目录

[1、需求分析 1](#_Toc500780991)

[1.1系统目标 1](#_Toc500780992)

[1.2系统功能需求 1](#_Toc500780993)

[1.3系统的输入输出需求 2](#_Toc500780994)

[1.4系统的性能需求 2](#_Toc500780995)

[2、可行性分析 2](#_Toc500780996)

[2.1 可行性研究前提 2](#_Toc500780997)

[2.2 要求 3](#_Toc500780998)

[3、数据库详细设计 4](#_Toc500780999)

[3.1 概念结构设计 4](#_Toc500781000)

[3.2 逻辑结构设计 5](#_Toc500781001)

[3.3 物理结构设计 6](#_Toc500781002)

[4、实现思路及相关技术 7](#_Toc500781003)

[4.1 开发条件 7](#_Toc500781004)

[4.2 系统界面设计 8](#_Toc500781005)

[4.3 代码设计 14](#_Toc500781006)

[5、实现要点难点 21](#_Toc500781007)

[5.1前端页面设计 21](#_Toc500781008)

[5.2 python语言与数据库的交互 22](#_Toc500781009)

[6、总结 24](#_Toc500781010)

# 1、需求分析

## 1.1系统目标

二手物品发布平台是一个专为校内同学提供二手物品交易平台的系统。同学们可以首先注册账号，然后登录。登录之后可以在网站上浏览其他人或自己发布的二手物品及其相关信息，可以收藏自己喜欢的物品。用户也可以发布自己的二手物品。如果有感兴趣的物品，可以通过卖家留下的联系方式，线下通过第三方渠道进行支付、交易。

该系统的目标是建立一个学校自己的二手物品交易平台，为需要出售二手闲置物品和需要购置二手物品的同学提供平台。

## 1.2系统功能需求

该二手物品交易系统针对用户群体和系统管理员。

用户的功能需求如下图所示：

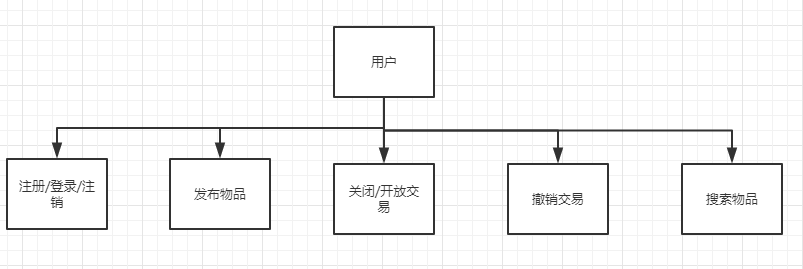


图1.1用户功能需求

管理员的功能需求包括管理所有已发布的物品、管理所有用户发布的评论，如下图所示：

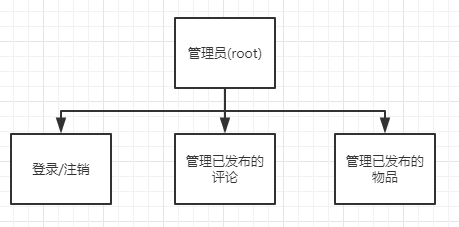


图1.2管理员功能需求

## 1.3系统的输入输出需求

系统的输入应满足最小量原则、简单性原则以及及早检验原则，即在满足处理要求的前提下应使输入量尽量小，系统能够计算出的数据不要由用户输入；输入过程应尽量简单；对输入数据的检验尽量接近原始数据发生点，使错误能及时得到改正。

系统的输出内容应简洁明了，使得用户一目了然；本系统中输出设备选择在屏幕上输出。

## 1.4系统的性能需求

系统的性能需求主要表现在数据库中的各个表需要频繁地被插入、删除以及更新。对于用户来说，系统地相应时间不宜太长，否则会降低用户体验，为此需要建立良好的表结构，加上足够的存储空间以及硬件性能。

# 2、可行性分析

## 2.1 可行性研究前提

随着互联网时代的飞速发展，人们的购物需求也在不断地增长。许多人也因此会产生许多闲置不用的物品，如何处理这些闲置的物品就成为一个难题。实地去二手市场进行买卖不失为一种选择，但是不符合互联网时代的“便捷”。在高校之中，学生们去二手市场进行交易显然是不可行的。因此，设计这样一个二手物品交易发布平台就显得十分必要。

系统要求界面简洁直观，交互方便。能方便管理员管理用户们发布的二手物品交易和用户发表的评论，维护一个干净的平台。对于用户而言，浏览、搜索自己想要的二手物品是必要的功能(包括收藏自己喜爱的物品)。除此之外，用户也应该能够管理自己已经发布的二手物品，可以进行关闭交易、打开交易、撤销交易等操作。

## 2.2 要求

2.2.1 **主要功能**

1.用户账户管理：用户可以在网站上自行注册及登录，离开时注销退出。

2.二手物品查询：用户以及管理员都可以在网站首页浏览所有用户发布的未关闭的二手物品，并且通过点击超链接进入查看详情。

3.用户评论管理：进入每一个已发布的未关闭的二手物品交易的详情页面之后，用户可以在下方发表评论。只有管理员可以管理删除这些评论。

4.二手物品管理：只有管理员可以管理所有人发布的二手物品交易，对于用户发布的不符合规范或者长期没有交易成功的物品撤销。

5.二手物品发布：所有非管理员用户均可发布二手物品，填写相应的资料和上传图片后即可发布。

6.搜索功能：可以根据物品名关键字进行搜索，搜索框中输入的字段必须完全包含在物品名称中。

2.2.2 **安全性**

具有一定的安全性。系统中只有两级权限级别，即普通用户和系统管理员。普通用户的权限仅限于发布物品、管理自己发布的物品、发布评论，系统管理员的权限更大，能够管理所有普通用户发布的物品和所有用户发布的评论。由于权限等级的划分，系统的安全性、稳定性得到了一定地满足。

2.2.3 **性能**

可以方便快捷地完成二手物品查询，发布等各项操作。有一定的输入数据合法性检查功能，如用户发布物品时上传的物品只能属于规定的格式集合，用户注册时输入的邮箱地址的合法性。

2.2.4 **可扩展性**

能够适应需求的变化，开发新的功能。数据库表结构清晰，不产生冗余。

# 3、数据库详细设计

## 3.1 概念结构设计

本二手物品交易平台中一共有用户、管理员、物品、评论这三个基本的模型(实体)。一个用户可以发布多个物品，而一个物品只能对应于一个用户。一个用户可以发布多条评论，而一条评论只能对应于一个用户。一个物品交易下可以有多条评论，但一条评论只能对应于一个物品。ER图如下：

