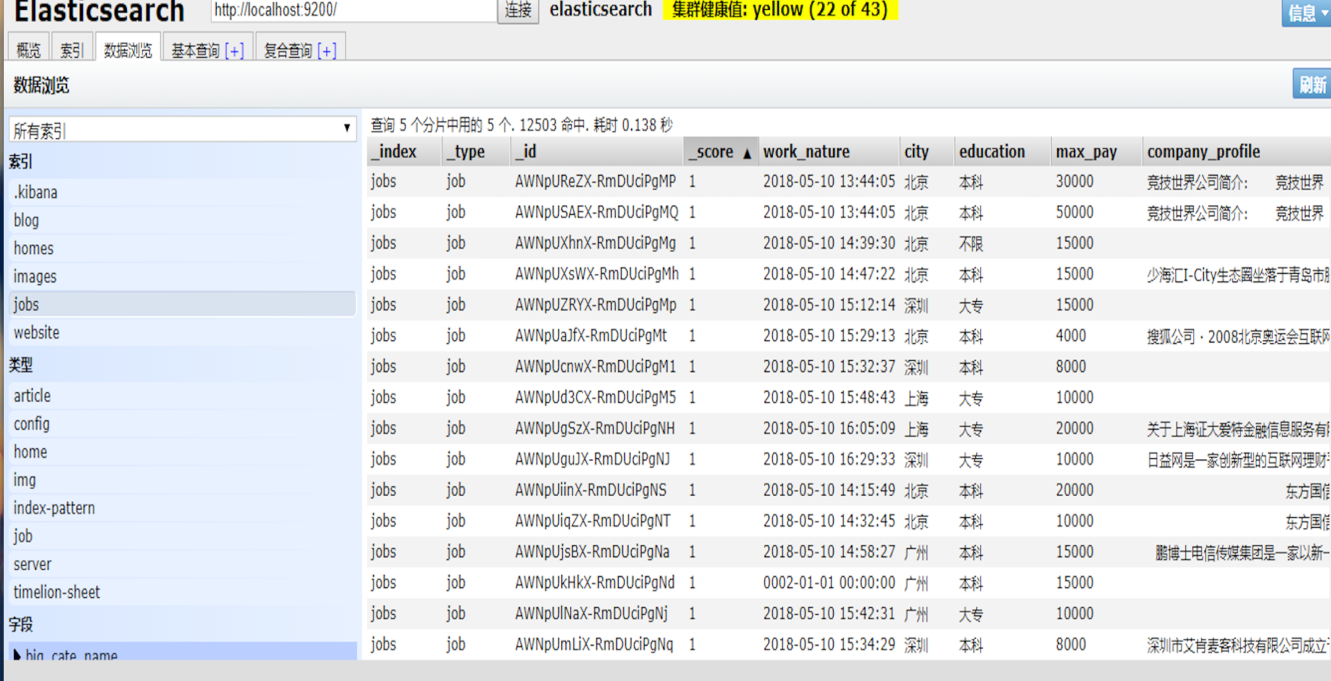


图4-2数据展示

## 4.2 租房信息爬取

基于Python的网路爬虫的租房信息的爬取，主要爬取的爱租房、58同城上的租房的信息；根据需求设计爬虫爬取规则，获得房源的名称、价格、地址、租房方式、朝向、信息简介、楼层等数据。从分析目标网站的页面拿到详情页数据会更加完整。获取数据列表页的URL更加方便。所以直接让爬虫爬取列表页数据的详情页URL，然后使用URL规则拼接完整的网址来实现跳转。按照以上数据格式要求把详情页的数据爬下来。面对爬取的初步数据有问题，比如朝向、时间、楼层等数据是放一起爬取的，在一个HTML标签内的信息，所以需要正则表达式匹配，数据清洗和过滤，根据页面信息设计匹配的规则拿到数据。之后是租房信息的下一页接口的获取，不仅仅要获取下一页的URL，还要对URL就行处理，从接口分析发现是一个列表数据形式，根据URL拼接网址爬取全部的数据。图4-4是租房信息爬虫起初拿到的数据展示，图4-5是把数据存入到elasticsearch检索库中。爬取信息如图4-4所示、如图4-5所示：

