**《Python设计报告》**

项目名称：自建公告板

姓 名：熊嘉恒

班 级：通信工程1702

学 号：U201713438

1. 需求分析

由于目前互联网发展迅速，人们之间的联系越来越紧密。各类论坛，BBS等工具层出不穷，人们可以在这些软件上面发布主题吸引更多的人来讨论此问题。也可以让自己的想法被更多人所知。通过部署到python自带的CGI服务器上来进行浏览器访问此BBS，并且通过数据库操作来存储和查找帖子。

1. 设计内容

本设计程序将使用CGI和可靠的数据库创建一个通过Web发布和回复消息的简单系统，它可以作为一个论坛使用。具体实现功能如下：

1. 显示当前所有消息的主题；
2. 支持在消息下方以缩放的方式进行回复；
3. 让用户能够查看既有的消息
4. 让用户能够恢复既有的消息
5. 设计工具
6. 操作系统：Windows 10
7. 编程工具：Pycharm2017
8. python环境：anaconda --python3.7
9. 数据库：postgreSQL 12
10. 浏览器：Firefox
11. 前期准备
12. 下载安装postgreSQL数据库，网址为https ://www.enterprisedb.com/downloads/postgres-postgresql-downloads#windows，PostgreSQL Version为12.0，选择系统为Windows x86-64，点击下载，之后根据引导进行安装
13. 进入SQLshell命令行工具，执行命令CREATE DATABASE baz;创建一个名为baz的数据库。执行命令\c baz;切换到数据库baz下，输入并执行命令：

CREATE TABLE messages (

id SERIAL PRIMARY KEY,

subject TEXT NOT NULL,

sender TEXT NOT NULL,

reply\_to INT ,

text TEXT NOT NULL

);建立messages表单：

1. id:用于标识消息 。每条消息都会自动获得由数据库管理器提供的独一无二的ID，因此无需在python代码中指定这些ID；
2. Subject:包含消息主题的字符串；
3. Sender:包含发送者的姓名，电子邮件地址或其他类似信息的字符串
4. Reply\_to:如果消息是另一条消息的回复，这个字段将包含那条消息的ID，否则为空；
5. Text:包含消息正文的字符串
6. 打开Anaconda promt,执行命令conda install psycopg2，安装psycopg2模块
7. 由于postgreSQL数据库服务器不支持GBK字元编码，因此需要将本地环境字元编码改为支持UTF-8:

点击【控制面板】->【时钟和区域】->【区域】->【管理】->【更改系统区域设置】;勾选 Beta版：使用Unicode UTF-8提供全球语言支持

1. 设计程序

删除脚本save.py

删除数据

点击删除

个人主页author.py

点击作者

点击消息

获取消息

数据库

消息查看器view.py

公告板主页main.py

.

点击发布主题

插入数据库

点击回复消息

保存脚本save.py

消息编辑器edit.py

保存

文件夹结构：www——cgi-bin——main.py

|\_\_\_\_\_\_\_\_view.py

|\_\_\_\_\_\_\_\_edit.py

|\_\_\_\_\_\_\_\_save.py

|\_\_\_\_\_\_\_\_author.py

|\_\_\_\_\_\_\_\_delete.py

1. 公告板主页main.py：

这个页面，我们需要将消息列表显示出来。并且，需要提供一个发布消息的文字链接，能够进入消息编辑页。显示消息列表的内容，我们需要查询出数据库中所有的消息内容，并将消息内容有层级的显示在页面上。

#!/usr/bin/python  
  
print('Content-type: text/html\n')  
  
**import** cgitb;  
  
cgitb.enable()  
  
**import** psycopg2.extras  
  
conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
curs = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
 <title>我的论坛</title>  
</head>  
<body>  
 <h3>熊嘉恒的BBS</h3>  
''')  
  
sql = 'select \* from messages'  
curs.execute(sql)  
messages = curs.fetchall()  
top\_level = []  
children = {}  
**for** message **in** messages:  
 parrent\_id = message['reply\_to']  
 **if** parrent\_id **is None**:  
 top\_level.append(message)  
 **else**:  
 children.setdefault(parrent\_id, []).append(message)  
  
  
**def format\_show**(message): # 创建递归函数  
 print('<h5><a href="author.py?id=%(id)i">%(sender)s：</a><a href="view.py?id=%(id)i">%(subject)s</a></h5>' % message)  
 print('<font size="2">{}</font>'.format(message['text']))  
 **try**:  
 kids = children[message['id']]  
 **except** KeyError:  
 **pass  
 else**:  
 print('<blockquote>')  
 **for** kid **in** kids:  
 format\_show(kid)  
 print('</blockquote>')  
  
  
**for** message **in** top\_level: # 遍历顶级消息列表  
 format\_show(message)  
  
print('''  
<hr/>  
<a href="edit.py"><font size="2">发布消息</font></a>  
</body>  
</html>  
''')

工作原理如下：

1. 对于每条消息，获取其reply\_to字段。如果这个字段为none(不是回复)，就将当前消息加入顶级消息列表中，否则就将其附加到子消息列表children[parent\_id]的末尾；
2. 对于每条顶级消息，调用format\_show函数，打印消息主题。如果它有子消息，就打印起始标签<blockquote>，对于每条子消息递归调用format\_show，再打印结束标签</blockquote>
3. 因为点击消息标题需要跳转到消息查看页，我们需要为消息标题加上链接，并且在链接中加入消息的id。消息查看页的名称是“view.py”，所以链接为：view.py?id=%(id)i。这是一个相对路径，它能够在与当前文件（main.py）同一目录下找到名为“view.py”的文件将其打开。链接中的问号“?”，表示后方内容为参数变量和值，“=”前方的“id”是变量，后方的“%(id)i”是值。“%(id)i”中的“%i”表示这里需要插入一个十进制的整数，“(id)”则能够从“message”字典中获取到。通过这样的处理，我们在打开“view.py”文件的时候，就能够通过cgi模块获取息id。同理，为作者名加上一个跳转到author.py的链接，以进入作者的主页
4. 消息查看器view.py：

