**基于Python的当当网商品信息爬虫系统设计**

**1 前言**

1.1 设计背景

随着网络的迅速发展，万维网成为大量信息的载体，如何有效地提取并利用这些信息成为一个巨大的挑战。搜索引擎（例如传统的通用搜索引擎AltaVista，Yahoo!和Google等）作为一个辅助人们检索信息的工具成为用户访问万维网的入口和指南。

但是，这些通用性搜索引擎也存在着一定的局限性，如：(1)不同领域、不同背景的用户往往具有不同的检索目的和需求，通用搜索引擎所返回的结果包含大量用户不关心的网页。(2)通用搜索引擎的目标是尽可能大的网络覆盖率，有限的搜索引擎服务器资源与无限的网络数据资源之间的矛盾将进一步加深。(3)万维网数据形式的丰富和网络技术的不断发展，图片、数据库、音频/视频多媒体等不同数据大量出现，通用搜索引擎往往对这些信息含量密集且具有一定结构的数据无能为力，不能很好地发现和获取。(4)通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询。

为了解决上述问题，定向抓取相关网页资源的聚焦爬虫应运而生。聚焦爬虫是一个自动下载网页的程序，它根据既定的抓取目标，有选择的访问万维网上的网页与相关的链接，获取所需要的信息。与通用爬虫(generalpurpose web crawler)不同，聚焦爬虫并不追求大的覆盖，而将目标定为抓取与某一特定主题内容相关的网页，为面向主题的用户查询准备数据资源。

1.2 设计目的

学习了解并熟练掌握python的语法规则和基本使用，对网络爬虫的基础知识进行了一定程度的理解，提高对网页源代码的认知水平，学习用正则表达式来完成匹配查找的工作，了解数据库的用途，学习数据库的安装和使用，及配合python的工作。

1.3 本设计应达到的要求

（1）对特定的网站爬取特定的数据；

（2）实现代码和得到结果；

（3）能够和数据库进行连接，将爬下的数据存储在数据库中。

（4）将爬下的数据储存在Excel中方便编辑。

**2 系统设计方案**

2.1 爬取方案

2.1.1所需爬取的数据

以当当网（dangdang.com）为例，所需要爬取的数据为商品标题、商品出售店铺、商品链接、商品评论数、商品价格这些直观信息。

2.1.2用python获取当当网的源代码

爬虫最主要的处理对象就是URL，它根据URL地址取得所需要的文件内容，然后对它进行进一步的处理。因此，准确地理解URL对理解网络爬虫至关重要。

URL是URI的一个子集。它是Uniform Resource Locator的缩写，译为“统一资源定位符”。通俗地说，URL是Internet上描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上。采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL的格式由三部分组成： ①第一部分是协议(或称为服务方式)。②第二部分是存有该资源的主机IP地址(有时也包括端口号)。③第三部分是主机资源的具体地址，如目录和文件名等。其中，第一部分和第二部分用“://”符号隔开，第二部分和第三部分用“/”符号隔开。第一部分和第二部分是不可缺少的，第三部分有时可以省略。

Python获取网页源代码可用urllib或urllib2函数进行，极其方便快捷，代码如下：

import urllib2

response = urllib2.urlopen('http://www.baidu.com/')

html = response.read()

print html

2.1.3应用python伪装成浏览器自动登陆当当网，加入变量打开多个网页。

有了源代码就可以进行数据的爬取了，在获取数据之前，需要对爬虫进行一下伪装，使其成为一个浏览器，以实现全网页的爬取工作。

应用opener和header的基础知识即可实现伪装成浏览器这一步骤。在伪装的同时，需要加入变量来打开多个网页，是的爬取工作可以顺利进行。

class DdSpider(scrapy.Spider):

name = 'dd'

allowed\_domains = ['dangdang.com']

start\_urls = ['http://category.dangdang.com/pgl-cid4008154.html']

header = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64:x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36',}

