

# PyCon JP 2012

## Hands on session

FlaskによるWebアプリケーションの実装と  
プログラミングツール



Atsuo Ishimoto

# 講師陣 - 質問はこちらまで



@atsuoishimoto



@jbking



@feiz

## 近くの席の人とも話し合ってみよう！

# 本日のメニュー

- 一応、Pythonの書き方は知ってる人を、「使いこなせる」段階に
- 文法の知識の次に必要な、実践的なテクニックを実習

# 環境設定

- 無線Lanつながってますか？
- Python2.6 or 2.7
- Flask環境のインストール

# FlaskによるWebアプリケーション

- 30分
- 単純なWebアプリケーション
- Flaskとは
  - @mitsuhiko /Armin Ronacher
  - 「マイクロフレームワーク」
  - シンプル
  - 拡張性重視
  - jinja2

# profileによるパフォーマンス測定

- 10分
- PythonのcProfileモジュールの使い方
- ソースコードのトレース
- ボトルネックの検出

# loggingモジュール

- 10分
- Loggingモジュールの使い方

# デバッガの使い方

- 20分
- pdbモジュールの使い方
- スタック/トレースなどの実行環境の説明



# タイムチャート

10:00 - 10:05	アジェンダ
10:05 - 10:15	環境設定
10:15 - 10:45	Flaskアプリケーションの開発
10:45 - 11:00	休憩
11:00 - 11:10	loggingモジュール
11:10 - 11:20	traceによるパフォーマンス測定
11:20 - 11:40	デバッガの使い方
11:40 - 11:45	Q/A

- 早めに課題が終わった方は、先に進んでも結構です
- 余裕があったら、周囲の人を助けてあげよう！

# 環境設定

- 無線Lan、つながってますね?
  - [www.python.org](http://www.python.org) の接続を確認してください
- Python2.6 or 2.7 動きますね?
  - コンソールで python の起動を確認してください

```
c:¥>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32
bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
```

# パッケージ管理ツール (Linux/OS-X/Cygwin)

- Flaskをインストールするためのパッケージ管理ツールを用意します
- `easy_install` がインストール済みならそのまま使ってください
- 無ければ
  - [http://python-distribute.org/distribute\\_setup.py](http://python-distribute.org/distribute_setup.py)をダウンロード
  - `$sudo python distribute_setup.py`

# パッケージ管理ツール (Windows)

- Flaskをインストールするためのパッケージ管理ツールを用意します
- easy\_install がインストール済みならそのまま使ってください
- 無ければ
  - [http://python-distribute.org/distribute\\_setup.py](http://python-distribute.org/distribute_setup.py)をダウンロード
  - `C:\¥Python27¥python.exe distribute_setup.py`

