

分 类 号: TP311

研究生学号: 201353H800

单位代码: 10183

密 级: 公 开



吉 林 大 学

硕士学位论文

(专业学位)

基于 Python 的图书信息系统的设计与实现

Design and Realization of Python-based Book Information System

作 者 姓 名: 王朝阳

类 别: 工程硕士

领域 (方向): 软件工程

指 导 教 师: 李雄飞 教授

培 养 单 位: 软件学院

2016 年 6 月

未经本论文作者的书面授权，依法收存和保管本论文书面版本、电子版本的任何单位和个人，均不得对本论文的全部或部分内容进行任何形式的复制、修改、发行、出租、改编等有碍作者著作权的商业性使用（但纯学术性使用不在此限）。否则，应承担侵权的法律责任。

吉林大学硕士学位论文原创性声明

本人郑重声明：所呈交学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：王朝晖

日期：2016年5月22日

基于 Python 的图书信息系统的设计与实现

Design and Realization of Python-based Book Information System

作者姓名：王朝阳

领域（方向）：软件工程

指导教师：李雄飞

类别：在职工程硕士

答辩日期：2016 年 5 月 22 日

摘 要

基于 Python 的网络应用框架问题研究

随着数据的不断发展，人们对数据的需求量也与日俱增，对于个人或中小型企业需要的较小规模数据可以通过网络爬虫来获取。网络爬虫程序通过 URL 地址读取网页信息，对网页文本信息加以解析和筛选，以形成目标数据。用 Python 编写的网络爬虫可以使用多线程技术提高程序运行效率，通过加锁来保证所获取数据的正确性，使用数据库存储爬取的大规模数据。爬虫程序的主要应用是实现信息的聚合，为用户提供更多可选择的信息。本设计中的爬虫程序主要为用户提供多个网站的图书信息，最终以网页的形式呈现给使用者。首先从数据库读取网页地址以及相应的规则，爬虫程序结合网页规则提取数据，把数据存入到已连接的数据库中，再把这些数据写入到 HTML 文本中，这样便完成了整个信息的聚合过程。

关键词：

网络爬虫；提取数据；信息聚合；数据库

Abstract

Python-based Web Application Framework Research

With the continuous development of the data, it is also increasing demand for data, the data for individual or smaller SMEs may need to get through the web crawler. Web crawlers to read through the web URL address information, to be screened on the web page parsing and text to form the target data. Web crawler written in Python using multi-threading technology to improve the efficiency of the program, by locking to ensure the accuracy of the data obtained, the use of large-scale data stored in the database crawling. The main application is to achieve crawlers aggregation of information, to provide users with more choice of information. The design of the crawler book information primarily to provide users with multiple sites, and ultimately in the form of Web pages presented to the user. First, read the web address from the database and the corresponding rules, combined with web crawlers rules to extract data, the stored data to a connected database, and then writes the data to HTML text, thus completing the entire aggregation of information process.

Key words:

Web crawler; Extract data; Information aggregation; Database

目 录

第 1 章 绪 论.....	1
1.1 课题背景.....	1
1.2 数据的发展.....	2
1.3 数据与垂直搜索.....	2
1.4 垂直搜索与爬虫.....	4
1.5 爬虫算法与实际应用.....	5
第 2 章 用到的模块以及 Python.....	7
2.1 Python 语言及其特点.....	7
2.1.1 Python 的优点.....	7
2.1.2 Python 中的异常处理机制.....	8
2.1.3 选择 Python 的原因	9
2.2 数据库介绍.....	10
2.3 开发工具介绍.....	10
2.4 开发使用的模块.....	12
第 3 章 程序的总体设计与详细设计.....	13
3.1 总体设计.....	13
3.1.1 概要设计.....	13
3.1.2 功能设计.....	15
3.2 数据库设计.....	17

3.2.1	数据库的 E-R 模型.....	17
3.2.2	表的具体设计.....	18
3.3	具体模块设计.....	20
3.3.1	正则表达式的设计.....	20
3.3.2	对数据库操作部分的设计.....	20
3.3.3	解析网页的设计.....	20
3.3.4	多线程的设计.....	20
3.3.5	生成网页 HTML 文本文档的设计.....	21
第 4 章	爬虫的实现与应用.....	22
4.1	编码问题.....	22
4.2	正则表达式的使用实例.....	23
4.3	读取网页信息.....	26
4.4	网页解析与多线程的实现.....	26
4.4.1	网页解析.....	26
4.4.2	多线程与锁.....	28
4.5	程序与数据库.....	30
4.5.1	数据库的具体实现.....	30
4.5.2	程序中数据库的应用.....	31
4.6	生成 HTML 显示界面.....	33
4.6.1	主界面文档的编写.....	33
4.6.2	子界面的生成与写入.....	34

4.6.3 最终界面的实现.....	35
结 论.....	38
参考文献.....	39
作者简介及在学期间所取得的科研成果.....	40

第 1 章 绪 论

1.1 课题背景

从中世纪开始一直到近代，人类一直在寻求着一种能够完成运算更加复杂、效率更高并能处理大量数据的计算机。正因为人们有这样的需求，以及人们对使用更高性能计算机的渴望，才促进了计算机科学的发展。跨越时间至现在，计算机伴随着台式机的出现已经得到了广泛的应用与普及，更成为社会正常运转过程中不可缺少的实用工具。也正因为随着使用计算机处理问题与实现需求的案例愈来愈多，亦为计算机产业带来更多领域上突破性的发展。人类与计算机变得越来越密不可分，人与人之间的交流也因为计算机作为媒介而变得更加频繁。在二十世纪后期，因为因特网的出现，人们的沟通方式大大改变了，而因特网技术是基于超文本这一概念的实现。

最初的超文本只是指向其它文档的链接文本文档，随着网络的数据类型越来越广泛，超文本早已不局限于普通的文本文档，它渐渐过渡到超媒体，其范围也已扩展到了音频，视频和图像^[1]。因特网技术让全球所有的个人计算机连成了一个密切且巨大的全球系统。在这样的历史环境下，英国的一位才华横溢计算机科学家蒂姆·伯纳斯·李通过因特网把计算机上存储的文档链接起来，形成了一个更加错综复杂的信息网，这便是万维网（World Wide Web），简称 Web。生活中，人们只要能连接上网络，便少不了通过链接来获取或交换 web 信息。

人们获取 web 信息结构图如图 1.1:

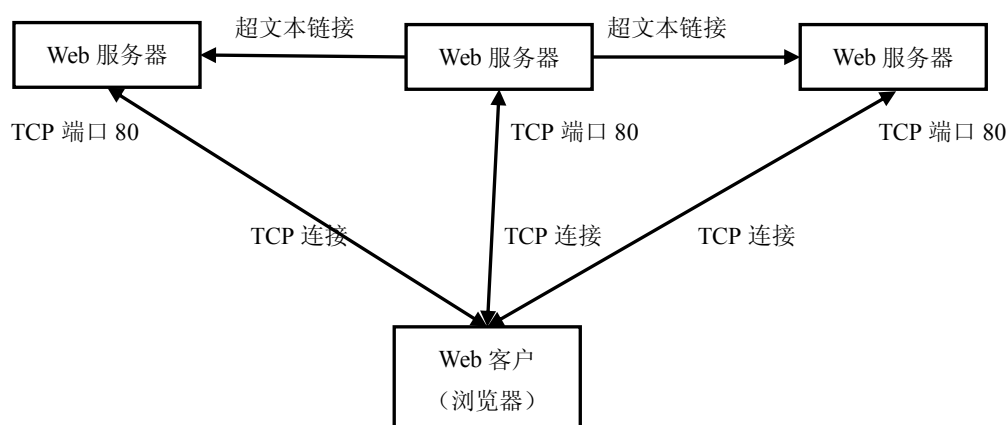


图 1.1 web 信息的获取

随着网络的深入普及，使用计算机并通过网络传递信息的人亦会迅速增加，这样一来，网络中需要存储和传递的数据也会随着使用人数的增加而增加^[2]。网络数据量的规

模和网络大小规模是成正比的，总之，人们正一步步地进入大数据时代。

1.2 数据的发展

如此多的数据存储在网络和主机中，如何有效管理如此庞大且混乱的数据，成了人们必须要面临的难题。如若从历史的角度解读这一问题，便能注意到当计算机应用在信息管理领域时，每个应用所用到或处理的数据都是独立存储和管理的，而应用本身当然也是个独立的系统^[4]。以一个公司信息为例，人事部门会有自己的人事记录数据，财务部会有员工的工资记录，客服部和销售部也都各自拥有自己的客户记录和销售记录等等，若公司各部门的信息记录都是独立分开的，会有同样或有共同部分的信息被记录多次的情况产生，该公司存储数据的方式自然而然会造成数据大量冗余和存储空间的大量浪费^[3]。

计算机领域的难题总能随着时间被一点一点的解决，使用关系型数据库便是对存储问题的有效解决方法，数据库是能够将超大规模数据集合转化为一个抽象且实用的工具的一个系统，若一个公司能够将各部门间数据关系整理清晰，通过特定的结构来存储和维护公司的数据，会让信息变得更加有价值，有效信息也对管理者作出正确决策有很大帮助。若这样处理数据的话，该公司便可建立面向数据库结构的信息系统了，而前面所提到的公司各部门独立存储数据，则是一种面向文件的信息系统。如今，一种有效且重要的信息管理工具就同时结合了数据挖掘技术和数据库技术。

数据与应用程序的关系如图 1.2:

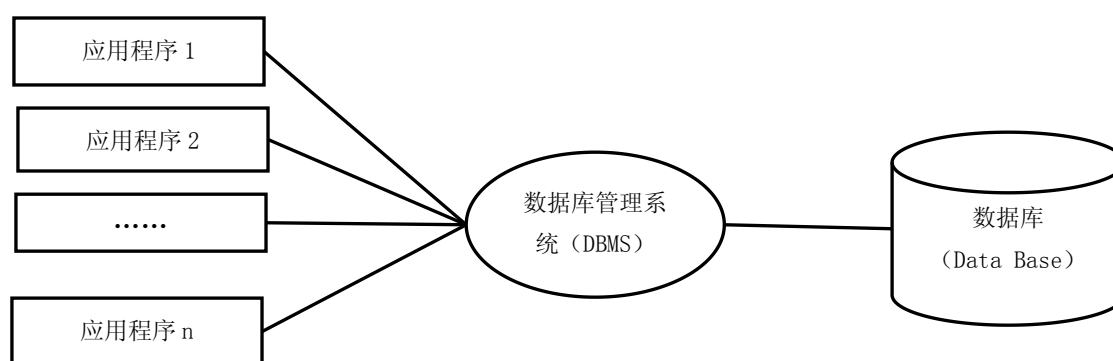


图 1.2 数据库系统数据与应用程序的关系

1.3 数据与垂直搜索

随着数据的发展与积累，应运而生了名为搜索引擎（search engine）的软件系统，而国内的人们最熟知的便是谷歌与百度这样的搜索引擎巨头了。这样的系统，通过关联并选择 web 上的文本信息，将筛选的 web 信息结果进行分类，进而为用户提供最有用的

数据信息，对数据的获取、处理与显示的难度可想而知^[5]。这样的搜索技术是不可能离开对数据信息的搜集和处理的，当然也要涉及许多非常复杂的算法才能实现这样的功能，毕竟面对着海量数据，会有一种无从下手的感觉。

搜索引擎体系结构如图 1.3 所示：

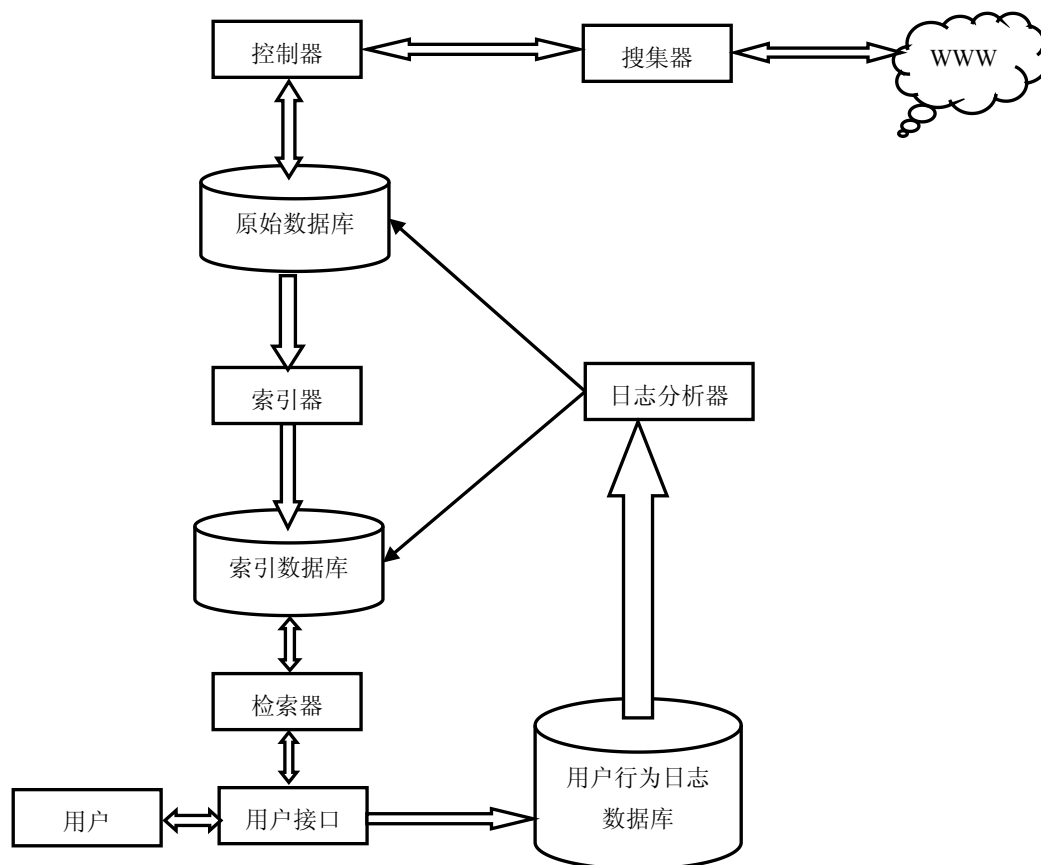


图 1.3 搜索引擎体系结构

而本篇论文讨论的主题也是对网络信息的搜集、提取和处理了，只是远没有像搜索引擎那么复杂。通过建立规则提取信息，通过数据与数据间的关系来建立数据库。但还有共同点的，那就是对信息的搜集和处理，从而为用户提供他们期望的信息结果，而达到用户的要求。由此可见，本论文讨论和研究的主题，内容上最相关的是现在大规模搜索引擎所研究的一个细小分支，人们称之为垂直搜索引擎^[6]。顾名思义，是按照一定的顺序、一定的范围、一定的规模去从 web 中获取用户想要的信息，相对而言，真正的搜索引擎网站所面对的信息量便是整个网络的 web 信息了。