这个页面中，我们需要根据从首页传过来的id，显示相应的消息内容。脚本view.py根据提供给他的CGI参数id来从数据库获得一条消息，再使用得到的值生成一个简单的HTML页面。这个页面包含一个返回到主页面(main.py)的链接，还包括一个到edit.py的链接，但是将reply\_to的值设置为id的值，以确保新消息是对当前消息的回复。

sys, psycopg2.extras  
  
form = cgi.FieldStorage()  
id = form.getvalue('id') # 获取URL中的id参数  
  
conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
curs = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
 <title>查看消息</title>  
</head>  
<body>  
''')  
**try**:  
 id = int(id) # 将id转换为整数类型  
**except**: # 如果转换异常  
 print('无效的消息id！')  
 sys.exit() # 退出脚本执行  
  
sql = 'select \* from messages where id=%i' % id  
curs.execute(sql)  
rows = curs.fetchall()  
**if not** rows: # 如果没有查询结果  
 print('消息不存在！')  
 sys.exit() # 退出脚本执行  
row = rows[0] # 获取查询结果中的第一个字典  
print('''  
<p>  
<h5>%(subject)s</h5>  
<font size="2"><a href="author.py?id=%(id)i">%(sender)s</a></font></br>  
<pre>%(text)s</pre>  
</p>  
<hr/>  
<a href="main.py"><font size="2">返回首页</font></a>|  
<a href="edit.py?reply\_to=%(id)s"><font size="2">回复消息</font></a>  
<a href="delete.py?id=%(id)s"><font size="2">删除消息</font></a>  
</body>  
</html>  
''' % row)

在上方代码中，HTML末尾部分，需要加入三个文字链接，分别能够返回首页，回复消息以及删除消息。返回首页的链接，可以在<a>标签的“href”属性中直接填入相对路径“main.py”。回复消息的链接，需要能够打开“edit.py”并且将被回复消息的id通过URL中的参数“reply\_to”进行传递。

1. 消息编辑器edit.py：

脚本edit.py实际上承担了双重职责：即用于编辑新消息，又用于编辑回复。这两项功能的差别并不大：如果在CGI请求中提供了reply\_to，就将其存储在编辑表单中隐藏的input元素中。在Web表单中，隐藏的input元素用于临时存储信息。它们不想文本区域等元素那样是用户能够看到的，但是它们的值也将传递给表单的属性acton指定的CGI脚本，这让生成表单的脚本能够向处理该表单的脚本传递信息。

#!/usr/bin/python  
  
print('Content-type: text/html\n')  
  
**import** cgitb;  
  
cgitb.enable()  
  
**import** sys, cgi, psycopg2.extras  
  
form = cgi.FieldStorage()  
reply\_to = form.getvalue('reply\_to') # 获取被回复消息的id  
senderr = '' # 创建标题变量  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
 <title>编辑消息</title>  
</head>  
<body>  
<form action="save.py" method="post">  
''') # 创建HTML基本代码并添加表单  
**if** reply\_to **is not None**: # 如果获取到被回复消息的id  
 **try**:  
 reply\_to = int(reply\_to) # 将id转换为整数类型  
 **except**: # 如果转换发生异常  
 print('无法回复该消息')  
 sys.exit() # 退出脚本执行  
 **else**: # 如果id可用  
 print(' <input type="hidden" value="%s" name="reply\_to"/>' % reply\_to)# 将被回复消息id写入表单元键值  
 conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
 curs = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)  
 sql = 'select sender from messages where id=%i' % reply\_to # 通过被回复消息id查询被回复消息的标题  
 curs.execute(sql)  
 sender = curs.fetchone()[0] # 将查询到的用户名存入变量  
 **if not** sender.startswith('回复：'): # 如果标题开头没有“回复：”字样  
 sender = '回复：' + sender # 为标题添加“回复：”字样  
  
print('''  
 <b><font size="2">主题：</font></b><br/>  
 <input type="text" value="%s" style="width:240px" name="subject"/><br/>  
 <b><font size="2">作者：</font></b><br/>  
 <input type="text" style="width:240px" name="sender"/><br/>  
 <b><font size="2">编辑内容：</font></b><br/>  
 <textarea rows="5" cols="50" style="width:240px" name="text"></textarea><br/>  
 <input type="submit" value="发布">  
</form>  
<hr/>  
<a href="main.py"><font size="2">返回首页</font></a>  
</body>  
</html>  
''' % sender)

如果是发布新消息，需要提供“subject”、“sender”和“text”三个字段内容。而如果是回复消息，在插入数据表时，还需要提供“reply\_to”字段。那么，是不是回复消息，我们可以从表单中尝试获取“reply\_to”的值，如果能够获取到id，就是回复消息，如果获取到的是None，则是新发布的消息。另外，在这个页面中，我们还需要对字段中的单引号“’”进行处理，因为SQL语句中的字符串字段会带有单引号，如果字段值中的单引号不做处理，将会引发异常。如果在字段值中有单引号的话，我们需要用两个单引号来表示。

1. 保存脚本save.py：

脚本save.py从edit.py生成的表单中接收有关一条消息的信息，并将其存储到数据库中。这意味着需要使用SQL INSERT命令，同时由于对数据库做了修改，必须调用conn.commit，避免丢失

#!/usr/bin/python  
  
print('Content-type: text/html\n')  
  
