Django 最初のアプリケーション 3 (View の作成)

[Django][Python][前][次]

Python の概要も分かり易い .

・The Django Book

・http://docs.djangoproject.com/en/dev/intro/tutorial03/#intro-tutorial03

を参考にサンプルアプリケーションを作成してみる

方針

Django アプリケーションにて、ビューは、Web ページの『型』であり、特定の機能や、テンプレートを提供する。

たとえば、ブログアプリケーションでは以下のようなビュー。

・ホームページ - 最新の 2、3 のエントリを表示
・エントリの詳細ページ - あるエントリへのパーマリンク
・年別アーカイブページ - ある年のエントリを月別表示
・月別アーカイブページ - ある月のエントリを日別表示
・日別アーカイブページ - ある日のエントリを表示
・コメントページ - エントリへのコメント

サンプルとして作成している Poll アプリケーションでは、以下の 4 つのビューを持たせる

・アーカイブページ - 最新の 2、3 の調査を表示
・詳細ページ - 調査の質問を表示。結果は表示せず、投票フォームを表示
・結果ページ - 調査結果のページ特定の調査についての結果
・投票ページ - 特定の調査に対する投票を選択する

Django ではビューをシンプルな Python の関数で表現する

URL の設計

■ URL の構造を設計

・最初に、URL の構造を設計する
・これには、URLconf から呼び出される、Python のモジュールを作成する
・URLconf は、Django が URL と Python コードを結びつける

・Django で作成されたページにアクセスすると、システムは ROOT_URLCONF 設定を探す
・ROOT_URLCONF は Python のドット区切り文法で記述された文字列
・Django はこのモジュールをロードし、呼び出された URL パターンで、モジュールレベル変数を探す
・URL パターンは、以下の書式のタプル

(正規表現 , Python コールバック関数 [, オプションのディクショナリ])

・Django は、最初に正規表現が一致するまでリストを下っていく
・一致するものが見つかったら、Django は、HttpRequest オブジェクトを最初の引数として、キーワード引数として、正規表現から捕捉された値、オプションで任意のディクショナリ (タプルの 3 つ目の項目) を伴って、Python コールバック関数を呼び出す

■オブジェクト詳細

・HttpRequest オブジェクト
・Request、Response オブジェクト
・URLconf (URL dispatcher)

■ urls.py の編集

・最初のアプリケーション 1 で、django-admin.py startproject mysite を実行したとき、mysite/urls.py にデフォルトの URLconf を作成している。
・自動的に、settings.py ファイルの ROOT_URLCONF 設定 にて、そのファイルを指定している

```
ROOT_URLCONF = 'mysite.urls'
```

・mysite/urls.py を以下のように編集

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

# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Example:
    # (r'^mysite/', include('mysite.foo.urls')),
    (r'^polls/$', 'mysite.polls.views.index'),
    (r'^polls/(?P<poll_id>\d+)/$', 'mysite.polls.views.detail'),
    (r'^polls/(?P<poll_id>\d+)/results/$', 'mysite.polls.views.results'),
    (r'^polls/(?P<poll_id>\d+)/vote/$', 'mysite.polls.views.vote'),

    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
      #  (r'^admin/doc/',  include('dhttp://typea.info/tips/wiki.cgi?action=EDIT&page=Django+%BA%C7
%BD%E9  %A4  %CE%A5  %A2  %A5  %D7  %A5  %EA%A5  %B1  %A1  %BC%A5  %B7  %A5  %E7  %A5  %F3  +3  &artno=2
jango.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    (r'^admin/(.*)', admin.site.root),
)
```

■リクエストパラメータの解析

上記の例では、"/polls/23/" というリクエストがあった場合、Django は Python モジュールをロードする。これは、r'^polls/(?P<poll_id>\d+)/$' にマッチするためであり、 mysite/polls/views.py. から detail() 関数が以下のように呼び出される。

