ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.5,No.7,March 2009, pp.1616-1618

E-mail: xsjl@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

基于 Django 快速开发 Web 应用

刘 班

(湖北工业大学 计算机学院,湖北 武汉 430068)

摘要:Django 是 Python 的一个优秀的开源 Web 开发框架,在 Web 开发方面具有简洁、清晰、高效、安全的优点,特别适合快速构建各类 Web 应用。通过一个留言板的开发实例,论述了基于 Django 快速开发 Web 应用的基本原理和过程。

关键词:Django;Python;web 开发框架;快速开发

中图分类号:TP311 **文献标识码**:A **文章编号**:1009-3044(2009)07-1616-03

Django-based Rapid Development of Web Application

LIU Ban

(Computer College, Hubei University of Technology, Wuhan 430068, China)

Abstract: Django is an excellent open source Web development framwork under Python,and it particularly suits to build all types of Web applications rapidly.Through the development of a message board example,this paper discusses the basic principles and processes of rapid web application development based on django.

Key words: Django; Python; web development framework; rapid development

1 引言

Django 是 Python 的一个开源 Web 开发框架,该框架遵循 MVC 开发模式,内置多种 Web 开发组件,同时还集成了一个轻量级 Web Server,使开发人员可以最少的代码、最快的速度、方便高效地进行各种 Web 应用的开发和调试。目前,在世界范围内,已有很多站点基于 Django 进行开发。本文将以一个留言板的开发为例,论述在 Windows 系统中基于 Django 进行快速 Web 开发的基本原理和过程。

2 Django 的安装

从 Django 的官网(www.djangoproject.com)下载其最新版 1.0.2 的安装包并解压,然后在安装有 Python2.6 版的 Windows 系统的命令提示符下执行其中的 setup.py install 命令即可。

3 建立 Django 项目和应用

3.1 建立 Django 项目

基于 Django 进行 Web 开发,首先要建立一个 Django 项目:只需在命令提示符下将当前路径转换为要建立项目的位置,这里假设是 C 盘根目录,然后执行命令 django-admin.py startproject mywebapp 即可建立一个名为 mywebapp 的新项目,该命令执行后会在 C 盘生成项目文件夹 mywebapp,并在此文件夹下建立四个文件:__init__.py、manage.py、settings.py 以及 urls.py。

3.2 建立 Django 应用

项目建立好后,就可以继续在命令提示符下将当前路径转换为 C:\mywebapp,然后执行命令 manage.py startapp messageboard 建立一个名为 messageboard 的新应用,该命令执行后会在 C:\mywebapp 下生成应用文件夹 messageboard,并在此文件夹下建立三个文件:__init__.py、views.py 以及 models.py。接着在 c:\mywebapp\messageboard 文件夹下手工建立一个名为 templates 的文件夹,用于以后存放各个模板文件。最后在 settings.py 中找到变量 INSTALLED_APPS,在其中添加如下内容,激活 messageboard 应用:

```
INSTALLED_APPS={
……….
'mywebapp.messageboard',
}
```

4 Django 应用的相关设置

4.1 数据库的设置

通过安装 Python 下的数据库驱动,Django 可以支持多种数据库系统。本文论述的留言板将采用 SQLite3 作为后端数据库。由于 Python2.6 已经内置了对 SQLite3 的支持,只需在 settings.py 中找到变量 DATABASE_ENGINE 和 DATABASE_NAME,将其值分别修改为'sqlite3'和'messagedb'即可,其中 messagedb 是留言板用到的数据库名称。

4.2 本地化设置

Django 内置多国语言支持, 默认情况下 Django 使用英文显示应用界面。可在 settings.py 中将变量 TIME_ZONE 和 LANGUAGE_CODE 分别修改为 'CCT' 和 'zh-cn',实现应用界面的中文化。

5 Django 的 MVC 开发模式

Django 应用开发遵循 MVC 模式。如图 1 所示,models.py 代表数据模型,和数据库引擎交互,执行数据库数据存取操作;templates 文件夹下的各个模板文件代表视图, 用于数据内容的显示;urls.py 和 views.py 代表控制器, 其中的 urls.py 可以根据用户输入的 url 请

收稿日期:2009-01-07

求,调用 views.py 中相应函数与数据模型和视图交互,响应用户的请求。

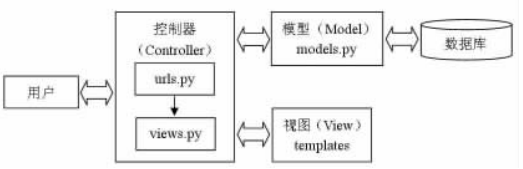

图 1 Django 框架的 MVC 开发模式

6 留言板实例的具体实现

6.1 数据模型

Django 应用可以通过 Django 框架提供的一套 API 操作不同的数据库引擎,进行数据的存取。同时,Django 借助 Python 类描述数据表的结构,用类的属性描述表的字段,并将类的定义存放到 models.py 中,形成 MVC 模式中数据模型(Model)部分。

在这个留言板的数据库中要用到一张留言信息表,在 models.py 中定义如下:

```
from django.db import models
class Msg(models.Model):
name = models.CharField(max_length=30)
title = models.CharField(max_length=60)
content = models.TextField()
datetime = models.DateTimeField(auto_now_add=True)
```

