目录

[一、用户管理模块 1](#_Toc125012303)

[1、安装和运行 1](#_Toc125012304)

[2、用户注册模块 2](#_Toc125012305)

[3、用户登录模块 5](#_Toc125012306)

[4、重置密码模块 7](#_Toc125012307)

[5、修改用户模块 9](#_Toc125012308)

[二、业务管理模块 11](#_Toc125012309)

[1、录入图书模块 11](#_Toc125012310)

[2、查询图书模块 15](#_Toc125012311)

[3、修改图书模块 16](#_Toc125012312)

[4、删除图书模块 17](#_Toc125012313)

# 一、用户管理模块

## 1、安装和运行

（1）安装

执行安装命令，

|  |
| --- |
| (p39) D:\> pip install django |

安装完毕后，可执行命令django-admin。

解压www\_book.zip到D:\www\_book

（2）启动服务器

|  |
| --- |
| (p39) D:\www\_book>py manage.py runserver |

（3）打开网页

在浏览器地址栏中输入<http://127.0.0.1:8000/>，打开如下【登录页面】。



图1 登录页面

用户名和密码都是1，登录后系统主页如下图所示：

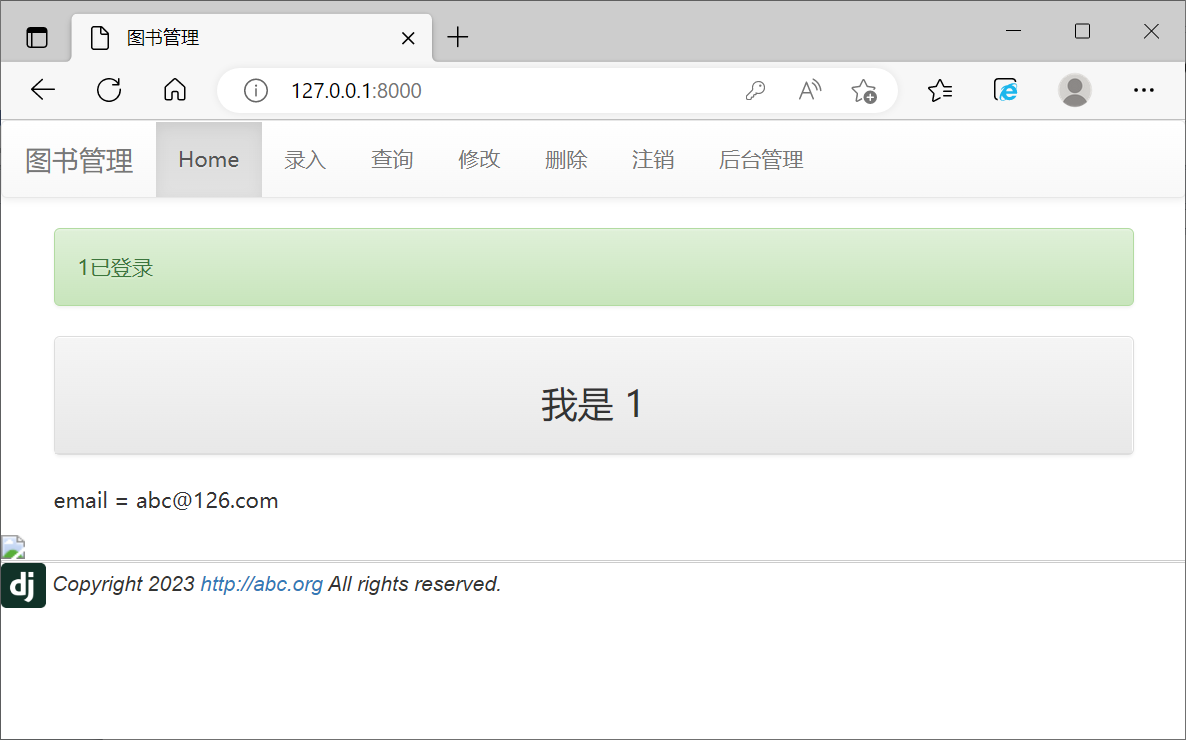


图2 登录成功页面

## 2、用户注册模块

本节说明用户注册模块的操作步骤和程序运行流程。注册模块有三个基本操作步骤：邮箱配置，用户注册与用户激活。

（1）邮箱配置，目的是允许程序发送电子邮件完成激活功能。在126.com邮箱中，单击主菜单项【配置】，单击菜单项【POP3/SMTP/IMAP】，系统打开如下设置页面。在页面中开启IMAP/SMTP服务，复制并保存系统显示的密码。

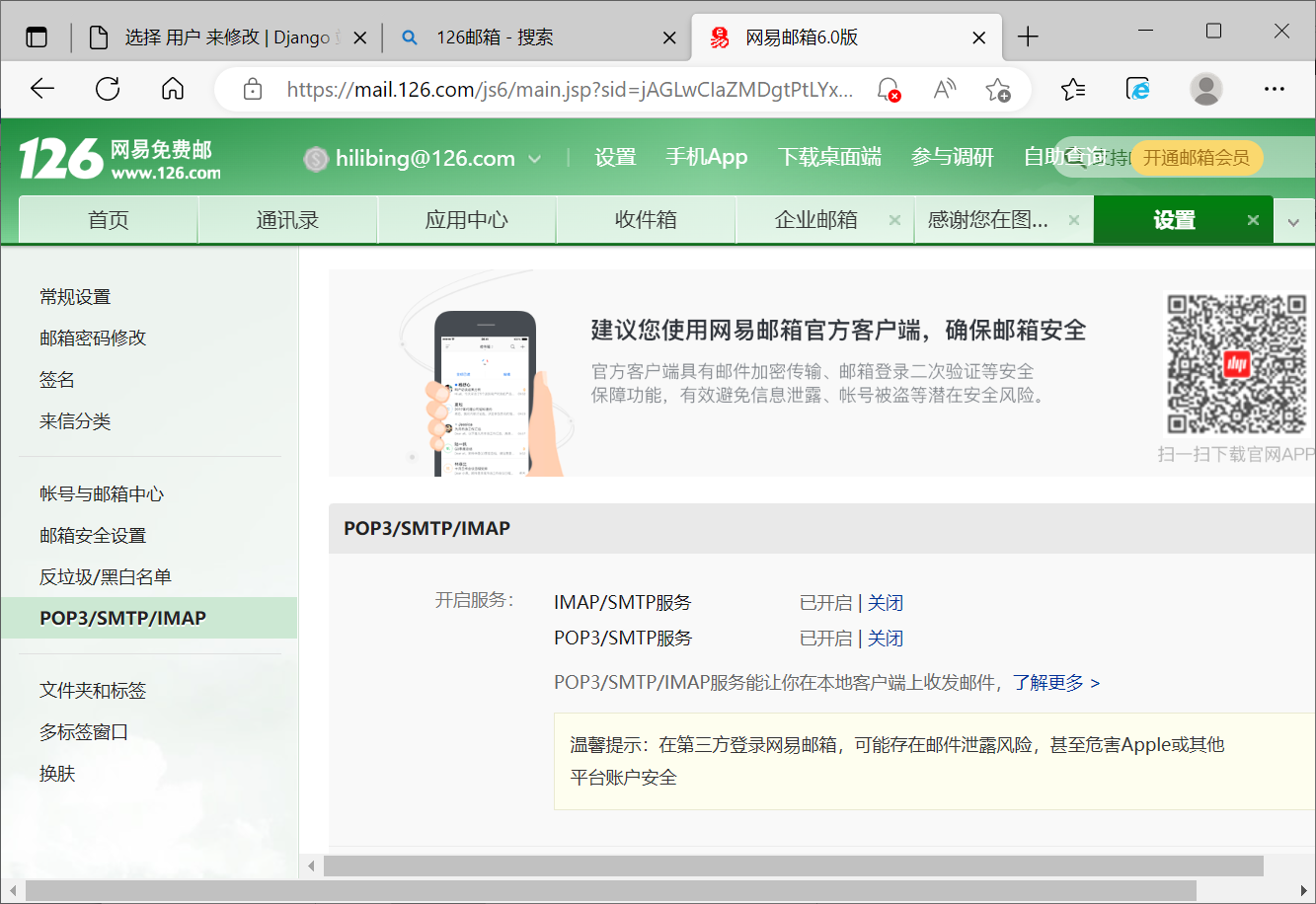


图3 126邮箱设置页面

（2）用户注册。单击【注册】菜单项，打开注册页面，如图4左所示。

图4 用户注册及结果页面

图4右图为手机登录页面，可用于测试响应式网页设计效果。在setting.py中配置ALLOWED\_HOSTS = ["\*",]，执行ipconfi查看手机访问地址，

|  |
| --- |
| C:\>ipconfig  无线局域网适配器 WLAN 2:  IPv4 地址 . . . . . . . . . . . . : 192.168.2.6 |

