购物比价平台的设计与实现

摘要

本文主要对当前社会互联网促生的网购现象和发展做了简要的分析，对当前电商平台研究的意义做了分析，对电商平台商品信息的抓取工具做了简单介绍，并展示了使用 python 获取不同购物平台商品信息的研究，介绍了使用基于 python 的 web 抓取框架 Scrapy 来抓取并分析不同电商平台的源码技术，从源码中获取商品的信息，并借助于 python 开发的 web 框架 Flask 和 mysql 数据库将获取到的信展示给用户的方法。然后对系统设计和实现做了介绍和分析，最后对系统的测试和结果做出了展示。系统以网站的形式展示不同平台的商品信息，让用户得以更加方便的使用和操作，做出更加正确的选择。

关键词：商品信息　抓取　python　Scrapy　Flask

The design and implementation of the shopping price platform

abstract

This paper makes a brief analysis on the current social phenomenon and promote Internet online shopping development, the significance of the research on the current business platform to do the analysis, business platform for commodity information capture tools to do a simple introduction, and shows the research using Python to obtain different shopping platform product information, introduced the use of Scrapy web capture the framework based on Python to capture and analyze the different business platform source technology, access to product information from the source code, and with the help of the python development framework of web Flask and MySQL database show will get to the letter to the user method. Then the design and implementation of the system are introduced and analyzed. Finally, the test and results of the system are displayed. The system shows the commodity information of different platforms in the form of Web site, so that users can be more convenient to use and operate, and make more correct choices.

Key Words: Commodity information; Grab;Python;Scrapy;Flask

目录

1.绪论

1.1 背景和研究意义

随着互联网的不断发展，快递业不断地优胜略汰，和各大电商平台功能日益完善，网购逐渐变得越来越便捷安全，而且正在成为一种时尚。在当今社会，时间无疑是人们最重要的资源，无论是工作，休息陪伴家人，大多数人都不会愿意把这种无价的资源浪费在对着手机或者电脑屏幕选择和比较商品上面。因此，对于不同电商平台，便捷、迅速的选择比较途径被越来越多的人所青睐。

1.2 论文研究主要内容

本文主要对不同电商平台商品数据的分析和抓取，以网站的形式展示给用户，为网购用户提供清晰的商品信息比较，以便于用户在最短的时间内，准确的选择性价比更高，更加适合自己的商品。

在接下来的第二章，第一节，介绍了本论文研究的主要任务即功能需求，第二节主要介绍了开发环境和软件，第三节介绍了本论文涉及的关键技术。

在第三章，主要介绍了对论文研究内容的系统设计，主要包括功能模块设计，数据库设计，页面设计，这三个内容分我将分为三节进行介绍。

第四章是系统的实现过程，也是最主要的部分，分别包括环境的搭建，数据库的相关内容，和 scrapy 爬取数据的过程，以及数据的存储和显示，分别在五个小节详细介绍。

最后一章是对本次论文的总结和展望，以及我在学习过程中我查询过的资料和来源，最后一节是我对大学四年以及这次论文研究的总结，感谢这四年遇到的老师和同学。

2.项目准备

2.1 明确功能需求

在项目中我们首先要考虑的是我们要做什么，我们要获取不同电商平台的商品数据并且将这些数据保存在数据库中，然后以网站的形式将这些数据展示在网页上。很显然要展示给用户来自不同的电商平台的商品信息，就需要我们去获取，所以我们要学习网络爬虫的技术，而 python 已经有很多定义好的模块可以给我们直接使用，这些经过无数前辈检验过的东西让我们可以不用考虑爬虫底层的细节问题而可以更加专注于我们需要获取的数据并且有更多的时间去做后续的处理工作。在我的系统种，我选择了 python 开发的 scrapy 作为项目开发的 web 抓取和提取数据的框架。在爬取淘宝等网站的数据时，根据经验，我们爬取数据的行为常常会被服务器检测到并且他们会采取一些手段对我们的爬虫动作进行屏蔽。因此我们需要将我们的爬虫伪装成浏览器的浏览操作，这就需要我们掌握浏览器伪装的技术。在我们从网站上爬取数据时，我们是通过分析网页源码规律，获取里面的某一段数据做到的，我使用正则表达式和 XPath 表达式来对网页的源代码进行分析和获取。当我们获取到数据，我们需要将获取的数据存储起来保存在文件中，也可以存在数据库中，我在项目中选择 mysql 数据库。页面的显示由很多优秀的框架可以用，我们可以任意挑选一种，在系统中，我选择使用 flask 框架。

2.2 开发环境和软件

我在 Windows10 的 pc 上进行的项目开发，python 环境是 python3.5.4,使用的 IDE 是由 JetBrains 开发的 PyCharm 使用 PyCharm5.0.3 版本。

2.3 关键技术介绍

2.3.1 网络爬虫

要了解网络爬虫，首先要对搜索引擎的基本知识有所了解。互联网上的网

页少说也有百亿，我们想要看淡的页面藏在这么大量的页面中，之所以可以被挑选出来呈现给我们，都要依靠搜索引擎。搜索引擎将以精选的URL为基础，到网络上去下载页面，分析页面中的信息，取出URL和一些关键字，并将提取到的URL放入URL队列，（搜索引擎会判断加入到队列的URL是否被使用过，以免重复爬取），作为下一次下载的URL以此循环，将所有能够爬取的网页都下载下来。至此，搜索引擎才完成了页面收集的步骤。在收集到所有的页面以后，还要将页面的关键词放入关键词列表，并对网页进行去噪和去停用词等操作，然后将网页的链接和关键词对应起来。接下来根据相关性，链接权重，等因素对网页进行排序。最后对用户输入的词语进行分词，索引匹配等处理返回用户需要的页面。根据我上面的理解，网络爬虫所做的应该是收集页面，分析页面的工作。

网络爬虫分为两种一种是通用爬虫另一种叫做聚焦爬虫。通用爬虫通常根据初始的URL获取网页并获取网页中的链接作为下一次要爬取的URL，这个步骤会不断地重复、循环，直到满足我们设定的停止条件。