垂直搜索结构图如图 1.4：

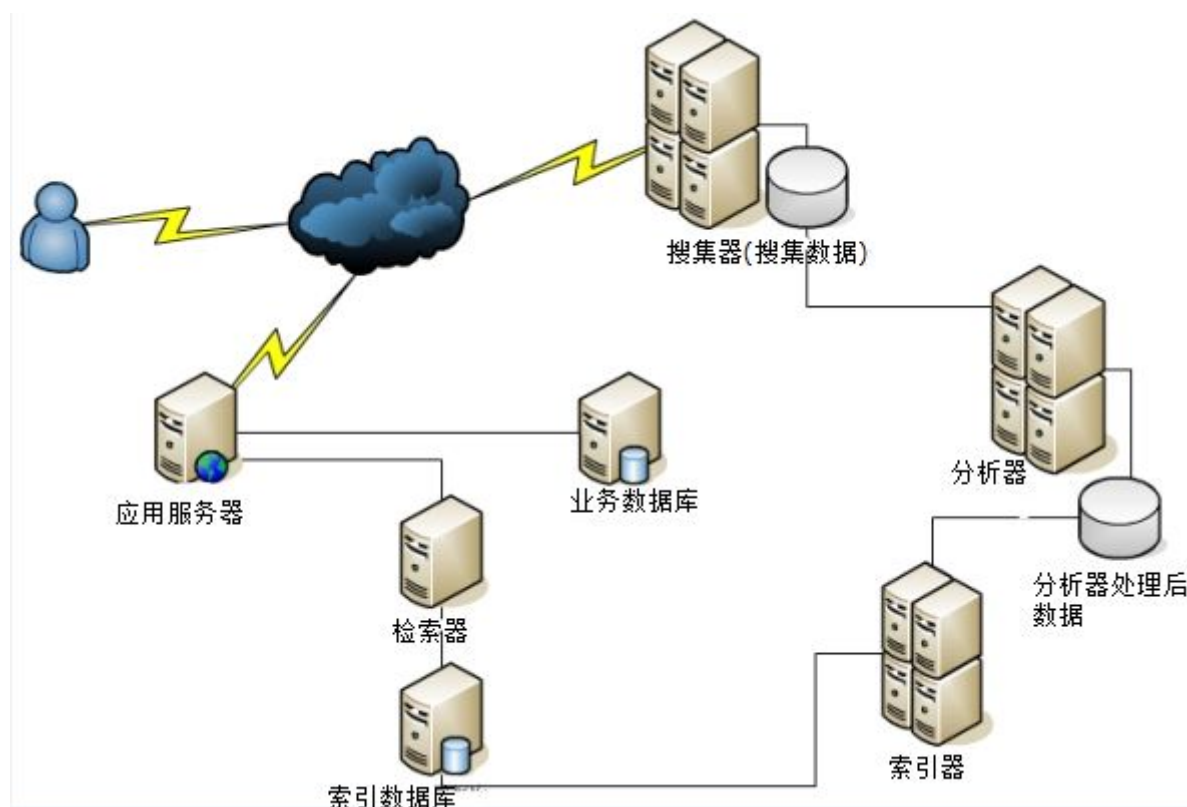


图 1.4 垂直搜索结构图

垂直搜索是对一特定行业内信息进行收集，因为它所搜集的信息之间有密切的联系，能为用户提供更广泛而有效的信息，更为人们生活带来极大的便利。这样例子在生活中总是能频繁见到的，比如人们身边充斥着新闻资讯、和生活紧密联系的天气信息、酒店和餐馆信息、列车飞机时刻信息、学术论文信息等等，会涉及到生活的方方面面^[7]。鉴于网络数据那么多，怎么才能高效率获取一个行业最新的信息，这便要通过搜集行业的垂直门户网站了，比如想为用户提供打折的商品信息，需要不断地从购物网站获取最新的打折数据，获取到这些数据后，再进行规整，能将各个购物网站打折信息集中到一个界面中显示，用户使用，能更容易对这些集中的信息进行对比而挑选出质量最好并尽可能便宜的商品。因此，垂直搜索特别受特定行业用户的喜爱，他们也喜欢使用垂直搜索用户网站。

1.4 垂直搜索与爬虫

垂直搜索致力于为人们提供最优的信息，特点体现在其实时性、高效性和正确性，倾向于实行面向结构数据和元数据的结构搜寻，在获取上千家门户网站的信息后，会自动对信息进行分类和去重，存储到自己的数据库当中，而形成自己的信息库，最后对自己的信息进行资源的整合。这种将数据存储到自己的数据库，而后显示出抓取信息的过程，叫做站内抓取，而谷歌或百度等现代意义上的搜索引擎都是站外搜索信息而整合的

^[8]。最需要提及的是，实现垂直搜索所需要的技术了，这是一种从网页爬取数据的技术，其名为网络蜘蛛（Web Spider），又名网络爬虫。

一个通用网络爬虫的框架示意图如图 1.5：

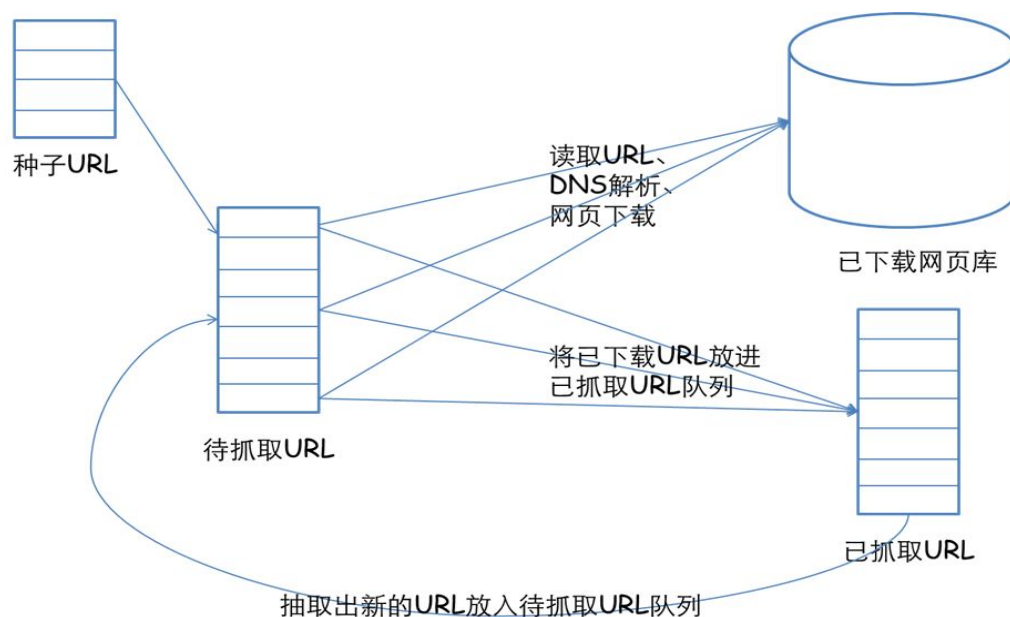


图 1.5 通用网络爬虫的框架示意图

由上图网络爬虫的框架可知，其本质就是一段抓取网页数据的程序，现如今的编程语言如此的强大，一个简单的爬虫程序是非常好实现的，爬虫程序之间的区分就在于该程序的效率是否高了，越优秀的爬虫程序能够更快速得到更多的信息，能够靠着强大的匹配规则对许多门户网站通用地抓取相关行业信息。

1.5 爬虫算法与实际应用

大多网络爬虫运行过程中使用的算法便是图论中的深度优先搜寻和广度优先搜索了，当然也有其他的算法，比如最佳优先搜索策略，它所关注的是网页或链接的相似度，前面的两种算法在实际应用中都会有各自的缺点，可以根据不用的需要而选用效率相对更高的算法^[9]。广度搜索策略是层层筛选，先选择一个链接地址以及其他许多关联性较大的地址，将无关的网页去除过滤，再对这些网页中的其他链接进行搜寻和抓取，但若随着每层要处理的网页量不断增多，那么算法的效率就会有明显的下降，深度优先搜索与之就不同了，它是先选择一个网页进行处理，再链接网页中其他的网页而搜索信息，若是它所处理的某个网页中的链接中又有其他更深层次的链接，会使程序深入停留在这个网页中不能链接到其他网页，而抓取得到的数据价值也在不断降低。

网络爬虫也可以借助大型搜索引擎返回的结果进行信息搜集，就是在爬虫程序中嵌入谷歌或百度的搜索接口，人们就可通过爬虫程序调用谷歌的搜索功能。网络爬虫的效

率要求是非常高的，而且爬虫的应用可不仅仅局限于抓取网页 web 信息，还可以对即时聊天工具中的聊天数据或者论坛中帖子的数据等进行追踪和检测，可以对敏感话题进行追踪并作出相应的判断，这样可以过滤网络中的不安全信息，爬虫也可以结合多种协议实现对信息监测。近年来关于数据的一个研究热点便是数据挖掘技术，而关于爬虫的许多研究就已应用到了 web 信息挖掘当中，当然，网络蜘蛛也可以应用于其他方方面面，总之就是应用很广泛，也有很好的发展前景。

第 2 章 用到的模块以及 Python

2.1 Python 语言及其特点

2.1.1 Python 的优点

Python 注重开发的软件质量与其一致性，使用 python 做开发的人们，总有一定的理由不愿使用别的开发语言，python 这门语言对开发者的吸引力就可见一斑了。在见到 python 的第一眼时，它良好的可读性一定会给人们留下深刻印象，许多人会把 python 语言定义为一种高效的脚本语言，这要取决于他们从什么角度说了，虽然 python 开发周期短、使用快捷，且 python 能胜任脚本语言的工作，但它可不仅是一种很好的 shell 脚本语言，也不仅仅是一种控制语言，只是 python 更致力于快速而灵活的开发模式，他所扮演的角色是非常多的^[10]。相比于传统的脚本语言，它更有优良的可维护性和可读性，这也是把 python 和其他脚本语言区别开来的凭据之一。另外，python 更有其他许多的优点，比如致力于软件质量、大量模块与标准库的支持、高开发效率、组件集成、可移植性强，这么多的优点，不仅使 python 这门语言本身很强大，更吸引许多编程者的眼球。

许多人在开始学习 Python 时，都会在官网获取资料。毕竟官网包含了 Python 的各种版本信息以及语法资料。如图 2.1 所示：

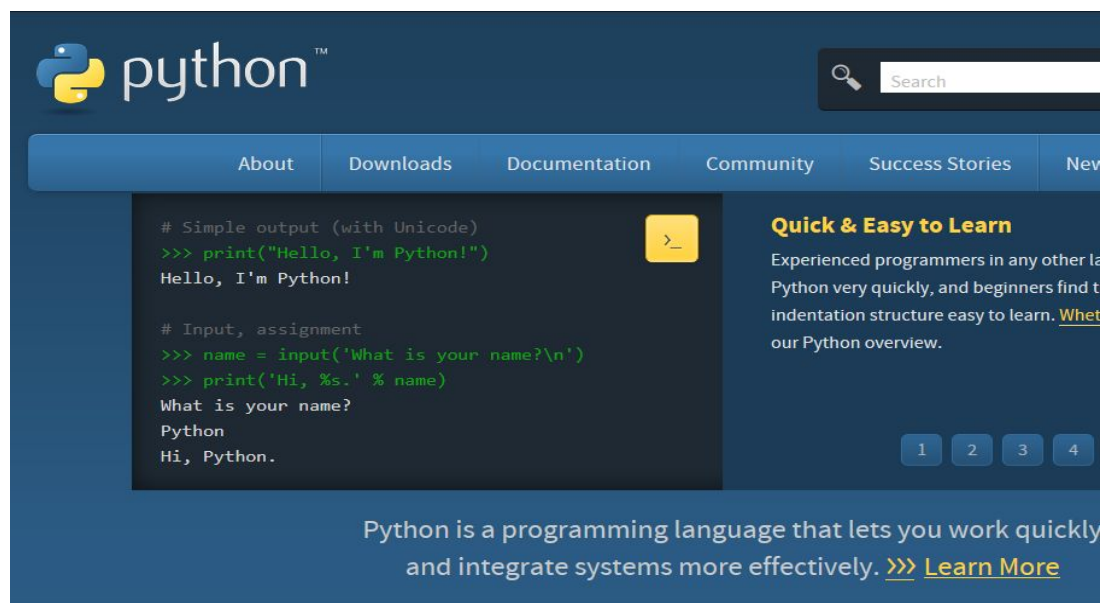


图 2.1 Python 学习的起点：Python 官网

Python 的代码需要有规整的缩进对齐，若是不符合缩进规则的，编辑器或者程序运行时就会提示错误。Python 有简洁的语法、无需编译的特点，而且开发速度是最优化的，相比其他语言，开发者使用 python 会缩短几倍的时间，开发效率和开发出的软件运行效率都不低。Python 比起 Java 和 C++ 更简单更易于使用，比 visual basic 具有更好的跨平台性能，比 Ruby 更成熟更具可读性，比 PHP 更通俗易懂用途宽广，比 Perl 设计更简单语法更简洁等等，人们会发现使用 python 比使用目前其他的编程语言或脚本语言都划算的来^[2]。说了那么多 python 的优点，不禁会使人产生疑问，难道 python 就没有缺点么，对于这一问题的答案是肯定的。那就是 python 相比于其他编译语言，它的执行速度还不算足够的快。一些程序的核心处理单元仍需要用 C 或 C++ 这样的编译语言来执行，当然也可以选择执行速度更快的语言，因为 python 并没有编译成二进制代码文件，它为了拥有优秀的可移植性由字节码的形式解释执行。