启动服务器命令为

|  |
| --- |
| (p39) D:\www\_book>py manage.py runserver 0.0.0.0:8000 |

在手机浏览器中输入网址192.168.2.6:8000显示图4（右）的手机登录页面。

输入各项数据时要满足页面提示的要求，E-mail需填写自己的126电子邮箱，最后单击【注册】按钮，结果如图5所示。

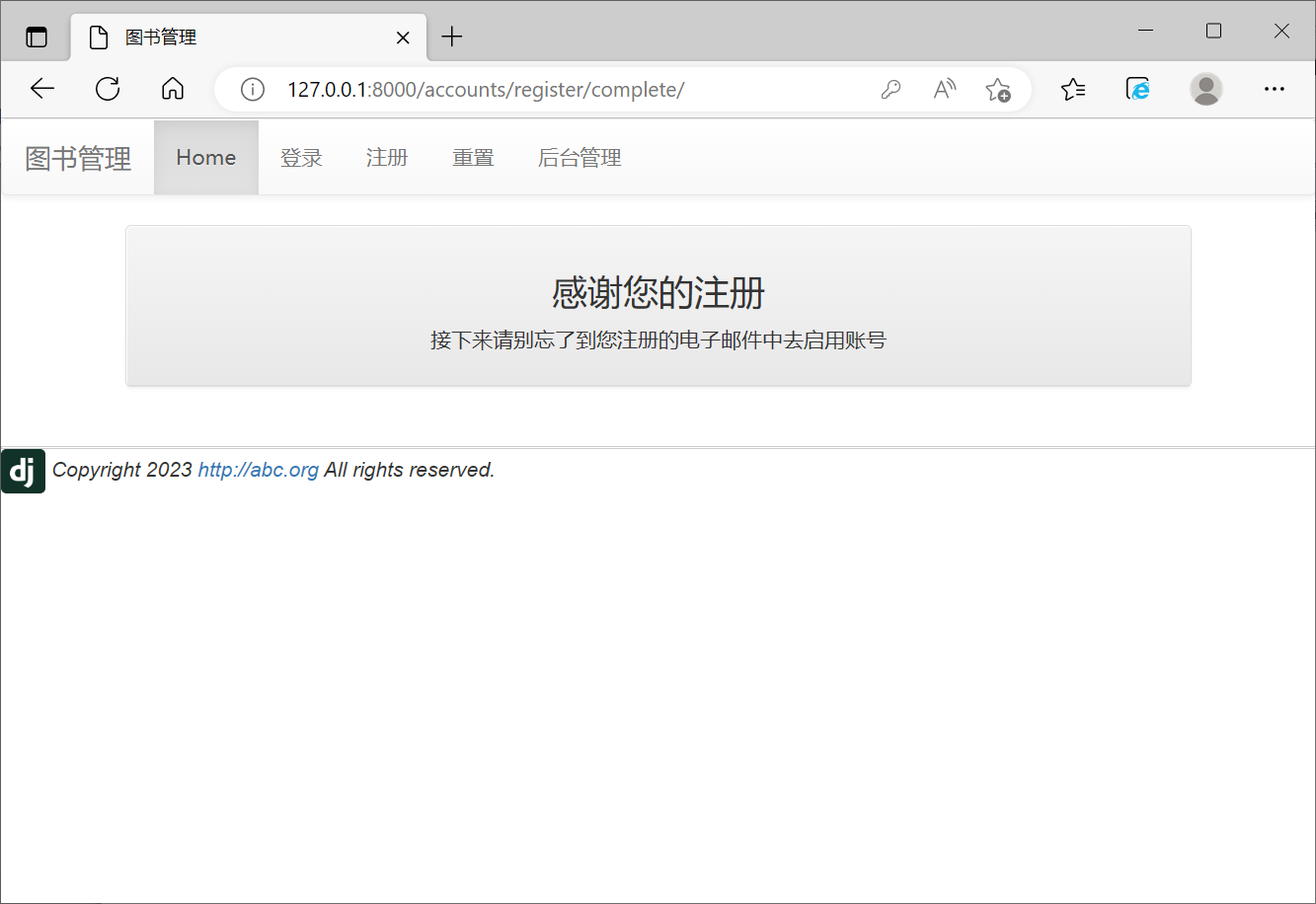


图5 用户注册及结果页面

（3）用户激活。在用户注册的邮箱中打开激活邮件，如图6所示。

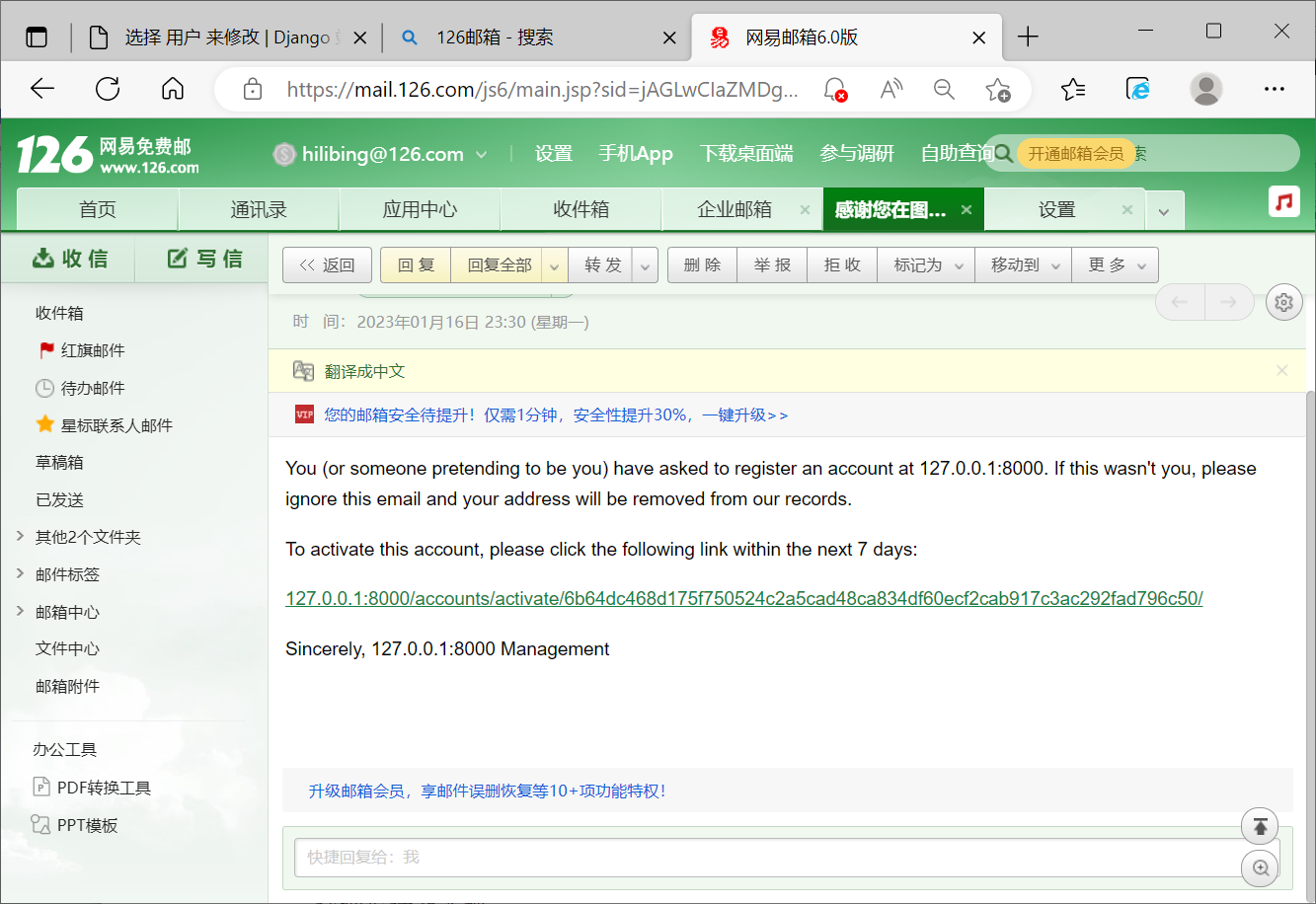


图6 126邮箱激活邮件

单击邮件中激活超链，系统显示激活结果，如图7所示。激活后的用户可执行登录功能，如果试图登录未激活的用户，系统将提示“登录失败”。

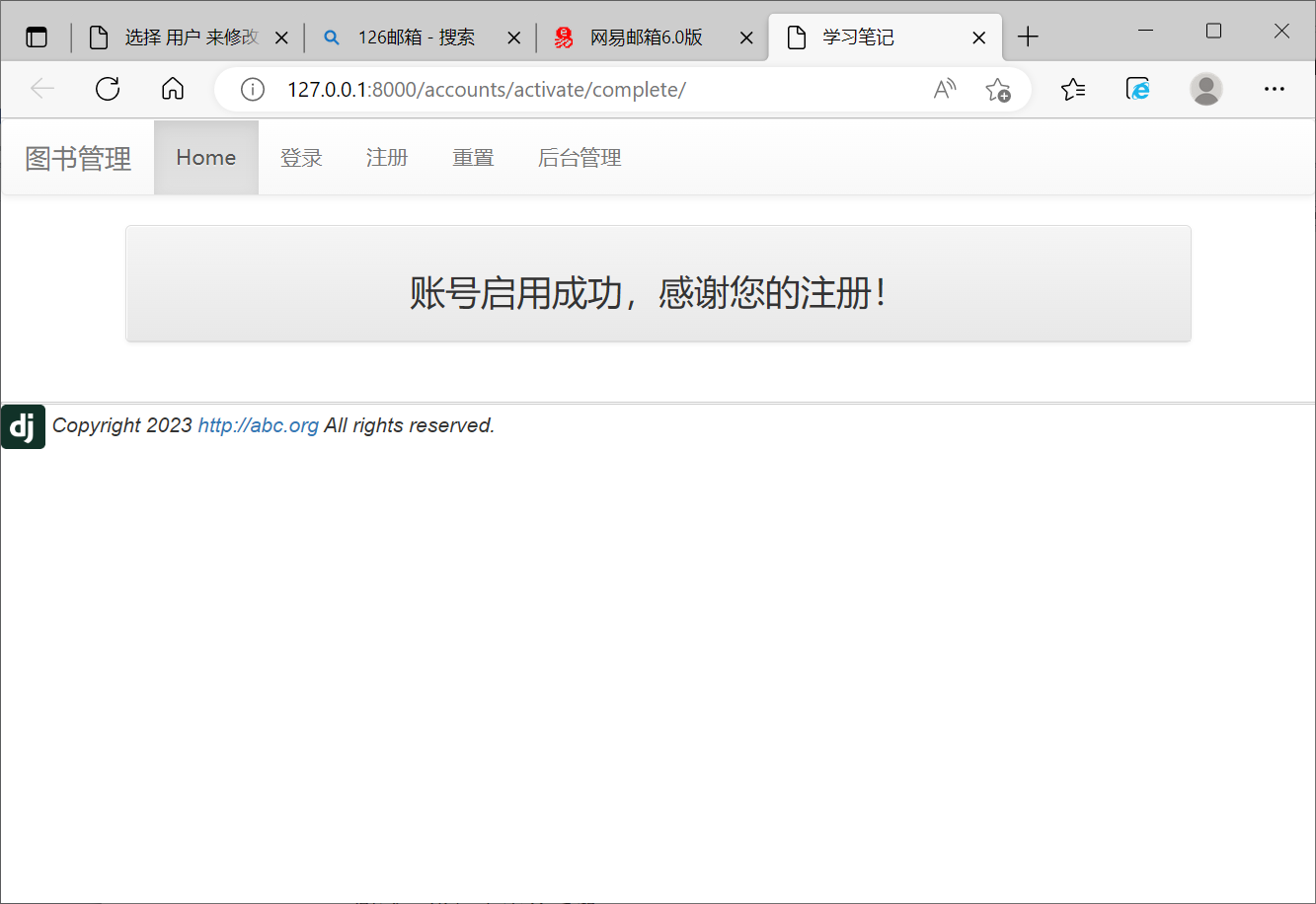


图7 邮件激活成功

如果激活操作失败（如用户已被删除），则系统显示失败页面，如图8所示。

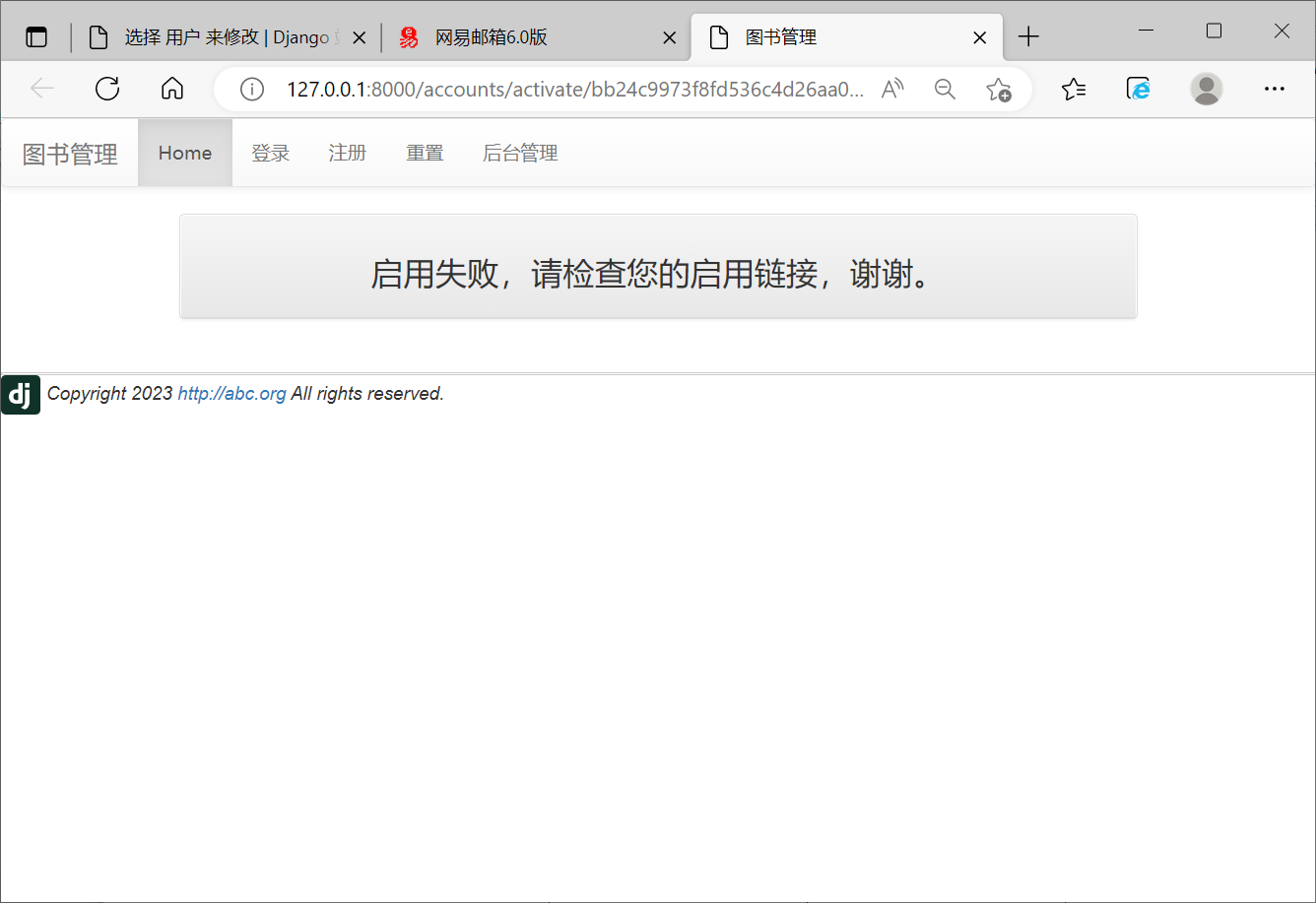


图8 邮件激活失败

注册模块需要在配置文件settings.py中配置发送邮件的邮箱信息，如下代码所示，

|  |
| --- |
| EMAIL\_HOST = 'smtp.126.com'  EMAIL\_PORT = 25  EMAIL\_HOST\_USER = 'abcd@126.com'  EMAIL\_HOST\_PASSWORD = 'abcd@126.com的密码'  