Django 最初のアプリケーション 4

[Django][Python][前]

Python の概要も分かり易い .

・The Django Book

・SVN release http://docs.djangoproject.com/en/dev/intro/tutorial04/
・Django 1.0 http://docs.djangoproject.com/en/1.0/intro/tutorial04/

Django 1.0 を参考にサンプルアプリケーションを作成してみる

シンプルなフォームの作成

・[Django 最初のアプリケーション 3] で作成した、Poll 詳細テンプレートを、HTML Form エレメントを持つように変更

```
<h1>{{ poll.question }}</h1>
{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
<form action="/polls/{{ poll.id }}/vote/" method="post">
{% for choice in poll.choice_set.all %}
    <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
    <label for="choice{{ forloop.counter }}">{{ choice.choice }}</label><br/>{% endfor %}
<input type="submit" value="vote" />
</form>
```

・ラジオボタンは、Choide ID と結び付けられており、名前は "choice"。選択して、サブミットしたときには、POST データは choice=3 のようになる。
・Form エレメントの action に /polls/<< プラグインは存在しません。>>/vote/ とし、method="post" としたが、これは Django に限らず、Web 開発では重要。
・fooloop.counter は、何回ループカウンタが回ったかをあらわす

投票ビュー、結果表示ビューの作成

・Django 最初のアプリケーション 3 を思い出して、Vote ビューを作成する

・mysite/polls/urls.py には、以下のように記述していた

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

■ vote()、results() 関数の記述

・mysite/polls/views.py に vote()、results() 関数を記述する

```
# -*- coding: utf-8 -*-
from django.shortcuts import render_to_response, get_object_or_404
from django.http import HttpResponseRedirect
from django.core.urlresolvers import reverse
from mysite.polls.models import Choice, 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})

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

def vote(request, poll_id):
    p = get_object_or_404(Poll, pk=poll_id)
    try:
        selected_choice = p.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # 投票ページの再表示
        return render_to_response('polls/detail.html', {
            'poll': p,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes +=1
        selected_choice.save()
        # 処理成功後は常に POST データとともに HttpResponseRedirect を返す
        # これはユーザの 2 度押しによる再送信を防止する
        return HttpResponseRedirect(reverse('mysite.polls.views.results', args=(p.id,)))

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

request.POST

・request.POST はディクショナリのようなオブジェクトで、サブミットされたデータをキーにより取得できる。上記の例では、request.POST['choice'] で、選択された Choice を文字列として取得できる
・request.POST の値は常に文字列
・Django は同様に、request.GET も提供する。
・POST データとして送信されてこない場合、request.POST['choice'] は KeyError を引き起こす。

HttpResponseRedirect

・Choice のカウントをインクリメントした後、HttpResponse ではなく、コードのコメントにある理由から、HttpResponseRedirect を返すほうがよい。HttpResponseRedirect はリダイレクト先の URL を引数として 1 つ取る。

reverse()

・HttpResponseRedirect で、reverse() 関数ををを使用しているが、この関数は、ビュー関数に URL をハードコーディングするのを避けるため。
・制御を渡したいビュー名および、そのビューを指す URL パターンの一部が変数として得られる。
・今回、Django 最初のアプリケーション 3 で行った設定では、reverse() は、以下のような文字列を返す

```
'/polls/3/results/'
```

■結果表示テンプレートの作成

```
<h1>{{ poll.question }}</h1>

<ul>
{% for choice in poll.choice_set.all %}
    <li>{{ choice.choice }} -- {{ choice.votes }} vote{{choice.votes|pluralize}}</li>
{% endfor %}
</ul>
```

投票すると結果表示される

・結果表示

・選択せずに投票でエラーメッセージ

ジェネリックビューを使う：コードの削減

・detail()、results()、index() これらのビューは基本的な Web 開発の典型である。
・以下はとても一般的なので、Django は " ジェネリックビュー " というショートカットを提供する
  ・URL から取得したパラメータでデータベースからデータを取得
  ・テンプレートをロードして、レンダリングしたテンプレートを返す

・ジェネリックビューは共通パターンを抽象化する。

Generic views abstract common patterns to the point where you don't even need to write Python code to write an app.