浙江理工大学
ZHEJIANG SCI-TECH UNIVERSITY

表单

《Web应用开发》 / 任课教师：罗志一

计算机科学与技术学院

表单

- 在Web程序中，表单是和用户交互最常见的方式之一。用户注册、登录、撰写文章、编辑设置，无一不用到表单。不过表单的处理却并不简单。我们不仅要创建表单，验证用户输入的内容，向用户显示错误提示，还要获取并保存数据。幸运的是，强大的WTForms可以帮我们解决这些问题。
- WTForms是一个使用Python编写的表单库，它使得表单的定义、验证（服务器端）和处理变得非常轻松。

概览 Overview

- HTML表单
- 使用Flask-WTF处理表单
- 处理表单数据

HTML表单

HTML表单

- 在HTML中通过<form>标签创建表单，表单中的字段使用<input>标签定义。

```
<form method="post">
    <label for="username">Username</label><br>
    <input type="text" name="username" placeholder="Héctor Rivera"><br>
    <label for="password">Password</label><br>
    <input type="password" name="password" placeholder="19001130"><br>
    <input id="remember" name="remember" type="checkbox" checked>
    <label for="remember"><small>Remember me</small></label><br>
    <input type="submit" name="submit" value="Log in">
</form>
```

HTML表单

- 在HTML中通过<form>标签创建表单，表单中的字段使用<input>标签定义。

```
<form method="post">
    <label for="username">Username</label><br>
    <input type="text" name="username" placeholder="Héctor Rivera"><br>
    <label for="password">Password</label><br>
    <input type="password" name="password" placeholder="19001130"><br>
    <input id="remember" name="remember" type="checkbox" checked>
    <label for="remember"><small>Remember me</small></label><br>
    <input type="submit" name="submit" value="Log in">
</form>
```

<input> 标签表示各种输入字段
<label>标签用来定义字段的标签文字

HTML表单

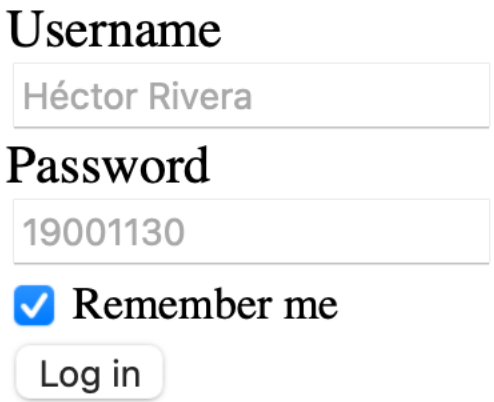
- 在HTML中通过<form>标签创建表单，表单中的字段使用<input>标签定义。

```
<form method="post">
    <label for="username">Username</label><br>
    <input type="text" name="username" placeholder="Héctor Rivera"><br>
    <label for="password">Password</label><br>
    <input type="password" name="password" placeholder="19001130"><br>
    <input id="remember" name="remember" type="checkbox" checked>
    <label for="remember"><small>Remember me</small></label><br>
    <input type="submit" name="submit" value="Log in">
</form>
```

<input> 标签表示各种输入字段
<label>标签用来定义字段的标签文字

HTML表单

- 在HTML中通过<form>标签创建表单，表单中的字段使用<input>标签定义。



关于HTML表单的具体定义和用法可以访问 https://www.w3.org/TR/html401/interact/forms.html 查看。

HTML表单

- WTForms支持在Python中使用类定义表单，然后直接通过类定义生成对应的HTML代码，这种方式更加方便，而且使表单更易于重用。

使用Flask-WTF处理表单

Flask-WTF扩展

- 扩展Flask-WTF集成了WTForms，使用它可以在Flask中更方便地使用WTForms。
- Flask-WTF将表单数据解析、CSRF保护、文件上传等功能与Flask集成，另外还附加了reCAPTCHA支持。

