

第10章数据库与多界面操作

主要内容

- 邮件暂存的需求
- 数据库的连接
- 邮件暂存库表设计及数据库基本操作
- 暂存邮件的浏览与删除

===邮件暂存的需求===

- 邮件之需求.邮件暂存、查询与删除



The screenshot shows an email composition window. At the top, there are four buttons: '发送' (Send) in green, '预览' (Preview) in grey, '存草稿' (Save Draft) in yellow, and '取消' (Cancel) in yellow. The '存草稿' button is circled in red. Below the buttons, there are input fields for '收件人:' (Recipient) with a hint '输入对方手机号, 就能给他发邮件' (Enter the other party's phone number, you can email him), '主题:' (Subject), and '添加附件 (最大3G)' (Add attachment (max 3G)) with a dropdown for '从手机上传图片' (Upload image from phone). Below these is a rich text editor toolbar with icons for bold, italic, underline, text color, background color, bulleted list, numbered list, link, image, video, emoji, calendar, microphone, translate, and more. The text area below the toolbar contains the hint '尝试下写信输入@, @联系人后系统自动把此人加到收件人' (Try writing an email with @, after @ contact, the system will automatically add this person to the recipient).

===邮件暂存的需求===

- 邮件之需求.邮件暂存与删除

Tile

发送

重置

暂存

浏览

接收人

Entry

接收人

Entry

题目

Entry

邮件内容

Label

Text

第10章数据库与多界面操作

主要内容

- 邮件暂存的需求
- 数据库的连接
- 邮件暂存库表设计及数据库基本操作
- 暂存邮件的浏览与删除

===数据库的连接===

● 数据库的连接

◆ 运行环境

PyMySQL: pip3 install PyMySQL

安装MySQL数据库

创建emaildb数据库

◆ 测试连接

```
import pymysql
# 打开数据库连接
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor() # 创建一个游标对象 cursor
cursor.execute("SELECT VERSION()") # 执行 SQL 查询
data = cursor.fetchone() # 获取单条数据.
print ("Database version : %s " % data)
db.close() # 关闭数据库连接
```

第10章数据库与多界面操作

主要内容

- 邮件暂存的需求
- 数据库的连接
- 邮件暂存库表设计及数据库基本操作
- 暂存邮件的浏览与删除

==邮件暂存库表设计及数据库基本操作==

● 数据库的设计

◆ 数据库表 (H_Email)

序号	字段	中文名	字段说明	类型与长度
1	id	用户编号	PK,主键	Integer
2	sender	发送人	NULL	VARCHAR(50)
3	receiver	接收人	NULL	VARCHAR(50)
4	context	发送内容	NULL	VARCHAR(1000)
5	c_date	创建日期	NULL	VARCHAR(19)
6	title	主题	NULL	VARCHAR(50)

◆ 创建表格

```
import pymysql
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor() #一个游标对象 cursor
# 使用预处理语句创建表
sql = """CREATE TABLE `H_Email` (
  `id` int NOT NULL AUTO_INCREMENT,
  `sender` varchar(50) NULL, `receiver` varchar(50) NULL,
  `context` varchar(1000) NULL, `p_date` varchar(19) NULL,
  `title` varchar(19) NULL, PRIMARY KEY (`id`) )"""
cursor.execute(sql)
db.close()
```

● 数据库的设计

◆ 添加数据操作

```
import pymysql
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor()
sql = """INSERT INTO h_email(sender,
                             receiver, context,p_date,title)
                             VALUES('hh@163.com', 'hh@hit.edu.cn' ,
                                     ‘成功’ , ‘2019-03-09 21:30:01’,’人生’)"""
try:
    cursor.execute(sql)
    db.commit()
except:
    db.rollback()
db.close()
```



```

import pymysql
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor()
tt='2019-03-09 11:30:01'
sql="SELECT id,sender,receiver,title,context,p_date FROM h_email\
    WHERE p_date > '%s' order by id" %tt # SQL 查询语句
try:
    cursor.execute(sql)
    results = cursor.fetchall()
    for r in results:
        t_id      =r[0]
        t_sender   =r[1]
        t_receiver =r[2]
        t_title    =r[3]
        t_context  =r[4]
        t_p_date   =r[5]
        print ("id=%s,发送人=%s,接收人=%s,主题=%s,内容=%s,\
            创建时间=%s"%\ (t_id,t_sender,t_receiver,t_title,t_context,t_p_date))
except:
    print ("Error: unable to fetch data")
db.close()