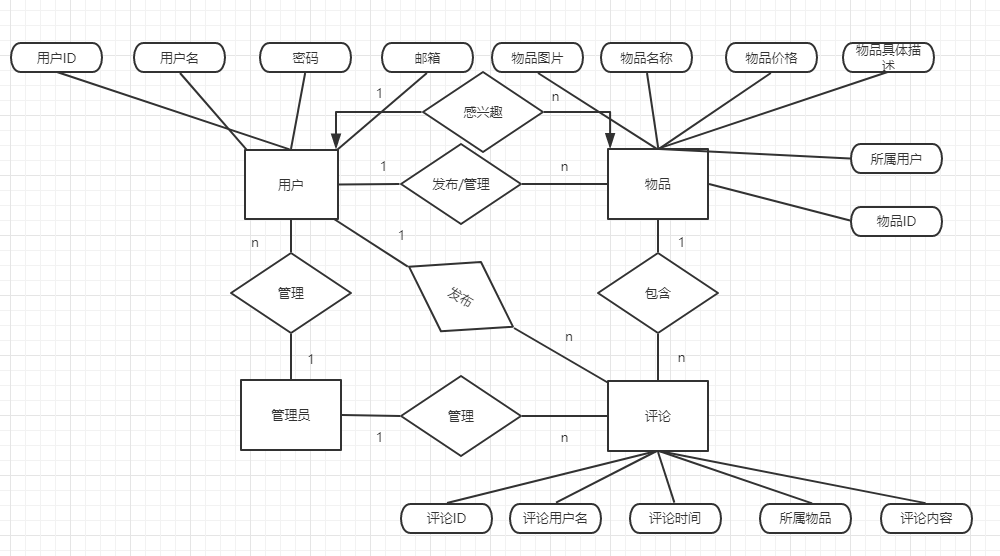


图3.1

用户(非管理员):用户ID、用户名、用户密码、用户邮箱。

物品：物品ID、物品名称、物品价格、物品详细描述、物品图片

评论：评论ID、评论内容、评论时间、评论所属物品、评论所属用户

感兴趣：用户ID、物品ID

管理员的属性和用户的属性基本相同，但管理员拥有管理所有发布的物品和发布的评论的功能，且管理员不能发布物品和评论。管理员是唯一的。

## 3.2 逻辑结构设计

将概念结构设计中的各个模型转化为DBMS支持的表结构，同时保持不会出现插入异常、删除异常和修改异常。表结构应该做到符合第三范式。在本系统中，一共有三个实体，实体转化为数据库模型为如下所示：

(1)实体转化为关系模式：

用户(用户ID、用户名、用户密码、用户邮箱、用户权限)

物品(物品ID、物品名称、物品价格、物品详细描述、物品图片)

评论(评论ID、评论内容、评论时间、评论所属用户ID、评论所属物品ID)

(2)一个一对多联系转化为一个关系模式

用户-物品(用户ID,物品ID)

用户-评论(用户ID,评论ID)

物品-评论(物品ID，评论ID)

兴趣(用户ID、物品ID)

(3)利用以上关系模式得到的所有数据表如下所示：

表3-1 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| Id | integer | 4 | Not null | 用户ID（主键） |
| Username | varchar | 20 | Not null | 用户名 |
| Password | varchar | 100 | Not null | 用户密码 |
| Email | varchar | 30 | Not null | 用户邮箱 |
| Authority | boolean | 1 | Not null | 用户权限 |

表3-2 物品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| Id | integer | 4 | Not null | 物品ID（主键） |
| Name | varchar | 30 | Not null | 物品名 |
| Description | text | 不定 | Null | 物品详细描述 |
| Price | flaot | 4 | Not null | 物品价格 |
| Imgpath | varchar | 300 | Null | 物品图片URL |
| Create\_time | datetime | 8 | Not null | 物品发布时间 |
| Is\_closed | integer | 4 | Not null | 交易是否被关闭 |
| Ownerid | Integer | 4 | Not null | 所属用户ID |

表3-3 评论表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| Id | integer | 4 | Not null | 评论ID（主键） |
| Content | text | 不定 | Not null | 评论内容 |
| Itemid | integer | 4 | Not null | 评论物品ID |
| Ownerid | integer | 4 | Not null | 评论所属用户ID |
| Create\_time | datetime | 8 | Not null | 评论创建时间 |

表3-4 兴趣表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 数据长度 | 可否为空 | 备注 |
| Id | integer | 4 | Not null | 兴趣ID（主键） |
| Userid | integer | 4 | Not null | 用户ID |
| Itemid | Integet | 4 | Not null | 物品ID |

其中除了兴趣之外的一对多关系模式在系统的具体实现中并没有建立实体的表结构，在使用flask框架中有相关的函数实现。

## 3.3 物理结构设计

1、选择关系模式的存取方式

对数据库逻辑结构设计中建立的表结构，用户表的用户ID属性唯一决定每一个用户元组，所以对用户表建立以用户ID为主关键字的索引。同理，对物品关系模式、评论关系模式也采用类似的索引存取方法。

2、数据表存储结构设计

本系统的所有数据表均存放在物理磁盘中。用户表、物品表和评论表的结构是相对稳定的，表中的已有记录是要长期保存的，在此基础上系统会相应用户的操作对数据表进行增、删、查、改等操作。

# 4、实现思路及相关技术

## 4.1 开发条件

* 4.1.1 开发语言

系统使用的开发语言是Python。Python是一种解释型、面向对象、动态数据类型的高级程序设计语言。Python 是由 Guido van Rossum 在八十年代末和九十年代初，在荷兰国家数学和计算机科学研究所设计出来的。Python 本身也是由诸多其他语言发展而来的,这包括 ABC、Modula-3、C、C++、Algol-68、SmallTalk、Unix shell 和其他的脚本语言等等。

Python语言的特点有易于学习、易于阅读、易于维护、广泛的标准库和第三方模块、可移植性、可扩展性、提供同意的数据库接口API等等。正是因为Python语言拥有如此诸多的优秀特性，选择它作为开发二手物品交易平台能够使得整个开发、调试过程更加高效。

* 4.1.2 开发框架

本二手物品交易系统是基于Web的系统，以网站的形式呈现给用户。使用的Web开发框架是flask。Flask是一个使用Python编写的轻量级Web应用框架。其WSGI工具箱采用的是Werkzeug，模板引擎使用的是Jinja2。Flask也被称为 “microframework” ，因为它使用简单的核心，用 extension 增加其他功能。Flask没有默认使用的数据库、窗体验证工具。