EMAIL\_USER\_TLS = True  EMAIL\_USER\_SSL = True  DEFAULT\_FROM\_EMAIL = EMAIL\_HOST\_USER  ACCOUNT\_ACTIVATION\_DAYS = 7 |

其中的EMAIL\_HOST\_USER和EMAIL\_HOST\_PASSWORD为演示数据，需要依据自己邮箱填写真实数据。

注册模块程序工作流程：

（1）浏览器发送请求GET /accounts/register/，服务器返回registration\_form.html；

（2）用户输入数据后单击提交时，浏览器发送请求POST /accounts/register/，服务器返回registration\_complete.html，并发向注册邮箱送激活邮件，邮件主题位于activation\_email\_subject.txt文件中；

（3）用户点击邮件中的超链执行激活操作，如果激活成功，服务器返回activation\_complete.html，否则返回activation\_complete.html。

上述流程中的模板文件位于www\_book\templates\registration目录中。用户激活后，该用户的is\_active字段为True，并且可以执行登录操作。如果激活操作时用户已被删除，将出现激活失败的结果。

## 3、用户登录模块

用户在登录页面中输入名称和密码，如图1所示。由系统验证名称和密码的正确性。如果输入正确，系统将打开功能页面，如图9所示，如果输入错误，系统返回登录页面并提示“登录失败”，如图10所示。

