**Flaskの簡単な使い方**

[Python](https://qiita.com/tags/python)[Flask](https://qiita.com/tags/flask)

144

この記事は最終更新日から1年以上が経過しています。

Pythonの軽量フレームワーク。[Flask](http://flask.pocoo.org/" \t "_blank)の最低限の使い方。  
とりあえず自分が使う範囲を調べてメモ。

後は[QuickStart](http://flask.pocoo.org/docs/0.12/quickstart/" \t "_blank)をやれば一通りわかる。  
Djangoな人は[こちら](http://qiita.com/zaburo/items/ab7f0eeeaec0e60d6b92)もどうぞ。

**準備と簡単な使い方**

**環境**

MacでPython3を入れて、venvで仮想環境を作成した状態。

**インストール**

pip install Flask

**Hello World**

作業フォルダを作成。とりあえずどこでも、なんでもよい。

cd

mkdir flask

cd flask

hello.pyという名前でコードを書く。

hello.py

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

name = "Hello World"

return name

@app.route('/good')

def good():

name = "Good"

return name

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**実行**

実行してみる。

python hello.py

下記URLにアクセスしてみます。

<http://localhost:5000/>   
<http://localhost:5000/good>

無事表示されました。簡単。

**Jinja2を利用する**

Flaskには標準でJinja2というテンプレートエンジンが付いているので使ってみます。  
templatesというフォルダを作る。そこに、htmlを置きます。

**共通テンプレート**

まずは共通テンプレートから。フッターとかヘッダーとか。ここではかなり手を抜きます。  
書き方はDjangoとよく似ていますね。

layout.html

<!doctype html>

<html>

<head>

<title>{{ title }}</title>

<body>

{% block content %}

<!-- ここにメインコンテンツを書く -->

{% endblock %}

</body>

</head>

**個別ページ**

次に個別ページ。 {% block content %}の内容等を定義します。

hello.html

{% extends "layout.html" %}

{% block content %}

<h3>Hello</h3>

こんにちは。{{ name }}さん。

{% endblock %}

**hello.pyの変更**

テンプレートを利用し、かつ、変数を渡すようにhello.pyを変更します。

hello.py

from flask import Flask, render\_template #追加

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

name = "Hoge"

#return name

return render\_template('hello.html', title='flask test', name=name) #変更

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

変更したら <http://localhost:5000/> にアクセスしてみます。

**DB(MySQL)を利用する**

せっかくなのでデータベース（MySQL）を利用してみます。  
まずはhello.pyをいじってみます。なおMySQLへの接続にはPyMySQLを利用します。

インストールがまだであれば、

pip install PyMySQL

でインストールして下さい。

**ロジックの実装**

ロジックというほどではありませんが、下記のようにしました。  
接続情報等は共通部品にするとか、with句を使うとかした方がいいと思いますがとりあえず。。。

hello.py

from flask import Flask, render\_template #追加

import pymysql #追加

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

#db setting

db = pymysql.connect(

host='localhost',

user='root',

password='root',

db='testdb',

charset='utf8',

cursorclass=pymysql.cursors.DictCursor,

)

cur = db.cursor()

sql = "select \* from members"

cur.execute(sql)

members = cur.fetchall()

cur.close()

db.close()

#return name

return render\_template('hello.html', title='flask test', members=members) #変更

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**個別ページ変更**

内容をforでループして表示させてみます。簡単です。

hello.html

{% extends "layout.html" %}

{% block content %}

<h3>一覧</h3>

<ul>

{% for member in members %}

<li>{{ member.name}} : {{ member.email }}</li>

{% endfor %}

</ul>

{% endblock %}

うまくいきました。

**パラメーター連携**

次に、簡単なパラメーター連動を。

**URLから**

モダン。URLから値を取得してみます。

hello.py

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/hello/<name>')

def hello(name=None):

#return name

return render\_template('hello.html', title='flask test', name=name)

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

下記のようなURLでアクセスしてみます。

<http://localhsot:5000/hello/hoge>

**POSTから**

次にPOST。formのHTMLは省いています。普通にpostするfromを作ってpostしてみてください。  
メソッドを明示しないと「Method not allowed」的なエラーが出るようです。

from flask import Flask, render\_template, request #追加

app = Flask(\_\_name\_\_)

@app.route('/hello', methods=['POST']) #Methodを明示する必要あり

def hello():

if request.method == 'POST':

name = request.form['name']

else:

name = "no name."

return render\_template('hello.html', title='flask test', name=name)

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**GETから**

最後にGETというかQueryString。

hello.py

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/hello')

def hello():

name = request.args.get('name')

return render\_template('hello.html', title='flask test', name=name)

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

下記のようなURLでアクセスしてみます。

<http://127.0.0.1:5000/hello?name=hoge>

**JSONを返す**

最後にJSONを返してみたいと思います。  
jsonifyを利用すれば簡単なのですが、日本語文字化け対応とソート順対応をしています。  
正しいかどうか微妙ですがとりあえず。

**手動生成したディクショナリを返す**

hello.py

from flask import Flask, render\_template, request, jsonify #追加

app = Flask(\_\_name\_\_)

app.config['JSON\_AS\_ASCII'] = False #日本語文字化け対策

app.config["JSON\_SORT\_KEYS"] = False #ソートをそのまま

@app.route('/hello')

def hello():

data = [

{"name":"山田"},

{"age":30}

]

return jsonify({

'status':'OK',

'data':data

})

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**DBの結果を返す**

DBのSELECT結果を返してみたいと思います。  
上記の例で、DBの接続情報部分を汎用化してみました。

hello.py

from flask import Flask, render\_template, jsonify #追加

import pymysql #追加

app = Flask(\_\_name\_\_)

app.config['JSON\_AS\_ASCII'] = False #日本語文字化け対策

app.config["JSON\_SORT\_KEYS"] = False #ソートをそのまま

def getConnection():

return pymysql.connect(

host='localhost',

user='root',

password='root',

db='testdb',

charset='utf8',

cursorclass=pymysql.cursors.DictCursor,

)

@app.route('/')

def hello():

db = getConnection()

cur = db.cursor()

sql = "select \* from members"

cur.execute(sql)

members = cur.fetchall()

cur.close()

db.close()

return jsonify({

'status':'OK',

'members':members

})

## おまじない

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

下記のようなJSONが返ります。

{

"status": "OK",

"members": [

{

"id": 1,

"name": "hoge",

"email": "hoge@hoge.com"

},

{

"id": 2,

"name": "foo",

"email": "foo@foo.com"

}

]

}

以上。また必要に合わせて追記したいと思います。