Flask

[Python][Angular]

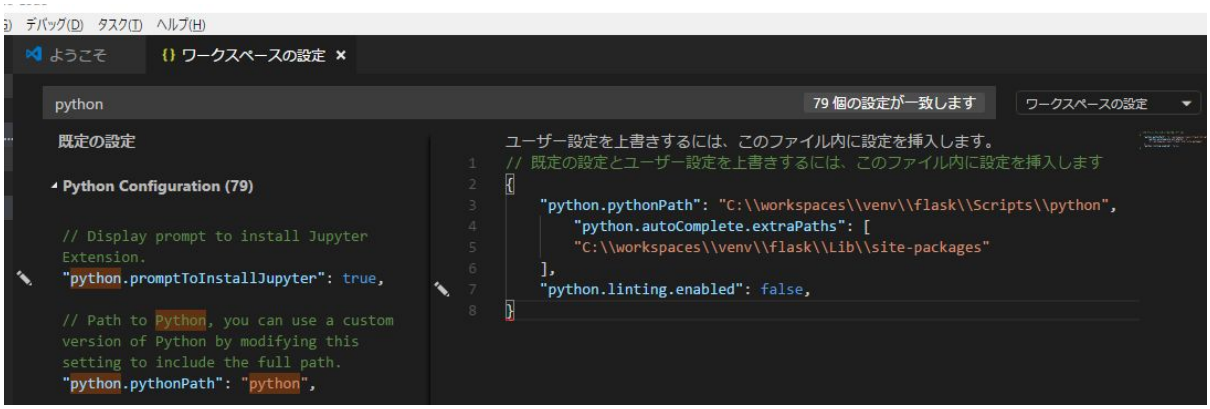
- http://flask.pocoo.org/

インストール

- http://flask.pocoo.org/docs/0.12/installation/
- Windows , python3

```
> pythoh -m venv flask
> cd flask\Scripts
> activate
(flask)> pip install Flask
```

Visual Studio Code




```
// 既定の設定とユーザー設定を上書きするには、このファイル内に設定を挿入します
{
    "python.pythonPath": "C:\\workspaces\\venv\\flask\\Scripts\\python",
        "python.autoComplete.extraPaths": [
        "C:\\workspaces\\venv\\flask\\Lib\\site-packages"
    ],
    "python.linting.enabled": false,
}
```

Git

- Github にリポジトリ作成
  - https://github.com/pppiroto/flask_sample.git

ローカルリポジトリを初期化

```
> git init
```

ローカルリポジトリにコミット

```
> git add .
> git commit -m "flask lesson init"
> git branch
* master
```

リモートリポジトリの設定

```
> git remote add origin https://github.com/pppiroto/flask_sample.git
```

