# 项目介绍

* 1. 项目描述简介

通过完成一个仿京东的电子商城项目，让大家掌握企业开发特点，以及解决问题的能力。

该项目会涉及以后工作中的一些非常有代表性的功能。

我们使用git作为项目的版本控制软件。

在项目中会使用很多其他的知识，比如系统架构、维护、优化等等。

* 1. 主要功能模块

用户模块,商品模块,购物车模块,订单模块

* 1. 开发环境和技术

|  |  |
| --- | --- |
| 开发环境 | Window |
| 开发工具 |  |
| 相关技术 |  |

* 1. 项目人员组成周期成本
     1. 人员组成

|  |  |  |
| --- | --- | --- |
| 职位 | 人数 | 备注 |
| 项目经理和组长 | 1 | 一般小公司由项目经理负责管理，中大型公司项目由项目经理或组长负责管理 |
| 开发人员 | 2~3 |  |
| UI设计人员 | 1 |  |
| 前端开发人员 | 1 | 专业前端不是必须的，所以前端开发和UI设计人员可以同一个人 |
| 测试人员 | 1~3 | 有些公司并未有专门的测试人员，测试人员可能由开发人员完成测试。  公司有测试部，测试部负责所有项目的测试。  项目测试由产品经理进行业务测试。 |

* + 1. 项目周期成本

|  |  |  |
| --- | --- | --- |
| 人数 | 周期 | 备注 |
| 3 | 四个月  第一个月需求设计  第二三个月编码实现  第四个月编码实现、测试 |  |

1. 创建项目

2.1. 问题

创建项目之后,项目在目录下不见了,但是能打开

重新删除建立了一次,能看见了

2.2. 渲染页面

渲染页面时,静态文件太多了,包括每个页面的图片等的加载,如何才能更加快速地添加静态资源?

2.3 创建父模板

开始时,不知道创建多少个,在早上讲了之后会创建

* 1. 用户账号注册功能模块
     1. 需求

1. 用户填写基本信息：正确的手机号，6个以上的密码，和密码相同的确认密码，填写完毕后点击获取验证码；

2. 获取验证码以后，填写验证码，点击立即注册。

* + 1. 流程
    2. 创建用户表

**class Users(**models.Model**):** gender\_choices **= (  
 (**1,**'男')**,  
 **(**2,**'女')**,  
 **(**3,**'保密')**,  
 **)** phoneNum **=** models.CharField**(**max\_length**=**11,  
 verbose\_name**='手机号'**,  
 validators**=[** RegexValidator**(r'^1[3-9]\d{9}$'**,**'手机号码格式错误')  
 ])** nickName **=** models.CharField**(**max\_length**=**16,null**=True**,  
 verbose\_name**='昵称'**,  
 validators**=[** MinLengthValidator**(**2,**'昵称至少为2个字符')  
 ])** password **=** models.CharField**(**max\_length**=**32**)** gender **=** models.SmallIntegerField**(**choices**=**gender\_choices,default**=**3**)** school **=** models.CharField**(**max\_length**=**200,null**=True**,verbose\_name**="学校名称")** home\_address **=** models.CharField**(**max\_length**=**200,null**=True**,verbose\_name**="老家地址")** detail\_address **=** models.CharField**(**max\_length**=**200,null**=True**,verbose\_name**='现住地址')** birthday **=** models.DateField**(**null**=True**,verbose\_name**='生日日期')** add\_time **=** models.DateTimeField**(**auto\_now\_add**=True**,verbose\_name**='添加时间')** change\_time **=** models.DateTimeField**(**auto\_now**=True**,verbose\_name**='更新时间')** is\_delete **=** models.BooleanField**(**default**=False**,verbose\_name**='是否删除')** head **=** models.ImageField**(**upload\_to**='head/%Y%m'**,default**='head/hlwtx.jpg'**,verbose\_name**='用户头像')  
  
 def \_\_str\_\_(**self**):  
 return** self.phoneNum  
 **class Meta:** db\_table **= 'users'**

1. 用户填写基本信息，填写完毕后数据点击获取验证码，如果手机号码格式没有问题，则向阿里云发起请求验证码。

2. 阿里云响应并返回验证码，用户填写获得的验证码后点击立即注册，数据被丢到了form验证层；

3. form验证数据是否合法，手机号码是否重复，如果密码长度不够，手机号码格式不正确或手机号码已被注册，则返回错误信息，提示用户注册失败；

4.如果用户填写的信息没有问题，则注册成功，跳转到用户登录界面。

5.手机验证还未做,先使用原始的注册方法:

(1)在用户users子应用下的models.py中建立Users模型,以建立表存储用户注册时的手机号和密码信息，验证手机号时加入正则方式验证

(2)在视图的Register类中进行操作,验证参数合法性

(3)为验证合法性,在users子应用下创建forms.py建立验证信息注册表单类模型,在模型类中验证账号和密码的合法性:

A.密码的位数及不能为空

B.手机号是否存在,手机号的位数限制,手机号不能为空

C.密码和确认密码不能相同

6.登录:

(1)在视图的Login类中进行操作,验证参数合法性

(2)在forms.py建立验证登录表单类模型,在模型类中验证登录信息合法性:

A.验证手机号不能填空,手机号的位数,手机号是否已经注册

B.验证密码位数,密码不能为空,以及手机号存在的情况下密码不正确

7.个人资料设置:

(1)最开始用的函数,决定修改为类,解决问题方便一点

(2)在类中定义get和post方法:

A.get解决用户点击进入界面,后台通过request调用用户session信息得到用户id,可以查询出用户的所有信息,然后回显在界面的表单所有信息

(3)在个人资料增加提交按钮,用户修改资料后点提交可以直接修改数据库中数据,然后回显当前界面

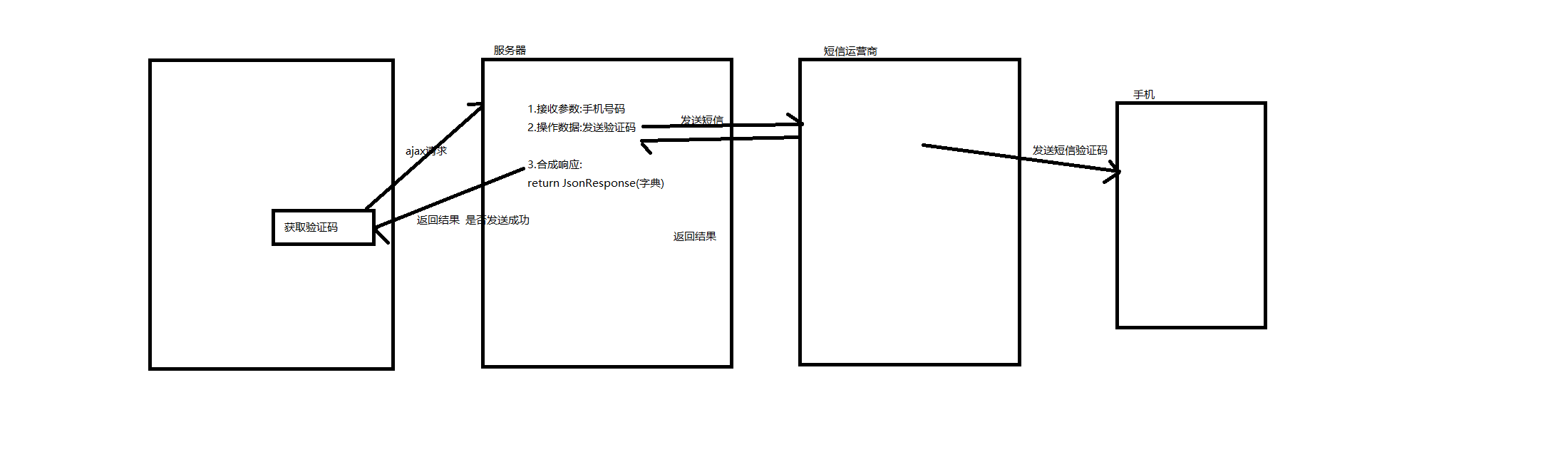
8.忘记密码:

(1)通过session获得用户ID,然后通过ID回显手机号在form表单中

(2)修改密码跟个人资料的修改差不多,只是需要在forms.py中重新建立验证类,需要验证用户是否已经修改了密码,还有密码的合法性

9.短信验证

(1)发送短信流程图



<**script type="application/javascript"**>  
 **$(function()**{  
 //1.绑定点击事件  
 **$('.yzm-hq')**.on**('click'**, **function ()** {  
 //获取手机号码,手机号码上添加一个id  
 **var** phoneNum = **$('#phoneNum')**.val**()**;  
 **var** reg = /^1[3-9]\d{9}$/;  
 // 验证手机号  
 **if (**!reg.test**(**phoneNum**))** {  
 //验证失败  
 *alert***('手机号码格式错误')  
 return**;  
 }  
 **var** self = **this**;  
 //发送ajax请求  
 **$**.**ajax(**{  
 **type**:**'post'**,  
 **url**:**'**{% **url 'users:发送验证码'** %}**'**,  
 **data**:{**'phoneNum'**:phoneNum,**'csrfmiddlewaretoken'**:**'**{{ **csrf\_token** }}**'**},  
 **dataType**:**'json'**,  
 success:**function (**data**)** {  
 **if (**data.**error** == 0**)** {  
 //1.按钮变灰  
 **$(**self**)**.css**('background-color'**,**'gray')**;  
 //2.按钮不能再点击  
 **$(**self**)**.prop**('disabled'**,**true)**;  
 //3.开始倒计时  
 **var** time = 60;  
 **var** msg = time + **'秒后重新发送'**;  
 **$(**self**)**.val**(**msg**)**;  
  
 **var** timer = window.setInterval**(function ()** {  
 time -= 1;  
 **if (**time == 0**)**{  
 //1.按钮变灰  
 **$(**self**)**.css**('background-color'**,**'#76bb2a')**;  
 //2.按钮不能再点击  
 **$(**self**)**.prop**('disabled'**,**false)**;  
 //3.返回原来的文字  
 **$(**self**)**.val**("获取验证码")**;  
 window.clearInterval**(**timer**)**;  
 **return**;  
 }  
 msg = time + **'秒后重新发送'**;  
 **$(**self**)**.val**(**msg**)**;  
 },1000**)**;  
 }**else** {  
 *alert***(**data.errMsg**)**;  
 }  
 }  
 }**)** }**)** }**)**</**script**>

1. 安全退出:

直接在该函数中清除session数据,用request.session.flush()方法

然后跳转到登录界面重新登录

1. 在后台管理中添加数据

手动添加,无技巧

1. 头像修改

涉及到头像的地址保存问题,在Users模型中创建 head=models.ImageField**(**upload\_to**='head/%Y%m'**,default**='head/hlwtx.jpg'**,verbose\_name**='用户头像')**

字段用来保存头像的地址

在Terminal中输入pip install pillow 安装上三方依赖库

在settings文件里，定义一个变量叫MEDIA\_URL。（目的是分配一个资源URL）

MEDIA\_URL = "/static/media/"

配置该URL对应的物理目录存储地址

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'static/media')

