Flask Note

1.虚拟环境

创建虚拟环境

- virtualenv venv

进入虚拟环境

- venv\Scripts\activate

生成需求文件

- pip freeze >requirements.txt

安装依赖

- pip install -r requirements.txt

2.Flask结构

最简Flask程序

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return '<h1>Hello World!</h1>'

if __name__ == '__main__':
    app.run(debug=True)
```

重定向

- redirect(url)

runserver

- python hello.py runserver --host 0.0.0.0

3.模板

渲染模板

- render_template('user.html', name=name)

控制结构

- 条件控制

```
{% if user %}
    Hello, {{ user }}!
{% else %}
    Hello, Stranger!
{% endif %}
```

- 循环

```
<ul>
    {% for comment in comments %}
        <li>{{ comment }}</li>
    {% endfor %}
</ul>
```

- 宏

```
{% macro render_comment(comment) %}
    <li>{{ comment }}</li>
{% endmacro %}

<ul>
    {% for comment in comments %}
        {{ render_comment(comment) }}
    {% endfor %}
</ul>
```

- 使用宏

```
{% import 'macros.html' as macros %}

<ul>
    {% for comment in comments %}
        {{ macros.render_comment(comment) }}
    {% endfor %}
</ul>
```

- 插入

```
{% include 'common.html' %}
```

- 继承

```
{% extends "base.html" %}

{% block title %}Index{% endblock %}

{% block head %}
    {{ super() }}
{% endblock %}
```

4.表单

表单类

```
class NameForm(Form):
    name = StringField('What is your name?', validators=[DataRequired()])
    submit = SubmitField('Submit')
```

把表单渲染成HTML

```
{% import "bootstrap/wtf.html" as wtf %}
{{ wtf.quick_form(form) }}
```

在视图函数中处理表单

```
@app.route('/', methods=['GET', 'POST'])
def index():
    form = NameForm()
    if form.validate_on_submit():
        session['name'] = form.name.data
        return redirect(url_for('index'))
    return render_template('index.html', form=form, name=session.get('name'))
```

5.数据库

配置

- app.config['SQLALCHEMY_DATABASE_URI']
- app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN']

模型

```
class Role(db.Model):
    __tablename__ = 'roles'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), unique=True)

    def __repr__(self):
        return '<Role %r>' % self.name
```

关系

- 一对多

```
class Role(db.Model):
    users = db.relationship('User', backref='role')

class User(db.Model):
    role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
```

- 一对一

```
    users = db.relationship('User', backref='role', uselist=False)
```

数据库操作

- db.create_all()
- db.drop_all()

- db.session.add(obj)
- db.session.add_all([obj1, obj2 ,...])
- db.session.commit()
- db.session.rollback()
- db.session.delete(obj)

- obj.query.all()
- obj.query.filter_by(xxx=xxx).all()
- obj.query.filter_by(xxx=xxx).first()

集成Python shell

```
def make_shell_context():
    return dict(app=app, db=db, User=User, Role=Role)
manager.add_command("shell", Shell(make_context=make_shell_context))
```

6.电子邮件

配置

```
app.config['MAIL_SERVER'] = 'smtp.googlemail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = os.environ.get('MAIL_USERNAME')
app.config['MAIL_PASSWORD'] = os.environ.get('MAIL_PASSWORD')
```

发送

```
def send_email(to, subject, template, **kwargs):
    msg = Message(subject,
                  sender='you@example.com',
                  recipients=[to])
    msg.body = render_template(template + '.txt', **kwargs)
    msg.html = render_template(template + '.html', **kwargs)
    mail.send(msg)
```

异步发送

