nsd1902_devweb_day04

django官方文档页：https://docs.djangoproject.com/zh-hans/2.2/实体类的声明，以便在后台中显示指定的字符串。

```
from django.db import models

# Create your models here.

class Question(models.Model):
    question_text = models.CharField(max_length=200, unique=True)
    pub_date = models.DateTimeField()

    def __str__(self):
        return self.question_text

class Choice(models.Model):
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
    question = models.ForeignKey(Question)

    def __str__(self):
        return "%s=>%s" % (self.question, self.choice_text)
```

django api

打开python shell

```
(nsd1902) [root@room8pc16 mysite]# python manage.py shell
>>> from polls.models import Question, Choice
```

操作数据库

创建问题记录有两种方法：

- 实例化
- 在django中，每个class都有一个名为objects的管理器，通过这个管理器创建

```
# 实例化
>>> q1 = Question(question_text='散伙饭去哪吃？', pub_date='2019-07-24 9:00:00')
>>> q1.save()

>>> from datetime import datetime
>>> dt1 = datetime.now()
>>> q2 = Question(question_text='散伙吃什么？', pub_date=dt1)
>>> q2.save()

# 通过objects管理器
>>> q3 = Question.objects.create(question_text="你打算到哪个城市找工作？", pub_date="2018-12-1
12:00:00")
```

创建选项的三种方法

- 实例化
- 通过objects管理器
- 通过问题的choice_set创建选项

```
# 实例化
>>> c1 = Choice(choice_text='北京', question=q3)
>>> c1.save()

# 通过objects管理器
>>> c2 = Choice.objects.create(choice_text='上海', question=q3)

# 因为问题和选项存在着一对多的关系，也就是一个问题可以对应多个选项。因为选项类名字是Choice，所以问题实例
都有一个choice_set。如果选项类名字是XuanXiang，那么问题实例就有一个xuanxiang_set。choice_set也是一个
管理器，它也有和objects一样的方法。
>>> c3 = q3.choice_set.create(choice_text='广州')
```

查询

查询的方法：

- get：要求必须返回一个实例，否则报错
- filter：返回0到多个实例的查询集

```
>>> Question.objects.get(id=1)
<Question: 你期待的工资是多少？>
>>> Question.objects.filter(id=1)
<QuerySet [<Question: 你期待的工资是多少？>]>

>>> q1 = Question.objects.get(id=1)
>>> q1.question_text
'你期待的工资是多少？'
>>> q1.pub_date
datetime.datetime(2019, 7, 23, 9, 32)

>>> qset1 = Question.objects.filter(id=1)
>>> len(qset1)
```

```
1
>>> q2 = qset1[0]   # 取出下标为0的实例
>>> q2.question_text
'你期待的工资是多少？'
```

查询条件

django查询时，使用属性加双下划线的方法进行比较

```
>>> Question.objects.filter(id__exact=1)  # 简写为以下形式
>>> q1 = Question.objects.get(id=1)
>>> Question.objects.filter(id__gt=2)   # id > 2
>>> Question.objects.filter(id__lt=2)   # id < 2
>>> Question.objects.filter(id__gte=2)  # id >= 2
>>> Question.objects.filter(id__lte=2)  # id <= 2

# 以 xxx 开头
>>> Question.objects.filter(question_text__startswith='你期待')
# 以 XXX 结尾
>>> Question.objects.filter(question_text__endswith='？')
# 包含 XXX
>>> Question.objects.filter(question_text__contains='工作')
# 年份是2018
>>> Question.objects.filter(pub_date__year=2018)
# 月份是7
>>> Question.objects.filter(pub_date__month=7)
```

取出数据

```
# 取出全部数据
>>> Question.objects.all()   # 返回查询集
>>> Question.objects.order_by('pub_date')  # 按pub_date升序排列
>>> Question.objects.order_by('-pub_date') # 按pub_date降序排列
# 先过滤，再排序
>>> Question.objects.filter(id__gt=2).order_by('-pub_date')
# 在过滤的结果集上继续进行过滤
>>> Question.objects.filter(id__gt=2).filter(id__lt=4)
```

修改和删除

```
# 修改前先获取到实例
>>> q1 = Question.objects.get(id=3)
>>> q1
<Question: 散伙饭去哪吃？>
# 修改就是属性重新赋值
>>> q1.question_text = '散伙饭去什么地方吃？'
>>> q1.save()

# 删除
>>> q2 = Question.objects.get(question_text__contains='哪个城市找工作')
>>> print(q2)
你打算到哪个城市找工作？
```

```
>>> q2.delete()   # delete后，数据库中已经删除对应的记录
>>> print(q2)   # 仍然存在q2实例，只不过数据库中没有记录与之对应
>>> q2.save()   # 再次保存，在表中创建新的记录
```

