nsd1907_devweb_day04

python api

```
# 打开python shell
(nsd1907) [root@room8pc16 mysite]# python manage.py shell
# 导入模型
>>> from polls.models import Question, Choice
# 创建问题方法一：直接创建实例
>>> q1 = Question(question_text='元旦放几天假？', pub_date='2019-12-25')
>>> q1.save()
# 创建问题方法二：使用objects管理器
# django为每个class都创建了一个名为objects的管理器，通过它可以实现对模型的增删改查操作
# get_or_create方法返回的是一个元组：(问题实例, True/False)。数据库中没有相关记录则创建，返回True；如果数据库中有记录则取出，返回Fase
result1 = Question.objects.get_or_create(question_text='春节放几天假', pub_date='2019-12-01')
>>> result1
(<Question: 问题: 春节放几天假>, True)
result2 = Question.objects.get_or_create(question_text='春节放几天假', pub_date='2019-12-01')
>>> result2
(<Question: 问题: 春节放几天假>, False)

# 创建选项方法一：实例化
>>> c1 = Choice(choice_text='1天', question=q1)
>>> c1.save()
# 创建选项方法二：通过objects管理器
>>> c_result1 = Choice.objects.get_or_create(choice_text='2天', question=q1)
# 创建选项方法三：通过问题实例的choice_set管理器创建
# 因为Question和Choice具有主外键约束，一个问题可以有很多选项。django为问题实例创建了名为xxxx_set的管理器（选项类名为Choice，管理器名为choice_set）。
>>> c_result2 = q1.choice_set.get_or_create(choice_text='3天')

# 修改、更新。只要将变量重新赋值即可
>>> q1.question_text = '你们公司元旦放几天假？'
>>> q1.save()

# 删除。调用实例的delete方法
>>> result1
(<Question: 问题: 春节放几天假>, True)
>>> q2 = result1[0]
>>> q2
<Question: 问题: 春节放几天假>
>>> q2.delete()
(1, {'polls.Choice': 0, 'polls.Question': 1})

# 查询
# 查询全部
>>> Question.objects.all()
```

```
# 查询全部，根据pub_date进行排序
>>> for q in Question.objects.order_by('pub_date'):
...   print(q.question_text, q.pub_date)
# 查询全部，根据pub_date进行降序排序
>>> for q in Question.objects.order_by('-pub_date'):
...   print(q.question_text, q.pub_date)

# get要求返回一个结果，否则报错
>>> Question.objects.get(id=10)  # 报错，不存在
>>> Question.objects.get(id__gt=1)  # 报错，结果有多个
# filter返回0到多个对象构成的列表
>>> Question.objects.filter(id=10)
<QuerySet []>
>>> Question.objects.filter(id__gt=1)
<QuerySet [<Question: 问题: 你期待哪家公司给你发Offer？>, <Questio放几天假？>]>
# 在django中，判断时，使用双下划线表示一个对象的属性或方法
>>> Question.objects.get(id__exact=1)
<Question: 问题: 第一份工作，你期待的工资是多少？>
# 上面的方法简写为以下形式：
>>> Question.objects.get(id=1)
<Question: 问题: 第一份工作，你期待的工资是多少？>

>>> Question.objects.filter(id__lt=2)  # id < 2
<QuerySet [<Question: 问题: 第一份工作，你期待的工资是多少？>]>
>>> Question.objects.filter(id__lte=2)  # id <= 2
<QuerySet [<Question: 问题: 第一份工作，你期待的工资是多少？>, <Qu司给你发Offer？>]>
>>> Question.objects.filter(id__gt=2)  # id > 2
<QuerySet [<Question: 问题: 你们公司元旦放几天假？>]>
>>> Question.objects.filter(id__gte=2)  # id >= 2
<QuerySet [<Question: 问题: 你期待哪家公司给你发Offer？>, <Questio放几天假？>]>

# 查询12月的问题
>>> Question.objects.filter(pub_date__month=12)
# 查询小于12月的问题
>>> Question.objects.filter(pub_date__month__lt=12)
# 查询以'你'开头的问题
>>> Question.objects.filter(question_text__startswith='你')
# 查询以'你'开头、12月的问题
>>> Question.objects.filter(question_text__startswith='你')\
.filter(pub_date__month=12)
```

