nsd1903_devweb_day04

django在线文档：https://docs.djangoproject.com

python shell

通过python shell操作模型

```
(nsd1903) [root@room8pc16 mysite]# python manage.py shell
>>> from polls.models import Question, Choice
# 增加问题方法一：创建实例
>>> q1 = Question(question_text="你打算到哪个城市找工作？", pub_date="2019-8-20")
>>> q1.save()

# 增加问题方法二：通过objects管理器
# django为每个模型都创建了名为objects的管理器，通过objects可以实现对模型的各种操作
>>> q2 = Question.objects.create(question_text="放假出游去哪玩？", pub_date="2019-8-15")

# 创建选项方法一：创建实例
>>> c1 = Choice(choice_text="成都", question=q1)
>>> c1.save()

# 创建选项方法二：通过objects管理器
>>> c2 = Choice.objects.create(choice_text="石家庄", question=q1)

# 创建选项方法三：通过问题实例的choice_set创建
# 因为Question和Choice有一对多的关系，django为问题实例创建了choice_set管理器，用于找到该问题的所有选项。如选项类名为xuanxiang，那么就是xuanxiang_set。
>>> c3 = q1.choice_set.create(choice_text="上海")

# 查询操作也是通过模型的objects管理器实现的
# 获取所有的问题
>>> qset1 = Question.objects.all()
>>> for q in qset1:
...   print(q.id, q.question_text, q.pub_date)
...
1 你期待的工资是多少？ 2019-08-21 17:26:00
2 你期待哪家公司给你发Offer？ 2019-08-01 12:00:00
3 散伙饭在哪吃？ 2019-08-10 18:00:00
4 你打算到哪个城市找工作？ 2019-08-20 00:00:00
5 放假出游去哪玩？ 2019-08-15 00:00:00

# get方法只能获取一个实例，0或多个实例都会报错
>>> q1 = Question.objects.get(question_text="你期待的工资是多少？")
>>> q1.id
1
>>> q1.question_text
'你期待的工资是多少？'
>>> q1.pub_date
```

```
datetime.datetime(2019, 8, 21, 17, 26)

# filter方法得到过滤后的集合，可以是0到多项，过滤条件与get的书写方式一样
>>> Question.objects.filter(id=1)
<QuerySet [<Question: 你期待的工资是多少？>]>
>>> Question.objects.filter(id=100)
<QuerySet []>

# 查询条件：在django中，查询时某一对象的属性不再使用句点表示，而是使用双下划线
>>> Question.objects.filter(id=1)   # 它是以下的简写
>>> Question.objects.filter(id__exact=1)
>>> Question.objects.filter(id__gt=2)   # id > 2
>>> Question.objects.filter(id__lt=2)   # id < 2
>>> Question.objects.filter(question_text__contains='工资')  # 包含工资
# 字符串以“你期待”开头
>>> Question.objects.filter(question_text__startswith="你期待")
# 发布时间的月份是8月的
>>> Question.objects.filter(pub_date__month=8)

# 修改模型
>>> q = Question.objects.get(question_text="你打算到哪个城市找工作？")
>>> q.question_text = "你计划到哪个城市工作？"
>>> q.save()

# 删除模型
>>> q = Question.objects.get(question_text="放假出游去哪玩？")
>>> q.delete()
```

修改视图函数

```
一、投票首页
# 1. 修改index，取出所有的问题，按发布时间降序排列
# views.py
from .models import Question

def index(request):
    # 取出所有问题，根据pub_date降序排列
    questions = Question.objects.order_by('-pub_date')
    return render(request, 'index.html', {'questions': questions})

# 2. 修改模板
# 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>
{{ questions }}
</body>
```

```
</html>

# 3. 完善模板，使得question可以按需求显示
<body>
<h1>投票首页</h1>
{% for question in questions %}
    <p>
        {{ forloop.counter }}.
        <a href="{% url 'detail' question.id %}" target="_blank">
            {{ question.question_text }}
        </a>
        {{ question.pub_date }}
    </p>
{% endfor %}
</body>
# 说明：{% %}是模板的标签，{{}}表示变量。fooloop.counter是模板语言的循环计数器，记录当前是第几次循环。
{% url 'detail' question.id %}，在urls.py中已经为问题详情url命名为detail，这里跳转的就是detail
url，这个url接受一个数字参数，表示问题的ID号，所以将question.id传递过去。
```

