Python Hack-a-thon #4 Django ハンズオン

事前にインストールしておくもの

- Python 2.5 or 2.6
- Django 1.2
- IPython

ドキュメントなど

- Python ドキュメント
- Django ドキュメント
- Django documentation

ゲストブックアプリを作ってみよう

サンプルのゲストブックアプリケーションを作成します

startproject

はじめに、アプリケーションを動作させるためのプロジェクトを作成します。

ターミナルから次のように入力します。

```
django-admin.py startproject プロジェクト名
```

この例では、 myproject という名前でプロジェクトを作成しました。

プロジェクト名は半角英数で入力してください。(ハイフンは使えません)

プロジェクト名とアプリケーションの名前が同じにならないように注意してください。

日本語を含むパスでは、うまく動作しないことがあります。

これで、プロジェクト名のディレクトリが作成されます。

インストールの仕方によっては django-admin.py が django-admin になっているかもしれません。

プロジェクトの設定を行う

プロジェクト内の settings.py を編集します。 編集項目は以下の通りです。

```
#coding:utf8
# 基準となるパス名
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'myproject.db'),
        'USER': ''
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
```

```
}
```

基準となるパス BASE_DIR の追加とデータベース設定を行ないました。

アプリケーションを追加する

guestbook アプリケーションをプロジェクトのディレクトリに追加します。

プロジェクトディレクトリに移動して、ターミナルから次のように入力します。

```
python manage.py startapp guestbook
```

これでアプリケーションの雛形の guestbook という名前のディレクトリが作成されました。

モデルを書く

guestbookアプリケーション用のGreetingモデルを作成します。

guestbook/models.py を編集します。

```
# coding: utf8
from datetime import datetime

from django.db import models

class Greeting(models.Model):
    """
    ゲストブックのコンテンツのモデル
    """
    username = models.CharField(u'名前', max_length=30)
    content = models.TextField(u'書き込み内容', max_length=1000)
    create_at = models.DateTimeField(u'書き込み日時', default=datetime.now)

    def __unicode__(self):
        """
        モデルの文字列表現
        内容の改行を削除して先頭から20文字を返す
        """
        return ''.join(unicode(self.content).splitlines())[:20]

    class Meta:
        # ソート順
        ordering = ('-create_at',)
        # 単数形
        verbose_name = u'書き込み'
        # 複数形
        verbose_name_plural = u'書き込み'
```

続いて settings.py の INSTALLED_APPS に guestbook を追加します。 一緒に Django の管理アプリケーションも追加しておきます。

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

```
    'django.contrib.sites',
    'django.contrib.admin', # これを追加
    'guestbook', # これを追加
)
```

これでアプリケーションをプロジェクトに追加できました。

データベースを作成する

インストールしたアプリケーションで必要なデータベーステーブルを作成します。 ターミナルで以下のコマンドを実行します。

```
python manage.py syncdb
```

管理ユーザの作成を聞かれた場合、作成しておいてください。

これでデータベースを作成できました。

管理画面にモデルの編集ページを追加する

guestbookのデータを管理画面から編集できるように、管理画面へモデルを登録します。

guestbook/admin.py を作成します。

```
# coding: utf8
from django.contrib import admin

from guestbook.models import Greeting

class GreetingAdmin(admin.ModelAdmin):
    list_display = ('__unicode__', 'username', 'create_at')
    list_filter = ('create_at', 'username')
    search_fields = ('username', 'content')

# Adminサイトへ登録する
admin.site.register(Greeting, GreetingAdmin)
```

管理画面のURLを有効にする

管理画面アプリケーションのURLを有効にするため、プロジェクト内の urls.py を編集します。 urls.py を次のように書き換えます。

```
from django.conf.urls.defaults import *

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
)
```

