一、原生表单

form1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form action="{{ url_for('form') }}" method="post">
    <p>用户名:  <input type="text" name="username"></p>
    <p>密码:  <input type="password" name="userpass"></p>
    <p><input type="submit" value="submit"></p>
</form>
</body>
</html>
```

manage.py

```
#原生表单
@app.route('/form/',methods=['GET','POST'])
def form():
    if request.method == 'GET':
        return render_template('form1.html')
    if request.method == 'POST':
        print(request.form)
        return '接收表单'
```

二、flask-wtf扩展库

表单扩展库 提供了csrf，表单校验等功能

安装:

pip3 install flask-wtf

三、常见字段类型 和 验证器

(1) 字段类型

字段类型	字段说明
StringField	普通文本字段
SubmitField	提交按钮
PasswordField	密码输入框
HiddleField	隐藏域
TextAreaField	多行文本域
DateField	日期字段
DateTimeField	日期时间字段
IntegerFIeld	整形字段
FloatField	浮点字段
BooleanField	bool类型字段
RadioField	单选
SelectField	下拉
FileField	文件上传

(2) 常见验证器

验证	说明
DateRequired	必填
Email	邮箱地址
IPAddress	IP地址
Length	内容的长度 有max 和min
NumberRange	值的范围 min和max
EqualTo	验证俩个字段是否相同
URL	验证URL地址
Regexp	正则匹配

注册实例(测试版)

manage.py

```
from flask import Flask,request,render_template
from flask_script import Manager
from flask_wtf import FlaskForm #导入表单基类
from wtforms import StringField,PasswordField,SubmitField
from wtforms.validators import DataRequired,Length,EqualTo,Email

app = Flask(__name__)
app.config['SECRET_KEY'] = 'SECRET_KEY'
maanger = Manager(app)

#自定义表单注册类
class Register(FlaskForm):
    #username为当前标签的name值 用户名为到那个前标签展示的左侧的label标签 点击用户名 出发当前标签选中节点
    username = StringField('用户名',validators=[DataRequired('请输入用户名'),Length(min=6,max=12,message='用户名长度范围在6~12位之间')])
    userpass = PasswordField('密码',validators=[DataRequired('请输入密码'),Length(min=6,max=12,message='密码长度范围在6~12位之间')])
    confirm = PasswordField('确认密码',validators=[DataRequired('请输入确认密码'),EqualTo('userpass',message='密码和确认密码不一致')])
    email = StringField('邮箱',validators=[DataRequired('请输入邮箱地址'),Email(message='请输入正确的邮箱')])
```

```
    submit = SubmitField('注册')

#注册
@app.route('/register/',methods=['GET','POST'])
def register():
    form = Register() #实例化表单类
    #这个方法是实现表单校验功能的 csrf，数据正确性 都通过了 则为真 否则为假
    if form.validate_on_submit():
        # print(request.form)
        print(form.username) #拿到username的整个标签
        print(form.username.data) #取出username里面的value值
        return '数据提交过来了'
    return render_template('register.html',form=form)


if __name__ == '__main__':
    maanger.run()
```

form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h2>flask-wtf的表单类</h2>
<form action="{{ url_for('register') }}" method="post">
    {{ form.csrf_token }}
    {{ form.username.label }}
    {{
form.username(style="color:red;",class='myself',placeholder='请输入用
户名...') }} #给当前的标签添加属性和值 关键字参数
    #循环迭代取出验证失败的错误信息（也就是你在验证器里的属性message的值）
    {% for error in form.username.errors %}
        <span style="color:red;">{{ error }}</span>
    {% endfor %}
    <br>
    {{ form.userpass.label }}
    {{ form.userpass }}
    {% for error in form.userpass.errors %}
```

```
            <span style="color:red;">{{ error }}</span>
        {% endfor %}
        <br>
        {{ form.confirm.label }}
        {{ form.confirm() }}
        {% for error in form.confirm.errors %}
            <span style="color:red;">{{ error }}</span>
        {% endfor %}
        <br>
        {{ form.email.label }}
        {{ form.email() }}
        {% for error in form.email.errors %}
            <span style="color:red;">{{ error }}</span>
        {% endfor %}
        <br>
        {{ form.submit() }}
    </form>
    </body>
    </html>
```

form.html 宏定义的表单

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
{# 定义表单宏 #}
{% macro filed(att) %}
    <tr>
        <td>{{ att.label() }}</td>
        <td>{{ att() }}</td>
        {% for error in att.errors %}
        <td style="color:red;">{{ error }}</td>
        {% endfor %}
    </tr>
{% endmacro %}
<form action="" method="post">
<table>
    {{ filed(form.username) }}
```