# Flaskのインストール

- Windowsの場合

- `C:\¥Python27¥Scripts¥easy_install.exe flask`

- Unix系(Linux/OS-X/Cygwin)の場合

- `sudo easy_install flask`

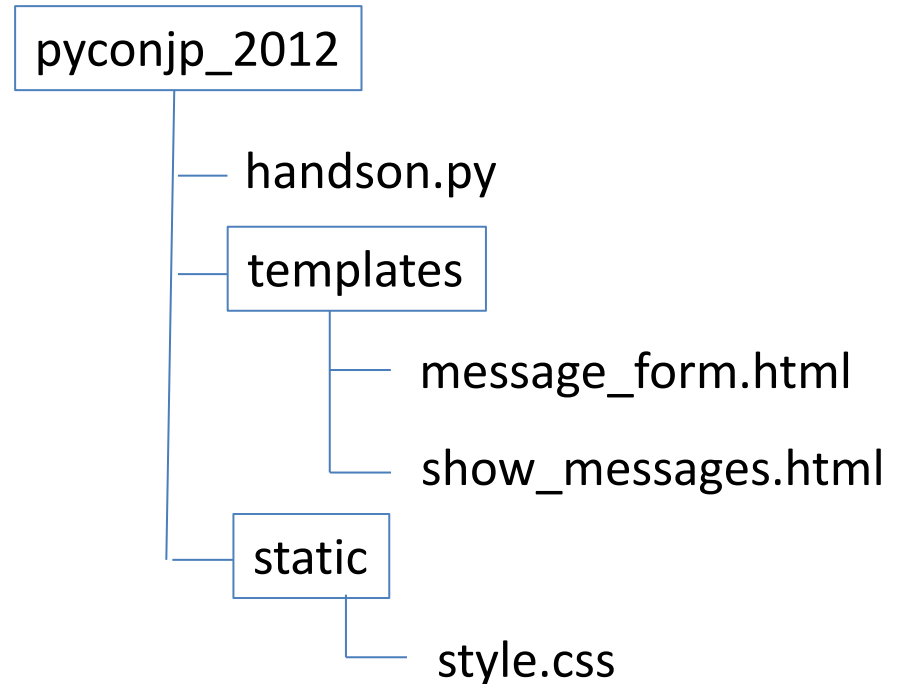
- 動作確認

- `python -c "import flask"` でエラーが出なければOK

# FlaskによるWebアプリケーション

## 1. プロジェクトディレクトリの作成

```
mkdir pyconjp_2012  
cd pyconjp_2012  
mkdir templates  
mkdir static
```



# FlaskによるWebアプリケーション

## 2. ソースファイルを編集

[https://github.com/atsuoishimoto/pyconjp\\_2012](https://github.com/atsuoishimoto/pyconjp_2012)

を参照してください。まじめに写経しても、コピペでもかまいません。

**文字コードはUTF-8で！**

Zipファイル

[https://github.com/atsuoishimoto/pyconjp\\_2012/zipball/master](https://github.com/atsuoishimoto/pyconjp_2012/zipball/master)

# handson.py

```
# -*- coding: utf-8 -*-

from flask import Flask, request, session
from flask import render_template, redirect, url_for

# Flaskのアプリケーション オブジェクトを作成
app = Flask(__name__)

# http://localhost:5000/でアクセスされる関数
@app.route('/')
def index_html():
    return """
<!doctype html>
<ul>
<li><a href="/message_form">メッセージ追加</a></li>
<li><a href="/show">メッセージ表示</a></li>
</ul>
"""

# /message_formでアクセスされる関数
@app.route('/message_form')
def message_form():
    # テンプレートファイル templates/message_form.htmlを表示
    return render_template('message_form.html')

# /add_messageでリクエストのメッセージを登録
@app.route('/add_message', methods=['POST'])
def add_message():
    # Sessionにメッセージを登録
    msgs = session.get('messages', [])
    msgs.append(request.form['message'])
    session['messages'] = msgs[-10:]
    return redirect(url_for('show_messages'))

# /showでリクエストのメッセージを登録
@app.route('/show')
def show_messages():
    # テンプレートファイル templates/show_messages.htmlを表示
    return render_template('show_messages.html',
        messages=reversed(session['messages']))

def main():
    app.secret_key = "secret"
    app.run(debug = True)

if __name__ == '__main__':
    main()
```



# message\_form.html

```
<!doctype html>
<link rel=stylesheet type=text/css href="{{ url_for('static',
filename='style.css') }}">
<title>メッセージ登録</title>

<h1>メッセージ登録</h1>
<form method="post" action="{{url_for('add_message')}}">
<div>メッセージ :
    <input type="text" name="message" size=40>
    <button type="submit">登録</button>
</div>

</form>
```

# show\_messages.html

```
<!doctype html>
<link rel=stylesheet type=text/css href="{{ url_for('static',
filename='style.css') }}">

<h1>登録済みメッセージ</h1>
{% for message in messages %}
    <div>{{ message }}</div>
{% endfor %}

<hr/>
<a
href="{{ url_for('message_form') }}">戻る</a>
```

# style.css

```
<!doctype html>
<link rel=stylesheet type=text/css href="{{ url_for('static',
filename='style.css') }}">
<title>メッセージ登録</title>

<h1>メッセージ登録</h1>
<form method="post" action="{{url_for('add_message')}}">
<div>メッセージ :
    <input type="text" name="message" size=40>
    <button type="submit">登録</button>
</div>

</form>
```

# FlaskによるWebアプリケーション

## 3. 実行

```
$python handson.py
```

ブラウザで <http://localhost:5000/> を開きます。

## 4. 終了方法

^C(Control+C) で終了します。

Windowsで、終了するまで時間がかかる場合、  
Control+Breakでも終了します。

# FlaskによるWebアプリケーション

## 5. ソースコード解説

```
app = Flask(__name__)
```

Flaskアプリケーションオブジェクトの作成

```
@app.route('/')  
def index_html():  
    return "Hello"
```