在模板中引入**bootstrap**

```
# 默认情况下，django在应用的static目录查找静态文件
# 将day02中的static拷贝到polls应用目录
(nsd1903) [root@room8pc16 mysite]# cp -r ../../day02/static polls/

# 在index.html中引入boostrap
{% 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>

# 修改index.html模板。
{% 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>

    <h1 class="text-warning text-center">投票首页</h1>
    <div class="h4" style="margin-top: 30px">
        {% for question in questions %}
        <p>
            {{ forloop.counter }}.
            <a href="{% url 'detail' question.id %}" target="_blank">
                {{ question.question_text }}
            </a>
            {{ question.pub_date }}
        </p>
    {% endfor %}
    </div>
    <div class="h4 text-center" style="margin-top: 30px">
        达内云计算学院 <a href="">nsd1903</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>
```

使用模板继承

- 为了使得所有的网页模板具有一致的风格，可以采用模板继承

- 将每个页面共性的内容放到基础模板中
- 基他页面继承基础模板

```
# 将index.html拷贝一份，改名为base.html。在base.html中将个性内容删除，使用block占位。title标题替换掉，body的主体h1标签和for循环所在div用block替换。
# templates/base.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="h4 text-center" style="margin-top: 30px">
        达内云计算学院 <a href="">nsd1903</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>

# index.html继承base.html，把共性部分删除，个性部分放到相应的block中
{% extends 'base.html' %}
{% load static %}
{% block title %}投票首页{% endblock %}
{% block content %}
<h1 class="text-warning text-center">投票首页</h1>
<div class="h4" style="margin-top: 30px">
    {% for question in questions %}
    <p>
        {{ forloop.counter }}.
        <a href="{% url 'detail' question.id %}" target="_blank">
            {{ question.question_text }}
        </a>
        {{ question.pub_date }}
    </p>
{% endfor %}
</div>
{% endblock %}


# 测试详情页继承结果, detail.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票详情页{% endblock %}
{% block content %}
<h1>{{ question_id }}号问题投票详情页</h1>
{% endblock %}
```

完成投票详情页

将某一问题和选项显示出来

```
# 修改视图函数views.py
def detail(request, question_id):
    question = Question.objects.get(id=question_id)
    return render(request, 'detail.html', {'question': question})

# 修改模板文件detail.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票详情页{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question.id }}号问题投票详情页</h1>
    <h2>{{ question.question_text }}</h2>
```

```
    <div class="h4">
        <form action="" method="post">
            {% csrf_token %}
            {% for choice in question.choice_set.all %}
                <div class="radio">
                    <label>
                        <input type="radio" name="choice_id" 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

- 投票功能就是将数据库中某一记录的票数加1
- 通过函数取出选项对应的实例，将实例的votes属性加1
- 在django中，执行函数通过访问url实现
- 表单的action是点击提交按钮时访问的url。访问该url时，还会把表单中的数据提交给url

```
# 1. 创建url用于和投票函数功能关联
# polls/urls.py
urlpatterns = [
    ... ...
    url(r'^(\d+)/vote/$', views.vote, name='vote'),
]
# 2. 修改detail.html中的表单，给action填加网址
   <form action="{% url 'vote' question.id %}" method="post">

# 3. 编写vote函数，实现计数功能
# views.py
def vote(request, question_id):
    question = Question.objects.get(id=question_id)
    # request是用户的请求，POST是请求中的字典，保存着提交数据
    choice_id = request.POST.get('choice_id')
    # 通过问题的选项集取出对应的选项实例
    choice = question.choice_set.get(id=choice_id)
    choice.votes += 1  # 选项票数加1
    choice.save()

    # 使用重定向，url将会变成result的url，如果仍然使用render
    # 那么url显示的是vote，但是页面是result的页面
    return redirect('result', question_id)
```

完成结果页

和投票详情页类似，只是在页面中显示的形式不一样。

```
# 1. 修改视图函数
# views.py
def result(request, question_id):
    question = Question.objects.get(id=question_id)
    return render(request, 'result.html', {'question': question})

# 2. 完成模板
# result.html
{% extends 'base.html' %}
{% load static %}
{% block title %}投票结果页{% endblock %}
{% block content %}
    <h1 class="text-center text-warning">{{ question.id }}号问题投票结果页</h1>
    <table class="table table-hover table-striped h4">
        <thead class="bg-primary h3">
            <tr>
                <td colspan="2">{{ question.question_text }}</td>
            </tr>
        </thead>
        {% for choice in question.choice_set.all %}
            <tr>
                <td>{{ choice.choice_text }}</td>
                <td>{{ choice.votes }}</td>
            </tr>
        {% endfor %}
    </table>
{% endblock %}
```