完成投票首页

```
# 修改函数
# 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})
... ...
```

```
# 修改模板文件，展示相关的问题
# 在模板文件里，{}内的部分是模板语法，{}外是html语法
# {% url 'detail' question.id %}，detail是在urls.py中定义的名称
# 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>
<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

```
# 将devweb/day02/static拷贝到应用目录下
# django默认在每个应用的static目录中寻找静态文件
(nsd1907) [root@room8pc16 mysite]# cp -r ../../day02/static polls/

# 在index.html中引入bootstrap
# templates/index.html
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>投票首页</title>
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
</head>
<body>
<div class="container">
    <div class="row">
        <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>
    <div class="row 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="row h4 text-center">
        达内云计算 <a href="#">NSD1907</a>
    </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>
```

模板继承

- 为了实现各个页面的风格统一，可以使用模板继承的方式
- 创建一个基础模板，将共性内容写到基础模板文件中；个性内容，使用block占位

- 其他页面继承模板，共性内容不再需要，只要将个性内容写到相应的block中

```
# 拷贝index.html为base.html
# 修改base.html。block后面的title和content是自定义的字符串
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
</head>
<body>
<div class="container">
    <div class="row">
        <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>
    <div class="row h4">
        {% block content %}{% endblock %}
    </div>
    <div class="row h4 text-center">
        达内云计算 <a href="#">NSD1907</a>
    </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>

# 修改index.html。将共性内容删除，个性内容写到相应的block中
{% extends 'base.html' %}
{% load static %}
{% block title %}投票首页{% endblock %}
{% block content %}
    <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>
{% endblock %}

# detail.html / result.html使用模板继承
# templates/detail.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票详情{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question_id }}号问题投票详情</h1>
{% endblock %}

# templates/result.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票结果{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question_id }}号问题投票结果</h1>
{% endblock %}
```

制作投票详情页

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

# templates/detail.html
```

```
{% extends 'base.html' %}
{% load static %}
{% block title %}投票详情{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question.id }}号问题投票详情</h1>
    <h2>{{ question.question_text }}</h2>
    <form action="{% url 'vote' question.id %}" method="post" class="h3">
        {% csrf_token %}
        {% for choice in question.choice_set.all %}
            <div class="checkbox">
                <label>
                    <input type="radio" name="choice_id" value="{{ choice.id }}">
                    {{ choice.choice_text }}
                </label>
            </div>
        {% endfor %}
        <div class="form-group">
            <input class="btn btn-primary" type="submit" value="提 交">
        </div>
    </form>
{% endblock %}
```

实现投票功能

- 实现投票功能，需要执行一个函数。函数用于将数据库中相应的votes字段加1。
- 执行投票函数需要访问一个url

```
# 为投票功能设计url
# polls/urls.py
... ...
    url(r'^(\d+)/vote/$', views.vote, name='vote'),
... ...

# 编写投票函数
# polls/views.py
from django.shortcuts import render, redirect
... ...
def vote(request, question_id):
    # 取出问题
    question = Question.objects.get(id=question_id)
    # request是一个对象，它的POST属性是一个字典，字典中存储着表单数据
    choice_id = request.POST.get('choice_id')
    # 获取choice_id对应的选项实例
    choice = question.choice_set.get(id=choice_id)
    # 将选项的票数加1
    choice.votes += 1
    choice.save()
    # 投票完成后，跳转到投票结果页
    return redirect('result', question.id)
```

制作结果页

```
# 取出问题，发给模板
# polls/views.py
... ...
def result(request, question_id):
    question = Question.objects.get(id=question_id)
    return render(request, 'result.html', {'question': question})
... ...

# 在模板页，显示投票结果
# templates/result.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票结果{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question.id }}号问题投票结果</h1>
    <h2>{{ question.question_text }}</h2>
    <table class="table table-hover table-striped">
        <thead class="bg-primary">
            <tr class="text-center">
                <td>选项</td>
                <td>票数</td>
            </tr>
        </thead>
        <tbody>
            {% for choice in question.choice_set.all %}
                <tr>
                    <td>{{ choice.choice_text }}</td>
                    <td>{{ choice.votes }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
    <a href="{% url 'index' %}">返回首页</a>
{% endblock %}
```