Chapter2 Structure of Programming

Note1：

钩子主要是处理如数据库连接，数据库处理，辅助函数使用。

主要有四种钩子：

1. before_first_request:注册一个函数，在处理第一个请求之前调用

eg: connect database firstly

usage:

```
@before_first_request()
def ():
    pass
```

2. before_request: 注册一个函数，在每次请求之前运行

3. after_request：注册一个函数，如果没有微处理的异常抛出，在每次请求之后运行

4. teardown_request：注册一个函数，如果没有微处理的异常抛出，在每次请求之后运行

Note2：

一般情况下的编写视图方法：

```
from flask import Flask
app = Flask(__name)
@app.route('/')
def index():
    return '<h1>Hello, Flask!</h1>'

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

首先 app 是实例化后的对象，Flask 类的构造函数只有一个参数，即程序主模块或包的名字。在 Python 的__name__变量就是所需的值。

接下来是路由和视图函数

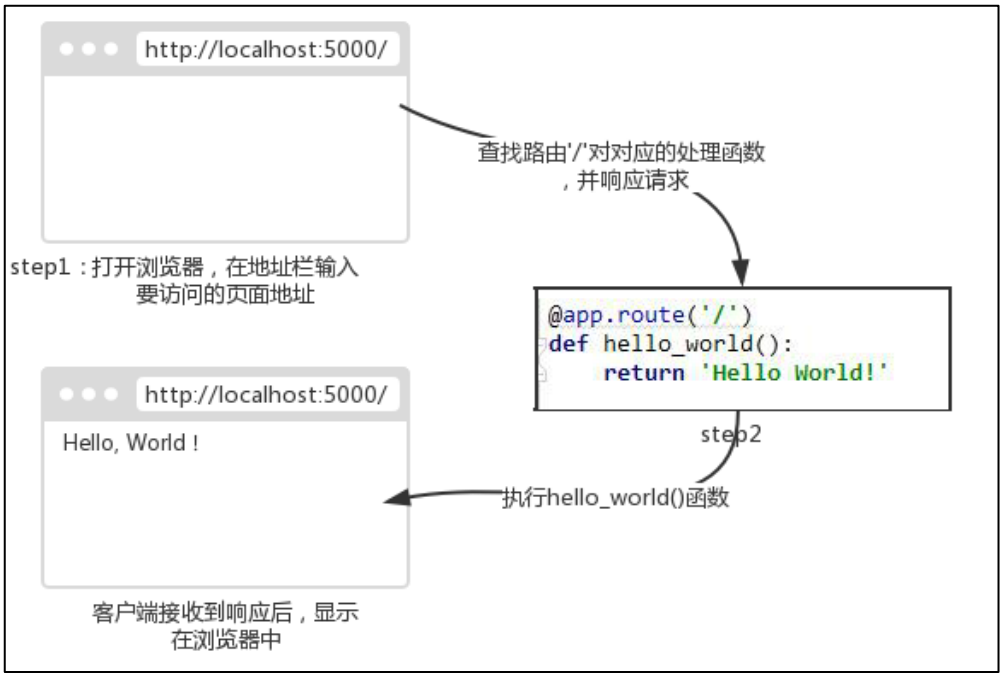


图 1 路由和视图

最后，启动服务器。__name__ == '__main__'是 Python 的常用用法。确保执行这个脚本启动开发服务器。

Note3：

带有参数：

```
@app.route('/user/<name>')
def hello(name):
    return '<p>Hello, %s'%name
```

```
@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id
```

有以下几种形式：

<int>,<float>,<path>,<any>等。path 也是字符串类型，但不把'/'作为分隔符。

Note4：

返回 Response 对象和 return ‘string’具有一样的效果。

```
@app.route('/')
def index():
    """make_response return response object"""
    response = make_response('<h1>Hello, flask!</h1>')
    # response
    return response
```

Note5：

特殊的重定向响应：

