1. Django API
2. **加载Python shell**

(djenv) [root@room9pc01 mysite]# python manage.py shell

>>> from polls.models import Question, Choice #导入模型

**2、模型管理器**

每一个模型默认都有一个**objects管理器**，可通过它实现增删改查。

如： Question.objects.all()

**3、创建问题（增）**

1）采用创建类的实例的方式

>>> from django.utils import timezone

>>> q1 = Question(question\_text='你的偶像?', pub\_date=timezone.now())

>>> q1.save()

1. 通过管理器创建

>>> q2 = Question.objects.create(question\_text='你的年龄?', pub\_date=timezone.now())

1. **创建选项（增）**

1）采用创建类的实例的方式

>>> c1 = Choice(choice\_text='周杰伦', question=q1)

>>> c1.save()

2）通过管理器创建

>>> c2 = Choice.objects.create(choice\_text='王菲', question=q1)

3）通过问题直接创建选项

因为Question和Choice有主外键关系，这是一个一对多的关系，也就是一个问题可以有很多个选项。这样每个问题的实例，都有一个**反向管理器xxx\_set**（xxx是外键所在表名），通过它可以直接创建选项。

>>> c3 = q1.choice\_set.create(choice\_text='林俊杰')

1. 查询或创建

>>> c4 = Choice.objects.get\_or\_create(choice\_text='刘', question=q1)

#get\_or\_create返回的是一个元组(实例, True|False)

**5、删除数据**

>>> q2 = Question(question\_text='你的偶像?', pub\_date=...)

>>> q2.delete()

**6、修改数据**

>>> q1 = Question(question\_text='你的偶像?', pub\_date=...)

>>> q1.question\_text = '你喜欢的歌手是?'

>>> q1.save()

**7、查询**

查询的结果，有的是一个具体的实例，有的是0到多个实例的集合

1）get获取一个实例，如果不是一个实例，则报错

>>> Question.objects.get(id=1) #id=1是id\_\_exact=1的简写

>>> 或Question.objects.get(id\_\_exact=1)

>>> q1 = Question.objects.get(id=1)

>>> q1.question\_text

'过年回来，体重有没有变化？'

2）获取多个问题

>>> qset1 = Question.objects.all()

>>> qset1[0] #第一个问题

>>> qset1[len(qset1) - 1] #最后一个问题

>>> qset1[:10] #前10个问题

>>> for q in qset1: #历遍打印所有问题

... print(q.question\_text)

3）排序

>>> qset2 = Question.objects.order\_by('pub\_date') #升序

>>> qset3 = Question.objects.order\_by('-pub\_date') #降序

>>> for q in qset2:

... print(q.question\_text, q.pub\_date)

4）过滤

数据的属性使用双下划线'\_\_'来表示

>>> Question.objects.filter(question\_text\_\_contains='毕业')

>>> qset4 = Question.objects.filter(question\_text\_\_contains='毕业').order\_by('-pub\_date') #将过滤的内容再进行排序

>>> Question.objects.filter(question\_text\_\_contains='毕业').filter(question\_text\_\_contains='薪资') #多重过滤包含"毕业"、"薪资"的问题

|  |  |
| --- | --- |
| **过滤方法** | **备注** |
| name\_\_exact="abc"或name="abc"  name\_\_iexact="abc" | 精确等于  精确等于，忽略大小写 |
| name\_\_contains="a"  name\_\_icontains="a" | 包含  包含，忽略大小写 |
| age\_\_gt=10 age\_\_lt=10  age\_\_gte=10 age\_\_lte=10 | 数值比较 |
| name\_\_regex="^abc"  name\_\_iregex="^abc" | 正则过滤  正则过滤，忽略大小写 |

1. 修改视图函数与模板

http://x.x.x.x/polls/ #投票首页，显示所有投票项目

http://x.x.x.x/polls/1/ #显示投票详情

http://x.x.x.x/polls/1/result #显示投票结果

http://x.x.x.x/polls/1/vote/ #用于接收投票时提交的表单数据

**1、编写首页**

首页的URL映射已在<Day04.Django管理应用>中编写：

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

1）修改视图函数

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

from django.shortcuts import render

from .models import Question

def index(request):

