

**基于WEB的程序设计与实践**

**实 验 报 告**

|  |  |
| --- | --- |
| 题目 | \*\*\*\*\*\*\*\* |
| 系(部) | 信息与计算科学系 |
| 姓名 | \*\*\*\* |
| 学号 | B1\*\*\*\*\* |
| 指导教师 | 宫 婧 |
| 起止日期 | 2020.9 — 2020.12 |

**目录**

1. 需求分析
2. 总体设计
3. 数据库设计
4. 详细设计
5. 系统测试
6. 总结与展望

**第一章 需求分析**

**第二章 总体设计**

网站结构图：

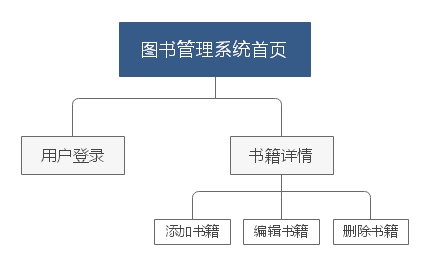
****

图2.1 – 网站结构图

进入图书管理系统，首页展示的是书籍的信息，包括书籍的id、书名、价格、作者、出版社、出版日期、分类等。并且还有添加书籍、修改书籍、删除书籍、登录的功能。在没有登录的情况下，无法进行添加、修改、删除等操作，而是会自动跳转到登录页面，输入指定的管理员账户和密码才能登陆。登录成功后会跳转回首页，此时可以进行添加、修改、删除操作。

点击添加按钮，跳转到添加页面，输入书名、价格、作者、出版社、出版日期、分类，点击确定就可以添加书籍了，回到首页就能看到新添加的书籍信息。

点击修改按钮，跳转到修改页面，输入框默认是原有的书籍信息，修改对应的字段后点击确定，回到首页就能看到书籍信息修改成功。

点击删除按钮，对应的书籍就会被删除，网页会自动刷新。

环境：django 2.2.13 + MySQL

语言：python

IDE：pycharm

**第三章 数据库设计**

此次设计选择的数据库系统是MySQL。

数据库一共包含4个表：books、publihers、sort、user

Books表里保存的是书籍的信息，包含id, name, price, author, pub\_date, publisher\_id, sort\_id 7个字段，其中publisher\_id和sort\_id是外键，分别参照表publishers和表sort。books表结构如下图：

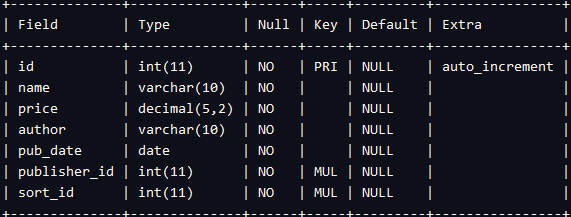


图3.1 – books表结构

Publishers表里保存的是出版社的信息，包含id, name, city三个字段，publishers表结构入下图：

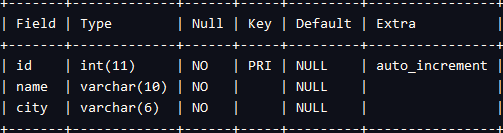


图3.2 – publishers表结构

sort表里保存的是分类的信息，包含id, name三个字段，sort表结构入下图：

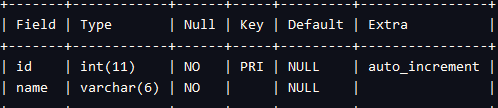


图3.3 – sort表结构

user表里保存的是管理员用户的信息，包含id, username, password三个字段，user表结构如下图：

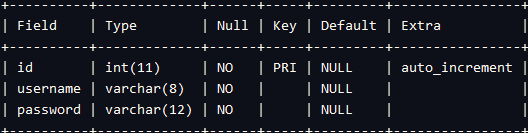
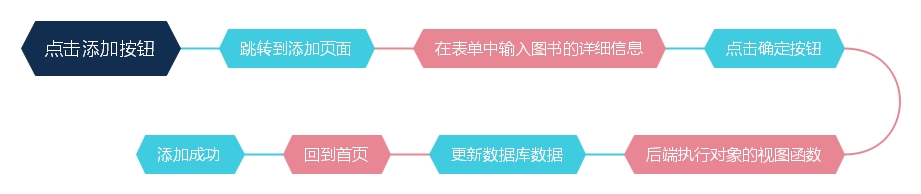