定义WTForms表单类

- 当使用WTForms创建表单时，表单由Python类表示，这个类继承从WTForms导入的Form基类。一个表单由若干个输入字段组成，这些字段分别用表单类的类属性来表示（字段即Field）。
- 接下来，我们定义一个LoginForm类，最终会生成我们在前面定义的HTML表单。

```
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
    remember = BooleanField('Remember me')
    submit = SubmitField('Log in')
```

定义WTForms表单类

- 每个字段属性通过实例化WTForms提供的字段类表示。字段属性的名称将作为对应HTML <input>元素的name属性及id属性值。
- 这里的LoginForm表单类中定义了四个字段：文本字段StringField、密码字段PasswordField，勾选框字段BooleanField和提交按钮字段SubmitField。

```
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
    remember = BooleanField('Remember me')
    submit = SubmitField('Log in')
```

定义WTForms表单类

```
<form method="post">
    <label for="username">Username</label><br>
    <input type="text" name="username" placeholder="Héctor Rivera"><br>
    <label for="password">Password</label><br>
    <input type="password" name="password" placeholder="19001130"><br>
    <input id="remember" name="remember" type="checkbox" checked>
    <label for="remember"><small>Remember me</small></label><br>
    <input type="submit" name="submit" value="Log in">
</form>
```

```
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
    remember = BooleanField('Remember me')
    submit = SubmitField('Log in')
```

定义WTForms表单类

- 字段类从wtforms包导入，常用的WTForms字段如下表所示。

字段类	说　明	对应的HTML表示
BooleanField	复选框，值会被处理为True或False	<input type="checkbox">
DateField	文本字段，值会被处理为 datetime.date 对象	<input type="text">
DateTimeField	文本字段，值会被处理为 datetime.datetime 对象	<input type="text">
FileField	文件上传字段	<input type="file">
FloatField	浮点数字段，值会被处理为整型	<input type="text">
RadioField	一组单选按钮	<input type="radio">
SelectField	下拉列表	<select><option></option></select>
SubmitField	提交按钮	<input type="submit">
StringField	文本字段	<input type="text">
TextAreaField	多行文本字段	<textarea></textarea>

定义WTForms表单类

- 通过实例化字段传入的参数，我们可以对字段进行设置，字段类构造方法接收的常用参数如下。

参　数	说　明
label	字段标签<label>的值，也就是渲染后显示在输入字段前的文字
render_kw	一个字典，用来设置对应的HTML <input>标签的属性，比如传入{‘placeholder’: ‘Your Name’}，渲染后的HTML代码会将<input>标签的placeholder属性设为 Your Name
validators	一个列表，包含一系列验证器，会在表单提交后被逐一调用验证表单数据
default	字符串或可调用对象，用来为表单字段设置默认值

定义WTForms表单类

- 在WTForms中，验证器(validators)是一系列用于验证字段数据的类，我们在实例化字段类时使用validators关键字来指定附加的验证器列表。验证器从wtforms.validators模块中导入，常用的验证器见下表。
- 在实例化验证类时，message参数用来传入自定义错误消息，如果没有设置则使用内置的英文错误消息。

验证器	说　明
DataRequired(message=None)	验证数据是否有效
Email(message=None)	验证Email地址
EqualTo(fieldname, message=None)	验证两个字段值是否相同
InputRequired(message=None)	验证是否有数据

定义WTForms表单类

（接上表）

验证器	说　明
Length(min=-1, max=-1, message=None)	验证输入值长度是否在给定范围内
NumberRange(min=-1, max=-1, message=None)	验证输入数字是否在给定范围内
Optional(strip_whitespace=None)	允许输入值为空，并跳过其他验证
URL(required_tld=True, message=None)	验证URL
Regexp(regex, flags=0, message=None)	使用正则表达式验证输入值
AnyOf(values, message=None, values_formatter=None)	确保输入值在可选值列表中
NoneOf(values, message=None, values_formatter=None)	确保输入值不在可选值列表中