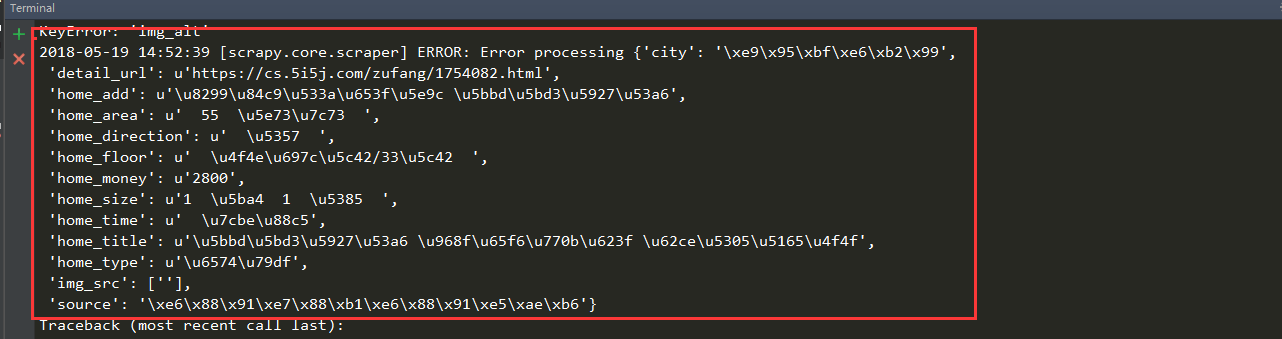


图4-4 房源信息爬取界面

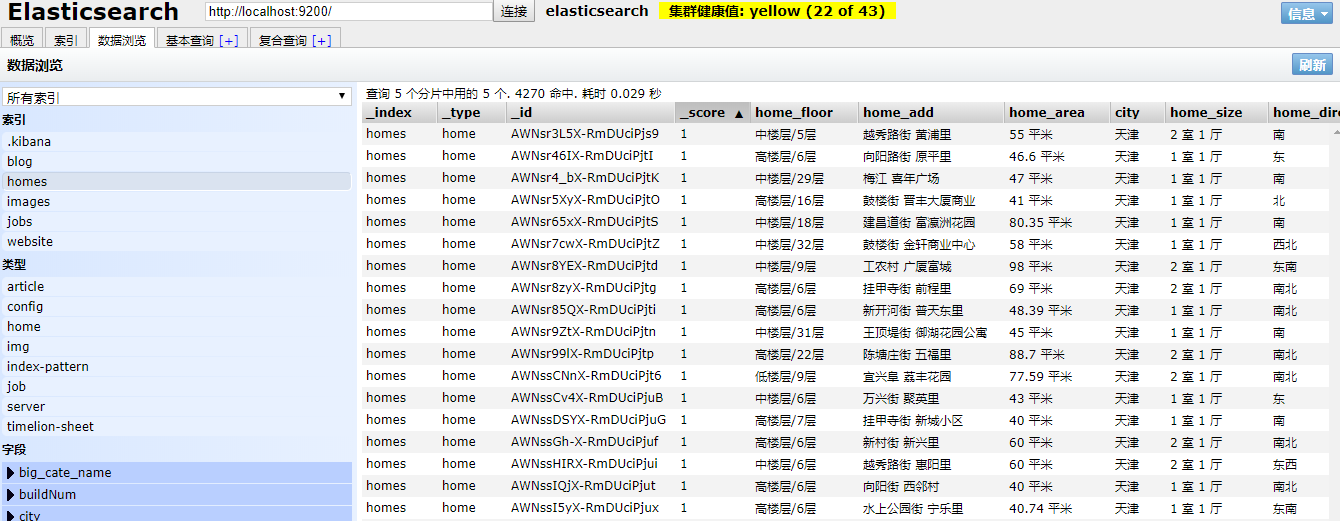


图 4-5 房源信息展示

## 4.3 图片资源爬取

根据数据展示功能模块对图片的要求，爬取图片资源的信息。根据图片的网站规则爬取，因为图片资源网站分为很多的类别的图片；网站需要的是所有图片类别，所以应该爬取所有的图片资源的信息；获取每个图片的类别。图片资源不同于其它网站数据，页面显示所有的图，但是页面显示的图片在具体的页面也有显示。所以只要获取全部图片的名字和URL，解析每个分类下的所有图片信息。最后需要爬取每张图片的名称、作者、上传时间、分类等信息。图片资源网站不仅仅要爬取所有图片的所有页面的数据，还需要爬取每个分类下所有页面的数据。图片资源数据一般不经过数据清洗，直接入库即可