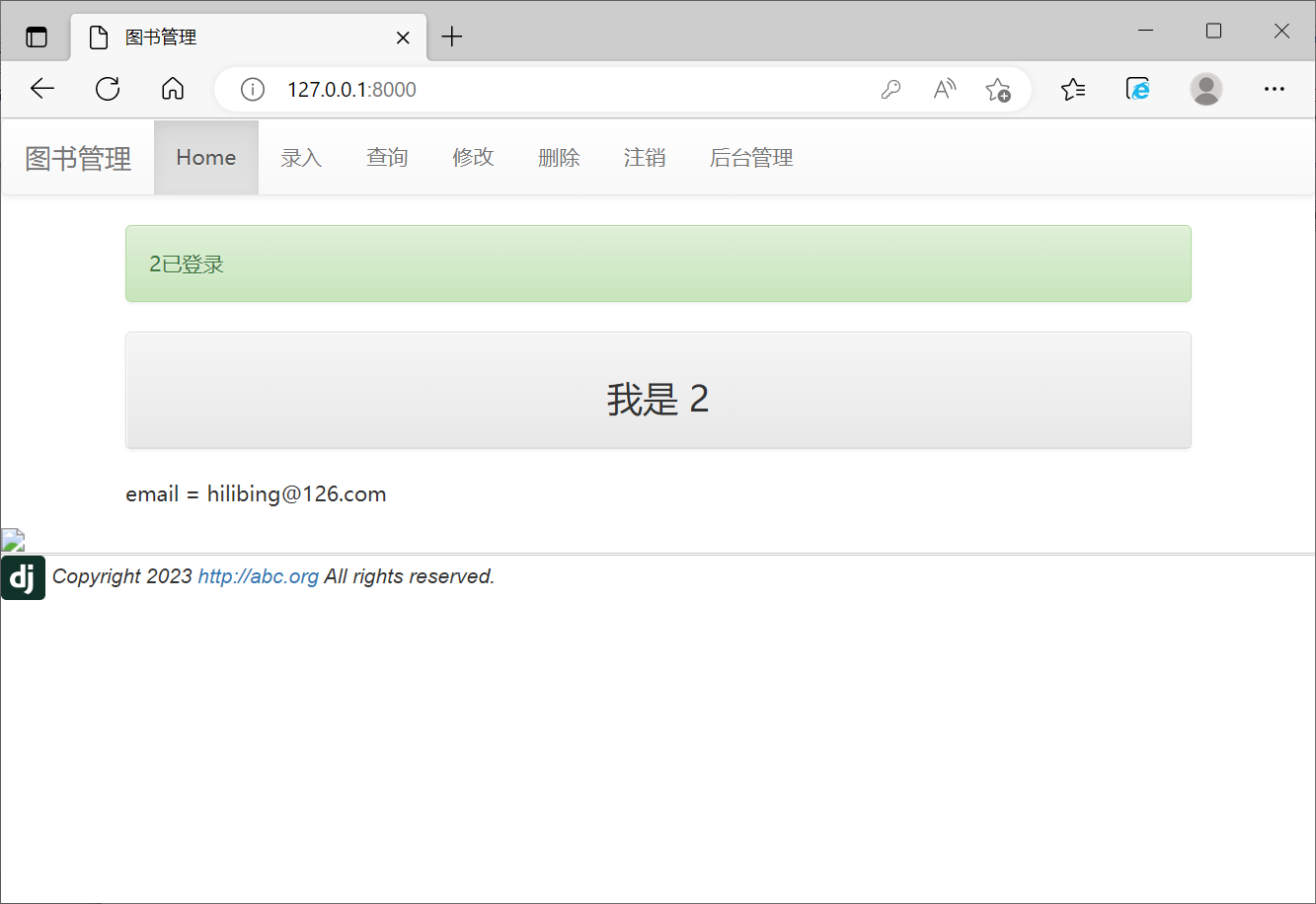


图9 登录成功页面

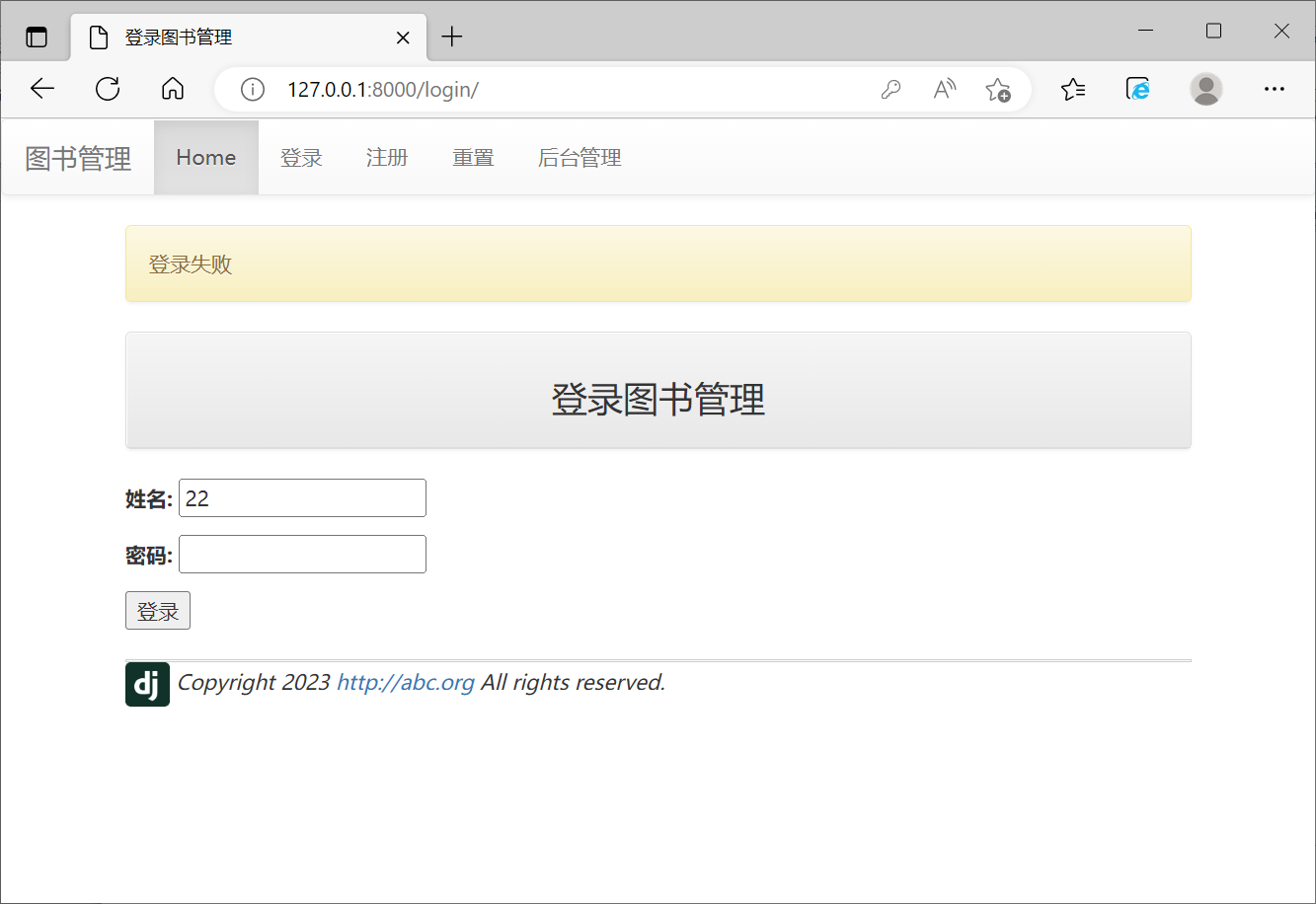


图10 登录失败页面

如果没有输入用户名或密码，系统提示“请填写此字段”，如图11所示。

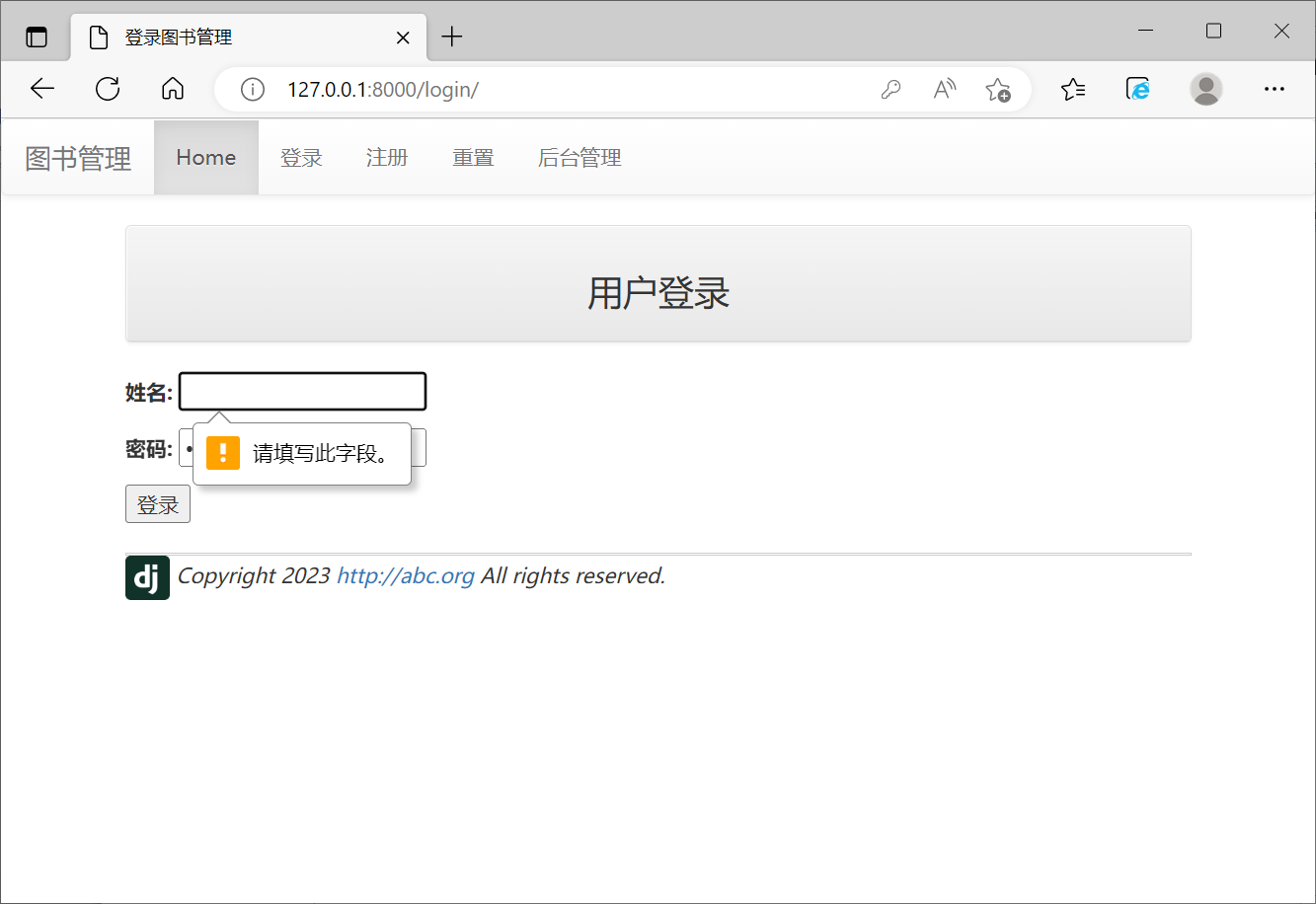


图11 填写错误页面

用户登录模块的工作流程：

（1）浏览器发送请求GET /logout；

（2）服务器基于urls.py查找相应函数， view.py的login，执行login返回模板login.html；

（3）用户输入数据后单击提交时，浏览器发送请求POST /login/；

（4）服务器执行login函数对请求的用户名和密码进行验证（与注册信息进行比较），返回模板login.html；

login函数首先判断请求类型。如果是GET请求，则返回空白表单，要求用户输入名称和密码；如果是POST请求，则参数request中提取用户输入的名称和密码，然后进行正确性和激活验证。如果验证通过，将用户请求重定向为根目录，服务器将受到请求GET /，这时会执行index函数，返回模板index.html。

index.html在用户成功登录后执行，所以在执行时需要对用户的登录是否登录进行判断。该判断操作可利用系统功能来完成，index函数的代码结构如下，

|  |
| --- |
| # view.py  from django.contrib.auth.decorators import login\_required  @login\_required(login\_url='/login')  def index(request):  username = request.user.username  return render(request, 'index.html', locals()) |

其中，login\_required的参数login\_url='/login/'用于指定在用户未登录时访问<http://127.0.0.1:8000/>将返回登录页面login.html，防止用户跳过登录环节直接操作系统。后函数index函数定义局部变量username，调用render返回渲染后的模板index.html。

系统的基础模板包括footer.html、heade.html、base.html三个文件。header.html定义了页面的顶部内容，footer.htm定义了页面底部的内容，base.htm包含了footer.html和footer.html，定义了整个页面。index.html文件的基本结构如下，

|  |
| --- |
| <!-- base.html -->  <html>  <head>  <title>{% block title %}{% endblock %}</title>  ……  </head>  <body>  ……  {% include "header.html" %}  {% block content %}{% endblock %}  {% include "footer.html" %}  </body>  </html> |

index.html，以及其他具有相同结构的页面继承base.html。在base.html文件中，定义了{% block content %}和{% block title %}。子模板通过重新定义这两个部分，展现自身独特的内容。index.html文件的基本结构如下，

|  |
| --- |
| <!-- index.html -->  {% extends "base.html" %}  {% block title %}……{% endblock %}  {% block content %}……{% endblock %} |