聚焦爬虫要根据一些算法去忽略掉大部分与主题无关的URL根据URL抓取网页信息的程序或者脚本，我们可以使用正则等技术对爬取的脚本进行分析以截取出我们所需要的文字内容。

2.3.2python语言

像Java一样，python也是一种面向对象语言，他有很多第三方库，也有很多内置的库，这些库是我们可以很方便的处理字符串，访问网络，处理网络信息。Python的代码要求很严格必须按照要求缩进，使代码看起来很清晰明了。所以我选择了python作为系统开发的基本语言。

2.3.3scrapy介绍

Scrapy可以抓取页面并从抓取到的页面中按照我们设定的规则提取到我们所需要的有结构的数据的应用框架。是用python编写的，在数据的挖掘，自动化测试等领域应用十分广泛。

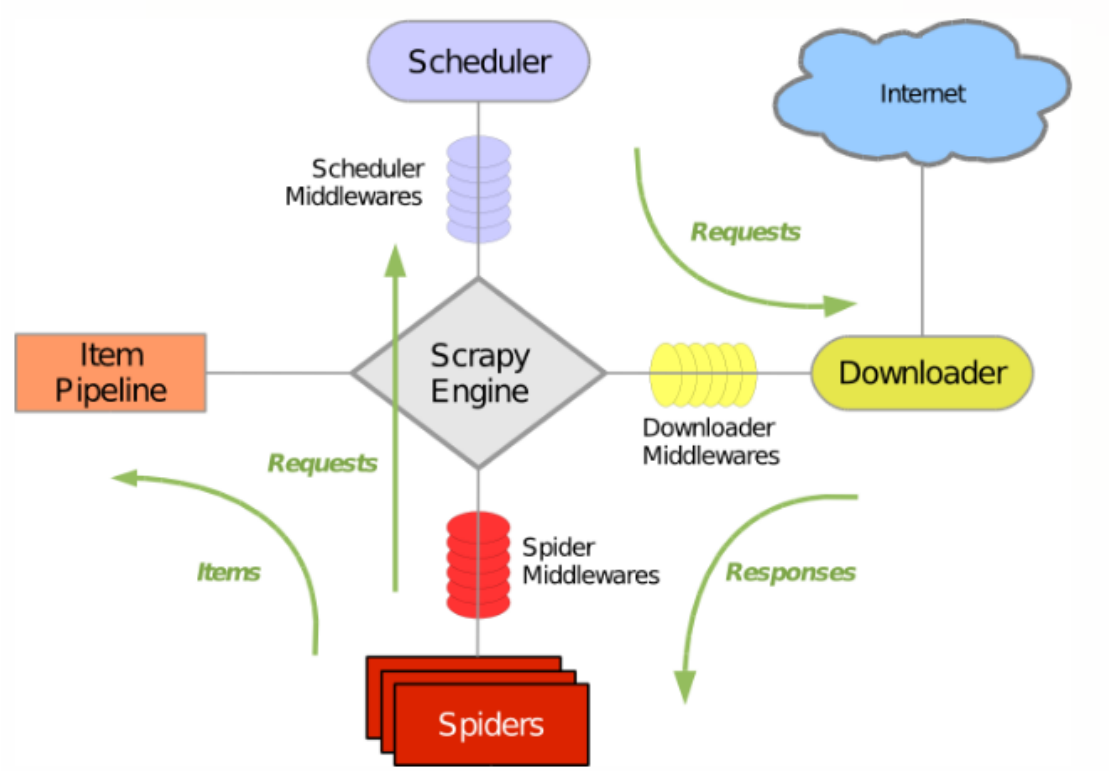

图 2-1 scrapy 原理图

Scrapy 主要由以上几部分

1、scrapy 引擎， 是整个爬虫系统的核心组成部分，主要用来控制整个爬虫系统数据的处理流程，接收和处理各个模块的响应和请求，出发事物的处理。

2、Scheduler 指的是爬虫系统的调度器，它负责管理和检测我们所爬取的 URL 的列表队列，根据优先级确定下一个要爬取的页面，而且我们的 URL 去重操作也是由调度器来完成，我们可以根据需求自己定制调度器。

3、Spider 是要我们自己定义的 python 文件是我们对网页解析的核心部分，每个 spider 都可以处理一组域名，抓取特定的网页，定义解析规则。

(1)spider 首先获取一个 URL 请求，也就是我们定义的 start_url，根据 URL 下载页面请求，系统会调用回调函数处理下载好的页面。 (2)回调函数解析网页返回的 reponse 并返回结构化的数据 Item，并获取网页中的其他 URL

(3)回调函数中我们通常用 Xpath 选择器 BeautifulSoup，正则表达式等任何可以达到目的程序。

(4)最后从蜘蛛返回的 Item 都会进入对应的 pipeline 中

4、Item Pipeline 主要的功能是对从 spider.py 文件中提取返回的结构化

数据 Item 的后续处理，对数据的验证，存储等操作。Pipeline 会接收 spider 解析页面后的结果，Item 会经过几个特定次序的处理。每个 pipeline 的组件都是由一个简单方法组成的 python 类。他们得到爬取的数据对数据进行处理筛选。

5、Downloader 是系统的下载器，主要负责抓取网页并将网页内容返回给 spider

6、Downloader middlewares 是介于 Scrapy Engine 和 Downloader 之间的框架，处理中间的请求和响应。

7、Spider middlewares 是 scrapy Engine 之间的框架，主要处理 spider 的响应输入和请求输出。我们可以使用自己定义的代码来处理发送给 spider 的请求和返回 spider 获取的响应内容和项目。

8、Scheduler middlewares 是 scrapy Engine 和 Scheduler 之间的中间件，主要处理 scrapy Engine 发到 Scheduler 的请求和响应。

2.3.4 防爬虫屏蔽之浏览器伪装技术

在我们爬取淘宝等一些大型的网站的时候，通常都会对我们的爬取动作采取一些屏蔽措施，因为它检测到了我们以非正常的手段对网站的访问，所以我们要把我们的爬虫动作伪装成浏览器的访问动作。

