# 基于 Scrapy 技术的数据采集系统的设计与实现

杨 君,陈春玲,余 瀚
(南京邮电大学 计算机学院,江苏 南京 210003)

**摘 要**:面对互联网信息极其庞大并且经常更新的问题,基于 Scrapy 爬虫框架设计并实现了一种数据采集系统。不仅可以根据用户自身需求获取数据,还可以对自身的采集任务进行简单的管理。介绍了系统开发的关键技术,探讨了系统框架设计、功能模块和数据库设计方案。使用 Django MTV 模式进行开发,底层数据采集框架使用 Scrapy,一种使用 Python 编写实现的网站数据异步爬虫应用框架,网页解析采用 XPath 和 Python 正则相结合的方法,采用 jQuery 树插件 zTree 实现了任务的树形管理,使用 bootstrap 实现了数据的任务名加关键字组合查询和页面效果。系统主要分为网页解析模块、数据处理模块、系统登录模块、任务新建模块、任务管理模块和数据查询模块。最后分析了浏览器端和服务器端的数据交互,以及网页数据定位和解析的实现。

**关键词**:Scrapy;Django;数据采集;网络爬虫
**中图分类号**:TP302 **文献标识码**:A **文章编号**:1673-629X(2018)10-0177-05
doi:10.3969/j.issn.1673-629X.2018.10.037

# Design and Implementation of Data Acquisition System Based on Scrapy Technology

YANG Jun,CHEN Chun-ling,YU Han
(School of Computer,Nanjing University of Posts and Telecommunications,
Nanjing 210003,China)

**Abstract**:For the huge and frequent updating of the Internet information,we design and implement a data acquisition system based on the Scrapy crawler framework,which can not only obtain data according to the user's own needs,but also manage its own collection tasks simply. The key technology of system development is introduced,and the frame design,function module and database design scheme of the system are discussed. The Django MTV mode is used for development,and the underlying data collection framework applies Scrapy,an asynchronous crawler application framework implemented by Python. The web page analysis uses the method in combination of XPath and Python regular. The jQuery zTree plug-in is utilized to realize tree management of tasks,the bootstrap to achieve the effect of task name with the keyword combination query and page. The system is divided into web page analysis module,data processing module,system login module,task module,task management module and data query module. Finally,the realization of data interaction between browser and server,and the web page data positioning and analysis are analyzed.

**Key words**:Scrapy;Django;data acquisition;Internet crawler

## 0 引 言

互联网在人们的生活中无处不在,通过互联网,人们可以便捷、高效地获取所需要的各种各样的信息[1]。然而在信息大爆炸式发展的今天,互联网上的网络数据量以其惊人的速度呈几何级增长。就网页数据而言,每天都如雨后春笋般涌出。互联网上的各种信息在带来便利的同时,因其数据繁杂且数据量众多,也带来了信息过载的问题。因此,面对互联网上的海量数据,如何快速且高效地获取所需要的数据是一个迫切需要解决的问题。

基于 Scrapy 框架设计的数据采集系统,可以高效、准确地获取所需要的数据资源。根据用户指定的主网及数据类型关键字,爬取所需要的数据,并且可以将获取到的数据进行清洗和分类。高效地数据获取、

**收稿日期**:2017-08-14 **修回日期**:2017-12-28 **网络出版时间**:2018-05-16
**基金项目**:国家自然科学基金(11501302)
**作者简介**:杨 君(1992-),男,硕士研究生,研究方向为软件工程;陈春玲,硕士,教授,硕导,研究方向为软件工程、分布式组件技术、网络信息安全及其应用;余 瀚,博士,副教授,硕导,研究方向为科学计算、图像处理、信息安全等。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20180515.1645.004.html

数据的实时性、数据的准确性对用户来说都具有十分重要的实际意义。

## 1 关键技术

### 1.1 Scrapy

Scrapy 是用 Python 编写实现的网站数据异步爬虫应用框架。用户可以根据需求进行修改,使用起来简单轻巧因而用途十分广泛,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领域。Scrapy 内部实现了包括并发请求、免登录、URL 去重等多种复杂的操作。Scrapy 设计理念先进而简单,效率高,可扩展性好,可移植性高,在主流的操作系统平台上都具有良好的性能[2]。