省略号部分按模块进行调整。

## 4、重置密码模块

在忘记密码时，不能直接修改密码，可用重置密码模块通过邮件验证的方式实现修改密码的功能。用户单击【重置】按钮打开重置按钮页面，如图12所示。

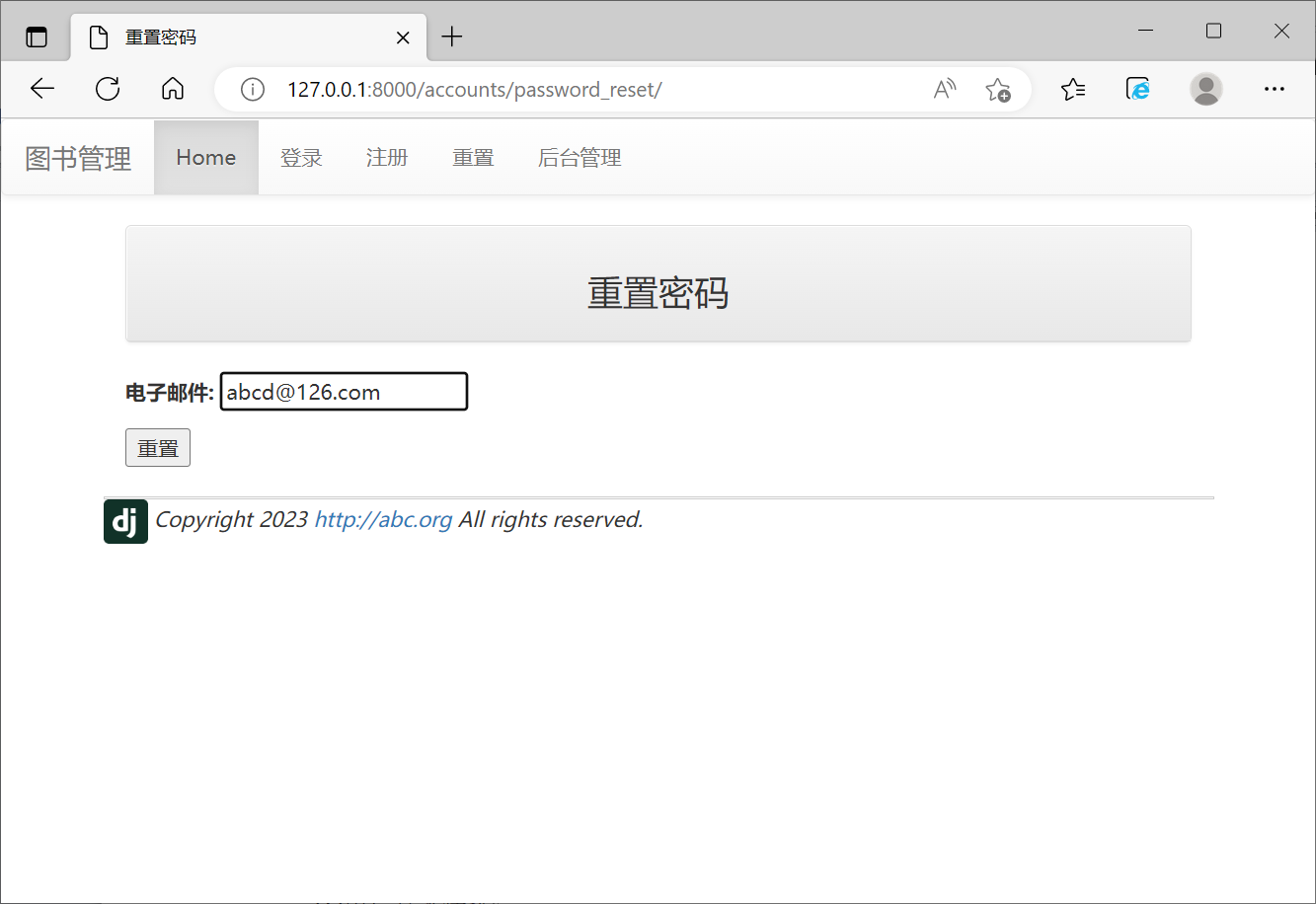


图12 重置密码页面

用户在【电子邮件】文本框中输入自己的邮箱地址（应该与注册邮箱相同），系统将将为所有具有该邮箱地址的账号发送激活邮件，显示重置结果页面如图13所示。



图13 重置密码结果页面

打开自己的邮箱，如图14所示。



图14 【重置密码】邮件

复制其中的网址，在浏览器中打开该网页，系统显示修改密码窗口如图15所示。

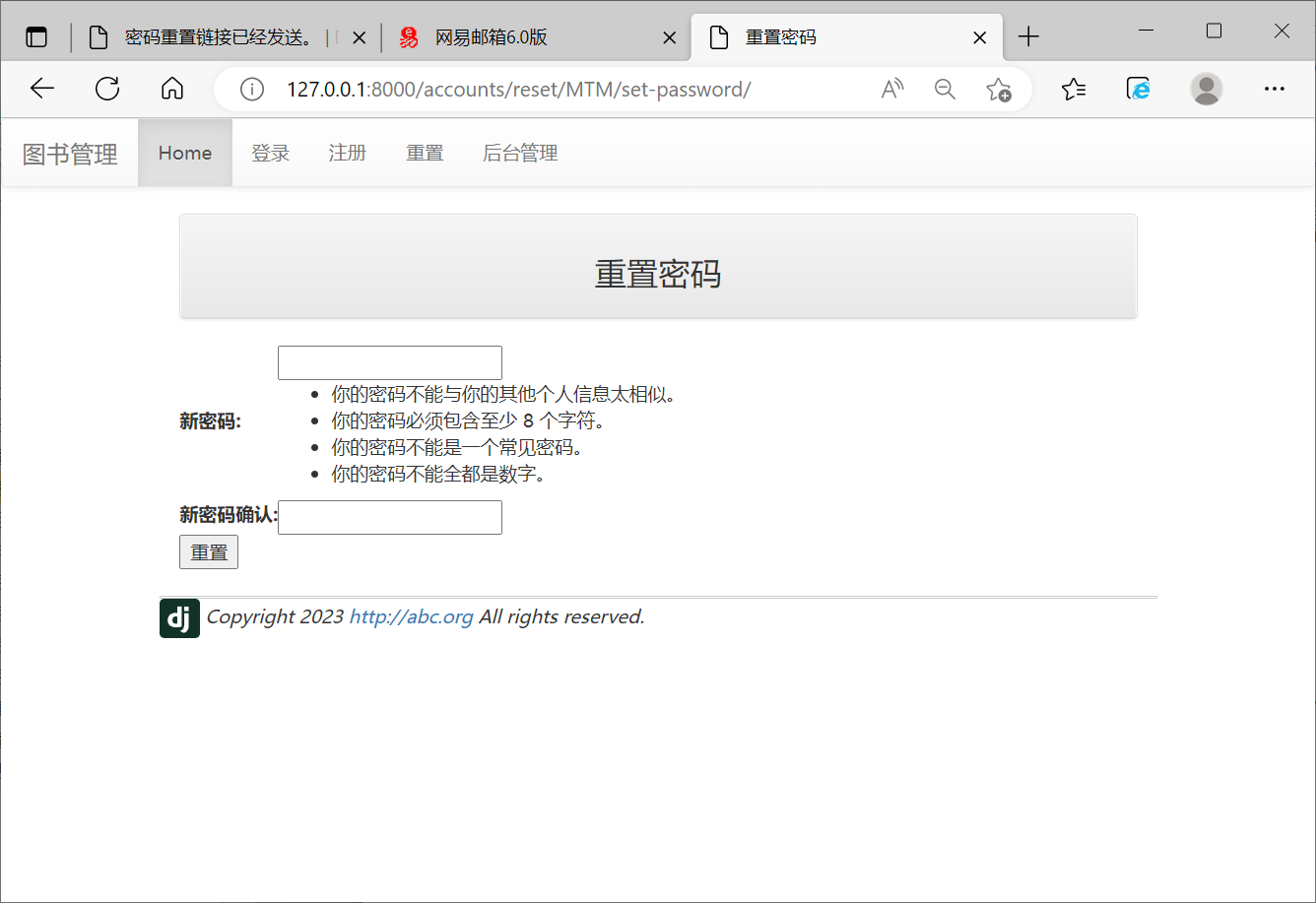


图15 重置密码页面

如果两次输入的密码不一致，单击【重置】按钮后，系统将给出提示，如图16所示。

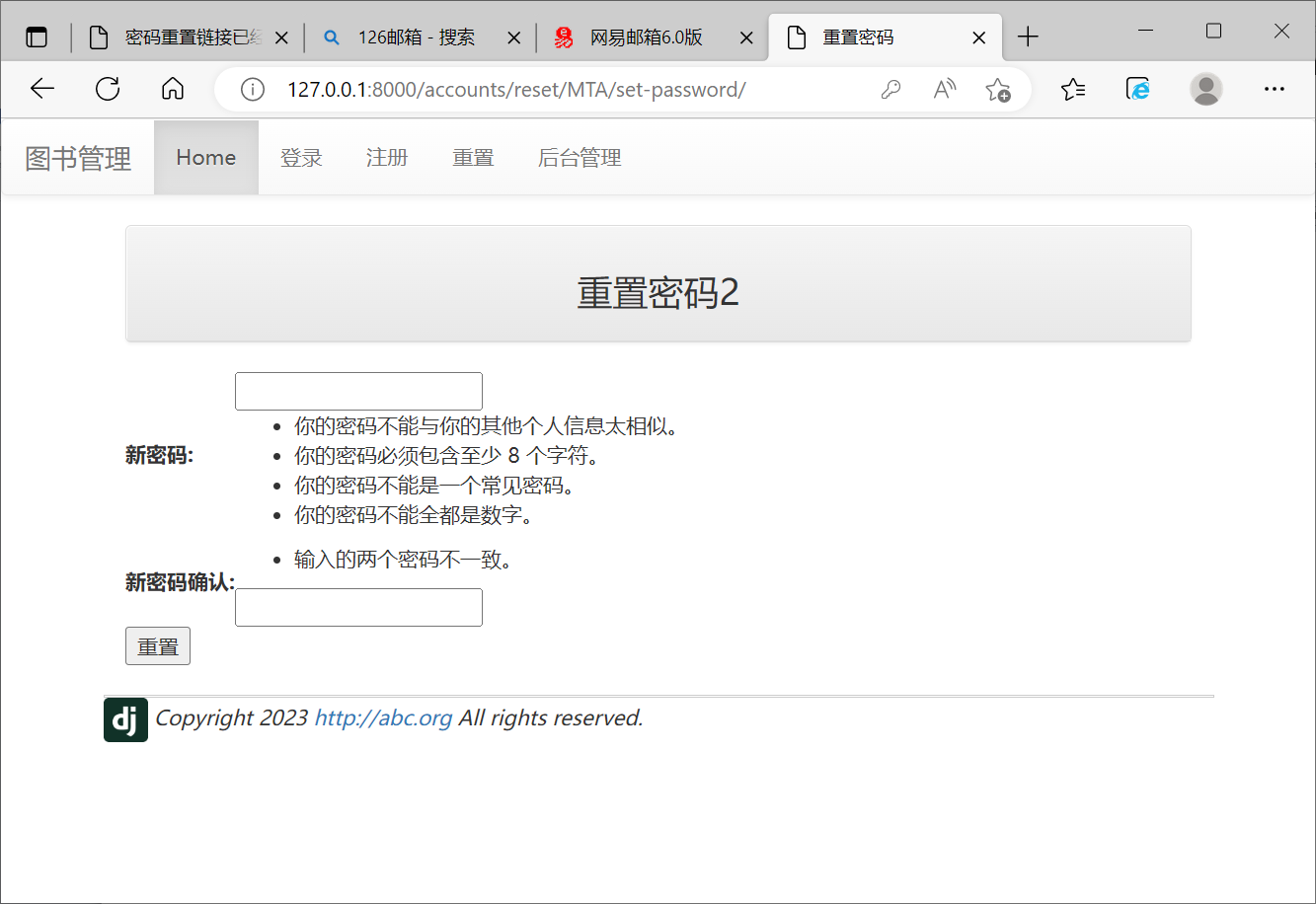


图16 两个密码不一致提示页面（在确认上方）

重置密码成功时，系统返回图17所示页面。



图16 重置密码成功页面

重置模块工作流程：

（1）浏览器发送请求GET /accounts/password\_reset/；

（2）服务器返回password\_reset\_form.html；

（3）用户输入邮箱数据后，单击【重置】按钮时，浏览器发送请求POST /accounts/ password\_reset/；

（4）服务器返回password\_reset\_done.html，并发向注册邮箱送激活邮件；

（5）用户点击邮件中的超链执行激活操作，如果激活成功，服务器返回password\_reset\_complete.html，

## 5、修改用户模块

用户修改模块可修改用户的用户名、用户密码、是否工作人员、是否超级用户等属性，还可完成删除用户操作。是否工作人员（字段if\_stuff）确定能否操作admin，是否超级用户（字段is\_superuser）确定是否拥有全部权限。图书管理系统直接利用admin来完成修改用户的有关操作。

用户单击菜单中的【后台管理】主菜单按钮，打开用户登录页面，如图17所示。

注意，图与标题不要分离！！！



图17 登录【后台管理】页面（已设置用户名和密码都是1）

完成登录操作后，单击左下角【用户】链接打开【浏览用户】的页面如图18所示。在该窗口中，选中用户名左边的复选框，在【动作】列表框选择【删除所选的用户】，再单击执行按钮可完成【删除用户】操作。