网站的服务器通过请求头的 UserAgent 字段来判断我们是以什么方式发出请求的，所以我们可以在爬取的时候带上浏览器的 UserAgent 让服务器觉得我们是以浏览器的方式访问的。

2.3.5XPath 表达式

XPath 是一种可以让我们在复杂的 XML 文件中迅速定义到我们需要的信息的语言。我们可以用特定的标签，符号，来定义我们需要的信息，XPath 包含一个标准库我们可以使用这些内置的函数来处理字符串值，日期，时间等。

XPath 以路径定位数据，XPath 基本语法：

表 2-1 常用路径表达式

表达式	结果

Nodenae	选取节点的所有子节点
/	从根节点选取
//	从当前结点选择文档中的节点，不考虑其位置
.	选择当前节点
..	选择父节点
@	选择属性
/html/head/title/text()	提取到标题
//li[@class='name']/a/@href	寻找所有 li 标签(class='name'的)下的 a 标签的 href 的属性

下表是一些带有谓语的表达式及结果

表 2-2 常用的谓语表达式

路径表达式	结果
/span/item[1]	选择属于 span 子元素的第一个 item 元素
/span/item[last()]	选择属于 span 子元素的最后一个 item 元素
/span/item[last(0-1)	选择属于 span 子元素的第倒数第二个 item 元素
/span/item[position()<3]	选择最前面两个属于 span 元素的 item 元素
//a[@lang]	选择所有拥有 lang 属性的 a 元素
//a[@lang=' ing']	选择所有拥有 ing 的值的属性的 a 元素
/span/item[count>66.00]	选取 span 元素的所有 item 元素而且 item 元素的 count 属性值大于 66.00
/span/item[count>66.00]/title	选取 count 属性值大于 66.00 的 item 元素的 title 元素

2.3.6scrapy-splash

我们使用 scrapy 爬虫的时候，只能爬取静态的页面，淘宝京东等网站都有反爬虫的策略，好多信息都是使用 javascrapy 在页面显示之后才加载到页面上的，所以如果我们单单使用 scrapy 的话，不能很好的爬取信息，需要采取一些办法，

Splash 是 python 支持的一个 javascript 渲染服务，使用 scrapy-splash 我们就可以很好的进行我们的数据爬取。

我们需要先安装 docker 使我们的 splash 运行在 dockers 上，安装好 dockers 后启动

安装 scrapy-splash:

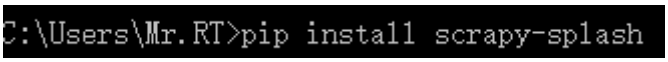

图 2-2 安装 scrapy-splash

拉取镜像：

图 2-3 拉去 splash 镜像

启动服务：

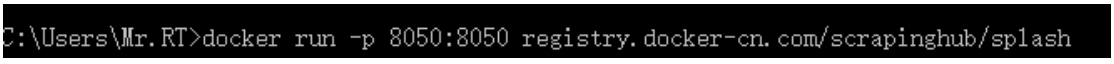

图 2-4 启动 splash

2.3.7 正则表达式

很多计算机语言都支持的用于检索、替换某些符合某个会规则的文本公式，在系统中我们使用比较多的元字符。

表 2-3 常用正则元字符

元字符	说明
$	字符串末尾。
()	开始和结束。
*	前面的子表达式 0 次或多次
+	前面的子表达式 1 次或多次。

.	除换行符之外的所有字符。
[]	中括号表达式的开始。
?	前面的子表达式 0 次或 1 次。
\	将字符标记为特殊字符、或原义字符、或向后引用。
^	字符串的开始位置，如果在方括号中使用，表示不接受该字符集合。
{}	限定符表达式的开始。
|	选择。

2.3.8flask 介绍

Flask 是一个 web 框架，它是使用 python 编写的在内部通过 python 的装饰器（类似于 Java 的注解）自动将 URL 和函数关联起来。安装 flask：

pip install flask

测试 flask:

```
1. from flask import Flask
2. app = Flask(__name__)
3. @app.route('/')
4. def index():
5.         return '<h1>Hello World</h1>'
6. if __name__ == '__main__':
7.         app.run()
```

在终端打开此文件 first.py 会自动监听 5000 端口，使用浏览器访问 http://localhost:5000/会看到 Hello World 字样

（3）Flask 路由：

```
1. @app.route('/')指明访问路径为根路径的时候调用下面修饰的函数
2. @app.route('/hello/<name>')
3. Def hello(name):
4.         Return 'Hello %s'% name
```

```
1. @app.route('/login',methon=['GET','POST'])
```

在浏览器访问 http://localhost:5000/hello/吴昊杰，将会看到“hello

吴昊杰“的字样，因为 hello 后面的名字被当作参数传到了页面。还可以在路径前添加转换器来转换参数类型比如:@app.route(‘/item/<int:item_id>’) 当在浏览器访问的时候 item/之后就必须输入 int 类型的值，否则浏览器将会报 404 错误我们可以在使用 methods 定义请求方法：

(4)使用模板 Flask 支持 jinja2 模板引擎，我们在系统中也会使用 jinja2 模板。

2.3.9 MVC 设计模式

为了方便开发和维护，在开发中，我们通常将处理请求的逻辑和包含变量显示数据的逻辑，以及模型数据分开即 model 模型层 view 视图层 control 控制器，简称 MVC