图3.4 – user表结构

**第四章 详细设计**

1.添加

添加图书的流程图如下：



4.1-添加图书流程图

点击添加图书按钮，进入到添加页面，填入书名，价格，作者，出版日期，选择对应的出版社，分类，点击确定，后端会调用相应的视图函数，数据就会提交到数据库，数据库更新完成，页面跳转回首页，并展示新的数据，此时可以看到添加成功的图书。

视图函数代码如下：

@required\_login

def save\_add(request):

name = request.POST['name']

price = request.POST['price']

author = request.POST['author']

publisher = request.POST['publisher']

pubdate = request.POST['pubdate']

sort = request.POST['sort']

publisher\_id = Publisher.objects.get(name=publisher).id

sort\_id = Sort.objects.get(name=sort).id

conn = pymysql.connect(host='121.199.28.54', port=3306, user='root', passwd='lyk0915',

db='book\_manage\_sys', charset='utf8')

cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

sql = "insert into books(name, price, author, publisher\_id, pub\_date, sort\_id) values ('{0}', {1}, '{2}', {3}, '{4}', {5});".format(name, price, author, publisher\_id, pubdate, sort\_id)

cursor.execute(sql)

conn.commit()

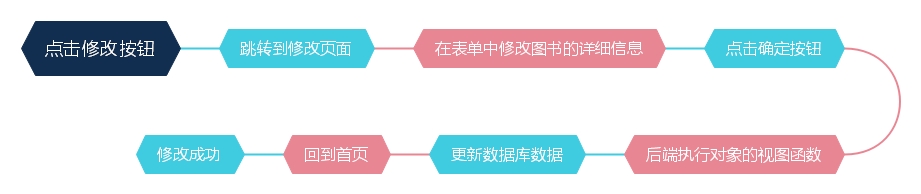
cursor.close()

conn.close()

return redirect('/')

2.修改

修改图书的流程图如下：



4.2-编辑图书流程图

点击编辑图书按钮，进入到编辑页面，此时表单中显示的是原有的图书信息，编辑新的书名，价格，作者，出版日期，选择对应的出版社，分类，点击确定，后端会调用相应的视图函数，数据就会提交到数据库，页面刷新，获取到新的数据，此时可以看到修改以后的图书信息。

视图函数如下：

@required\_login

def save\_edit(request, edit\_id):

name = request.POST['name']

price = request.POST['price']

author = request.POST['author']

publisher = request.POST['publisher']

pubdate = request.POST['pubdate']

sort = request.POST['sort']

publisher\_id = Publisher.objects.get(name=publisher).id

sort\_id = Sort.objects.get(name=sort).id

conn = pymysql.connect(host='121.199.28.54', port=3306, user='root', passwd='lyk0915',

db='book\_manage\_sys', charset='utf8')

cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)

sql = "update books set name='{0}', price={1}, author='{2}', publisher\_id={3}, pub\_date='{4}', sort\_id={5} " \

"where id={6};".format(name,price,author,publisher\_id,pubdate,sort\_id,edit\_id)

cursor.execute(sql)

conn.commit()

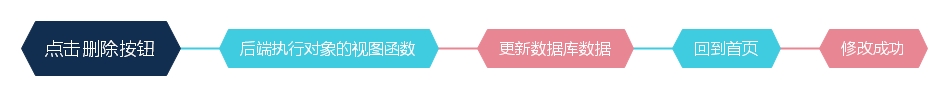
cursor.close()

conn.close()

return redirect('/')

3.删除

删除图书的流程图如下：



4.3-删除图书流程图

点击删除按钮，后端会调用相应的视图函数，将数据库中对应的数据删除，接着刷新页面，可以看到首页被删除的图书不再显示了。

视图函数如下：

@required\_login

def del\_book(request, del\_id):

b = Book.objects.get(id=del\_id)

print('bbb', b)