Scrapy 主要由 5 个部分组成:Scrapy Engine(Scrapy 引擎)、Scheduler(调度器)、Spiders(蜘蛛)、Item Pipeline(数据处理流水线)和 Downloader(下载器)[3]。爬取过程是调度器把初始 URL 交给下载器;下载器向网络服务器发送服务请求,得到响应后将下载的数据转交给蜘蛛;蜘蛛一般是用户自己定义的程序,蜘蛛会对网页进行详细的分析,分析得到的待爬取链接会请求调度器;调度器再次调度处理这些 URL,分析得到的数据,会转交给数据处理流水线继续处理[4]。Item 会定义数据输出格式等,最后由 Pipeline 输出到文件中,或者存到数据库中等。总之数据处理流水线对数据进行的后期处理包括过滤、去重和存储等[5]。

### 1.2 Django

Django 框架是基于 Python 语言编写的一个开源免费的 Web 应用框架。Django 具备较为完美的模版机制、对象关系映射机制,还能够创建出动态管理后台信息的界面。相较于其他基于 Python 的框架,Django 框架更加注重组件之间的可重复利用性和可插拔性,可以相对简单地开发出功能复杂的带有数据库驱动的网站,以及遵守敏捷开发原则和 DRY 法则[6]。

Django 框架是一个松耦合、高内聚的框架,每个由 Django 驱动的 Web 应用都有着明确的目的,并且可独立更改而不影响到其他部分,可以让开发人员更专注于编写清晰、易维护的代码和应用程序的开发。它可以运行在启用了 mod_python 或 mod_wsgi 的 Apache2 上,或者任何兼容 WSGI 的 Web 服务器。一个 Django 工程中,主要分为 3 个 Python 的文件(urls.py、views.py、models.py)和 html 模板文件(index.html)。urls.py 指出了 URL 调用 views.py 中所调用的视图;views.py 用来定义程序的业务逻辑,主要实现 URL 的分发处理和定义处理 Web 请求的函数;models.py 定义了程序的数据存储结构。index.html 是 html 模板,它描述了这个 html 页面是如何设计的。

Django 的基本结构是 MTV(model, templates, view),其中模型模块(model)负责与数据有关的全部事务处理,包含数据存取、数据验证、数据行为以及提供访问数据库的 API[7]。视图模块(view)是业务逻辑处理模块,负责模型的存取和正确调用模板,用来定义如何渲染 Templates,是联系模型和模板的纽带。模板模块(templates)用来定义数据的显示格式,负责与展现相关的事务[8]。Django 的 MTV 模式实际上只实现了普通 MVC 模式中的 M 和 V。Django 的控制器是框架本身,框架将根据 Django 的 URL 配置向合适的视图函数发送请求。

## 2 系统设计

### 2.1 系统分层架构设计

系统采用典型的 B/S(browser/server)架构(见图 1),分为三层:前端、后端、数据库。

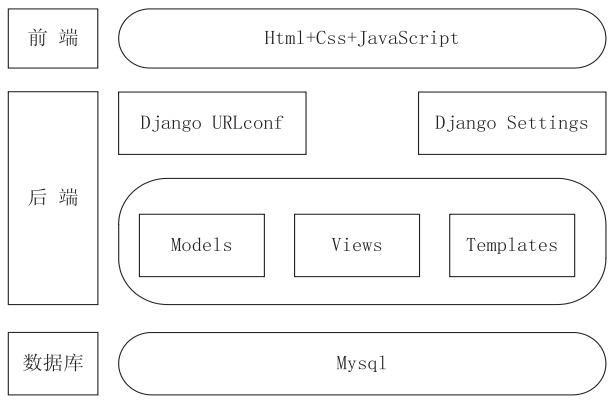


图 1 系统分层架构

前端采用典型 Html、Css、JavaScript 技术来呈现网页,Css 使用 bootstrap 样式库,JavaScript 使用 jQuery,并且使用 Ajax 异步刷新技术。