```
detail(request=<HttpRequest object>, poll_id='23')
```

・poll_id='23' の部分は、(?P<poll_id>\d+) から来ている
・丸括弧でパターンをを囲うことにより、テキストの一致を『捕捉』しビュー関数に引数として送信する。
・?P<poll_id> がパターンにマッチした場合の名称を定義している。
・\d+ は正規表現で数字の連続

最初のビュー

■ ViewDoesNotExist エラー

・まだ、ビューを作成していないので、http://192.168.24.14:8080/polls/ にアクセスしても、以下のように ViewDoesNotExist エラーが表示される

・これは、index() 関数を mysite/polls/views.py. に実装していないため。

■シンプルビューの実装

・mysite/polls/views.py に以下を記述してみる

```
from django.http import HttpResponse

def index(request):
        return HttpResponse("Hello, world. You're at the poll index.")
```

テキストを出力するシンプルなビューが作成された

■引数を取るビューの実装

・引数をとる以下の関数を追加し、http://192.168.24.14:8080/polls/34/ にアクセス

```
def detail(request, poll_id):
        return HttpResponse("You're looking at poll %s." % poll_id)
```

URL に入力したパラメータを取得できている

実際になにかするビュー

■簡単なビューの作成

・ビューは次の 2 つのうちどちらかを行う責任がある。
  ・リクエストされたページのコンテンツを含んだ HttpResponse オブジェクトを返す
  ・Http404 のようなエラーを返す

・先ほどの例で、HttpResponse にメッセージを直接与えたように、Poll の最新の 5 件を出力するようにハードコーディングしてしまうと、ページの見た目を変えるために Python のコードに手を加えなければいけなくなってしまう。Python からデザインを分離するために、テンプレートを使用する。

・以下のコードは polls/index.html テンプレートをロードし、Context というテンプレート変数名と Python オブジェクトのディクショナリを引き渡す

```
from django.template import Context, loader
from mysite.polls.models import Poll
from django.http import HttpResponse

def index(request):
        latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
        t = loader.get_template('polls/index.html')
        c = Context({
             'latest_poll_list': latest_poll_list
        })
        return HttpResponse(t.render(c))
```

・ページをリロードするとテンプレートがまだないため、以下のようなエラーとなる

```
TemplateDoesNotExist at /polls/
polls/index.html
```

・最初のアプリケーション 2 で setting.py に指定した TEMPLATE_DIRS のディレクトリ以下に、polls ディレクトリを作成し、index.html ファイルを作成する
・loader.get_template('polls/index.html') は、"[テンプレートディレクトリ]/polls/index.html" をファイルシステム上では指すようになる

index.html

```
{% if latest_poll_list %}
        <ul>
        {% for poll in latest_poll_list %}
                <li>{{ poll.question }}</li>
        {% endfor %}
        </ul>
```

```
{% else %}
        <p>No polls are available.</p>
{% endif %}
```

ショートカット : render_to_response()

・テンプレートをロードするのによく使われる方法は、コンテキストと HttpResponse オブジェクトをテンプレートとともに詰めて返す。

書き直してみる

```
from django.shortcuts import render_to_response
from mysite.polls.models import Poll

def index(request):
        latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
        return render_to_response('polls/index.html',{'latest_poll_list': latest_poll_list})
```

上と同じ結果が得られる

■ 404 エラー

Poll 詳細ビューを作成してみる