* 4.1.3 集成开发环境(IDE)

编程所使用的集成开发环境是PyCharm,一款智能的Python编程集成开发环境。Pycharm拥有一般IDE具备的功能，如调试、语法高亮、项目管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。除此之外，Pycharm还提供了一些功能用于常用的Python的Web开发框架，如本系统用到的Flask框架，Pycharm就有非常友好的支持。

* 4.1.4 数据库管理系统(DBMS)

本系统使用的数据库管理系统是Mysql Community，版本号为5.7。MySQL是一个关系型数据库管理系统**，**由瑞典MySQL AB 公司开发，目前属于Oracle旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

系统中的数据库以及数据库中的所有关系模式都使用Mysql进行管理。

* 4.1.5 系统前端框架

由于本系统是Web应用，所以使用了html+css+js的方式实现前端页面。实现过程中参考了Bootstrap前端开发框架。Bootstrap是Twitter退出的一个用于前端开发的开源工具包。在设计前端页面时，参考了Bootstrap的相关开源代码。

## 4.2 系统界面设计

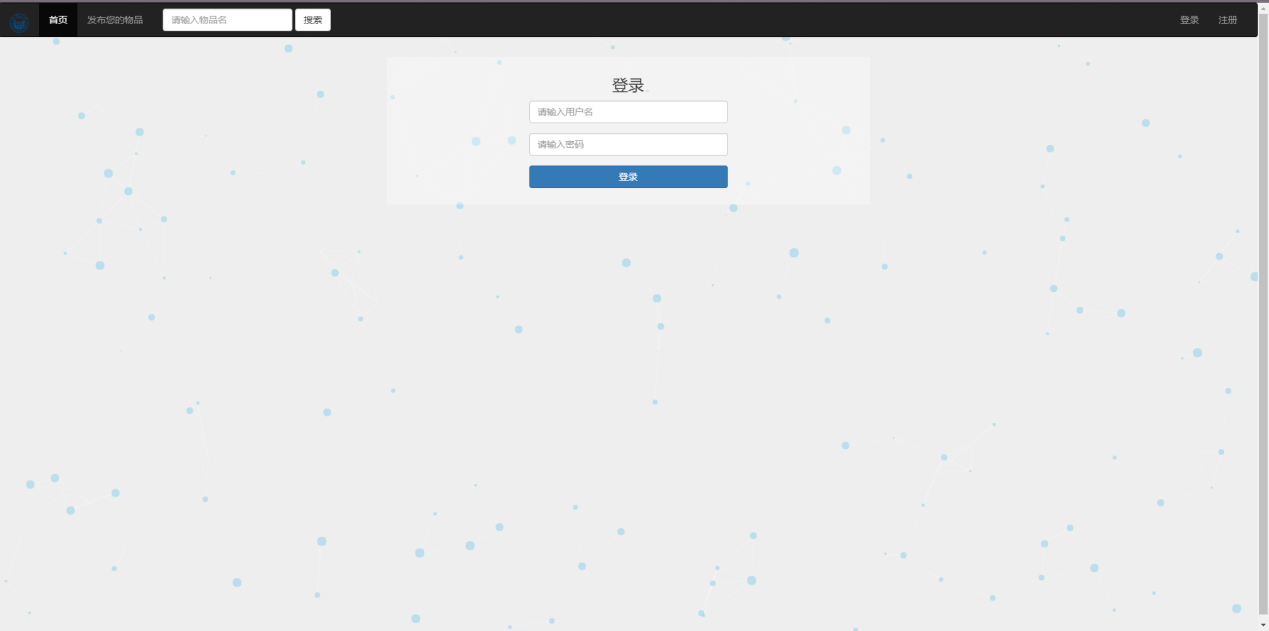


图4.1 用户登录界面

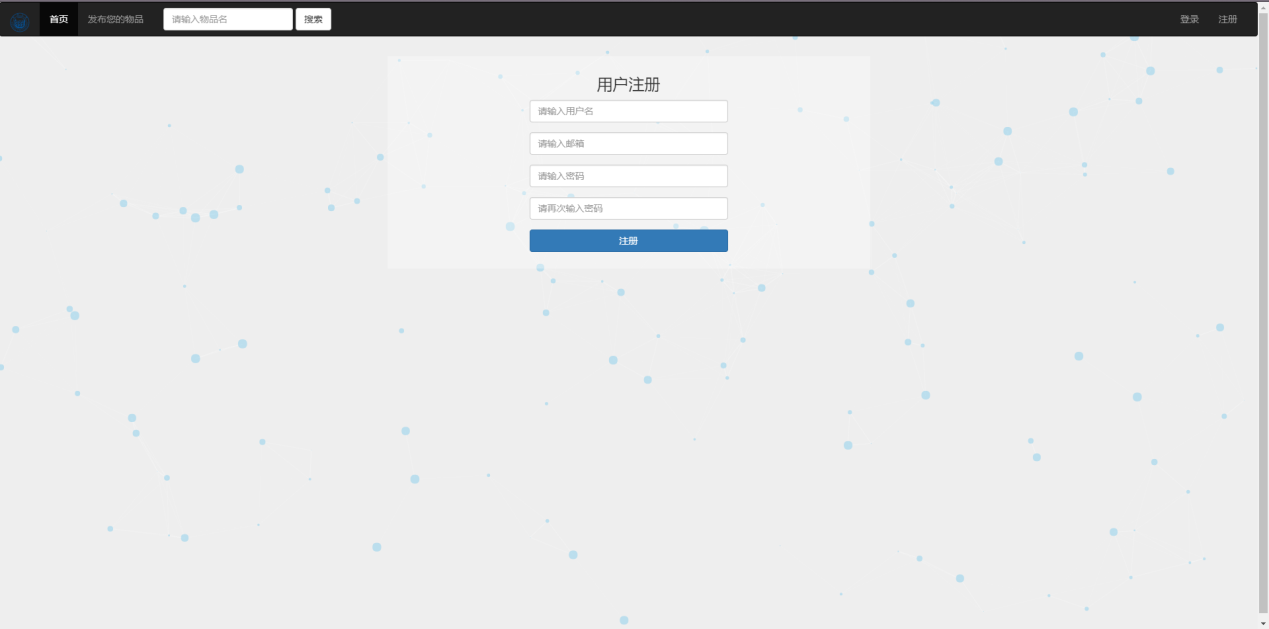


图4.2 用户注册界面



图4.3 首页



图4.4 物品详情页面

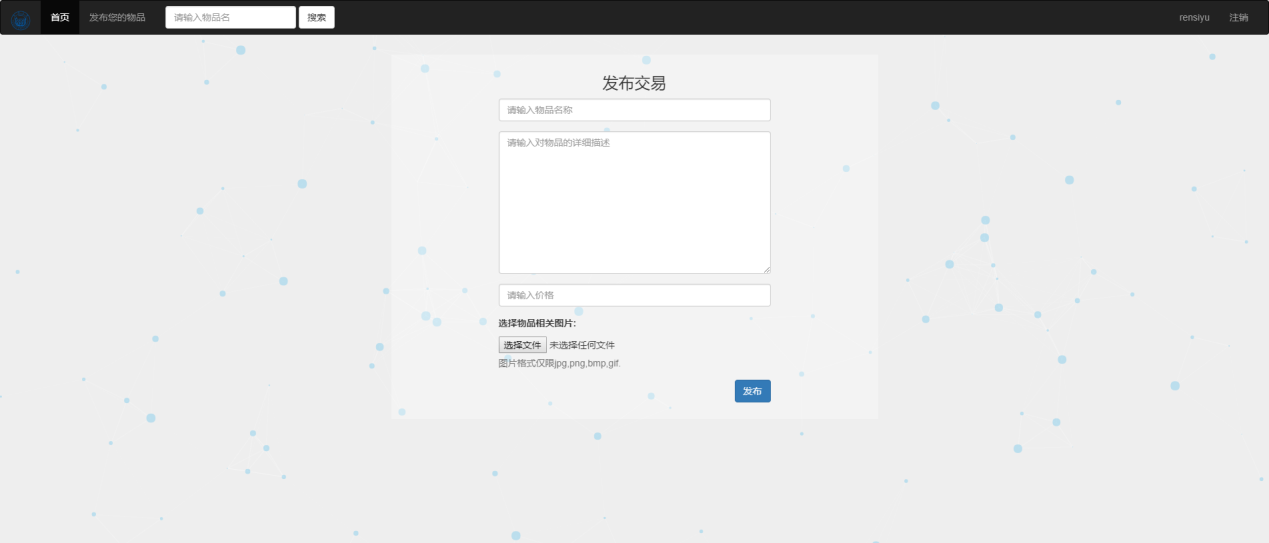


图4.5 发布物品页面

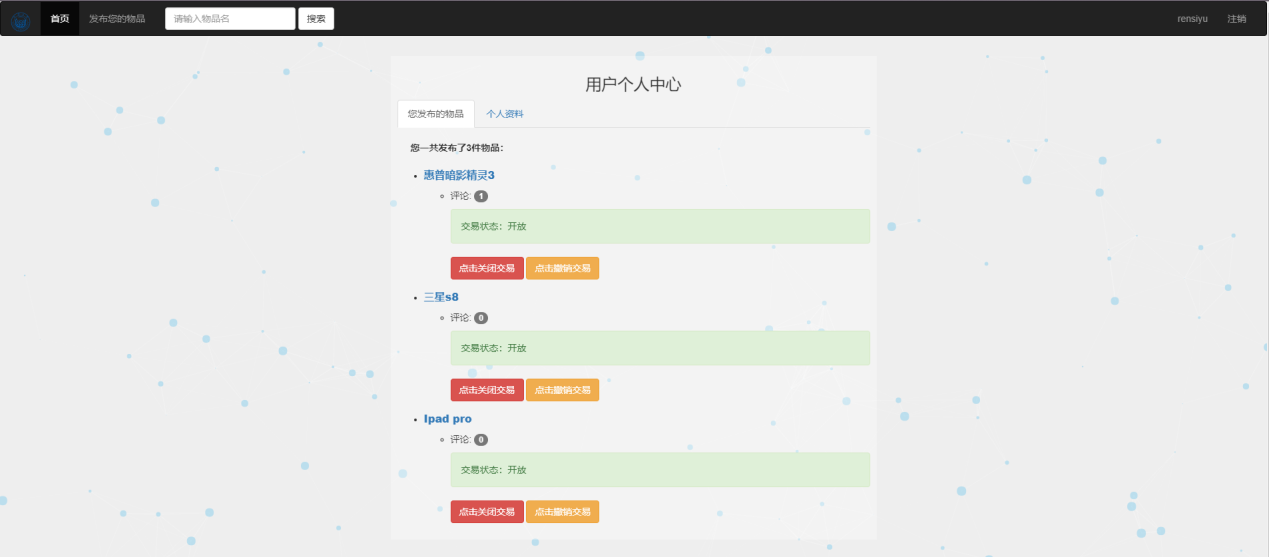


图4.6 用户个人中心页面之一



图4.7 用户个人中心页面之二

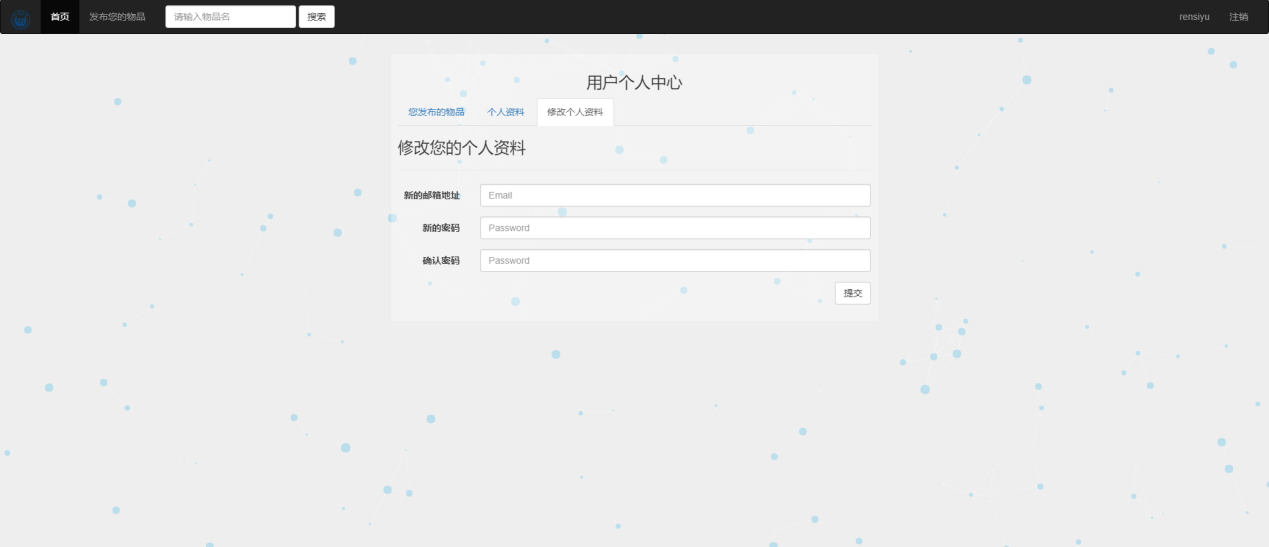


图4.8 修改个人资料