**import** sys, cgi, psycopg2  
  
  
**def quote**(string): # 定义处理单引号的函数  
 **if** string: # 如果不是空值或None值  
 **return** string.replace("'", "''") # 将单引号替换为两个单引号  
 **else**:  
 **return** string  
  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
</head>  
<body>  
''')  
  
form = cgi.FieldStorage()  
subject = quote(form.getvalue('subject')) # 获取字段值并进行单引号处理  
sender = quote(form.getvalue('sender')) # 获取字段值并进行单引号处理  
reply\_to = form.getvalue('reply\_to')  
text = quote(form.getvalue('text')) # 获取字段值并进行单引号处理  
**if** reply\_to **is not None**: # 如果有被回复消息的id  
 **try**:  
 reply\_to = int(reply\_to) # 将id转换为整数类型  
 **except**: # 如果转换异常  
 print('''  
 <font size="2" color="red">无法发布该消息！</font>  
 <hr/>  
 <a href="main.py"><font size="2">返回首页</font></a>  
 ''')  
 sys.exit() # 退出脚本执行  
**if not** (subject **and** sender **and** text): # 如果有任何一个字段值为空值或者None值  
 print('''  
 <font size="2" color="red">请输入回复内容！</font>  
 <hr/>  
 <a href="edit.py?reply\_to=%s"><font size="2">返回编辑</font></a>  
 ''' % reply\_to)  
**else**: # 如果是符合要求的输入  
 conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
 curs = conn.cursor()  
 **if** reply\_to **is None**: # 如果是新发布消息  
 sql = "insert into messages(subject,sender,text) values('%s','%s','%s')" % (subject, sender, text)  
 **else**: # 如果是回复消息  
 sql = "insert into messages(subject,sender,reply\_to,text) values('%s','%s','%i','%s')" % (  
 subject, sender, reply\_to, text)  
 curs.execute(sql)  
 conn.commit()  
 print('''  
 <font size="2" color="green">发布成功</font>  
 <hr/>  
 <a href="main.py"><font size="2">返回首页</font></a>  
 ''')  
  
 print('''  
 </body>  
 </html>  
 ''')

1. 删除脚本delete.py：

脚本delete,py从view.py接收数据，并使用delete命令将此数据和在它下面的回复一起删除

#!/usr/bin/python  
  
print('Content-type: text/html\n')  
  
**import** sys, cgi, psycopg2  
  
  
form = cgi.FieldStorage()  
id = form.getvalue('id') # 获取URL中的id参数  
**try**:  
 id = int(id) # 将id转换为整数类型  
**except**: # 如果转换异常  
 print('无效的消息id！')  
 sys.exit() # 退出脚本执行  
  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
</head>  
<body>  
''')  
  
id = str(id)  
  
conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
curs = conn.cursor()  
sql = "delete from messages where id=%s or reply\_to=%s" %(id,id)  
curs.execute(sql)  
conn.commit()  
print('''  
<font size="2" color="green">删除成功</font>  
<hr/>  
<a href="main.py"><font size="2">返回首页</font></a>  
 ''')  
  
print('''  
</body>  
</html>  
''')

1. 个人主页author.py：

通过在脚本main.py和view.py对作者名字设置链接，将数据id传输进入author.py。打印数据库中sender字段与所传授数据sender字段相同的数据，并过滤掉reply\_to不为空的字段，以找到该作者发布的主题

#!/usr/bin/python  
print('Content-type: text/html\n')  
  
**import** cgitb;  
  
cgitb.enable()  
  
**import** cgi,sys, psycopg2.extras  
  
form = cgi.FieldStorage()  
id = (form.getvalue('id') ) # 获取URL中的id参数  
**try**:  
 id = int(id) # 将id转换为整数类型  
**except**: # 如果转换异常  
 print('无效的消息id！')  
 sys.exit() # 退出脚本执行  
  
  
conn = psycopg2.connect(database='baz', user='postgres', password='123456')  
curs = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)  
sql = 'select sender from messages where id=%i'%id  
curs.execute(sql)  
sender = curs.fetchone()[0]  
  
  
print('''  
<!DOCTYPE html>  
<html lang="zh\_cn">  
<head>  
 <meta charset="utf-8">  
 <title>%s的主页</title>  
</head>  
<body>  
 <h3>%s发布过的主题</h3>  
'''%(sender,sender))  
  
sql = 'select \* from messages'  
curs.execute(sql)  
messages = curs.fetchall()  
  
top\_level = []  
children = {}  
**for** message **in** messages:  
 parrent\_id = message['reply\_to']  
 **if** parrent\_id **is None**:  
 top\_level.append(message)  
 **else**:  
 children.setdefault(parrent\_id, []).append(message)  
  
  
  
**for** message **in** top\_level: # 遍历顶级消息列表  
 **if** message['sender']==sender **and** message['reply\_to'] **is None**:  
 print('<h5><a href="view.py?id=%(id)i">%(sender)s：%(subject)s</a></h5>' % message)  
 print('<font size="2">{}</font>'.format(message['text']))  
  
  
print('''  
<hr/>  
<a href="main.py"><font size="2">返回首页</font></a>  
</body>  
</html>  
''')