图18 浏览用户页面

单击用户名打开【修改用户】窗口，如图19所示。



图19 修改用户页面

在修改用户页面中，可以观察到密码字段已加密，不能浏览和编辑。下面利用django来观察密码字段数据，操作过程如下，

|  |
| --- |
| D:\www\_book>py manage.py shell  In [1]: from django.contrib.auth.models import User  In [2]: User.objects.first().password  Out[2]: 'pbkdf2\_sha256$390000$OGclx0Yx2dqiJr4JQ6LYCD$ui3F2fhVC8qvRMVHWY6P92BXiilt5MS7+nZ6k1aMxI4=' |

首先执行py manage.py shell打开shell窗口，然后导入User模块，User.objects.first()返回数据表的第一条记录，password则是该记录的密码字段。可以看到加密字段是加密数据，不能直接访问用户的真实密码。

在修改用户页面中修改数据后，单击窗口底部的【保存】按钮。

# 二、业务管理模块

## 1、录入图书模块

录入图书模块用于录入新图书的数据。它要求用户在登录后才能启动此项操作。图书编号字段不是书号，每本图书有自己唯一的编号，系统应检查图书编号是否重复。对每本图书可上传一张图片。

用户单击【录入】按钮，打开录入图书页面，如图20所示。编号字段被设计为整数类型，只能输入数字符号，不能输入字母，价格字段也是如此。

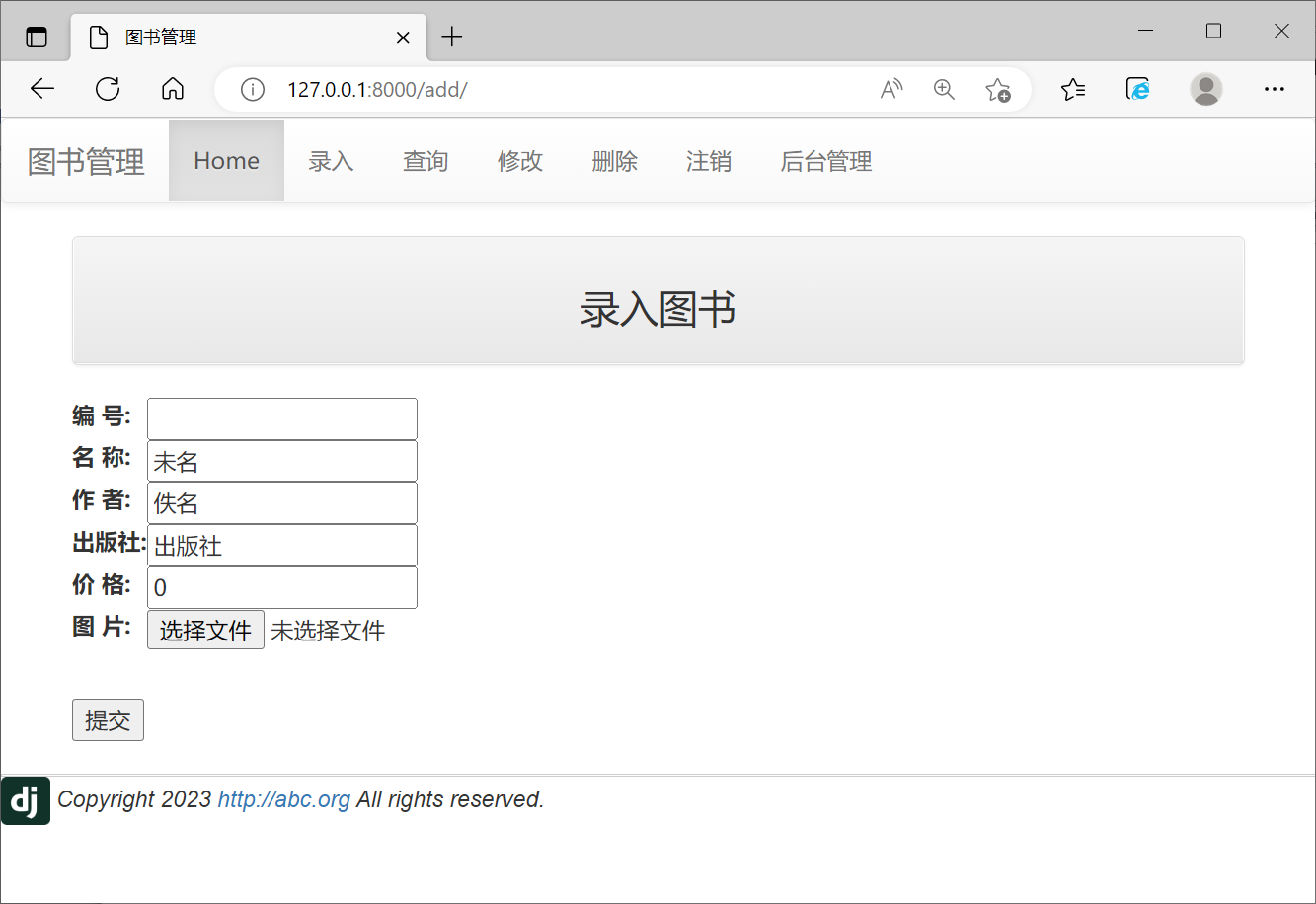


图20 录入图书页面

上传单击【选择文件】按钮，打开【选择文件】对话框，选择图片文件，如图21所示。单击【打开】按钮后，系统完成上传操作。

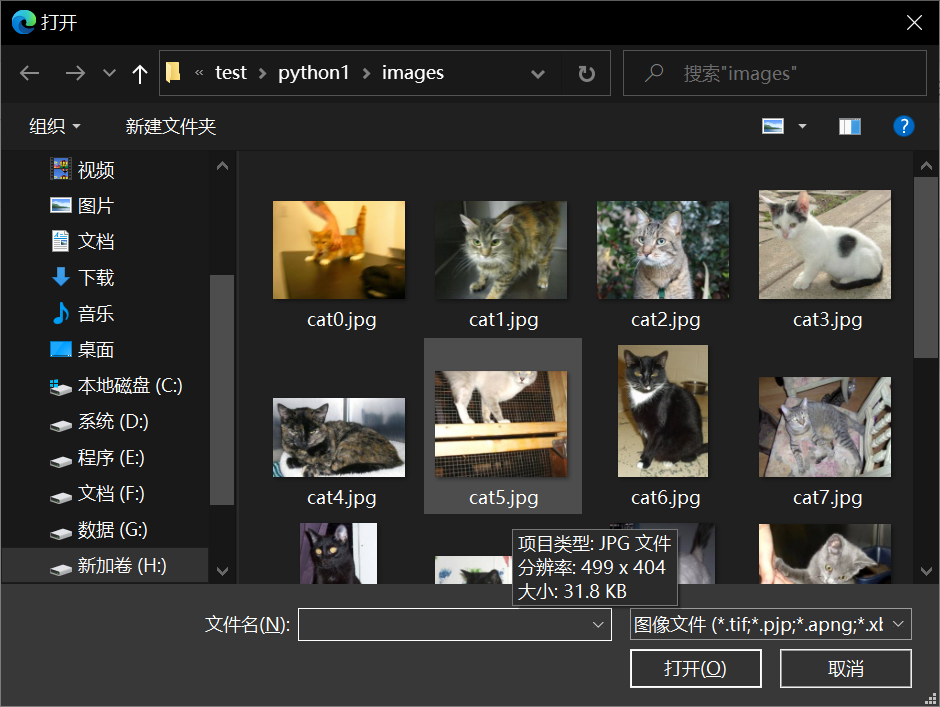


图21 【选择文件】对话框

图书编号具有唯一性，如果输入重复的编号，单击【提交】按钮后，系统提示出错，如图22所示。



图22 图书编号重复提示

输入数据，执行提交操作成功，系统提示“已已保存”，并再次打开录入图书页面，如图23所示。



图23 系统提示“已保存”

录入图书模块工作流程：

（1）浏览器发送请求GET /add/；

（2）服务器基于urls.py查找并执行响应函数add，返回模板add.html；

（3）用户输入数据后单击提交时，浏览器发送请求POST /add/；

（4）服务器再次执行add函数保存用户输入数据，并返回模板add.html；

数据库使用python默认的sqlite3，系统配置如下，

|  |
| --- |
| D:\www\_book>py manage.py makemigrations  D:\www\_book>py manage.py migrate |

图书管理系统利用Books类来实现对图书的增加、修改、删除和查询操作。Books类是django类models.Model的子类，定义在文件book/models.py中，代码如下，

|  |
| --- |
| # models.py  from django.db import models  class Books(models.Model):  bookId = models.IntegerField(null=False, unique=True)  bookName = models.CharField(max\_length=30, default='未名')  bookAuthor = models.CharField(max\_length=20, default='佚名')  bookPress = models.CharField(max\_length=20, default='出版社')  bookPrice = models.FloatField(default=0)  bookImage = models.ImageField(upload\_to='book', default='1.png') |

bookId字段具有唯一性要求，其他字段要求取非空值。为了便于测试，各字段的定义包含了默认值。bookImage是图片字段。图片保存在H:\www\_book\media\book目录中，而不是在数据库中（数据库中只有文件名）。如果替换图片，系统并未删除原来的图片。在新旧图片同名的情况下，系统自动新图片的文件名的末尾添加字符以示区别，需要专门删除不用的文件。处理图像文件，需要做以下配置，

|  |
| --- |
| # book\urls.py  from . import settings  from django.views.static import serve  urlpatterns = [……,  re\_path(r'^media/(?P<path>.\*)$', serve, {"document\_root":settings.MEDIA\_ROOT}),  ……] |
| # www\_book\settings.py  ……  MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media/')  MEDIA\_URL ='/media/'  #  from django.contrib import admin  from book import models  admin.site.register(models.Books) |

修改数据库结构后，执行下列命令完成数据更新：

|  |
| --- |
| D:\www\_book>py manage.py makemigrations  D:\www\_book>py manage.py migrate |