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

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

#{'questions': questions}相当于把questions=问题集合，发送到模板

2）修改首页模板

Jinja2模板语法：(1)变量放到双花括号中；(2)语句块放在{% %}；(3)在花括号外的是普通字符

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

<body>

<h1>这是投票首页</h1>

{% for question in questions %}

<div>

{{ forloop.counter }}. #forloop是模板语言中的对象

<a href="/polls/{{ question.id }}/" target="\_blank">

<a href="{% url 'detail' question\_id=question.id %}" target="\_blank"> #详情页跳转链接

{{ question.question\_text }}

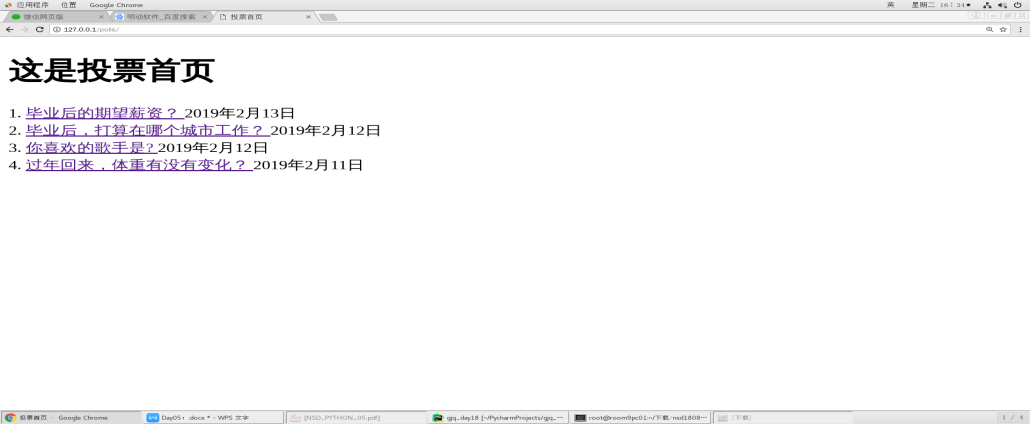
</a>

{{ question.pub\_date }}

</div>

{% endfor %}

</body>



**2、编写投票详情页**

1）编写URL映射

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

from django.conf.urls import url

from . import views

urlpatterns = [

... ...

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

#?P<question\_id>：把(\d+)正则匹配的内容传递给View函数

]

2）编写视图函数

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

from django.shortcuts import render

from .models import Question

... ...

def detail(request, question\_id):

question = Question.objects.get(id=question\_id)

return render(request, 'detail.html', {'question': question})

1. 修改详情页模板

在详情页中列出问题和它相关的选项

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

<body>

<h1>投票详情</h1>

{{ question.question\_text }}

<ul> #使用列表列出问题选项

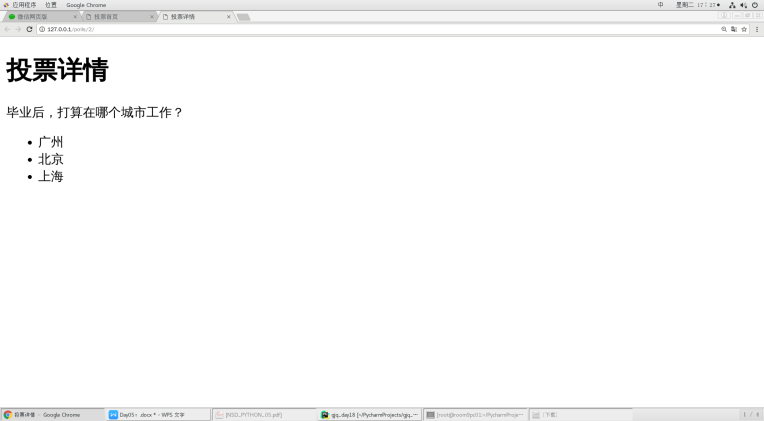
{% for choice in question.choice\_set.all %}

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

{% endfor %}

</ul>

</body>



**3、编写投票结果页**

1）编写URL映射

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

from django.conf.urls import url

from . import views

urlpatterns = [

... ...

