[Flask](http://www.ishenping.com/Search/Art/FLASK)[HTML](http://www.ishenping.com/Search/Art/HTML)[validator](http://www.ishenping.com/Search/Art/VALIDATOR)[Email](http://www.ishenping.com/Search/Art/EMAIL)[Bootstrap](http://www.ishenping.com/Search/Art/BOOTSTRAP)

**flask-wtf表单处理**

**flask-wtf必要性**

- 避免重复操作， 表单操作很多;

- 防止表单遭遇跨站请求伪造(csrf===cross-site request forgery);

**1. 为什么使用Flask-WTF？**

request对象公开了所有客户端发送的请求信息。特别是request.form可以访问POST请求提交的表单数据。

尽管Flask的request对象提供的支持足以处理web表单，但依然有许多任务会变得单调且重复。

表单的HTML代码生成和验证提交的表单数据就是两个很好的例子。

优势：

Flask-WTF扩展使得处理web表单能获得更愉快的体验。该扩展是一个封装了与框架无关的WTForms包的Flask集成。

**2. 什么是表单处理?**

在网页中，为了和用户进行信息交互总是不得不出现一些表单。

flask设计了WTForm表单库来使flask可以更加简便地管理操作表单数据。

WTForm中最重要的几个概念如下：

1). Form类，开发者自定义的表单必须继承自Form类或者其子类。

Form类最主要的功能是通过其所包含的Field类提供对表单内数据的快捷访问方式。

　　2). 各种Field类，即字段。一般而言每个Field类都对应一个input的HTML标签。

比如WTForm自带的一些Field类比如BooleanField就对应<input type="checkbox">，

SubmitField就对应<input type="submit">等等。

　　3). Validator类。这个类用于验证用户输入的数据的合法性。

比如Length验证器可以用于验证输入数据的长度，

FileAllowed验证上传文件的类型等等。

　　另外，flask为了防范csfr(cross-site request forgery)攻击，

默认在使用flask-wtf之前要求app一定要设置过secret\_key。

最简单地可以通过app.config['SECRET\_KEY'] = 'xxxx'来配置。

**3. 常见的Field类**

　　PasswordField　　 密码字段，自动将输入转化为小黑点

　　DateField　　 文本字段，格式要求为datetime.date一样

　　IntergerField　　 文本字段，格式要求是整数

　　DecimalField　　 文本字段，格式要求和decimal.Decimal一样

　　FloatField　　 文本字段，值是浮点数

　　BooleanField　　 复选框，值为True或者False

　　RadioField　　 一组单选框

　　SelectField　　 下拉列表，需要注意一下的是choices参数确定了下拉选项，

但是和HTML中的<select> 标签一样，其是一个tuple组成的列表，

可以认为每个tuple的第一项是选项的真正的值，而第二项是alias。

　　MultipleSelectField　　可选多个值的下拉列表

Validator是验证函数:

Validator是验证函数，把一个字段绑定某个验证函数之后，flask会在接收表单中的数据之前对数据做一个验证，

如果验证成功才会接收数据。验证函数Validator如下，具体的validator可能需要的参数不太一样，这里只给出

一些常用的，更多详细的用法可以参见wtforms/validators.py文件的源码，参看每一个validator类需要哪些参数：

　　\*基本上每一个validator都有message参数，指出当输入数据不符合validator要求时显示什么信息。

　　Email

验证电子邮件地址的合法性，要求正则模式是^.+@([^.@][^@]+)$

　　EqualTo

比较两个字段的值，通常用于输入两次密码等场景，可写参数fieldname，

不过注意其是一个字符串变量，指向同表单中的另一个字段的字段名

　　IPAddress

验证IPv4地址，参数默认ipv4=True,ipv6=False。如果想要验证ipv6可以

设置这两个参数反过来。

　　Length

验证输入的字符串的长度，可以有min,max两个参数指出要设置的长度下限和上限,

注意参数类型是字符串，不是INT!!