为了在视图和模板中重用模型中的字段信息，在book\form.py中定义了模型表单，代码如下。

|  |
| --- |
| # forms.py  from django import forms  from . import models  class BookForm(forms.ModelForm):  class Meta:  model = models.Books  fields = '\_\_all\_\_'  labels = {'bookId':'编 号', 'bookName':'名 称', 'bookPress':'出版社',  'bookAuthor':'作 者', 'bookPrice':'价 格', 'bookImage':'图 片'} |

add函数位于view.py文件中。add函数定义username局部变量，供footer.html确定用户是否登录系统，从而在主页面中显示不同的主菜单，add判断请求种类，如果是GET请求，则返回空白表单，要求用户输入各字段数据，如果是POST请求，则提取请求中的各字段数据，验证数据的正确性（是否重复），通过验证时执行存盘操作，最后返回空白表单供用户输入其他图书数据，没有通过验证时，则返回错误提示。

add函数代码如下（省略导入语句，由于模块间的相似性及论文篇幅的要求，其余模块不再详细说明设计细节），

|  |
| --- |
| # views.py  @login\_required(login\_url='/login/')  def add(request):  username = request.user.username  if request.method == 'GET':  book\_form = forms.BookForm()  else:  book\_form = forms.BookForm(request.POST, request.FILES)  if book\_form.is\_valid():  book\_form.save()  messages.add\_message(request, messages.SUCCESS, '已保存')  return redirect('/add')  else:  messages.add\_message(request, messages.INFO, '请检查所填内容')  return render(request, 'add.html', locals()) |

add,html继承了base.html，与前面index.html有类似的结果，不再详细说明。add页面的首部信息位于header.html中，它利用add函数中定义的username来区分是否登录，从而显示不同的主菜单内容，代码结构如下，

|  |
| --- |
| <!-- header.html -->  <nav class='navbar navbar-default'>  ……  {% if username %}  <li><a href='/add'>录入</a></li>  <li><a href='/select'>查询</a></li>  <li><a href='/update'>修改</a></li>  <li><a href='/delete'>删除</a></li>  <li><a href='/logout'>注销</a></li>  {% else %}  <li><a href='/login'>登录</a></li>  <li><a href='/accounts/register'>注册</a></li>  <li><a href='/accounts/password\_reset'>重置</a></li>  {% endif %}  <li><a href='/admin'>后台管理</a></li>  ……  </nav> |
|  |

## 2、查询图书模块

系统实现按书名查询数据的功能，结果为所有含有输入符号的图书。输入书名后，单击提交按钮，系统显示满足条件的图书信息。如果不输入查询关键字，则显示所有图书，如图24所示。（要求大家自行补充操作流程、程序流程）

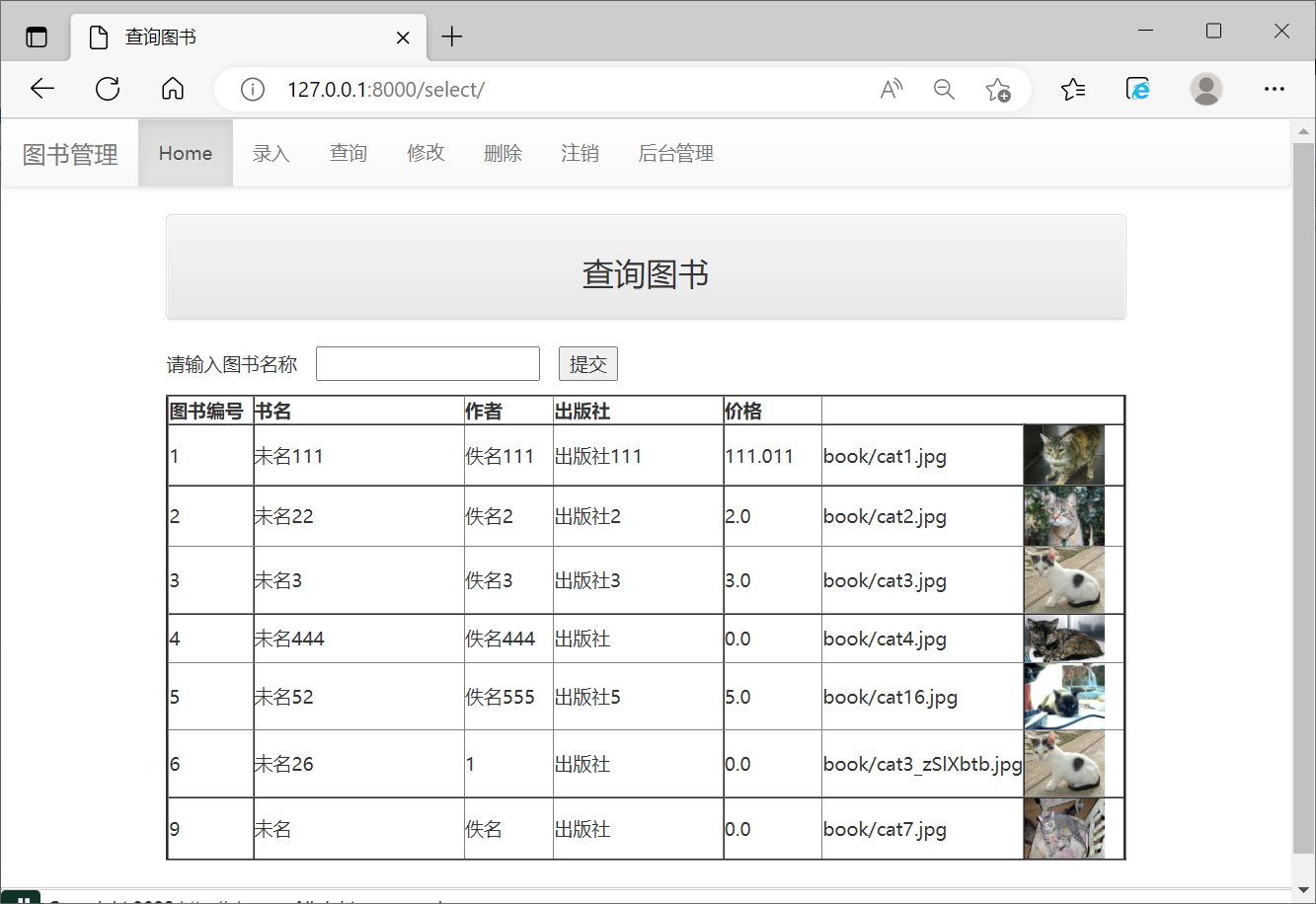


图24 【查询图书】页面

## 3、修改图书模块

修改图书分两步操作，先显示按书名的查询结果（图25上），单击【修改】按钮后打开编辑页面（图25下），最后单击提交按钮存盘。（要求大家自行补充操作流程、程序流程）

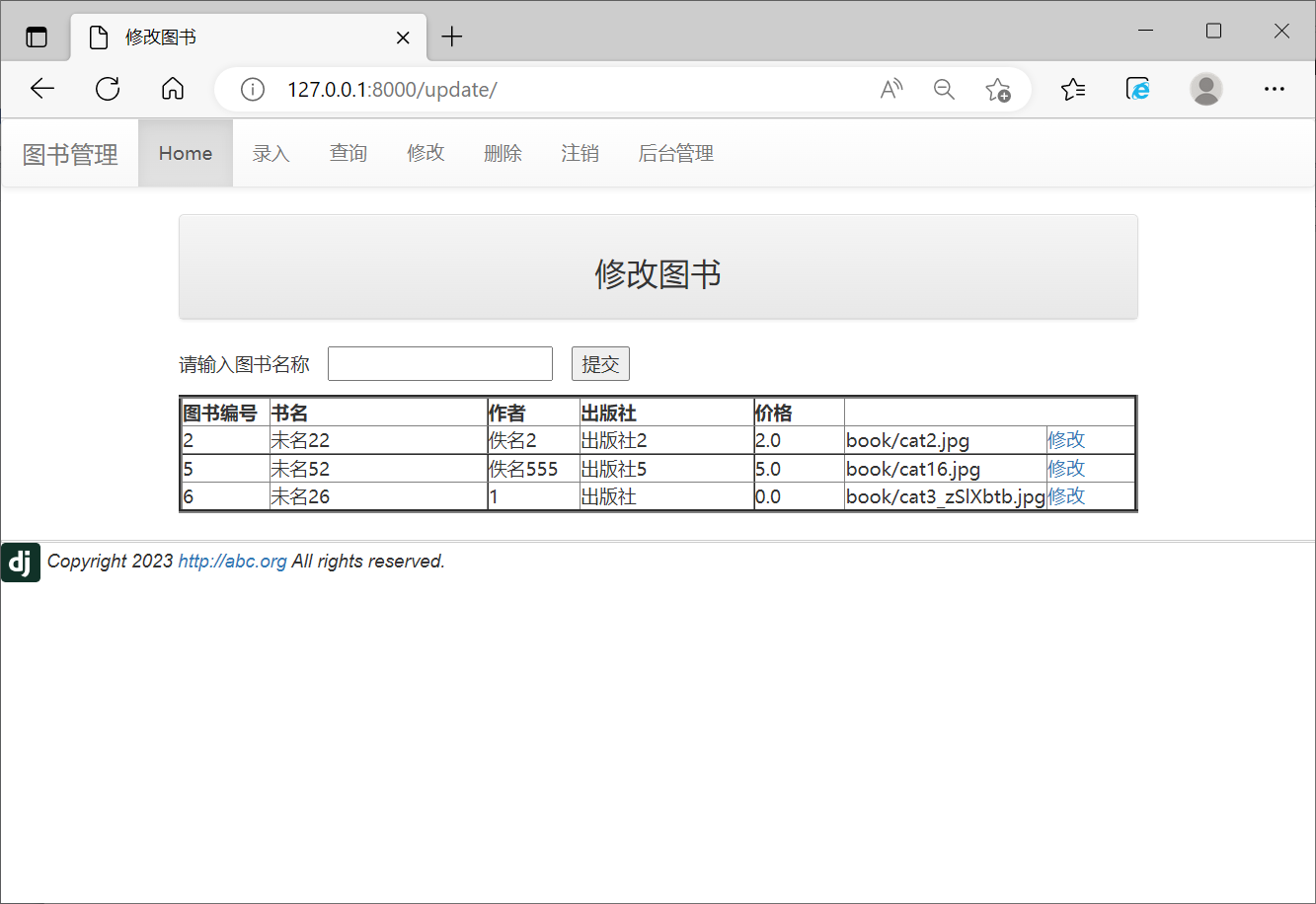
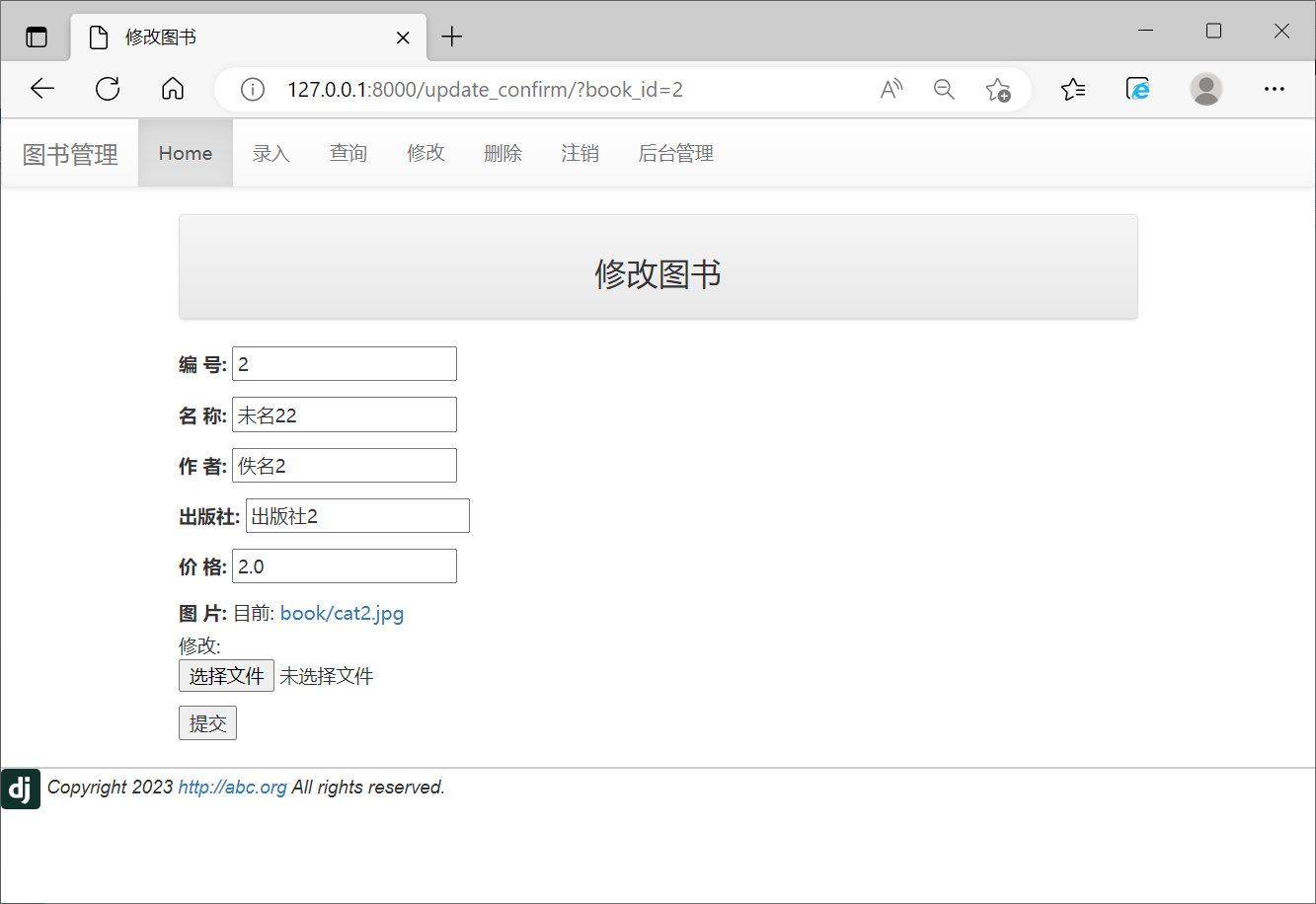
 