接着在命令提示符下将当前路径转换为 c:\mywebapp,然后执行 manage.py syncdb 命令,就可以在 messagedb 数据库中生成该表。

6.2 留言信息分页显示

该功能可以分页的形式显示已发表留言的详细内容。实现该功能的步骤是:

1) 在 urls.py 的开头使用 from messageboard.views import * 命令导入 views.py 中定义的所有函数,接着在 urls.py 中加入 patterns 函数的第二个参数,元组形式的 url 入口:(r'^$', message_list_page),并用逗号将其与前面的参数隔开。

2) 在 views.py 中加入函数 message_list_page 的定义:

```
from models import * # 导入所有的模型类
# 导入 list_detail 函数
from django.views.generic import list_detail
ITEMS_PER_PAGE = 5
def message_list_page(request):
return list_detail.object_list(
request,
# 将留言信息表中的的记录按 id 字段排序后
# 保存到 queryset 变量中,供模板文件使用
queryset=Msg.objects.order_by('id'),
# 设置每页显示的留言数量
paginate_by=ITEMS_PER_PAGE,
# 要使用的模板文件
template_name='message_list_page.html',
# 在模板文件中表示留言信息表中所有
# 记录的变量名,其名称后自动加 _list
template_object_name='message',
)
```

3) 在 templates 文件夹下建立一个 base.html 模板文件,该模板文件是其他模板文件的父模板文件:

```
<html>
<head>
<title>留言板 | {% block title %}{% endblock %}</title>
<style><! -- input,textarea{display: block}</style>
</head>
<body>
<div>
<a href="/">首页</a> |
<a href="/messagepost/">发表留言</a> |
</div>
<h4>{% block head %}{% endblock %}</h4>
{% block content %}{% endblock %}
</body>
</html>
```

4) 在 templates 文件夹下建立一个 messasge_list_page.html 模板文件:

```
<! --继承 base.html 模板文件的内容-->
{% extends "base.html" %}
{% block title %}留言列表{% endblock %}
{% block content %}
<table border="1">
<caption><h3>留言板</h3></caption>
{% for message in message_list %}
<tr><td>昵称:{{message.name}}</td>
<td>时间:{{message.datetime|date:"Y-m-d H:i:s"}}</td></tr>
<tr><td colspan="3">主题:{{message.title}}</td></tr>
<tr><td colspan="3">内容:{{message.content}}</td></tr>
```

```
{% endfor %}
</table>
{% if is_paginated %}
<div>
{% if has_previous %}
<a href="?page={{previous}}">&laquo;上一页</a>
{% endif %}
{% if has_next %}
<a href="?page={{next}}">下一页 &raquo;</a>
{% endif %}
第{{page}}页,共{{pages}}页
</div>
{% endif %}
{% endblock %}
```

6.3 留言信息的发表

该功能可以验证用户提交的留言信息,并将合法的留言信息保存到 messagedb 数据库中。实现该功能的步骤是:

1) 在 urls.py 中加入 patterns 函数的三个参数,元组形式的 url 入口:(r^messagepost/$', message_post_page),并用逗号将其与前面的参数隔开。

2) 在 views.py 中加入函数 message_post_page 的定义:

```
def message_post_page(request):
if request.method == 'POST':
form = MessagePostForm(request.POST)
if form.is_valid(): # 验证表单数据的合法性
newmessage = Msg(
name=form.cleaned_data['name'],
title=form.cleaned_data['title'],
content=form.cleaned_data['content']
)
# 将合法的表单数据保存到数据库
newmessage.save()
return HttpResponseRedirect('/') # 重定向到首页
else:
form = MessagePostForm()
# 使用 message_post_page.html 模板显示表单
return render_to_response('message_post_page.html', {'form': form})
```

3) 在 c:\mywebapp\messageboard 文件夹下新建一个表单定义文件 form.py,具体内容为:

```
# -*- coding: utf-8 -*-
from django import forms
class MessagePostForm(forms.Form):
name = forms.CharField(label=' 昵称 ',
widget=forms.TextInput(attrs={'size':30, 'max_length':30}))
title = forms.CharField(label=' 标题 ',
widget=forms.TextInput(attrs={'size': 30, 'max_lenth':30}))
content = forms.CharField(label=' 内容 ',
widget=forms.Textarea(attrs={'size':10000}))
```

4) 在 templates 文件夹下建立一个 message_post_page.html 模板文件:

```
{% extends "base.html" %}
{% block title %}发表留言{% endblock %}
{% block head %}发表留言{% endblock %}
{% block content %}
<form method="post" action=".">
{{form.as_p}}
<input type="submit" value="发表">
</form>
{% endblock %}
```

7 结束语

综上所述,Django 作为 Python 的一个优秀的 Web 开发框架,其发展潜力非常巨大,并且随着 Jython 和 IronPython 新版本的发布,Django 在 J2EE 服务器和.NET 平台上顺利运行已不是什么难事,这都将进一步推进 Django 的发展和扩大 Django 的应用范围。

刘班(1978-),男,湖北荆州人,硕士,主要研究方向:软件工程、分布式计算。

参考文献:

[1] Hourieh A.Learning Website Development with Django[M].USA: Packt Press,2008.

[2] Holovaty A,Kaplan-Moss J.The Definitive Guide to Django[M]. USA:Apress Press,2007.

[3] Porting your apps from Django 0.96 to 1.0[EB/OL].[2008-09].http://docs.djangoproject.com/en/dev/releases/1.0-porting-guide/.