


Flask Micro Framework

Mohammad Reza Kamalifard





@itmard



Codes

https://github.com/itmard/zconf_flask_intro


 itmard / **zconf_flask_intro** Unwatch ▾ 1






Introduction to Flask Micro Framework for ZCONF 5 — Edit

 3 commits  1 branch  0 releases  1 contributor

 branch: **master** ▾ **zconf_flask_intro** / + 

new structure

 Mohammad Reza Kamalifard authored a day ago latest commit 043c5a89dd

 code	new structure	a day ago
 paper	new structure	a day ago
 slide	new structure	a day ago
 .gitignore	Initial commit	20 hours ago
 LICENSE	Initial commit	20 hours ago

What is a Web Application Framework?

What is a Web Application Framework?

- URL Routing

What is a Web Application Framework?

- URL Routing
- Request and Response Objects

What is a Web Application Framework?

- URL Routing
- Request and Response Objects
- Template Engine

What is a Web Application Framework?

- URL Routing
- Request and Response Objects
- Template Engine
- Development Web Server

What is a Web Application Framework?

- URL Routing
- Request and Response Objects
- Template Engine
- Development Web Server
- Database Object Relational Mapper

What is a Web Application Framework?

- URL Routing
- Request and Response Objects
- Template Engine
- Development Web Server
- Database Object Relational Mapper
- Form Validation

Why Python?

- Python is an easy to learn
- Powerful
- Clean syntax and code readability
- Open Source
- Cross-platform
- Rich set of libraries
- Large Number of open source tools

Python web frameworks

- Django
- Flask
- Werkzeug
- Tornado
- Pyramid
- Bottle
- ...

Flask

Flask

- Micro Framework

Flask

- Micro Framework
- Simple but extensible core

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit
- Jinja2 template engine

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit
- Jinja2 template engine
- Provides a simple template for web development

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit
- Jinja2 template engine
- Provides a simple template for web development
- No ORM

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit
- Jinja2 template engine
- Provides a simple template for web development
- No ORM
- No Form validation

Flask

- Micro Framework
- Simple but extensible core
- Werkzeug WSGI toolkit
- Jinja2 template engine
- Provides a simple template for web development
- No ORM
- No Form validation
- Supports extensions

Simple Web Application with Flask

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

@app.route("/")
def hello():
    return "Welcome to ZCONF 5"

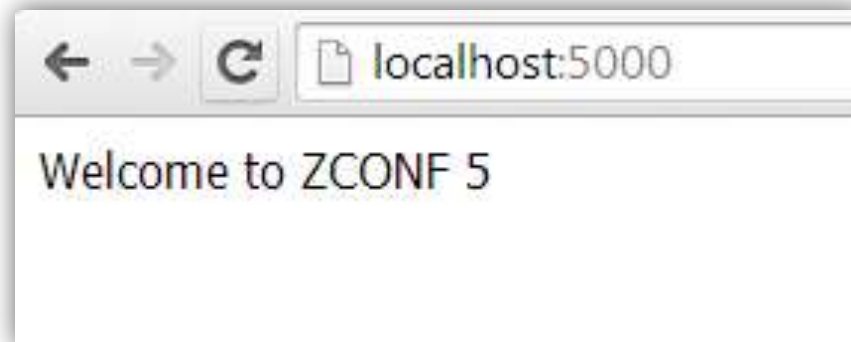
if __name__ == "__main__":
    app.run()
```

Simple Web Application with Flask

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

@app.route("/")
def hello():
    return "Welcome to ZCONF 5"

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



Features

- Built in development server and debugger
- Integrated unit testing support
- RESTful request dispatching
- Uses Jinja2 templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode based
- Extensively documented

Flask Routing

- Modern web applications have beautiful URLs
- Routing using decorators
- `route()` decorator is used to bind a function to a URL
- Make certain parts of the URL dynamic
- Attach multiple rules to a function

Flask Routing

- Modern web applications have beautiful URLs
- Routing using decorators
- `route()` decorator is used to bind a function to a URL
- Make certain parts of the URL dynamic
- Attach multiple rules to a function

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

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

Variable Rules

- <variable_name>

```
@app.route('/user/<username>')  
def show_user_profile(username):  
    return 'User %s' % username
```

- <converter:variable_name>

```
@app.route('/post/<int:post_id>')  
def show_post(post_id):  
    return 'Post %d' % post_id
```

Request Object

- Easy way to access request data

```
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'], request.form['password']):
            return log_the_user_in(request.form['username'])
    else:
        error = 'Invalid username/password'
    return render_template('login.html', error=error)
```

Template Engine

- Jinja2 (default)

```
from flask import render_template
@app.route('/hello/ ')
@app.route('/hello/<name> ')
def hello(name=None):
    return render_template('hello.html', name=name)
```

hello.html

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
<h1>Hello {{ name }}!</h1>
{% else %}
<h1>Hello World!</h1>
{% endif %}
```

Development server and debugger

- Easy to use local development server
- Interactive Error page with debug mode

TypeError

TypeError: cannot concatenate 'str' and 'NoneType' objects

Traceback (most recent call last)

File `"/Users/mitsuhiko/Development/flask/flask.py"`, line 650, in `__call__`

```
    return self.wsgi_app(environ, start_response)
```

File `"/Users/mitsuhiko/Development/werkzeug-main/werkzeug/wsgi.py"`, line 406, in `__call__`

```
    return self.app(environ, start_response)
```

File `"/Users/mitsuhiko/Development/flask/flask.py"`, line 616, in `wsgi_app`

```
    rv = self.dispatch_request()
```

File `"/Users/mitsuhiko/Development/flask/flask.py"`, line 535, in `dispatch_request`

```
    return self.view_functions[endpoint](**values)
```

File `"/Users/mitsuhiko/Development/flask/test.py"`, line 8, in `index`

```
    return 'Hello ' + name
```

```
[console ready]
```

```
>>> type(name)
```

```
<type 'NoneType'>
```

```
>>> 
```

Database management

- No ORM inside flask
- Extensions:
 - SQLAlchemy
 - Peewee
 - Mongoengine
 - MongoKIT
 - PyMongo


```
#sqlalchemy
from flask import Flask
from flask.ext.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)
```

```
#peewee
```

```
import datetime
from peewee import *
```

```
class Note(db.Model):
    message = TextField()
    created = DateTimeField(default=datetime.datetime.now)
```

Form validation

- Flask WTFORM

#class based forms

```
from flask_wtf import Form
from wtforms import StringField
from wtforms.validators import DataRequired
class MyForm(Form):
    name = StringField('name', validators=[DataRequired()])
#view
@app.route('/submit', methods=('GET', 'POST'))
def submit():
    form = MyForm()
    if form.validate_on_submit():
        return redirect('/success')
    return render_template('submit.html', form=form)
```

Architectural Pattern

- Single file application
- MVC , MV* patterns

project

---apps

-----app1

-----models.py

-----views.py

-----forms.py

-----apis.py

---statics

---templates

-----models

-----views

-----forms

-----statics

-----templates

Mohammad Efazati

Mehdi Khoshnody

Reza Shalbafzadeh

K1 Hedayati

Thank you :)



Copyright 2014 Mohammad Reza Kamalifard
All rights reserved.

This work is licensed under the Creative Commons Attribution-NoDerivs 3.0
Unported License.

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-nd/3.0/>