该系统的架构设计完全基于 Django 的 MTV 架构实现。Django URLconf 实现了 Http 请求的 URL 匹配功能,寻找对应的视图函数。Django Setting 配置了整个系统的设置。

Models 与数据库 Mysql 实现 ORM 关系映射,不需要关注数据库中数据的结构层次,直接通过 Models 对象来操作数据库。

### 2.2 系统功能设计

按照用户使用的方便性和对系统需求的分析,基于 Scrapy 的数据采集系统包含服务器端和浏览器端功能模块,如图 2 所示。

#### 2.2.1 服务器端功能模块

(1)网页解析模块:当抓取网页数据时,需要从访问到的 HTML 源码中提取数据。Scrapy 提取数据有自己的一套机制,它们被称作选择器(selectors),因为

它们通过特定的 XPath 或正则表达式来“选择”HTML 文件中的某个部分。而这里每个字段的 XPath 表达式和 Python 正则表达式都是在任务新建模块进行配置的。

(2)数据处理模块:当一个任务完成后,该模块会将得到的数据进行简单清洗,比如删除重复的数据,并且将数据关联到它们所在的任务种类,方便后面的数据查询。

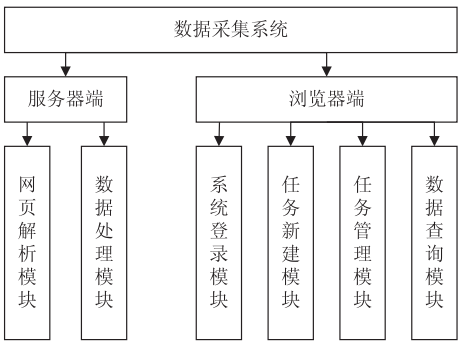


图 2　系统功能结构

2.2.2　浏览器端功能模块

(1)系统登录模块:判断客户端用户的合法性,如果输入的用户验证信息正确,就可以进入自己的任务管理模块。新用户可以点击“用户注册”进入注册界面,完成自己信息录入。

(2)任务新建模块:在该模块可以新建自己的子任务,并且归类到自己的任务树中。需要配置的信息包括任务名称、任务的主网地址以及该任务所属父节点,然后新建需要爬取的字段,包括字段名称、XPath 表达式和正则表达式。

(3)任务管理模块:在该模块,所有的任务和任务类名会形成一棵任务树。根节点是用户名,用户可以根据自己的需要,手动新建或删除叶子节点,以增加、删除任务类别。当删除父节点时,所有的子节点都会被删除,其中的任务类别和子任务都会被删除。还可以通过手动拖动节点,改变任务间关系。

(4)数据查询模块:用户可以通过输入任务名以及数据关键字进行组合搜索,搜索到的数据可以根据需要设置成一页展示的数据条目数,如一页五条、一页二十条或一页一百条等等。

## 2.3　数据库设计

在当前应用环境中,运用合理的方法设计并抽象出最佳的数据模型,并且在此基础上搭建上层应用系统,在此基础上创建的数据库能有效地存取数据,满足系统需求和用户需求。根据该数据采集系统中的设计,采用 MySql 数据库。

用户信息表中包括用户名、用户邮箱、用户号码、用户创建时间和更新时间,如表 1 所示。

表 1　用户信息表结构

| 字段名称 | 数据类型 | 备注 |
|---|---|---|
| username | 文本 | 用户名 |
| email | 文本 | 用户邮箱 |
| phonenumber | 数字 | 用户号码 |
| createtime | 文本 | 创建时间 |
| updatetime | 文本 | 更新时间 |

任务信息表包括任务英文名、任务中文名、任务目标网址、网址关键字、任务创建时间和任务更新时间(见表 2),其中网址关键字用于任务进行时 url 的爬取筛选,这样爬取的 url 具有针对性,可以过滤大量冗余的 url。

表 2　任务信息表结构

| 字段名称 | 数据类型 | 备注 |
|---|---|---|
| usertask | 文本 | 任务英文名 |
| usertask_zh | 文本 | 任务中文名 |
| start_url | 文本 | 任务目标网址 |
| allowed_url | 文本 | 网址局限 |
| url_xpath | 文本 | 网址关键字 |
| createtime | 文本 | 创建时间 |
| updatetime | 文本 | 更新时间 |