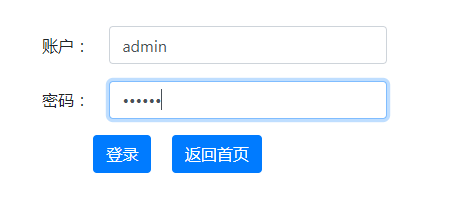
b.delete()

return HttpResponseRedirect('/')

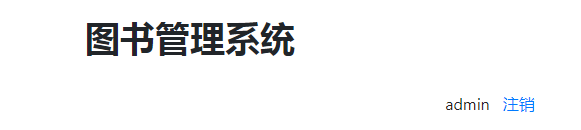
**第五章 功能测试**

1.登录模块

点击登录按钮，输入用户名和密码，点击确定



登录成功之后，自动返回首页，右上角会显示登陆成功的用户的用户名，



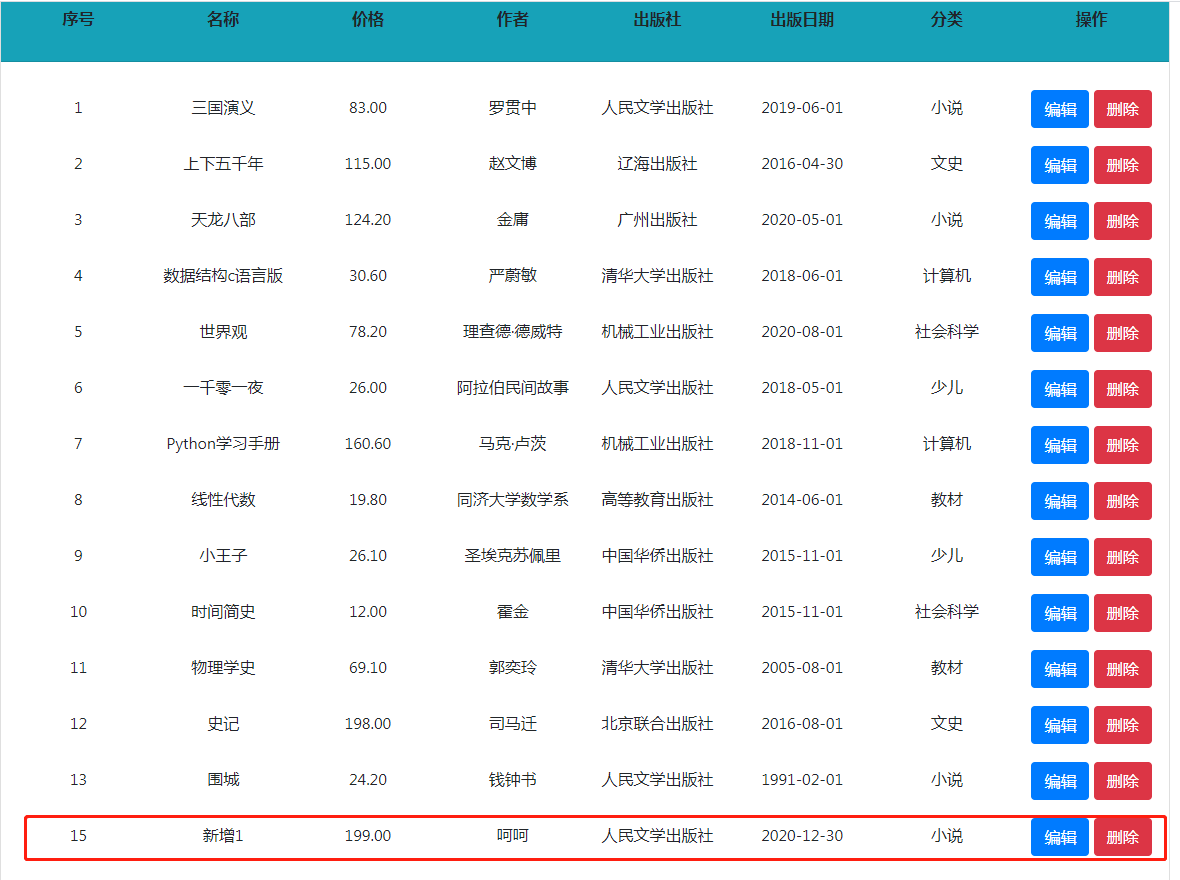