制作投票首页

1. 首页显示所有的问题，所以首页视图函数需要读取model模型，将所有的问题取出。

```
# polls/views.py
from .models import Question
def index(request):
    question_set = Question.objects.order_by('-pub_date')
    return render(request, 'index.html', {'questions': question_set})
```

2. 修改web页面

```
<body>
<h1>投票首页</h1>
{#<div>{{ questions }}</div>#}
<ol>
        <li>
            <a href="{% url 'detail' question.id %}" target="_blank">
                {{ question.question_text }}
            </a>
            {{ question.pub_date }}
        </li>
    {% endfor %}
</ol>
</body>
```

模板语言

https://docs.djangoproject.com/en/1.11/ref/templates/language/

```
# 变量通过{{var}}来表示

# for 循环，变量在标签中不需要使用{{}}
<ul>
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
</ul>

# 判断
{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
```

```
    No athletes.
{% endif %}
```

3. 引入bootstrap

```
# 把day02的static目录拷贝到应用目录下
(nsd1902) [root@room8pc16 mysite]# ls -d polls/static/
polls/static

# 静态文件目录，通过settings.py的STATICFILES_DIRS设置
  # 项目目录的static目录
  # 应用目录的static目录

# 修改index.html，第一行加入以下内容：
{% load static %}
# 在index.html的head标签中，加入以下内容：
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
```

4. 配置index.html的页面

```
{% 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="row h4">
        <div class="col-sm-12">
            <h1 class="text-center text-warning">投票首页</h1>
            {#<div>{{ questions }}</div>#}
            <ol>
                {% for question in questions %}
                    <li>
            {#        detail是urls.py中为详情页起的名字，它接受一个参数#}
                      <a href="{% url 'detail' question.id %}" target="_blank">
                          {{ question.question_text }}
                      </a>
                      {{ question.pub_date }}
                    </li>
                {% endfor %}
            </ol>
        </div>
    </div>
    <div class="row text-center h4">
        <div class="col-sm-12">
            <a href="#">tedu nsd1902</a>
        </div>
    </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>
```

5. 使用模板继承

制作网页时，多个页面呈现的形式是一样的，可以先创建一个基础模板，把共性内容写到基础模板中。具体的模板文件都继承于基础模板，把个性化内容写到模板文件。

```
# templates/base.html   保留各个页面的共性部分，个性部分用block标签代替
{% 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>
    <div class="row h4">
        <div class="col-sm-12">
            {% block content %}{% endblock %}
        </div>
    </div>
    <div class="row text-center h4">
        <div class="col-sm-12">
            <a href="#">达内云计算 nsd1902</a>
        </div>
    </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>
```

```
# 将index.html中共性部分删除，个性部分写到对应的block中
# index.html
{% 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 %}
```

制作详情页

1. 先把模板页继承base.html

```
# 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 %}
```

2. 修改视图函数

详情视图，只要把对应ID的问题取出即可

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

3. 修改视图函数

```
# templates/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>
    <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 %}
        <div class="form-group">
            <input class="btn btn-primary" type="submit" value="提 交">
        </div>
    </form>
{% endblock %}
```

实现投票功能

在投票详情页，当用户提交表单时，将会把数据post给action对应的URL。

为选项投票就是把选项的票数votes加1，通过函数实现投票功能。在django里，访问一个url就会执行函数。

1. url

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

2. 视图函数

```
def vote(request, question_id):
    question = Question.objects.get(id=question_id)
    choice_id = request.POST.get('choice_id')  # 从用户post的表单中取出choice_id
    choice = question.choice_set.get(id=choice_id)  # 获取选项实例
    choice.votes += 1
    choice.save()

    return redirect('result', question_id)  # 重定向到result页面
```

3. 修改表单的action值

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

完成投票结果页

1. 视图函数

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

2. 修改模板

```
{% 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>
    <h2>{{ question.question_text }}</h2>
    <table class="table table-bordered table-hover table-striped">
        <thead class="bg-primary text-center">
            <tr>
                <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>
{% endblock %}
```

附：

前台页面模板：https://coreui.io/

django2 by example: http://www.conyli.cc/django-2-by-example