def parse(self, response):

item = DangdangItem()

item["title"] = response.xpath('//p[@name="title"]/a/@title').extract()

item["link"] = response.xpath('//p[@name="title"]/a/@href').extract()

item["comment"] = response.xpath('//a[@dd\_name="单品评论"]/text()').extract()

item["shop"] = response.xpath('//p/a[@dd\_name="单品店铺"]/text()').extract()

item["price"] = response.xpath('//span[@class="price\_n"]/text()').extract()

print(item)

yield item # 将数据提交到Pipeline中

for i in range(2,2001):

url = 'http://category.dangdang.com/pg'+str(i)+'-cid4008154.html'

# callback为回调函数

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

2.1.4用正则表达式分析网页源代码

正则表达式是用于处理字符串的强大工具，它并不是Python的一部分。其他编程语言中也有正则表达式的概念，区别只在于不同的编程语言实现支持的语法数量不同。它拥有自己独特的语法以及一个独立的处理引擎，在提供了正则表达式的语言里，正则表达式的语法都是一样的。

正则表达式的大致匹配过程是：依次拿出表达式和文本中的字符比较，如果每一个字符都能匹配，则匹配成功；一旦有匹配不成功的字符则匹配失败。如果表达式中有量词或边界，这个过程会稍微有一些不同。

Python是通过re模块实现对正则表达式的调用的。程序代码如下：

find\_re = re.compile(r"absolute.+?span>(.\*?)</.+?&gt;&gt;</a>(.\*?)</h2>.+?</b>(

.\*?)</s.+?</b>(.\*?)</s.+?</b>(.\*?)</sp.+?</b>(.\*?)</sp.+?</b>", re.DOTALL)

2.2 存储方案

数据储存在数据库中。数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。将数据储存在数据库中也具有直观简洁的特点。

Python调用数据库是用pymysql模块，相关代码如下：

import pymysql

def condb():

conn = pymysql.connect(host='localhost',port=3306,user='root',

password='cxks20121221!',db='dangdang',charset='utf8mb4')

return conn

class DangdangPipeline(object):

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

dbObject = condb()

cursor = dbObject.cursor()

sql = 'INSERT INTO products(shop,title,link,price,comment) VALUES(%s,%s,

%s,%s,%s)'

# 将数据导入到本地数据库中

for j in range(len(item['title'])):

try:

cursor.execute(sql,(item['shop'][j],item['title'][j],item['link'][j],

item['price'][j],item['comment'][j],))

cursor.close()

dbObject.commit()

except Exception as e:

print(e)

dbObject.rollback()

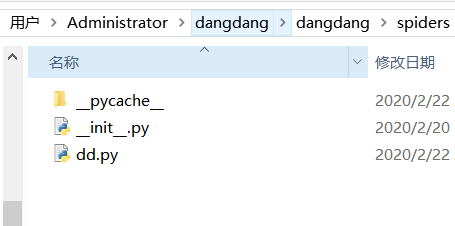
return item

**3 系统实现步骤**

第一步，打开cmd，创建爬虫项目，命令为：scrapy startproject dangdang

第二步，创建一个以基础爬虫为模板的爬虫文件，命令为：dd.py。

具体操作：在cmd界面输入：scrapy genspider -t dd dangdang.com即可。结果如下图所示，在spiders文件夹中生成了文件dd.py。



第三步，在配置文件items.py中创建要爬取的数据定义，具体代码如下：

import scrapy

class DangdangItem(scrapy.Item):

# define the fields for your item here like:

# name = scrapy.Field()

title = scrapy.Field()

link = scrapy.Field()

comment = scrapy.Field()

shop = scrapy.Field()

price = scrapy.Field()