字段表结构包括字段英文名、字段中文名、XPath 表达式、正则表达式、创建时间和更新时间(见表 3),其中 XPath 表达式用来定位网页数据位置,正则表达式是根据需要解析数据的。

表 3　字段表结构

| 字段名称 | 数据类型 | 备注 |
|---|---|---|
| fieldname | 文本 | 字段英文名 |
| fieldname_zh | 文本 | 字段中文名 |
| xpath | 文本 | XPath 表达式 |
| re | 文本 | 正则表达式 |
| createtime | 文本 | 创建时间 |
| updatetime | 文本 | 更新时间 |

任务树表包括节点名称、父节点名称、创建时间和更新时间(见表 4),其中根节点就是用户名,剩余非叶子节点的父节点是用户的任务类型,叶子节点是用户的单个采集任务,名称即任务名。

表 4　任务树表结构

| 字段名称 | 数据类型 | 备注 |
|---|---|---|
| codename | 文本 | 节点名称 |
| codefather | 文本 | 父节点名称 |
| createtime | 文本 | 创建时间 |
| updatetime | 文本 | 更新时间 |

## 3 系统功能实现

### 3.1 浏览器端与服务器端数据交互的实现

Django 采用 MTV 开发模式,在该数据采集系统中,一个完整的请求处理过程如图 3 所示。

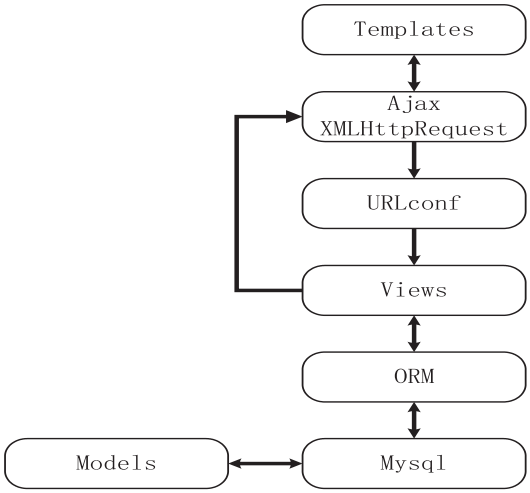


图 3 数据交互处理过程

(1)用户在浏览器端填写好表单,前台使用 jQuery 进行表单验证,验证通过后将表单数据做成 Json 格式;

(2)jQuery 启动 Ajax 引擎,使用 XMLHttpRequest 发送 Http 请求;

(3)请求通过 Django 框架的 urls. py 中定义的路由表,匹配对应 URL,请求到达 URL 对应的视图;

(4)调用视图中的相关请求处理函数,对获取到的请求数据进行处理;

(5)视图中的方法可以通过 ORM 访问底层数据库,进行操作;

(6)再次通过 XMLHttpRequest 对象获取服务器数据;

(7)模板获取数据,浏览器端使用 JavaScript 操作,将数据渲染在页面上。

Django 的 URL 映射非常灵活,URLconf 中保存了 URL 模式与所调用视图函数之间的映射表,系统拥有六大 app 应用,分别对应六个功能模块,每个 app 应用都有自己的 URL 映射表,包含的 URL 几个到几十个不等,通过正则表达式进行匹配。每个 URL 都含有自己的视图函数,视图函数中包含如常见的 POST、GET 的 HTTP 请求,请求数据及请求类型在 HTTP 的报文中携带。Django 的视图负责处理用户请求并且返回响应,一个视图函数就是接受了 Web 请求并且返回 Web 响应的 Python 函数,通过使用 Python 强大的类库,用户可以在视图函数中处理各种复杂的逻辑[9]。Django 通过使用对象关系映射(ORM)将模型同关系数据库连接起来,并提供给用户易于使用的数据库 API[10]。主要的数据库 ORM 包括数据库表结构的创建和数据的增删改查,其中字段类型包括 CharField、EmailField、URLField、GenericIPAddressField 等,字段参数包括 null、default、primary_key、db_column、db_index、unique 等。提供了 Ajax 请求方式,不需要刷新整个页面,仅仅更新需要的数据,降低了服务器的处理时间,带来了更好的用户体验。