Python 源程序的执行过程如图 2.2 所示：

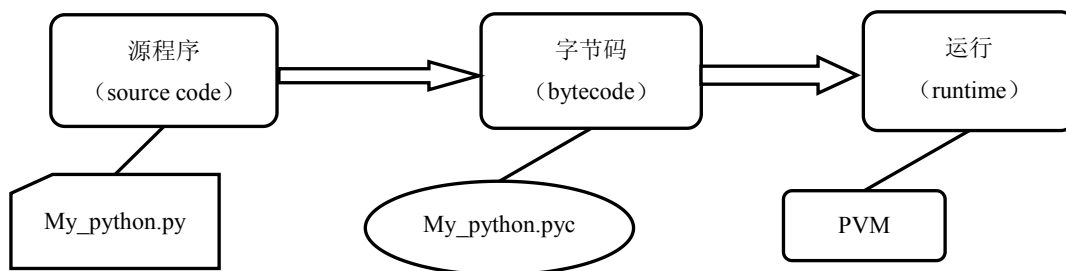


图 2.2 Python 源程序运行过程

Python 的源程序是以 .py 作为后缀结尾的，当源文件被编译成所谓字节码的形式保存时，文件会以 .pyc 为后缀保存起来，pyc 表示的意思就是 python byte code，即字节码文件，最后 python 字节码会在 PVM 即 python 虚拟机中运行^[11]。Python 解释器和大量的标准库可以以源文件或二进制文件形式在 python 网站的主要平台免费获取，并可以免费地发布 python 文件，python 官网中也包含分布了许多免费的第三方模块工具和程序，以及额外的文件。Python 解释器很容易用执行在 C 或 C++（或其他从 C 可调用的语言）的新函数和数据类型来扩展功能。Python 也适合作为用户自定义应用的一种扩展语言。Python 官网上的指导教程简单地为学习者介绍了基本的概念和 python 语言和系统的基本特性，它为编程者提供帮助，在 python 解释器这个平台上便利地增加亲自实践的经验。

2.1.2 Python 中的异常处理机制

若在 python 编程中出现错误，python 也有它处理的办法。或许可以使用 if/else 处理，也或者可使用 try/except 的异常处理机制对程序运行出现的错误异常进行特别处理。虽然以上两种方法都可以避免程序运行时出现错误，但是相比而言，还是使用

try/except 异常处理机制的优点多，若是 if/else 判断机制，不能判断错误的类型，就算避免了运行错误，但也有可能会是程序出现逻辑错误^[12]。Try/except 异常处理机制可以捕捉异常并回复错误，不管是由程序引起的还是编程者引起的异常。

Python 异常处理使用的关键字有 try、except、else、finally，程序允许的异常结构如图 2.3 所示：

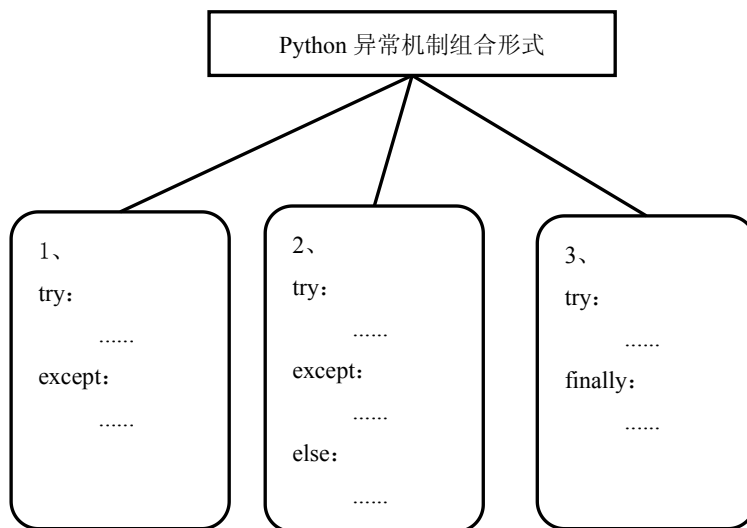


图 2.3 异常处理机制结构图

异常机制在编程中可以起到的作用有事件通知、错误处理、终止行为、特殊情况处理和非常规控制流程。简明扼要的异常处理在 python 中只不过是一种相当简单的工具而已，人们可以把任意程序片段包装在异常处理器中，假设程序一切正常工作，因为出错时程序的控制权会自动立刻跳到处理器上，只需要在处理器中人为地对异常进行处理即可，没有必要让所有代码都预防错误的发生，可以大大缩短或是避免编写处理或检查错误的程序代码。

2.1.3 选择 Python 的原因

首要的原因，还是 python 语言本身的强大功能，其次是 python 用起来很方便，而且 Python 当中有大量封装起来的模块和库，为本课题所论述的爬虫程序的实现提供了极大的便利。Python 在 Web 编程中提供了丰富的模块来支持 HTML 技术和 XML 技术；而在数据库编程中，则通过 Python DB-API（数据库应用程序编程接口）模块来连接许多数据库，比如 SQL Server、Oracle、MySQL、DB2 和 SQLite 等数据库，并与这些数据库进行通信；还有在文本编程中，python 也提供了许多模块，在整个程序中应用最多的便是关于正则表达式的模块了，这个模块对信息的匹配和提取起了很大的帮助。

2.2 数据库介绍

现今的技术，比如磁盘等硬件的发展，已经能够存储相当大的数据量，但是，如果人们不能找出所要存储信息的有用信息项，那么这样存储的数据集合就是无用的。数据库系统就是用来研究这些数据之间的关系，并从非常庞大的数据集合中整合出有用的信息。通常情况下，数据库会涉及多个软件层，主要有两个，即数据库管理层和应用层。应用软件处理的是用户与数据库之间的通信，这个过程可能会很复杂。

用户从应用的角度看到的数据库是应用软件，应用软件从数据库模型的角度看到的数据库是 DBMS（数据库管理系统），而数据库管理系统从实际组织的角度看到的数据库就是实际的数据库了。事实上也就是，用户若想操作实际的数据库，需要通过应用软件和数据库管理系统两层的跨度，用户只有通过数据库管理系统实际地在数据库中增加或删除数据，若用户想请求检索信息，那实际上也是通过数据库管理系统完成所要求的信息检索。数据库管理系统和应用软件分离开来有许多好处，这样的分层结构提供了对数据库进行访问控制的一种手段，这样做也能获得数据的独立性，当需要数据的修改或改变时，只需要修改数据库，而不用改变应用软件，另一个好处是允许使用和构建抽象工具，数据库管理系统屏蔽了数据库实际如何存放数据的细节，这使得设计应用软件的工作得以大大简化了。

2.3 开发工具介绍

用到的开发工具有：Python3.3, Eclipse, PyDev, MySQL, MySQL connector Python 以及 MySQL Workbench。安装 Python 时，会有一个开发 Python 程序的界面，可以一边编写语句，一边运行所编写的程序。

Python 自带的开发界面如图 2.4 所示：

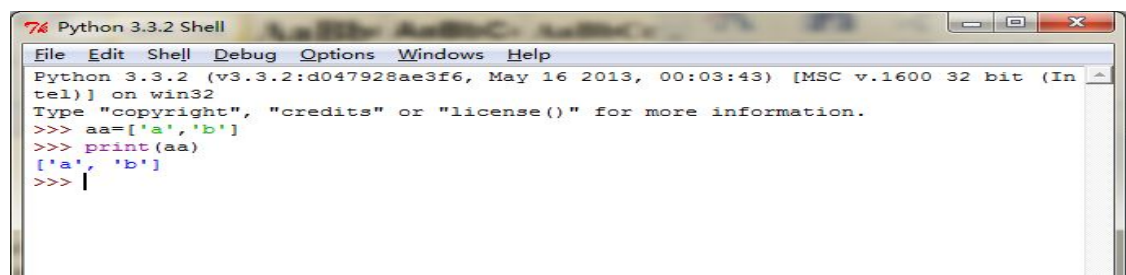


图 2.4 Python 的 IDLE

Eclipse 是 Java 开发的集成开发环境（IDE），它本身也是一个开源项目，它能够通过其他大量的插件去支持其他编程语言的开发，例如 C 和 C++、PHP，当然还支持本设计所用的 python 了，这一切要归功于 Eclipse 的扩展性很强了。Eclipse 不仅功能强大，还提供了比 PythonWin 更强大的调试能力，开发工具 Eclipse 的主界面主要分为编辑器

和视图两个部分，编辑器主要包括 Java 源代码和 python 源代码编辑器，视图部分包括源代码视图和文件系统视图^[13]。Eclipse 的设计思想很有意思，它一切以插件为中心，软件本身的内核非常小，虽然主要是用来开发 Java 项目，但也要有 Java 开发环境插件，若想开发 python 程序，自然就要安装其相应的插件了，所有其它的功能也均是在其核心上安装插件而实现的。

PyDev 是用来作为 python 集成开发环境而嵌入在 Eclipse 中的插件（它也支持 Jython 和 IronPython）。它使用高级的类型推断技术提供像代码不全提示和代码分析检测的功能，当也能提供许多其他的性能，例如调试器、交互式控制台、代码重构、代码提示错误、代码编辑提示等等。

在 Eclipse 中运行 Python 程序界面视图如图 2.5 所示：

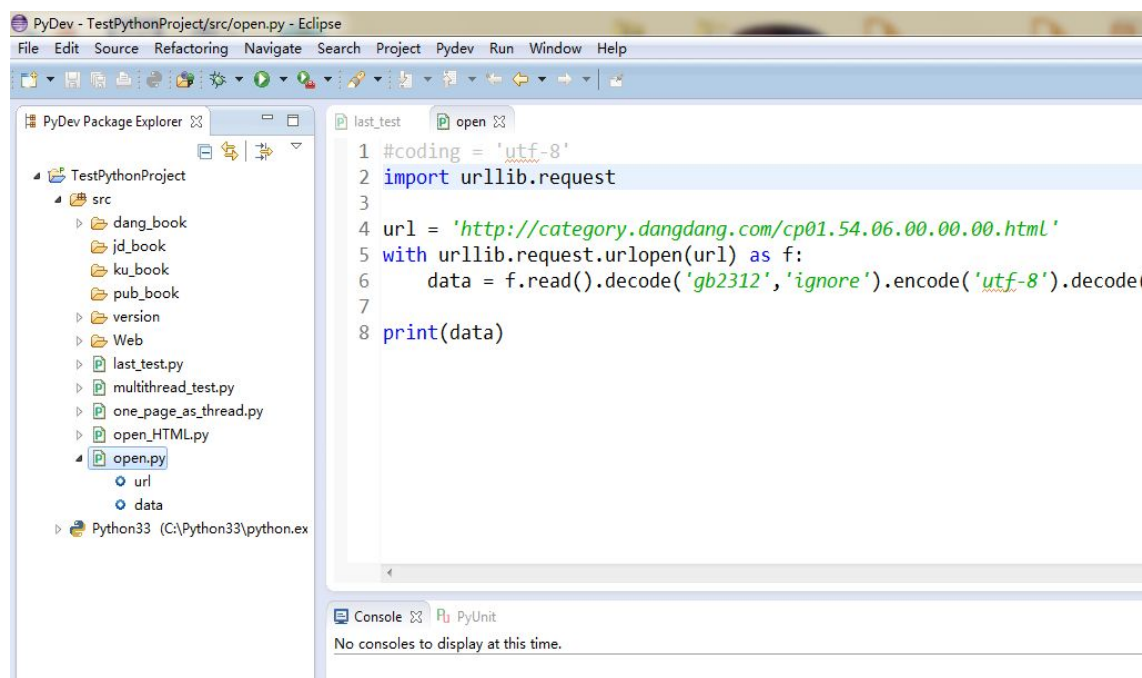


图 2.5 Eclipse+PyDev 开发界面

MySQL connector Python 是 python 连接 MySQL 而用到的第三方模块的安装软件，通过该模块还可以操作数据库。MySQL Workbench 是 MySQL 的图形界面软件，界面很漂亮。

MySQL Workbench 的开发视图界面如图 2.6 所示：

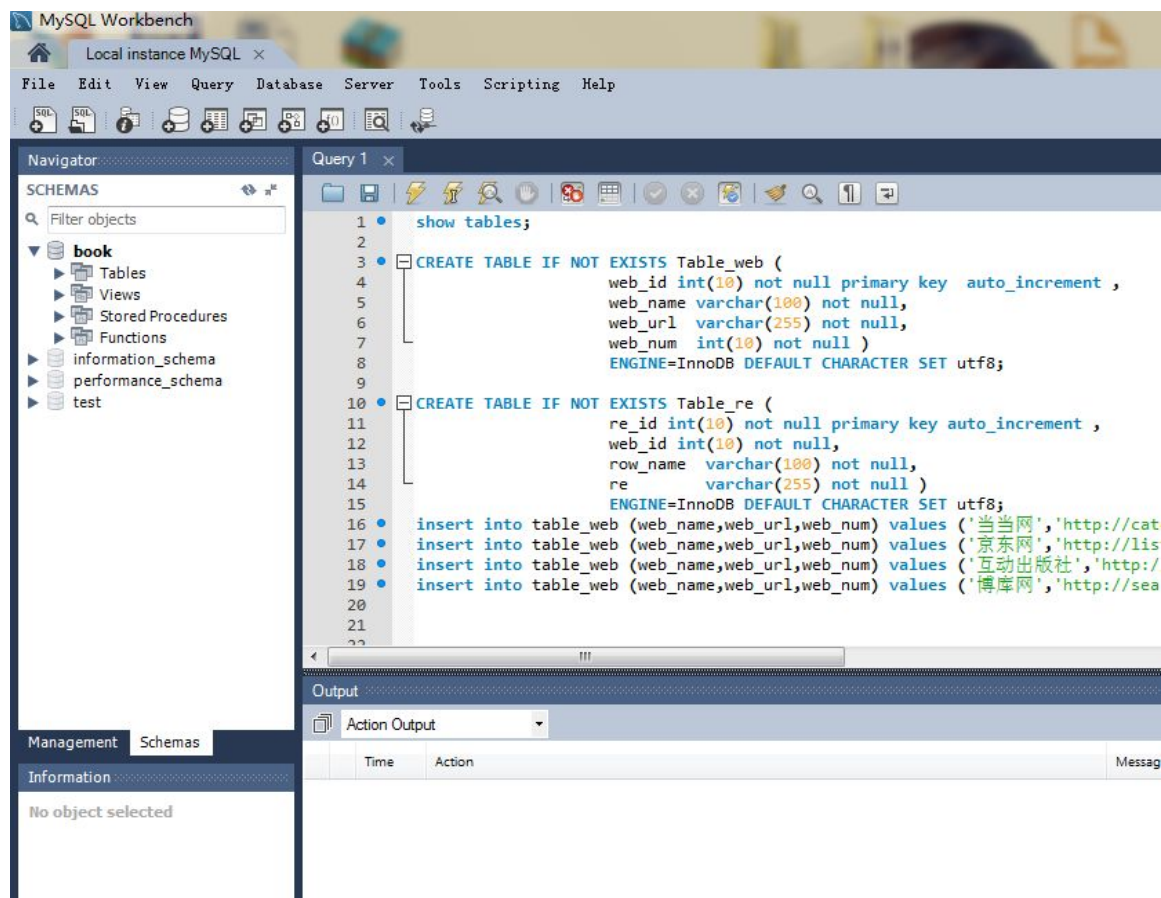


图 2-6 Workbench 的开发视图界面

MySQL 数据库是基于关系型的数据库管理系统，虽然跟其他的数据库相比它还是有缺点的，但是它的缺点丝毫不影响它受欢迎的程度，因为它所提供的功能，能够远远满足中小型企业和个人用户的需求，MySQL 开源、免费、体积小，为用户提供极大便利，这也是选择 MySQL 作为数据库的原因，而且在做本设计之前，个人也一直没有接触过这个数据库，但是一旦接触后，真的会喜欢上 MySQL，它的简单实用真的是深得人心。