Control：介于用户，模型层，视图层之间，是应用处理和用户操作请求的入口，解析用户的输入，决定哪一个 model 和 view 去处理请求。

Model:对业务的处理，比如操作数据库，受控制器的调用提供数据给控制模块

View：主要用于显示从控制模块获取经由模型模块处理的数据，并进行一定的格式化，最后将数据在页面上显示。MVC 设计架构：

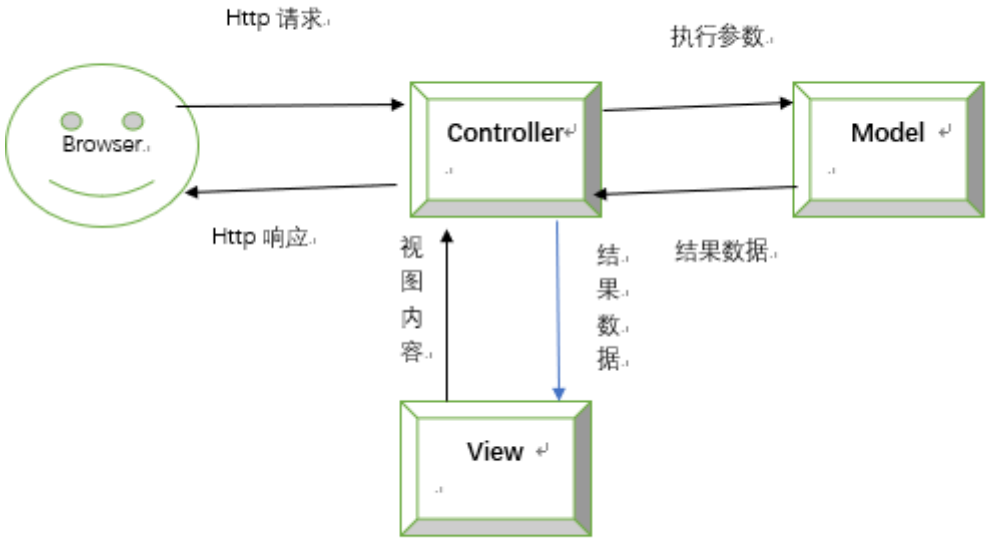

图 2-5 MVC 架构

2.3.10 基于 python 的模板引擎 jinja2

为了使我们页面和处理逻辑分开，使代码容易理解，可读性强，方便后期维护。我们必须使用模板，模板文件经过动态赋值后返回给用户。

jinja2 就是基于 python 的模板引擎，jinja2 的控制结构{% %},变量取值{{}}，注释{##}，控制结构继承等特性使得 jinja2 更加方便和高性能，我们系统中使用的 flask 也使用 jinja2 作为框架的模板系统。

jinja2 属于第三方模块需要我们安装，jinja2 安装方法十分简单，我们可以直接在命令行执行如下语句：pip install jinja2

图 2-6 安装 jinja2

使用 import 导入 jija2 模块，如果命令行中没有错误报出就说明我们已经成功的安装了 jinja2：

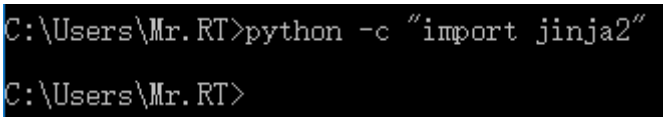

图 2-7 测试 jinja2 是否安装成功

3. 系统设计

3.1 功能模块设计

我们的目的很明确，需要从不同电商平台上爬取商品数据，将这些数据展示给用户。为了系统更加方便维护和管理，我们不得不首先将爬虫系统爬取的数据存储到数据库中，在用于展示商品的网页，再去数据库查询这些数据。所以我们主要将系统分为抓取数据，存储数据，查询并显示数据三部分，在每一部分我们都会用 python 的一些框架和模块，使我们更加专注于我们需要做的事情而不必花太多时间在底层的细节之上，当然，对于模块和框架底层的东西，还是要掌握的，因为我们可能需要根据需要，对这些框架进行设置甚至做一些简单的调整。

3.2 数据库设计

我们的设计是，将爬取的数据实时的存在数据库，在根据价格倒序显示，在展示的页面上，对于商品的选择我们需要商品的价格、名称、店铺、以及销售量等信息，在每一次的查询显示操作之后，系统会将数据清空，以保证我们查询的数据都是实时数据，不受之前数据的影响，所以我们数据库的设计必须按照这些需求进行设计：

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	不是 null
id	varchar	20	0	☑
content	varchar	100	0	☐
picurl	varchar	100	0	☐
price	float	0	0	☐
paycount	varchar	20	0	☐
addr	varchar	20	0	☐
store	varchar	20	0	☐
valuation	varchar	20	0	☐
link	varchar	100	0	☐
resource	varchar	20	0	☐
goodsid	varchar	20	0	☐

图 3-1 数据库设计

我们将每个字段都允许为空，因为每个电商平台设计都是不同的，我们设计表的字段尽可能的多，所以一定不能保证爬取的每条数据都满足存在这些字段的条件。

3.3 页面设计

我们的页面用于展示商品，就像其他电商平台一样，只不过我们展示的商品可能来自不同的电扇平台，就像下图展示的一样

图 3-2 页面

4.系统实现

4.1 搭建开发环境

4.1.1 安装 python 环境

我使用的是 3.5.4 的版本，从官网下载 python3.5.4 并进行安装，安装成功后要将安装的路径添加到环境变量中，如果在安装的时候勾选了添加安装路径到环境变量中，便不用再配置。最后可以再终端命令行中进行测试是否安装成功，输入 python 如果输出对应的版本等信息，就说明我们已经成功的安装好了 python。如下图：

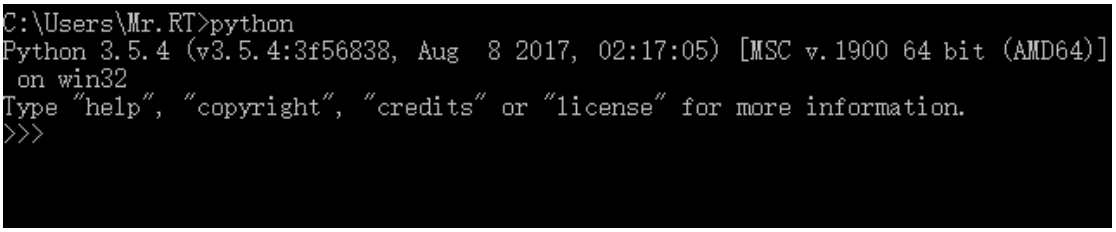

图 4-1 python 环境安装

4.1.2 安装集成开发环境 PyCharm

我使用的是 PyCharm5.0.3 版本直接下载对应的版本安装即可

4.1.3 安装 scrapy 模块

因为 scrapy 需要依赖 Windows 编译的第三方库，我们直接用 pip install scrapy 时候会报错，所以我们需要首先从下面的网址中下载 scrapy 依赖的.whl 文件：http://www.lfd.uci.edu/~gohlke/pythonlibs/