1. 运行结果

将文件夹www放置于桌面上，打开Windows命令行，执行

cd C:\Users\xiongjiaheng\Desktop\WWW

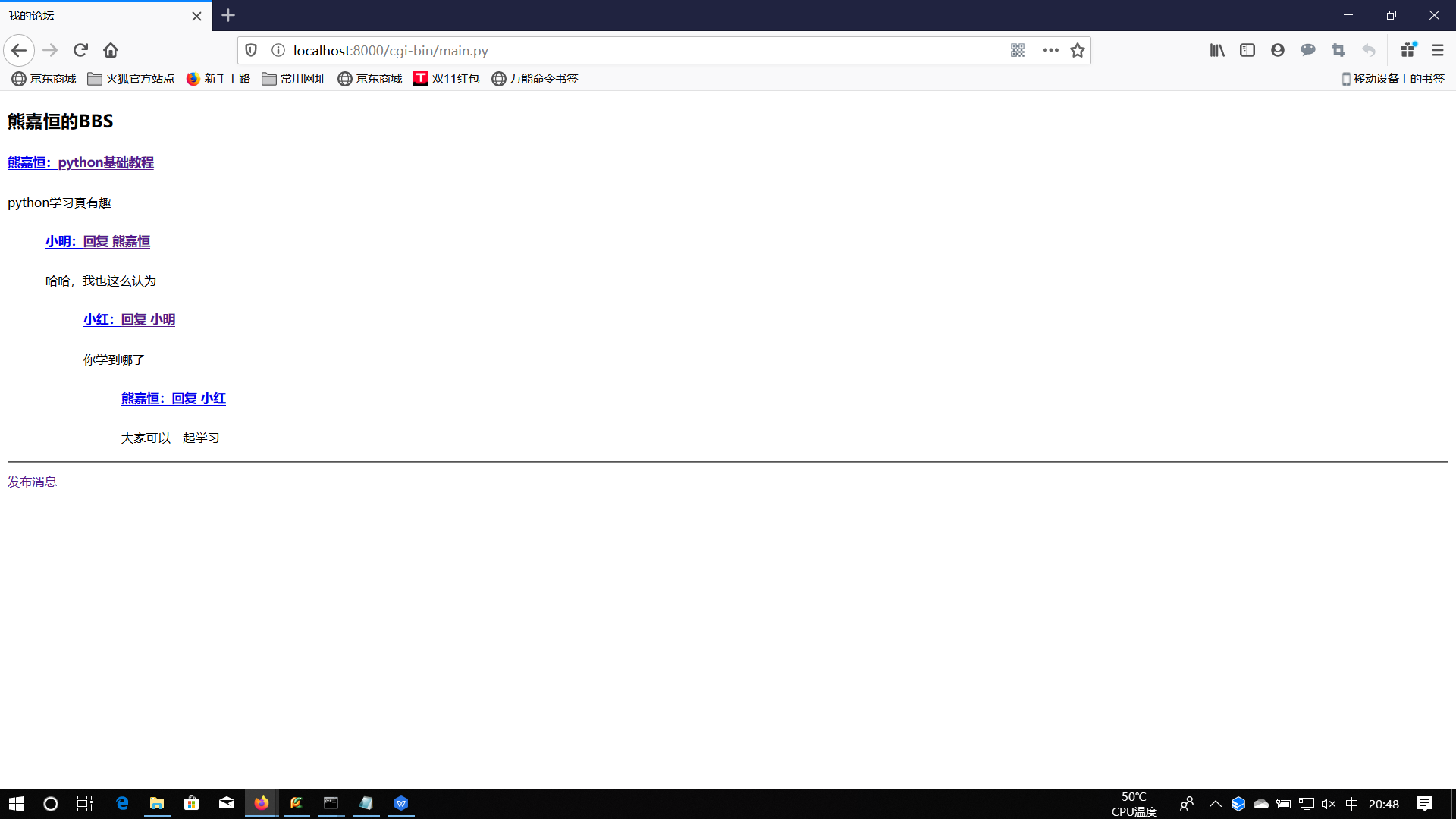
将目录切换至www文件夹下，执行指令

python -m http.server --cgi 8000

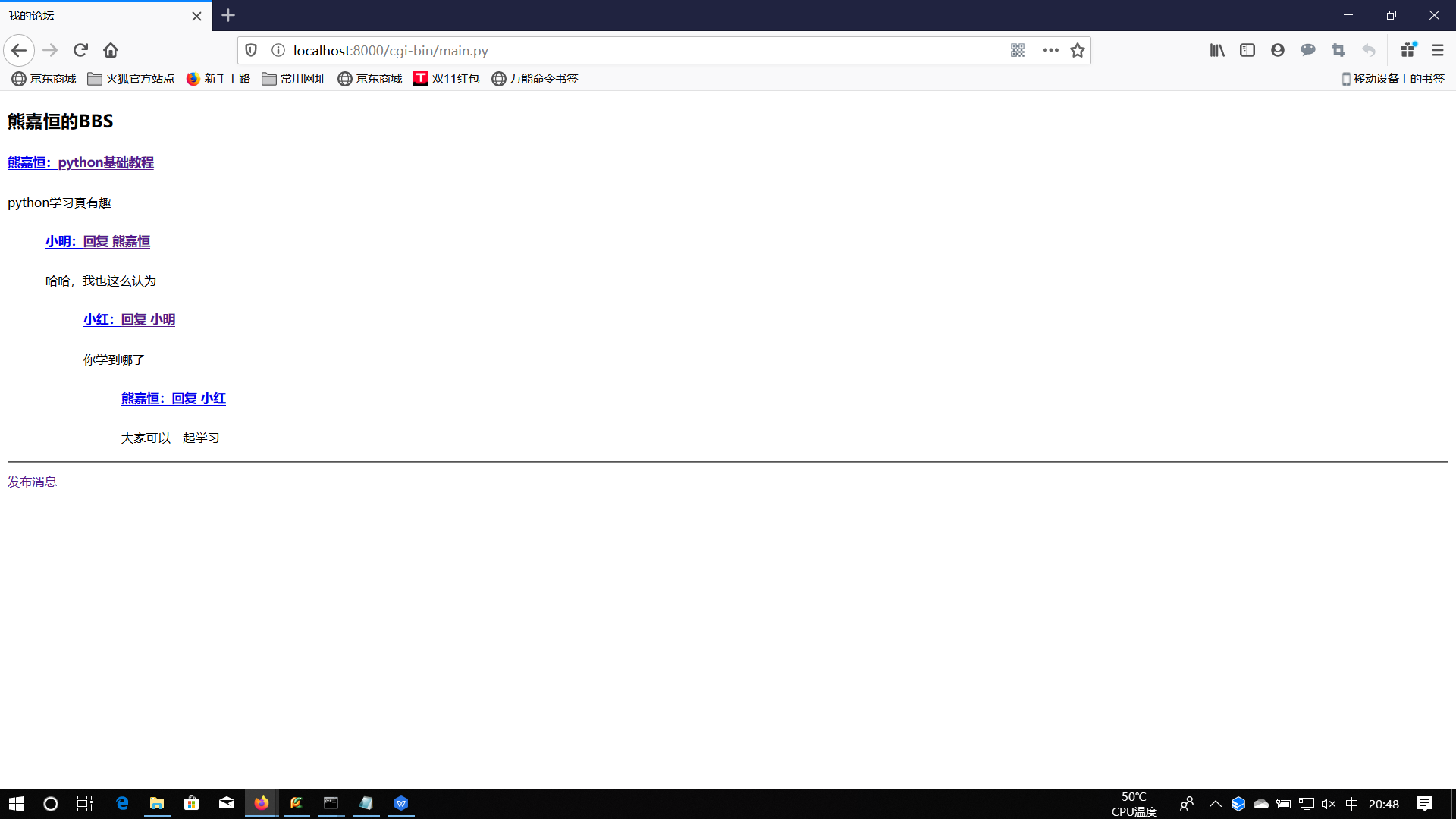
启动python3.7.5自带的简易CGI服务器，并在主机的8000端口上运行http服务