```
    {{ filed(form.userpass) }}
    {{ filed(form.confirm) }}
    {{ filed(form.email) }}
    <tr>
        <td>{{ form.submit() }}</td>
    </tr>
</table>
</form>
</body>
</html>
```

使用bootstrap渲染表单

```
{% extends 'common/base.html' %}
{% block title %}
    注册
{% endblock %}
{% from 'bootstrap/wtf.html' import quick_form %}
{% block pagecontent %}
    <div class="row">
        <div class="col-md-8">.col-md-8</div>
        <div class="col-md-4">{{ quick_form(form) }}</div>
    </div>
{% endblock %}
```

(3) 自定义表单验证器

在表单类中添加验证方法

导入验证失败的错误提示方法

```
from wtforms.validators import ValidationError
# 需求，帮我去查看表单中的用户名和邮箱是否在用户表中已存在
def validate_username(self,field):
    # print(field.data)
    # print(self.username.data)
    #     if User.objects.filter(username=field.data).exists()
    if field.data == 'zhangsan':
        raise ValidationError('该用户已存在 请重新输入')
        #验证邮箱
def validate_email(self,field):
     if field.data == '793390457@qq.com':
        raise ValidationError('该邮箱已存在 请重新输入')
```

(4) 登录实例

```
#登录表单类
class Login(FlaskForm):
    username = StringField('用户名',validators=
[DataRequired(message='用户名不能为空'),Length(min=6,max=12,message="用户名长度为6～12位之间")])
    userpass = PasswordField('密码', validators=[DataRequired('请输入密码'), Length(min=6, max=12, message='密码长度范围在6~12位之间')])
    submit = SubmitField('登录')

#登录
@app.route('/login/',methods=['GET','POST'])
def login():
    form = Login()
    if form.validate_on_submit():
        print(request.form)
    return render_template('user/login.html',form=form)
```

登录模板和注册模板一样

(5) 所有的字段和验证器

```
from flask import Flask,render_template,request
from flask_script import Manager
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
```

```
from wtforms import
SubmitField,HiddenField,TextAreaField,DateField,DateTimeField,Integer
Field,FloatField,BooleanField,RadioField,SelectField,FileField,String
Field
from wtforms.validators import IPAddress,NumberRange,URL,Regexp

app = Flask(__name__)
app.config['SECRET_KEY'] = 'abcdesdfdsfsfs'
bootstrap = Bootstrap(app)
manager = Manager(app)

#测试表单模型类
class Test(FlaskForm):
    hidde = HiddenField()
    info = TextAreaField('个人简介',render_kw=
{'style':'resize:none','placeholder':'默认值'})
    datefield = DateField('日期',format='%Y/%m/%d')
    datetimefield = DateTimeField('日期和时间')
    integerfield = IntegerField('整形')
    floatfield = FloatField('浮点数')
    boolfield = BooleanField('布尔值')
    radiofield = RadioField('性别',choices=[(0,'男'),(1,'女')])
    selectfield = SelectField('地址',choices=[('1001','北京'),
('1002','上海')])
    filefield = FileField('文件上传')
    ip = StringField('ip地址',validators=[IPAddress(message='输入有效
的ip地址')])
    age = IntegerField('年龄',validators=
[NumberRange(min=6,max=70,message='年龄范围在6～70之间')])
    url = StringField('资源地址',validators=[URL(message='您输入的资源
地址不存在')])
    iphone = IntegerField('手机号码',validators=[Regexp('^1[3-8][0-9]
{9}$',message='请输入正确的手机号码')])
    iphone = StringField('手机号码',validators=[Regexp('^1[3-8][0-9]
{9}$',message='请输入正确的手机号码')])
    submit = SubmitField('test')


@app.route('/test/',methods=['GET','POST'])
def test():
    form = Test()
    form.hidde.data = '我是默认值'
```

```
    form.radiofield.data = '0'
    form.selectfield.data = '1002'
    if form.validate_on_submit():
        print(request.form)
        return '过来了'
    return render_template('testwtf.html',form=form)
```