当在编写爬虫过程中发现图片资源的展示遇到了无法展示的问题，原因是在项目的页面数据展示的时候会发生跨站操作错误。为了解决这个问题需要爬虫把需要的图片资源爬取下来并保存到本地；这样图片资源就不存在跨域现象了，从而实现最终的展示功能。

图4-6是图片信息资源爬取的界面，图4-7是把图片资源下载到本地的结果展示。如图4-6所示、图4-7所示：



图4-6 爬取数据

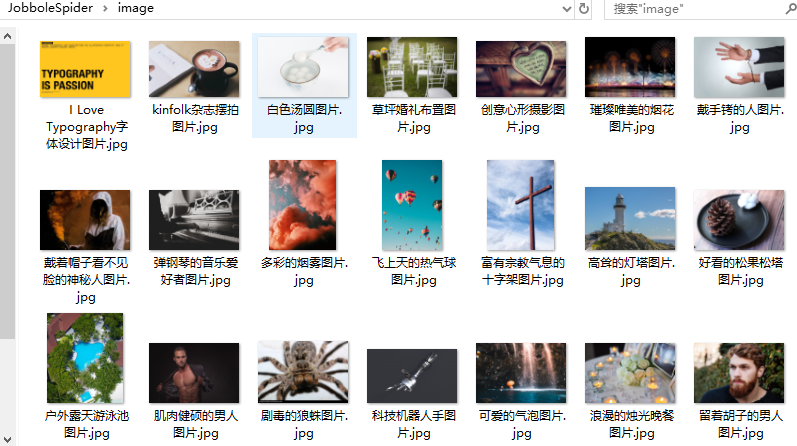


图4-7 图片本地下载

## 4.4 反反爬虫措施设计

爬虫是为数据数据展示模块提供数据支持的，因此我们需要爬取大量的数据信息。一般情况下网站都设计了反爬虫措施，当爬虫爬取一部分数据后报会发生404找不到页面的错误或者服务器403禁止错误以及IP被封等问题。面对这些反爬虫措施，为了解决这些问题设计了一些反反爬虫的应对措施，主要有如下几个措施。

(1)当爬虫访问网页的时候会有一个Headers头部请求信息，我们设计一个头部请求信息的列表；然后循环更改Headers请求头信息。爬虫模仿不同浏览器请求页面的操作可以减少爬虫问题。

(2)修改爬虫请求接口的时间间隔。一般网路爬虫项目时间是没有间隔；不间断向服务器发送网路请求。这样容易容易被目标网站发现爬虫操作。所以适当的设置请求时间间隔来模拟人点击页面浏览信息的操作，可以避免被发现爬虫行为。

(3)利用IP代理池，设置IP代理池是目前比较好用的的一种反反爬虫方式。但是需要利用免费的代理IP或者购买付费代理IP。设计一个IP代理池来存放IP；爬虫项目模仿浏览器发送请求时；随机更改IP代理。这样目标网站无法判断哪个代理是爬虫行为。IP不容易被封，如果被封，剩余的IP也可以继续爬取数据，然后IP会自动解封。

## 4.5 爬虫部署

郑州市便民服务网站设计与实现爬虫部分主要采用分布式爬虫。因为数据需求比较大要爬取多个平台的大量数据；就算使用MySQL的多线程存储，爬取所有的数据也要几天才可以完成。但是有些数据是具有时效性的，所以必须要提高爬取的效率。因此采用了分布式爬取，分布式爬取主要使用的Redis消息对垒服务的分布式功能。首先把爬虫项目中的配置文件设置为分布式，然后将爬虫代码放到多态的Linux服务器上，Linux服务器并不是越多越好，需要根据数据量来选择Linux服务器数量。最后所有服务器同时开始跑数据；在Redis终端中输入存储 网址的命令，然后开始启动分布式爬虫。

# 第5章 系统测试

## 5.1 数据爬虫

数据爬虫的数据测试用例，分布式的网络爬虫是数据爬虫核心，根据不同的规则设计不同的爬取结果，为数据展示模块提供数据支持。

### 5.1.1 数据爬虫

数据爬虫测试用例，测试数据爬虫这个功能能否爬取目标数据，如表5-1所示。

表5-1 数据爬虫测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-1 | | 编写者 | 王贺 | |
| 测试项 | 数据爬虫是否可以正常爬取数据成功 | | | | |
| 前提条件 | 打开数据爬虫项目 | | | | |
| 执行动作 | 打开数据爬虫项目的爬虫文件 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 数据爬虫 | 输入错误爬虫名称或者爬虫命令 | 爬取失败 | | | 爬取失败 |
| 正确的爬虫命令(Pyspider crawl 项目名) | 爬取成功 | | | 爬取成功 | |