2.4 开发使用的模块

本课题中使用到的 python 模块主要有 re 模块、urllib 模块、HTML 模块、os 模块、sys 模块、threading 模块，还有第三方下载的 mysql.connector 模块。

第3章 程序的总体设计与详细设计

3.1 总体设计

3.1.1 概要设计

为了实现图书信息的聚合，本设计致力于设计一个爬取多个网站的图书信息的网络爬虫程序。总的来说，先是程序通过数据库读取各个 URL 的地址和相应网站需要抓取的页数，获取程序的数据来源，再经过 python 程序的处理爬取到想要的图书目的信息，并将爬取的数据存入到自己的数据库中，再取出并写入 HTML 文档中，以网页的形式显示给用户。

对于系统维护人员来说，要将用户需要的网页地址和抓取页数写入数据库 book 中的 Table_web 表中，还需将用于匹配该网页图书各种信息的正则表达式写入数据库 book 中的 Table_re 表中。程序运行时先是需要连接数据库 book，再从该数据中读取网站 URL 地址和页数，以及该网页的各种规则，然后循环抓取各个网站图书信息。循环队列不为空时，获取该网页的具体正则表达式，再通过函数获取其他页面的 URL 地址，该地址和正则表达式作为参数传入到提取该页面图书信息函数，提取一个页面图书的函数作为一个线程来运行。那么该网站需要提取几个页面，该网站对应的线程数目就有多少。若该网站的所有线程运行完毕，那么该网站的信息也就提取完毕，可以进行下一个图书网站的信息提取的过程了，而且该网站需要抓取的书的信息也全部写入到数据中了，每运行完一个网站，循环列表的长度就减一，这样运行直到循环列表长度为零为止。

既然能够完成抓取信息的任务，下面的任务就是需要把存入数据库中的图书信息分页面显示出来，本设计中是把一个网站的信息通过一个 HTML 页面为用户显示出来，先写一个主界面的 HTML 文本，再通过程序生成各个网页的 HTML 文本文档。网页的 HTML 程序比较繁琐，设计的过程中要对各个标签结构进行仔细的考虑，还要考虑到页面的整体设计与规划，在尽量为用户展现出美观易懂的网页界面的基础上，实现信息的最大聚合。

图 3.1 是本设计程序的概要流程图：

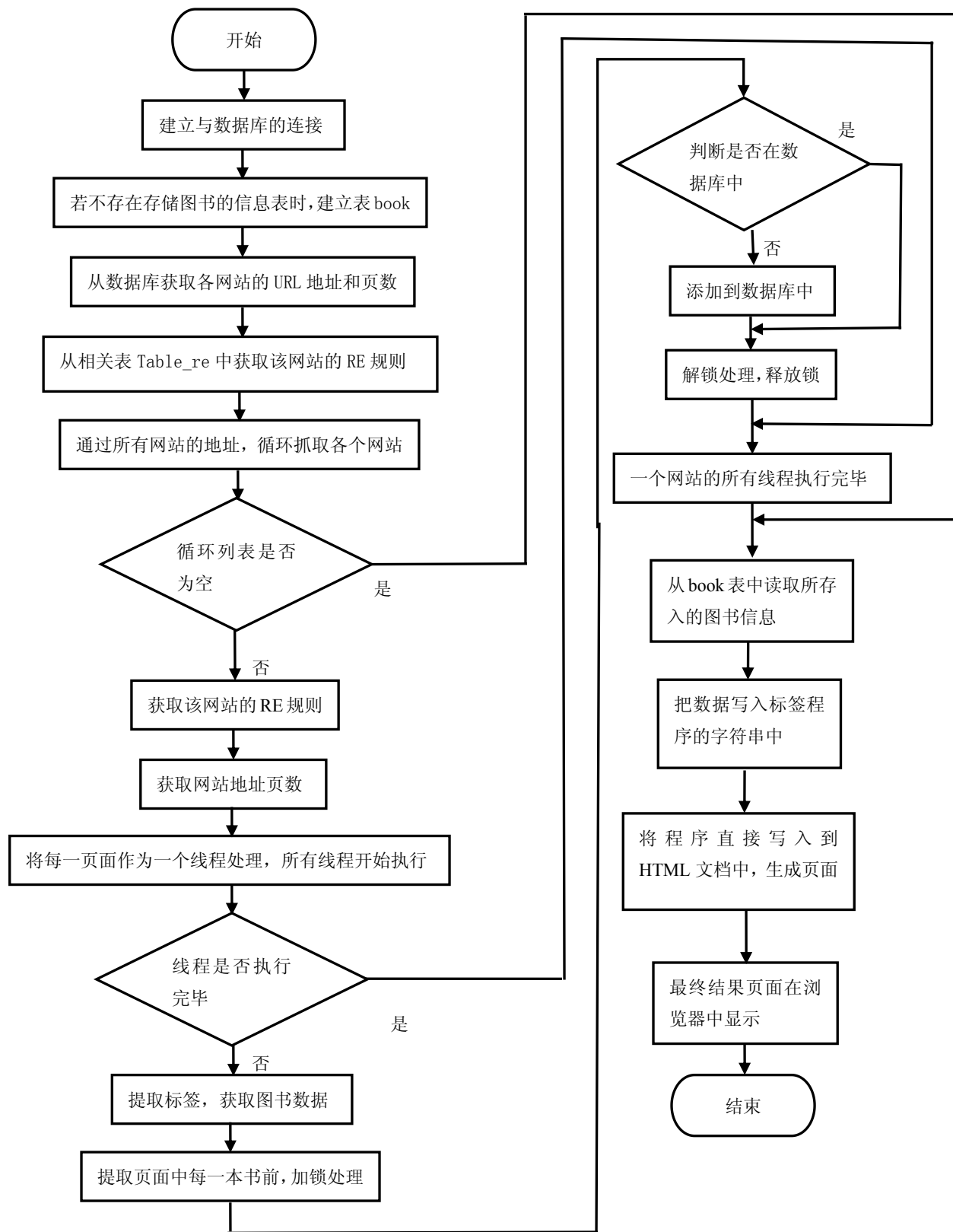


图 3.1 程序流程图

3.1.2 功能设计

本设计中的网络爬虫程序中需要实现的最基本的功能有：连接数据库，读取数据库，写入数据库，读取网页信息，正确分析网页信息，正确生成 HTML 程序的文档等等。

这些功能是对本设计中爬虫程序的最基本要求，若是其中一个功能不能实现，那么整体上来看也就是失败的设计。对于爬虫程序的功能设计是基于为用户考虑的角度，先来设计实现的目标，之后再对每一要求分成多个模块具体地去实现其功能。在实现每一小功能之后，在此基础上要把握好程序的整体结构，把实现的每一小功能程序，放到它应该放置的位置，使得程序在结构上达到最优化，效率上达到最高效。具体功能设计有以下六点：

(1) 连接数据库，能够使得在 python 程序和 MySQL 数据库之间建立正确而稳定的桥梁，这一功能是实现其他对数据库操作功能的基础。

(2) 读取数据库，这一功能可以读取所有网站的 URL 地址和抓取页数以及对应网站的正则表达式。若是不能实现这一功能，那么就需要将各个正则表达式手动地写入程序当中，这样的话，最多就只能提取一个网站的数据，而且需要对程序进行大量的修改，为抓取数据带来了极大的不便。

(3) 写入数据库，用来实现将提取的图书各方面信息写入到数据库的图书表中，该表有多个列，每一列代表了每本图书的一个属性，写数据库这一功能的实现，是整个获取数据过程的最后一个步骤。

(4) 读取网页信息，该功能是抓取数据的第一步，当网页地址作为一个参数传入抓取网页的函数时，该函数的第一步就是使用这一功能打开一个网页的 HTML 文档信息，之后在进行其他的处理。若是打开一个网页，后面就要有关闭打开网页的处理，如何有效地关闭网站，也是在功能设计考虑的范围之内。

(5) 正确分析网页信息，在获取网页的 HTML 文档后，这一功能就发挥了作用。在一个图书销售页面中会有多本书，首先就是要获取每整本书的标签部分，再对该部分提取书的其他属性信息。该函数中会包含许多函数或者类的对象，书的信息会包括价格，书名，作者等等各方面，若是用一个函数来实现，难度就可想而知了，而且这样写出的函数会很复杂，不利于阅读及维护。用多个函数或对象来提取多种属性，而每个页面作为一个线程运行时，或许会出现信息的交叉提取，需要考虑如何避免这一情况。

(6) 正确生成 HTML 程序的文档，这一功能是为用户提供显示界面而设置的。该功能部分可以设计多个 HTML 页面，或者是通过 python 程序生成 HTML 代码并写入该界面的文档，最后会有多个 HTML 文档。需要自己去编写显示的主界面，而通过程序生成写入的文件会与主界面结合在一起在浏览器中显示。提取了几个网站的信息，就会自动生成相应 HTML 文档来显示该网站的图书数据。

在功能设计过程中还需要考虑的几点问题，首先是各个界面的编码问题，各个界面的编码需要统一，不然会出现乱码问题。其次是在写入数据库时，不能重复地对同一图书进行数据库的插入，在此之前，需要对该图书是否已经插入数据库进行一次判断，若是已插入数据库，则输出已插入数据库的提示信息，若该图书没有插入数据库，则尝试性的插入数据库。还需要考虑对程序出现的异常进行相应的处理，更需考虑到程序的结构不至于混乱，结构上需要尽量保持清晰，易于阅读，不然混乱的代码不仅不利于修改，更不利于维护。还有值得考虑的一点，就是在以多线程提取网站各个页面信息时，是否会出现图书的属性混乱的情况，具体来说就是这本书的书名或其他信息不能写入到其他书的信息当中，必须保证每本图书信息的正确性和一致性，做不到这一点的话，提取的信息也便失去了它的价值和意义。

3.2 数据库设计

3.2.1 数据库的 E-R 模型

该数据库的 E-R 模型如图 3.2 所示：

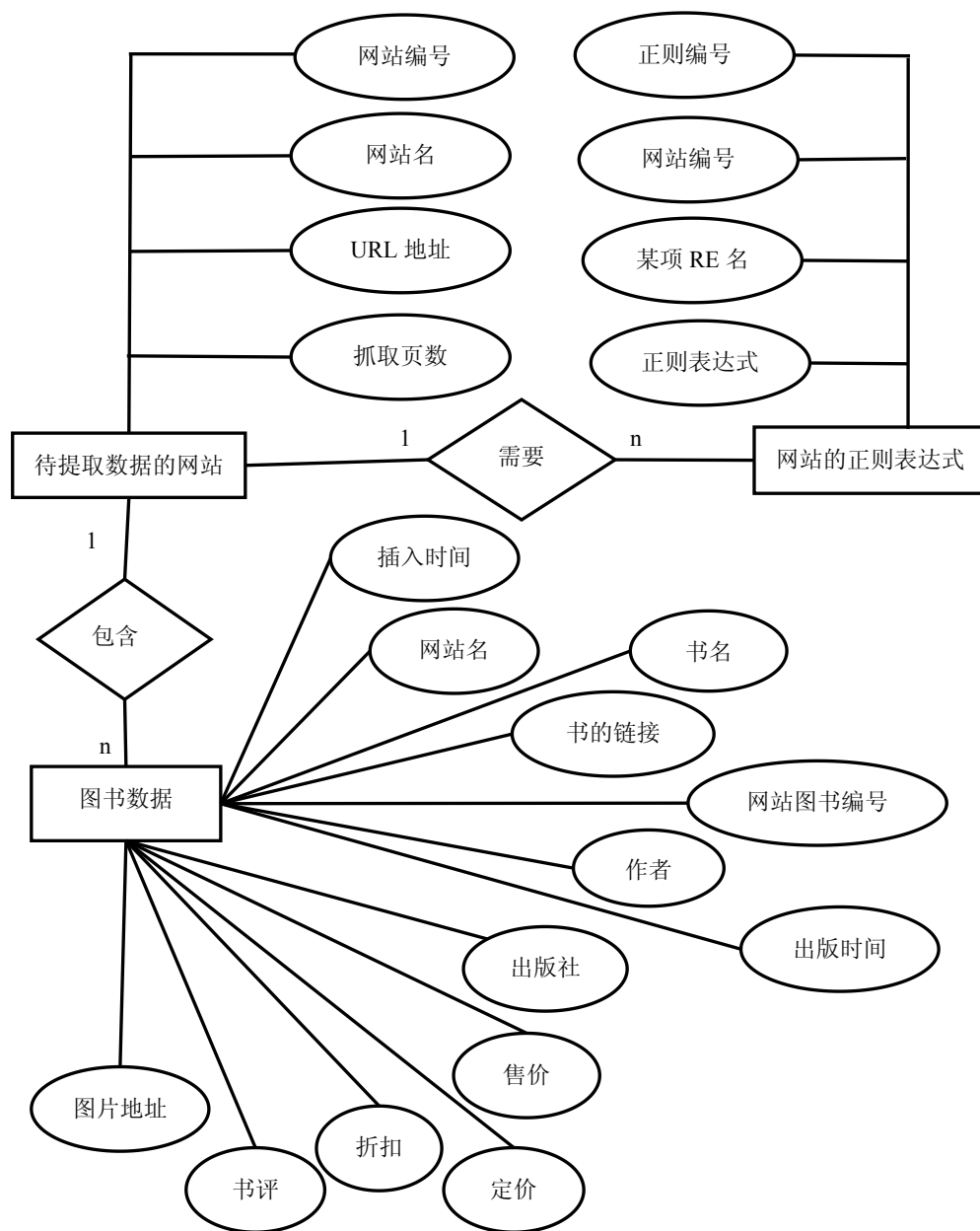


图 3-2 E-R 模型

3.2.2 表的具体设计

由上文的功能设计可以看出，爬虫程序的数据库中至少需要三个表，第一个表用来存储需要提取的网站和抓取页数，第二个表用来存储相应网站的正则表达式和其他信息，第三个表用来存储获取图书的数据信息，而由上面的 E-R 图可以清晰明了的看到这三个表之间的关系。

(1) 表 Table_web 的结构设计如表 3-1:

表 3-1 网站信息 Table_web 表

字段	类型	空值	长度	主键	备注
web_id	INT	否	10	▲	网站编号
web_name	VARCHAR	否	100		网站名
web_url	VARCHAR	否	255		地址
Web_num	INT	否	10		页面数目