lxml 4.2.1 cp35 cp35m win_amd64.whl

Twisted 17.9.0 cp35 cp35m win_amd64.whl

Cp35 匹配 3.*的 python 版本-win_amd64 匹配 64 位，下载好文件后再终端进入下载目录执行命令：

Pip lxml 4.2.1 cp35 cp35m win_amd64.whl

Pip Twisted 17.9.0 cp35 cp35m win_amd64.whl

Pip install scrapy

4.1.4 安装 flask

直接在命令行运行下面的命令即可：

pip install flask

4.2 创建数据库

4.2.1 安装数据库链接驱动

直接在命令行输入 Pip install mysql-connector

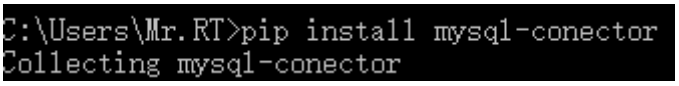

图 4-2 安装数据库驱动

4.2.2 连接数据库：

```
1. import mysql.connector
2. conn = mysql.connector.connect(user='root', password='', database='db_goodsc
   ompare')
3. cursor = conn.cursor()
```

4.2.3 创建商品信息表

```
1. cursor.execute('create table item (id VARCHAR (20) PRIMARY KEY, content VARC
   HAR (100), picurl VARCHAR (100), price FLOAT (20), paycount VARCHAR (20),' a
   ddr VARCHAR (20), store VARCHAR (20), valuation VARCHAR (20), link VARCHAR (
   100), resource VARCHAR (20), goodsid VARCHAR (20))')
```

4.2.4 测试数据库连接

```
1. cursor.execute('insert into item values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s
   , %s)',[
2. '1', '运动套装女春装', 'http://g-search1.alicdn.jpg', '218.00', '1143 人付款
   ', '广东 佛山', 'mengyuan 维诺专卖店', '10000+条评论
```

```
', 'https://detail.tmall?id=563474042059', '淘宝', '563474042059'])
```

插入结果：

id	content	picurl	price	paycount	addr	store	valuation	link	resc
1	运动套装女春装2018新款潮	http://g-search1.alicdn.co	218	1143人付款	广东 佛山	mengyuan维诺专卖店	10000+条评论	https://detail.tmall.com/it	淘宝

图 4-3 数据显示

4.3 数据获取

4.3.1 创建一个 scrapy 项目

在终端将目录切换到想要存储项目的目录下然后运行命令：scrapy startproject [‘projectname’]如下：

```
D:\>cd bishe

D:\bishe>scrapy startproject GoodsCompare
```

图 4-4 创建项目

此命令创建了一个 scrapy 项目，项目目录结构如下：

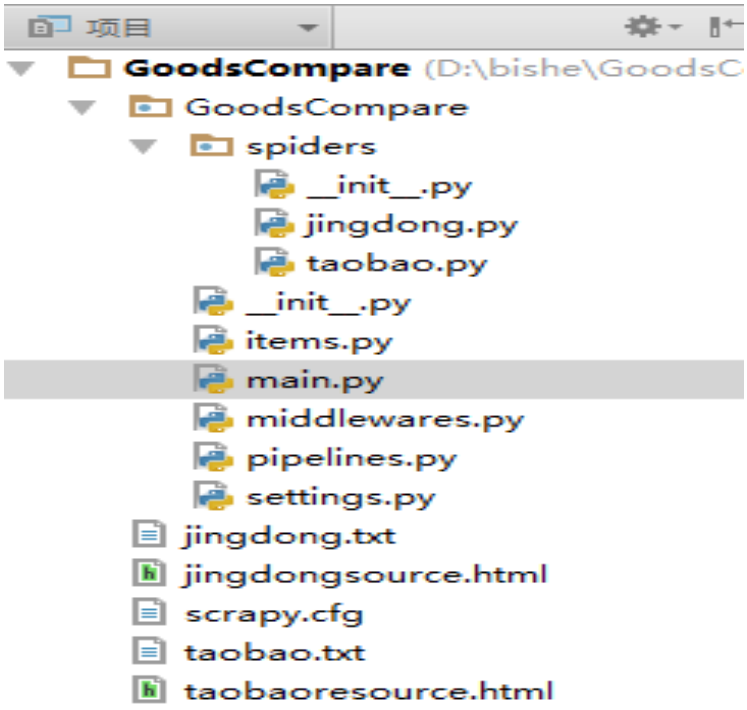

图 4-5 项目结构

其中

scrapy.cfg：项目的配置文件

GoodsCompare /__init__.py: 该项目的初始化文件。

GoodsCompare /: 该项目的 python 模块。

GoodsCompare /items.py: 项目中提取到的数据进行类型定义的文件

GoodsCompare /pipelines.py: 对获取到的数据进行处理的文件

GoodsCompare /settings.py: 这个文件是项目的配置文件

GoodsCompare /spiders/: 这个文件夹下防止爬虫的代码

GoodsCompare /main.py:从这里开始执行

4.3.2 定义 item 文件

我们需要依据需求和 spider 文件中可以提取的数据来定义 Item，使我们爬取的非结构化数据变成结构化数据，以便于做后续的处理。我们从电商平台获取相关信息，对 item 进行建模，把商品信息存储在数据库中。就像下面的这条数据一样，我们需要给 item 添加名称描述，价格，发货地等属性

我们需要用属性名=scrapy.Field()的方式给 item 添加属性，我们可以把 item 想象成一种存储数据的容器。

```
1.  import scrapy
2.  class GoodscompareItem(scrapy.Item):
3.      # define the fields for your item here like:
4.      # name = scrapy.Field()
5.      content = scrapy.Field()    #商品内容（描述）
6.      picurl = scrapy.Field()     #商品图片 url
7.      price = scrapy.Field()      #商品价格
8.      paycount = scrapy.Field()   #商品售出数量
9.      addr = scrapy.Field()       #商品发货地址
10.     store = scrapy.Field()      #店铺名字
11.     valuation = scrapy.Field()  #评论条数
12.     link = scrapy.Field()       #商品具体信息链接
13.     resource = scrapy.Field()   #平台信息
14.     goodsid = scrapy.Field()    #商品号 id
```