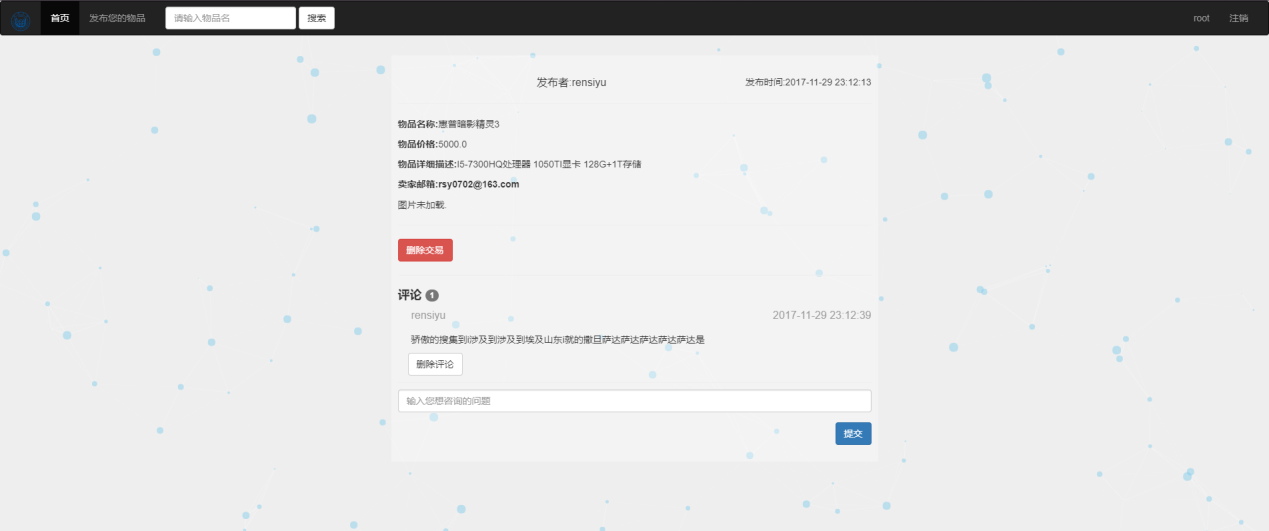


图4.8 管理员物品详情页面



图4.9 搜索物品栏



图4.10 登录用户及注销

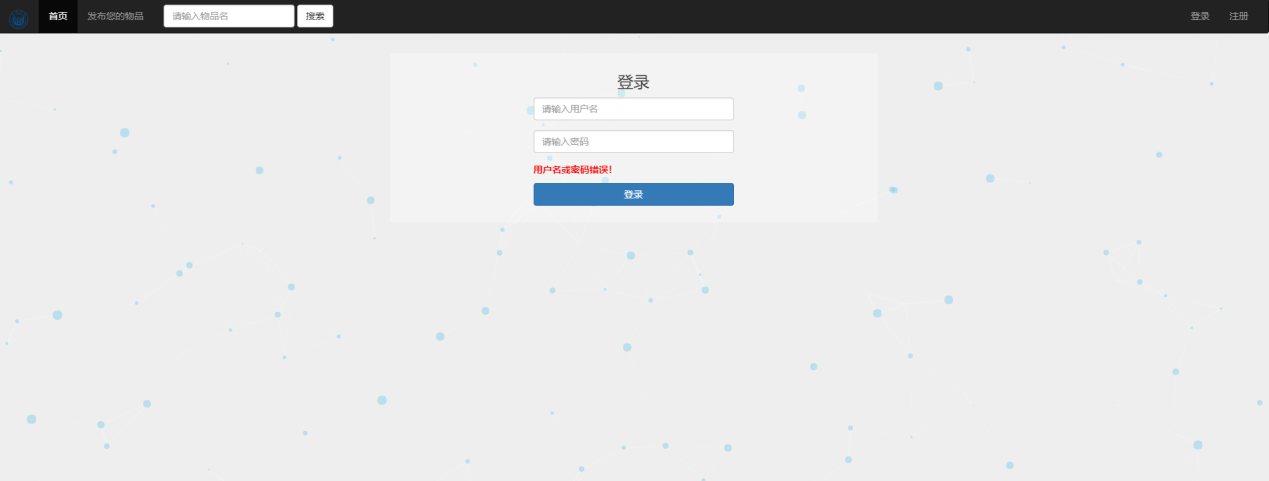


图4.11 用户登录出错

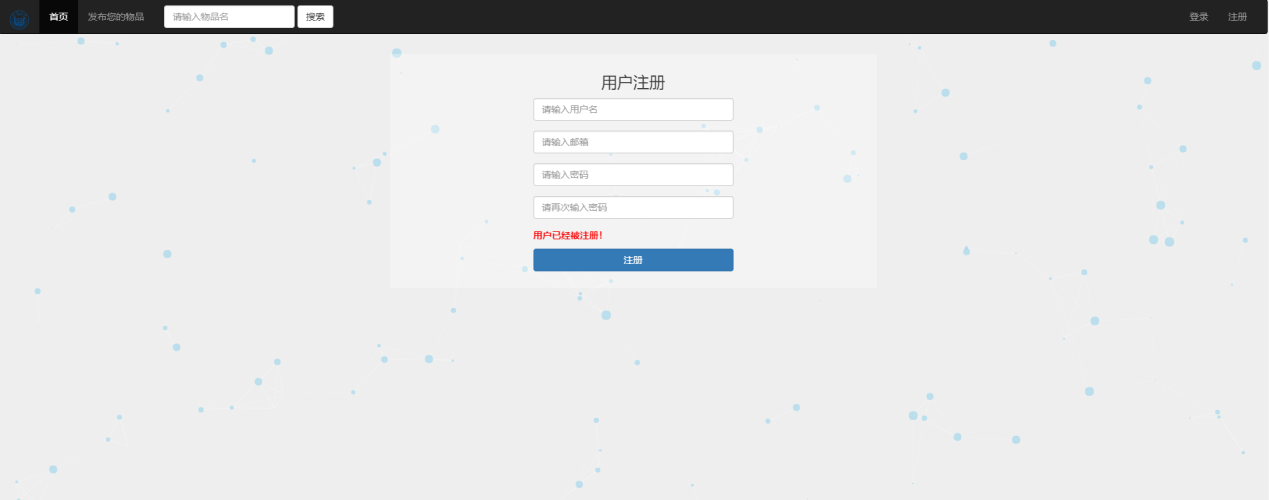
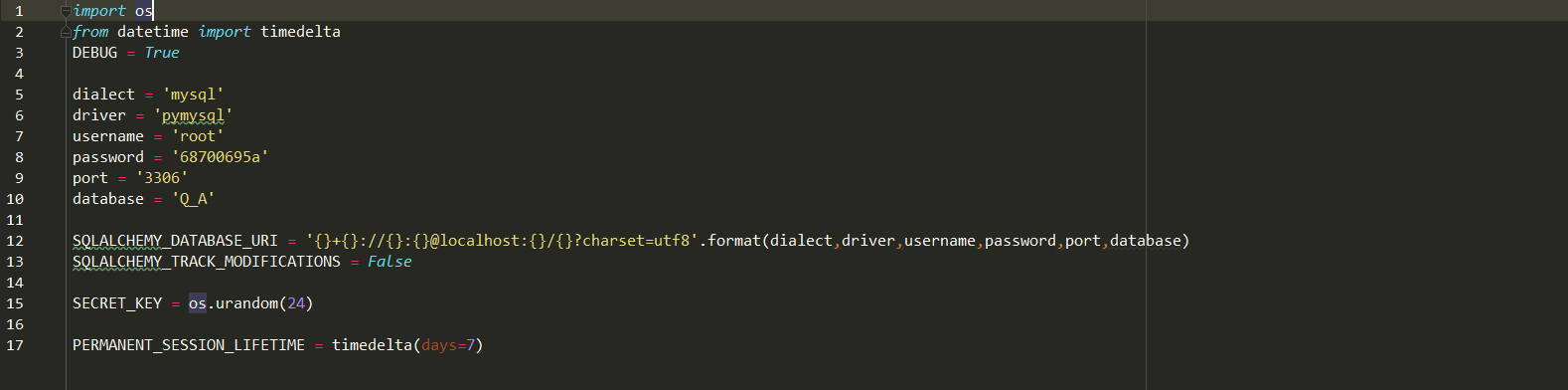


图4.12 用户注册同名账号

## 4.3 代码设计

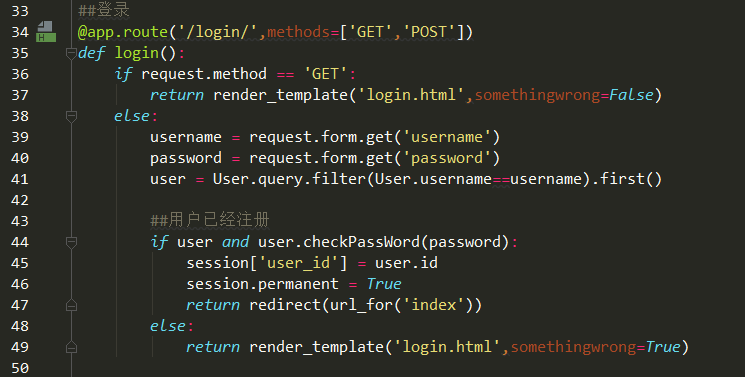
* 4.3.1 Flask项目配置

使用Flask框架的flask\_sqlalchemy插件配置连接数据库所需的参数，如使用的python数据库驱动、使用的DBMS、数据库名、端口号、用户名、密码等。除此之外，配置文件还包括用户登录之后session保存的时间长度,session加密密钥等。代码如下图所示：