　　NumberRange

验证输入数字是否在范围内，可以有min和max两个参数指出数字上限下限，注意参数

类型是字符串，不是I数里可以设置%(min)s和%(max)s两个格式化部分，来告诉前端

这个范围到底是多少。其他validator也有这种类似的小技巧，可以参看源码。

　　Optional

无输入值时跳过同字段的其他验证函数

　　Required

必填字段

　　Regexp

用正则表达式验证值，参数regex='正则模式'

　　URL

验证URL，要求正则模式是^[a-z]+://(?P<host>[^/:]+)(?P<port>:[0-9]+)?

(?P<path>\/.\*)?$

　　AnyOf

确保值在可选值列表中。

　　NoneOf

确保值不在可选值列表中

**flask-wtf的使用流程**

# 1). 编写forms.py文件， 定义一个关于表单的类;(\*\*\*注意： 一定要有提交的按钮);

# 2). 业务逻辑文件app.py中，

- 实例化表单类; form = RegisterForm()

- 验证表单内容是否提交成功? form.validate\_on\_submit():

- 获取表单里面的内容(两种方法):

- 通过request对象获取: request.form['key值']；

- 通过form对象获取: form.key值.data

# 3). 表现逻辑文件wtf.html中:

- 导入wtf模块: {% import "bootstrap/wtf.html" as wtf %}

- 自动生成表单对应的html： {{ wtf.quick\_form(form) }}

测试表单的应用案例-注册页面

**forms.py文件，定义表单**

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, RadioField, SubmitField, SelectMultipleField, TextAreaField, SelectField

from wtforms.validators import DataRequired, Length, Email, EqualTo, Regexp

class RegisterForm(FlaskForm):

# StringField是字符串类型的数据传入

name = StringField(

# 给需要填写的信息注释，服务于客户

label='用户名',

# 验证: 用户名不能为空的

validators=[DataRequired(), ]

)

# PasswordField是传入密码的数据，会自动加密

password = PasswordField(

label='密码',

validators=[DataRequired(),

# 验证密码长度是否为6～8之间， 如果不是， 则报错;

Length(6, 16, message='密码格式不正确'), ]

)

repassword = PasswordField(

label='确认密码',

validators=[DataRequired(),

# 验证当前表单输入的内容和password这个表单输入的内容是否一致， 如果不一致， 报错;

EqualTo('password', message='密码不一致'), ]

)

email = StringField(

label='邮箱',

validators=(DataRequired(),

# 验证当前表单输入的内容是否为一个邮箱地址， 如果不是， 则报错;

Email(message='邮箱格式错误'))

)

phone = StringField(

label='手机号码',

validators=(DataRequired(),

# 验证当前表单输入的电话号码是否符合首位为1，由11位数字组成的正则表达式， 如果不是， 则报错;

Regexp(r'1\d{10}', message='电话号码格式错误'))

)

# 可以实现单选按钮， 但是不美观

# gender=RadioField(

# label='性别',

# 关键字coerce表明我们把表单中的数据强制规定为int

# coerce=int,

# 且显示的是男女，存入的信息为数字1，2

# choices=[(1,'男'),(2,'女')]

# )

# 可以实现单选按钮， 而且美观

gender = SelectField(

label='性别',

# 关键字coerce表明我们把表单中的数据强制规定为int

coerce=int,

# 且显示的是男女，存入的信息为数字1，2

choices=[(1, '男'), (2, '女')]

)

#多选按钮的实现，需按住Ctrl键进行多选

tech=SelectMultipleField(

label='擅长领域',

# 关键字coerce表明我们把表单中的数据强制规定为int

coerce=int,

#显示的是文本信息，保存的是数据信息

choices=[(1, 'python'), (2, 'php'), (3, 'java'), (4, 'linux'), (5, 'web'), (6, 'c++')]

)

#提交按钮，必须提交才能有数据交互

submit = SubmitField(label="注册")

**在app.py中实现产生登录**

from flask import Flask, render\_template, redirect, request, url\_for, flash, session

from functools import wraps

from flask\_bootstrap import Bootstrap

from forms import RegisterForm

users = [

{

'username': 'root',

'password': 'root',

'email':'hello@qq.com',

'phone':'12345678901',

'gender':1,

'tech':[1,2,3,4]