リモートリポジトリに push

```
> git push origin master
```

クイックスタート

・http://flask.pocoo.org/docs/0.12/quickstart/

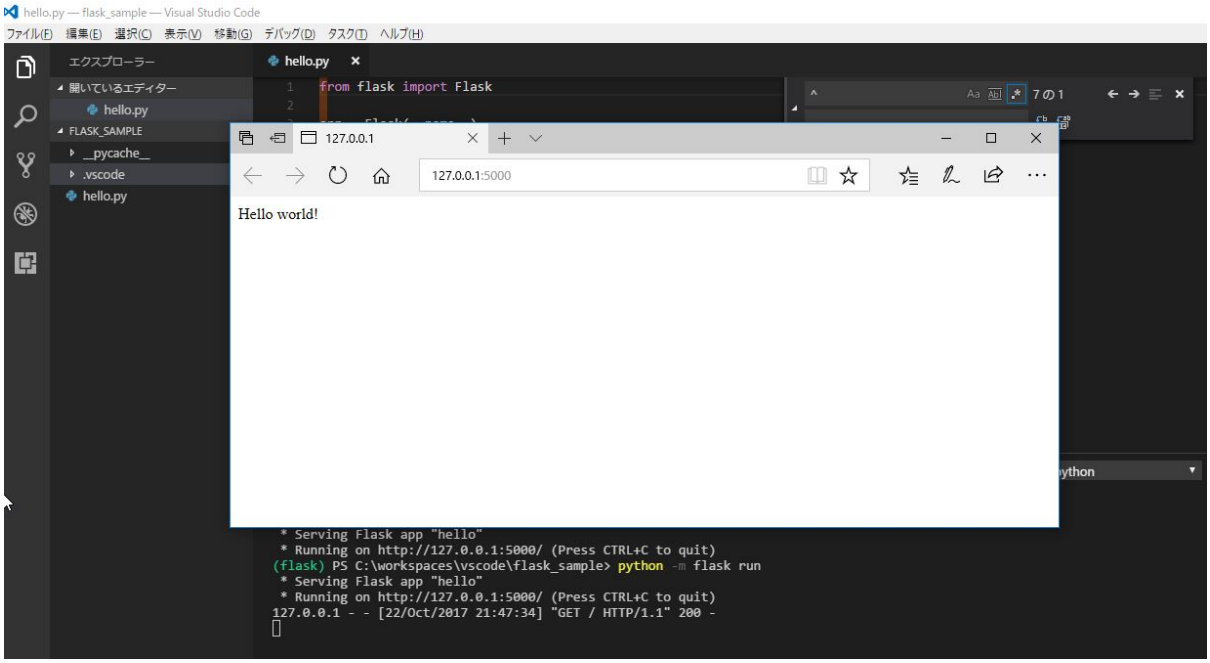
/hello.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return "Hello world!"
```

サーバー起動

・Visual Studio Code、Powershell

```
(flask) PS C:\workspaces\vscode\flask_sample> set-item env:FLASK_APP hello.py
(flask) PS C:\workspaces\vscode\flask_sample> python -m flask run
 * Serving Flask app "hello"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



■デバッグモード

```
PS> set-item env:FLASK_DEBUG 1
```

ルーティング

```
from flask import Flask

app = Flask(__name__)

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

@app.route('/hello')
def hello_world():
    return 'Hello world!'
```

変数

```
@app.route('/user/<username>/<int:age>')
def show_user_profile(username, age):
    return 'User {0} {1}'.format(username,age)
```

< コンバーター : 変数 >

コンバーター	説明
string	'/' 以外のすべてのテキスト (デフォルト)
int	整数
float	浮動小数点数
path	'/' を受け付ける
any	与えられるアイテムの一つに一致
uuid	UUID 文字列

ユニーク URL / リダイレクトの振る舞い

- Flask の URL ルールは、Werkzeug のルーティングモジュールに基づく
- このモジュールの背後にあるアイディアは、Apache など過去の HTTP サーバーの先例にもとづいた、美しくユニークな URL ををを保証すること。

```
@app.route('/projects/')
def projects():
    return 'The project page'


@app.route('/about')
def about():
    return 'The about page'
```

- これらは同じように見えるが、末尾の / の扱いが異なる。
- projects エンドポイントのための正統な URL は末尾に / を伴う。ファイルシステムのフォルダと同様
- 末尾の / がない状態で、アクセスした場合、Flask は末尾に / がある正統な URL にリダイレクトする
- 末尾の / ない例は、ファイルシステムのファイルと同様、末尾に / を伴ってアクセスされた場合、404 Not Found エラーとなる

URL 生成

・url_for() 関数は、関数名を最初の、関数の引数を以降の引数としてとり、URL を生成する

```
from flask import Flask,url_for

app = Flask(__name__)

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

@app.route('/hello')
def hello_world():
    return 'Hello world!'

@app.route('/user/<username>/<int:age>')
def show_user_profile(username, age):
    return 'User {0} {1}'.format(username,age)

@app.route('/urls')
def print_urls():
    return '''
        <ol>
        <li>index() = {0}
        <li>hello_world() = {1}
        <li>show_user_profile('piroto',46) = {2}
        </ol>
    '''.format(
        url_for('index'),
        url_for('hello_world'),
        url_for('show_user_profile',username='piroto',age=46))
```

