FLASK 빠른 참조

라우트, 템플릿, 요청, 블루프린트, 데이터베이스, 확장

설정

최소 앱

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

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

앱 실행

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

프로젝트 구조

`app.py`	애플리케이션 진입점
`templates/`	Jinja2 HTML 템플릿
`static/`	CSS, JS, 이미지
`models.py`	데이터베이스 모델
`requirements.txt`	Python 의존성

라우트

기본 라우트

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

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

URL 변수

`<variable>`	문자열 (기본값)
`<int:id>`	정수
`<float:price>`	부동소수점
`<path:subpath>`	슬래시 포함 문자열
`<uuid:item_id>`	UUID

HTTP 메서드

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

URL 빌드

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

템플릿

템플릿 렌더링

```
from flask import render_template

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

Jinja2 문법

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

템플릿 상속

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

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

자주 쓰는 필터

`|safe`	원시 HTML 렌더링
`|escape`	HTML 이스케이프
`|length`	항목 수
`|default('N/A')`	빈 값의 대체값
`|tojson`	JSON으로 직렬화

요청 & 응답

요청 객체

```
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
```

요청 속성

`request.args`	쿼리 문자열 파라미터
`request.form`	폼 POST 데이터
`request.json`	파싱된 JSON 본문
`request.files`	업로드된 파일
`request.headers`	HTTP 헤더
`request.cookies`	쿠키 값

응답 헬퍼

```
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'
```

세션

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

폼

WTForms 통합

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

폼 정의

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

뷰에서 사용

```
@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)
```

템플릿에서 폼

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

데이터베이스

SQLAlchemy 설정

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

모델 정의

```
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 작업

```
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()
```

자주 쓰는 쿼리

`Model.query.all()`	모든 레코드
`Model.query.get(id)`	기본 키로 조회
`.filter_by(name='X')`	단순 등가 필터
`.filter(Model.age > 18)`	표현식 필터
`.order_by(Model.name)`	결과 정렬
`.limit(10).offset(20)`	결과 페이지네이션

블루프린트

블루프린트 생성

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

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

블루프린트 등록

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

블루프린트 URL 빌드

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

블루프린트 구조

`url_prefix`	블루프린트의 모든 라우트에 접두사 추가
`template_folder`	커스텀 템플릿 디렉터리
`static_folder`	블루프린트 전용 정적 파일
`@bp.before_request`	각 블루프린트 요청 전에 실행

에러 처리

커스텀 에러 페이지

```
@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
```

요청 중단

```
from flask import abort

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

커스텀 예외

```
from werkzeug.exceptions import HTTPException

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

로깅

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

설정

설정 방법

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

설정 클래스 패턴

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

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

주요 설정

`SECRET_KEY`	세션 서명 키 (필수)
`DEBUG`	디버그 모드 활성화
`TESTING`	테스트 모드 활성화
`SQLALCHEMY_DATABASE_URI`	데이터베이스 연결 문자열
`MAX_CONTENT_LENGTH`	최대 업로드 크기 (바이트)
`JSON_SORT_KEYS`	JSON 출력 키 정렬

확장

주요 확장

`Flask-SQLAlchemy`	ORM 통합
`Flask-Migrate`	Alembic 데이터베이스 마이그레이션
`Flask-WTF`	CSRF 포함 폼 처리
`Flask-Login`	사용자 세션 관리
`Flask-Mail`	이메일 발송
`Flask-CORS`	크로스 오리진 리소스 공유
`Flask-RESTful`	REST API 구축
`Flask-Caching`	응답 및 함수 캐싱

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
```