# nsd1912-devweb-day04

操作模型

```
# 通过python shell熟悉模型操作
[root@db1 mysite]# python3 manage.py shell
>>> from polls.models import Question, Choice
# 创建问题方法一：实例化
>>> q1 = Question(question_text='散伙饭哪里吃？', pub_date='2020-5-11')
>>> q1.save()
>>> q1.id
3
>>> q1.question_text
'散伙饭哪里吃？'
>>> q1.pub_date
'2020-5-11'

# 创建问题方法二：通过 objects管理器
# django为每个模型都创建了一个名为 objects的管理器，用户可以通过这个管理器进行增删改 查。它的
get_or_create方法返回一个元组 (实例，True/False)
>>> result = Question.objects.get_or_create(question_text='你们的城市解封了吗？',
pub_date='2020-5-10')
>>> result
(<Question: 问题:你们的城市解封了吗？>, True)
>>> q2 = result[0]
>>> q2.id
4
>>> q2.question_text
'你们的城市解封了吗？'
>>> q2.pub_date
'2020-5-10'

# 创建选项方法一：实例化
>>> c1 = Choice(choice_text='KFC', question=q1)
>>> c1.save()

# 创建选项方法二：通过 objects管理器
>>> c2 = Choice.objects.get_or_create(choice_text='全聚德', question=q1)[0]

# 创建选项方法三：通过问题实例创建
# Question和Choice有主外键约束，一个问题可以有一到多个选项。选项的类名叫 Choice，那么问题的选项集就
叫choice_set；如果选项的类名叫 XuanXiang，那么问题的选项集就叫 xuanxiang_set
>>> c3 = q1.choice_set.get_or_create(choice_text='路边烧烤')[0]
>>> c3
<Choice: 问题:散伙饭哪里吃？=>[路边烧烤]>
 >>> c3.id
7
>>> c3.choice_text
'路边烧烤'
>>> c3.votes
0
>>> c3.question
<Question: 问题:散伙饭去哪吃？>



# 查询
# get方法要求必须返回一项内容，否则报错
>>> Question.objects.get(id=1)

<Question: 问题:你期待工作后的工资是多少？>
```

```
# filter方法返回一个查询集，查询集长度不限
>>> Question.objects.filter(id=10)
<QuerySet []>
# all方法返回所有实例构成的查询集
>>> Question.objects.all()
# order_by方法可以根据某一属性排序，默认升序
>>> for q in Question.objects.order_by('pub_date'):
...     print(q.question_text, q.pub_date)
...
你期待进入哪家公司工作？ 2020-05-01 12:00:00
你期待工作后的工资是多少？ 2020-05-09 16:17:00
你们的城市解封了吗？ 2020-05-10 00:00:00
散伙饭哪里吃？ 2020-05-11 00:00:00

# 降序排列
>>> for q in Question.objects.order_by('-pub_date'):
...     print(q.question_text, q.pub_date)
...
散伙饭哪里吃？ 2020-05-11 00:00:00
你们的城市解封了吗？ 2020-05-10 00:00:00
你期待工作后的工资是多少？ 2020-05-09 16:17:00
你期待进入哪家公司工作？ 2020-05-01 12:00:00


# 在查询时，get和filter参数采用相同的形式。它们的参数使用双下划线表示属性和方法
>>> Question.objects.filter(id=1)  # 它是以下的简写
>>> Question.objects.filter(id__exact=1) # id == 1
>>> Question.objects.filter(id__gt=1)    # id > 1
>>> Question.objects.filter(id__gte=1)   # id >= 1
>>> Question.objects.filter(id__lt=1)    # id < 1
>>> Question.objects.filter(id__lte=1)   # id <= 1
>>> Question.objects.exclude(id=1)       # id != 1
# question_text中包含'工作'
>>> Question.objects.filter(question_text__contains='工作')
# 字符串以'你期待'开头
>>> Question.objects.filter(question_text__startswith='你期待')
# 1号发布的问题
>>> Question.objects.filter(pub_date__day=1)
# filter可以支持多次调用
>>> Question.objects.filter(pub_date__day=1).filter(pub_date__month=5)

# 修改问题
>>> q1 = Question.objects.get(question_text='你期待工作后的工资是多少？ ')
>>> q1
<Question: 问题:你期待工作后的工资是多少？ >
>>> q1.question_text = '你期待毕业后第一份工作的工资是多少？ '
>>> q1.save()

# 删除选项
>>> c1 = Choice.objects.get(choice_text='路边烧烤')
>>> c1
<Choice: 问题:散伙饭哪里吃？ =>[路边烧烤]>
>>> c1.delete()
(1, {'polls.Choice': 1})
```

