

Tham Khảo Nhanh Flask

Routes, templates, requests, blueprints, database, extensions

Cài Đặt

Ứng Dụng Tối Giản

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

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

Chạy Ứng Dụng

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

Cấu Trúc Dự Án

app.py	Điểm vào ứng dụng
templates/	Template HTML Jinja2
static/	CSS, JS, ảnh
models.py	Các model database
requirements.txt	Thư viện Python phụ thuộc

Routes

Routes Cơ Bản

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

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

Biến URL

<variable>	Chuỗi (mặc định)
<int:id>	Số nguyên
<float:price>	Số thực
<path:subpath>	Chuỗi có dấu gạch chéo
<uuid:item_id>	UUID

Phương Thức HTTP

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

Xây Dựng URL

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

Templates

Render Template

```
from flask import render_template

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

Cú Pháp Jinja2

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

Kế Thừa Template

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

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

Bộ Lọc Phổ Biến

 safe	Render HTML thô
 escape	Escape chuỗi HTML
 length	Đếm số phần tử
 default('N/A')	Giá trị dự phòng khi rỗng
 tojson	Serialize thành JSON

Request & Response

Đối Tượng Request

```
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
```

Thuộc Tính Request

request.args	Tham số query string
request.form	Dữ liệu POST từ form
request.json	Body JSON đã phân tích
request.files	File được tải lên
request.headers	HTTP headers
request.cookies	Giá trị cookie

Helper Response

```
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')
```

Forms

Tích Hợp WTForms

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

Định Nghĩa Form

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

Dùng Trong View

```
@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)
```

Form Trong Template

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

Database

Cài Đặt SQLAlchemy

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

Định Nghĩa Model

```
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')
```

Thao Tác CRUD

```
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()
```

Truy Vấn Phổ Biến

Model.query.all()	Tất cả bản ghi
Model.query.get(id)	Theo khóa chính
.filter_by(name='X')	Lọc bằng đẳng thức đơn giản
.filter(Model.age > 18)	Lọc bằng biểu thức
.order_by(Model.name)	Sắp xếp kết quả
.limit(10).offset(20)	Phân trang kết quả

Blueprints

Tạo Blueprint

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

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

Đăng Ký Blueprint

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

Xây Dựng URL Blueprint

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

Tham Khảo Nhanh Flask

Cấu Trúc Blueprint

url_prefix	Tiền tố cho tất cả route trong blueprint
template_folder	Thư mục template tùy chỉnh
static_folder	File tĩnh riêng của blueprint
@bp.before_request	Chạy trước mỗi request của blueprint

Xử Lý Lỗi

Trang Lỗi Tùy Chỉnh

```
@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
```

Hủy Request

```
from flask import abort

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

Exception Tùy Chỉnh

```
from werkzeug.exceptions import HTTPException

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

Logging

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

Cấu Hình

Phương Pháp Cấu Hình

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

Mẫu Config Class

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

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

Cài Đặt Phổ Biến

SECRET_KEY	Khóa ký session (bắt buộc)
DEBUG	Bật chế độ debug
TESTING	Bật chế độ test
SQLALCHEMY_DATABASE_URI	Chuỗi kết nối database
MAX_CONTENT_LENGTH	Kích thước upload tối đa tính bằng byte
JSON_SORT_KEYS	Sắp xếp khóa JSON output

Extensions

Extensions Phổ Biến

Flask-SQLAlchemy	Tích hợp ORM
Flask-Migrate	Migration database với Alembic
Flask-WTF	Xử lý form với CSRF
Flask-Login	Quản lý session người dùng
Flask-Mail	Gửi email
Flask-CORS	Chia sẻ tài nguyên cross-origin
Flask-RESTful	Xây dựng REST API
Flask-Caching	Cache response và hàm

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
```