4.3.3 编写爬虫文件

(1)分析网页源码

我们所需要的商品数据都在网站的源码中，我们通过对网站源码规律的分

析，得出结论，提取信息，我们可以直接在浏览器中检查源码，复制出 XPath 表达式或者使用正则表达式使爬虫安照我们制定的规则提取信息数据。首先登陆京东首页，输入要搜索的商品，右键检查，我们可以看到网页的节点信息，将鼠标放到需要截取的片段上右键 Copy-CopyXPath 我们可以直接得到路径：

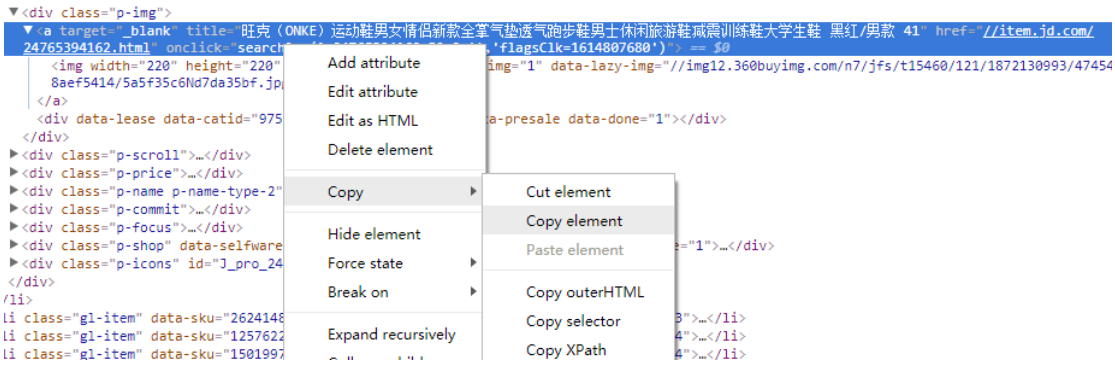

图 4-6 选择 Xpath

得到的路径如下：

//*[@id="J_goodsList"]/ul/li[51]/div/div[1]/a

分析代码可以知道，每件商品的所有信息都在 id 为 J_goodsList 的 div 下的 ul 标签下的 li 标签中，每个 li 标签包含一件商品的信息，我们获取的不仅仅是选中的那个商品的某条信息，所以我们需要定位到所有的 li 标签，将复制出来的 XPath 表达式修改为：

//*[@id="J_goodsList"]/ul//li/div/div[1]/a 我们需要获取标签的内容就要在后面加上/text()，如果需要获取某个属性值比如图片的 src 属性，就要定位到<img>标签然后使用@src，全部按照 XPath 的规则去获取。

(2)创建爬虫

首先在命令行中进入 GoodsCompare 目录并运行命令：

Scrapy genspider [spidername] [域名]

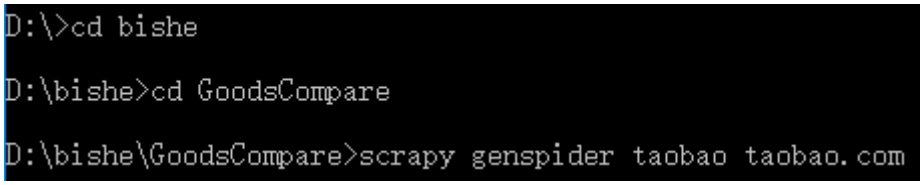

图 4-7 新建 spider

该命令会在 GoodsCompare 项目下的 spiders 文件夹中创建一个爬虫文件，taobao 是爬虫的名字，taobao.com 是域名，查看 spiders 下目录结构：

图 4-8 spiders 目录结构

Spiders 存储我们自己编写的爬虫文件，如上图中的 jingdong.py 和 taobao. Py,打开其中的一个显示如下的代码：

```
1. import re
2. class Taobao(scrapy.Spider):
3.     name = "taobao"
4.     allowed_domains = ["taobao.com"]
5.     start_urls = (
6.         '',
7.     )
8.     def parse(self, response):
9.         pass
```

我们爬虫的操作主要由我们在此处编辑的爬虫文件来做，包括我们爬取网页的动作，如何分析网页制定规则提取数据等，Spider 决定了我们要爬取哪个网页，如何分析网页，以及从下载的网页中获取哪些数据。我们创建的 spider 需要继承 scrapy.Spider 类，并且至少需要包含下面的三个属性:

name: 这个是爬虫的名字，也是一个爬虫的唯一标识，所以他必须是唯一的，我们的程序用它来区别不同的爬虫。start_urls:刚开始爬虫的时候所爬取的 url，allowed_domains:我们可以爬取网站的域名

parse:这是一个回调方法，当我们爬取我们传入的 url 的时候，爬取的返回结果会作为函数的参数传入，在这个方法中，我们编写返回数据的解析并分析返回数据结构，采取手段提取出结构化数据，如果需要的话，我们还可以从返回数据中提取需要跟进的链接，作为下一次需要爬取的 URL。

该文件主要代码如下：

首先需要导入所需要的包和定义的类，主要是 scrapy 模块和我们定义的 item 类：

```
import scrapy
import re
import urllib
from GoodsCompare.items import GoodscompareItem
```

接下来定义 name、allowed_domains、start_urls 这些属性，如果我们通过命令生成爬虫文件的话，scrapy 会自动为我们创建这些属性。

```
name = "taobao"
allowed_domains = ["taobao.com"]
start_urls = (
    '[url]',
    )
```