# 制作投票首页

```
# 修改视图函数，取出所有的问题，并按时间降序排列
# polls/views.py
from django.shortcuts import render
from polls.models import Question

# 用户发起的请求，将会被 django作为第一个参数传给函数，所以函数需要有一个参数
def index(request):
    questions = Question.objects.order_by('-pub_date')
    # 通过render函数找到名为index.hml的网页模板文件，返回给用户
    return render(request, 'index.html', {'questions': questions})

# 修改模板文件
# 模板语法中，变量用{{ var }}表示，标签用{% %}来表示。在{}外面的部分都是html语法。
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>投票首页</title>
</head>
<body>
<h1>投票首页</h1>
<ol>
    {% for question in questions %}
        <li>
            <a href="{% url 'detail' question.id %}" target="_blank">
                {{ question.question_text }}
            </a>
            {{ question.pub_date }}
        </li>
    {% endfor %}
</ol>
</body>
</html>
```

# 引入bootstrap

```
# 把day02的static目录拷贝到应用目录下
[root@db1 mysite]# cp -r /root/nsd2019/nsd1912/devweb/day02/static/ polls/
# 修改templates/index.html
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>投票首页</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
</head>
<body>
<div class="container">
    <div id="linux-carousel" class="carousel slide">
        <ol class="carousel-indicators">
            <li class="active" data-target="#linux-carousel" data-slide-to="0"></li>
            <li data-target="#linux-carousel" data-slide-to="1"></li>
            <li data-target="#linux-carousel" data-slide-to="2"></li>
        </ol>
        <div class="carousel-inner">
            <div class="item active">
                <a href="http://www.sogou.com" target="_blank">
                    <img src="{% static 'imgs/first.jpg' %}">
                </a>
            </div>
            <div class="item">
                <img src="{% static 'imgs/second.jpg' %}">
            </div>
            <div class="item">
                <img src="{% static 'imgs/third.jpg' %}">
            </div>
        </div>
        <a href="#linux-carousel" data-slide="prev" class="carousel-control left">
            <span class="glyphicon glyphicon-chevron-left"></span>
        </a>
        <a href="#linux-carousel" data-slide="next" class="carousel-control right">
            <span class="glyphicon glyphicon-chevron-right"></span>
        </a>
    </div>
    <div class="h4">
        <h1 class="text-center text-warning">投票首页</h1>
        <ol>
            {% for question in questions %}
                <li>
                    <a href="{% url 'detail' question.id %}" target="_blank">
                        {{ question.question_text }}
                    </a>
                    {{ question.pub_date }}
                </li>
            {% endfor %}
        </ol>
    </div>
    <div class="text-center h4">
        <a href="#">达内云计算学院</a> &copy; nsd1912

    </div>
```

```
</div>

<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>
</html>
```