},

{

'username': 'westos',

'password': 'root',

'email':'hello@qq.com',

'phone':'12345678901',

'gender':1,

'tech':[1,2,3,4]

}

]

app = Flask(\_\_name\_\_)

# 设置secret值，保护信息不被泄露

app.config['SECRET\_KEY'] = 'aasswde'

#实例化一个Bootstrap对象，否则form表单里的信息不能在html里被导入

bootstrap=Bootstrap(app)

@app.route('/wtf/',methods=['GET','POST'])

def wtf():

# 1. 实例化表单， 将来传递到前端， 进行生成对应的html;

form=RegisterForm()

# 2. 验证表单内容是否提交成功

if form.validate\_on\_submit():

# 1.用get方法获取对应的数据信息

username=request.form.get('name')

# 2.直接在form表单中索引

password=request.form['password']

# 3.通过data取得相应的信息

email=form.email.data

phone=form.phone.data

gender=form.gender.data

tech=form.tech.data

# 当所有的信息遍历结束， 都没有发现注册的用户存在，

# 则将注册的用户添加到服务器， 并跳转登录界面;

for user in users:

if user['username']==username:

flash('用户%s已经存在'%username)

return redirect(url\_for('wtf'))

else:

users.append(dict(username=username,password=password,email=email,

phone=phone,gender=gender,tech=tech))

flash('用户%s注册成功'%username)

return redirect(url\_for('login'))

return render\_template('wtf.html',form=form)

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**wtf.html中实现前端的展示以及填写数据**

base.html与上一章中一样，就是一个基模板

{% extends 'base.html' %}

{#导入模块， 并重命名为wtf#}

{% import 'bootstrap/wtf.html' as wtf %}

{% block title %}

表单注册页面测试页

{% endblock %}

{% block content %}

<div class="container" style="margin-top: 50px">

<div class="col-lg-8 col-lg-offset-2">

<h1>注册-wtf测试</h1>

{# 根据后台传递过来的form对象生成对应的html代码; #}

{{ wtf.quick\_form(form) }}

</div>

</div>

{% endblock %}



**list.html中可看数据是否以及已经提交与保存**

{% extends 'base.html' %}

{% block title %}

用户信息

{% endblock %}

{% block content %}

{# 设置页面为十二格模式 #}

<div class="container">

{# 设置居中，占八格，开头留两个空格#}

<div class="col-lg-8 col-lg-offset-2">

<h1 style="color:blueviolet">用户信息显示</h1>

<br/>

<br/>

<table class="table table-hover">

<tr>

<td>用户名</td>

<td>密码</td>

<td>邮箱</td>

<td>手机号码</td>

<td>性别</td>

<td>擅长领域</td>

<td>删除</td>

</tr>

{% for user in users %}

<tr>

{# user是一个字典， 获取字典的key值， user.keyname #}

<td>{{ user.username }}</td>

<td>{{ user.password }}</td>

<td>{{ user.email }}</td>

<td>{{ user.phone }}</td>

<td>{{ user.gender }}</td>

<td>{{ user.tech }}</td>

{# 寻找删除的按钮 #}

<td>

<a href="{{ url\_for('delete',username=user.username) }}" class="btn btn-default btn-danger"

