Flask表单操作

Form表单，在Web应用中无处不在。比如: 用户登录表单, 用户注册表单。

- 所有的表单项都有共性，比如有文字输入框，单选框，密码输入框等；
- 表单的验证也有共性，比如有非空验证，长度限制，类型验证等。

如果有个框架，能把这些共性抽象出来，那就能大量简化我们的工作。Python的`WTForms`就提供了这些功能，这里我们就要结合Flask的`WTForms`扩展，`Flask-WTF`，来介绍如何在Web应用中制作表单。

安装和启用

```
pip install Flask-WTF
```

一个简单的表单

表单文件

`WTForms`让我们在后端代码中定义表单类，并列出表单的字段和相应的验证规则。现在让我们先定义一个`MyForm`类:

```
from flask_wtf import Form
from wtforms import StringField
from wtforms.validators import DataRequired

class LoginForm(FlaskForm):
    email = StringField(u'邮箱', validators=[
                DataRequired(message= u'邮箱不能为空'), Length(1, 64),
                Email(message= u'请输入有效的邮箱地址，比如：username@domain.com')])
    password = PasswordField(u'密码',
                  validators=[Required(message= u'密码不能为空')])
    submit = SubmitField(u'登录')
```

常见的表单域类型

表单域类型	描述
StringField	文本框
TextAreaField	多行文本框
PasswordField	密码输入框
HiddenField	隐藏文本框
DateField	接收给定格式datetime.date型的文本框
DateTimeField	接收给定格式datetime.datetime的文本框
IntegerField	接收整型的文本框
DecimalField	接收decimal.Decimal型的文本框
FloadField	接收浮点型的文本框
BooleanField	带有True和False的复选框
RadioField	一组单选框
SelectField	下拉选择框
SelectMultipleField	下拉多选框
FileField	文件上传框
SubmitField	表单提交按钮
FormField	将一个表单作为表单域嵌入到容器表单中
FieldList	给定类型的表单域列表

常见验证规则

验证规则	说明
DataRequired	验证必填项
Email	验证邮件地址格式
EqualTo	验证必须同另一个字段值相同，它需传入另一个字段的名称"fieldname"
Length	验证输入字符串长度，它有两个参数："min"最小长度，"max"最大长度，缺省的话就不检查
NumberRange	验证输入数值的范围，它有两个参数："min"最小值，"max"最大值，缺省的话就不检查
URL	验证URL格式
IPAddress	验证IP地址格式，默认IPV4，你可以传入"ipv6=True"来验证IPV6地址
MacAddress	验证Mac地址格式
AnyOf	传入一个列表作为参数，验证是否匹配列表中的任一值
NoneOf	传入一个列表作为参数，验证是否与列表中的所有值都不同
Regexp	正则表达式验证，需传入一个正则表达式，它还有一个flags参数，如果你传入"re.IGNORECASE"，就会忽略大小写

视图函数文件

```
@app.route('/login/', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        email = form.email.data
        ...
    return render_template('login.html', form=form)
```

验证数据

点击了表单上的提交按钮时，`form.validate_on_submit()`判断会做下面两件事情：

- 通过`is_submitted()`通过判断HTTP方法来确认是否提交了表单
- 通过`WTForms`提供的`validate()`来验证表单数据

  当validate()验证未通过时，会在表单字段下面显示我们传进去的错误提示（例如message= u'邮箱不能为空'）。

自定义验证

你可以在表单类创建自定义的验证函数，一个简单的例子：

```
def validate_username(self, field):
    # field.data是用户输入的数据。
    if field.data  == 'admin':
        # ValidationError从wtforms导入，用来向用户显示错误信息，
        # 验证函数的名称由validate_fieldname组成。
        raise ValidationError(u'超级管理员已被注册，换一个吧。')
```

也可以在这里对用户数据进行预处理：

```
# 这个函数对用户输入的网址进行处理（字段名为website）。
def validate_website(self, field):
    if field.data[:4] != "http":
        field.data = "http://" + field.data
```

获取数据

验证通过后，

- 使用`form.email.data`来获得数据，

```
form.email.data
```

- `WTForms`提供的静态方法.data返回一个以字段名（field name）和字段值（field value）作为键值对的字典。

```
form.data['email']
```

前端文件

- `form.hidden_tag()`会生成一个隐藏的
  标签，其中会渲染任何隐藏的字段，最主要的是`CSRF`字段。
- `CSRF`（Cross-Site Request Forgery跨站请求伪造）是一种通过伪装来自受信任用户的请求，来发送恶意攻击的方法，`WTForms`默认开启`CSRF`保护。

```
<form method="POST" action="/login/">
    {{ form.hidden_tag() }}
    {{ form.user.label }}: {{ form.user(size=20) }}
    <input type="submit" value="Submit">
</form>
```