

Referensi Cepat Flask

Route, template, request, blueprint, database, extension

Setup

App Minimal

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

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

Jalankan App

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

Struktur Proyek

app.py	Titik masuk aplikasi
templates/	Template HTML Jinja2
static/	CSS, JS, gambar
models.py	Model database
requirements.txt	Dependensi Python

Route

Route Dasar

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

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

Variabel URL

<variable>	String (default)
<int:id>	Integer
<float:price>	Float
<path:subpath>	String dengan slash
<uuid:item_id>	UUID

HTTP Method

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

Pembuatan URL

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

Template

Render Template

```
from flask import render_template

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

Sintaks Jinja2

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

Pewarisan Template

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

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

Filter Umum

 safe	Render HTML mentah
 escape	HTML-escape string
 length	Hitung item
 default('N/A')	Fallback untuk nilai kosong
 tojson	Serialisasi ke JSON

Request & Response

Objek 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
```

Properti Request

request.args	Parameter query string
request.form	Data POST form
request.json	Body JSON yang diparsing
request.files	File yang diunggah
request.headers	Header HTTP
request.cookies	Nilai 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')
```

Form

Integrasi WTForms

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

Definisi Form

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

Gunakan di 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 di Template

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

Database

Setup SQLAlchemy

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

Definisi 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')
```

Operasi 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()
```

Query Umum

Model.query.all()	Semua record
Model.query.get(id)	Berdasarkan primary key
.filter_by(name='X')	Filter kesetaraan sederhana
.filter(Model.age > 18)	Filter ekspresi
.order_by(Model.name)	Urutkan hasil
.limit(10).offset(20)	Paginasi hasil

Blueprint

Buat Blueprint

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

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

Daftarkan Blueprint

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

Pembuatan URL Blueprint

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

Referensi Cepat Flask

Struktur Blueprint

url_prefix	Prefix semua route dalam blueprint
template_folder	Direktori template kustom
static_folder	File statis khusus blueprint
@bp.before_request	Jalankan sebelum setiap request blueprint

Penanganan Error

Halaman Error Kustom

```
@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
```

Batalkan Request

```
from flask import abort

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

Exception Kustom

```
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)
```

Konfigurasi

Metode Konfigurasi

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

Pola 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'
```

Pengaturan Umum

SECRET_KEY	Kunci signing session (wajib)
DEBUG	Aktifkan mode debug
TESTING	Aktifkan mode test
SQLALCHEMY_DATABASE_URI	String koneksi database
MAX_CONTENT_LENGTH	Ukuran upload maksimum dalam byte
JSON_SORT_KEYS	Urutkan key output JSON

Extension

Extension Populer

Flask-SQLAlchemy	Integrasi ORM
Flask-Migrate	Migrasi database Alembic
Flask-WTF	Penanganan form dengan CSRF
Flask-Login	Manajemen sesi pengguna
Flask-Mail	Pengiriman email
Flask-CORS	Cross-origin resource sharing
Flask-RESTful	Pembuatan REST API
Flask-Caching	Caching response dan fungsi

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
```