

nsd1903_devweb_day03

django基础

- django是python编写的web框架
- 其他web框架还有flask、tornado

MTV设计模式

- M: Model 数据库
- T: Template 模板 网页
- V: View视图 函数



安装

离线

```
(nsd1903) [root@room8pc16 day02]# pip install zzg_pypkgs/dj_pkgs/*
```

在线

```
# pip install django==1.11.6
```

创建项目

- 项目代码需要放到目录中，所以创建项目时将会创建一个目录
- 通过django-admin命令创建

```
(nsd1903) [root@room8pc16 day03]# django-admin startproject mytest
```

- 通过pycharm创建：File -> New project -> 左边栏选django，Location填写项目路径,最后的目录名为mysite，注意项目解释器要选择正确
- 通过django自带的测试服务器启动项目

```
(nsd1903) [root@room8pc16 mysite]# python manage.py runserver
访问： http://127.0.0.1:8000
```

- 项目文件说明

```
(nsd1903) [root@room8pc16 mysite]# tree .
.
├── db.sqlite3          # 文件型数据库
├── manage.py          # 项目管理文件
├── mysite              # 项目管理目录
│   ├── __init__.py    # 项目初始化文件
│   ├── settings.py    # 配置文件
│   ├── urls.py        # URLConf路由文件
│   └── wsgi.py        # 部署时的配置文件
└── templates          # 模板目录
```

- 搭建mariadb服务器

配置django

在数据库服务器上创建名为dj1903的数据库

```
[root@room8pc16 devweb]# mysql -uroot -ptedu.cn
MariaDB [(none)]> CREATE DATABASE dj1903 DEFAULT CHARSET utf8;
```

配置django

```
# mysite/settings.py
# BASE_DIR设置了外层mysite是项目的根目录
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
ALLOWED_HOSTS = ['*'] # 设置所有的主机均可访问
DATABASES = { # 使用mysql数据库
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dj1903',
        'USER': 'root',
        'PASSWORD': 'tedu.cn',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
LANGUAGE_CODE = 'zh-hans'
TIME_ZONE = 'Asia/Shanghai'
USE_TZ = False # 不使用标准时区

# 在__init__.py中声明将pymysql安装为MySQLdb
```

```
# vimmysite/__init__.py
import pymysql
pymysql.install_as_MySQLdb()
```

启动django服务器监听在0.0.0.0:80

```
(nsd1903) [root@room8pc16 mysite]# python manage.py runserver 0:80
```

生成项目缺省的数据库

```
# 首先查看数据库表
[root@room8pc16 nsd2019]# mysql -uroot -ptedu.cn
MariaDB [(none)]> USE dj1903;
MariaDB [dj1903]> show tables;

# 生成表
(nsd1903) [root@room8pc16 mysite]# python manage.py makemigrations
(nsd1903) [root@room8pc16 mysite]# python manage.py migrate

# 再次查询数据库表
MariaDB [dj1903]> show tables;
```

创建管理员账号

```
(nsd1903) [root@room8pc16 mysite]# python manage.py createsuperuser
```

登陆后台：<http://127.0.0.1/admin>

管理应用

- 项目由一到多个应用构成，如首页，博客，论坛，留言，新闻发布，投票
- 一个应用对应一个目录
- 应用目录可以创建在任何位置，习惯创建在项目根目录下
- 一个应用可以部署到多个项目

创建名为polls的应用

```
(nsd1903) [root@room8pc16 mysite]# python manage.py startapp polls
(nsd1903) [root@room8pc16 mysite]# ls
db.sqlite3  manage.py  mysite  polls  templates
```

将应用部署到项目中

```
# vim mysite/settings.py
INSTALLED_APPS = [
    ...
    'polls',
]
```

项目规划

URL规划

```
http://127.0.0.1/polls/: 投票首页，显示所有的投票项
http://127.0.0.1/polls/1/: 1号问题的投票详情页
http://127.0.0.1/polls/1/result/: 1号问题的投票结果
```

URL授权

- 将每个应用都设计出有相应特点的url，如博客都放到/blog/中，论坛都放到/forum/中，投票都放到/polls/中
- 如果将所有的url与函数的映射关系都放到项目的urls.py中，那么这个文件将会有大量的url
- 可以将每个应用的url授权给应用管理

```
# vim mysite/urls.py
from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^polls/', include('polls.urls')),
]

# vim polls/urls.py
from django.conf.urls import url

urlpatterns = [
]
```