## 配置模板继承

- 为了让各个页面拥有一致的布局，并且简化配置可以使用模板继承
- 将各个页面共有的内容放到一个html文件（基础模板）中，个性化的内容使用block占位。具体的每个页面把个性化的内容使用block显示，共性内容继承于模板。

```
[root@localhost mysite]# cp templates/index.html templates/basic.html
# 在basic中，把个性内容使用block替代
# templates/basic.html
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
</head>
<body>
<div class="container">
    <div id="linux-carousel" class="carousel slide">
        <ol class="carousel-indicators">
            <li class="active" data-target="#linux-carousel" data-slide-to="0"></li>
            <li data-target="#linux-carousel" data-slide-to="1"></li>
            <li data-target="#linux-carousel" data-slide-to="2"></li>
        </ol>
        <div class="carousel-inner">
            <div class="item active">
                <a href="http://www.sogou.com" target="_blank">
                    <img src="{% static 'imgs/first.jpg' %}">
                </a>
            </div>
            <div class="item">
                <img src="{% static 'imgs/second.jpg' %}">
            </div>
            <div class="item">
                <img src="{% static 'imgs/third.jpg' %}">
            </div>
        </div>
        <a href="#linux-carousel" data-slide="prev" class="carousel-control left">
            <span class="glyphicon glyphicon-chevron-left"></span>
        </a>
        <a href="#linux-carousel" data-slide="next" class="carousel-control right">
            <span class="glyphicon glyphicon-chevron-right"></span>
        </a>
    </div>
    {% block content %}{% endblock %}
    <div class="text-center h4">
        <a href="#">达内云计算学院</a> &copy; nsd1912
    </div>
</div>

<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>
</html>
```

```
# 在index.html中，继承basic.html，把个性内容写到block中
{% extends 'basic.html' %}
{% block title %}投票首页{% endblock %}
{% block content %}
    <div class="h4">
        <h1 class="text-center text-warning">投票首页</h1>
        <ol>
            {% for question in questions %}
                <li>
                    <a href="{% url 'detail' question.id %}" target="_blank">
                        {{ question.question_text }}
                    </a>
                    {{ question.pub_date }}
                </li>
            {% endfor %}
        </ol>
    </div>
{% endblock %}
```

## 制作投票详情页

```
# polls/views.py
def detail(request, qid):
    # qid用于接收从url传来的参数
    question = Question.objects.get(id=qid)
    # 字典中的内容将以关键字参数 key=val的形式传给网页模板，如 qid=10
    return render(request, 'detail.html', {'question': question})

# templates/detail.html
{% extends 'basic.html' %}
{% block title %}投票详情{% endblock %}
{% block content %}
    <div class="h4">
        <h1 class="text-center text-warning">
            {{ question.id }}号问题的投票详情
        </h1>
        <h2>{{ question.question_text }}</h2>
        <form action="" method="post">
            {% csrf_token %}
            {% for choice in question.choice_set.all %}
                <div class="radio">
                    <label>
                        <input type="radio" name="cid" value="{{ choice.id }}">
                        {{ choice.choice_text }}
                    </label>
                </div>
            {% endfor %}
            <input class="btn btn-primary" type="submit" value="提 交">
        </form>
    </div>
{% endblock %}
```

# 实现投票功能

- 投票功能的实现，需要执行函数
- 在django中执行函数可以通过访问url来实现
- 表单的action行为指定点击提交按钮时访问的url

```
# polls/urls.py
    url(r'^(\d+)/vote/$', views.vote, name='vote'),

# polls/views.py
from django.shortcuts import render, redirect
... ...
def vote(request, qid):
    # 取出待投票的问题
    question = Question.objects.get(id=qid)
    # 取出前端网页发送过来的选项 id
    # request保存的是用户请求，它是一个像字典一样的结构，可以通过 request.POST得到post数据
    # request.post也是一个字典结构，可以通过 key取出val
    cid = request.POST.get('cid')
    # 取出choice_id对应的选项实例
    choice = question.choice_set.get(id=cid)
    choice.votes += 1  # 票数加1
    choice.save()
    # 投票完成后，跳转到投票结果页， result的结果url的名字
    return redirect('result', qid)

# 修改detail.html中form的action
        <form action="{% url 'vote' question.id %}" method="post">
```

# 制作投票结果页

```python
# polls/views.py
def result(request, qid):
    question = Question.objects.get(id=qid)
    return render(request, 'result.html', {'question': question})

# templates/result.html
{% extends 'basic.html' %}
{% block title %}投票结果{% endblock %}
{% block content %}
    <div class="h4">
        <h1 class="text-center text-warning">
            {{ question.id }}号问题的投票结果
        </h1>
        <h2>{{ question.question_text }}</h2>
        <table class="table table-bordered table-striped table-hover">
            {% for choice in question.choice_set.all %}
                <tr>
                    <td>{{ choice.choice_text }}</td>
                    <td>{{ choice.votes }}</td>
                </tr>
            {% endfor %}
        </table>
        <a class="btn btn-primary" href="{% url 'index' %}">返回首页</a>
    </div>
{% endblock %}
```