### 3.2 网页数据定位与解析的实现

以新闻出版广电总局(http://www. sapprft. gov. cn/)网站为例,用户想要获取所有新闻出版物的信息,包括音像、图书、电子期刊等。首先定义新闻出版物的 7 个信息字段(名称,前缀,地区,主办单位,主管单位,类型,网址)。Scrapy 根据定义的字段表,生成用来装载数据的 7 个 items 容器,该容器会在 spider 中用到,用来承载其抓取下来的实际数据。

Scrapy 的整个数据处理流程由 Scrapy 引擎进行控制,引擎打开第一个域名,在这里就是任务信息表中的 start_ulr,即 http://www. sapprft. gov. cn/,然后创建一个 http. Request 请求,并把请求放入任务队列中,并且构建 parse_request 方法来处理请求[11]。这里的 parse_request 是一个回调函数。

请求执行之后,http. Response 作为执行结果被返回。请求结果就是 response. body,即网页请求响应内容。然后根据任务信息表中定义的 url_xpath 来解析 response. body 获取需要的 urls,可能会获取 0 个到几百个,然后通过 allowed_url 约束条件,将不符合正则表达式的 urls 剔除,然后将合理的 urls 放入任务队列中,给引擎执行下一步抓取[12]。

同时使用 Scrapy Selectors,即字段表中的 XPath 表达式和正则表达式来解析 response. body 获取相应字段的数据转载进入 items 容器[13]。XPath 在配置时尽量使用 html 的 id、class 或者属性值去定位,这样既准确又方便,正则表达式是为了去除冗余的数据,匹配需要的数据。

使用 yield 将 items 内容传递到 ItemPipeline[14],引擎将在该处判断传递来的数据是否完整,内容是否重复,最后根据定义的字段表按设定输出,存储到相应的数据库中。

网页访问的速度直接影响了抓取的速度,有些网站本身结构清晰,访问快,自然数据抓取的速度也会非常快[15]。选取的新闻出版广电总局,网站结构统一,响应速度快,所以在系统开启任务后,用了短短 6 ~ 10 s 的时间就将几十万条出版物信息抓取下来,并且放入数据库中。当第二次启动该任务时,时间会缩短,因为系统不需要再去创建数据库和数据表,也减少了一些任务话费的时间。有些网站访问速度较慢,如土地交易网,系统花费了 28 分钟抓取了 200 万条数据,这

样的效率确实不尽如人意,这个问题将在后面的开发中进一步改善。

## 4　结束语

通过对 Scrapy 爬虫框架和 Django Web 开发框架技术的研究,设计了一种基于 Scrapy 技术的数据采集系统,并对浏览器端和服务器端的功能进行了编码实现。在系统的设计和实现过程中,对系统的整体设计框架、系统功能设计和数据库设计进行了详细说明,采用了模块化的设计方法,使系统具有良好的可扩展性。通过 Django 框架中的 MTV 开发模式实现了浏览器端和服务器端的数据交互,利用 Scrapy 技术实现了网络爬虫,jQuery 树插件 zTree 技术实现了任务管理。通过对该系统的测试,用户可以根据需求爬取数据,并将数据保存下来,进行数据处理和数据查询,达到了系统开发的要求。

**参考文献:**


[1] 于　娟,刘　强. 主题网络爬虫研究综述[J]. 计算机工程与科学,2015,37(2):231-237.

[2] 巩保胜,魏春苗. 基于网络爬虫的地理空间信息采集方法[J]. 甘肃科技,2016,32(7):17-18.

[3] 彭纪奔,吴　林,陈　贤,等. 基于爬虫技术的网络负面情绪挖掘系统设计与实现[J]. 计算机应用与软件,2016,33(10):9-13.

[4] 章博亨,刘　健,朱宇翔,等. 基于大数据和机器学习的微博用户行为分析系统[J]. 电脑知识与技术,2017,13(6):212-213.