* 4.3.2 用户登录、注册功能代码设计

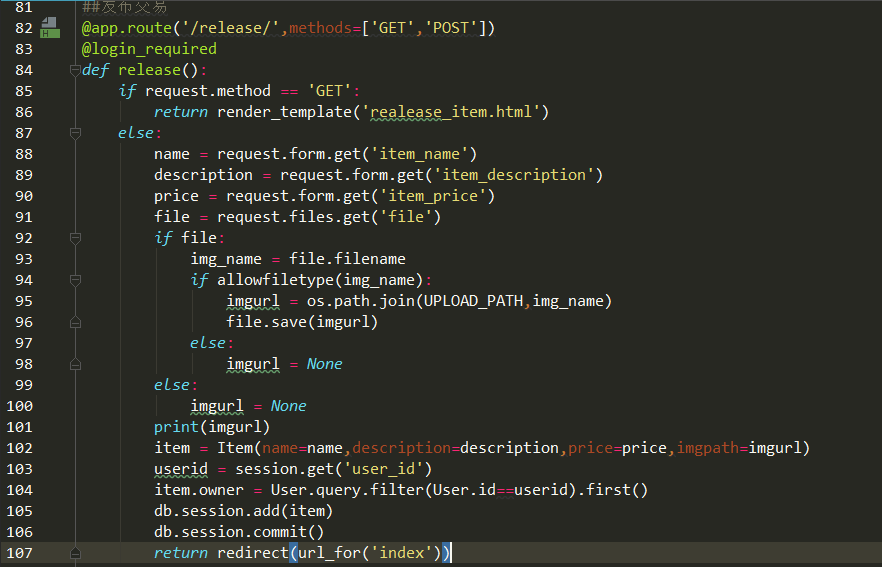
用户登录功能是通过读取用户在登录界面输入的账号和密码，在用户表中查找有无对应的项，如果有，则登录成功，跳转到首页，否则登录失败。代码如下图所示：



代码中会判断当前访问URL的形式是get还是post，如果是get则渲染让用户填写登录信息的页面。如果是post则根据用户的输入，在用户表中首先查找是否有用户名匹配的项，如果有，则使用User模型的checkPassWord函数检查用户输入的密码通过同样的加密处理后是否于数据库中的项匹配，匹配则重定向至首页的URL。

* 4.3.3 发布物品功能代码设计

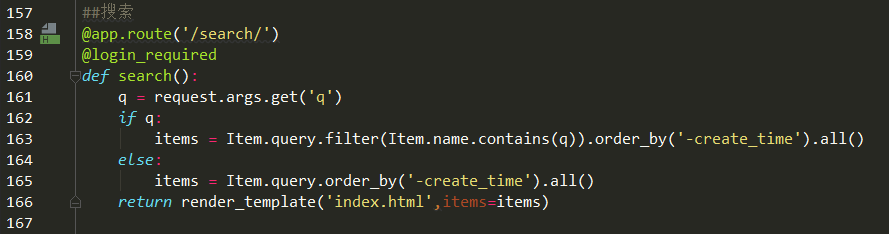
发布物品功能是通过读取用户在物品发布界面输入的物品名，物品详细描述，物品价格，以及上传的物品图片，创建一个新的物品对象，将其映射到实际的数据库中的物品表中。代码如下图所示：