```
# redirect
@app.route('/')
def index():
    return redirect('https://www.baidu.com')
```

Note6:

特殊的响应：异常

```
def load_user(id):
    pass
@app.route('/user/<int:id>')
def get_user(id):
    user = load_user(id)
    if not user:
        abort(404)
    return '<h1>Hello, %s</h1>'%user.name
```

Note7:

扩展

为了使用命令行下启动开发服务器，类似 Django 里面的 python manage.py runserver 0.0.0.0:8000 一样，所以使用 Flask-Script 来支持这种形式。

```
from flask_script import Manager
app = Flask(__name__)
```

```
manager = Manager(app=app)


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

运行：

```
python flasku.py runserver --port 8000 --host 0.0.0.0
```

```
usage: flasku.py runserver [-?] [-h HOST] [-p PORT] [--threaded]
                           [--processes PROCESSES] [--passthrough-errors] [-d]
                           [-D] [-r] [-R]
```

Chapter3 Template

Note1：

业务逻辑和表现逻辑：

业务逻辑是从用户的角度看，具体的操作，而表现逻辑是指后台发生的动作。

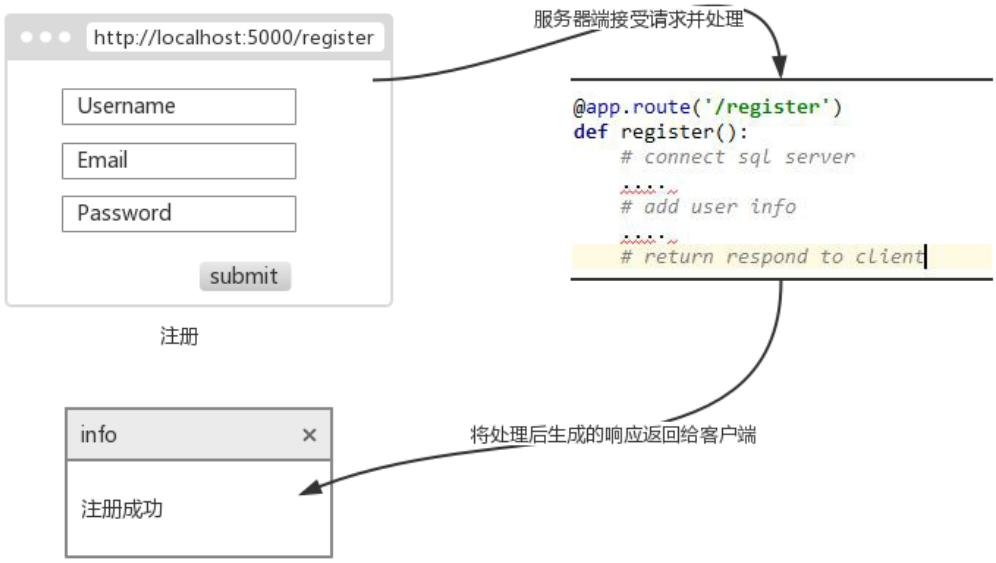


图 2 业务逻辑和表现逻辑

Note2：

使用模板渲染，默认情况下模板文件放在 templates 文件中。

flasku.py

```
from flask import render_template
…
@app.route('/')
def hello_world():
    # return 'Hello World!'
    return render_template('index.html')
…
@app.route('/user/<name>')
def hello(name):
```

```
        """name=name:left depict template user.html placeholder,and right depict region name varible"""
        return render_template('user.html', name=name)
…
```

index.html

```
<p>Hello World</p>
```

user.html

```
<p>Hello {{ name }}</p>
```

有以下注意的几点：

1. return render_template('user.html', name=name)中的第一个参数为模板文件的名称，第二个参数是传递的参数，其中 name=name，name 表示模板 user.html 中的{{ name }}中的 name,它是一个占位符。而 name 表示/user/<name>，是一个变量。