点击注销按钮，登录的用户就会注销。

2.添加模块

点击添加模块，输入要添加的图书信息，点击确定，



添加成功后自动跳转回首页，并且可以在底部看到刚才添加的图书，

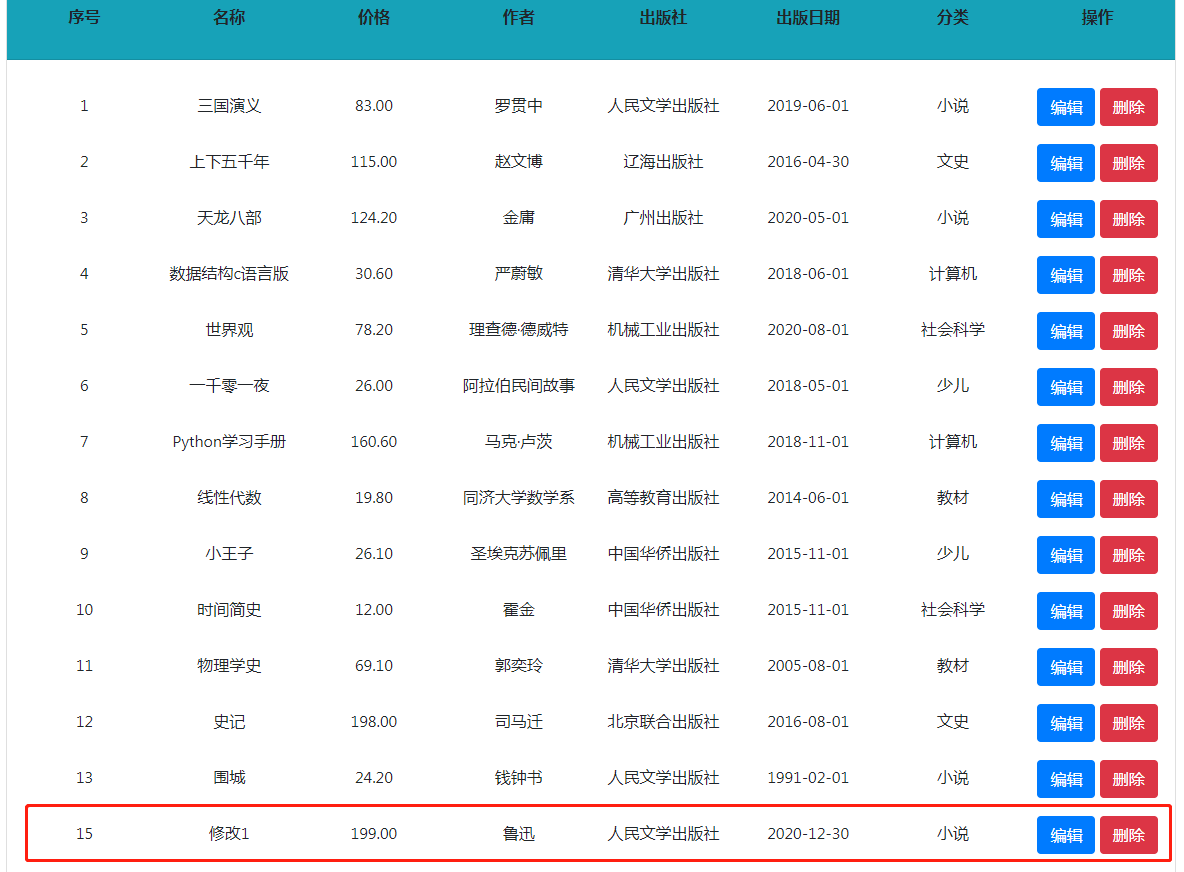


3.编辑模块

点击编辑按钮，编辑刚才新增的图书，修改书名和作者，



点击确定按钮后，自动跳转回首页，可以看到修改后的图书信息。



4.删除模块

点击删除按钮，成功将刚才添加的书籍删除。