第四步，在爬虫文件dd.py中基于Xpath爬取网站信息，并且实现多页爬取。具体代码如下：

import sys

import os

curPath = os.path.abspath(os.path.dirname(\_\_file\_\_))

rootPath = os.path.split(curPath)[0]

sys.path.append(rootPath)

import scrapy

from dangdang.items import DangdangItem

from scrapy.http import Request # 依次爬取的工具

class DdSpider(scrapy.Spider):

name = 'dd'

allowed\_domains = ['dangdang.com']

start\_urls = ['http://category.dangdang.com/pgl-cid4008154.html']

header = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64:x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36',}

def parse(self, response):

item = DangdangItem()

item["title"] = response.xpath('//p[@name="title"]/a/@title').extract()

item["link"] = response.xpath('//p[@name="title"]/a/@href').extract()

item["comment"] = response.xpath('//a[@dd\_name="单品评论"]/text()').extract()

item["shop"] = response.xpath('//p/a[@dd\_name="单品店铺"]/text()').extract()

item["price"] = response.xpath('//span[@class="price\_n"]/text()').extract()

print(item)

yield item # 将数据提交到Pipeline中

for i in range(2,2001):

url = 'http://category.dangdang.com/pg'+str(i)+'-cid4008154.html'

# callback为回调函数

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

第五步，由于可能会出现爬虫失败，以及我们需要将获取的数据进行相关处理，因此我们在settings.py文件处，要修改三处：

① BOT\_NAME = 'dangdang

SPIDER\_MODULES = ['dangdang.spiders']

NEWSPIDER\_MODULE = 'dangdang.spiders'

② ROBOTSTXT\_OBEY = False

③ ITEM\_PIPELINES = { 'dangdang.pipelines.DangdangPipeline': 300,}

第六步，在Pipelines.py文件中编写相应的数据库链接函数，以及数据库存储操作将数据存入本地数据库中，具体代码如下：

import pymysql

def condb():

conn = pymysql.connect(host='localhost',port=3306,user='root',

password='cxks20121221!',db='dangdang',charset='utf8mb4')

return conn

class DangdangPipeline(object):

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

dbObject = condb()

cursor = dbObject.cursor()

sql = 'INSERT INTO products(shop,title,link,price,comment) VALUES(%s,%s,

%s,%s,%s)'

# 将数据导入到本地数据库中

for j in range(len(item['title'])):

try:

cursor.execute(sql,(item['shop'][j],item['title'][j],item['link'][j],

item['price'][j],item['comment'][j],))

cursor.close()

dbObject.commit()

except Exception as e:

print(e)

dbObject.rollback()

return item

第七步，打开Navicat Premium，新建数据库，并且创建相应的SQL表，用于存储数据，命名为products。查询代码如下：

CREATE TABLE `products`(

`id` int(10) NOT NULL AUTO\_INCREMENT,

`shop` varchar(100) DEFAULT NULL COMMENT '出售店铺',

`title` varchar(100) DEFAULT NULL COMMENT '商品标题',

`link` varchar(100) DEFAULT NULL COMMENT '商品链接',

`price` varchar(100) DEFAULT NULL COMMENT '商品价格',

`comment` varchar(100) DEFAULT NULL COMMENT '商品评论',

PRIMARY KEY (`id`)

)ENGINE=INNODB DEFAULT CHARSET=utf8;