2. Flask 使用的是 Jinja2 模板，因此能识别出所有的变量，比如列表，对象，对象。

3. Jinja2 模板支持使用过滤器修改变量，格式为{{ placeholder | safe | … }}，safe 可以用到输入密码时。

Note3：

控制语句：

判断语句：

```
{% if user %}
    <p>Hello, {{ user }}</p>
{% else %}
    <p>Hello Stranger.</p>
{% endif %}
```

循环语句：

```
<ul>
{% for u in user %}
    <li>{{ u }}</li>
{% endfor %}
</ul>
```

支持宏：

```
{% macro input(name, value='', type='text', size=20) -%}
```

```
    <input type="{{ type }}" name="{{ name }}" value="{{
        value|e }}" size="{{ size }}">
{%- endmacro %}
```

基模板继承：类似类继承

定义一个基模板，并在基模板上定义 title，head，body 等块，title 包含在 head 之中。

base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    {% block head %}
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %} - My Applications</title>
    {% endblock %}
</head>
<body>
    {% block body %}
    {% endblock %}
</body>
</html>
```

index.html 为 base.html 的衍生模板。

index.html

```
{% extends 'base.html' %}
{% block head %}
    {{ super() }}
    <style>
    </style>
{% endblock %}

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

```
{% block body %}
    <h1>Hello World</h1>
{% endblock %}
```

Note：

使用 Bootstrap 扩展：

flasku.py

```
from flask_bootstrap import Bootstrap
…
bootstrap = Bootstrap(app)
…
```

user.html

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

{% block title %}Flasku{% endblock %}

{% block navbar %}
    <div class="navbar navbar-inverse" role="navigation">
    <div class="container">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Flasku</a>
    </div>
```

```
        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
            </ul>

        </div><!-- /.navbar-collapse -->
    </div><!-- /.container-fluid -->
        </div>
{% endblock %}

{% block content %}
        <div class="container">
            <div class="page-header">
                <h1>Hello, {{ name }}!</h1>
            </div>
        </div>
{% endblock %}
```

定义的块：

Block name	Outer Block	Purpose
doc		Outermost block.
html	doc	Contains the complete content of the `<html>` tag.
html_attribs	doc	Attributes for the HTML tag.
head	doc	Contains the complete content of the `<head>` tag.
body	doc	Contains the complete content of the `<body>` tag.
body_attribs	body	Attributes for the Body Tag.
title	head	Contains the complete content of the `<title>` tag.
styles	head	Contains all CSS style `<link>` tags inside head.
metas	head	Contains all `<meta>` tags inside head.
navbar	body	An empty block directly above *content*.
content	body	Convenience block inside the body. Put stuff here.
scripts	body	Contains all `<script>` tags at the end of the body.

图 3 基模板定义的块

Note：

定义 404 和 500 错误处理：

```
# define 404 and 500
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404

@app.errorhandler(500)
def internal_server_error(e):
    return render_template('500.html'), 500
```

Note:

链接：

```
{%  block head %}
    {{ super() }}
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
    type="image/x-icon">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico' }}" type="image/x-icon">
{% endblock %}
if __name__ == '__main__':
    manager.run()
```

url_for('view_func', para1='…', para2='…')

Note:

本地化日期和时间

```
from flask import Moment
from datetime import datetime
moment = Moment(app) #  在模板中加载 moment.js

@app.route('/')
def index():
```

```
    return render_template('index.html', current_time=datetime.utcnow())
```

base.html

```
{% block script %}
    {{ super() }}
    {{ moment.include_moment() }}
{% endblock %}
```

super()的意思是保证原基模板 bootstrap/base.html 的 script 块原始内容。

index.html

```
<p>Local date and time is {{ moment(current_time).format('LLL') }}</p>
<p>That was{{ moment(current_time).fromNow(refresh=True) }}</p>
```

Chapter4 Web Form

Note1:

跨站请求伪造保护：

Django:{{ csrf_token() }} -> html template

Flask:app.config[‘SECRET_KEY’]=’hard to gesss string’

定义表单：继承 Form

Form 基类从 Flask-WTF 扩展定义，所以从 flask.ext.wtf 中导入

字段和验证函数可以直接从 WTForms 包中导入

