在Flask中，为了处理web表单，我们一般使用Flask-WTF扩展，它封装了WTForms，并且它有验证表单数据的功能。

```
pip install flask-wtf
```

使用Flask-WTF需要配置参数SECRET_KEY，Flask-WTF提供 CSRF 防跨域请求攻击。 SECRET_KEY用来生成加密令牌，当CSRF激活的时候，该设置会根据设置的密匙生成加密令牌

1.WTForms 支持表单类型

字段对象	说明
StringField	文本字段
TextAreaField	多行文本字段
PasswordField	密码文本字段
HiddenField	隐藏文本字段
DateField	文本字段，值为datetime.date格式
DateTimeField	文本字段，值为datetime.datetime格式
IntegerField	文本字段，值为整数
DecimalField	文本字段，值为decimal.Decimal
FloatField	文本字段，值为浮点数

字段对象	说明
BooleanField	复选框，值为True和False
RadioField	一组单选框
SelectField	下拉列表
SelectMultipleField	下拉列表，可选择多个值
FileField	文本上传字段
SubmitField	表单提交按钮
FormField	把表单作为字段嵌入另一个表单
FieldList	一组指定类型的字段

2.WTForms常用验证函数

验证函数	说明
URL	验证 URL
Email	验证电子邮件地址
Length	验证输入字符串的长度
DataRequired	确保字段中有数据
EqualTo	比较两个字段的值；常用于要求输入两次密码进行确认的情况
IPAddress	验证 IPv4 网络地址

验证函数	说明
NumberRange	验证输入的值在数字范围内
Optional	无输入值时跳过其他验证函数
Regexp	使用正则表达式验证输入值
AnyOf	确保输入值在可选值列表中
NoneOf	确保输入值不在可选值列表中

在Flask中如果不使用WTF扩展表单，那么只能在前端使用js验证，或者将数据传给服务器验证。

这些验证过程都需要开发人员去完成。使用扩展之后这些验证工作都交给第三方扩展去实现，开发人员无需再写这些验证代码。

3.案例

```
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask, request, render_template,
url_for,redirect
# 导入wtf扩展的表单类
from flask_wtf import FlaskForm
# 导入自定义表单需要的字段
from wtforms import SubmitField, StringField, PasswordField,
SelectMultipleField, RadioField
# 导入wtf扩展提供的表单验证器
from wtforms.validators import DataRequired, EqualTo, Length,
Regexp
```

```
app = Flask(__name__,
            template_folder='templates')

# secret_key 必填
app.config['SECRET_KEY'] = '&^%%kdsjjlsd*&%$%'


class Login(FlaskForm):
    """自定义表单类，文本字段、密码字段、提交按钮"""
    # 用户名表单，必填字段，8-16个字符
    username = StringField(label=u'用户：',validators=
[DataRequired(),Length(8,16)])
    # 密码表单，必填字段,使用正则表达式校验
    password = PasswordField(label=u'密码',validators=
[DataRequired(),Regexp('[0-9a-zA-Z!@#$%]{6}')])

    # 确认密码表单
    check_password = PasswordField(label=u'确认密码',
validators=[DataRequired(),
EqualTo('password', 'check_password')])
    # 下拉多选框
    select = SelectMultipleField('domain_ports', choices=
[('0', 'flask'),('1', 'python')], validators=[DataRequired()])

    # 单选框,这里特别注意 coerce参数指定元祖中第一个参数的数据类
型，否则校验失败
    # 如果是 字符串 coerce必须需要知道为str,数字则为int
    radio = RadioField(2, choices=[(1, 'flask'), (2, 'web')],
coerce=int)

    # 提交表单
    submit = SubmitField(u'提交')
    # u"字符串。"
    # 作用：后面字符串以 Unicode 格式 进行编码，一般用在中文字符串
前面，防止因为源码储存格式问题，导致再次使用时出现乱码。
    # 不是仅仅是针对中文，可以针对任何的字符串，代表是对字符串进
行。一般英文字符在使用各种编码下,，基本都可以正常解析，所以一般不带
u。
```

```
@app.route('/')
def index():
    return render_template('wtf_index.html')


@app.route('/login', methods=['GET', 'POST'])
def login():

    # 创建表单对象
    form = Login()
    # 用户提交数据，验证成功返回True,否则False
    if form.validate_on_submit():
        # 获取用户输入的数据
        name = form.username.data
        password = form.password.data
        check_password = form.check_password.data
        sel = form.select.data
        radio = form.radio.data
        print(name, password, check_password, sel, radio)
        # 数据校验成功重定向到首页
        return redirect(url_for('index'))
    else:
        # 如果是post请求过来，但是数据校验失败
        if request.method == 'POST':
            # 验证失败的字段
            print(form.errors)
            return render_template('wtf_login.html', error='账
号或密码错误', form=form)
        else:
            # 用户发的是get，直接将返回模板
            return render_template('wtf_login.html',
form=form)


if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form method="post">

  {{ form.csrf_token() }}
  {{ form.username.label }}
  <p>{{ form.username }}</p>
  {{ form.password.label }}
  <p>{{ form.password }}</p>
  {{ form.check_password.label }}
  <p>{{ form.check_password }}</p>
  {{ form.select }}
  {{ form.radio }}
  <p>{{ form.submit() }}</p>

</form>
</body>
</html>
```

127.0.0.1:5000/login

用户:

密码

确认密码

flask
python

- flask
- web

提交