一、入口映射

**1、URLConfig**

mysite/polls/urls.py

from . import views

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^hello/$', views.hello, name='aa', {'age':12,} ),

]

别名

附加参数

mysite/polls/views.py

def hello(request):

retuan HttpResponse("hello world!!!")

**2、URL正则匹配与参数提取**

mysite/polls/urls.py

urlpatterns = [

url(r'^(?P<name>\w+)/(?P<age>\w+)/$', views.test, name='aa'),

]

mysite/polls/views.py

def test(request, name='', age='')

1. **提取GET/POST信息**

mysite/polls/templates/detail.html

<form action="{% url 'vote' question\_id=question.id %}" method="post">

<input type="radio" name="choice\_id" value="{{ choice.id }}">

</form>

mysite/polls/views.py

def vote(request, question\_id):

choice\_id = request.POST.get('choice\_id')

1. 模板详解

**1、配置模板**

Django项目关于模板的设置是在：mysite/mysite/settings.py

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

#指向django内置后端模板引擎

'DIRS': [], #指定模板目录

'APP\_DIRS': True, #True自动进入应用中查找模板(templates目录)

'OPTIONS': {

... ...

},

},

]

**2、调用模板**

在views.py的对应视图函数中，使用from django.shortcuts import render函数指定模板，并传递上下文对象

from django.shortcuts import render, redirect

from .models import Question, Choice

def index(request):

questions = Question.objects.order\_by('-pub\_date')

#与后端模型交互

return render(request, 'index.html', {'questions': questions})

#调用index.html模板，传递Context变量

**3、模板语法**

1）变量的表示

- 简单变量：{{cname}}

- 对象变量：{{ student.cname}}

- 列表对象：{{ student.1 }}

- 字典对象：{{ student.cname }}

1. 循环结构

{% for choice in question.choice\_set.all %}

<h4>{{ choice.choice\_text }}</h4>

{% endfor %}

1. 条件分支

{% if score >= 90 %}

优秀

{% elif score >= 80 %}

良好

{% else %}

需要努力！

{% endif %}

1. 使用过滤（|管道）

{{ msg }}

{{ msg | truncatewords:20 }} #只显示前20个字符

{{ msg | capfirst }} #首字母大写

... ...

**4、模板继承**

**1）在模板引入bootstrap**

[root@room9pc01 mysite]# ls ../../day03/static

css fonts imgs js

[root@room9pc01 mysite]# cp -r ../../day03/static polls/

#拷贝boostrap的static目录

[root@room9pc01 mysite]# vim polls/templates/index.html

{% load static %}

... ...

<head>

... ...

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">

</head>

<body>

... ...

<script src="{% static 'js/jquery.min.js' %}"></script>

<script src="{% static 'js/bootstrap.min.js' %}"></script>

<script type="text/javascript">

$('#linux-carousel').carousel({

interval : 3000

});

</script>

</body>

1. **制作基础模板**

a、index.html复制一份，命名为base.html

b、将base.html中的个性化信息使用block替换

[root@room9pc01 mysite]# vim polls/templates/base.html

{% load static %}

... ...

<head>

<title>{% block title %}{% endblock %}</title>

</head>

<body>

... #共性页眉：轮询图片

<div class="main">

{% block content %}{% endblock %}

</div>

... #共性页脚

</body>

**3）使用模块继承**

[root@room9pc01 mysite]# vim polls/templates/index.html

{% extends 'base.html' %}

{% load static %}

{% block title %}投票首页{% endblock %}

{% block content %}

<h1>这是投票首页</h1>

<div class="h4">

... ...

</div>

{% endblock %}

**4）静态文件管理**

a：静态文件包括：图像、JavaScript、CSS等

b：关于静态文件管理的设置在：mysite/mysite/settings.py

INSTALLED\_APPS = [

... ...

'django.contrib.staticfiles', #辅助管理静态文件

]

STATIC\_URL = '/static/' #定义静态文件目录名

c：在模板中，使用static模板标签来指定静态文件url

{% load static %}

<img src="{% static "img/text.jpg" %}" alt="My image"/>

d：在应用中，将静态文件存储在static目录下，例如：

mysite/polls/static/imgp/text.jpg

三、模型详解

数据库模型编程又称ORM编程，提供了定义表、增删改查等功能。

1. **定义表与字段**

参考：Day04 -> 三、Django模型 -> 1、ORM映射

[root@room9pc01 mysite]# vim polls/models.py

from django.db import models

class Question(models.Model):

question\_text = models.CharField(

max\_length=200,

null = False,

unique=True)

pub\_date = models.DateField()

def \_\_str\_\_(self):