图中的@login\_required是python语言中的装饰器，它的作用是根据用户是否登录而返回不同的页面。当用户已经登录时：如果当前访问URL的形式是post形式，则根据用户提供的物品相关信息，创建一个新的物品对象，映射到物品表中。如果用户没有登录，则重定向至用户登录界面。

* 4.3.4 搜索物品功能代码设计

搜索物品功能是根据用户在搜索框中输入的关键字，在物品表中查找物品名称包含该关键字的项。然后在首页中按发布时间从最近发布的到最早发布的顺序列出。代码如下图所示：



* 4.3.5 查看物品详情功能代码设计

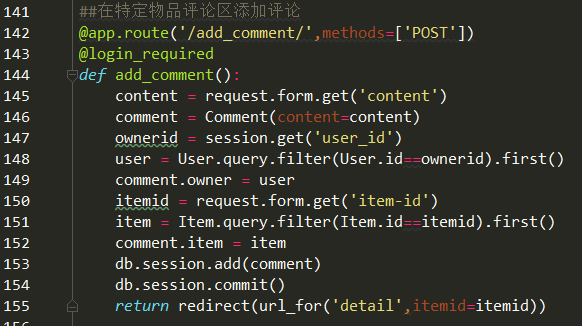
查看物品详情功能是通过点击网站首页的超链接进入的。代码如下图所示：



物品详情页面根据用户的权限而有所不同。首先根据用户登录时保存在session中的用户ID在用户表中查询，获取用户的权限。如果是管理员，则渲染出的额html页面中会有删除该物品和删除物品下方指定评论的功能按钮。如果是普通用户，则没有上述功能，只有查看物品相关信息，收藏按钮，发布评论等功能。

* 4.3.6 在特定物品发布评论功能设计

用户通过点击首页的物品名，转到物品详情页面之后，额可以在下方发布评论。然后点击提交，重新渲染物品详情页面，将新的评论映射到评论表中。代码如下图所示：



代码中的@login\_required装饰器作用跟之前出现的作用相同。代码的功能是通过读取post请求提供的评论内容创建一个新的评论对象，然后根据当前登录用户的ID和当前所评论的物品的ID建立flask中的特殊关联对象relationship连接到用户表和物品表的对应项。

* 4.3.7 个人中心功能代码设计

点击导航栏右部的用户名，可以进入用户个人中心。个人中心的功能包括：查看用户已发布的物品，并能够开启/关闭交易、永久撤销交易；查看个人资料；查看用户收藏的感兴趣的物品；修改用户的个人资料。代码设计如下图所示：



图中的代码包含了给用户个人中心的三个分页面的实现。通过给定URL中的<target>的参数不同，渲染不同的页面返回给用户。

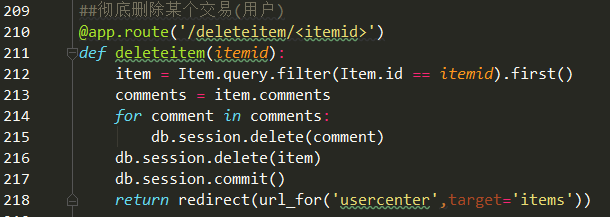
* 4.3.8 关闭/开放物品交易功能代码设计

关闭/开放某个物品交易功能的实现很简单，只需要根据物品ID在物品表中查找到对应的项，然后更改其isclosed属性即可。物品的isclosed属性更改以后，在网站首页上只会显示isclosed属性为0的那些物品。代码如下图所示：



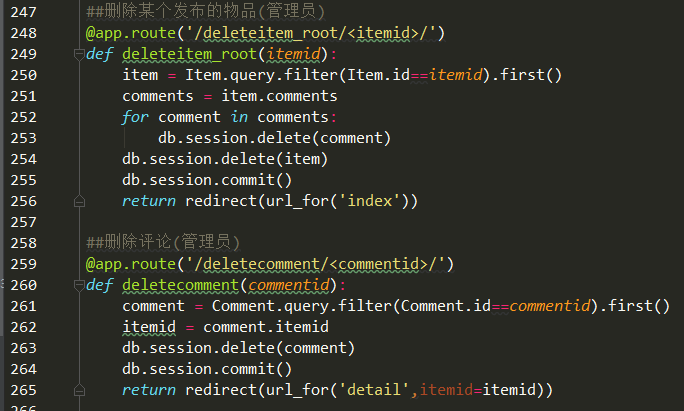
* 4.3.9 撤销物品交易功能代码设计

撤销物品交易功能的实现比单纯的关闭物品交易复杂。在删除该物品交易之前首先需要删除该物品下所有的评论，然后再删除该物品。代码如下图所示：



* 4.3.10 管理员删除二手物品、评论功能代码设计

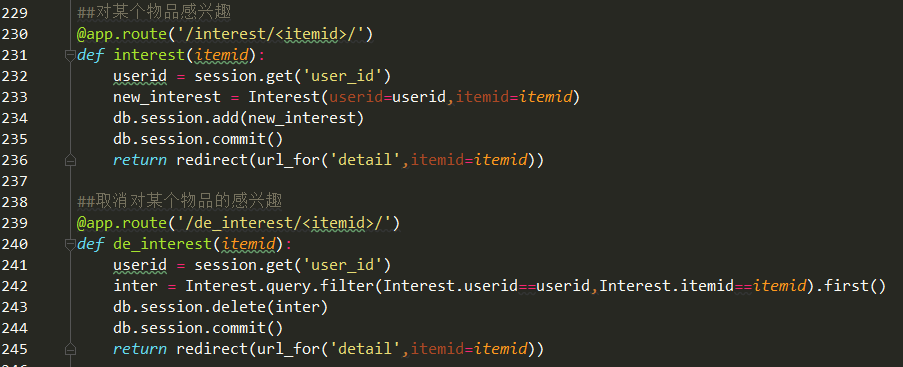
当登录用户具有管理员管理员时，在点击进入物品详情页面之后，可以选择删除当前的物品，也可以选择删除指定的评论。代码如下图所示：