存储网页地址和页面数目表的设计，表名为 Table_web，该表中有四列。第一列代表该网站在该表中的唯一编号，设为自动增长字段的主键，列名为 web_id；第二列代表该网站的名字，设为非空的字符字段，列名为 web_name；第三列为该网站的 URL 地址，设为非空的字符字段，列名为 web_url；第四列为该网站需要实际抓取的页数，设为非空的数字字段，列名为 web_num。

(2) 表 Table_re 的结构设计如表 3-2:

表 3-2 网站 RE 信息 Table_re 表

字段	类型	空值	长度	主键	备注
re_id	INT	否	10	▲	正则编号
web_id	INT	否	10		网站编号
row_name	VARCHAR	否	100		RE 名
re	VARCHAR	否	255		正则表达式

存储网页正则表达式表的设计，表名为 Table_re，该表中有四列。第一列代表该行在该表中的唯一编号，设为自动增长字段的主键，列名为 re_id；第二列代表该对应网站的唯一编号，值要与表 Table_web 中的编号保持一致，列名为 web_id；第三列为该网站对应 RE 的名字，设为可非空的字符字段，列名为 row_name；第四列为该网站具体的

正则表达式，设为非空的字符字段，列名为 re。

(3) 表 book 的结构设计如表 3-3:

表 3-3 图书信息 book 表

字段	类型	空值	长度	主键	备注
inser_time	VARCHAR	否	default		插入时间
web_name	VARCHAR	否	100	▲	网站名称
title	VARCHAR	否	255		书名
book_url	VARCHAR	否	255		书的链接
book_id	INT	否	10	▲	网站图书编号
author	VARCHAR	是	255		作者
time	VARCHAR	是	200		出版时间
publisher	VARCHAR	是	300		出版社
current_price	VARCHAR	是	100		售价
fixed_price	VARCHAR	是	100		定价
discount	VARCHAR	是	100		折扣
remarks	MEDIUMTEXT	是	default		书评
picture_url	VARCHAR	是	255		图片地址

存储图书各种信息表的设计，表名为 book，该表中有十三列。第一列代表图书信息插入的时间，设为时间字段而且插入时会自动生成值，列名为 inser_time；第二列代表这本图书爬取的网站名，设为非空的字符字段，值要与表 Table_web 中的网站名保持一致，列名为 web_name；第三列为该图书的名字，设为非空的字符字段，列名为 title；第四列为该图书的链接地址，设为非空的字符字段，列名为 book_url；第五列为该图书所在网站的编号，设为非空的整型字段，列名为 book_id；第六列表示图书的作者，设为可为空的字符字段，列名为 author；第七列表示图书的出版时间，设为可为空的字符字段，列名为 time；第八列表示图书的出版社，设为可为空的字符字段，列名为 publisher；第九列表示图书的销售价格，设为可为空的字符字段，列名为 current_price；第十列表示图书的定价，设为可为空的字符字段，列名为 fixed_price；第十一列表示图书的折扣，设为可为空的字符字段，列名为 discount；第十二列表示网站上图书的评论，设为可为空的文本字段，列名为 remarks；最后一列表示网页中该书

的图片地址，设为可为空的字符字段，列名为 picture_url。

3.3 具体模块设计

3.3.1 正则表达式的设计

使用正则表达式，能够提取出特定的字段，正则表达式在设计时，需要保证写出能够匹配目的信息的正确正则表达式。一个页面中有许多本书，那么就会需要匹配各本书整体信息的正则表达式，将匹配的每本书的字段存入一个列表中。还要提取图书的书名时，就要有一个专门匹配书名的正则表达式，除此之外还要提取图书的价格、出版社、出版时间、书的评论等等的信息，每提取一项信息时，就要有一个匹配出该信息的正则表达式。

3.3.2 对数据库操作部分的设计

对数据库的操作的设计依赖于数据库中表的设计，在程序运行时，频繁读取或者写入数据库会让程序的效率十分低下。若是数据库设计的不好，比如从一个网站中读取的数据存到一个表中，那就要尝试连接许多个表再写入，而且没有关联的表太多会让数据库的管理更加困难。对数据库的操作无外乎有连接数据库、读取数据库、写入数据库等，在实际操作时要使用 python 中的异常机制，对数据库尝试性地操作，若出现错误，再针对错误特定处理。

3.3.3 解析网页的设计

Python 的 html.parser 模块中有一个专门对网页 HTML 程序标签进行解析的类 HTMLParser，该类中有一个 feed 函数以标签的字符串作为参数，依次处理传入的标签字段。若使用这个类处理标签，需要编写一个继承此类的子类，在子类中重载处理开始标签、处理标签间内容、处理结束标签的函数，这样在子类的对象中调用 feed 函数，就能对标签正确处理了。还可以在子类中添加函数，以返回想要获取的值。

3.3.4 多线程的设计

Python 支持多线程编程，使用多线程可以提高程序运行效率。在最开始的设计中，本来想把抓取一个网站作为一个线程来运行，可是每个网站的正则表达式都不一样，这样设置运行的效果和一个一个执行程序没有两样，所以行不通。后来在设计中改变策略，

把网站的一个页面作为一个线程来执行，这样该网站的多个页面可以同时进行抓取，抓取一个网站后循环抓取其他的网站，直至抓取完毕。

3.3.5 生成网页 HTML 文本文档的设计

HTML 文档设计为多个界面，首先是主界面的 HTML 文档，这个需要自己动手编写，将显示各个网站书本的界面作为一个框架整合在这个主界面的文档中，这个 HTML 文档在程序运行过程中不变，而其他的 HTML 则是靠程序运行时生成并写入的。有两类 HTML 文档，一个作为存储网站名称的 HTML 文档，另一个是存储相应网站图书的 HTML 文档，当程序运行完毕后，生成这些文档后，再通过主界面在浏览器中显示，这样显示界面就完成了。

第 4 章 爬虫的实现与应用

4.1 编码问题

在打开一个文档后，需要用 read 函数读取文档的内容，之后会有一个不得不解决的问题，就是编码问题，最初开始写这个程序时，被编码问题绊了好几天，最后问题终于得到了解决。网页常见的编码有 UTF8、GBK 和 GB2312 等，这是我们经常浏览的网页最常用的一些编码。首先是要设置所使用的编辑器的编码，要在 Eclipse 中设置默认编码为 UTF8，之后要在程序的第一或第二行加上 ‘#coding = 'utf-8'’ 这样的注释，为了告诉 python 解释器所编辑的源程序的编码是 UTF8，这样程序的编辑界面中就可以正常显示程序的汉语注释了。Python 中用注释声明源程序的编码，这一情况在其他编程语言中也是不常见的，很有特性。这样就可以保证编辑器的编码和源程序的编码一致了，但是和网页的编码不一致那岂不是要出错了，不管想要获取的网页编码是什么，有一种统一的解决办法，这需要用到内置的解码函数 decode() 和编码函数 encode()。

在读取 HTML 文档后，先使用 decode() 函数将其解码为 Unicode 编码，之后再使用函数 encode() 编码为 UTF8，这样就解决了编码不一致的问题。程序中的具体调用为：
`data = f.read().decode('gb2312','ignore').encode('utf-8').decode('utf-8')`。f 为一打开的网页文本信息，假设 f 获取的网页编码为 gb2312。

编码正确的输出如图 4-1 所示：



图 4-1 编码正确的输出

图中打开的网页为当当网的一个界面，这样就实现了将网页信息读取存入字符变量 data 中，并保证了编码的正确性。

4.2 正则表达式的使用实例

正确获取 HTML 文档后，便要用到 re 模块来对数据进行匹配和获取了，这时所获取的文档作为一个很大的字符串暂时存储在一个字符串变量中。re 模块中 compile 函数的作用是将正则表达式的模式编译为正则表达式对象，然后将这种对象作为其他匹配函数的参数，获得与该正则表达式相匹配的数据。本设计中最开始抓取的数据是当当网销售图书的一个网页，这个网页中包含了许多本书，目的就是用正则表达式匹配每本书的信息，获取当当网的图书数据。

本设计致力于一个获取图书销售行业的图书信息，用一个程序来获取多个网站的数据，可以把爬虫程序看为一个获取多个网站数据的模板。在当当网的 HTML 文本信息中，可以写出获取整本书信息的正则表达式，将正则作为参数传入上面说所的函数中，返回一个正则表达式对象，该对象再调用 re 模块中的 findall() 函数，参数为整个网页的文本信息。调用该函数后，会返回一个列表，列表中的内容就是处理的网页中每本书的数据信息了。对于存有每本书信息的列表，列表中的每一项也是一个字符串，这时需要用一个循环，对每本书进行细节的数据提取，当然获取书的具体信息时，一样是要用到正

则表达式来匹配信息。使用 findall() 函数的具体调用为：

```
datapattern = re.compile(match['onebook'], re.DOTALL)
```

```
dataTuple = datapattern.findall(data)
```

match['onebook'] 是从数据库读取的正则表达式，通过 datapattern 调用 findall() 来将匹配的信息返回并存入到列表 dataTuple 中，列表的每一个元素为一个元组。如图 4-2：



```
Python 3.3.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os, re
>>> os.chdir('C:\\Users\\admin\\Desktop\\Web')
>>> f = open("menu.html")
>>> content = f.read()
>>> f.close()
>>> data1 = re.compile('<body>.*?</body>', re.DOTALL)
>>> dataTuple = data1.findall(content)
>>> print(dataTuple)
[('<body>\n<ul>\n<li>\n<a href = "default1.html" target = "show">当当网</a>\n</li>\n<li>\n<a href = "default2.html" target = "show">博库网</a>\n</li>\n<li>\n<a href = "default3.html" target = "show">互动网</a>\n</li>\n<li>\n<a href = "default4.html" target = "show">京东网</a>\n</li>\n</ul>\n</body>')]
```

图 4-2 findall() 函数使用示例

当匹配信息时，最需要的一个函数是 re 模块中的 search 函数了，它的调用形式是 re.search(pattern, string, flags=0)，通过扫描参数中的 string 字符串，来确定符合正则表达式 pattern 子字符串的位置，并返回相应的匹配对象，如果在字符串中没有找到与正则表达式相符的子字符串时会返回空值，而这里的空值和长度为零的字符串是不一样的。若是能找到符合规则的字符串是，返回值的类型并不是字符串，需要将其类型转化为字符串类型，这时需要调用 group(0) 函数，python 当中是允许方法连用的，通过符号 ‘.’ 来实现函数的连用，能使 python 程序更简洁易懂。使用过 search 函数后，得到的值是空值时，调用 group 函数转化为字符串时程序会出现错误，可以使用 try/except 的异常处理机制对程序运行出现的错误异常进行特别处理。程序中使用 re.search() 函数的具体调用为：

```
get_title = re.search(match['title'], each, re.DOTALL).group(0)
```

match['title'] 存储从数据库读出的匹配书名的正则表达式，each 为整本书的标签部分，re.DOTALL 这一参数使得正则表达式中的 ‘.’ 也能够匹配空格换行等字符，防止了匹配信息的错误。

该函数实际使用示例如图 4-3：



```
Python 3.3.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os,re
>>> os.chdir('C:\\Users\\admin\\Desktop\\Web')
>>> f = open("menu.html")
>>> content = f.read()
>>> f.close()
>>> data2 = re.search('<body>.*?<ul>',content,re.DOTALL).group(0)
>>> print(data2)
<body>
<ul>
>>> |
```

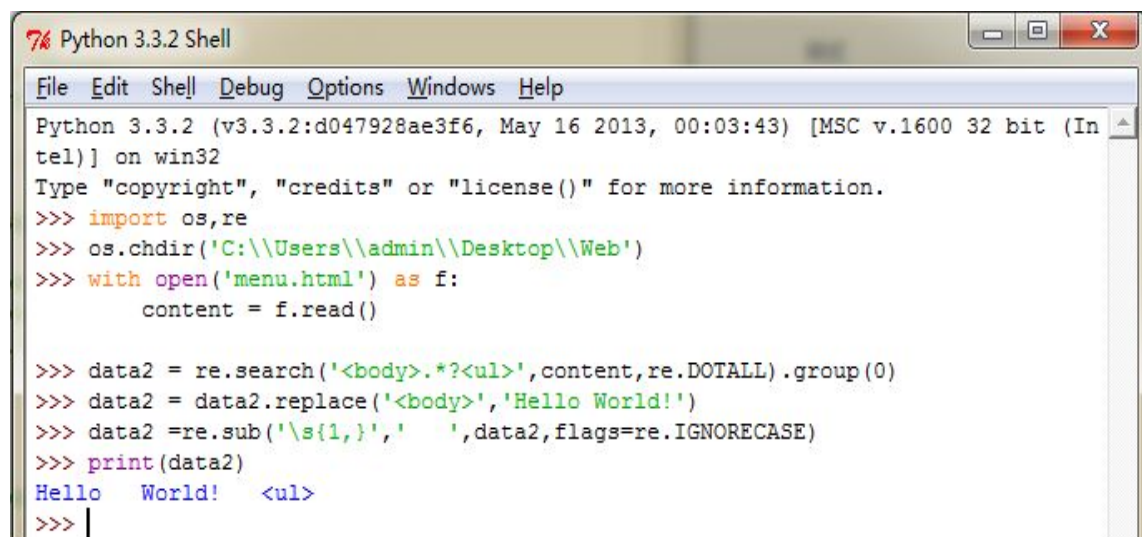
图 4-3 re.search() 函数使用示例

re 模块中的 sub 函数，它的调用形式是 re.sub(pattern, repl, string, count=0, flags=0)，它的作用是在 string 寻找符合正则表达式 pattern 的字符串，并将正则表达式 pattern 能匹配的信息替换为 repl 参数处的内容，count 和 flags 的默认参数均为 0。在 python 当中有一个内置函数 replace()，它的作用和 sub 函数很相似，也是将字符串中的字符替换为其他字符，不过这个函数只能替换一个字符，本设计中也有用到过这个内置函数。re.sub() 和 replace() 使用实例如下：

```
data = data.replace('/', '')
```

```
data =re.sub(r'\s{2,}',' ',data,flags=re.IGNORECASE)
```

该函数实际使用示例如图 4-4:



```
Python 3.3.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os,re
>>> os.chdir('C:\\Users\\admin\\Desktop\\Web')
>>> with open('menu.html') as f:
>>>     content = f.read()
>>>
>>> data2 = re.search('<body>.*?<ul>',content,re.DOTALL).group(0)
>>> data2 = data2.replace('<body>','Hello World!')
>>> data2 =re.sub('\\s{1,}',' ',data2,flags=re.IGNORECASE)
>>> print(data2)
Hello World! <ul>
>>> |
```

