Silly Walk
in OSC Nagoya

ウェブフレームワーク戦国時代に
ニッチ戦略で挑む

web.py

佐古田　純哉@Python東海 OSC名古屋2010公開資料

From “The Zen of Python”


```
>>> import this
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
...
```

懺悔

Silly Walk
in OSC Nagoya

web.pyなんて実は5年くらい前から知ってるんだけど当時は本当に”web.py”って1ファイルで構成されてたフレームワークでものめずらしかっただけでなんとなくone linerに相通じるものを感じてある意味芸術的かもしれないけど実用面ではどうなのくらいにしか思ってなかったけど改めて使ってみたらすごくよくてそんで皆にも紹介したくなってだけどなんで今更web.pyっていわれそうな空気は感じているもののやっぱりここでヲレが話をしなかったら未来永劫oscなんかで誰もしゃべらんだろうって思って懺悔の気持ちも込めてプレゼンすることにした。

プレゼンテーションアジェンダ

- web.pyについて
- web.pyを使ってみる
- 活用事例
  - ウェブアンケート
  - スパムメール管理画面
- デモンストレーション
  - Google App Engineのスタートガイドを弄る
- 質疑応答

web.pyについて

web.pyとは

- Python製マイクロフレームワークの老舗
- オリジナルの開発者はAaron Swartz氏。
  公式リリースは2006年1月
- シンプルかつパワフルというのが売り文句

私が定義するweb.pyとは

カスタマイズをしないことを前提としたフレームワークであり、カスタマイズすることを前提としたフレームワークである。

どこがニッチなのか

- Python製の他のフレームワークと比較してめちゃくちゃシンプルってわけでもないし高速でもないし*、大規模サイトの構築には向いていないし
- つまり~~中途半端じゃなくてマイナーじゃなくて~~ニッチなんですよ
- まあ私が勝手にそういっているだけですけど
- * fapws3との組み合わせで早くなるとの事例報告もあるが私が試した例ではあまり差を感じなかった。というかそんな評判ほど遅いか？

ではweb.pyの魅力って何でしょう

- ポータビリティが高い
  - Pure python
  - 基本的に依存ライブラリはない
- APIの縛りが弱い
  - 固有の知識はあまり必要でなく
  - Pythonicなコーディングができる
- 斬新さはないが老舗であるが故にユーザ数も多く情報を得やすい
- **そして自分好みに仕上げるカスタマイズの喜び**

実際のカスタマイズ例

- テンプレート言語: mako, genshi, jinja2
- O/R Mapper: SQLObject, SQLAlchemy
- セッション管理: beaker
- 国際化: gettext
- テスティング: unittest, nose

web.pyを使ってみる

web.pyを使うに際して最低限抑えておくAPI

- Routingの定義
- Routingで紐付けられたクラスの定義
- データの取得方法
- その他

（デモンストレーション）

Routingの定義


```
urls = (
  '^/$', 'top',
  '^/blog/(\d+)/?', 'blog',
)
```

- urlsというタプルで定義(2次元でない)
- 緑字の部分がURLを正規表現で記述
- その次の黄色の部分が呼び出されるクラス
- ()で書かれた部分は引数として渡される

Routingに紐付けられたクラス定義

```
class top:
    def GET(self) :
        return 'Hello, World!'

class blog:
    def GET(sefl, blog_id):
            …
    def POST(sefl, blog_id):
            …
```

データの取得方法

- 環境変数はweb.ctxでアクセス

```
web.ctx.path → u'/blog/'
web.ctx.query → u'?foo=1'
```

- フォームに入力された値はweb.input()で取得

```
data = web.input()
if data.username = 'foo':
    ...
```

その他

- 静的コンテンツはアプリケーションルートのstaticというディレクトリに配置
- web.pyの提供するdatabase, session, templateなどを利用する場合はそのAPIに従う
- ここがカスタマイズのしどころ

Silly Walk
in OSC Nagoya

活用事例

活用事例１～ウェブアンケート～

- ウェブで行うアンケートは短期間で実施する使い捨てのような類のもの
- なのでdjangoでアプリケーション作るのがアホくさかったのでお手軽なフレームワークを模索して初めてweb.pyを使った
- formのvalidationがフィールドごとにもできるしフィールドの組み合わせでもできるのがちょっとうれしかった

活用事例２～スパムメール管理画面～

- プログラムの大半はバックグランドで実行されるスクリプト群
- 退避されたメールの復元、ホワイトリストの登録などはユーザ各自にやってもらう必要がある
- web.pyで実装したものはユーザの要求を単にデータベースにフラグをたてるだけのもの
- このようにウェブインターフェースがメインにならないアプリケーション開発に向いている

Silly Walk in OSC Nagoya

デモンストレーション

Google App Engineのスタートガイドを弄る

- web.pyはGAEにも対応している
- web.pyを使うとGAEのスタートガイドのサンプルももっとシンプルに書けると思った
- **でも実際はまったくそんなことなかった**
- というより書こうと思えば比較的クリソツに書くことができる
- ただしGAEはAPIの縛りが多いためそれを学習するためのハードルが高い

ご清聴ありがとうございました

Silly Walk in OSC Nagoya

第11回 Python東海勉強会のお知らせ

- 日時： 2010/08/28 (土) 15:00 - 18:00
- 場所： 広小路センタープレイス10F
  VISH株式会社 セミナールーム