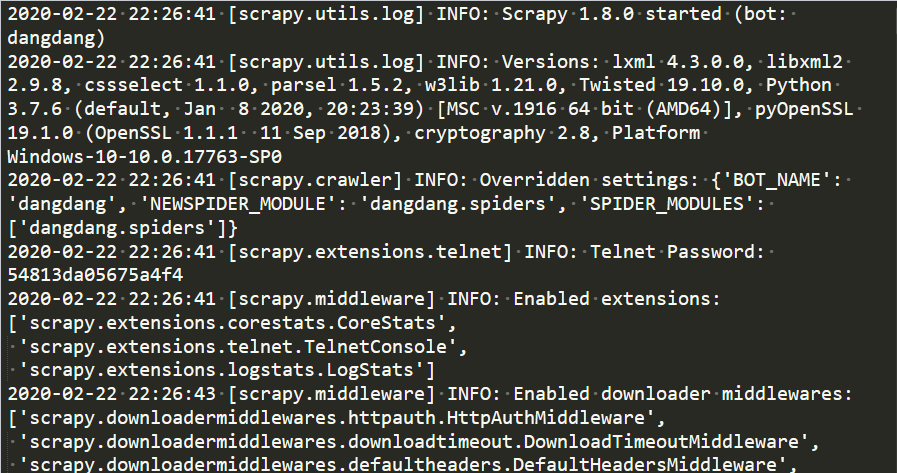
第八步，新建run.py，用于直接运行爬虫。具体代码如下：

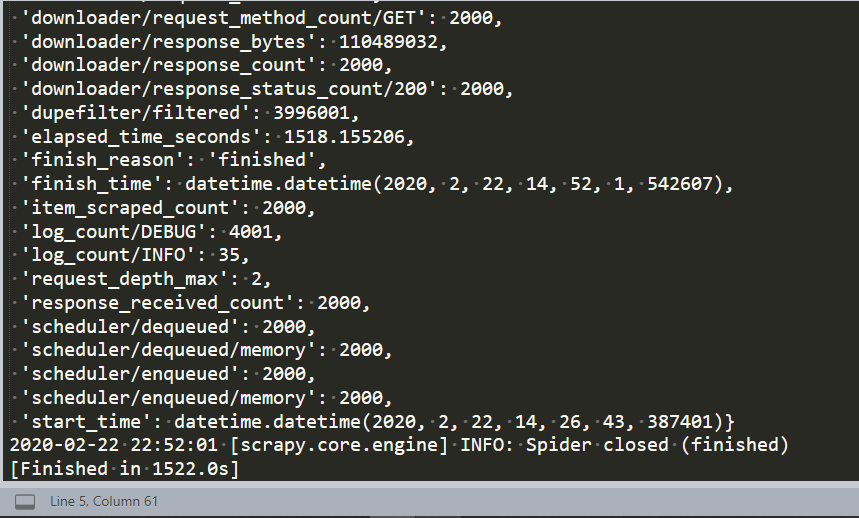
from scrapy import cmdline

cmdline.execute('scrapy crawl zhihu\_topic'.split())