```
from flask.ext.wtf import Form
from wtforms import StringField,SubmitField
from wtforms.validators import Required


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

表4-1 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

图 3 支持的 HTML 标准字段

表4-2 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
Required	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

图 5 验证函数

渲染表单：

index.html

```
{% import 'bootstrap/wtf.html' as wtf %}


{{ wtf.quick_form(form) }}
```

flasku.py

```
@app.route('/', methods=['GET', 'POST'])
def hello_world():
    name = None
    form = NameForm()
    if form.validate_on_submit():
        name = form.name.data
        form.name.data = ''
    return render_template('index.html', form=form, name=name)
```

这样做会造成一刷新就得重新提交最后的表单数据。为了避免，改代码：

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

```
    # return 'Hello World!'
    # return render_template('index.html', current_time=datetime.utcnow())
    form = NameForm()
    if form.validate_on_submit():
        session['name'] = form.name.data
        return redirect(url_for('hello_world'))
    return render_template('index.html', form=form, name=session.get('name'))
```

使用 session 用户会话，保存在用户会话中。最后调用 redirect()函数，参数为地址。url_for 必须包含端点名，即路由内部的名字。

Note：

Flash 消息：

```
…
from flask import flash
…
@app.route('/', methods=['GET', 'POST'])
def hello_world():
    # return 'Hello World!'
    # return render_template('index.html', current_time=datetime.utcnow())
    form = NameForm()
    if form.validate_on_submit():
        old_name = session.get('name')
        if old_name is not None and old_name!= form.name.data:
            flash('Looks like you have change your name!')
        session['name'] = form.name.data
        return redirect(url_for('hello_world'))
    return render_template('index.html', form=form, name=session.get('name'))
…
```

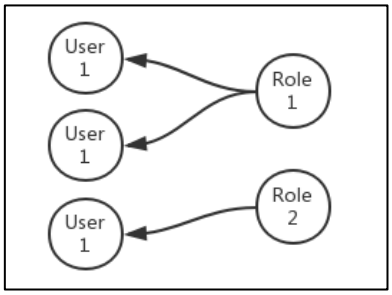
渲染：get_flashed_messages()用来获取并渲染消息的。

base.html

```
{% block content %}
    <div class="container">
        {% for message in get_flashed_messages() %}
            <div class="alert alert-warning" role="alert">
                <button type="button" class="close" data-dismiss="alert">&amp;times;</button>
                {{ message }}
            </div>
        {% endfor %}
        {% block page_content %}{% endblock %}
    </div>
{% endblock %}
```

Chapter5 Database

Note:



Cahpter6 Mail

Chapter7 Scale Web Structure

Chapter8 User Auth

```
(venv) C:\Users\Administrator\PycharmProjects\flasky>dir
 Volume in drive C is windows-10-software
 Volume Serial Number is C038-3181

 Directory of C:\Users\Administrator\PycharmProjects\flasky

2017/06/04  12:22    <DIR>          .
2017/06/04  12:22    <DIR>          ..
2017/06/04  12:22    <DIR>          .idea
2017/06/04  12:05    <DIR>          app
2017/06/04  12:08             1,572 config.py
2017/06/04  12:08             2,450 config.pyc
2017/06/04  12:12             7,168 data-dev.sqlite
2017/06/04  12:22             5,120 data-test.sqlite
2017/06/04  12:21               826 manage.py
2017/06/04  12:12    <DIR>          migrations
2017/06/04  12:14               417 requirement.txt
2017/06/04  12:22    <DIR>          tests
               6 File(s)         17,553 bytes
               6 Dir(s)  13,356,617,728 bytes free
```

图 1 程序结构