最后我们开始编写 spider 的核心代码即 parse 方法。

在此之前我们可以定义首次爬取的信息，使用 start_requests 方法，以便于我们将请求模拟成浏览器去获取数据：

我们可以 User-Agent 的获取我们可以打开浏览器如 Firefox 随便打开一个网页，右键查看元素调到 net 视图下，出发一个请求动作，就可以在请求头中看到 User-Agent 的信息

```
def start_requests(self):
    ua = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:57.0) Gecko/20100101 Firefox/57.0"}
    yield Request('https://s.taobao.com/search?q=%0306',headers=ua)
def parse(self, response):
    item = GoodscompareItem()
    item['content'] = re.findall(r',"raw_title":"(.*?)",',(str)(response.body))
    item['picurl'] = re.findall(r',"pic_url":"(.*?)",',(str)(response.body))

    item['price'] = re.findall(r',"view_price":"(.*?)",',(str)(response.body))

    item['paycount'] = re.findall(r',"view_sales":"(.*?)",',(str)(response.body))
    item['addr'] = re.findall(r',"item_loc":"(.*?)",',(str)(response.body))

    item['store'] = re.findall(r',"nick":"(.*?)",',(str)(response.body))
    item['goodsid']= re.findall(r',"nid":"(.*?)",',(str)(response.body))
```

```
13.     #item['valuation'] = re.findall(r',"title":"(.*?)",',(str)(response.body))

14.     item['link'] = "https://detail.tmall.com/item.htm?id="
15.     item['resource'] = "淘宝"
16.     yield item
```

在 parse 函数中，我们首先应该去定义一个 Item,然后通过 Xpath 或者正则表达式等手段解析回调参数中的 response，提取有用的信息，结构化并返回。

4.3.4 数据处理文件

我们对数据的处理都会放在 pipelines.py 文件中，打开这个文件可以看到：

```
1.  # -*- coding: utf-8 -*-
2.  # Define your item pipelines here
3.  #
4.  # Don't forget to add your pipeline to the ITEM_PIPELINES setting
5.  # See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html
6.  class GoodscomparePipeline(object):
7.      def process_item(self, item, spider):
8.          return item
```

我们在 spider 中返回的 item 会被 scrapy 传到这里，我们可以将传过来的 item 放到数据库中，以便于以后的操作。但是我们必须先修改一下 settings.py 文件才能做这些操作：

```
1.  ROBORSTXT_OBEY = True
```

Settings.py 文件中 ROBORSTXT_OBEY 默认位 True,robots.txt 是一个文件，这个文件遵循了 Robot 协议，这个文件会禁止我们爬取某些网站中的部分内容。它存在于网站的服务器中，我们当然不希望我们所要爬取的信息存在禁止爬取的可能，所以我们必须将这一项设置为 False

我们最主要的事情是要在 settings.py 文件中将 pipelines 配置进去，scrapy 才会在运行的时候起用该 pipelines GoodscomparePipeline 对应 pipelines.py 中的类名：

```
ITEM_PIPELINES = {
    'GoodsCompare.pipelines.GoodscomparePipeline': 300,
}
```

后面的 300 我们可以理解为每个类的优先级，一般 0-1000 之间的整数，item 通过 pipeline 的顺序会先通过数值更大的那个类。

4.4 存储数据

（1）我们在 spider 文件中提取到的数据转化为结构化的 item 后，会被 scrapy 传到 pipeline.py 中，在我的系统中，我使用 MySQL 数据库存储提取到的 Item 结构化数据。如果我们需要在 python 中使用 MySQL，我们必须安装 MySQL 的驱动，这样 python 才可以支持 MySQL，MySQL 驱动的安装非常简单可以直接使用 pip 命令安装。

（2）为了使我们的项目便于维护和开发我们需要先在 scrapy 的 settings.py 文件中对数据库配置信息进行设置：

```
MYSQL_HOST = 'localhost'
MYSQL_DBNAME = 'db_goodscompare'
MYSQL_USER = 'root'
MYSQL_PASSWORD = ''
```

MYSQL_HOST 配置主机名
MYSQL_DBNAME 配置数据库名
MYSQL_USER 配置数据库用户
MYSQL_PASSWORD = ''配置密码

（3）创建连接，我们首先要在 pipeline 文件中读取设置在 setting.py 中的配置，并创建一个数据库连接：

```
def __init__(self):
        # 连接数据库
        self.connect = pymysql.connect(
            host=settings.MYSQL_HOST,
            db=settings.MYSQL_DBNAME,
            user=settings.MYSQL_USER,
            passwd=settings.MYSQL_PASSWD,
            charset='utf8',
            use_unicode=True)
        # 通过 cursor 执行增删查改
```

```
11.         self.cursor = self.connect.cursor();
```

（4）在 pipeline.py 中需要实现 process_item 方法，对 spider 中爬取并结构化的数据进行处理，该方法需要返回一个 Item 对象，否则需要抛出异常。

在 process_item()函数中，参数 item 是我们定义的结构化数据 item，spider 是改 item 对应的爬虫。

```
1.  def process_item(self, item, spider):
2.          try:
3.              # 插入数据
4.              self.cursor.execute(
5.                  """insert into item()
6.                  value (%s, %s, %s, %s, %s, %s)""",
7.                  ([爬取的数据])……..
8.              # 提交 sql 语句
9.              self.connect.commit()
10.         except Exception as error:
11.             # 出现错误时打印错误日志
12.             log(error)
13.         return item
```

（5）一定要注意的是，我们每次使用一个 pipeline 组件，都要将其类名添加到 seetings.py 文件中

```
1.  ITEM_PIPELINES = {
2.      'GoodsCompare.pipelines.GoodscomparePipeline': 200,
3.  }
```

4.5 测试

我们访问写好的页面如下

图 4-9 访问页面

我们可以在输入框输入想要购买的商品系统就会自动为我们获取不同平台商品信息并保存在数据库中，然后通过查询数据库，查询出数据，根据算法以价格和评价分数不同比重排序展示在页面上，如下：