**4 运行结果**

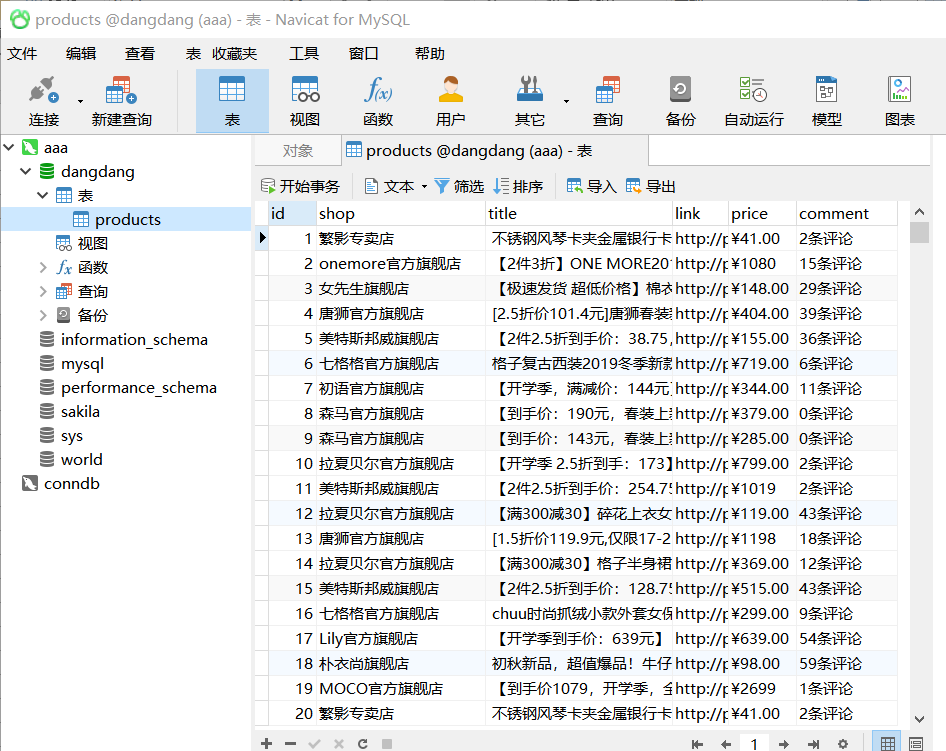
日志文件显示：

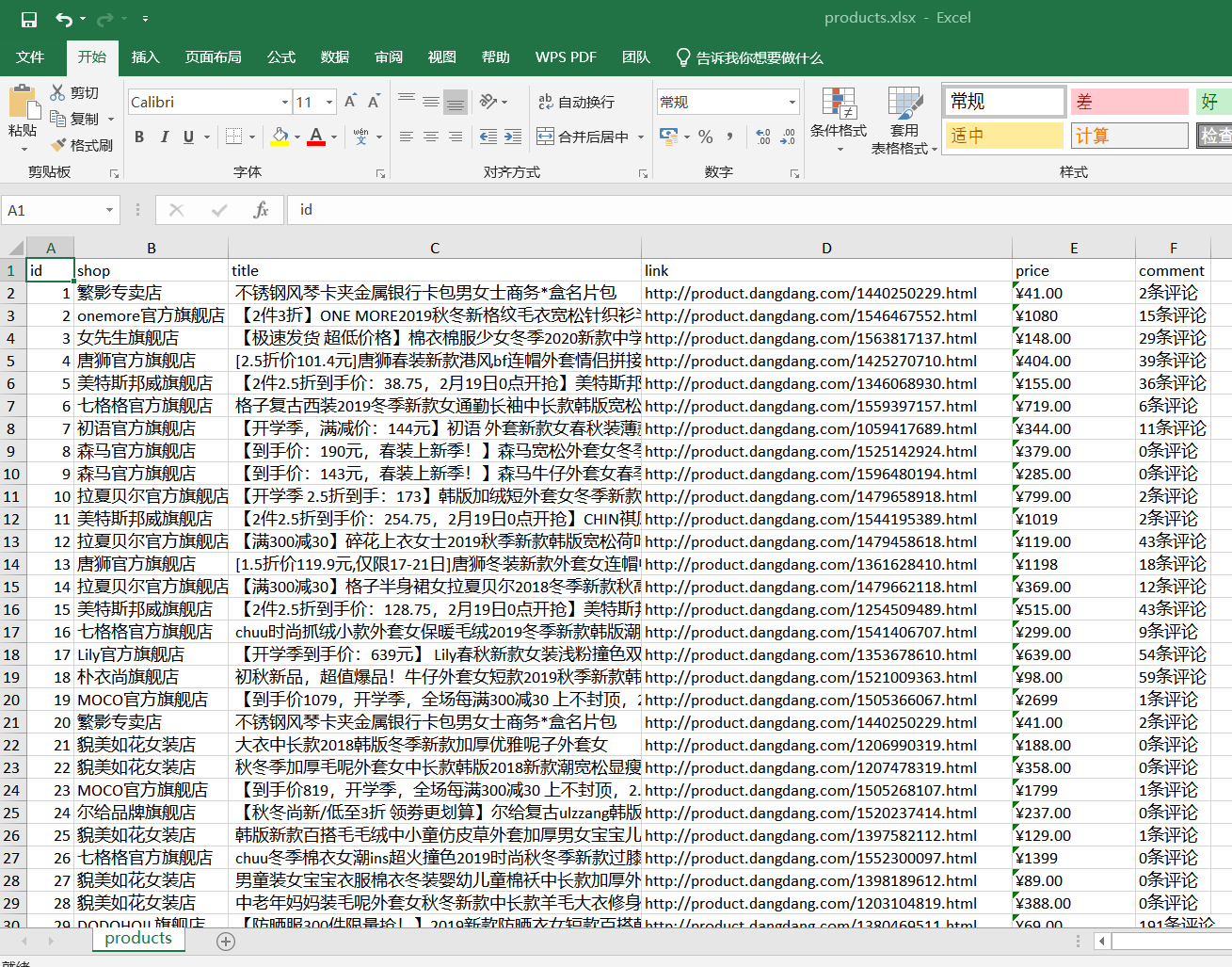


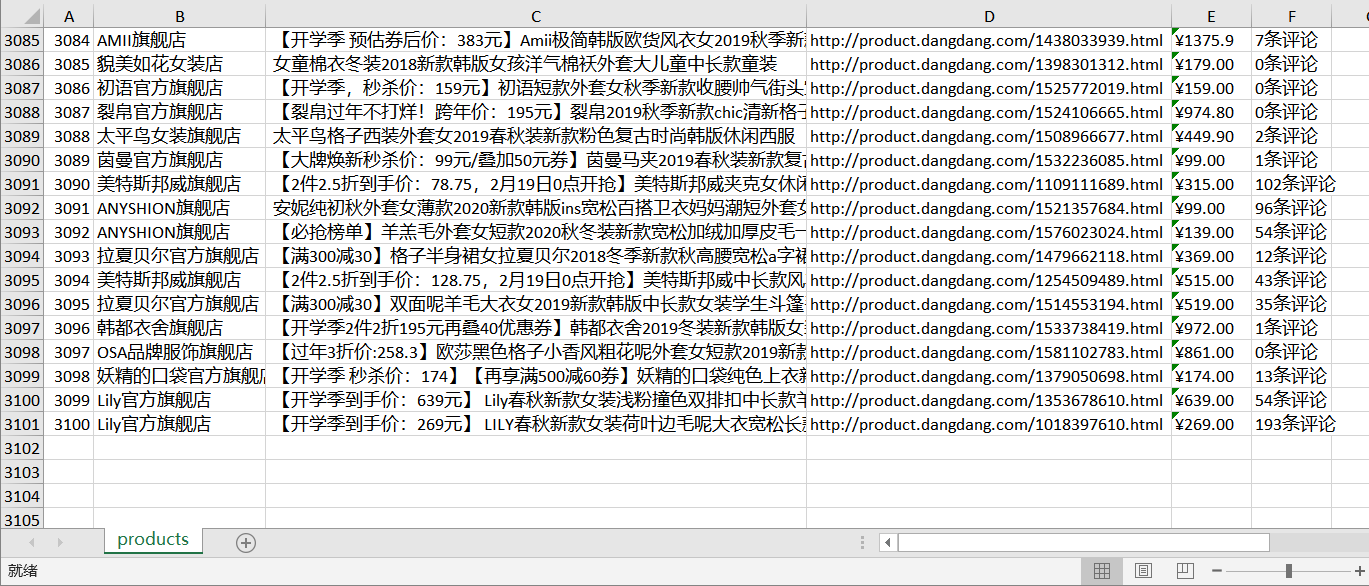


爬取到的数据用MySQL数据库保存，将数据导出为Excel文件，名为“products.xlsx”，该文件在dangdang文件夹中可以找到。

爬虫数据结果部分截图如下：







**5 总结**

本程序利用了python语言编写网络爬虫程序，实现了从当当网爬取商品数据，使用urllib函数以及re模块、pymysql模块进行源代码的获取、编辑和数据的导出、使用MySQL数据库存储数据。总的来说程序设计简便、实用性强、便于读取和再利用。