在HTML渲染中，增加上传URL前缀   
1. 模型中的ImageField实际在数据库中就是varchar结构，在渲染时，就是一个字符串   
Django提供了一个{{ MEDIA\_URL }}渲染引擎变量   
2. 根据   
[https://docs.djangoproject.com/en/1.11/ref/settings/#media-url](https://docs.djangoproject.com/en/1.11/ref/settings/" \l "media-url" \t "https://www.zybuluo.com/yqcdwg/note/_blank)   
需要配置一个渲染处理器就可以使用这个变量了。

1. TEMPLATES = [
2. {
3. ...
4. 'OPTIONS': {
5. 'context\_processors': [
6. ... ...
7. 'django.template.context\_processors.media',
8. ],
9. },
10. },
11. ]

然后就可以在后台上传图片

首页:

添加模型

商品分类表

# 商品分类表  
**class GoodsClass(**BaseModel**):** class\_name **=** models.CharField**(**max\_length**=**50,verbose\_name**='分类名')** class\_intro **=** models.CharField**(**max\_length**=**250,verbose\_name**='分类介绍'**,null**=True)  
  
 def \_\_str\_\_(**self**):  
 return** self.class\_name  
  
 **class Meta:** db\_table **= 'goods\_class'** verbose\_name **= '商品分类'** verbose\_name\_plural **=** verbose\_name

商品SPU表

**class GoodsSpu(**BaseModel**):** name **=** models.CharField**(**max\_length**=**50,verbose\_name**='商品spu名称')** desc **=** RichTextUploadingField**(**verbose\_name**='商品spu详情')  
  
 def \_\_str\_\_(**self**):  
 return** self.name  
  
 **class Meta:** db\_table **= 'goods\_spu'** verbose\_name **= '商品Spu表'** verbose\_name\_plural **=** verbose\_name

商品单位表

# 商品单位表  
**class GoodsUnit(**BaseModel**):** unitName **=** models.CharField**(**max\_length**=**20,verbose\_name**='单位名称')  
  
 def \_\_str\_\_(**self**):  
 return** self.unitName  
  
 **class Meta:** db\_table **= 'goods\_unit'** verbose\_name **= '商品单位表'** verbose\_name\_plural **=** verbose\_name

商品相册表

# 商品相册表  
**class GoodsPhotos(**BaseModel**):** image **=** models.ImageField**(**upload\_to**='goods/%Y%m'**,verbose\_name**='图片地址')** goods\_sku **=** models.ForeignKey**(**to**='GoodsSku'**,verbose\_name**='商品sku\_ID')  
  
 def \_\_str\_\_(**self**):  
 return "商品相册"  
  
 class Meta:** db\_table **= 'goods\_photos'** verbose\_name **= '商品相册表'** verbose\_name\_plural **=** verbose\_name

商品SKU表

# 商品Sku表  
**class GoodsSku(**BaseModel**):** goods\_name **=** models.CharField**(**max\_length**=**100,verbose\_name**='商品名')** goods\_intro **=** models.TextField**(**null**=True**,verbose\_name**='商品介绍')** price **=** models.DecimalField**(**decimal\_places**=**2,max\_digits**=**10,verbose\_name**='商品价格')** unit **=** models.ForeignKey**(**to**='GoodsUnit'**,verbose\_name**='商品单位')** num **=** models.PositiveIntegerField**(**verbose\_name**='库存')** sellNum **=** models.PositiveIntegerField**(**verbose\_name**='销量')** logo **=** models.ImageField**(**upload\_to**='goods/%Y%m'**,verbose\_name**='商品图片')** is\_putaway **=** models.BooleanField**(**default**=False**,verbose\_name**='是否上架')** goods\_cate **=** models.ForeignKey**(**to**='GoodsClass'**,verbose\_name**='商品分类')** goods\_spu **=** models.ForeignKey**(**to**='GoodsSpu'**,verbose\_name**='商品spu')  
  
 def \_\_str\_\_(**self**):  
 return** self.goods\_name  
  
 **class Meta:** db\_table **= 'goods\_sku'** verbose\_name **= '商品sku表'** verbose\_name\_plural **=** verbose\_name

轮播表

# 轮播  
**class Carousel(**BaseModel**):** goods\_name **=** models.CharField**(**max\_length**=**150,verbose\_name**='轮播活动名称')** goodsSku\_id **=** models.ForeignKey**(**to**='GoodsSku'**,max\_length**=**50,verbose\_name**='商品id')** image **=** models.ImageField**(**upload\_to**='banner/%Y%m/%d'**,verbose\_name**='图片地址')** order **=** models.SmallIntegerField**(**verbose\_name**='排序'**,default**=**0**)  
  
  
 def \_\_str\_\_(**self**):  
 return** self.goods\_name  
  
 **class Meta:** db\_table **= 'Carousel'** verbose\_name **= '轮播表'** verbose\_name\_plural **=** verbose\_name

首页活动

**class Activity(**BaseModel**):** title **=** models.CharField**(**verbose\_name**='活动名称'**,max\_length**=**150**)** img\_url **=** models.ImageField**(**verbose\_name**='活动图片地址'**,upload\_to**='activity/%Y%m/%d')** activity\_url **=** models.URLField**(**verbose\_name**='活动的url地址'**,max\_length**=**200**)  
  
 def \_\_str\_\_(**self**):  
 return** self.title  
  
 **class Meta:** verbose\_name **= '活动管理'** verbose\_name\_plural **=** verbose\_name

特色专区

# 特色专区  
**class Activity\_zone(**BaseModel**):** title **=** models.CharField**(**verbose\_name**='活动专区名称'**,max\_length**=**150**)** title\_intro **=** models.CharField**(**verbose\_name**='活动专区简介'**,  
 max\_length**=**200,  
 null**=True**,  
 blank**=True)** order **=** models.SmallIntegerField**(**verbose\_name**='排序'**,  
 default**=**0**)** is\_putaway **=** models.BooleanField**(**verbose\_name**='是否上线'**,  
 choices**=((False**,**'下架')**,**(True**,**'上架')**,**)**,  
 default**=**0**)** goods\_sku **=** models.ManyToManyField**(**to**='GoodsSku'**,verbose\_name**="商品")  
  