return '<问题：%s>' % self.question\_text

1. **模型的增删改查**

参考：Day05 -> 一、Django API -> 3~7增删改查

四、自定义后台显示

**1、修改admin.py**

[root@room9pc01 mysite]# vim polls/admin.py

from django.contrib import admin

from .models import Question, Choice

class QuestionAdmin(admin.ModelAdmin): #自定义问题页面后台

list\_display = ('question\_text', 'pub\_date')#显示问题内容和时间

date\_hierarchy = 'pub\_date' #时间轴

list\_filter = ('pub\_date',) #过滤器

search\_fields = ('question\_text',) #搜索框

ordering = ('-pub\_date',) #排序按钮

class ChoiceAdmin(admin.ModelAdmin): #自定义选项页面后台

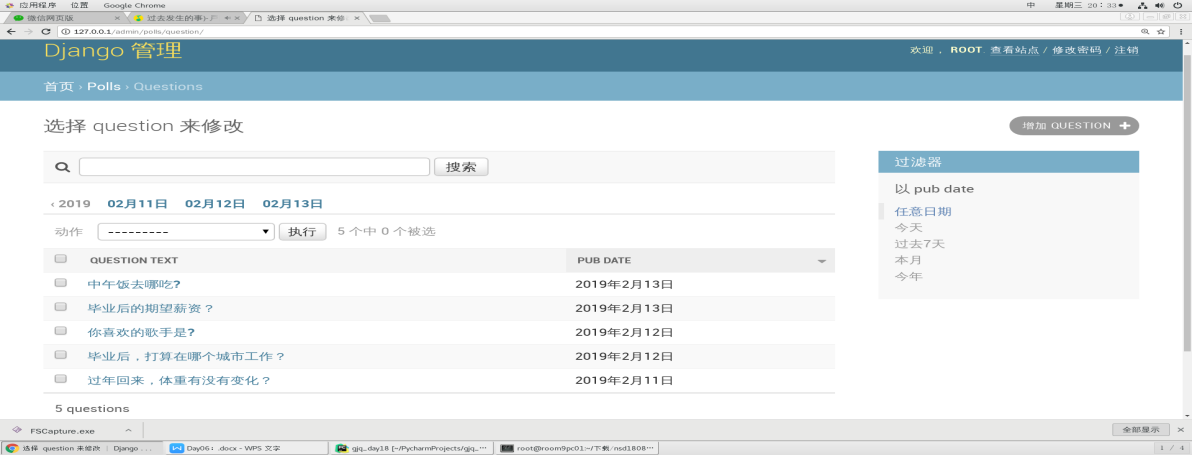
raw\_id\_fields = ('question',) #通过外键查找问题

admin.site.register(Question, QuestionAdmin)

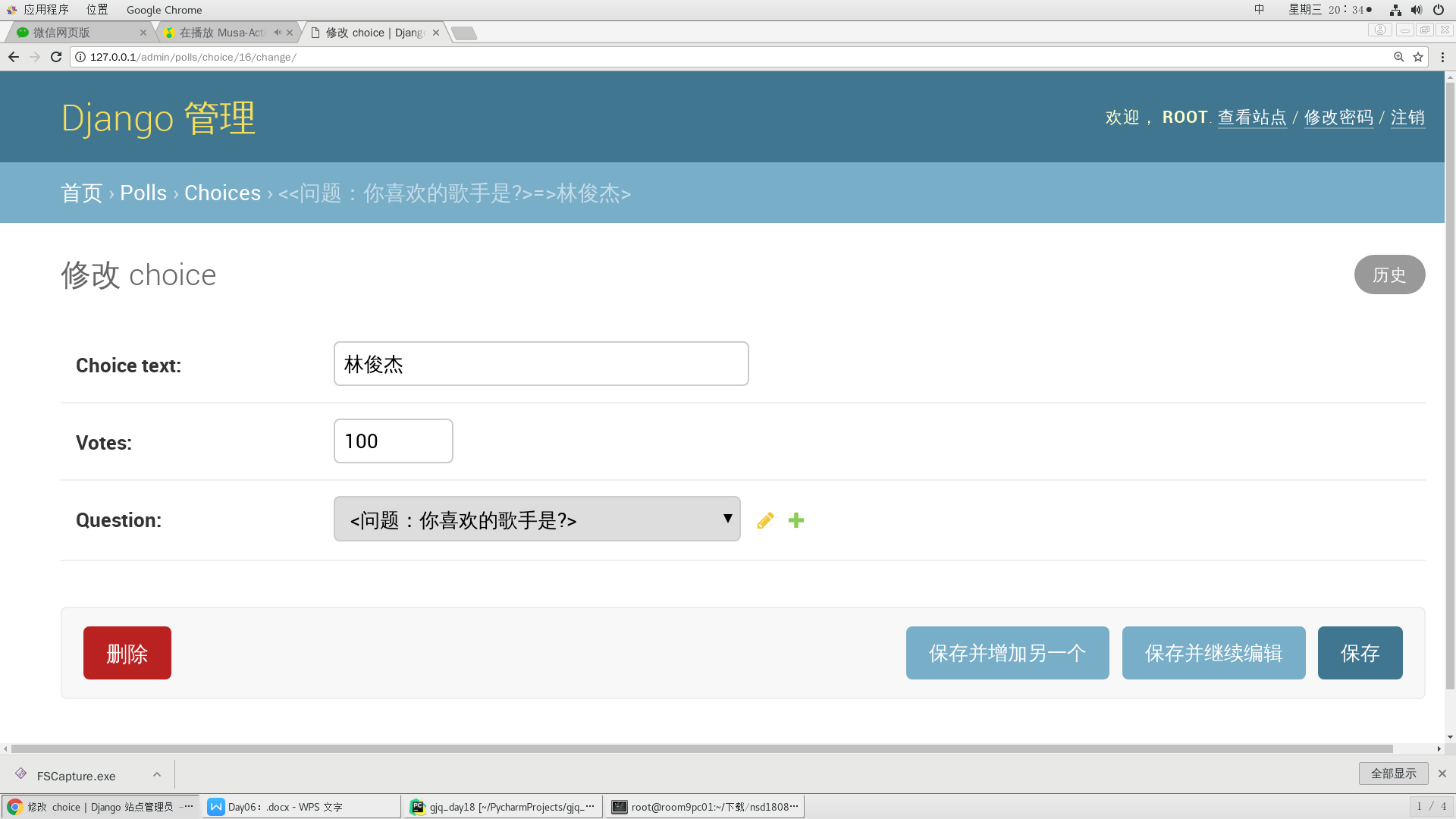
admin.site.register(Choice, ChoiceAdmin)