使用浏览器访问http://localhost:8000/cgi-bin/main.py

1. 公告板主页main.py

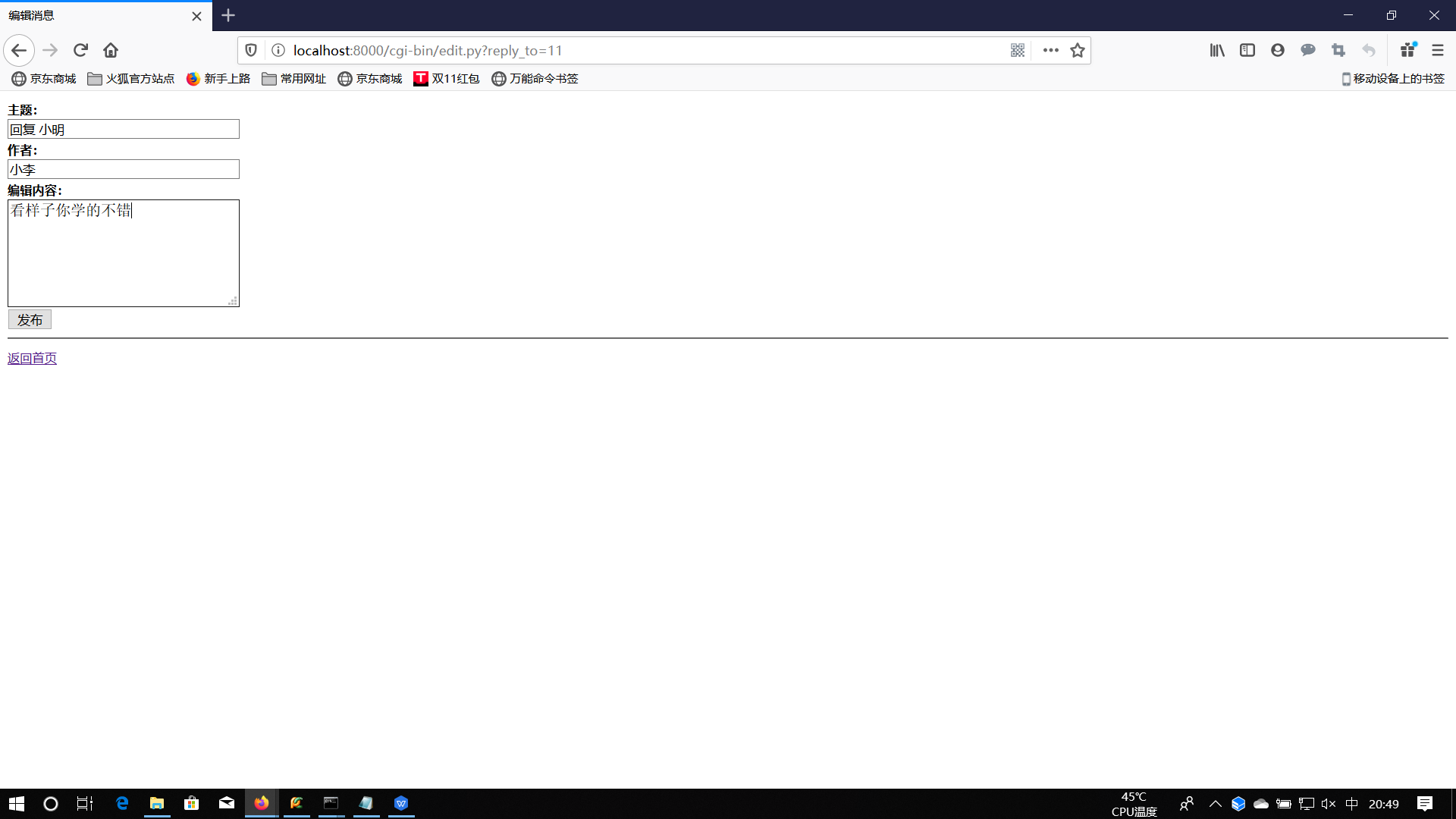


1. 消息查看器view.py

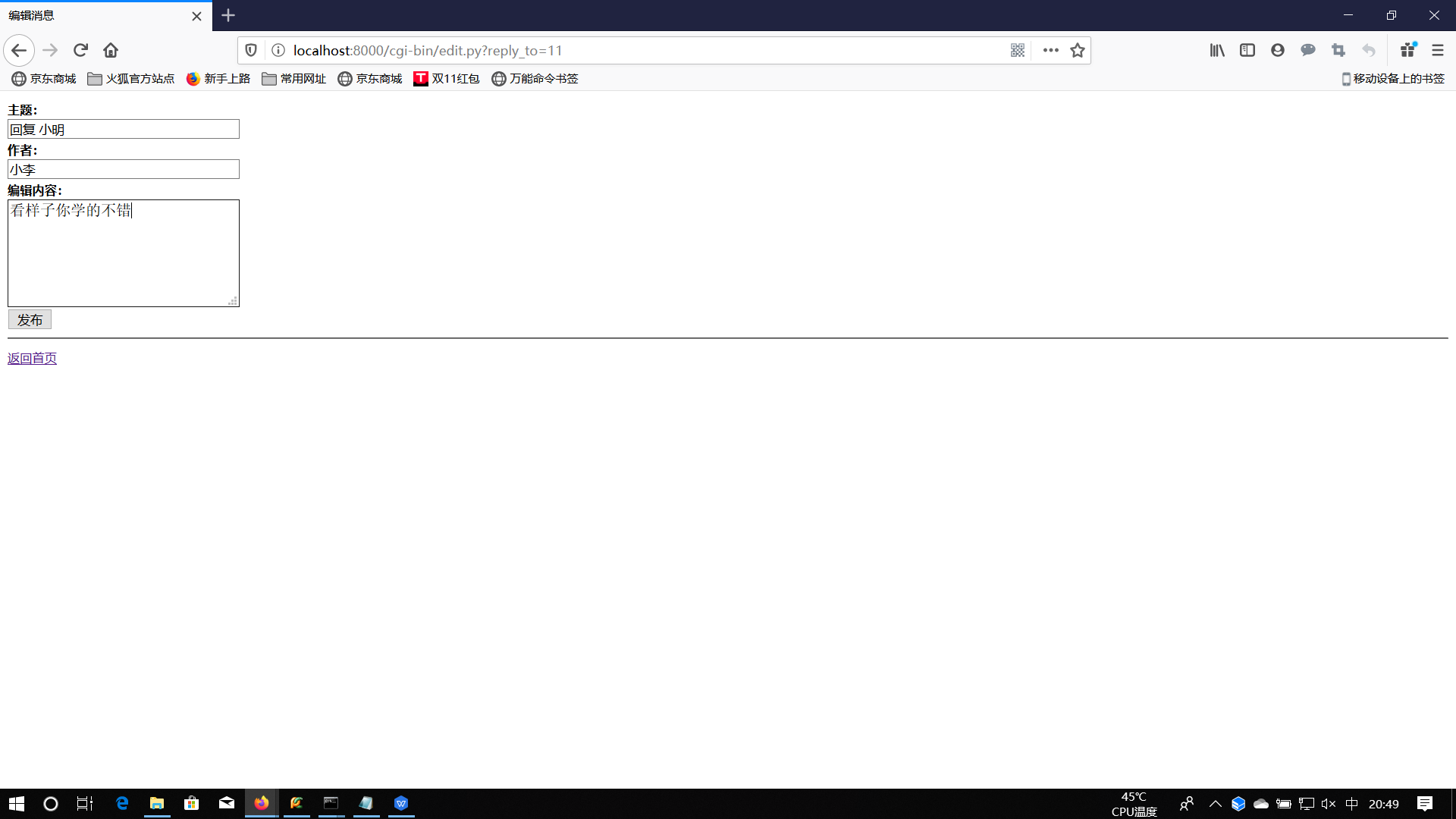


（3）消息编辑器edit.py