 def \_\_str\_\_(**self**):  
 return** self.title  
  
 **class Meta:** verbose\_name **= '活动专区管理'** verbose\_name\_plural **=** verbose\_name

1. 轮播图

有轮播模型,后台中在轮播模型中添加图片,然后渲染出来即可

1. 定位

在高德地图中注册账号,创建应用得到key值,然后修改示例

到首页代码中修改

1. 超市-->进入商品分类

分类的渲染

先将商品分类中的所有类查询出来,合成上下文渲染页面

在html中可直接通过for in 展示出来

分类的列表

将商品的sku表全部商品查询出来,合成上下文渲染

在html中可以直接查询数据

1. 将spu的商品详情设置为富文本

1.在您的python路径中安装或添加django-ckeditor。

1. pip install django-ckeditor

2.将ckeditor添加到您的INSTALLED\_APPS设置中。

1. INSTALLED\_APPS = [
2. 'django.contrib.admin',
3. 'django.contrib.auth',
4. 'django.contrib.contenttypes',
5. 'django.contrib.sessions',
6. 'django.contrib.messages',
7. 'django.contrib.staticfiles',
8. 'sp\_user.apps.SpUserConfig', # 用户模块
9. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
10. 'ckeditor', # 添加ckeditor富文本编辑器
11. ]

3.配置静态文件路径   
看文档：[https://docs.djangoproject.com/en/dev/howto/static-files/](https://docs.djangoproject.com/en/dev/howto/static-files/" \t "https://www.zybuluo.com/yqcdwg/note/_blank)

删除之前配置的STATICFILES\_DIRS（本来上线就要删除），修改为STATIC\_ROOT

1. #STATICFILES\_DIRS = [
2. # os.path.join(BASE\_DIR, "static"),
3. #]
4. # 设置静态文件根目录 上线的时候使用
5. STATIC\_ROOT = os.path.join(BASE\_DIR, "static")

使用以下命令，将django中安装在INSTALLED\_APPS的其他应用所有的静态文件收集到STATIC\_ROOT指定目录下

1. python.exe manage.py collectstatic

这个时候 我们在 static 目录下就能看的 ckeditor 目录，里面就是ckeditor使用到的所有的静态文件（css,js,image）。

这个时候又将之前的修改改回去，现在还没有上线，还得用之前的配置。

1. STATICFILES\_DIRS = [
2. os.path.join(BASE\_DIR, "static"),
3. ]
4. # 设置静态文件根目录 上线的时候使用
5. # STATIC\_ROOT = os.path.join(BASE\_DIR, "static")

4.设置ckeditor的上传目录   
这个目录是相对目录，相对与 MEDIA\_ROOT

1. CKEDITOR\_UPLOAD\_PATH = "uploads/"

将ckeditor\_uploader添加到您的INSTALLED\_APPS设置中。

1. INSTALLED\_APPS = [
2. 'django.contrib.admin',
3. 'django.contrib.auth',
4. 'django.contrib.contenttypes',
5. 'django.contrib.sessions',
6. 'django.contrib.messages',
7. 'django.contrib.staticfiles',
8. 'sp\_user.apps.SpUserConfig', # 用户模块
9. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
10. 'ckeditor', # 添加ckeditor富文本编辑器
11. 'ckeditor\_uploader', # 添加ckeditor富文本编辑器文件上传部件
12. ]

5.在主路由中配置 ckeditor 上传文件使用到的url地址   
ckeditor上传文件内部机制使用 ajax 异步上传文件，需要配置上传访问地址才能被访问到。

1. urlpatterns = [
2. url(r'^admin/', admin.site.urls),
3. # 上传部件自动调用的上传地址
4. url(r'^ckeditor/', include("ckeditor\_uploader.urls")),
5. # 用户模块
6. url(r'^user/', include("sp\_user.urls", namespace="sp\_user")),
7. # 其他模块
8. ]

6.安装pillow(已经安装的不用重复安装)   
ckeditor上传文件使用pillow

7.设置ckeditor在后台显示的样式（显示效果）   
使用默认配置就行，具体看文档修改

1. # 编辑器样式配置
2. CKEDITOR\_CONFIGS = {
3. 'default': {
4. 'toolbar': 'full',
5. },
6. }

8.将商品SPU表的详情字段修改RichTextUploadingField，使用ckeditor为们单独提供的字段类型，修改完后，后台自动显示富文本编辑器样式，可以上传图片

1. # 导入ckeditor上富文本编辑器自带字段
2. from ckeditor\_uploader.fields import RichTextUploadingField
3. class GoodsSPU(BaseModel):
4. """
5. 商品SPU表
6. """
7. spu\_name = models.CharField(verbose\_name='商品SPU名称',
8. max\_length=20,
9. )
10. # 使用ckeditor为我们提供的字段，不用重新迁移就可以
11. content = RichTextUploadingField(verbose\_name="商品详情")
12. def \_\_str\_\_(self):
13. return self.spu\_name
14. class Meta:
15. verbose\_name = "商品SPU"
16. verbose\_name\_plural = verbose\_name

9.前台模板显示   
{{content|safe}}

问题及难点:

1.短信验证的难点是如何完成ajax的传参设计

2.分类的current选中问题

3.写的过程中,经常出现少写某些参数的问题,导致莫名其妙的报错

总结应该是自己写代码过少的问题

1. 建立商品模型多表时的外键的确立
2. 连表查询的技巧问题,可以省掉很多的时间

技巧: 1.前期未建基础模型,发现每次都要多写三个字段,后来在根目录下创建了db包,里面创建了基础模型BaseModel,后面的所有模型都继承自BaseModel

def show\_img(self):  
 return "<img src='%s%s'>"%(settings.MEDIA\_URL, self.img)  
show\_img.allow\_tags = True  
show\_img.short\_description = "商品图片"

2. 如何在添加详细说明的时候能不仅仅只添加文字信息，还可以进行排版以及添加图片等等信息；

使用一个富文本插件：ckeditor后端编辑器

Ckeditor是django的一个插件，配置好该插件后可以使用RichTextUploadingField属性，该属性可以对上传的信息进行编辑，并且能上传图片、排版、修改文字颜色等；

使用方法：

* + - 1. 1.在您的python路径中安装或添加django-ckeditor。
      2. pip install django-ckeditor
      3. 2.将ckeditor添加到您的INSTALLED\_APPS设置中。
      4. INSTALLED\_APPS = [
      5. 'django.contrib.admin',
      6. 'django.contrib.auth',
      7. 'django.contrib.contenttypes',
      8. 'django.contrib.sessions',
      9. 'django.contrib.messages',
      10. 'django.contrib.staticfiles',
      11. 'sp\_user.apps.SpUserConfig', # 用户模块
      12. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
      13. 'ckeditor', # 添加ckeditor富文本编辑器
      14. ]
      15. 3.配置静态文件路径
      16. 看文档：https://docs.djangoproject.com/en/dev/howto/static-files/
      17. 删除之前配置的STATICFILES\_DIRS（本来上线就要删除），修改为STATIC\_ROOT
      18. #STATICFILES\_DIRS = [
      19. # os.path.join(BASE\_DIR, "static"),
      20. #]
      21. # 设置静态文件根目录 上线的时候使用
      22. STATIC\_ROOT = os.path.join(BASE\_DIR, "static")
      23. 使用以下命令，将django中安装在INSTALLED\_APPS的其他应用所有的静态文件收集到STATIC\_ROOT指定目录下
      24. python.exe manage.py collectstatic
      25. 这个时候 我们在 static 目录下就能看的 ckeditor 目录，里面就是ckeditor使用到的所有的静态文件（css,js,image）。
      26. 这个时候又将之前的修改改回去，现在还没有上线，还得用之前的配置。
      27. STATICFILES\_DIRS = [
      28. os.path.join(BASE\_DIR, "static"),
      29. ]
      30. # 设置静态文件根目录 上线的时候使用
      31. # STATIC\_ROOT = os.path.join(BASE\_DIR, "static")
      32. 4.设置ckeditor的上传目录
      33. 这个目录是相对目录，相对与 MEDIA\_ROOT
      34. CKEDITOR\_UPLOAD\_PATH = "uploads/"
      35. 将ckeditor\_uploader添加到您的INSTALLED\_APPS设置中。
      36. INSTALLED\_APPS = [
      37. 'django.contrib.admin',
      38. 'django.contrib.auth',
      39. 'django.contrib.contenttypes',
      40. 'django.contrib.sessions',
      41. 'django.contrib.messages',
      42. 'django.contrib.staticfiles',
      43. 'sp\_user.apps.SpUserConfig', # 用户模块
      44. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
      45. 'ckeditor', # 添加ckeditor富文本编辑器
      46. 'ckeditor\_uploader', # 添加ckeditor富文本编辑器文件上传部件
      47. ]
      48. 5.在主路由中配置 ckeditor 上传文件使用到的url地址
      49. ckeditor上传文件内部机制使用 ajax 异步上传文件，需要配置上传访问地址才能被访问到。
      50. urlpatterns = [
      51. url(r'^admin/', admin.site.urls),
      52. # 上传部件自动调用的上传地址
      53. url(r'^ckeditor/', include("ckeditor\_uploader.urls")),
      54. # 用户模块
      55. url(r'^user/', include("sp\_user.urls", namespace="sp\_user")),
      56. # 其他模块
      57. ]
      58. 6.安装pillow(已经安装的不用重复安装)
      59. ckeditor上传文件使用pillow
      60. 7.设置ckeditor在后台显示的样式（显示效果）
      61. 使用默认配置就行，具体看文档修改
      62. # 编辑器样式配置
      63. CKEDITOR\_CONFIGS = {
      64. 'default': {
      65. 'toolbar': 'full',
      66. },
      67. }
      68. 8.将商品SPU表的详情字段修改RichTextUploadingField，使用ckeditor为们单独提供的字段类型，修改完后，后台自动显示富文本编辑器样式，可以上传图片
      69. # 导入ckeditor上富文本编辑器自带字段
      70. from ckeditor\_uploader.fields import RichTextUploadingField
      71. class GoodsSPU(BaseModel):
      72. """
      73. 商品SPU表
      74. """
      75. spu\_name = models.CharField(verbose\_name='商品SPU名称',
      76. max\_length=20,
      77. )
      78. # 使用ckeditor为我们提供的字段，不用重新迁移就可以
      79. content = RichTextUploadingField(verbose\_name="商品详情")
      80. def \_\_str\_\_(self):
      81. return self.spu\_name
      82. class Meta:
      83. verbose\_name = "商品SPU"
      84. verbose\_name\_plural = verbose\_name
      85. 9.前台模板显示

{{content|safe}}

* 1. 首页模块
     1. 需求
        1. 渲染首页的banner部分；
        2. 渲染首页不规则展示推荐信息；
        3. 查找出商品活动信息，渲染出活动以及该活动下的商品信息。
     2. 流程
        1. 渲染不规则推荐信息部分