role="button">

删除

</a>

</td>

</tr>

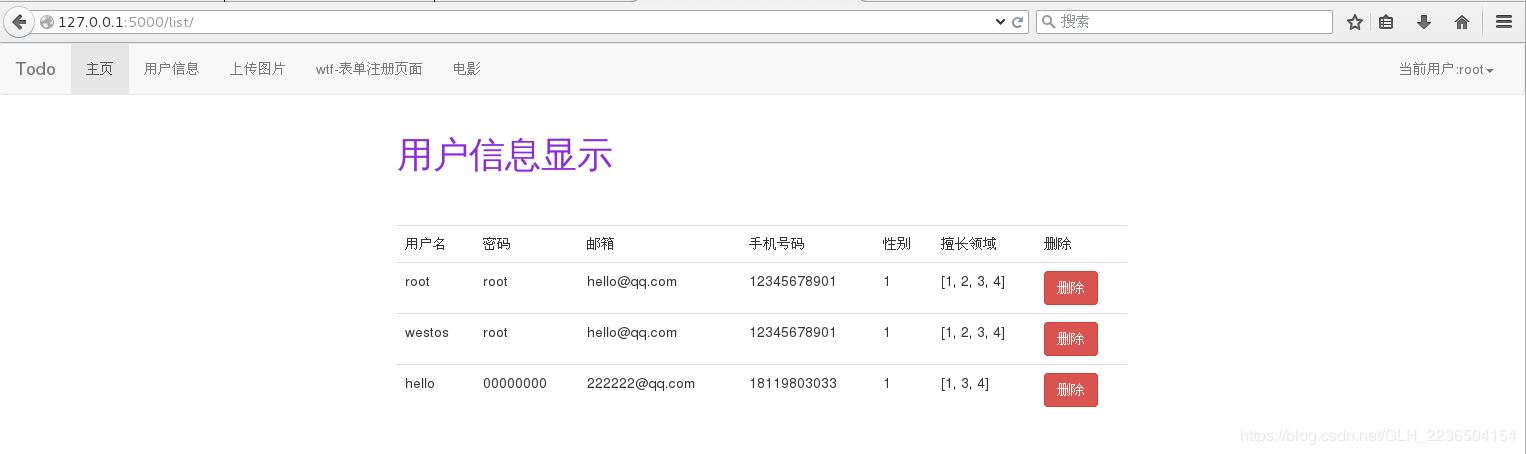
{% endfor %}

</table>

</div>

</div>

{% endblock %}



Flask利用forms.py实现集成邮件发送

**发送邮件的时候需要设置什么?**

- 发件人账户

- 密码

- 收件人

- 邮件标题

- 邮件正文

- QQ邮件服务器的域名或者IP;

**forms.py中定义表单**

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, RadioField, SubmitField, SelectMultipleField, TextAreaField, SelectField

from wtforms.validators import DataRequired, Length, Email, EqualTo, Regexp

class MailForm(FlaskForm):

# 接收邮件的地址

toEmail=TextAreaField(

label='接受的邮件地址',

validators=[DataRequired()],

# 表单的详细描述， 一般不写

description='每个邮箱地址之间用逗号分隔'

)

toFilename=SelectField(

label='邮件内容的类型',

# "company", 是传递给后台的， "公司简介"是在前端显示的;

choices =[("company", "公司简介"), ("course", "课程简介")]

)

submit = SubmitField(

label="发送"

)

**app.py中**

import os

from flask import Flask, render\_template, redirect, request, url\_for, flash, session

from functools import wraps

from flask\_bootstrap import Bootstrap

from forms import RegisterForm,MailForm

from flask\_mail import Mail,Message

app = Flask(\_\_name\_\_)

# 设置secret值，保护信息不被泄露

app.config['SECRET\_KEY'] = 'aasswde'

#实例化一个Bootstrap对象，否则form表单里的信息不能在html里被导入

bootstrap=Bootstrap(app)

# 配置发送邮件的相关信息;

# 指定邮件服务器的域名或者IP

app.config['MAIL\_SERVER'] = 'smtp.qq.com'

# 指定端口， 默认25， 但qq邮箱默认为 端口号465或587；

app.config['MAIL\_PORT'] = 465

app.config['MAIL\_USE\_SSL'] = True

app.config['MAIL\_USERNAME'] = '2236504154'

# 此处的密码并非邮箱登录密码， 而是开启pop3的密码

app.config['MAIL\_PASSWORD'] = 'ytrlrowsjqdwdjhj'

def send\_mail(to,subject,template,\*\*kwargs):

mail=Mail(app)