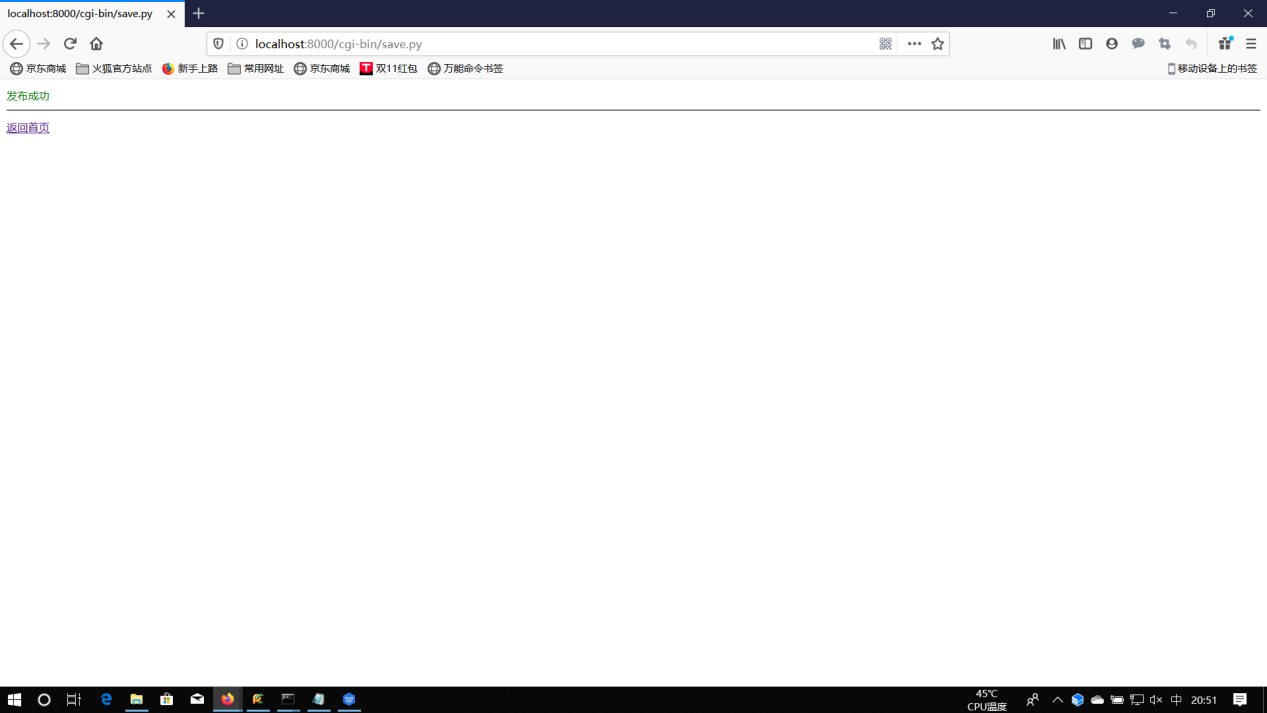
回复消息：



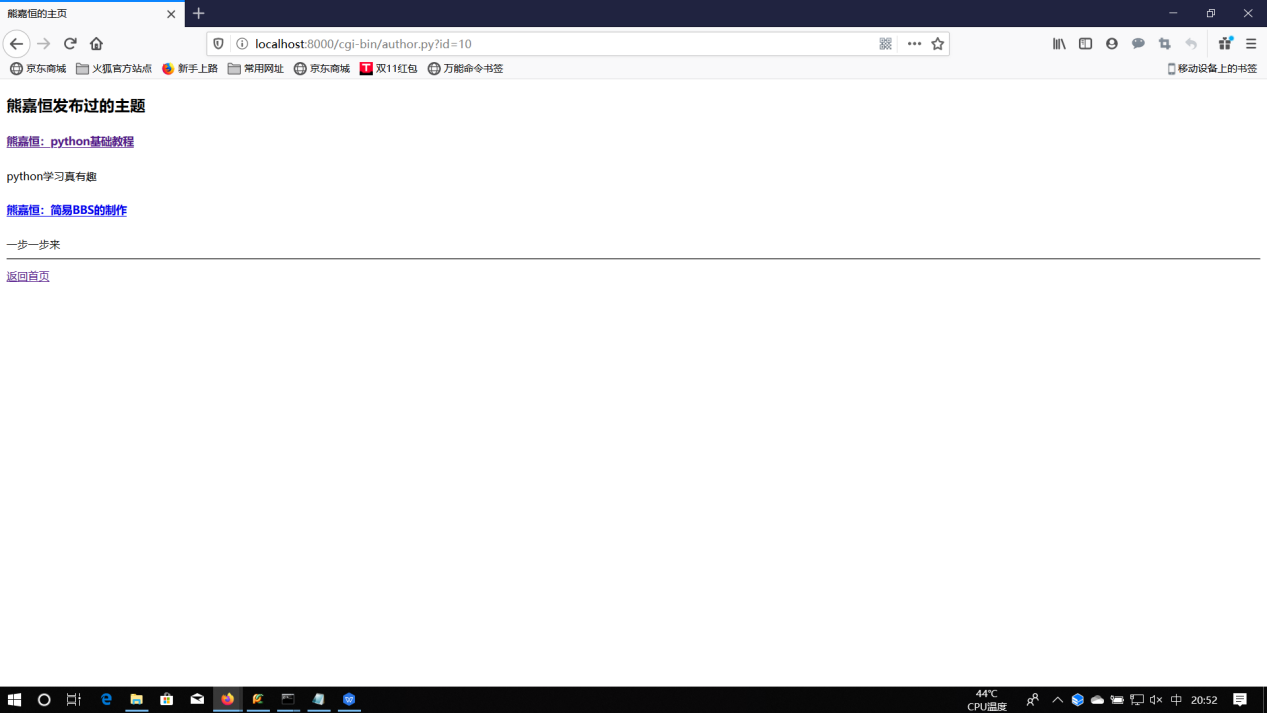
发布新消息：



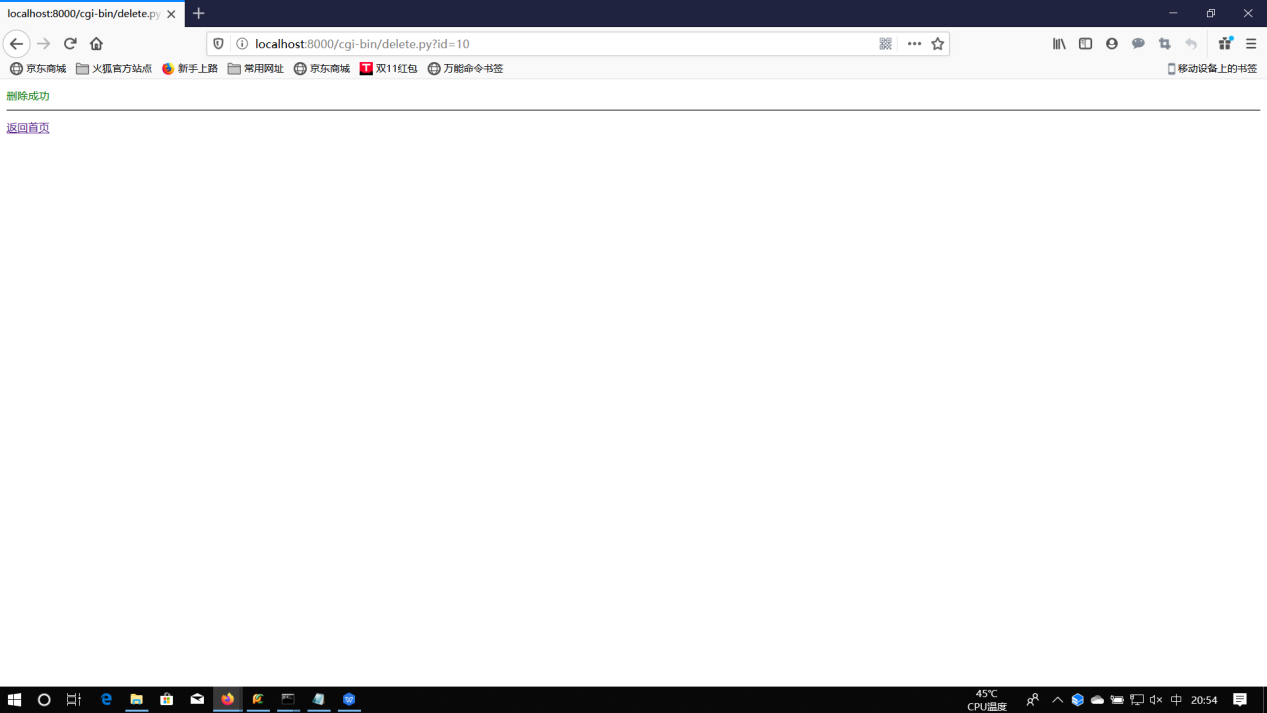
1. 保存页面save.py