编写polls应用

编写首页

```
1. 编写url与函数的对应关系
# polls/urls.py
from django.conf.urls import url
from . import views    # 从当前目录中导入views模块

urlpatterns = [
    # url(路径正则, 调用的函数, 该url的名字)
    url(r'^$', views.index, name='index'),
]

2. 编写函数
# polls/views.py
from django.shortcuts import render, HttpResponse

def index(request):
    # django将会把http请求作为参数传递给函数，因此，函数必须至少有一个参数
    return HttpResponse('<h1>首页</h1>')
```

3. 访问 `http://x.x.x.x/polls/` 测试

4. 将html页面内容中的所有字符都通过HttpResponse返回是不现实的，所以采用返回html文档的方式修改代码

```
def index(request):  
    return render(request, 'index.html')
```

5. 在项目目录下的templates中创建index.html模板文件

```
# templates/index.html  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>投票首页</title>  
</head>  
<body>  
<h1>投票首页</h1>  
</body>  
</html>
```

编写投票详情页

```
1. urls.py  
urlpatterns = [  
    # url(路径正则, 调用的函数, 该url的名字)  
    url(r'^$', views.index, name='index'),  
    # (\d+)将会把匹配到的数字作为参数传递给detail函数  
    url(r'^(\d+)/$', views.detail, name='detail'),  
]
```

```
2. views.py  
def detail(request, question_id):  
    return render(request, 'detail.html', {'question_id': question_id})  
# 字典中的key将作为前台模板的变量, val是变量的值
```

```
3. detail.html  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>投票详情页</title>  
</head>  
<body>  
<h1>{{ question_id }}号问题投票详情页</h1>  
</body>  
</html>
```

4. 访问 `http://x.x.x.x/polls/数字` 测试

编写投票结果页

```

1. urls.py
urlpatterns = [
    # url(路径正则, 调用的函数, 该url的名字)
    url(r'^$', views.index, name='index'),
    # (\d+)将会把匹配到的数字作为参数传递给detail函数
    url(r'^(\d+)/$', views.detail, name='detail'),
    url(r'^(\d+)/result/$', views.result, name='result'),
]

2. views.py
def result(request, question_id):
    return render(request, 'result.html', {'question_id': question_id})

3. templates/result.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>投票结果页</title>
</head>
<body>
<h1>{{ question_id }}号问题投票结果页</h1>
</body>
</html>

```

模型设计

ORM : 对象关系映射

Object : python的对象

Relationship : 关系型数据库

Mapper : 映射

数据库表对应python中的class

数据库表的字段对应class中的变量

数据库表中的行（记录）对应class的实例

模型设计

问题表 :

问题ID	问题内容	发布时间
1	工资 ?	2019-6-1
2	公司 ?	2019-8-3
3	城市 ?	2018-12-25

选项表 :

选项ID	选项内容	问题ID	票数
1	<5000	1	0
2	5000-8000	1	10
3	华为	2	20

编写models.py

```
from django.db import models

class Question(models.Model): # 父类是固定的
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField()

class Choice(models.Model):
    choice_text = models.CharField(max_length=100)
    votes = models.IntegerField(default=0)
    q = models.ForeignKey(Question)

# 说明：
# 模型没有明确给定主键，django将自动创建一个名为id的字段作为主键
# 实体类对应的表名，默认是：应用名_类名 全部采用小写字母
# 外键在数据库表中的名称是：类变量_id
```

生成表并查看

```
(nsd1903) [root@room8pc16 mysite]# python manage.py makemigrations
(nsd1903) [root@room8pc16 mysite]# python manage.py migrate
MariaDB [dj1903]> show tables;
MariaDB [dj1903]> desc polls_question;
```

数据迁移：将Choice模型的q改为question，观察数据库表中字段的变化

```
# models.py
class Choice(models.Model):
    choice_text = models.CharField(max_length=100)
    votes = models.IntegerField(default=0)
    question = models.ForeignKey(Question)

(nsd1903) [root@room8pc16 mysite]# python manage.py makemigrations
Did you rename choice.q to choice.question (a ForeignKey)? [y/N] y
(nsd1903) [root@room8pc16 mysite]# python manage.py migrate
MariaDB [dj1903]> desc polls_choice;
```

注册模型到后台

```
# polls/admin.py
from django.contrib import admin
from .models import Question, Choice

admin.site.register(Question)
admin.site.register(Choice)

# 访问 http://x.x.x.x/admin/ 添加数据
```

在后台管理页面添加完问题后，问题显示为“Question object”，通过为模型加入magic方法解决。

```
# models.py
class Question(models.Model): # 父类是固定的
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField()

    def __str__(self):
        return self.question_text
```