图 4-4 re.sub() 和 replace() 示例

第一个字符串调用 `replace()` 函数，可去掉 `data` 中的左斜杠字符；第二个是通过 `re.sub()` 使 `data` 字符串中长度大于等于 2 的空格或换行替换为一个空格字符。这样处理可以去掉提取的字符串中不必要的字符，以节省存储空间。参数 `flags=re.IGNORECASE`，能使得匹配时忽略字母的大小写。

4.3 读取网页信息

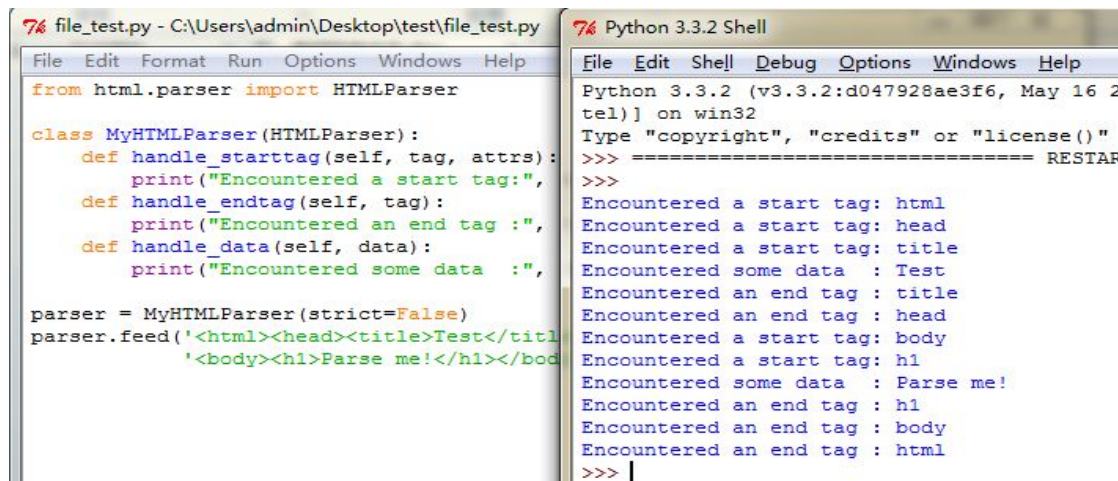
在应用正则表达式之前，首先要做的是通过一个网页的 URL 地址，获取该网页的 HTML 文档，当获取到文档时，便可以用正则表达式进行数据信息的匹配了，利用正则表达式的规则，获取用户想要的有用信息。那么怎么通过一个网页的 URL 地址而获取该网页的 HTML 文档呢，那就需要先导入 `urllib` 这个模块了。在程序中导入 `urllib.request` 模块，便可以直接使用里面的函数了，使用里面的 `urllib.request.urlopen()` 函数便可以打开一个网页的 HTML 页面了，该方法的参数是具体的 URL 地址。类似于文件的打开与关闭，用这个函数打开一个文档，相应的也要关闭该文档，不然容易出现异常，python 语法中有 `with` 关键字，使用这个关键词的话，打开一个文档后，会自动关闭该文档，为程序的简洁提供了帮助。打开并读取 HTML 文档的示例如上文图中图 4-1 所示。

4.4 网页解析与多线程的实现

4.4.1 网页解析

成功获取网页之后，就要开始解析 HTML 文档的工作了。在程序开始时，使用 `from html.parser import HTMLParser` 语句直接导入模块中的类，这样在编程时就可以直接使用这个类的方法或者加以继承。在本设计中，在获取每本书的各种信息时，除了用正则表达式相关的函数或字符串外，最重要的就是继承了类 `HTMLParser` 的子类，使用子类的重载方法和其他方法可以轻松对字符串中的标签进行解析和过滤。类 `HTMLParser` 中有 `handle_starttag`、`handle_endtag`、`handle_data`、`handle_comment` 等函数可供重载，分别是处理开始标签、结束标签、标签间的数据、处理 HTML 中注释信息。

该类的使用继承和使用如图 4-5 所示：

The image shows a Python IDE with two windows. The left window, titled 'file_test.py', contains a class 'MyHTMLParser' that inherits from 'HTMLParser'. It overrides 'handle_starttag', 'handle_endtag', and 'handle_data' methods to print out the structure of an HTML document. The right window, titled 'Python 3.3.2 Shell', shows the execution of the script. It prints out the encountered start and end tags and the data content, such as 'html', 'head', 'title', 'Test', 'body', 'h1', and 'Parse me!'.

```
file_test.py - C:\Users\admin\Desktop\test\file_test.py
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Encountered a start tag:", tag)
    def handle_endtag(self, tag):
        print("Encountered an end tag :", tag)
    def handle_data(self, data):
        print("Encountered some data :", data)

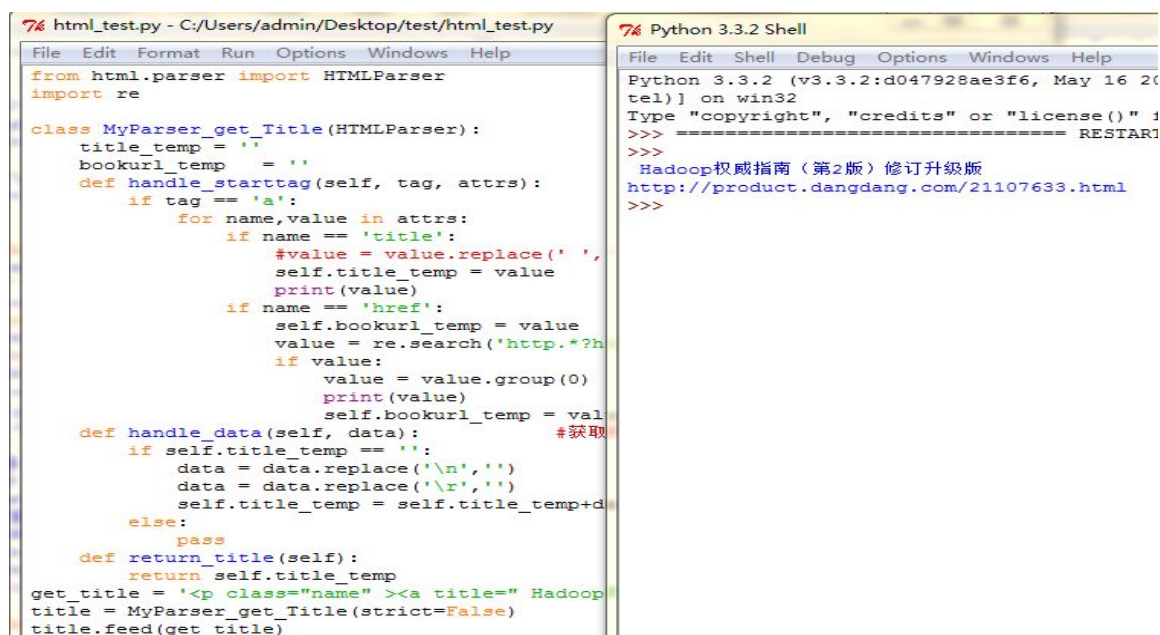
parser = MyHTMLParser(strict=False)
parser.feed('<html><head><title>Test</title><body><h1>Parse me!</h1></body>')

Python 3.3.2 Shell
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2012) on win32
Type "copyright", "credits" or "license()"
>>> ===== RESTART
>>>
Encountered a start tag: html
Encountered a start tag: head
Encountered a start tag: title
Encountered some data : Test
Encountered an end tag : title
Encountered an end tag : head
Encountered a start tag: body
Encountered a start tag: h1
Encountered some data : Parse me!
Encountered an end tag : h1
Encountered an end tag : body
Encountered an end tag : html
>>> |
```

图 4-5 标签解析示例

以提取一本书的标题为例，对设计程序中该类的应用进行说明。程序中使用继承了 HTMLParser 的子类，并在子类中编写函数和重写 HTMLParser 中可以重载的方法。

用实例如图 4-6 所示：

The image shows a Python IDE with two windows. The left window, titled 'html_test.py', shows a class 'MyParser_get_Title' that inherits from 'HTMLParser'. It overrides 'handle_starttag' and 'handle_data' methods to extract the title and book URL from an HTML document. The right window, titled 'Python 3.3.2 Shell', shows the execution of the script, which prints out the extracted title 'Hadoop权威指南（第2版）修订升级版' and the book URL 'http://product.dangdang.com/21107633.html'.

```
html_test.py - C:\Users\admin\Desktop\test\html_test.py
from html.parser import HTMLParser
import re

class MyParser_get_Title(HTMLParser):
    title_temp = ''
    bookurl_temp = ''
    def handle_starttag(self, tag, attrs):
        if tag == 'a':
            for name,value in attrs:
                if name == 'title':
                    #value = value.replace(' ', '')
                    self.title_temp = value
                    print(value)
                if name == 'href':
                    self.bookurl_temp = value
                    value = re.search('http.*?h', value)
                    if value:
                        value = value.group(0)
                        print(value)
                        self.bookurl_temp = value
    def handle_data(self, data):
        if self.title_temp == '':
            data = data.replace('\n','')
            data = data.replace('\r','')
            self.title_temp = self.title_temp+data
        else:
            pass
    def return_title(self):
        return self.title_temp

get_title = '<p class="name"><a title=" Hadoop权威指南（第2版）修订升级版'
title = MyParser_get_Title(strict=False)
title.feed(get_title)

Python 3.3.2 Shell
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2012) on win32
Type "copyright", "credits" or "license()"
>>> ===== RESTART
>>>
Hadoop权威指南（第2版）修订升级版
http://product.dangdang.com/21107633.html
>>>
```

图 4-6 继承类 HTMLParser 使用实例

从图中可看到子类名为 MyParser_get_Title 的子类，重载了 handle_starttag 和 handle_data 的方法，因为这两个方法已经可以满足提取网页中书名字段的需求，亦无需重载其他方法了，子类中也有自己的方法 return_title，用来返回所提取的书名信息。handle_starttag 的参数中 attrs 获取标签中的值，它本身为一个列表，每一个元素为一元组，如图中所示，由自己编写方法里面的代码，具体处理标签。handle_data 获取

标签之间的书名，之所以在这里还是用这个方法，是因为有些网站的书名在开始标签中可以获取的到，而有些网站的书名只能在标签之间获取，需用这个方法进行提取，该子类中方法 `return_title` 用来返回所获取的书名字符串。在程序中定义一个该类的对象，使用实例如下：

```
title = MyParser_get_Title(strict=False)
```

```
title.feed(get_title)
```

`get_title` 为包含书名信息的标签字符串，该类的对象为 `title`，`title` 调用 `feed` 函数来对字符串进行解析，解析过程仅仅使用在子类重载的函数。

4.4.2 多线程与锁

若是不考虑爬虫的多线程，只是过程性地对网页读取解析，那么程序万一出现错误，整个程序就会停止执行，而且速度低下，特别是还有读写数据库的操作或者是下载的操作，都能降低程序的运行效率。就像最先设计中，以一个网站作为单个线程，一个线程中顺序地处理网页，总体来看和顺序处理的过程没区别。现在的实现是以单个网页作为一个线程，顺序地处理。每提取一个网站时，同时运行多个页面的信息提取线程，将生成的线程添加到一个列表中，使用循环使列表中的线程依次启动并运行，直至所有子线程运行完毕。

图 4-7 中是一个线程运行的示例：

The screenshot shows a Python IDE window titled 'threading_lock.py - C:\Users\admin\Desktop\test\threading_lock.py'. The code defines a `myThread` class that inherits from `threading.Thread`. It includes methods for initialization (`__init__`), running (`run`), and printing time (`print_time`). The `run` method uses a lock to ensure thread safety when printing. The main thread creates two `myThread` instances, starts them, and then joins them before exiting.

Overlaid on the IDE is a 'Python 3.3.2 Shell' window showing the execution output. The output indicates that two threads, 'Thread-1' and 'Thread-2', are running concurrently, each printing their name and the current time. The main thread prints 'Exiting Main Thread' after both child threads have completed their execution.

```
#!/usr/bin/python
import threading
import time

class myThread (threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter
    def run(self):
        print ("Starting " + self.name)
        # 获得锁，成功获得锁定后返回
        # 可选的timeout参数不填时将一直占用直到超时
        threadLock.acquire()
        print_time(self.name, self.threadID, self.counter)
        # 释放锁
        threadLock.release()
    def print_time(threadName, delay, counter):
        while counter:
            time.sleep(delay)
            print ("%s: %s" % (threadName, time.ctime(time.time())))
            counter -= 1

threadLock = threading.Lock()
threads = []
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 1)
thread1.start()
thread2.start()
threads.append(thread1)
threads.append(thread2)
for t in threads:
    t.join()
print ("Exiting Main Thread")
```

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:00:00) on win32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART =====
>>>
Starting Thread-1Starting Thread-2
Thread-1: Tue Jun 17 16:12:55 2014
Thread-1: Tue Jun 17 16:12:56 2014
Thread-1: Tue Jun 17 16:12:57 2014
Thread-2: Tue Jun 17 16:12:59 2014
Thread-2: Tue Jun 17 16:13:01 2014
Thread-2: Tue Jun 17 16:13:03 2014
Exiting Main Thread
>>>
```

图 4-7 多线程运行示例

图中的示例是使用两个线程来输出线程名和当前时间，而设计中的程序思路是和图 4-7 是一致的，图 4-7 中有加锁和解锁的使用，而设计中也一样要使用加锁和解锁，以防止数据混乱。程序中的线程使用实例为：