```

◆ 查询数据操作

- **fetchone()**: 该方法获取下一个查询结果集。
- **fetchall()**: 接收全部的返回结果行。
- **rowcount**: 返回执行execute()方法后影响的行数。

● 数据库的设计

◆ 更新数据操作

```
import pymysql
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor()
tt='hh1@hit.edu.cn'
# SQL 更新语句
sql = "UPDATE h_email SET title = '必须成功' WHERE \
      receiver = '%s'" % tt
try:
    cursor.execute(sql)# 执行SQL语句
    db.commit()# 提交到数据库执行
except:
    db.rollback()# 发生错误时回滚
db.close()# 关闭数据库连接
```

● 数据库的设计

◆ 删除数据操作

```
import pymysql
db = pymysql.connect("localhost","root","123456","emaildb" )
cursor = db.cursor()
tt='hh1@hit.edu.cn'
# SQL 更新语句
sql = "delete from h_email WHERE receiver = '%s'" % tt
try:
    cursor.execute(sql)# 执行SQL语句
    db.commit()# 提交到数据库执行
except:
    db.rollback()# 发生错误时回滚
db.close()# 关闭数据库连接
```

第10章数据库与多界面操作

主要内容

- 邮件暂存的需求
- 数据库的连接
- 邮件暂存库表设计及数据库基本操作
- 暂存邮件增加、浏览与删除

==暂存邮件增加、浏览与删除==

- 邮件暂存与其他操作

Tile

发送

重置

暂存

浏览

接收人

Entry

接收人

Entry

题目

Entry

邮件内容

Text

=暂存邮件增加、浏览与删除 =

● 邮件暂存

◆ 暂存操作

```
def saveEmail():
    #连接数据库
    p= pymysql.connect("localhost","root","123456","emaildb" )
    dbsql="insert into h_email(sender,receiver,title,context,p_date)
          values('"+s.get()+"', '"
    dbsql+=t.get()+"', '"+r.get()+"', '"+t_show.get(0.0,tk.END)+"', '"
    dbsql+=datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')+'''"
    print(dbsql)
    p.InsertDB(dbsql)
```

绑定按钮

```
Button(frm_S, text= “暂存” , command=saveEmail,width=6,
      height=1, font=('Arial', 10)).pack(side=RIGHT)
```

==暂存邮件增加、浏览与删除 ==

- 邮件浏览界面

Tile

题目

查询

删除

序号	发送人	接收人	题目

=暂存邮件增加、浏览与删除 =

- grid布局

邮件浏览					
题目		<input type="text"/>		查询	删除
序号		发送人	接收人	题目	

=暂存邮件增加、浏览与删除 =

● grid布局

◆ 创建Toplevel（顶级窗口）

```
top = Toplevel()  
top.title('邮件暂存')
```

◆ 放置组件

```
Label(top,width=5).grid(row=0,column=0,sticky=E)  
Label(top,text="题目",width=10).grid(row=0,column=1,sticky=W)  
e1 = Entry(top,textvariable=v1,width=30)  
e1.grid(row=0,column=2,padx=1,pady=1)  
e2=Button(top,text='查询',command=lambda :search(tv,e1),width=10)  
e2.grid(row=0,column=3,padx=1,pady=1)  
e3=Button(top,text='删除',command=lambda :delrow(tv),width=10)  
e3.grid(row=0,column=4,padx=1,pady=1)  
Label(top,width=5).grid(row=0,column=5,sticky=E)
```