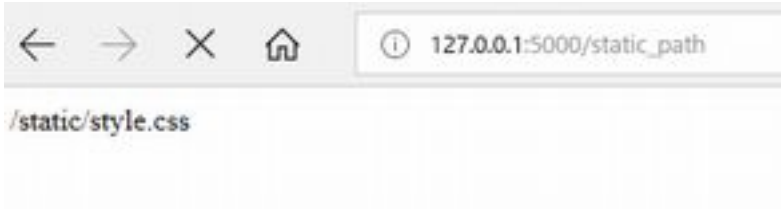
HTTP メソッド

・

```
from flask import request

@app.route('/method_check', methods=['GET','POST','HEAD'])
def method_check():
    if request.method == 'GET':
        return "HTTP METHOD GET"
    return "HTTP METHOD {0}".format(request.method)
```

静的ファイル

```
@app.route('/static_path')
def print_static_file_url():
    return url_for('static', filename='style.css')
```



テンプレート

・Flask では、Jinja2 テンプレートエンジンが自動で設定される
・アプリケーションがモジュールか、パッケージ化によって、以下の構成の templetes フォルダに配置する

モジュール

```
/アプリケーション.py
/templates
    /テンプレート.html
```

パッケージ

```
/アプリケーション
    /__init__.py
    /templates
        /テンプレート.html
```

テンプレート利用

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

テンプレート

```
<!doctype html>
<title>Rendering Sample</title>
{% if param %}
  <h1>Parameter : {{ param }}!</h1>
{% else %}
  <h1>No parameter found.</h1>
{% endif %}
```

・テンプレートでは、request 、session、および g オブジェクトにアクセスできる
・g オブジェクトは、必要に応じて情報を格納できる
・テンプレートは特に [継承 |]http://flask.pocoo.org/docs/0.12/patterns/templateinheritance/#template-inheritance される場合有用
・基本的にテンプレートは各ページにヘッダーやナビゲーションなどを確実に表示する

■自動エスケープ

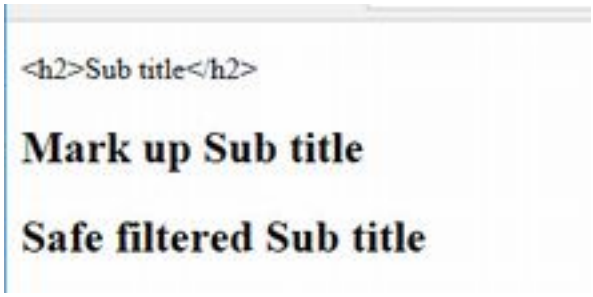
・自動エスケープは有効

・安全と確信できる場合、Markup クラスもしくは、 |safe フィルターを使用する

テンプレート

```
<!doctype html>
<title>Safe Rendering Sample</title>
<p>{{ param1 }}</p>
<p>{{ param2 }}</p>
<p>{{ param3 | safe}}</p>
```

```
from flask import Markup
@app.route('/render_safe_var')
def render_safe_var(param=None):
    return render_template('render_safe_var.html',
        param1="<h2>Sub title</h2>",
        param2=Markup("<h2>Mark up Sub title</h2>"),
        param3="<h2>Safe filtered Sub title</h2>")
```



リクエストデータへのアクセス

・クライアントが送信したリクエストデータは、request グローバルオブジェクトにより提供される
・context locals により、グローバルオブジェクトをどうやってスレッドセーフにし管理している

■ Context　Locals

・Flask のグローバルオブジェクトは実際には特定コンテキストローカルオブジェクトのプロキシ

Tips

Session

・http://flask.pocoo.org/docs/0.12/quickstart/#sessions

```
from flask import Flask, session
app = Flask(__name__, static_folder='app')
app.secret_key = "hoge"

def use_session:
    sessio["foo"] = "bar"
```

CRSF

・http://flask.pocoo.org/snippets/3/

・Angular + HttpClientXsrfModule + Flask で CSRF

Cookie

・http://flask.pocoo.org/snippets/30/
・https://stackoverflow.com/questions/37068604/flask-sessions-where-are-the-cookies-stored

テンプレート

■ディレクトリを変える

・http://d.hatena.ne.jp/aroma_black/20101108/1289226596