定义WTForms表单类

- 使用Flask-WTF定义表单时，表单类要继承Flask-WTF提供的FlaskForm类。

```
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
    remember = BooleanField('Remember me')
    submit = SubmitField('Log in')
```

定义WTForms表单类

- 在username和password字段里，我们都使用了DataRequired验证器，用来验证输入的数据是否有效。另外，password字段里还添加了一个Length验证器，用来验证输入的数据长度是否在给定的范围内。验证器的第一个参数一般为错误提示消息，我们可以使用message关键字传递参数，通过传入自定义错误信息来覆盖内置消息。

```
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Length

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
    remember = BooleanField('Remember me')
    submit = SubmitField('Log in')
```

输出HTML代码

- 以使用WTForms创建的LoginForm为例，实例化表单，然后将实例属性转换成字符串或可以获取表单字段对应的HTML代码
- 字段<label>元素的HTML代码则可以通过“form.字段名.label”的形式获取

```
>>> form = LoginForm()
>>> form.username()
u'<input id="username" name="username" type="text" value="">'
>>> form.submit()
u'<input id="submit" name="submit" type="submit" value="Submit">'

>>> form.username.label()
u'<label for="username">Username</label>'
>>> form.submit.label()
u'<label for="submit">Submit</label>'
```

输出HTML代码

- 在创建HTML表单时，我们经常会需要使用HTML <input>元素的其他属性来对字段进行设置。比如，添加class属性设置对应的CSS类为字段添加样式；添加placeholder属性设置占位文本。默认情况下，WTForms输出的字段HTML代码只会包含id和name属性，属性值均为表单类中对应的字段属性名称。如果要添加额外的属性，通常有两种方法：
  - 1. 使用render_kw属性，比如下面为username字段使用render_kw设置了placeholder HTML属性

```
username = StringField('Username', render_kw={'placeholder': 'Your Username'})
```

```
<input type="text" id="username" name="username" placeholder="Your Username">
```

输出HTML代码

- 在创建HTML表单时，我们经常会需要使用HTML <input>元素的其他属性来对字段进行设置。比如，添加class属性设置对应的CSS类为字段添加样式；添加placeholder属性设置占位文本。默认情况下，WTForms输出的字段HTML代码只会包含id和name属性，属性值均为表单类中对应的字段属性名称。如果要添加额外的属性，通常有两种方法：
  - 2．在调用字段时，通过添加括号使用关键字参数的形式也可以传入字段额外的HTML属性

```
>>> form.username(style='width: 200px;', class_='bar')
u'<input class="bar" id="username" name="username" style="width: 200px;" type="text">'
```

在模板中渲染表单

- 为了能够在模板中渲染表单，我们需要把表单类实例传入数据。首先在视图函数里实例化表单类LoginForm，然后在render_template()函数中使用关键字参数form将表单实例传入模板。

```
@app.route('/basic')
def basic():
    form = LoginForm()
    return render_template('basic.html', form=form)
```

```
<form method="post">
    {{ form.csrf_token }}
    {{ form.username.label }}<br> {{ form.username }}<br>
    {{ form.password.label }}<br> {{ form.password }}<br>
    {{ form.remember }}{{ form.remember.label }}<br> {{ form.submit }}<br>
</form>
```

在模板中渲染表单

- 调用form.csrf_token属性渲染Flask-WTF为表单类自动创建CSRF令牌字段。form.csrf_token字段包含了自动生成的CSRF令牌值，在提交表单后会自动被验证，为了确保表单通过验证，我们必须在表单中手动渲染这个字段。
- 渲染后得到的HTML代码如下

```
<form method="post">
    <input id="csrf_token" name="csrf_token" type="hidden" value="E9...nrZAyI5H3uT1BWOtc">
    <label for="username">Username</label><br>
    <input id="username" name="username" required type="text" value=""><br>
    <label for="password">Password</label><br>
    <input id="password" maxlength="128" minlength="8" name="password" required type="password" value=""><br>
    <input id="remember" name="remember" type="checkbox" value="y"><label for="remember">Remember me</label><br>

    <input id="submit" name="submit" type="submit" value="Log in"><br>
</form>
```