### 5.1.2 新闻资讯爬虫

分布式爬虫依据不同的网站指定不同的爬取规则和新闻资讯的爬取测试用例。主要测试当前功能能否满足特定规则爬取特定的数据。如表5-2、5-3所示。

表5-2 新闻资讯爬取测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-2 | | 编写者 | 王贺 | |
| 测试项 | 新闻资讯网站是否能正常爬取数据 | | | | |
| 前提条件 | 打开数据爬虫项目 | | | | |
| 执行动作 | 打开数据爬虫中新闻资讯爬取项目 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 新闻资讯网站 | 无 | 爬取失败 | | | 爬取失败 |
| 在终端输入错误命令 | 爬取失败 | | | 爬取失败 | |
| 正确的爬虫命令(Pyspider crawl 项目名) | 打印爬取结果 | | | 打印项目的爬取结果 | |

表5-3 新闻资讯爬取指定数据测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-3 | | 编写者 | 王贺 | |
| 测试项 | 新闻资讯网站爬取指定数据 | | | | |
| 前提条件 | 链接数据库 | | | | |
| 执行动作 | 点击新闻资讯数据库 🡪 查看爬取的数据 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 新闻资讯网站 | 无 | 结果显示为空 | | | 结果显示为空 |
| 点击新闻资讯数据库 | 显示点击的新闻数据数据 | | | 显示点击的新闻数据数据 | |

### 5.1.3 租房爬取

因为不同网站和页面数据结构的不同爬取的规则和结果也都不一样。下面租房数据的爬取测试用例；主要测试当前功能能否满足特定规则爬取特定的数据。如表5-4、5-5所示。

表5-4 租房数据爬取测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-4 | | 编写者 | 王贺 | |
| 测试项 | 租房爬虫是否爬取数据 | | | | |
| 前提条件 | 打开数据爬虫爬虫项目 | | | | |
| 执行动作 | 打开数据爬虫中房源爬取项目 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 租房网站 | 无 | 爬取失败 | | | 爬取失败 |
| 在shell输入不符合爬虫名称 | 爬取失败 | | | 爬取失败 | |
| 正确的爬虫命令(Pyspider crawl 项目名) | 打印项目爬取结果 | | | 打印项目的爬取结果 | |

表5-5 获取指定租房数据测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-5 | | 编写者 | 王贺 | |
| 测试项 | 租房信息爬取特定需要的数据 | | | | |
| 前提条件 | 链接数据库 | | | | |
| 执行动作 | 点击租房信息数据库 🡪 点击查看爬取数据 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 租房网站 | 无 | 结果显示为空 | | | 结果显示为空 |
| 点击查看信息信息爬取结果 | 显示点击的租房信息列表 | | | 显示点击的租房信息列表 | |

### 5.1.4 问答平台爬取

爬虫平台的爬取测试用例主要测试当前功能能否满足特定规则爬取特定的数据。爬取的数据是否满足需求。问答平台内容爬取不仅仅要获取所需要的数据，还需要获取源码的数据，显示提供数据的出处。如表5-6、5-7所示。

表5-6 问答网站爬虫测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-6 | | 编写者 | 王贺 | |
| 测试项 | 问答爬虫是否爬取数据 | | | | |
| 前提条件 | 打开数据爬虫项目 | | | | |
| 执行动作 | 打开数据爬虫中问答平台爬虫的项目 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 问答爬虫 | 无 | 爬取失败 | | | 爬取失败 |
| 在shell输入不符合爬虫名称 | 爬取失败 | | | 爬取失败 | |
| 正确的爬虫命令(Pyspider crawl 项目名) | 打印项目爬取结果 | | | 打印项目爬取结果 | |

表5-7 获取指定问答平台数据测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-7 | | 编写者 | 王贺 | |
| 测试项 | 问答平台爬虫是否爬取指定数据 | | | | |
| 前提条件 | 链接数据库 | | | | |
| 执行动作 | 点击问答平台的数据库 🡪 查看爬取结果 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 问答爬虫 | 无 | 没有结果显示 | | | 没有结果显示 |
| 点击问答平台信息的数据库 | 显示指定的问答平台的数据 | | | 显示指定的问答平台的数据 | |