查询出不规则展示信息的，将所有的信息以字典形式传递到主页，在主页上渲染出所有不规则信息。

* + - 1. 渲染banner部分

查询出所有banner信息，将banner信息传递到首页上，首页遍历出所有的banner图片，渲染到首页的banner框内；

* + - 1. 渲染首页的活动信息

在视图函数内查询出所有的活动表；

遍历表内的信息，得到所有的活动；

通过活动信息，利用外键查询，得到所有的商品信息；

遍历商品信息，得到所有的商品。

* + 1. 设计要点（数据库和页面交互）
       1. 数据库

首页轮播商品表：DisplayGoods

Name 名称 char

Skuinfo 商品sku信息 foreignkey

Pic 图片 image

Goods\_order 排序 smallint

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

商品活动表：ActivateModel

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

Name 名称 char

Img 商品图片 Image

url 商品地址 URL

首页专区活动和商品：ActivateGoodsModel

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

Area 首页活动 foreignkey

Goods 商品 foreignkey

* + - 1. 页面交互

无

* + 1. 要点难点及解决方案

难点1：

查询活动以及活动下的商品信息

解决方案：

先查询出所有活动信息，遍历活动信息，利用，每一个活动来查询出活动模块下的所有商品信息，遍历这些信息，利用遍历得到的信息渲染出活动模块。

* 1. 商品分类模块
     1. 需求
        1. 得到所有的分类信息；
        2. 遍历分类信息，将所有分类显示到页面上；
        3. 通过分类查询出该分类下的商品；
        4. 将查询到的商品信息渲染到对应的分类目录下。
     2. 流程
        1. 查询出所有的分类信息；
        2. 在静态页面中遍历并渲染分类信息；
        3. 利用分类信息查询到该目录下的所有商品；
        4. 遍历出所有商品，将商品渲染到相应的分类下。
     3. 设计要点（数据库和页面交互）
        1. 数据库

商品分类表：GoodsCategoriesModel

Classname 分类名 varchar

Classintro 分类简介 varchar

Is\_delete 是否删除 Boolean

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

商品单位表: GoodsUnitModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Unitname 单位名称 varchar

商品SPU: SpuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Name 商品SPU名称 varchar

Desc 商品spu描述 RichTextUploadingField

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

2. 页面交互

如果分类下有商品信息，则显示出商品的信息，如果不存在商品，显示出没有商品的提示信息。

* + 1. 要点难点及解决方案
       1. 难点1：关于参数问题

解决方案：

为了方便排序，url地址都使用两个参数，一个是商品的分类的id信息，一个是排序的序号信息；

难点2： 点击每个分类时，如何查询渲染才能展示出当前分类下的商品;

解决方案：

查询出分类下所有商品信息，用商品的主键pk来区分不同的商品信息。

* 1. 商品排序模块
     1. 需求
        1. 进入超市页面,点击不同方式的排序按钮能够使商品信息按照相应的规律进行排序;;
        2. 点击综合排序按照综合信息进行排序;
        3. 点击销量,商品以销量进行降序排序;
        4. 点击价格:

价格按升序排序

按价格降序排序

* + - 1. 点击新品,按照商品发布时间进行升序排序。
    1. 流程
       1. 需要实现的效果：当点击某一项方式进行排序时，所有的商品按照该方式进行排序；
       2. 视图函数中传递两个参数，用来方便排序以及显示分类下的商品信息；第一个参数是分类的id,第二个参数是排序方式的id.
       3. 先用第一个参数作为主键来查询商品分类，通过分类得到分类下的所有商品；
       4. 设置一个排序列表，列表中的字段是用于排序的字段（来自model中）；
       5. 商品排序，将排序列表作为参数传递给排序方法，将第二个参数order传递给作为参数传递给排序列表。
       6. 在前端渲染页面上，将排序的参数传递到相应的位置，进行排序。
    2. 设计要点（数据库和页面交互）
       1. 数据库

商品分类表：GoodsCategoriesModel

Classname 分类名 varchar

Classintro 分类简介 varchar

Is\_delete 是否删除 Boolean

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

* + 1. 要点难点及解决方案
       1. 难点1： 对价格进行排序

价格既可以升序排列，又可以降序排列，两个排序方式都需要提供给用户，如何才能实现在用户点击升序或者降序排列的按钮时能实现相应的排序。

解决方案：

利用ajax进行判断；默认的排序方式是升序，为排序添加一个类，用该类来作为判断标识；当再次点击时，判断排序的序号为降序或升序，如果为降序，则将他的排序方式修改为升序，反之，则修改为升序。

* 1. 商品详情展示模块
     1. 需求
        1. 点击某个商品后，会跳转到该商品的详情页面；
        2. 该详情页面会渲染出商品的基本信息，商品图片，商品名称，价格，单位，详情。
     2. 流程
        1. 实现效果：详情页能够轮播该商品的图片信息；渲染出该商品的基本信息。
        2. 视图函数有一个参数gid, 使用gid查询出商品信息，再利用商品信息查询出商品的轮播图片。
     3. 设计要点（数据库和页面交互）
        1. 数据库

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

商品相册表：Goods\_Photography

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Img 图片 imagefield

Goods 商品id foreignkey

* + 1. 要点难点及解决方案
       1. 无

1. 购物车功能模块
   1. 购物车商品添加
      1. 需求

在商品详情页面，点击商品下的+号，选择加入购物车的数量， 点击加入购物车，将该数量的商品加入购物车。

在商品分两类页面，点击商品信息下的+号，将商品加入购物车，点击一次加一个商品。

* + 1. 流程

1. 前端通过ajax post请求方式，添加sku\_id（sku商品id）和count（商品的数量）  
3. 验证是否登录，如果没有登录，高数ajax中的js代码实现跳转到登录页面location.href=url  
4. 验证数据的合法性，必须都为整数  
5. 验证山沟是否存在  
6. 验证库存是否足够  
7. 保存到redis