处理表单数据

处理表单数据

- 解析请求，获取表单数据
- 对数据进行必要的转换，比如将勾选框的值转换成Python的布尔值
- 验证数据是否符合要求，同时验证CSRF令牌
- 如果验证未通过则需要生成错误信息，并在模板中显式错误消息
- 如果通过验证，就把数据保存到数据库或做进一步处理

提交表单

- 在HTML中，当<form>标签声明的表单中类型为submit的提交字段被单击时，就会创建一个提交表单的HTTP请求，请求中包含表单中各个字段的数据。表单的提交行为主要由三个属性控制。

属　性	默认值	说　明
action	当前URL，即页面对应的URL	表单提交时发送请求的目标URL
method	get	提交表单的HTTP请求方法，目前仅支持使用GET和POST方法
enctype	application/x-www-form-urlencoded	表单数据的编码类型，当表单中包含文件上传字段时，需要设为multipart/form-data，还可以设为纯文本类型text/plain

验证表单数据

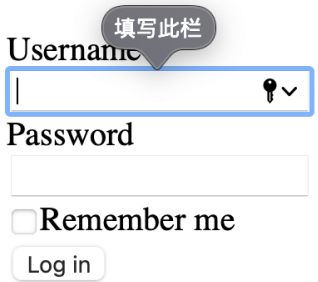
- 表单数据的验证是Web表单中最重要的主题之一，接下来我们会学习如何使用Flask-WTF验证并获取表单数据。

客户端验证

- 客户端验证是指在客户端对用户的输入值进行验证。比如，使用HTML5内置的验证属性即可实现基本的客户端验证（type、required、min、max、accept等）。比如，下面的username字段添加了required标志：

```
<input type="text" name="username" required>
```

如果用户没有输入内容而按下提交按钮，会弹出浏览器内置的错误提示。



服务器端验证

- 服务器端验证是指用户把输入的数据提交到服务器端，在服务器端对数据进行验证。如果验证出错，就在返回的响应中加入错误信息。用户修改后再次提交表单，直到通过验证。我们在Flask程序中使用WTForms实现的就是服务器端验证。

WTForms验证机制

- WTForms验证表单字段的方式是在实例化表单类时传入表单数据，然后对表单实例调用validate()方法。这会逐个对字段调用字段实例化时定义的验证器，返回表示验证结果的布尔值。如果验证失败，就把错误消息存储到表单实例的errors属性对应的字典中。

```
from wtforms import Form, StringField, PasswordField
from wtforms.validators import DataRequired, Length

class LoginForm(Form):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])
```

WTForms验证机制

```
from wtforms import Form, StringField, PasswordField
from wtforms.validators import DataRequired, Length

class LoginForm(Form):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired(), Length(8, 128)])

>>> form = LoginForm(username='', password='123')
>>> form.data  # 表单数据字典
{'username': '', 'password': '123'}
>>> form.validate()
False
>>> form.errors  # 错误消息字典
{'username': [u'This field is required.'], 'password': {u'Field must be between 8 and 1
28 characters long.'}}
>>> form2 =  LoginForm(username='jack', password='12345678')
>>> form2.validate()
True
>>> form2.errors
[]
```

WTForms验证机制

- 因为我们的表单使用POST方法提交，如果单纯使用WTForms，我们在实例化表单时需要首先把request.form传入表单类，而使用Flask-WTF时，表单类继承的FlaskForm基类默认会从request.form获取表单数据，所以不需要手动传入。
- 使用POST方法提交的表单，其数据会被Flask解析为一个字典，可以通过请求对象的form属性获取（request.form）；使用GET方法提交的表单的数据同样会被解析为字典，不过要通过请求对象的args属性获取（request.args）。