[5] 龚　鸣,余杨志,邓宏涛. 基于 Python Django 的可扩展智能家居系统[J]. 江汉大学学报:自然科学版,2016,44(6):534-540.

[6] TANG Chong, BAGHERI H, PAISARNSRISOMSUK S, et al. Towards designing effective data persistence through tradeoff space analysis[C]//Proceedings of the 39th international conference on software engineering companion. [s. l.]:IEEE,2017:353-355.

[7] 纪培培,何顶新. 基于 Erlang/OTP 和 Django 的 WEB 实时会话系统的设计与实现[J]. 电脑知识与技术,2016,12(8):119-120.

[8] DUMOULIN J, AFFI D, MUGELLINI E, et al. eRS: a system to facilitate emotion recognition in movies[C]//Proceedings of the 23rd ACM international conference on multimedia. Brisbane, Australia: IEEE, 2015:697-700.

[9] 柴庆龙,谢　刚,陈泽华,等. 基于 Django 框架的故障诊断和安全评估平台[J]. 电子技术应用,2015,41(4):163-166.

[10] 张　台,章　杰,林培杰,等. 基于 Django 的快件揽收服务器的开发与应用[J]. 单片机与嵌入式系统应用,2016,16(2):51-54.

[11] KELLY M, NELSON M L, WEIGLE M C. A framework for aggregating private and public web archives[C]//Proceedings of the 38th international conference on software engineering. [s. l.]:[s. n.],2016:368-379.

[12] 岳雨俭. 基于 Hadoop 的分布式网络爬虫技术的设计与实现[J]. 电脑知识与技术,2015,11(8):36-38.

[13] 孙　歆,戴　桦,孔晓昀,等. 基于 Scrapy 的工业漏洞爬虫设计[J]. 网络空间安全,2017,8(1):66-71.

[14] XU Keyang, LIU Zhengzhong, CALLAN J. De-duping URLs with sequence-to-sequence neural networks[C]//Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval. Shinjuku, Tokyo, Japan: ACM, 2017:1157-1160.

[15] 赵鹏程. 分布式书籍网络爬虫系统的设计与实现[D]. 成都:西南交通大学,2014.


(上接第 176 页)


the schema lurking behind JSON documents[J]. Knowledge-Based Systems,2016,103:52-55.

[9] 林　玲,伊力亚尔. 自组织映射神经网络(SOM)在图像分类中的应用[J]. 伊犁师范学院学报:自然科学版,2010,13(1):46-48.

[10] 张建华,冀荣华,袁　雪,等. 基于径向基支持向量机的棉花虫害识别[J]. 农业机械学报,2011,42(8):178-183.

[11] 谭文学,赵春江,吴华瑞,等. 基于弹性动量深度学习神经网络的果体病理图像识别[J]. 农业机械学报,2015,46(1):20-25.

[12] ALBUKHANAJER W A, JIN Yaochu, BRIFFA J A. Classifier ensembles for image identification using multi-objective Pareto features[J]. Neurocomputing, 2017, 238(C):316-327.

[13] GUO Jingming, PRASETYO H, SU Huaisheng. Image indexing using the color and bit pattern feature fusion[J]. Journal of Visual Communication and Image Representation, 2013, 24(8):1360-1379.

[14] 徐　唐,王　锦,杨　丹. 基于 JPEG 算法的 Android 图像压缩技术研究[J]. 电脑知识与技术,2016,12(22):176-178.

[15] 陈孟原,李　峰,殷苌茗. 一种适用于有损压缩图像的重采样检测算法[J]. 计算机工程,2012,38(1):217-219.

[16] ZHOU Siwang, LIU Yonghe, ZHANG Wei. Compressed sensing of image signals with threshold processing[J]. International Journal for Light and Electron Optics, 2017, 131:671-677.

[17] DOMÍNGUEZ C, HERAS J, PASCUAL V. IJ-OpenCV: combining imageJ and OpenCV for processing images in biomedicine[J]. Computers in Biology and Medicine, 2017, 84:189-194.