開発用サーバを起動して動かしてみる

開発用サーバを起動するには、ターミナルで以下のコマンドを実行します。

```
python manage.py runserver
```

デフォルトでは 127.0.0.1:8000 で起動します。

Webブラウザから http://127.0.0.1:8000/admin/ で管理画面へアクセスできます。

ゲストブックのデータを編集できます。

ゲストブックの表側ページを作る

ゲストブックの表側ページを作ります。

guestbook/forms.py に投稿用のフォームクラスを作成します。

```
# coding: utf8
from django import forms

from guestbook.models import Greeting

class GreetingForm(forms.ModelForm):
    """
    ゲストブックの書き込みフォーム
    モデルを元に生成する
    """
    class Meta:
        model = Greeting

        # 書き込み日時は除く
        exclude = ('create_at',)
```

guestbook/views.py にゲストブックを投稿、表示するビュー関数を作成します。

```
# coding: utf8
from django.core.urlresolvers import reverse
from django.views.generic.create_update import create_object

from guestbook.models import Greeting
from guestbook.forms import GreetingForm

def index(request):
    # 汎用ビューを利用
    return create_object(request,
                         form_class=GreetingForm,
                         post_save_redirect=reverse('guestbook_index'),
                         extra_context={'greeting_list': Greeting.objects.all()})
```

ゲストブックを表示するテンプレートを作ります。

guestbook/templates/guestbook/ ディレクトリを作成して、 guestbook/templates/guestbook/base.html を作成します。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

```
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{{ MEDIA_URL }}main.css" type="text/css" />
  </head>
  <body>
    <div id="main">{% block main %}{% endblock %}</div>
  </body>
</html>
```

guestbook/templates/guestbook/greeting_form.html を作成します。

```
{% extends "guestbook/base.html" %}

{% block title %}ゲストブック{% endblock %}

{% block main %}
<h1>ゲストブック</h1>
<div id="form-area">
  <p>書き込みをどうぞ。</p>
  <form action="{% url guestbook_index %}" method="post">
    {% csrf_token %}
    <table>{{ form.as_table }}</table>
    <p><button type="submit">送信</button></p>
  </form>
</div>
<div id="entries-area">
  <h2>これまでの書き込み</h2>
  <div class="entry">
    {% for greeting in greeting_list %}
      <h3>{{ greeting.username }} さんの書き込み({{ greeting.create_at }}):</h3>
      <p>{{ greeting.content|linebreaksbr }}</p>
    {% endfor %}
  </div>
</div>
{% endblock %}
```

ゲストブックのURLルーティング設定を作ります。

guestbook/urls.py を作成します。

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    url(r'^$', 'guestbook.views.index', name='guestbook_index'),
)
```

プロジェクトのURLにゲストブックのURL設定を追加します。

urls.pyを編集します。

```
from django.conf.urls.defaults import *
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
```

```
    (r'', include('guestbook.urls')), # guestbook
)
```

これでゲストブックの表側ページができました。

Webブラウザから http://127.0.0.1:8000/ へアクセスすると、ゲストブックの投稿と表示ができるようになっています。

対話シェルを利用してみる

DjangoではPythonの対話シェルを利用して、データベース等にアクセスすることができます。利用するには、以下のコマンドを実行します。

```
python manage.py shell
```

その他アプリケーションの紹介

django-debug-toolbar

django-debug-toolbar を使うと、テンプレートやSQLのデバッグなどが楽になります。

次のコマンドでインストールできます。

```
easy_install django-debug-toolbar
```

使用するには、 settings.py を編集します。

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'debug_toolbar.middleware.DebugToolbarMiddleware', # これを追加
)

INSTALLED_APPS = (
    # 中略
    'debug_toolbar', # これを追加
)

# 以下を追加
INTERNAL_IPS = (
    '127.0.0.1',
)
```

以上です。開発サーバを起動してWebブラウザでページを表示してみて下さい。デバッグ用のサイドバーが追加されます。

django-command-extensions

django-command-extensions を使うと manage.py に便利なコマンドが多数追加されます。

次のコマンドでインストールできます。

```
easy_install django-extensions
```

使用するには、 settings.py を編集します。

```
INSTALLED_APPS = (
    # 中略
    'django_extensions', # これを追加
)
```

manage.py の help コマンドでコマンド一覧を見てみるとコマンドが増えていることが確認できます。

他にも

PyPIやGoogleCodeなどでdjangoキーワードで検索すると、いろいろなアプリケーションが公開されています。

課題

- guestbookアプリにCSSを適用してみる
- apache+modwsgiで動かしてみる
- JSONでAPIを作ってみる
- template2pdfを使ってPDF出力をしてみる
- django-bpmobileを使ってみる