### 5.1.5 图片爬取

图片信息的爬取测试用例主测试当前功能能否满足特定规则爬取特定的数据。如表5-10、5-11所示。

表5-10 图片爬虫测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-10 | | 编写者 | 王贺 | |
| 测试项 | 图片爬虫是否爬取图片数据 | | | | |
| 前提条件 | 打开数据爬虫项目 | | | | |
| 执行动作 | 打开数据爬虫中图片爬虫项目 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 图片信息 | 无 | 爬取失败 | | | 爬取失败 |
| 在shell输入错误命令 | 爬取失败 | | | 爬取失败 | |
| 正确的爬虫命令(Pyspider crawl 项目名) | 打印项目爬取结果 | | | 打印项目的爬取结果 | |
|  |  |  | | |  | |

表5-11 图片数据测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-1-11 | | 编写者 | 王贺 | |
| 测试项 | 图片爬虫是否爬取指定数据 | | | | |
| 前提条件 | 链接数据库 | | | | |
| 执行动作 | 点击图片信息数据库 🡪 查看爬取结果 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 图片信息 | 无 | 没有结果显示 | | | 没有结果显示 |
| 选择图片信息的数据库 | 显示指定的图片的数据 | | | 显示指定的图片的数据 | |

## 5.2 反反爬虫

测试这个环节需要连贯整个项目发展过程，每个功能模块都需要测试，爬虫系统才可以没有错误的正常运行。

### 5.2.1 伪装浏览器

伪装浏览器的测试用例，测试这个功能模块是否正常运行。是否能够正常爬取目标网站的页面并且不发生异常。伪装浏览器的测试用例如表5-12所示。

表5-12 伪装浏览器测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-1 | | 编写者 | 王贺 | |
| 测试项 | 伪装浏览器 | | | | |
| 前提条件 | 获得爬虫项目的网页中浏览器headers信息 | | | | |
| 执行动作 | 打开数据爬虫配置文件配置请求头文件 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 伪装浏览器 | 无 | 没有成功伪装 | | | 没有成功伪装 |
| 爬虫命令(Pyspider crawl 项目名) | 页面爬取并且打印结果爬取数据的结果 | | | 页面爬取并且打印结果爬取数据的结果 | |

### 5.2.2 改变请请求时间间隔

改变请求时间间隔测试用例主要测试爬虫的效率是否降低。如表5-13和表5-14所示。

表5-13 修改变请求时间间隔的测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-2 | | 编写者 | 王贺 | |
| 测试项 | 改变请求时间间隔能否爬虫 | | | | |
| 前提条件 | 修改爬虫配置文件中的爬虫时间配置 | | | | |
| 执行动作 | 输入爬虫命令爬取数据 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 改变请求时间间隔 | 无 | 没有信息展示 | | | 没有信息展示 |
| 爬虫命令(Pyspider crawl 项目名) | 显示爬虫爬取数据的结果 | | | 显示爬虫爬取数据的结果 |

表5-14 改变请求时间间隔测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-3 | | 编写者 | 王贺 | |
| 测试项 | 改变请求时间间隔爬取速度放慢 | | | | |
| 前提条件 | 配置文件中请求时间间隔默认值已经修改 | | | | |
| 执行动作 | 在shell输入命令 爬取结果 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 改变请求时间间隔 | 爬虫命令(Pyspider crawl 项目名) | 在shell显示爬取速度明显放慢 隔规定时间爬取一条数据 | | | 在shell显示爬取速度明显放慢 隔规定时间爬取一条数据 |

### 5.2.3 设置Cookie模拟登录

设置Cookie模拟登录的测试用例主要测试当前的请求是否可以跳过登录限制直接访问。如表5-15、5-16所示。

表5-15 Cookie模拟登录测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-4 | | 编写者 | 王贺 | |
| 测试项 | 分析Cookie信息登录 | | | | |
| 前提条件 | 爬取一个有登录限制的网站 | | | | |
| 执行动作 | 打开数据爬虫项目运行 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| Cookie模拟登录 | 无 | 登录失败 | | | 登录失败 |
| 爬虫命令(Pyspider crawl 项目名) | 跳过登录，项目正常运行 | | | 跳过登录，项目正常运行 |

