Django笔记

笔记本:	我的第一个笔记本		
创建时间:	2019/10/24 13:42	**更新时间:**	2019/10/24 15:10
作者:	631149573@qq.com		

基本应用

创建项目

django-admin startproject 项目名称

目录结构:

- **manage.py**.是项目管理文件，通过它管理项目。
- 与项目同名的目录。
- **init**.py是一个空文件，作用是这个目录test1可以被当作包使用。
- **settings.py**.是项目的整体配置文件。
- **urls.py**.是项目的URL配置文件。
- **wsgi.py**.是项目与WSGI兼容的Web服务器入口

创建应用

python manage.py startapp 应用名称

应用目录结构

- **init.py**.是一个空文件，表示当前目录booktest可以当作一个python包使用。
- **tests.py**.文件用于开发测试用例，在实际开发中会有专门的测试人员，这个事情不需要我们来做。
- **models.py**.文件跟数据库操作相关。
- **views.py**.文件跟接收浏览器请求，进行处理，返回页面相关。
- **admin.py**.文件跟网站的后台管理相关。
- **migrations**.文件夹保存迁移文件记录。

安装应用

在settings.py文件中设置:INSTALLED_APPS = { ‘应用名称’ }

MTV框架

M:数据模型层

- 模型类定义在**models.py**文件中，继承自models.Model类。
- 模型常用类型：
- > CharField:字符串字段，有必要参数max_length
- > DateField:日期字段，有参数（auto_now：每次保存时将字段设置为现在，auto_now_add：首次创建对象时将字段设置为现在）
- > DecimalField:固定精度的十进制数字，有必要参数（max_digits:允许最大位数，decimal_places:小数位数）
- > EmailField:检查是否是邮箱字段，类似CharField带邮箱正则表达式
- > FileField：文件上传字段，参数upload_to:设置上传目录和文件名
- > FloatField:类似python中的float实例表示的浮点数
- > ImageField:继承了FileField的方法和属性，验证上传的对象是有效的图像
- > IntegerField:范围-2147483648到的值2147483647的整数
- > SlugField:某些事物的简短标签，通常在URL中使用，类似CharField
- > URLField:继承CharField,同时验证是否是URL
- > BooleanField:True或False字段
- ```python
  from django.db import models
  class Order(models.Model):
      name = models.CharField(max_length=50, verbose_name='姓名')
      phone = models.CharField(max_length=11, verbose_name='手机', blank=True)
      city = models.CharField(max_length=100, verbose_name='城市')
      address = models.CharField(max_length=250, verbose_name='地址')
      created = models.DateTimeField(auto_now_add=True)
      updated = models.DateTimeField(auto_now=True)
      paid = models.BooleanField(default=False)
      email = models.EmailField()
  ```

字段选项：
  ```

- null：如果为True，表示允许为空，默认值是False。
- blank：如果为True，则该字段允许为空白，默认值是False。对比：null是数据库范畴的概念，blank是表单验证范畴的。
- db_column：字段的名称，如果未指定，则使用属性的名称。
- db_index：若值为True, 则在表中会为此字段创建索引，默认值是False。
- default：默认值。
- primary_key：若为True，则该字段会成为模型的主键字段，默认值是False，一般作为AutoField的选项使用。
- unique：如果为True, 这个字段在表中必须有唯一值，默认值是False。

模型迁移：

1. 生成迁移文件: python manage.py makemigrations
2. 根据第一步生成的迁移文件在数据库中创建表: python manage.py migrate

返回查询集的过滤器如下：

1. all()：返回所有数据。
2. filter()：返回满足条件的数据。
3. exclude()：返回满足条件之外的数据，相当于sql语句中where部分的not关键字。
4. order_by()：对结果进行排序。

条件运算符：

1. 字段__exact：表示等于，也可以直接用等号表示：list=模型名.objects.filter(id__exact=1)
2. 字段__contains：是否包含说明：如果要包含%无需转义，直接写即可。list = 模型名.objects.filter(title__contains='传')
3. startswith、endswith：以指定值开头或结尾list = 模型名.objects.filter(title__endswith='部')以上运算符都区分大小写，在这些运算符前加上i表示不区分大小写，如iexact、icontains、istartswith、iendswith.
4. isnull空查询list = 模型名.objects.filter(title__isnull=False)
5. in：是否包含在范围内list = 模型名.objects.filter(id__in=[1, 3, 5])
6. gt、gte、lt、lte：大于、大于等于、小于、小于等于list = 模型名.objects.filter(id__gt=3)
7. 不等于的运算符，使用exclude()过滤器list = 模型名.objects.exclude(id=3)
8. 日期查询year、month、day、week_day、hour、minute、second：对日期时间类型的属性进行运算list = 模型名.objects.filter(pub_date__year=1980)例：查询1980年1月1日后。list = BookInfo.objects.filter(pub_date__gt=date(1980, 1, 1))

后台管理:

1. 管理界面本地化打开settings.py文件，找到语言编码、时区的设置项，将内容改为如下：

LANGUAGE_CODE = 'zh-hans' #使用中国语言

TIME_ZONE = 'Asia/Shanghai' #使用中国上海时间

2. 创建管理员python manage.py createsuperuser

在浏览器中输入: http://127.0.0.1:8000/admin/进行数据库的页面访问

3. 注册模型类