两个函数分别对应删除物品和删除评论。在删除物品函数中，首先进行的是删除该物品下的所有评论，然后再删除物品，返回网站首页。在删除指定评论的函数中，根据评论ID删除评论表中的相应项即可，返回当前页面。

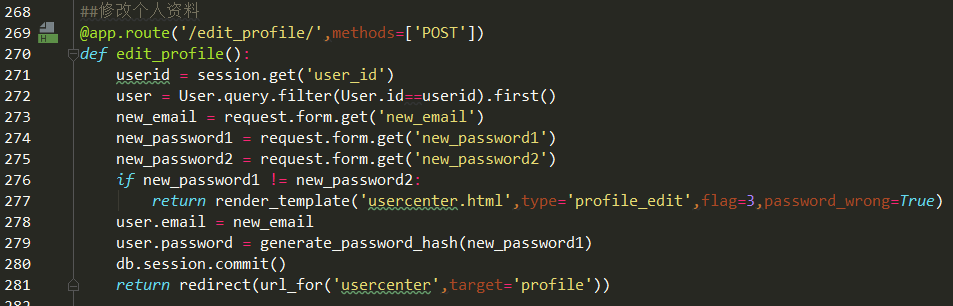
* 4.3.11 用户收藏感兴趣物品功能代码设计

用户登录以后，点击进入物品详情页面之后如果感兴趣可以点击感兴趣按钮收藏该物品。收藏之后，可以在用户个人中心中查看自己的收藏列表。如果对该物品不再感兴趣，点击不感兴趣按钮即可。代码如下图所示：



* 4.3.12 用户修改个人资料功能代码设计

用户可以在个人中心修改自己的资料，比如修改密码和邮箱地址。一次修改过程必须填写密码和邮箱，如果某项不需要更改，填写原来的内容即可。代码如下图所示：



# 5、实现要点难点

## 5.1前端页面设计

本系统的Web前端页面的所有界面都有相同的部分，如页面顶部的导航栏。



大致上系统的的每一个功能模块都有对应着一个html页面，如果所有的html页面都要手动加上导航栏的相关代码，则实现起来将会十分繁杂、麻烦。为了解决这个共享的“模板”问题，采用的技术是flask框架的jinja2引擎所提供的父模板抽离功能。将需要共享的html代码段放在一个html文件中作为父模板，然后在需要使用模板的页面的html文件中使用{%extends ‘base.html’%}的形式继承父模板即可。父模板中可以用{%block xxx%}{%endblock%}的形式来提供给子页面来填充不同的内容，在子页面中在{%block xxx%}{%endblock%}中填入需要的html代码即可。

在页面中需要根据从数据库中查询到的数据进行内容填充的部分的实现是通过flask框架的Jinja2引擎中的变量、条件语句、循环语句、过滤器等功能来实现的。在Jinja2中，变量通过下图的形式引入：



图5.1

条件语句通过下图的形式引入：



图5.2



图5.3

循环语句通过下图的形式引入：



图5.4



图5.5

有了Jinja2引擎的丰富功能，就可以在视图函数中根据传递的参数的不同渲染不同的页面。

## 5.2 python语言与数据库的交互

在DBMS中使用SQL语句与数据库进行交互是自然而又方便的。但是在使用python进行实现的时候如何与数据库(mysql)进行交互就是一个必须解决的问题。第一种解决方案是使用python的mysql驱动，然后创建游标，将sql语句放在python的字符串中，然后执行sql语句。这种方法可以完成Python与数据库的交互，但是太过底层。Flask框架提供了一套ORM框架可以让我们方便地进行与数据库的交互。ORM，全称是Object Relational Mapping，即对象关系映射。对象关系映射是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换。从效果上来说，它其实是创建了一个可在编程语言里使用的“虚拟对象数据库”。

在Flask框架中，通过使用flask\_sqlalchemy插件，让python的class机继承自SqlAlchemy()对象的Model,然后在class的定义中创建一系列类变量和数据库中的列一一对应。如下图所示：

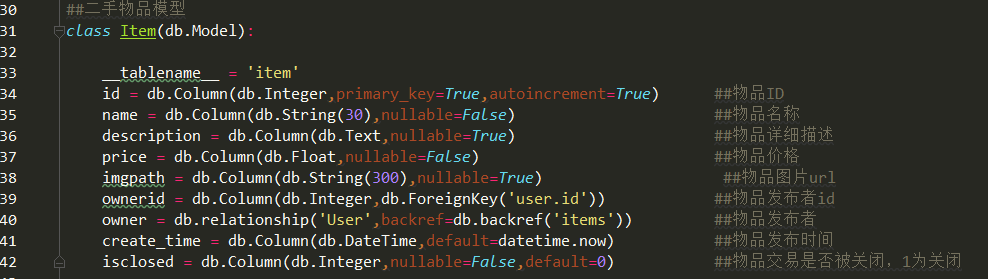


图5.6

Flask\_sqlalchemy插件中包含有sql中各种数据类型对应的对象，以及一系列约束条件，如外键约束等等。除了这些基本的属性对应之外，flask\_alchemy还提供了十分方便的relation关系，可以快速实现一对多模型的建立。

模型和表的映射建立完成之后，需要考虑的是如何将对模型对象的增、删、查、改等操作映射到数据库中。Flask\_sqlalchemy插件也提供了这些操作的python实现。增加数据和删除数据是通过db.add函数和db.delete函数实现，查询是通过继承自db.Model的类对象的filter函数进行筛选的，修改数据是通过直接在筛选得到的类对象修改相应属性实现的。代码如下图所示：

添加：



图5.7

删除：



图5.8

筛选：



图5.9

修改：



图5.10

所有的操作需要进行事务提交才能映射到数据库中生效：



图5.11

# 6、总结

设计一个二手物品发布平台需要进行需求分析，可行性分析等一系列研究。分析用户在使用我的系统时需要哪些功能，怎样设计WEB页面能够让用户简单快捷地进行想要的操作。在数据库中建立所需要的数据表，并且最大限度地保证没有冗余，且查询等操作的速度较快。有一点经验是在编写代码的过程中需求可能是变化的，比如某个时刻可能需要给一个表添加新的属性，那么通过flask\_script和flask\_manage插件可以完成数据库的migrate和upgrade，更新实际表的内容。

系统实现的功能包括了基本需求，注册、登录、发布、评论、个人中心等等。但没有实现类似淘宝、京东那样的购物网站一样的交易功能，本系统的交易功能是以提供卖家的联系方式让用户和卖家通过第三方渠道交易实现的，相当于提供了一个类似预定的功能。