* + 1. 设计要点（数据库和页面交互）
       1. 数据库设计：

使用hash对象 保存

hset key 属性 值

hset cart\_user\_id 商品id 数量

key: user\_telephone

filed: sku\_id

value sku\_count

* + - 1. 页面交互

用ajax进行数据交互以及页面跳转。

* + 1. 要点难点及解决方案

难点1：用Ajax实现购物车数据的添加；

解决方案：查文档

难点2：用redis数据库保存购物车内的信息；

解决方案：使用redis的对象方法进行保存，该方法有三个参数，正好解决数据的存储问题，第一个值存储用户的唯一信息，电话或者id; 第二个值保存商品的sku信息，作为商品的唯一标识； 第三个值作为商品的数量。

* 1. 购物车商品删除
     1. 需求

在商品详情页面，添加商品到购物车的时候，可以点击-减去刚刚添加到购物车内的商品；

在购物车页面，点击商品下的-号可以删除购物车内的该商品信息，点击一次删除一个商品。

* + 1. 流程
       1. 前端通过ajax post请求方式，添加sku\_id（sku商品id）和count（商品的数量）  
          3. 验证是否登录，如果没有登录，高数ajax中的js代码实现跳转到登录页面location.href=url  
          4. 验证请求删除的数据是否存在；

5. 当商品存在时，判断商品的数量是否大于1；

6. 如果商品数量大于1时，删除数据。

* + 1. 设计要点（数据库和页面交互）
       1. 数据库设计：

使用hash对象 保存

hset key 属性 值

hset cart\_user\_id 商品id 数量

key: user\_telephone

filed: sku\_id

value sku\_count

* + - 1. 页面交互

用ajax进行数据交互以及页面跳转。

* + 1. 要点难点及解决方案

难点1：用Ajax实现购物车数据的添加；

解决方案：查文档

难点2：用redis数据库保存购物车内的信息；

解决方案：使用redis的对象方法进行保存，该方法有三个参数，正好解决数据的存储问题，第一个值存储用户的唯一信息，电话或者id; 第二个值保存商品的sku信息，作为商品的唯一标识； 第三个值作为商品的数量。

* 1. 购物车商品显示
     1. 需求

1.将商品加入购物车；

2. 进入购物车页面，显示出加入购物车的商品；

* + 1. 流程
       1. 从数据库中取出sku\_id, count；
       2. 从数据库取出完整的商品信息；
       3. 计算总金额和总商品数量；  
          4. 在购物车页面渲染出商品的信息， 总价， 总数量。
    2. 设计要点（数据库和页面交互）
       1. 数据库设计：

使用redis的对象进行保存数据

key: user\_telephone

filed: sku\_id

value sku\_count

* + - 1. 页面交互

用ajax进行数据交互以及页面跳转。

* + 1. 要点难点及解决方案

难点1：勾选上全选，所有的商品信息被选中，显示所有商品的价格的综合；

解决方案：利用ajax判断商品是否全选，如果全选就调用视图函数中计算得到的总金额。

难点2：选中不同的商品，如何精确的计算出价格；

解决方案：在商品列表中，添加一个类来监测被选中的商品信息，用ajax得到商品的选中状态，用find方法查找出选中状态商品的价格，将价格进行累加，将结果用toFixed过滤一下得到商品价格精确结果。

1. 订单功能模块
   1. 确认订单
      1. 需求
         1. 在购物车页面，选择好要购买的商品后，点击去结算按钮，进入到确认订单页面；
         2. 在订单页面内显示出用户在购物车内选择好的商品；
         3. 用户选择或添加地址；
         4. 点击确认订单，进入下单页面。
      2. 流程
         1. 登录状态的用户选择要结算的商品点击提交订单；
         2. 在redis数据库中查询出从购物车提交过来的商品id和商品数量；
         3. 根据商品的数量和价格得到总价，在加上用户选择的运输方式的价格得到应付总金额；
         4. 选择或添加地址信息，当选择某个地址为默认地址时，之前的默认地址会被修改为普通地址，默认地址被渲染到页面上。
      3. 设计要点（数据库和页面交互）
         1. 数据库设计：

Redis数据库：使用hash对象 保存

hset key 属性 值

hset cart\_user\_id 商品id 数量

key: user\_telephone

filed: sku\_id

value sku\_count

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

运输方式表：AdressModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

User 所属用户 foreignkey

Name 收货人姓名 char

Telephone 收货人电话 char

Hcity 省份 char

Hproper 市区 char

Harea 地区 char

Brief 详细地址 char

Isdefault 是否默认 Boolean

订单信息表：OrderInfoModel

Order\_number 订单编号

order\_number 订单编号 char

total\_money 总金额 decimal  
user 所属用户 foreignkey  
username 收货人 char  
telephone 收货人人电话 char   
address 订单地址 char  
status 状态 choice  
transport 运输方式 foreignkey  
transport\_price 运费 decimal  
payway 付款方式 char  
paymoney 实付金额 decimal  
description 备注说明 char

* + - 1. 页面交互

将配送方式以下拉框形式显示，便于选择。

* + 1. 要点难点及解决方案

难点1：服务器高并发库存问题

解决方案：使用select\_for\_update()乐观锁，当出现库存问题时，订单提交失败，事务回滚。

* 1. 提交订单
     1. 需求
        1. 提交订单后，产生订单编号；
        2. 使用订单编号查询出订单信息；
        3. 查询出订单内的商品信息；
        4. 将成功创建的订单显示出来。
     2. 流程
        1. 登录状态的用户在确认订单页面点击提交订单按钮；
        2. 创建订单成功，订单编号以参数的形式传递到url内，使用ajax进行页面跳转；
        3. 将创建好的订单内的商品和价格信息渲染出来。
     3. 设计要点（数据库和页面交互）
        1. 数据库设计：