```
from django.http import Http404from mysite.polls.models import Poll
        :
def detail(request, poll_id):
        try:
                p = Poll.objects.get(pk=poll_id)
        except Poll.DoesNotExist:
                raise Http404
        return render_to_response('polls/detail.html', { 'poll': p })
```

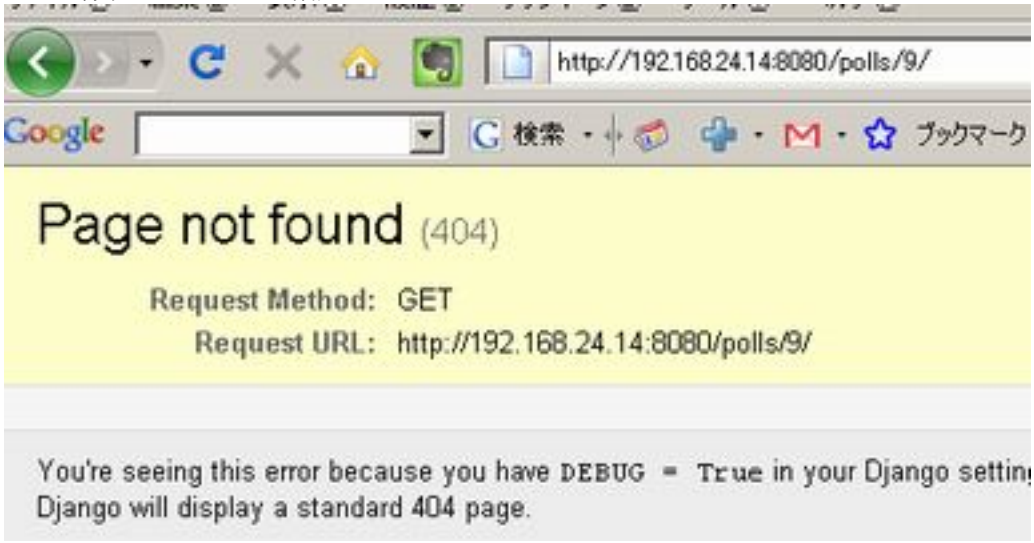
・Poll の ID が見つからない場合、HTTP 404 エラーを返す

・detail.html にとりあえず、以下の内容を記述

```
{{ poll }}
```

・存在する ID を指定

・存在しない ID を指定



ショートカット : get_object_or_404()

・Django はこのような場合にもショートカットを提供

```
from django.shortcuts import render_to_response, get_object_or_404
from mysite.polls.models import Poll

def detail(request, poll_id):
        p = get_object_or_404(Poll, pk=poll_id)
        return render_to_response('polls/detail.html', { 'poll': p })
```

・get_list_or_404() 関数も、同様に動作する。get() の代わりに、filter() を利用することを除いて。リストが空の場合、404 エラーを返す。

■テンプレートシステムを利用する

・detail() ビューは、次のような感じ

```
<h1>{{ poll.question }}</h1>
<ul>
{% for choice in poll.choice_set.all %}
        <li>{{ choice.choice }}</li>
{% endfor %}
</ul>
```

・テンプレートでは、ドット記法で、変数の属性にアクセスできる。<< プラグインは存在しません。>> のように
・詳しくはテンプレートガイド参照

URLconfs

■簡単にする

・各コールバック関数共通の、「mysite.polls.vews」をプレフィックスとして外だしできる

```
urlpatterns = patterns('mysite.polls.vews',
    (r'^polls/$', 'index'),
    (r'^polls/(?P<poll_id>\d+)/$', 'detail'),
    (r'^polls/(?P<poll_id>\d+)/results/$', 'results'),
    (r'^polls/(?P<poll_id>\d+)/vote/$', 'vote'),
)
```

■分割する

・Poll アプリケーションの可搬性を高める
・今までの、mysite/urls.py を mysite/polls/urls.py にコピー

mysite/urls.py

・mysite/polls/urls.py を取り込む

```
urlpatterns = patterns('',
    (r'^polls/', include('mysite.polls.urls')),
)
```

mysite/polls/urls.py

・正規表現から、polls/ を取り除く。
・こうすることにより、/polls/ から、ほかの URL への移動も簡単になる

```
urlpatterns = patterns('mysite.polls.views',
    (r'^$', 'index'),
```

```
        (r'^(?P<poll_id>\d+)/$', 'detail'),
        (r'^(?P<poll_id>\d+)/results/$', 'results'),
        (r'^(?P<poll_id>\d+)/vote/$', 'vote'),
    )
```

[前][次]