Pemrograman Web: Flask

Ali Akbar Septiandri

Universitas Al-Azhar Indonesia

aliakbars@live.com

March 13, 2017

Overview

1. Initial Setup
2. Flask
   - Hello, World
   - Routing
   - Views
3. Object Relational Mapper
   - Mengakses Database
   - SQL Alchemy
4. Django

Bahan Bacaan

1. Flask Documentation (8 Maret 2017). `http://flask.pocoo.org/docs/0.12/.latex/Flask.pdf`
2. `http://flask.pocoo.org/`
3. Grinberg, M. (2014). Flask web development: Developing web applications with Python. O'Reilly Media, Inc.
4. `https://www.sqlalchemy.org/`

Initial Setup

Virtual Environments

- Terkadang, kita perlu mengelola beberapa proyek dengan *dependencies* yang berbeda versi
- Perlu dilakukan isolasi lingkungan pemrograman
- Salah satu solusinya adalah dengan menggunakan **virtual environments**

`virtualenv` dan `conda`

- Dua kakas yang sering digunakan untuk *virtual environments* adalah `virtualenv` dan `conda`
- `conda` bersifat lebih generik dibandingkan `virtualenv` — mendukung bahasa selain Python
- Keduanya bisa mengisolasi lingkungan pemrograman, versi paket yang digunakan bisa dicatat dan dikelola dengan baik

Flask

Microframework

Gambar : Logo Flask

- Flask adalah sebuah contoh **microframework** — lawannya adalah *full-stack framework*, e.g. Django

Microframework

Gambar : Logo Flask

- Flask adalah sebuah contoh **microframework** — lawannya adalah *full-stack framework*, e.g. Django
- Tidak banyak modul yang disediakan: tidak ada modul otentikasi, abstraksi basis data, atau sanitasi masukan

Microframework

Gambar : Logo Flask

- Flask adalah sebuah contoh **microframework** — lawannya adalah *full-stack framework*, e.g. Django
- Tidak banyak modul yang disediakan: tidak ada modul otentikasi, abstraksi basis data, atau sanitasi masukan
- Contoh *microframeworks*: Bottle untuk Python, Sinatra untuk Ruby, Lumen untuk PHP

Flask

Flask mencoba membuat *deployment* aplikasi web semudah menggunakan PHP dengan LAMP stack.

Example (main.py)

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

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Struktur Direktori

Untuk aplikasi yang sederhana, struktur direktori Flask dapat berbentuk sebagai berikut.

Direktori

```
/yourapplication
    yourapplication.py
    /static
        style.css
    /templates
        layout.html
        index.html
        login.html
        ...
```

Routing

- Aplikasi web modern menggunakan URL yang “cantik”

Routing

- Aplikasi web modern menggunakan URL yang “cantik”
- Penamaan URL ini memudahkan untuk mengakses langsung ke halaman yang dituju dari sebuah situs

Routing

- Aplikasi web modern menggunakan URL yang “cantik”
- Penamaan URL ini memudahkan untuk mengakses langsung ke halaman yang dituju dari sebuah situs
- Flask menggunakan *decorator* untuk mengatur *routing*

Routing

- Aplikasi web modern menggunakan URL yang “cantik”
- Penamaan URL ini memudahkan untuk mengakses langsung ke halaman yang dituju dari sebuah situs
- Flask menggunakan *decorator* untuk mengatur *routing*
- URL tersebut juga dapat menerima variabel

Contoh Routing pada Flask

Example

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

@app.route('/hello')
def hello():
    return 'Hello, World'