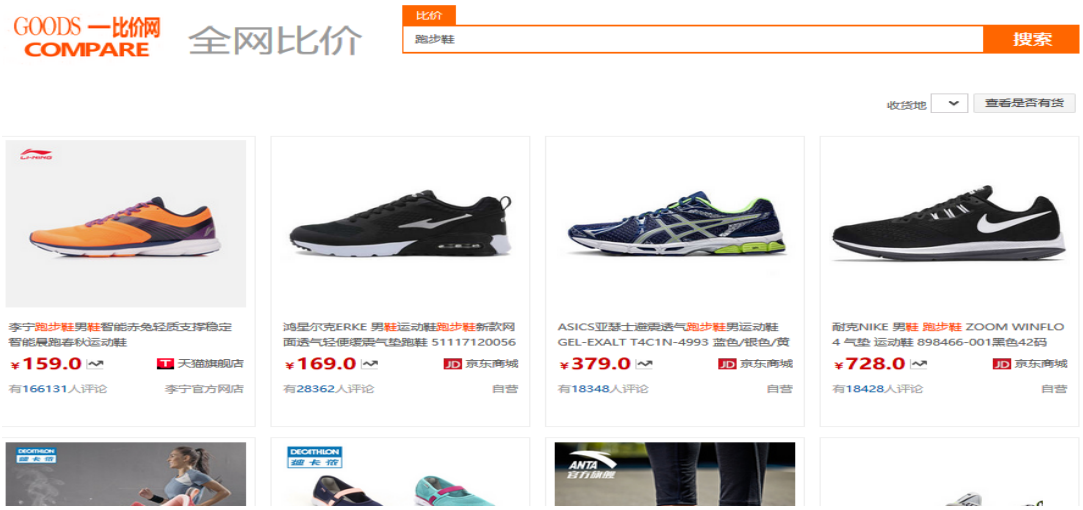
GOODS 一比价网
COMPARE
全网比价
比价
跑步鞋
搜索
收货地
查看是否有货
李宁跑步鞋男鞋智能赤兔轻质支撑稳定智能最跑春秋运动鞋
¥159.0
天猫旗舰店
有166131人评论
李宁官方网店
鸿星尔克ERKE 男鞋运动鞋跑步鞋新款网面透气轻便缓震气垫跑鞋 51117120056
¥169.0
京东商城
有28362人评论
自营
ASICS亚瑟士避震透气跑步鞋男运动鞋 GEL-EXALT T4C1N-4993 蓝色/银色/黄
¥379.0
京东商城
有18348人评论
自营
耐克NIKE 男鞋 跑步鞋 ZOOM WINFLO 4 气垫 运动鞋 898466-001黑色42码
¥728.0
京东商城
有18428人评论
自营

图 4-10 页面展示

5.结论

5.1 总结

在本课题的研究中，我们分了三个步骤，首先使用 python 的 scrapy 框架将各个电商平台的商品数据进行抓取，抓取之后，对下载下来的网页进行分析使用 XPath 表达式和正则表达式对我们所需要的商品信息获取，我们最后要展示商品名称、价格、店铺、平台以及购买链接等非结构化信息结构化为 Scrapy 内置的 item，然后在 scrapy 的 pipaline.py 中对结构化的 item 进行处理，添加到数据库中，然后使用 python 的 web 框架搭建了一个用于显示商品信息的网站，将数据库中的信息进行了展示。

5.2 展望

本课题的研究旨在为广大网购用户提供一个清晰、便捷、直接的购物比较体验，目前网络上存在众多的电商购物平台，网上购物选择花费了用户大量的时间和精力。为了解决这个问题，我们设计了这个系统。但是在开发的过程中，我也发现了系统的一些不足，目前的一些网站为了提升用户的体验质量，很多的页面都是用 javascript 生成的，但是我们使用的 scrapy 框架没有 JS engine，所以对于 javaScript 动态生成的页面无法获得，希望以后可以不断提高自身的技能，对本课题不断的关注，用自己学到的知识去完善系统，解决这些问题，提供一个更加完美的系统，可以更好的服务于更多的网购人群。

参考文献

[1]舒德华. 基于 Scrapy 爬取电商平台数据及自动问答系统的构建[D]. 华中师范大学, 2016.

[2]王静. 基于 Scrapy 的电子商务网络测量与网络特征分析[D]. 北京交通大学, 2012.

[3]李乔宇, 尚明华, 王富军, 刘淑云. 基于 Scrapy 的农业网络数据爬取[J]. 山东农业科学, 2018, 50(01): 142-147.

[4]狄博, 王晓丹. 基于 Python 语言的面向对象程序设计课程教学[J]. 计算机工程与科学, 2014, 36(S1): 122-125.

[5]朱赟. Python 语言的 Web 开发应用[J]. 电脑知识与技术, 2017, 13(32): 95-96.

[6]彭小明. 主题爬虫的设计与实现[D]. 北京邮电大学, 2013.

[7]郭一峰. 分布式在线图书爬虫系统的设计与实现[D]. 北京交通大学, 2016.

[8]房瑾堂. 基于网络爬虫的在线教育平台设计与实现[D]. 北京交通大学, 2016.

[9]王明军. 基于 Web 的空间数据爬取与度量研究[D]. 武汉大学, 2013.

[10]吴桐. 基于 Flask 框架的物资管理系统的设计与实现[D]. 南京大学, 2016.

[11]叶锋. Python 最新 Web 编程框架 Flask 研究[J]. 电脑编程技巧与维护, 2015(15): 27-28.

[12]雷亮辉, 鄂旭, 杨芳, 周津, 刘春晓. 基于开源框架 Flask 的教务系统的设计与实现[J]. 信息与电脑(理论版), 2016(20): 107-109.

[13]李文龙. 基于 Docker 集群的分布式爬虫研究与设计[D]. 浙江理工大学, 2017.

[14]赖英旭, 刘增辉, 李毛毛. MVC 模式在 B/S 系统开发中的应用研究[J]. 微计算机信息, 2006(30): 62-64+113.

[15]郭太飞, 何洁月.归纳学习 XPATH Web 信息提取规则[J]. 计算机技术与发展, 2007(03): 98-101.

[16]胡军伟, 秦奕青, 张伟. 正则表达式在 Web 信息抽取中的应用[J]. 北京信息科技大学学报(自然科学版), 2011, 26(06): 86-89.