### 5.2.4 IP代理池

使用IP代理池反反爬虫的测试用例主要测试IP是否使用设置得IP代理池直爬取数据。如表5-16所示。

表5-16 IP代理测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-2-6 | | 编写者 | 王贺 | |
| 测试项 | IP代理池反爬虫 | | | | |
| 前提条件 | 获取免费IP代理或者购买付费的IP代理 | | | | |
| 执行动作 | 爬虫输入命令开始爬取结果 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| IP代理池 | 正确的爬虫命令(Pyspider crawl 项目名) | 爬取结果会显示不停再切换IP，表示在切换IP代理 | | | 爬取结果会显示不停再切换IP，表示在切换IP代理 |

## 5.3 数据入库

MySQL数据库存储数据的测试用例，爬虫爬取的数据会先将数据存储到服务器上得MySQL数据库，然后检查爬取得数据是否满足项目要求；如果满足项目要求后会将数据存到MongoDB非关系数据库中进行二次利用。MySql数据库测试用例主要测试MySql数据库中是否有根据规则爬取的目标网站的数据，爬取的数据是否存在数据为空和乱码的问题。MySQL数据库测试用例如表5-17所示。

表5-17 MySQL数据库存储显示测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-3-1 | | 编写者 | 王贺 | |
| 测试项 | MySQL数据库存储是否有数据 | | | | |
| 前提条件 | 连接链接数据库查看信息 | | | | |
| 执行动作 | 查看数据库中数据信息 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| MySQL数据库 | 无 | 没有任何数据结果显示 | | | 没有任何数据结果显示 |
| 点击新闻资讯数据库 | 展示新闻资讯的数据 | | | 展示新闻资讯的数据 |
| 查看租房数据库 | 显示租房的数据 | | | 显示租房的数据 |
| 查看问答平台数据库 | 显示问答平台数据 | | | 显示问答平台数据 |
| 查看图片数据库 | 显示图片的信息和数据 | | | 显示图片的信息和数据 |

## 5.4 分布式爬虫

分布式爬虫的测试用例主要测试这个爬虫项目可以不可以同时分别在不同的服务器上运行；这样的目是为了提高数据的获取速度。如下表所示。

表5-18 分布式部署测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-4-1 | | 编写者 | 王贺 | |
| 前提条件 | 多台服务器 | | | | | |
| 执行动作 | 开启Redis消息队列服务和设置配置文件 | | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 分布式爬虫 | 输入不正确Redis命令 | 开启服务失败 | | | 开启服务失败 |
| 开启Redis服务  (redis-server) | 开启服务成功 | | | 开启服务成功 |

表5-21 分布式爬虫测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | FUN-4-2 | | 编写者 | 王贺 | |
| 测试项 | 分布式爬虫爬取数据 | | | | |
| 前提条件 | 多台Linux服务器 | | | | |
| 执行动作 | 开启Redis服务配置分布式配置数据库 | | | | |
|  | 输入数据/动作 | 预期结果 | | | 实际结果 |
| 分布式爬虫 | 无 | 爬取失败 | | | 爬取失败 |
| Redis输入分布式爬虫的爬取的命令(lpush 爬取网址) | Linux服务器显示爬取数据，并且数据不重复 | | | Linux服务器显示爬取数据，并且数据不重复 |

# 结束语

时间如白驹过隙，我的大四就快要毕业了。本次的毕业设计课题也是我大学生活中最后一次的课题。回顾过去几年的大学生活以及课题的经历，深刻体会到了学校每学期设定课题的良苦用心；通过多次做课题的经验积累，毕业设计的做的游刃有余。通过这次我们小组完成毕业设计的过程，我们体会到了团队合作的重要性以及编程中的乐趣，同时也让我学到了新的技术，明白了学无止境。

从向老师推出想要写这个项目的时候，我们就开始规划这个项目的架构。从项目开始到完成的过程中我们付出了很大的精力。特别是我在衡量公司的工作和学校的事情显得有点不知所措，但是在最后老师的帮助下慢慢的调整好了这一段时间的心态。明白了做事不能操之过急，需要心平气和静下心来慢慢理清事情的思路；才能慢慢的达到目标。

这次的毕业设计项目中我采用了近两年比较流行的编程语言Python,；这种编程语言相对而言还是比较好学的，在我大三的时候我就已经接触过了。项目中我主要负责的是数据爬取模块，而Python中的对数据处理的能力还是非常不错的，其中爬虫框架PySpider是我们项目中主要使用的框架。在我们配置整个项目框架中其实还是遇到很多问题的，当项目报错的时候着急找不到解决原因的时候，我们会彼此交互对项目的理解和建议；经过一系列的挫折我们也对Python也有了深刻的理解，PySpider框架也有了进一步的认识。比如说当我们爬取58同城的租房信息的时候一些加密数据真的找不到规律，真的是不知道从哪里开始下手破解数据，最近和小伙们一起商量最终还是耐心下来找到了解决的方法。