```
from django.contrib import admin

# Register your models here.
from .models import Question, Comment

class QuestionAdmin(admin.ModelAdmin):
    list_display = ('question_text', 'author', '
    list_filter = ('question_text','author', 'pu
    change_list_template = "admin/change_list_fi
    search_fields = ('question_text',)
    list_per_page = 10

class CommentAdmin(admin.ModelAdmin):
    list_display = ('comment_text', 'author','qu
    list_filter = ('comment_text','author','ques
    change_list_template = "admin/change_list_fi
    search_fields = ('comment_text',)
    list_per_page = 10

admin.site.register(Question, QuestionAdmin)
admin.site.register(Comment, CommentAdmin)
```

```python
from django.contrib import admin
from .models import Order, OrderItem

# Register your models here.


class OrderItemInline(admin.TabularInline):
    model = OrderItem
    raw_id_fields = ['product']


@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ['id', 'name', 'phone', 'email',
                    'address', 'city', 'paid',
                    'created', 'updated']
    list_filter = ['paid', 'created', 'updated']
    inlines = [OrderItemInline]
```

admin.site.register(模型名)与@admin.register(模型名效果一样)

Django中模型类的方法与数据库操作对应:

1. save(): 生成insert, update语句
2. delete(): 生成delete语句
3. all(), get(): 生成select语句

T：网页模板层

内建标签:

- block :定义一个块，子模版可以覆盖
  {% block main %}
  {{ body }}
  {% endblock %}
- comment:注释
  {% comment %}{% endcomment %}
- csrf_token：用于跨站请求保护，表单提交需添加
- entends:继承父模板{% extends "base.html" %}
- if:如果判断标签
- for:循环标签
- include: 加载模板并使用当前上下文呈现它 {% include "foo/bar.html" %}

- load:自定义模板标签集
- url:返回与给定视图和可选参数匹配的绝对路径引用

```
{% url "cart:order_create" %}
```

- with:更简单的名称缓存复杂的变量
  {% with total=business.employees.count %}
  {{ total }} employee{{ total|pluralize }}
  {% endwith %}

内置过滤器：

- add:将参数添加到值,例：{{ value|add:"2" }}
- capfirst: 将值的第一个字符大写。如果第一个字符不是字母，则此过滤器无效,例：{{ value|capfirst }}
- center: 将值居中在给定宽度的字段中,例：{{ value|center:"15" }}
- cut：从给定的字符串中删除所有arg值，例：{{ value|cut:" " }}
- date：根据给定的格式格式化日期：{{ value|date:"D d M Y" }}
- default：如果value计算为False，则使用给定的默认值。否则，使用该值。{{ value|default:"nothing" }}
- dictsort：获取字典列表，并返回按参数中给定键排序的列表。
- escape：转义字符串的HTML
- first：返回列表中的第一项
- floatformat：不带参数使用时，将浮点数四舍五入到小数点后一位-但仅当要显示的是小数部分时
- get_digit：给定一个整数，返回所请求的数字，其中1是最右边的数字，2是最右边的第二个数字，依此类推
- last：返回列表中的最后一项
- length：返回值的长度。这适用于字符串和列表
- ljust：将值在给定宽度的字段中左对齐
- lower：将字符串转换为所有小写字母
- make_list：返回变成列表的值。对于字符串，它是一个字符列表。对于整数，在创建列表之前将参数强制转换为字符串
- random：从给定列表中返回一个随机项目
- rjust：在给定宽度的字段中将值右对齐
- slice：返回列表的一部分。使用与Python的列表切片相同的语法
- title：通过使单词以大写字母开头，其余字符为小写字母，将字符串转换为大写字母
- upper：将字符串转换为全部大写
- wordwrap：以指定的行长换行

V：视图层：返回处理用户的请求并返回响应

1. 创建app的url

```python
from django.urls import path
from . import views


app_name = 'cart'

urlpatterns = [
    path('add/<int:product_id>/', views.cart_add, name='cart_add'),
    path('update/<int:product_id>', views.cart_update, name='cart_update'),
    path('cart/', views.cart_detail, name='cart_detail'),
    path('list/', views.orders, name='orders'),
    path('cart_order/', views.order_create, name='order_create'),
    path('cart_remove/<int:product_id>', views.cart_remove, name='cart_remove')
]
```

2. 在**view.py**创建对应函数

```python
def cart_remove(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    cart.remove(product)
    return redirect('cart:cart_detail')
```

> render():将给定的模板与给定的上下文字典组合在一起，并以渲染的文本返回一个 HttpResponse 对象
>
> > request用于生成此响应的请求对象
> > 要使用的模板的全名或模板名称的序列。如果给定一个序列，则将使用存在的第一个模板
> > redirect():将一个 HttpResponseRedirect 返回到传递的参数的适当URL
> > 可以是对应的函数，也可以是具体地址
> > get_object_or_404()：调用get()给定的模型管理器，但引发Http404而不是模型 DoesNotExist异常
> > 必选参数:调用的数据模型

模型实例方法

1.str()：在将对象转换成字符串时会被调用。
2.save()：将模型对象保存到数据表中
3.delete()：将模型对象从数据表中删除