```
webpage = 1
for each_url in url_list:

    task_threads.append(threading.Thread(target=whole_page, args=(each_url, webpage)))

    webpage = webpage+1

for task in task_threads:#启动线程
```



```

task.start()

for task in task_threads:#等待所有线程结束

    task.join(5)

```

以上程序便是线程的具体应用了，在每一线程中为防止网页之间的图书信息发生错乱，就要把语句 `lock.acquire()` 和 `lock.release()` 分别放置在处理一本书的程序代码开始和结束处，这样每次只能有一本书获得加锁，而其他线程不能获得加锁，每次只提取一本书信息就不会混乱了，解锁后又有可能被其他线程获得加锁处理，当所有线程运行完后也就都结束了。

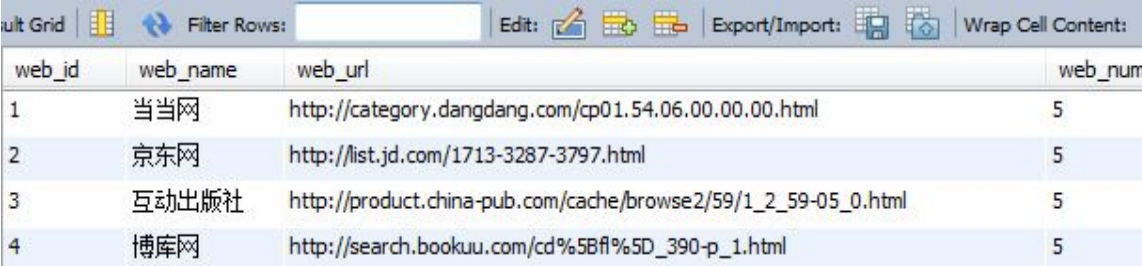
4.5 程序与数据库

4.5.1 数据库的具体实现

数据库的设计在上一章已经说明完毕，现在说说数据库中三个表的具体实现，第一个存储网站地址的 `Table_web` 表和存储正则表达式的 `Table_re` 表，需要自行建表和存入数据。

建立完这两个表之后，就需要往这两个表中添加数据，可以选择用语句插入，也可以在可视化界面中直接往表中添加数据。

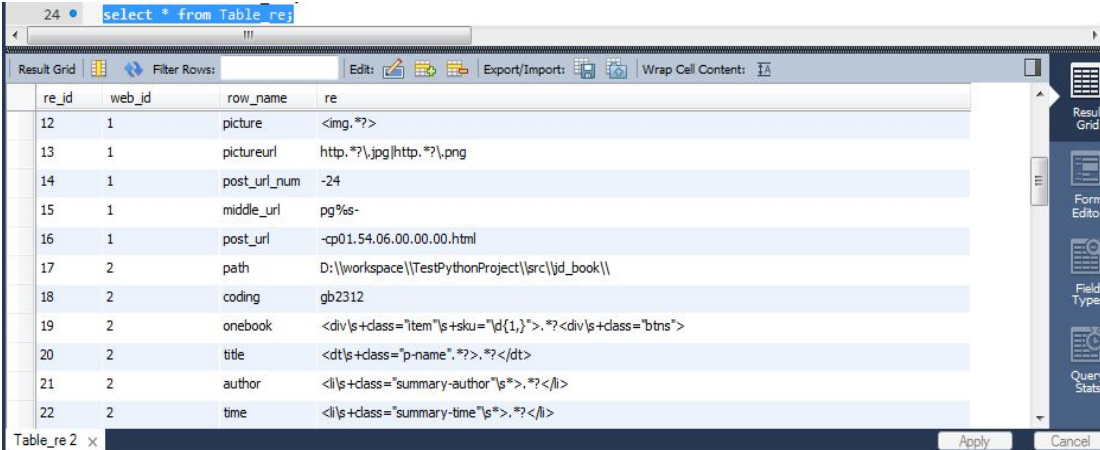
添加完的网站地址数据如图 4-8 所示：



web_id	web_name	web_url	web_num
1	当当网	http://category.dangdang.com/cp01.54.06.00.00.00.html	5
2	京东网	http://list.jd.com/1713-3287-3797.html	5
3	互动出版社	http://product.china-pub.com/cache/browse2/59/1_2_59-05_0.html	5
4	博库网	http://search.bookuu.com/cd%5Bfl%5D_390-p_1.html	5

图 4-8 存 URL 的 `Table_web` 表数据

添加完的正则表达式数据如图 4-9 所示：



re_id	web_id	row_name	re
12	1	picture	<img.*?>
13	1	pictureurl	http.*?\.jpg http.*?\.png
14	1	post_url_num	-24
15	1	middle_url	pg%6s-
16	1	post_url	-cp01.54.06.00.00.00.html
17	2	path	D:\workspace\TestPythonProject\src\jd_book\
18	2	coding	gb2312
19	2	onebook	<div\s+class="item"\s+sku="id{1,}">.*?<div\s+class="btns">
20	2	title	<dt\s+class="p-name".*?>.*?</dt>
21	2	author	<li\s+class="summary-author"\s*>.*?
22	2	time	<li\s+class="summary-time"\s*>.*?

图 4-9 存 RE 的 Table_re 表数据

只有这样预先添加完数据后，设计的网络爬虫才能读取这些数据并爬取网页数据。表中的正则表达式在需要添加某一项时，需结合网页具体的 HTML 文档进行编写，在编写过程中可能会需要用到转义字符，比如添加某一路径就用到了转义字符。MySQL 中对于插入的 re 表达式处理时，要在需转义的字符前面加上 ‘\’ 字符，在插入数据时表示一次转义，而正则表达式中也可能有用 ‘\’ 处理的转义字符，在添加数据时这些都需要特定处理。

4.5.2 程序中数据库的应用

程序中的数据库操作应用，有很多。第一步的应用就是把程序连接到名为 book 的数据库，首先要设置连接数据库的参数，然后通过第三方下载的 mysql.connector 模块中的函数执行连接数据库的操作。其他地方的应用有使用这个函数读取表 Table_web 和表 Table_re 中的数据，而且这个函数还将提取的图书信息写入到表 book 中，在使用此函数对数据库读或写之前，要写出语法正确的 SQL 语句，把该语句的字符串形式作为参数传入到执行函数中。连接数据库的参数为：

```
config={'user': 'root',
        'password': 'lihao',
        'port': 3306,
```

```
'host': '127.0.0.1',

'database': 'book',

'charset': 'utf8'}
```

连接的参数为一个字典，将其传入到 `cnx=mysql.connector.connect(**config)` 语句中就能连接数据库，经过语句 `cursor = cnx.cursor()` 生成游标，该游标直接调用 `execute` 函数来执行对数据库的操作，如 `cursor.execute(string)`。读取完规则表中的数据后，在程序运行的每个线程中，每次都需将获取的数据写入 book 信息表中。

程序运行的结果如图 4-10 所示：

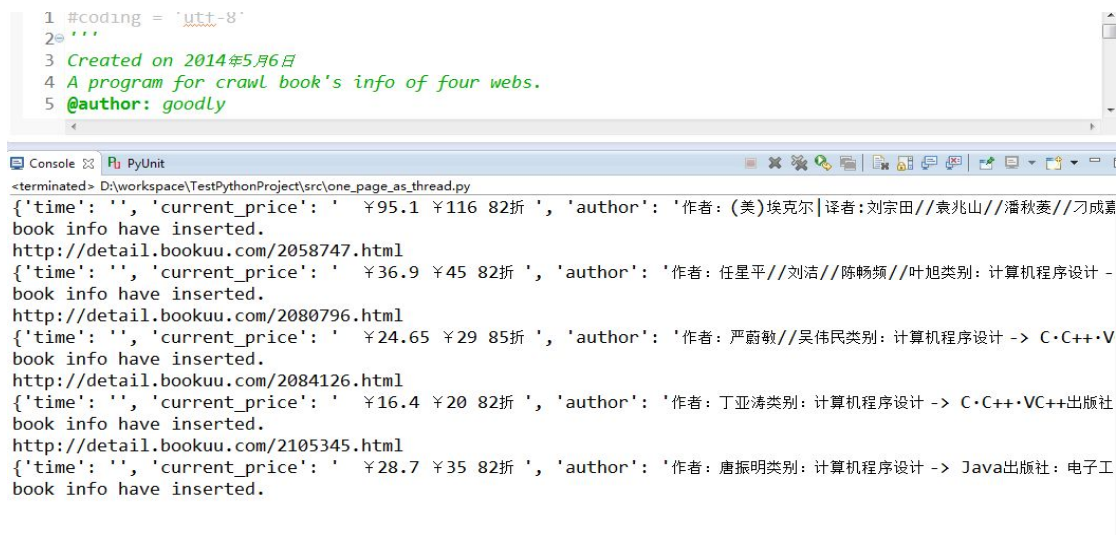


图 4-10 程序运行效果图

程序每次都要写入数据库中，为了降低写入数据库带来的效率影响，可以首先对获取的数据进行判断，若是数据库已有的数据便不插入，若数据库没有数据就执行插入操作。程序中的 `judgement` 列表中保存已插入数据库数据的书名和编号，每次都先判断再执行相应操作。程序运行完毕后，数据便写入了数据库的 book 表中。

写入表中的数据如图 4-11 所示：

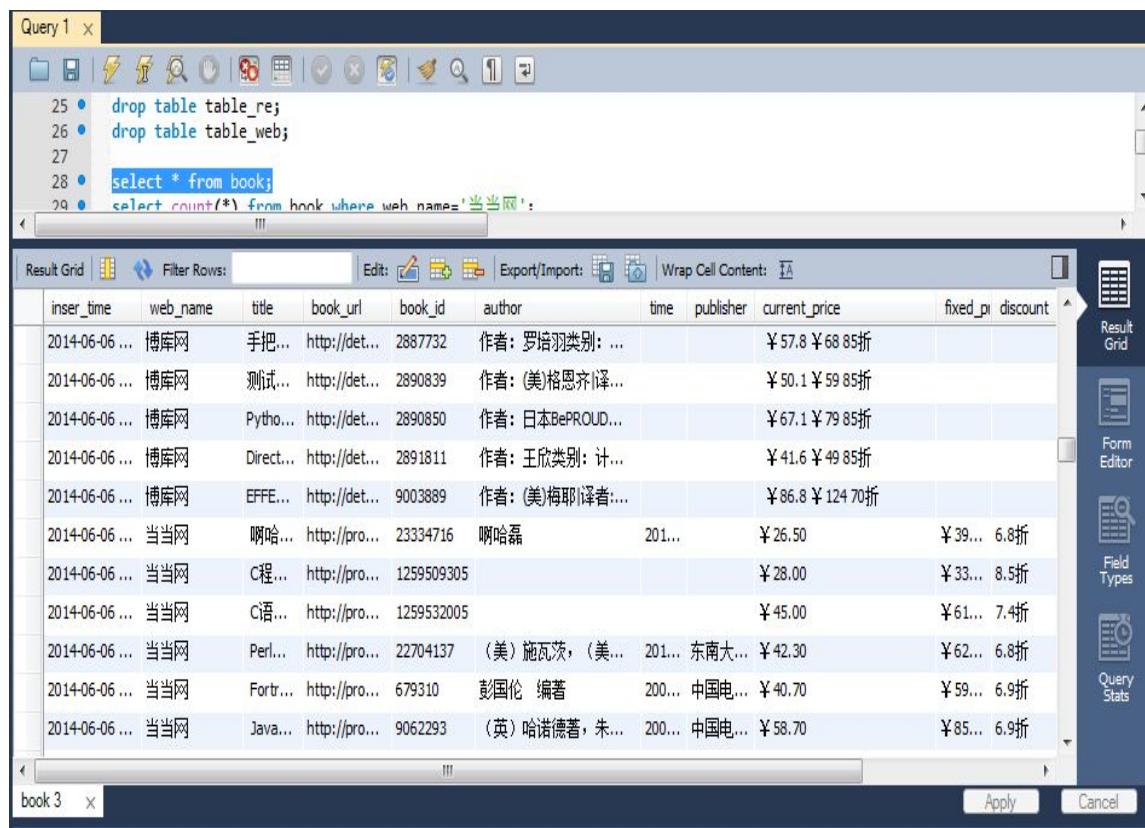


图 4-11 插入数据库 book 表的数据

在对数据库实际操作时，要使用try/except异常处理机制来尝试性执行操作。在程序的最后需要一些语句对与数据库的联系做最后处理，如cnx.commit()的事务提交，cursor.close()和cnx.close()的关闭连接操作。

4.6 生成 HTML 显示界面

4.6.1 主界面文档的编写

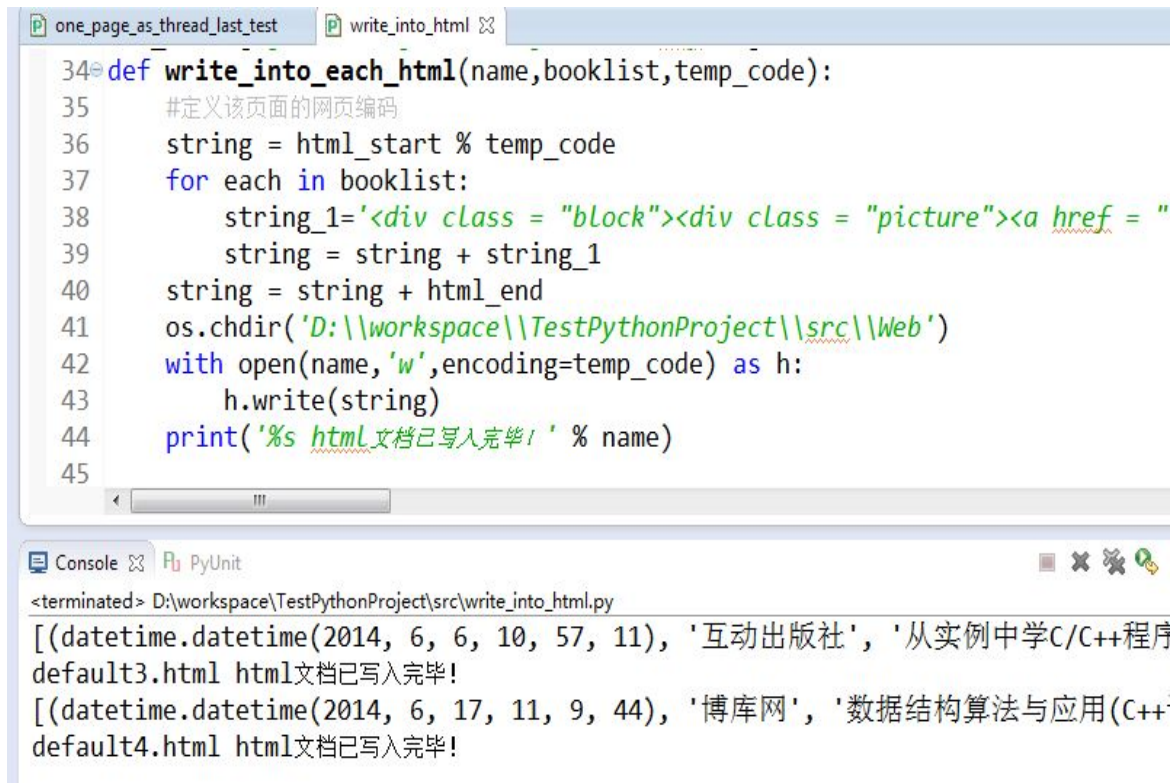
为用户提供抓取数据的显示界面，需要编写或生成 HTML 文档，网页的编码设为gb2312。整个页面有两个框架部分，一个是显示网站名称的菜单框架，另一个是显示相应网站书本信息的框架。在主界面的 HTML 文档编写时，使用框架集合将这两个页面框架集合在一起到浏览器中显示。在frameset 标签中设置cols 的属性值为“15%, 85%”，会使得菜单框架和图书显示框架的比例分别为 15%和 85%。在第一个 frame 标签中添加菜单的 HTML 文档，在第二个 frame 标签中添加图书界面的 HTML 文档，两个标签中都添加属性 noresize，其属性值为“noresize”，来确保界面比例的固定，第二个标签添加属

性 name = "show", 来显示不同网站的界面, 实现的效果为点击界面中的某一网站名, 右边的框架总就会显示相应网站的图书信息。

4.6.2 子界面的生成与写入

主界面完成后, 菜单界面和图书显示界面的 HTML 代码需要用 Python 程序中的字符串拼接形成, 之后写入文档, 每次运行程序就会重新写入。不用界面的 HTML 代码的模式是一样的, 只需要改变写入的图书数据。需要注意的一点是程序每次运行时, 所爬取的网页图书数据会不同, 会有新的数据被写入数据库, 数据库中的图书数量就会增加, 那么写入 HTML 文档中的图书信息也会增加, 这样就保证了图书信息的更新。

生成界面的程序源码和输出如图 4-12 所示:



```
34 def write_into_html(name, booklist, temp_code):
35     # 定义该页面的网页编码
36     string = html_start % temp_code
37     for each in booklist:
38         string_1 = '<div class = "block"><div class = "picture"><a href = "'
39         string = string + string_1
40     string = string + html_end
41     os.chdir('D:\\workspace\\TestPythonProject\\src\\Web')
42     with open(name, 'w', encoding=temp_code) as h:
43         h.write(string)
44     print('%s html文档已写入完毕!' % name)
45
```

```
<terminated> D:\workspace\TestPythonProject\src\write_into_html.py
[(datetime.datetime(2014, 6, 6, 10, 57, 11), '互动出版社', '从实例中学C/C++程序 default3.html html文档已写入完毕!
[(datetime.datetime(2014, 6, 17, 11, 9, 44), '博库网', '数据结构算法与应用(C++ default4.html html文档已写入完毕!
```

图 4-12 生成显示界面的 HTML 文档

图 4-12 中程序能生成图书的显示界面, 首先将需要生成的 HTML 文档的开头和结尾部分先写好, 当从数据库读取完每本书各方面信息之后, 将这些信息添加到一本书的布局中, 把这部分的 HTML 程序添加到文档的中间, 最后再把整个字符串写入到新的 HTML 文档中, 就生成了一个显示该网站图书信息的子界面, 其它子界面的生成过程也都是通过图 4-12 中的函数而实现的。

4.6.3 实验运行和调试

Pycharm 对测试脚本提供了灵活的运行和调试支持。

运行 MyParser_get_Title 文件下所有的测试类

运行测试类的所有测试脚本

运行一个测试类的某个测试脚本

如果要断点调试，则使用 Debug 模式，即可对单个函数运行和断点调试了。

通过对 testSuit 操作，可以实现以上功能，但是 IDE 却提供了更灵活直接的选择。

选择在 IDE 中运行此程序，会看到如下效果图 4-13：

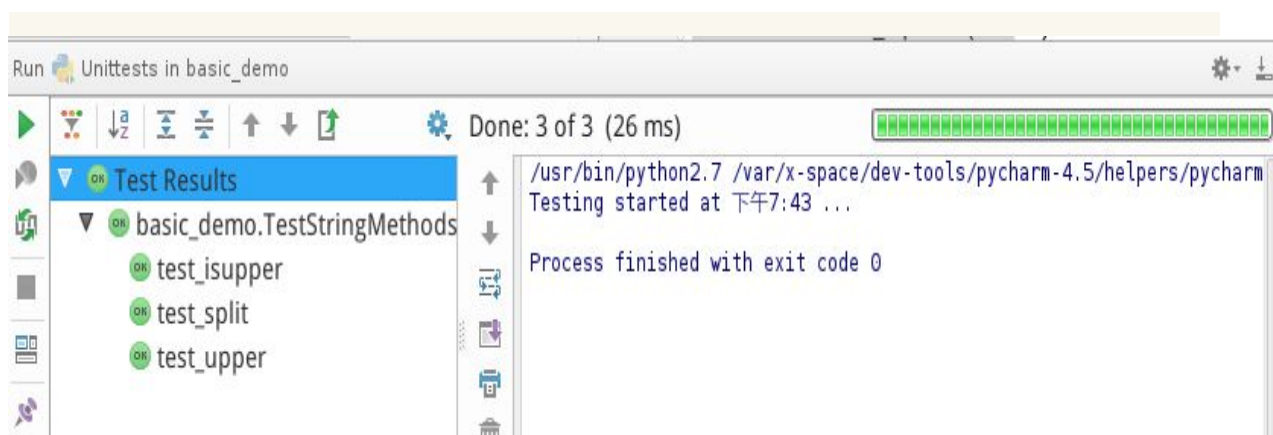


图 4-13 IDE 运行程序图

可以看到全部运行通过。如果刻意将其中一个弄成不通过的，则会显示如下的结果图 4-14：

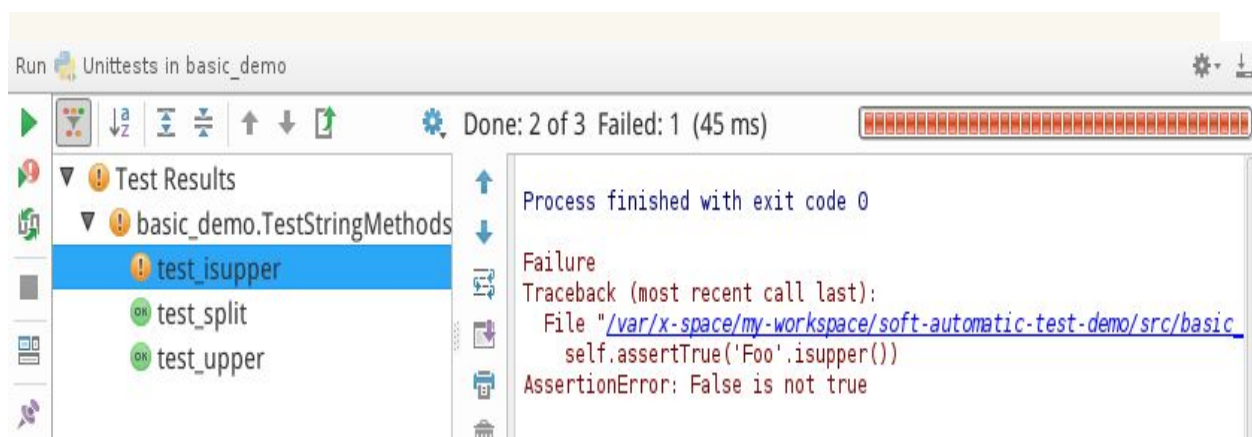
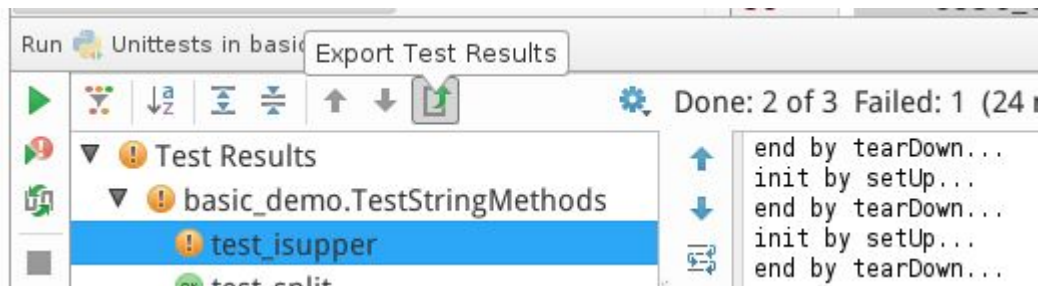


图 4-14 IDE 断点调试运行程序图

在 Pycharm 测试结果显示框上的一个功能按钮上生成测试报告图 4-15。



程序运行结束后形成的最终页面中，每本书所占页面的大小是一样的，而且当鼠标移动到某一本书时，会显示该本图书的详细信息。

具体如图 4-16 所示：



图 4-16 最终的显示界面

由上图可以看到，通过程序所爬取的数据，已成功写入到 HTML 文本中，并为用户显示显示到一个界面当中。整个程序实现了数据抓取、数据库存储、信息聚合等功能，到此整个设计便结束了。

4. 6. 4 性能比较

本文所提出的网络应用框架与其它的 Python Web 框架相比，速度要快很多。下面针对最简单的 Hello, world 例子作了一个测试。对于 Django 和 web.py，我们使用 Apache/mod_wsgi 的方式来带，CherryPy 就让它自己裸跑。这也是在生产环境中各框架常用的部署方案。对于我们的 方法，使用的部署方案为前端使用 nginx 做反向代理，带动 4 个线程模式的网络应用框架，这种网络应用框架也是我们推荐的在生产环境下的部署框架（根据具体的硬件情况，我们推荐一个 CPU 核对应一个网络应用框架伺服实例，我们的负载测试使用的是四核处理器）。我们使用 Apache Benchmark (ab)，在另外一台机器上使用了如下指令进行负载测试：

在 AMD Opteron 2.4GHz 的四核机器上，结果如下图 4-17 所示：

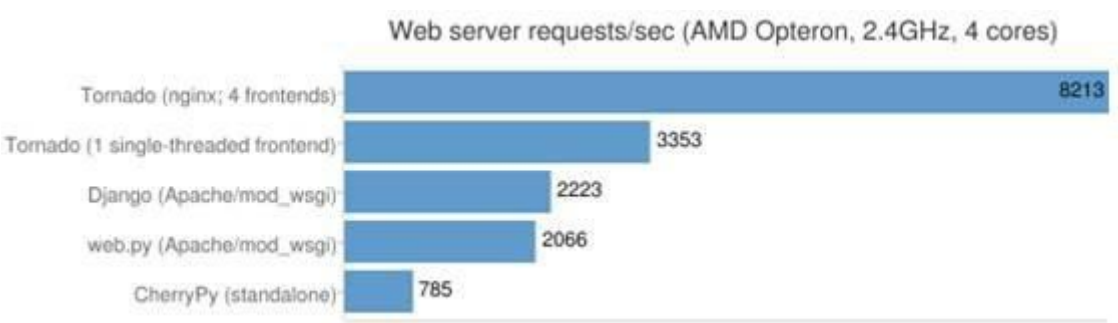


图 4-17 测试效果比较图

在我们的测试当中，相较于第二快的服务器，我们提出的网络应用框架法在数据上的表现也是它的 4 倍之多。即使只用了一个 CPU 核的裸跑模式，我们提出的网络应用框架也有 33% 的优势。”

结 论

整个网络爬虫能够爬取当当网、京东网、互动出版社、博库网等网站上的图书信息，实现了信息的聚合和数据的处理。整个程序的设计思路是从数据库读取网页地址以及该网页的正则表达式规则，爬虫程序通过地址读取整个网页，再通过正则表达式在整个网页信息中匹配出想获取的数据。在爬取数据的过程中，使用多线程来同时抓取一个网站的多个页面，使用加锁与解锁保证每本图书信息的正确无误。爬取数据结束后，将爬到的数据按照一定的顺序存入到 MySQL 数据库中，等到使用数据的时候，通过程序来读取数据库，再将读取的数据处理成想要的结果就可以了。最后需要做的是为用户提供显示界面，当数据从数据库读出后，写入 HTML 文本当中，就可以显示最后的结果了。若是还想爬取其他网站的信息，只需要在数据库中添加网站名和该网站 URL 地址便可以了，并且在相应的表中添加该网站的正则表达式即可。Python 的源代码是以‘.py’为后缀来存储在硬盘上的，若想保证代码不被泄露或者想为用户的使用提供方便，可以将代码文件编译为后缀为‘.pyc’的字节码文件，点击该字节码文件程序就能自己运行了。

参考文献

- [1] Jon Duckett. HTML&CSS 设计与构建网站[M]. 刘涛, 陈学敏, 译. 北京: 清华大学出版社, 2013.
- [2] Mark Lutz. Python 学习手册[M]. 李军, 刘红伟, 译. 4 版. 北京: 机械工业出版社, 2011.
- [3] Doug Hellmann. Python 标准库[M]. 刘炽, 译. 北京: 机械工业出版社, 2012.
- [4] Paul Barry. Head First Python[M]. 林琪, 郭静, 译. 北京: 中国电力出版社, 2012.
- [5] 齐鹏, 李隐峰, 宋玉伟. 基于 Python 的 Web 数据采集技术[J]. 电子科技, 2012, 25 (11): 118-120.
- [6] 张超, 闫宏印. 多线程网络爬虫的设计与实现[J]. 电脑开发与应用, 2012, 25 (6): 65-70.
- [7] 杜亚军, 严兵, 宋亮. 爬行虫算法设计与程序实现[J]. 计算机应用, 2004, 24 (1): 33-35.
- [8] 倪贤贵, 蔡明. 基于链接结构和内容相似度的聚焦爬虫系统[J]. 计算机工程与设计, 2008, 29 (7): 1709-1763.
- [9] 胡晟. 基于网络爬虫的 Web 挖掘应用[J]. 软件, 2012, 33 (7): 145-147.
- [10] 周立柱, 林玲. 聚焦爬虫技术研究综述[J]. 计算机应用, 2005, 25 (9): 1965-1969.
- [11] Ritika Hans, Gaurav Garg. Web Crawlers and Search Engines[J]. International Journal of Engineering Sciences and Research Technology, 2013, 2 (6): 1545-1551.
- [12] Prashant Dahiwal, M. M. Raghuvanshi, Latesh Malik. Design of Improved Web Crawler By Analysing Irrelevant Result[J]. International Journal of Computer Science and Mobile Computing, 2013, 2 (8): 243-247.
- [13] Dhiraj Khurana, Satish Kumar. Web Crawler: A Review[J]. International Journal of Computer Science and Management Studies, 2012, 12(1): 401-405.

作者简介及在学期间所取得的科研成果

作者简介：王朝阳、男、1979 年 2 月 2 日、汉族、出生地哈尔滨市宾县；文学学士学位、助理研究员；本科、齐齐哈尔大学外国语学院教科办副科级员；联系方式

致 谢

首先，必须感谢我的导师李雄飞老师，感谢老师对本论文的悉心指导，感谢老师的严格督促。从设计开题到设计完成，离不开老师的帮助、支持。大学期间深受老师的广博学识、严谨治学态度、积极的人生态度影响，不仅传授于我丰富的专业技能，还为我指引前进方向。论文编写过程中，在老师悉心指导和认真修改下，最终完成了论文编写工作。

其次，要感谢学院各老师的谆谆教导，在学习、生活上受到各老师的关照，我能学到丰富的专业知识及顺利完成学业，都得归功于学院老师教诲。

最后，感谢读研期间同学的帮助与支持，让我度过了一个有意义的大学生活