1. **页面效果**

http://127.0.0.1/admin/polls/question/



http://127.0.0.1/admin/polls/choice/16/change/



案例：编写博客应用

**一、创建应用**

1、生成应用

(djenv) [root@room8pc16 mysite]# python manage.py startapp blog

2、集成应用到项目

# mysite/settings.py

INSTALLED\_APPS = [

... ...

'blog',

]

3、授权，以http://x.x.x.x/blog/开头的URL，交给应用处理

# mysite/urls.py

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^polls/', include('polls.urls')),

url(r'^blog/', include('blog.urls')),

]

# vim blog/urls.py

from django.conf.urls import url

urlpatterns = [

]

**二、规划url**

博客首页: http://x.x.x.x/blog/

具体文章: http://x.x.x.x/blog/1/

**三、创建模型**

1、编写blog/models.py

from django.db import models

class Article(models.Model):

title = models.CharField(max\_length=100, null=False)

pub\_date = models.DateTimeField(auto\_now\_add=True)

#自动使用发布文章时的时间

content = models.TextField()

def \_\_str\_\_(self):

return self.title

2、生成数据表

(djenv) [root@room8pc16 mysite]# python manage.py makemigrations

(djenv) [root@room8pc16 mysite]# python manage.py migrate

3、把博客注册到后台管理界面

# blog/admin.py

from django.contrib import admin

from .models import Article

admin.site.register(Article)

**四、编写视图函数**

1、博客首页视图

# blog/urls.py

from django.conf.urls import url

from . import views

urlpatterns = [

url(r'^$', views.index, name='blog\_index'),

]

# blog/views.py

from django.shortcuts import render

from .models import Article

def index(request):

articles = Article.objects.order\_by('-pub\_date')

return render(request, 'blog\_index.html', {'articles': articles})

2、编写具体文章的视图

# blog/urls.py

url(r'^(?P<article\_id>\d+)/$', views.show\_article, name='show\_article'),

# blog/views.py

def show\_article(request, article\_id):

article = Article.objects.get(id=article\_id)

return render(request, 'show\_article.html', {'article': article})

**五、创建模板**

1、首页模板

(djenv) [root@room8pc16 mysite]# mkdir blog/templates

# blog/templates/blog\_index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>博客首页</title>

</head>

<body>

{% for article in articles %}

<h2>{{ article.title }}</h2>

<h5>{{ article.pub\_date }}</h5>

<div>

{{ article.content | truncatewords:20 }}

</div>

{% endfor %}

</body>

</html>

2、博客文章模板

# blog/templates/show\_article.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>博客文章</title>

</head>

<body>

<h2>{{ article.title }}</h2>

<h5>{{ article.pub\_date }}</h5>

<div>

{{ article.content | linebreaksbr }}

</div>

</body>

</html>

3、修改博客首页，增加超链接

<body>

{% for article in articles %}

<h2>

<a href="{% url 'show\_article' article\_id=article.id %}" target="\_blank">

{{ article.title }}

</a>

</h2>

<h5>{{ article.pub\_date }}</h5>

<div>

{{ article.content | truncatewords:20 }}

</div>

{% endfor %}

</body>