```
def send_async_email(app, msg):
    with app.app_context():
        mail.send(msg)

thr = Thread(target=send_async_email, args=[app, msg])
thr.start()
```

附表

全局上下文

变量名	上下文	说明
current_app	程序上下文	当前激活程序的程序实例
g	程序上下文	处理请求时用作临时存储的对象。每次请求都会重设这个变量
request	请求上下文	请求对象，封装了客户端发出的HTTP 请求中的内容
session	请求上下文	用户会话，用于存储请求之间需要“记住”的值的词典

WTForms

WTForms支持的HTML标准字段

字段类型	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
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	文件上传字段
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

WTForms验证函数

验证函数	说明
Email	验证电子邮件地址
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
Length	验证输入字符串的长度
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

DataRequired	确保字段中有数据
Regexp	使用正则表达式验证输入值
URL	验证URL
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

SQLAlchemy

Flask-SQLAlchemy数据库URL

数据库引擎	URL
MySQL	mysql://username:password@hostname/database
Postgres	postgresql://username:password@hostname/database
SQLite（Unix）	sqlite:////absolute/path/to/database
SQLite（Windows）	sqlite:///c:/absolute/path/to/database

最常用的SQLAlchemy列类型

类型名	Python类型	说明
Integer	int	普通整数，一般是32位
SmallInteger	int	取值范围小的整数，一般是16位
BigInteger	int 或 long	不限制精度的整数
Float	float	浮点数
Numeric	decimal.Decimal	定点数
String	str	变长字符串
Text	str	变长字符串，对较长或不限长度的字符串做了优化
Unicode	unicode	变长Unicode 字符串
UnicodeText	unicode	变长Unicode 字符串，对较长或不限长度的字符串做了优化

Boolean	bool	布尔值
Date	datetime.date	日期
Time	datetime.time	时间
DateTime	datetime.datetime	日期和时间
Interval	datetime.timedelta	时间间隔
Enum	str	一组字符串
PickleType	任何Python对象	自动使用Pickle序列化
LargeBinary	str	二进制文件

最常使用的SQLAlchemy列选项

选项名	说明
primary_key	如果设为True，这列就是表的主键
unique	如果设为True，这列不允许出现重复的值
index	如果设为True，为这列创建索引，提升查询效率
nullable	如果设为True，这列允许使用空值；如果设为False，这列不允许使用空值
default	为这列定义默认值

常用的SQLAlchemy关系选项

选项名	说明
backref	在关系的另一个模型中添加反向引用
primaryjoin	明确指定两个模型之间使用的联结条件。只在模棱两可的关系中需要指定
lazy	指定如何加载相关记录。可选值有select（首次访问时按需加载）、immediate（源对象加载后就加载）、joined（加载记录，但使用联结）、subquery（立即加载，但使用子查询），noload（永不加载）和dynamic（不加载记录，但提供加载记录的查询）
uselist	如果设为Fales，不使用列表，而使用标量值
order_by	指定关系中记录的排序方式
secondary	指定多对多关系中关系表的名字
secondaryjoin	SQLAlchemy 无法自行决定时，指定多对多关系中的二级联结条件

常用的SQLAlchemy查询过滤器

过滤器	说明
filter()	把过滤器添加到原查询上，返回一个新查询
filter_by()	把等值过滤器添加到原查询上，返回一个新查询
limit()	使用指定的值限制原查询返回的结果数量，返回一个新查询
offset()	偏移原查询返回的结果，返回一个新查询
order_by()	根据指定条件对原查询结果进行排序，返回一个新查询
group_by()	根据指定条件对原查询结果进行分组，返回一个新查询

最常使用的SQLAlchemy查询执行函数

方法	说明
all()	以列表形式返回查询的所有结果
first()	返回查询的第一个结果，如果没有结果，则返回None
first_or_404()	返回查询的第一个结果，如果没有结果，则终止请求，返回404 错误响应
get()	返回指定主键对应的行，如果没有对应的行，则返回None
get_or_404()	返回指定主键对应的行，如果没找到指定的主键，则终止请求，返回404错误响应
count()	返回查询结果的数量
paginate()	返回一个Paginate 对象，它包含指定范围内的结果

Flask-Mail

Flask-Mail SMTP服务器的配置

配置	默认值	说明
MAIL_SERVER	localhost	电子邮件服务器的主机名或IP地址
MAIL_PORT	25	电子邮件服务器的端口
MAIL_USE_TLS	False	启用传输层安全（Transport Layer Security，TLS）协议
MAIL_USE_SSL	False	启用安全套接层（Secure Sockets Layer，SSL）协议

MAIL_USERNAME	None	邮件账户的用户名
MAIL_PASSWORD	None	邮件账户的密码