=暂存邮件增加、浏览与删除 =

● treeview组件

◆ 创建treeview

```
columns = ("序号", "发件人", "接收人", "题目", "发送内容", "创建时间")  
tv = ttk.Treeview(top, height=18, show="headings", columns=columns)
```

◆ 设置列宽及对齐方式

```
tv.column("序号", width=100, anchor='center')  
tv.column("发件人", width=100, anchor='center')  
tv.column("接收人", width=100, anchor='center')  
tv.column("题目", width=100, anchor='center')  
tv.column("发送内容", width=200, anchor='center')  
tv.column("创建时间", width=200, anchor='center')
```

=暂存邮件增加、浏览与删除 =

● treeview组件

◆ 显示表头

```
tv.heading("序号", text="序号") # 显示表头
tv.heading("发件人", text="发件人")
tv.heading("接收人", text="接收人")
tv.heading("题目", text="题目")
tv.heading("发送内容", text="发送内容")
tv.heading("创建时间", text="创建时间")
```

◆ 摆放

```
tv.grid(row=1, columnspan=6, padx=1, pady=1)
```

=暂存邮件增加、浏览与删除 =

● 显示全部数据

#检索所有数据

```
def searchall(tv): #tv为treeview
```

```
    #连接数据库
```

```
    p=DBOper("localhost","root","123456","emaildb" )
```

```
    #查询所有内容
```

```
    dbsql="select id,sender,receiver,title,context,p_date from h_email"
```

```
    results=p.SelDB(dbsql)
```

```
    k=0
```

```
    # 写入数据
```

```
    for row in results:
```

```
        tv.insert(", k, values=(row[0],row[1],row[2],row[3],row[4],row[5]))
```

```
        k=k+1
```

=暂存邮件增加、浏览与删除 =

● 查找数据

1.清除treeview中数据

```
def deltree(tree):
    x=tree.get_children()
    for item in x:
        tree.delete(item)
```

2. 查找数据

```
def finditem(tv,e1):
    #得到用户输入条件
    dbsql="select id,sender,receiver,title,context,p_date from h_email
        where title like '"+e1.get()+"%"
    p=DBOper("localhost","root","123456","emaildb" )
    results=p.SelDB(dbsql)
    k=0
    for row in results:
        tv.insert("", k, values=(row[0],row[1],row[2],row[3],row[4],row[5]))
        k=k+1
```

3.显示调用

```
def search(tv,e1):
    deltree(tv)
    finditem(tv,e1)
```

4.与按钮关联

```
e2=Button(top,text='查询',command=lambda :search(tv,e1),
           width=10)
e2.grid(row=0,column=3,padx=1,pady=1)
```

=暂存邮件增加、浏览与删除 =

● 删除数据

```
def delrow(tv):
    a=tkinter.messagebox.askokcancel('提示','要执行此操作吗')
    if a:
        item =tv.selection()
        item_text = tv.item(item,"values")
        dbsql="delete from h_email where id='"+item_text[0]+"'"
        print(dbsql)
        #连接数据库
        p=DBOper("localhost","root","123456","emaildb" )
        p.DelDB(dbsql)
        deltree(tv)
        searchall(tv)
    else:
        return
```

```
e3=Button(top,text='删除',command=lambda :delrow(tv),width=10)
e3.grid(row=0,column=4,padx=1,pady=1)
```

=暂存邮件增加、浏览与删除 =

● 双击指令

#双击函数

```
def treeviewClick(event):  
    print ('双击')  
    for item in tv.selection():  
        item_text = tv.item(item,"values")  
        clear()  
        s.insert(10,item_text[1])  
        t.insert(10,item_text[2])  
        r.insert(10,item_text[3])  
        t_show.insert(1.0,item_text[4])
```

双击后，将数据返回主界面，继续进行编辑

#绑定双击事件：<Double-1>

```
tv.bind('<Double-1>', treeviewClick)
```

本章小结

- 邮件暂存的需求
- 数据库的连接
- 邮件暂存库表设计及数据库基本操作
- 暂存邮件增加、浏览与删除