@app.route('/hello/<username>')
def show_profile(username):
    return 'Hello, {}'.format(username)
```

HTTP Methods

- Ada beberapa metode HTTP untuk mengakses URL
- Metode ini dapat diganti pada dekorator `route()`
- Metode yang umum digunakan: `GET`, `POST`, `PUT`, `DELETE`

Reversing URL

- Di level kode, merujuk pada nama fungsi untuk menghasilkan URL yang dibutuhkan dapat lebih deskriptif

Reversing URL

- Di level kode, merujuk pada nama fungsi untuk menghasilkan URL yang dibutuhkan dapat lebih deskriptif
- URL tersebut dapat berubah sesuai kebutuhan, tetapi dengan nama fungsi yang sama

Reversing URL

- Di level kode, merujuk pada nama fungsi untuk menghasilkan URL yang dibutuhkan dapat lebih deskriptif
- URL tersebut dapat berubah sesuai kebutuhan, tetapi dengan nama fungsi yang sama
- Bisa digunakan untuk menghindari *escaping of special characters*

Reversing URL

- Di level kode, merujuk pada nama fungsi untuk menghasilkan URL yang dibutuhkan dapat lebih deskriptif
- URL tersebut dapat berubah sesuai kebutuhan, tetapi dengan nama fungsi yang sama
- Bisa digunakan untuk menghindari *escaping of special characters*
- Menggunakan fungsi `url_for()`

Reversing URL

- Di level kode, merujuk pada nama fungsi untuk menghasilkan URL yang dibutuhkan dapat lebih deskriptif
- URL tersebut dapat berubah sesuai kebutuhan, tetapi dengan nama fungsi yang sama
- Bisa digunakan untuk menghindari *escaping of special characters*
- Menggunakan fungsi `url_for()`
- `url_for()` juga digunakan untuk menampilkan *static files*

Rendering Templates

- *"Generating HTML from within Python is not fun, and actually pretty cumbersome..."*

Rendering Templates

- "*Generating HTML from within Python is not fun, and actually pretty cumbersome...*"
- Kita dapat menggunakan *templates*

Rendering Templates

- "*Generating HTML from within Python is not fun, and actually pretty cumbersome...*"
- Kita dapat menggunakan *templates*
- Metode untuk me-*render*-nya dapat menggunakan `render_template()`

Rendering Templates

- "*Generating HTML from within Python is not fun, and actually pretty cumbersome...*"
- Kita dapat menggunakan *templates*
- Metode untuk me-*render*-nya dapat menggunakan `render_template()`
- File yang akan di-*render* disimpan dalam folder `templates`

Templating Engine

Gambar : Logo Jinja

- Flask menggunakan *templating engine* untuk memudahkan pembuatan **web dinamis**

Templating Engine

Gambar : Logo Jinja

- Flask menggunakan *templating engine* untuk memudahkan pembuatan **web dinamis**
- Contoh kasus: pembuatan *dashboard* yang memiliki *sidebar* yang sama

Templating Engine

Gambar : Logo Jinja

- Flask menggunakan *templating engine* untuk memudahkan pembuatan **web dinamis**
- Contoh kasus: pembuatan *dashboard* yang memiliki *sidebar* yang sama
- Secara bawaan, *templating engine* untuk Flask adalah **Jinja2**

Contoh Jinja2

Example

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

Membuat Halaman Login Sederhana

Kebutuhan utama:

- Mengakses objek `request` (data masukan dari pengguna)
- *Redirects* dan halaman error
- Penyimpanan *sessions*

Let's see the codes!

Object Relational Mapper

Database Interfaces

Python punya *interfaces* untuk menghubungkan dengan bermacam-macam DBMS, antara lain:

1. MySQL
2. PostgreSQL
3. Microsoft SQL Server
4. SQLite
5. CouchDB
6. MongoDB
7. dsb.

MySQL Interface

Untuk menghubungkan dengan MySQL saja, ada beberapa pilihan *connectors*:

1. `MySQLdb` (legacy version)
2. `mysqlclient` (fork dari MySQLdb)
3. `mysql-connector` (dimuat dalam dokumentasi resmi MySQL)

Demo

Pengenalan ORM

- Pada dasarnya, setiap tabel dalam basis data dapat dianggap sebagai kelas

Pengenalan ORM

- Pada dasarnya, setiap tabel dalam basis data dapat dianggap sebagai kelas
- *Object-relational mapper* (ORM) berfungsi untuk **memetakan kelas ke basis data**

Pengenalan ORM

- Pada dasarnya, setiap tabel dalam basis data dapat dianggap sebagai kelas
- *Object-relational mapper* (ORM) berfungsi untuk **memetakan kelas ke basis data**
- Dengan demikian **kolom pada tabel** dapat diakses seperti mengakses **atribut suatu kelas**

Pengenalan ORM

- Pada dasarnya, setiap tabel dalam basis data dapat dianggap sebagai kelas
- *Object-relational mapper* (ORM) berfungsi untuk **memetakan kelas ke basis data**
- Dengan demikian **kolom pada tabel** dapat diakses seperti mengakses **atribut suatu kelas**
- ORM juga memungkinkan untuk manipulasi antartabel, e.g. join

Pengenalan ORM

- Pada dasarnya, setiap tabel dalam basis data dapat dianggap sebagai kelas
- *Object-relational mapper* (ORM) berfungsi untuk **memetakan kelas ke basis data**
- Dengan demikian **kolom pada tabel** dapat diakses seperti mengakses **atribut suatu kelas**
- ORM juga memungkinkan untuk manipulasi antartabel, e.g. join
- Diharapkan dapat mengurangi tugas yang *redundant*!

Flask-SQLAlchemy

Example

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////tmp/test.db'
db = SQLAlchemy(app)


class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    email = db.Column(db.String(120), unique=True)

    def __init__(self, username, email):
        self.username = username
        self.email = email

    def __repr__(self):
        return '<User %r>' % self.username
```

Inserting & Deleting Records

Example (insert)

```
>>> from yourapp import User
>>> me = User('admin', 'admin@example.com')
>>> db.session.add(me)
>>> db.session.commit()
```

Example (delete)

```
>>> db.session.delete(me)
>>> db.session.commit()
```

Querying Records

Example (filter)

```
>>> peter = User.query.filter_by(username='peter').first()
>>> peter.id
1
>>> peter.email
u'peter@example.org'
```

Example (complex filter)

```
>>> User.query.filter(User.email.endswith('@example.com')).all()
[<User u'admin'>, <User u'guest'>]
```

Querying Records

Example (ordering & limit)

```
>>> User.query.order_by(User.username)
[<User u'admin'>, <User u'guest'>, <User u'peter'>]
>>> User.query.limit(1).all()
[<User u'admin'>]
```

Example (primary key)

```
>>> User.query.get(1)
<User u'admin'>
```

Demo

Django

Django: A Full-stack Framework

Django makes it easier to build better Web apps more quickly and with less code.

Model-View-Template

- Django membagi arsitektur aplikasi dengan lebih jelas sebagai **Model-View-Template (MVT)**
- Dalam bahasa yang lebih generik, model arsitektur ini biasa dikenal juga dengan **Model-View-Controller (MVC)**
- Prinsip utama: "Don't repeat yourself" (DRY)

Full-stack Framework

Sebagai *full-stack framework*, beberapa komponen yang sudah masuk dalam lingkup Django dibandingkan Flask:

- Model (tidak perlu instalasi SQLAlchemy lagi)
- Validasi form
- Administrasi
- Keamanan aplikasi, e.g. proteksi CSRF dan SQL injection
- Pengiriman e-mail

https://www.djangoproject.com/

Terima kasih