# 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
  2. 开发环境和技术

|  |  |
| --- | --- |
| 开发环境 | Window |
| 开发工具 | Django |
| 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. 用户模块
   1. 用户登录注册
      1. 功能需求

1.登录:

1.用户名为手机号,验证手机号格式是否正确和注册,格式不正确提示不是正确手机号,格式正确验证是否注册,如果未注册提示请注册,如果已注册验证密码是否正确,

2.如果正确跳转到个人中心,如果错误提示账号或密码错误

2.注册:

验证字段:手机号,密码,重复密码,验证码

手机号: 1必填,不填提示错误;

2.验证格式是否正确,不正确提示不是有效手机号

密码: 1必填,不填提示错误;

2.验证格式,格式不正确提示错误

重复密码: 1.必填,不填提示错误;

2.验证格式,格式不正确提示错误;

3.验证是否和密码填写一致,不一致提示错误

验证码:

* + 1. 流程

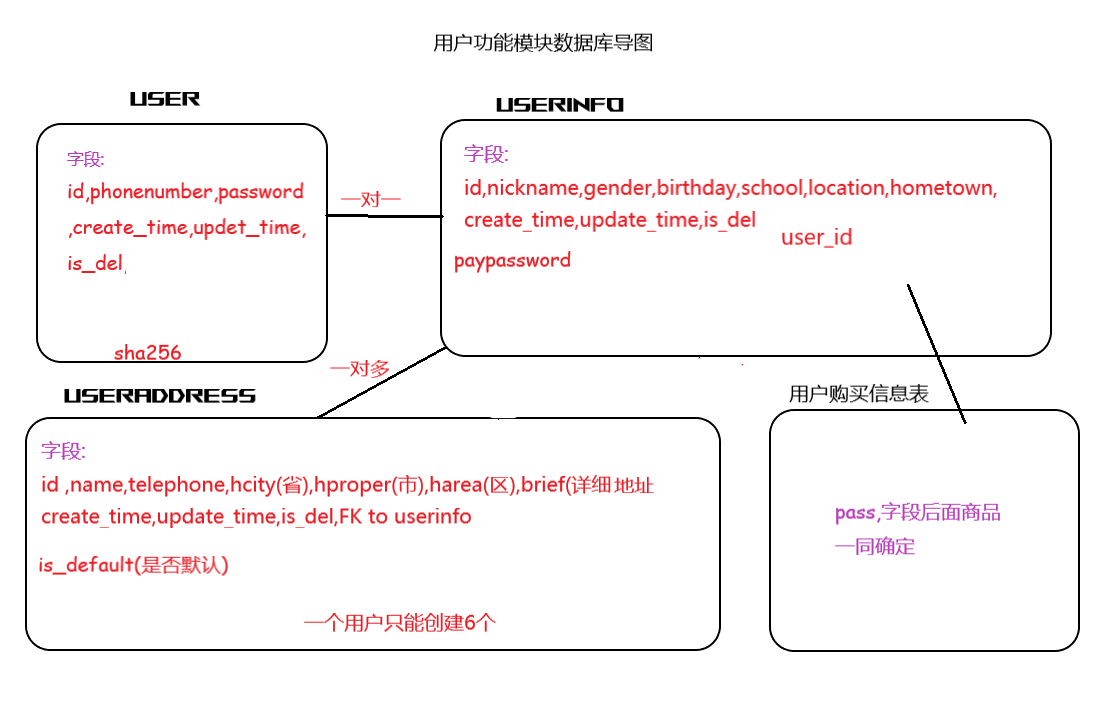
个人中心---->登录页面:

登录成功---发送session---个人中心

登录失败:

注册----注册成功----个人中心

* + 1. 设计要点（数据库和页面交互）



Sha256:字段长度设为64位,加密单独建立函数,需要用时在引用

一对一关系表,先添加基本的,所以把关键id设在userinfo表上

Userinfo 字段和表单允许为空

设计model.form,密码,重复密码,用户协议勾选需单独设立字段验证

用户地址表需要添加是否默认选择字段,默认否

优化:

1. 每个表都有相同字段,可以设置模型基类用于继承
2. 登录,修改信息都会设置session,所以可以设置一个函数减少代码
3. 很多页面都会在登录后才能显示,说以可以定义一个装饰器:

装饰器返回调用原函数时要加(),返回新函数时不加()

1. 有些功能视图会以视图类形式创建(1个页面,会有多个请求方式访问时),但原装饰器只能在函数上使用(函数第一参数为request,类第一个函数为self),所有可以通过设置一个视图基类来增强原父视图类的dispatch方法,通过继承,来减少代码量:

用@method\_decorator(check\_session)来装饰dispatch,不能直接用原函数装饰器直接装饰

* + 1. 要点难点及解决方案

要点:

确定数据库关系,和字段属性(能否为空),

多个模型(视图)使用相同字段(方法)时可以建立父模型(父视图)继承,

出现多次使用的代码可以尽量进行函数封装,

注册密码要sha256(长度64位)加密,登录时登录密码先加密后再验证

注册成功跳转到登录界面

登录成功设置session

设置session过期时间,不填为两周,0为关闭浏览器过期,固定过期时间,None永不过期  
 request.session.set\_expiry()

session保存到redis数据库减少服务器压力(安装pip install django-redis库,使用时要开始redis-server)

CACHES = {  
 "default": {  
 "BACKEND": "django\_redis.cache.RedisCache",  
 "LOCATION": "redis://127.0.0.1:6379/1",

#域名/端口号/redis数据库(0-15)  
 #有密码格式: redis://[:password]@localhost:6379/0  
 "OPTIONS": {  
 "CLIENT\_CLASS": "django\_redis.client.DefaultClient",  
 }  
 }  
 }  
session存取方式:db(数据库),cache(内存),cache\_db(不能内存就数据库)   
SESSION\_ENGINE = "django.contrib.sessions.backends.cache"  
SESSION\_CACHE\_ALIAS = "default"

难点:

验证码验证:

* 1. 用户修改密码,手机号,忘记密码
     1. 需求

忘记密码:用户通过手机号接收验证码,设置新密码.如果手机号存在则替换原密码,手机号不存在则创建新用户

修改密码:用户登录后通过个人中心进入安全设置进入修改密码,旧密码输入一次,新密码输入两次,两次必须一致

更换手机号:用户登录后通过个人中心进入安全设置进入更换手机号,通过新手机号获取验证码替换原手机号

* + 1. 流程

忘记密码:

登录---->忘记密码---->验证手机号是否注册,

未注册-->提示手机号未注册

已注册-->发送验证码验证

合法--->更改密码

不合法-->返回提示错误

登录---->忘记密码---->失败返回当前页并进行错误提示;

修改密码:

登录成功--->个人中心--->安全设置--->修改密码

-->修改成功,返回个人中心

-->修改失败,返回当前页并进行错误提示

更换手机号:

登录成功--->个人中心--->安全设置--->更换手机号

--->验证手机号是否存在,

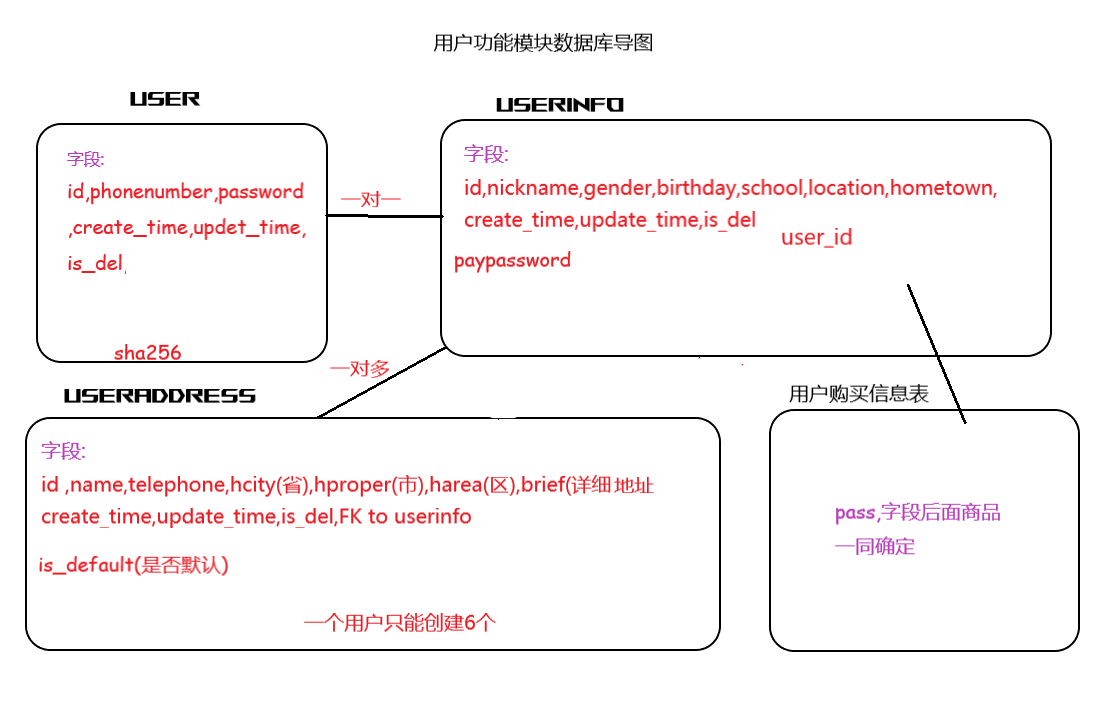
---存在,返回并提示错误信息

不存在,发送验证码验证

验证通过-->修改成功,返回个人中心

验证失败-->修改失败,返回当前页并进行错误提示

* + 1. 设计要点（数据库和页面交互）



忘记密码:

根据post请求传入的手机号进行判断,存在修改该手机号的密码,不存在添加手机号和密码

Get\_or\_create

修改密码:

根据Session传入user的主键ID查询原密码

验证原密码是否正确,正确验证新密码是否合法,不正确返回进行错误提示

合法,修改原密码,不合法返回进行错误提示

更换手机号:

根据Session传入user的主键ID查询手机号

验证新手机号是否合法

不合法返回进行错误提示

合法,验证验证码,

正确:修改手机

不正确:返回进行提示

* + 1. 要点难点及解决方案

要点:

忘记密码的form可以继承注册的form,去掉用户协议验证属性和重新修改手机号验证方法即可

修改密码时要验证输入密码是否输入正确,在form中自定义一个方法,使用session得到id验证,

使用add\_error添加错误信息

难点:

短信验证码

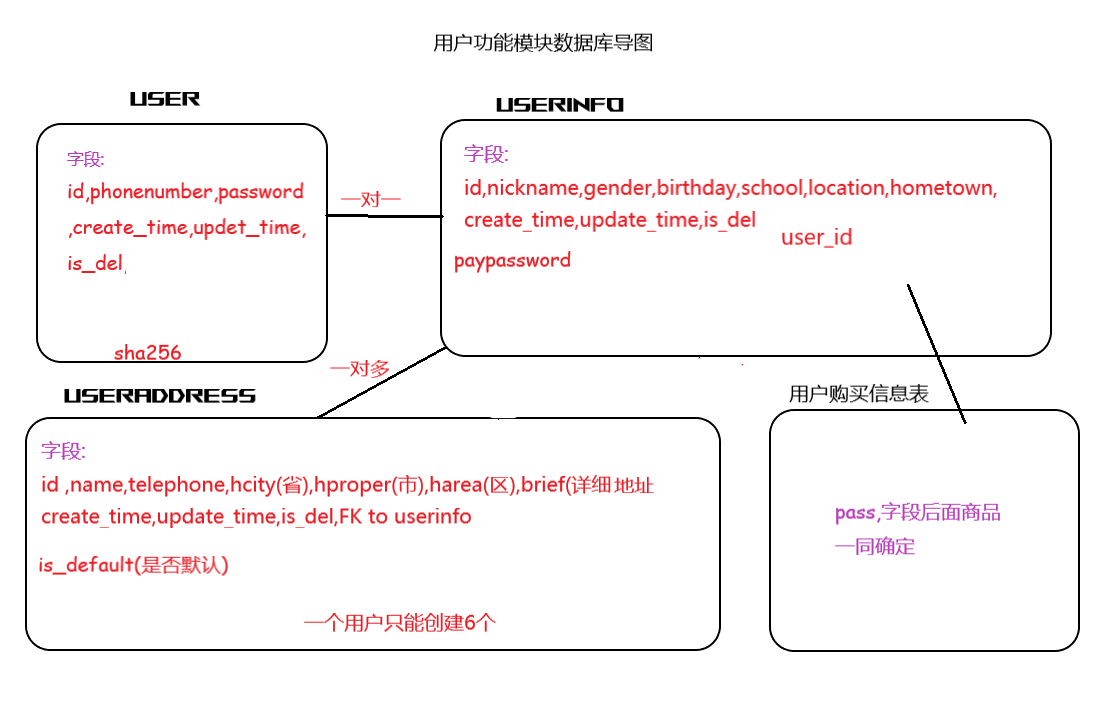
* 1. 用户个人资料
     1. 需求

展示个人资料,头像切换

* + 1. 流程

个人中心--->个人资料---保存成功返回个人中心页面

* + 1. 设计要点（数据库和页面交互）



根据关联字段添加用户信息

字段属性null=True的时候blank也要设置True

* + 1. 要点难点及解决方案

要点:

可以增加JS遮蔽电话号码中间4位数:

<script type="application/javascript">

var phone = $('#phone').val();

var newphone = "";

if (phone.length > 0) {

for (var i = 0; i < phone.length; i++) {

if (i < 3 || i >= phone.length - 4) {

newphone += phone[i];