图25 修改图书页面

## 4、删除图书模块

删除图书也分两步操作，先显示按书名的查询结果），单击【删除】按钮执行删除操作（删除前不再提示）。

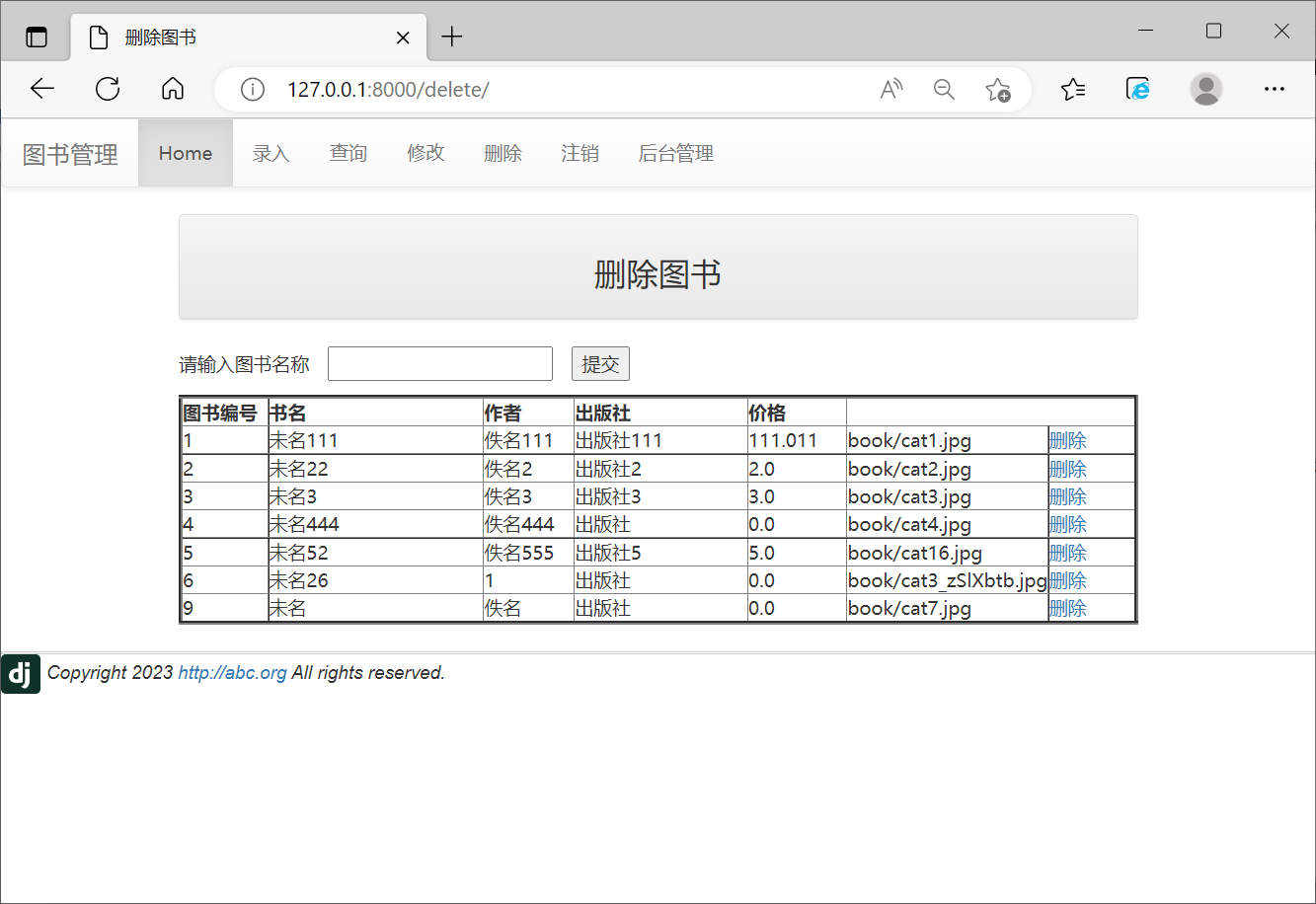


图26 删除图书页面

可说明两位用户同时执行删除操作，不会引发异常。（要求大家自行补充操作流程、程序流程）

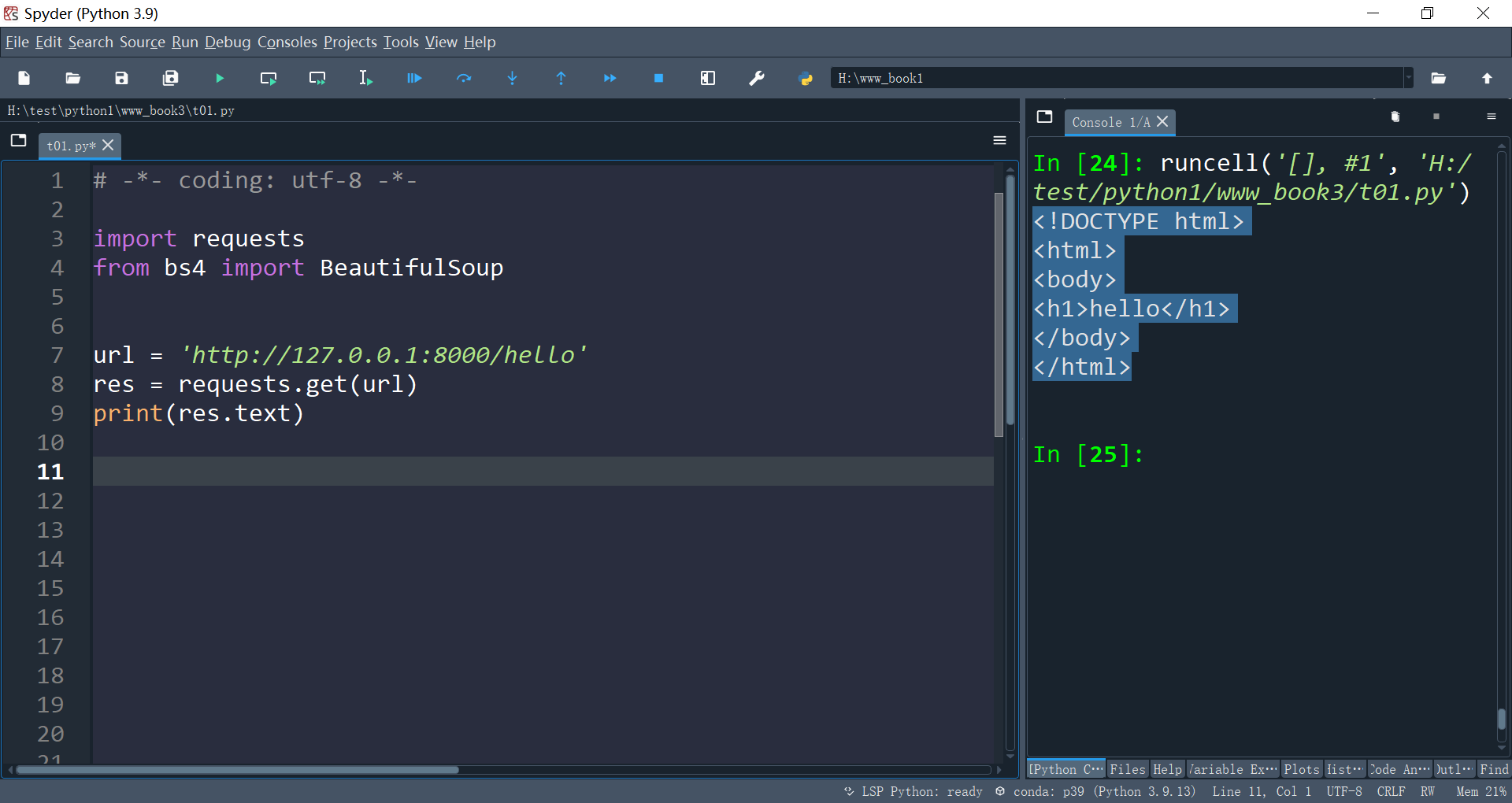
也可在spyder中发送请求，如图27所示。

图27 在命令窗口中观察服务器的返回结果