从这次的毕业设计中，让我看到了自身的在一些方面的不足。这让我懂得了在以后的工作中，需要不断的学习新的知识技术才能不落后时代。我认为通过自身一步一个脚印的努力，知识也会一点一滴的积累；也让我明白，在我们的生活中，即使遇到艰难险阻，也不可以放弃，因为只要勇敢面对，保持积极的心态，百折不挠，相信遇到的问题一定会迎刃而解的。

# 参考文献

[1] 李琳. 基于Python的网络爬虫系统的设计与实现[J]. 信息通信,2017(9):26-27.

[2] 郭丽蓉. 基于Python的网络爬虫程序设计[J]. 电子技术与软件工程, 2017(23):248-249.

[3] 姜杉彪, 黄凯林, 卢昱江,等. 基于Python的专业网络爬虫的设计与实现[J]. 企业科技与发展, 2016(8):17-19.

[4] 戚利娜, 刘建东. 基于Python的简单网络爬虫的实现[J]. 电脑编程技巧与维护, 2017(8):72-73.

[5] 高强. 基于Redis的分布式爬虫框架的设计[J]. 农业网络信息, 2017(8).

[6] 李代祎, 谢丽艳, 钱慎一,等. 基于Scrapy的分布式爬虫系统的设计与实现[J]. 湖北民族学院学报(自然科学版), 2017, 35(3):317-322.

[7] 林伟坚. 基于Scrapy框架的新闻实时抓取及处理系统的设计与实现[D]. 南开大学, 2012.

[8] 耿大伟. 基于Python技术的校园网搜索引擎的设计与实现[D]. 燕山大学, 2015.

[9] 刘志凯, 张太红. Django框架在web开发中的应用[J]. 农业网络信息, 2015(2):51-52.

[10] Gormley C, Tong Z. Elasticsearch: The Definitive Guide[M]. O'Reilly Media, Inc. 2015.

[11] Booch, Grady et al., 面向对象分析与设计[D], 人民邮电出版社, 2009.

[12] thepythonguru.com, Python Operator Overloading - The Python[D]. Guru, 2015.

[13] Python Software Foundation, 6. Modules — Python 3.5.1[D]. documentation, 2015.

[14] Gormley C, Tong Z. Elasticsearch the definitive guide[J]. Oreilly Media, 2015.

[15] Bennett, James. Practical Django Projects[J]. 2008, 9(1):77–81.

[16] Seidman C, Smith P. MySQL: The Complete Reference[M]. McGraw-Hill, Inc. 2003.

附录

## 附录A 郑州市便民服务网站设计与实现爬虫模块的核心代码

部分代码

# -\*- coding: utf-8 -\*-

import scrapy

from scrapy import Selector, signals, Request

from selenium import webdriver

from scrapy.xlib.pydispatch import dispatcher

from zufangspider.items import HouseItem

class LianjiaSpider(scrapy.Spider):

name = 'lianjia'

allowed\_domains = ['cd.lianjia.com']

start\_url = 'https://cd.lianjia.com/zufang/pg{page}'

def start\_requests(self):

for i in range(1, 101):

url = self.start\_url.format(page=i)

yield Request(url=url, callback=self.parse)

def parse(self, response):

result = Selector(response)

result\_lis = result.xpath('//\*[@id="house-lst"]/li')

for li in result\_lis:

item = HouseItem()

item['title'] = li.xpath('./div[2]/h2/a/text()').extract\_first()

item['house\_url'] = li.xpath('./div[2]/h2/a/@href').extract\_first()

location = li.xpath('./div[2]/div[1]/div[1]/a/span/text()').extract\_first()

item['location'] = location.strip()

item['source'] = '链家租房'

yield item

import logging

import requests

from fake\_useragent import UserAgent

from scrapy.http import HtmlResponse

class RequestsChrometmiddware(object): # 浏览器访问中间件

def process\_request(self, request, spider): # 重写process\_request请求方法

if spider.name == 'lvmama': # 判断爬虫名称为pach时执行

spider.browser.get(request.url) # 用谷歌浏览器访问url

import time

time.sleep(3)

print('访问：{0}'.format(request.url)) # 打印访问网址

# 设置响应信息，由浏览器响应信息返回

return HtmlResponse(url=spider.browser.current\_url, body=spider.browser.page\_source, encoding='utf-8',

request=request)

class ProxyMiddleware(object):

def \_\_init\_\_(self, proxy\_url):

self.logger = logging.getLogger(\_\_name\_\_)

self.proxy\_url = proxy\_url

def get\_random\_proxy(self):

try:

response = requests.get(self.proxy\_url)

if response.status\_code == 200:

proxy = response.text

return proxy

except requests.ConnectionError:

return False

def process\_request(self, request, spider):