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

]

2）编写视图函数

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

from django.shortcuts import render

from .models import Question

... ...

def result(request, question\_id):

question = Question.objects.get(id=question\_id)

return render(request, 'result.html', {'question': question})

3）修改投票结果页模板

以表格形式展示投票结果

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

<body>

<h1>投票结果</h1>

<table border="1px">

<tr>

<td colspan="2">{{ question.question\_text }}</td>

</tr>

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

<tr>

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

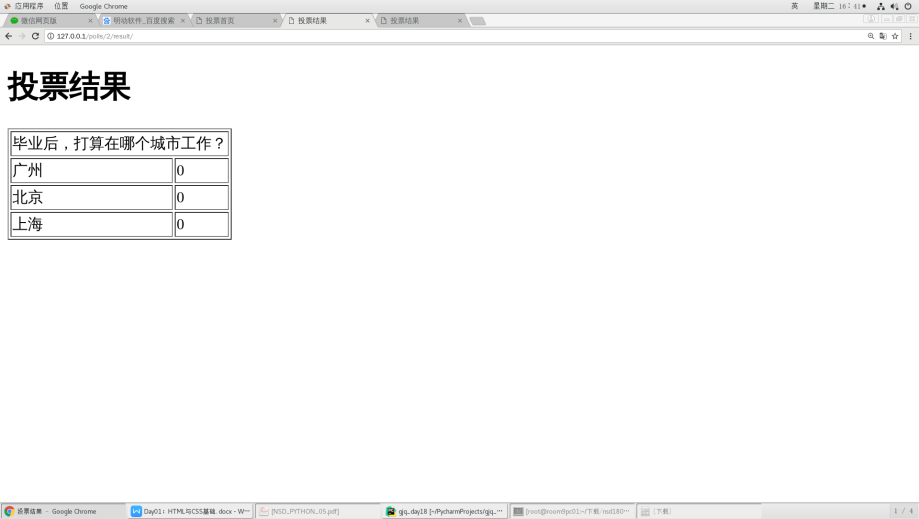
<td>{{ choice.votes }}</td>

</tr>

{% endfor %}

</table>

</body>



**4、完成投票功能、连接前后台**

1）修改detail.html，增加form表单

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

<body>

<h1>投票详情</h1>

{{ question.question\_text }}

<hr>

<form action="" method="post"> #在步骤3)添加

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

<div>

<label>

<input type="radio" name="choice\_id" value="{{ choice.id }}"> #将choice.id以变量名choice\_id提交给服务器

{{ choice.choice\_text }}

</label>

</div>

{% endfor %}

<div><input type="submit" value="投 票"></div>

</form>

</body>

1. 为投票功能增加URL映射

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

from django.conf.urls import url

from . import views

urlpatterns = [

... ...

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

]

1. 设置表单的action地址

<form action="/polls/{{ question.id }}/vote/" ...>

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

#相当于给别名为'vote'的url发送参数为question.id的请求

1. 编写视图函数vote

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

from django.shortcuts import render, redirect

from .models import Question

... ...

def vote(request, question\_id):

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

#获取客户端提交的数据，request.POST是一个类似字典的数据类型，可以使用get方法

question = Question.objects.get(id=question\_id)

#通过objects管理器查询获得1个问题实例

choice = question.choice\_set.get(id=choice\_id)

#通过choice\_set反向管理器查询获得1个选项实例

choice.votes += 1

#将选项votes字段加1

choice.save()

return redirect('result', question\_id=question\_id)

#这里没有使用render，是因为render会将客户机的请求request内容也传递过去。

#redirect表示重定向，相当于是打开一个窗口，写入新的网址进行访问。

#redirect的第一个参数是html文件名，第二个是传参{}

**5、编写投票脚本**

1）在http://127.0.0.1/polls/4/ 右键查看网页源代码

2）找到表单提交的URL "/polls/4/vote/"

3）编写投票脚本：

import requests

url = 'http://127.0.0.1/polls/4/vote/'

data = {"choice\_id": 16}

for i in range(100):

r = requests.post(url, data=data)