FLASK 快速参考

路由、模板、请求、蓝图、数据库、扩展

初始化

最简应用

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello, World!'
```

运行应用

```
pip install flask
flask --app app run --debug
# or: python -m flask run --debug
```

项目结构

`app.py`	应用入口
`templates/`	Jinja2 HTML 模板
`static/`	CSS、JS、图片
`models.py`	数据库模型
`requirements.txt`	Python 依赖

路由

基础路由

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

@app.route('/user/<username>')
def profile(username):
    return f'User: {username}'
```

URL 变量

`<variable>`	字符串（默认）
`<int:id>`	整数
`<float:price>`	浮点数
`<path:subpath>`	含斜杠的字符串
`<uuid:item_id>`	UUID

HTTP 方法

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_login()
    return render_template('login.html')
```

URL 构建

```
from flask import url_for
url_for('profile', username='alice')
# => '/user/alice'
```

模板

渲染模板

```
from flask import render_template

@app.route('/posts')
def posts():
    items = get_posts()
    return render_template('posts.html', posts=items)
```

Jinja2 语法

```
{{ variable }}
{% if user %}Welcome, {{ user.name }}!{% endif %}
{% for item in items %}
  <li>{{ item }}</li>
{% endfor %}
```

模板继承

```
{# base.html #}
<html><body>{% block content %}{% endblock %}</body></html>

{# child.html #}
{% extends "base.html" %}
{% block content %}<h1>Page</h1>{% endblock %}
```

常用过滤器

`|safe`	渲染原始 HTML
`|escape`	HTML 转义字符串
`|length`	统计元素数量
`|default('N/A')`	空值时的回退
`|tojson`	序列化为 JSON

请求与响应

Request 对象

```
from flask import request

request.method          # 'GET', 'POST'
request.args.get('q')   # query string ?q=value
request.form['name']    # form POST data
request.json            # parsed JSON body
```

请求属性

`request.args`	查询字符串参数
`request.form`	表单 POST 数据
`request.json`	解析后的 JSON 请求体
`request.files`	上传的文件
`request.headers`	HTTP 头
`request.cookies`	Cookie 值

响应辅助函数

```
from flask import jsonify, redirect, make_response

return jsonify({'status': 'ok'})  # JSON response
return redirect(url_for('index'))  # redirect
resp = make_response('body', 200)
resp.headers['X-Custom'] = 'value'
```

Session

```
from flask import session
app.secret_key = 'your-secret-key'
session['user_id'] = 42
uid = session.get('user_id')
```

表单

WTForms 集成

```
pip install flask-wtf
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField
from wtforms.validators import DataRequired
```

定义表单

```
class LoginForm(FlaskForm):
    username = StringField('User', validators=[DataRequired()])
    password = PasswordField('Pass', validators=[DataRequired()])
```

在视图中使用

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = form.username.data
        return redirect(url_for('dashboard'))
    return render_template('login.html', form=form)
```

模板中的表单

```
<form method="post">
  {{ form.hidden_tag() }}
  {{ form.username.label }} {{ form.username() }}
  {{ form.password.label }} {{ form.password() }}
  <button type="submit">Login</button>
</form>
```

数据库

SQLAlchemy 初始化

```
pip install flask-sqlalchemy
from flask_sqlalchemy import SQLAlchemy
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'
db = SQLAlchemy(app)
```

定义模型

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    email = db.Column(db.String(120), unique=True)
    posts = db.relationship('Post', backref='author')
```

CRUD 操作

```
user = User(name='Alice', email='alice@example.com')
db.session.add(user)
db.session.commit()
User.query.filter_by(name='Alice').first()
db.session.delete(user)
db.session.commit()
```

常用查询

`Model.query.all()`	所有记录
`Model.query.get(id)`	按主键查询
`.filter_by(name='X')`	简单等值过滤
`.filter(Model.age > 18)`	表达式过滤
`.order_by(Model.name)`	排序
`.limit(10).offset(20)`	分页

蓝图

创建蓝图

```
from flask import Blueprint
blog = Blueprint('blog', __name__, url_prefix='/blog')

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

注册蓝图

```
# app.py
from blog import blog
app.register_blueprint(blog)
```

蓝图 URL 构建

```
url_for('blog.index')  # => '/blog/'
url_for('blog.post', id=5)  # => '/blog/post/5'
```

蓝图结构

`url_prefix`	为蓝图内所有路由添加前缀
`template_folder`	自定义模板目录
`static_folder`	蓝图专属静态文件
`@bp.before_request`	每次蓝图请求前执行

错误处理

自定义错误页面

```
@app.errorhandler(404)
def not_found(e):
    return render_template('404.html'), 404

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

中止请求

```
from flask import abort

@app.route('/admin')
def admin():
    if not current_user.is_admin:
        abort(403)
    return render_template('admin.html')
```

自定义异常

```
from werkzeug.exceptions import HTTPException

class InsufficientFunds(HTTPException):
    code = 402
    description = 'Insufficient funds'
```

日志

```
app.logger.info('User %s logged in', username)
app.logger.warning('Disk space low')
app.logger.error('Payment failed: %s', err)
```

配置

配置方式

```
app.config['DEBUG'] = True
app.config.from_object('config.ProductionConfig')
app.config.from_envvar('APP_SETTINGS')
```

配置类模式

```
class Config:
    SECRET_KEY = os.environ.get('SECRET_KEY')
    SQLALCHEMY_TRACK_MODIFICATIONS = False

class DevConfig(Config):
    DEBUG = True
    SQLALCHEMY_DATABASE_URI = 'sqlite:///dev.db'
```

常用配置项

`SECRET_KEY`	Session 签名密钥（必填）
`DEBUG`	启用调试模式
`TESTING`	启用测试模式
`SQLALCHEMY_DATABASE_URI`	数据库连接字符串
`MAX_CONTENT_LENGTH`	最大上传大小（字节）
`JSON_SORT_KEYS`	对 JSON 输出键排序

扩展

常用扩展

`Flask-SQLAlchemy`	ORM 集成
`Flask-Migrate`	Alembic 数据库迁移
`Flask-WTF`	表单处理与 CSRF
`Flask-Login`	用户 session 管理
`Flask-Mail`	邮件发送
`Flask-CORS`	跨域资源共享
`Flask-RESTful`	REST API 构建
`Flask-Caching`	响应与函数缓存

Flask-Login

```
from flask_login import LoginManager, login_required
login_manager = LoginManager(app)
login_manager.login_view = 'login'

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

Flask-Migrate

```
from flask_migrate import Migrate
migrate = Migrate(app, db)
# flask db init    (once)
# flask db migrate -m "add users"
# flask db upgrade
```