# if request.meta.get('retry\_times'):

# proxy = self.get\_random\_proxy()

proxy = '101.236.60.52:8866'

if proxy:

uri = 'https://{proxy}'.format(proxy=proxy)

self.logger.debug('使用代理 ' + proxy)

request.meta['proxy'] = uri

@classmethod

def from\_crawler(cls, crawler):

settings = crawler.settings

return cls(

proxy\_url=settings.get('PROXY\_URL')

)

class RandomUserAgentMiddleware(object): # 自定义浏览器代理中间件

# 随机更换Requests请求头信息的User-Agent浏览器用户代理

def \_\_init\_\_(self, crawler):

# 获取上一级父类基类的，\_\_init\_\_方法里的对象封装值

super(RandomUserAgentMiddleware, self).\_\_init\_\_()

# 实例化浏览器用户代理模块类

self.ua = UserAgent()

# 获取settings.py配置文件里的RANDOM\_UA\_TYPE配置的浏览器类型，

# 如果没有，默认random，随机获取各种浏览器类型

self.ua\_type = crawler.settings.get('RANDOM\_UA\_TYPE', 'random')

@classmethod # 函数上面用上装饰符@classmethod，函数里有一个必写形式参数cls用来接收当前类名称

def from\_crawler(cls, crawler): # 重载from\_crawler方法

return cls(crawler) # 将crawler爬虫返回给类

def process\_request(self, request, spider): # 重载process\_request方法

def get\_ua(): # 自定义函数，返回浏览器代理对象里指定类型的浏览器信息

return getattr(self.ua, self.ua\_type)

request.headers.setdefault('User-Agent', get\_ua()) # 将浏览器代理信息添加到Requests请求

import pymongo

import pymysql

from pymysql import IntegrityError

from scrapy import Request

from scrapy.exceptions import DropItem

from scrapy.pipelines.images import ImagesPipeline

class MongoPipeline(object):

def \_\_init\_\_(self, mongo\_uri, mongo\_db):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

@classmethod

def from\_crawler(cls, crawler):

return cls(

mongo\_uri=crawler.settings.get('MONGO\_URI'),

mongo\_db=crawler.settings.get('MONGO\_DB')

)

def open\_spider(self, spider):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

def process\_item(self, item, spider):

name = item.collection

self.db[name].insert(dict(item))

return item

def close\_spider(self, spider):

self.client.close()

class MysqlPipeline(object):

def \_\_init\_\_(self, host, database, user, password, port):

self.host = host

self.database = database

self.user = user

self.password = password

self.port = port

@classmethod

def from\_crawler(cls, crawler):

return cls(

host=crawler.settings.get('MYSQL\_HOST'),

database=crawler.settings.get('MYSQL\_DATABASE'),

user=crawler.settings.get('MYSQL\_USER'),

password=crawler.settings.get('MYSQL\_PASSWORD'),

port=crawler.settings.get('MYSQL\_PORT'),

)

def open\_spider(self, spider):

self.db = pymysql.connect(self.host, self.user, self.password, self.database, charset='utf8',

port=self.port)

self.cursor = self.db.cursor()

def close\_spider(self, spider):

self.db.close()

def process\_item(self, item, spider):

try:

print(item['title'])

data = dict(item)

keys = ', '.join(data.keys())

values = ', '.join(['%s'] \* len(data))

sql = 'insert into %s (%s) values (%s)' % (item.table, keys, values)

self.cursor.execute(sql, tuple(data.values()))

self.db.commit()

except IntegrityError:

pass

return item

class ImagePipeline(ImagesPipeline):

def file\_path(self, request, response=None, info=None):

url = request.url

file\_name = url.split('/')[-1]

return file\_name

def item\_completed(self, results, item, info):

image\_paths = [x['path'] for ok, x in results if ok]

if not image\_paths:

raise DropItem('Image Downloaded Failed')

return item

def get\_media\_requests(self, item, info):

yield Request(item['url'])