**@app.route('URL')** で、URLへの  
リクエストハンドラを指定

```
render_template(  
    'テンプレートファイル',  
    arg=vale)
```

Jinja2テンプレートを実行

```
app.secret_key = "secret"
```

セッションを利用するためのおまじない

# FlaskによるWebアプリケーション

## 6. jinja2テンプレート解説

```
<link rel=stylesheet type=text/css  
href="{{ url_for('static', filename='style.css') }}">
```

- **{{ 式 }}** で式をHTMLに展開
- **{{ url\_for(...) }}** でファイルへのURLを取得

```
{% for message in messages %}  
    <div>{{ message }}</div>  
{% endfor %}
```

**{% for x in xx %} ~ {% endfor %}** で for ループ

# FlaskによるWebアプリケーション

## 7. 自由演習

時間があったらどうぞ

- a. 各ページに、現在時刻を表示してみよう
- b. メッセージの長さチェック処理を入れてみよう
- c. メッセージをデータベースに格納してみよう

# cProfileモジュール

(注) Debian/Ubuntu では、  
`$sudo apt-get install python-profiler`  
が必要な場合があります

- profile/cProfileはPythonスクリプトの実行速度を測定するモジュールです
- 関数の呼び出し回数や処理時間を集計します
- profileとcProfileの機能はほぼ同じですが、C言語版のcProfileの方が高速です



# cProfileモジュール

- `show_messages()` 関数を修正します (Python2.6用)

```
@app.route('/show')
def show_messages():
    return render_template('show_messages.html',
                           messages=reversed(session['messages']))
```



```
@app.route('/show')
def show_messages():
    import cProfile
    localvars = locals().copy()
    cProfile.runctx("""ret = render_template('show_messages.html',
        messages=reversed(session['messages']))""",
        globals(), localvars)

    return localvars["ret"]
```

# cProfileモジュール

- `show_messages()` 関数を修正します (Python2.7用)

```
@app.route('/show')
def show_messages():
    return render_template('show_messages.html',
                           messages=reversed(session['messages']))
```



```
@app.route('/show')
def show_messages():
    import cProfile
    localvars = locals().copy()
    cProfile.runctx("""ret = render_template('show_messages.html',
        messages=reversed(session['messages']))""",
        globals(), localvars, sort='cumulative')

    return localvars["ret"]
```

# ローカル変数を参照するコードをプロファイル

```
def spam():  
    ham = 100  
    value = egg(ham)  
    return value
```



```
def spam():  
    ham = 100  
    localvars = locals().copy()  
    cProfile.runctx(  
        "value = egg(ham)",  
        globals(), localvars)  
    return localvars['value']
```