Redis数据库：使用hash对象 保存

hset key 属性 值

hset cart\_user\_id 商品id 数量

key: user\_telephone

filed: sku\_id

value sku\_count

用户表：UserModel

Id 主键

Create\_time 创建时间 datetime（用户注册的时间）

Update\_time 更新时间 datetime （用户修改信息的时间）

Is\_delete 是否删除 boolean （假删除）

Nickname 用户名 varchar

Telephone 手机号 varchar

Password 密码 varchar (哈希加密)

Gender 性别 choice (男， 女， 保密)

Birthday 生日 datetime

School 学校 varchar

Location 地址 varchar

Hometown 地址 varchar

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

运输方式表：AdressModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

User 所属用户 foreignkey

Name 收货人姓名 char

Telephone 收货人电话 char

Hcity 省份 char

Hproper 市区 char

Harea 地区 char

Brief 详细地址 char

Isdefault 是否默认 Boolean

订单信息表：OrderInfoModel

order\_number 订单编号 char

total\_money 总金额 decimal  
user 所属用户 foreignkey  
username 收货人 char  
telephone 收货人人电话 char   
address 订单地址 char  
status 状态 choice  
transport 运输方式 foreignkey  
transport\_price 运费 decimal  
payway 付款方式 char  
paymoney 实付金额 decimal  
description 备注说明 char

商品订单表：OrderGoods

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Orderinfo 订单信息 foreignkey

Skuinfo 商品信息 foreignkey

Price 价格 decimal

Count 商品数量 int

* + - 1. 页面交互

确认订单页面的信息不允许修改。

* + 1. 要点难点及解决方案

难点1：服务器高并发库存问题

解决方案：使用select\_for\_update()悲观锁，当出现库存问题时，订单提交失败，事务回滚。

难点2：当出现错误时，订单也会创建成功

解决方案：使用事务机制，当出现错误时进行事务回滚，当创建出现错误时，不会再创建出订单。

1. 常见面试问题

① http请求的执行流程：

在接受一个Http请求之前的准备,启动一个支持WSGI网关协议的服务器监听端口等待外界的Http请求，比如Django自带的开发者服务器或者uWSGI服务器。

服务器根据WSGI协议指定相应的Handler来处理Http请求，并且初始化该Handler，在Django框架中由框架自身负责实现这一个Handler。 此时服务器已处于监听状态，可以接受外界的Http请求 当一个http请求到达服务器的时候

服务器根据WSGI协议从Http请求中提取出必要的参数组成一个字典（environ）并传入Handler中进行处理。

在Handler中对已经符合WSGI协议标准规定的http请求进行分析，比如加载Django提供的中间件，路由分配，调用路由匹配的视图等。 返回一个可以被浏览器解析的符合Http协议的HttpResponse。

② 查询集返回的列表过滤器：

all()：返回所有数据

filter()：返回满足条件的数据

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

der\_by()：排序

③ WSGI / uwsgi/ uWSGI区分

orWSGI：WSGI的全称是Web Server Gateway Interface（Web服务器网关接口），它不是服务器、python模块、框架、API或者任何软件，只是一种描述web服务器（如nginx，uWSGI等服务器）如何与web应用程序（如用Django、Flask框架写的程序）通信的规范。

server和application的规范在PEP3333中有具体描述，要实现WSGI协议，必须同时实现web server和web application，当前运行在WSGI协议之上的web框架有Bottle, Flask, Django。

uWSGI：uWSGI是一个全功能的HTTP服务器，实现了WSGI协议、uwsgi协议、http协议等。它要做的就是把HTTP协议转化成语言支持的网络协议。比如把HTTP协议转化成WSGI协议，让Python可以直接使用。

uwsgi：与WSGI一样，是uWSGI服务器的独占通信协议，用于定义传输信息的类型(type of information)。每一个uwsgi packet前4byte为传输信息类型的描述，与WSGI协议是两种东西，据说该协议是fcgi【FCGI：fast common gateway interface 快速通用网关接口协议的10倍快。

④ blank和null

blank:设置为True时，字段可以为空。设置为False时，字段是必须填写的。字符型字段CharField和TextField是用空字符串来存储空值的。如果为True，字段允许为空，默认不允许。

null:设置为True时，django用Null来存储空值。日期型、时间型和数字型字段不接受空字符串。所以设置IntegerField，DateTimeField型字段可以为空时，需要将blank，null均设为True。

如果为True，空值将会被存储为NULL，默认为False。如果想设置BooleanField为空时可以选用NullBooleanField型字段。

null 是针对数据库而言，如果 null=True, 表示数据库的该字段可以为空。NULL represents non-existent data.

blank 是针对表单的，如果 blank=True，表示你的表单填写该字段的时候可以不填。比如 admin 界面下增加 model 一条记录的时候。直观的看到就是该字段不是粗体

1. 技术点
   * + 1. 阿里短信验证
       2. ckeditor富文本编辑器使用
       3. session保存到redis缓存中,缓解数据库压力
       4. 页面数据的缓存（缓解数据库压力）
       5. 搜索（ whoosh 索引 分词）
       6. 购物车存放 redis 哈希，解决频繁数据库操作问题
       7. 前端使用ajax请求后端接口(完成购物车添加,减少,删除)
       8. 下单的时候使用事务，保证数据的安全性
       9. 高并发的库存问题的解决 （悲观锁）(select\_for\_update())
       10. 父模板的提取
       11. 图片上传到七牛云
       12. 头像上传 使用webuploader