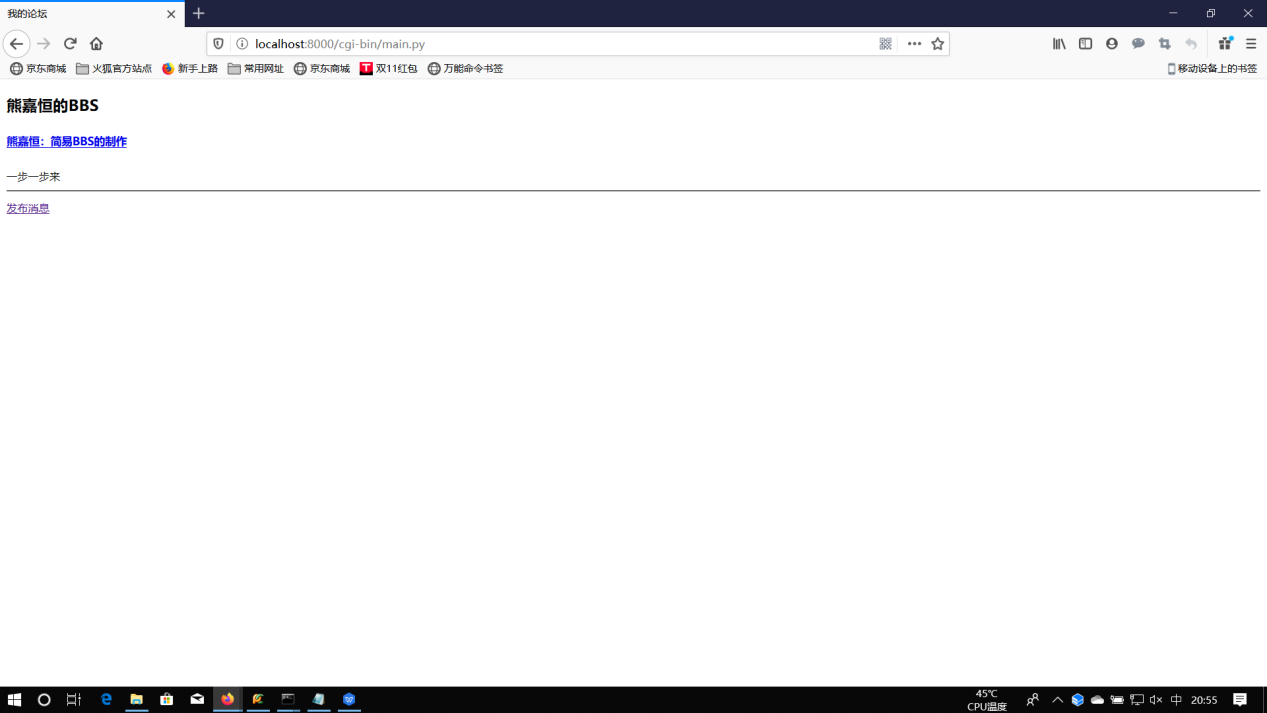
1. 个人主页author.py



1. 删除脚本delete.py



删除结果：



1. 结果分析
2. 实现程序基本功能：
3. 在公告板主页上展示已经发出的消息及其主题和作者
4. 可以编辑发布新消息，并将这些消息存储到数据库中
5. 可以回复别人发的消息，并将回复的消息缩进展示

1. 不足之处
2. 网页设计比较简陋
3. 功能较为单一
4. 改进意见
5. 可以设置点赞功能，使用户间互动更为频繁
6. 增加用户注册内容，将用户存入数据库中，从而使不同和用户对不同的主题由于不同的操作权限
7. 可以使用Web框架进行操作
8. 归纳总结

通过本项目的设计与实现，了解了CGI编程，较为熟练地掌握了python对数据库postgreSQL的操作