# app.app\_context(): 将之前Flask创建的app作为参数传入AppContext类中,

# 用于存储当前app的相关信息;称为应用上下文

with app.app\_context():

msg=Message(subject=subject,

sender='2236504154@qq.com',

recipients=to,

html=render\_template(template+'.html',\*\*kwargs))

mail.send(msg)

@app.route('/mail/',methods=['GET','POST'])

def mail():

# 实例化表单

form =MailForm()

if form.validate\_on\_submit():

# 获取用户提交的数据;

# 将一堆邮箱地址分开

toEmail=form.toEmail.data.split(',')

toFilename=form.toFilename.data

try:

send\_mail(toEmail,subject='测试',template=toFilename)

except Exception as e:

flash('邮件发送失败:',e)

else:

flash('邮件发送成功')

return redirect(url\_for('mail'))

return render\_template('mail.html',form=form)

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**mail.html中导入forms表单中写入的**

{% extends 'base.html' %}

{% import 'bootstrap/wtf.html' as wtf %}

{% block title %}

邮件发送

{% endblock %}

{% block content %}

<div class="container">

<div class="col-lg-8 col-lg-offset-2">

<h1>发送邮件测试案例</h1>

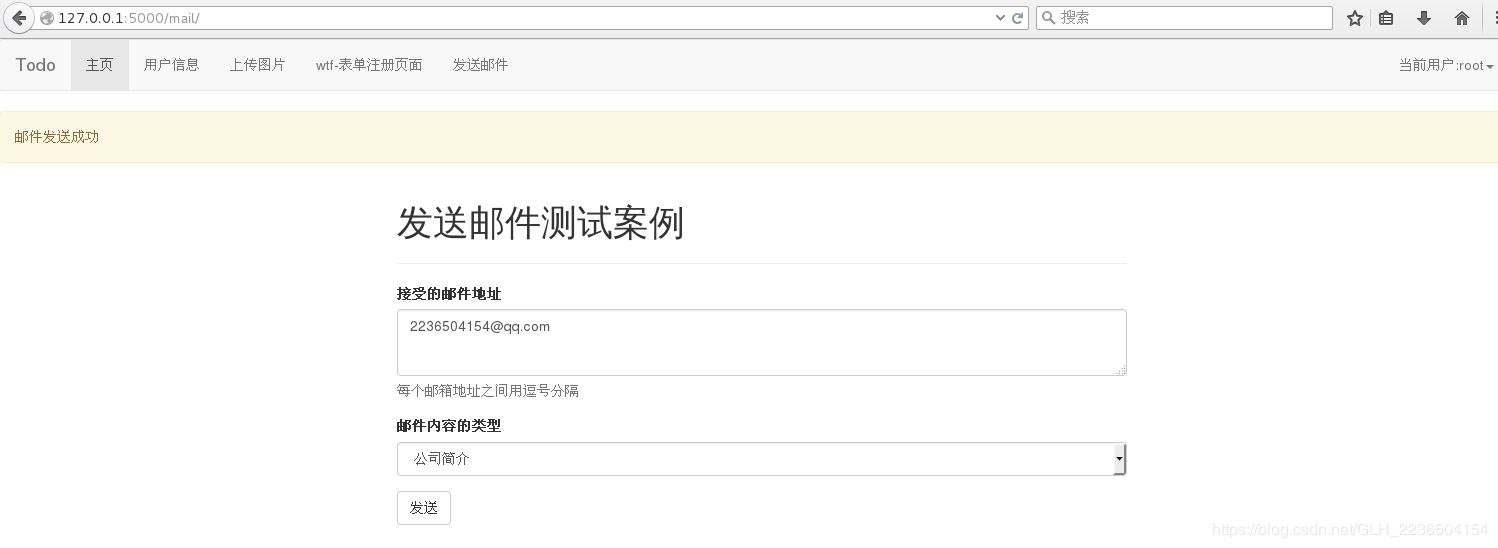
<hr/>

{{ wtf.quick\_form(form) }}

</div>

</div>

{% endblock %}



**关注神评公众号**