**cProfile.runctx() には、  
ローカル変数辞書をコピーして渡す**

locals() が返す辞書 (sys.\_getframe().f\_locals) は、  
インタプリタの都合で初期化されてしまう場合があるため

**更新されたローカル変数は、コピー  
した辞書から値を取得する**

# cProfileモジュール

## 実行結果

```
4569 function calls (4272 primitive calls) in 0.005 seconds
```

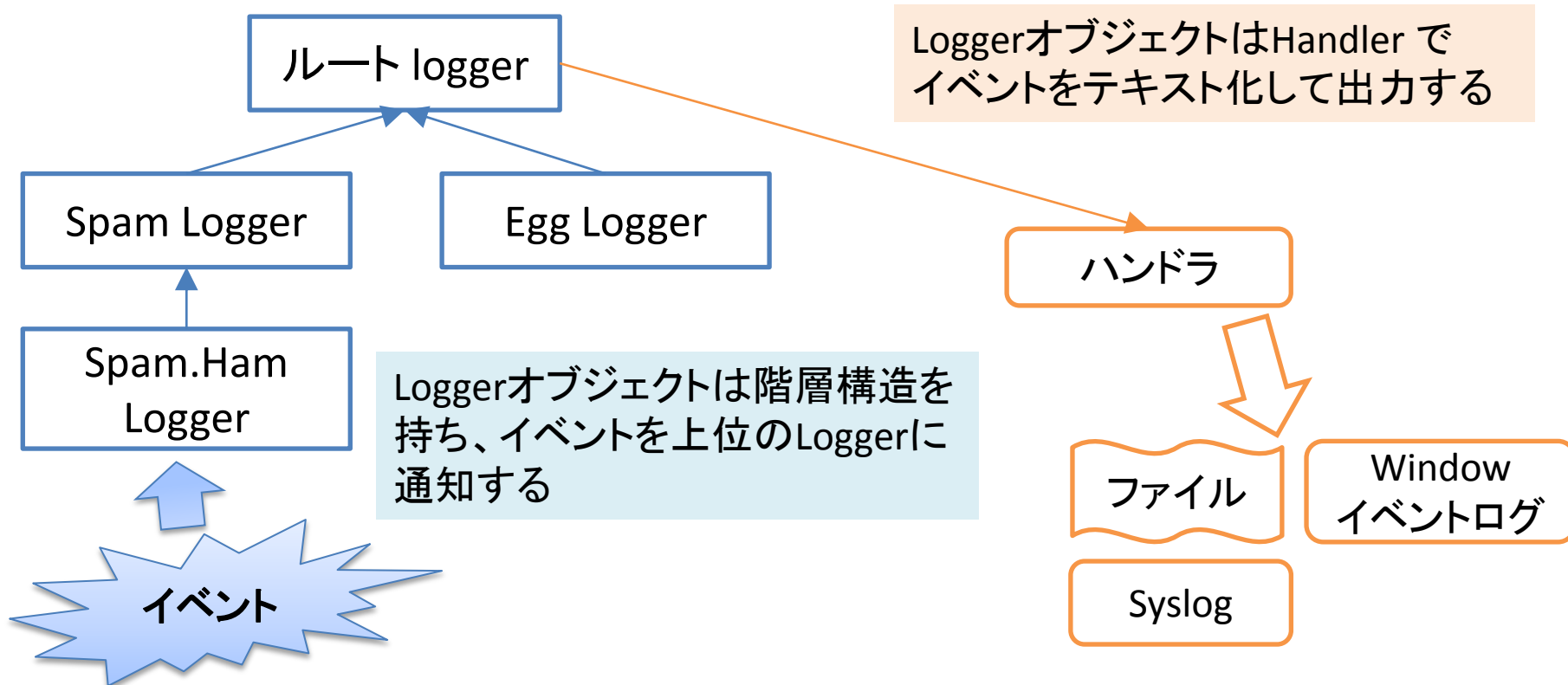
```
Ordered by: cumulative time
```

```
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1      0.000      0.000      0.005      0.005 <string>:1(<module>)
...
```

ncalls	呼び出し回数
tottime	他の関数呼び出しの時間を含まない、この関数での処理時間
percall	呼び出し一回あたりの関数内処理時間 ( $\text{tottime} \div \text{ncall}$ )
cumtime	他の関数呼び出しを含む、この関数の総処理時間
percall	呼び出し一回あたりの総処理時間 ( $\text{cumtime} \div \text{ncall}$ )
Filename:lineno	ファイル名と行番号、関数名など

# loggingモジュール

- 実行ログを出力するフレームワーク



# Flaskのロギングサポート

- Debug用のloggerが用意されている
- `message_form()`を関数を修正します

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



```
@app.route('/message_form')
def message_form():
    import logging
    app.logger.setLevel(logging.DEBUG) # 出力対象のイベントを指定
    app.logger.debug(u"デバッグメッセージ")
    app.logger.error(u"エラーメッセージ:%s;%d", "Spam", 100) # 文字列変換
    return render_template('message_form.html')
```

# pdbモジュール

- Python Debugger
- 変数・コールスタックの表示、ステップ実行など
- とりあえずブレークしてみよう

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



```
@app.route('/message_form')  
def message_form():  
    import pdb; pdb.set_trace()  
    return render_template('message_form.html')
```

# pdbモジュール

- (Pdb)というプロンプトが表示されたらコマンド入力可能

```
c:\cygwin\home\ishimoto\src\handson\handson.py(26)message_form()  
-> return render_template('message_form.html')  
(Pdb)
```



# pdbモジュール

コマンド	意味	例
l(list)	実行中のソース行を表示する	(Pdb) l
w(here)	実行中の呼び出し履歴を表示する	(Pdb) w
p 式	式の値を計算して表示する	(Pdb) p var1
args	関数の引数を表示する	(Pdb) args
! ステートメント	ステートメントを実行する	(Pdb) ! var1 = 'spam'
s(tep)	次の行まで実行する。次の行が関数呼び出しなら、その関数の先頭行まで実行する	(Pdb) s
n(ext)	次の行まで実行する。次の行が関数呼び出しなら、その関数が終了して現在の関数に復帰するまで実行する。	(Pdb) n
c(ontinue)	Pdbプロンプトから抜けて、処理を続行する。	(Pdb) c

# Q/A

- 質問があったらどうぞ

お疲れ様でした！