}

else {

newphone += '\*';

}

$('#phone').val(newphone)

}

}

</script>

更换头像:

要在镜像环境中安装第三方包:pip install pillow

配置文件:https://www.zybuluo.com/yqcdwg/note/1243942

字段属性设置:

logo= models.ImageField(upload\_to='shop/%Y%m/%d',default='default/shop1.png',verbose\_name='店铺LOGO')

使用request.FILES['file']接收用户上传图片

* 1. 验证码验证
     1. 需求

在注册,忘记密码,和更换手机号时要使用短信验证码登录

* + 1. 流程

点击获取验证码---->判断手机号码填写格式

错误--->提示格式错误

正确--->发送验证码--->验证验证码是否正确

* + 1. 设计要点（数据库和页面交互）

在父模板中的预留位添加ajax请求:

需在加载完页面再执行,先利用id或class绑定点击事件,

和使用.val()方式得到电话号码进行正则验证格式是否正确,

错误alter弹出显示错误,正确进行ajax请求:

<script type="application/javascript">

{#页面加载完再执行#}

$(function () {

$('.yzm-hq').on('click', function () {

{#绑定点击事件#}

var phone\_num = $('#phone\_num').val();

var re = /^1[3-9]\d{9}$/;

{#获取手机号,然后正则验证#}

if (!re.test(phone\_num)) {

alert('手机号码格式不正确!');

return;

}

var self = this;

{#绑定获取验证码按钮#}

//发送ajax请求

$.ajax({

type: 'post',

url: '{% url "user:发送验证码" %}',

data: {'phone\_num': phone\_num, 'csrfmiddlewaretoken': '{{ csrf\_token }}'},

dataType: 'json',

success: function (data) {

//响应数据,判断是否成功

if (data.error == 0) {

//发送成功按钮变灰色

$('.yzm-hq').css("background-color",'gray');

//禁止点击

$(self).prop('disabled', true);

//显示倒计时

var time = 120;#2分钟后过期

var msg = time + '秒后重新发送';

$(self).val(msg); //更改注册按钮的值

var timer = window.setInterval(function () {

//设定定时执行方法

time -= 1;

if (time == 0) {

//倒计时结束变回原样

$('.yzm-hq').css("background-color", '#76bb2a');

$(self).prop('disabled', false);

$(self).val('获取验证码');

//清除定时方法

window.clearInterval(timer);

return;

}

msg = time + '秒后重新发送';

$(self).val(msg)

}, 1000); //毫秒

}else {

alert(data.errmsg)

}

}

})

})

});

</script>

在视图中添加发送验证码试图类,根据请求方式获取手机号,并用random.randint和列表推导式

获取随机验证码,数字需转换为字符串类型:

"".join([str(random.randint(0,9)) for \_ in range(6)])

* + 1. 要点难点及解决方案

要点:

点击获取验证码后,点击按钮要进入倒计时并禁止点击触发事件

$(self).prop('disabled', true)或$(self).attr('disabled', 'disabled')

验证码根据键值对用redis保存,减少服务器压力

并且redis自带expire方法设置键值过期时间和incr方法限制获取验证码次数

注意,在redis中根据键get到的值为二进制,数字可以强转,字符串职能decode(utf-8)

进行转码

短信验证码:

设置一个视图函数,接收ajax请求

注册云服务获取密钥并下载安装服务商第三方包

获取签名和模板code

在setting中添加密钥值

添加链接服务商接口方法:

# 完成 定义一个方法 发送短消

# 注意：不要更改

REGION = "cn-hangzhou"

PRODUCT\_NAME = "Dysmsapi"

DOMAIN = "dysmsapi.aliyuncs.com"

acs\_client = AcsClient(ACCESS\_KEY\_ID, ACCESS\_KEY\_SECRET, REGION)

region\_provider.add\_endpoint(PRODUCT\_NAME, REGION, DOMAIN)

def send\_sms(business\_id, phone\_numbers, sign\_name, template\_code, template\_param=None):

smsRequest = SendSmsRequest.SendSmsRequest()

# 申请的短信模板编码,必填

smsRequest.set\_TemplateCode(template\_code)

# 短信模板变量参数

if template\_param is not None:

smsRequest.set\_TemplateParam(template\_param)

# 设置业务请求流水号，必填。

smsRequest.set\_OutId(business\_id)

# 短信签名

smsRequest.set\_SignName(sign\_name)

# 数据提交方式

# smsRequest.set\_method(MT.POST)

# 数据提交格式

# smsRequest.set\_accept\_format(FT.JSON)

# 短信发送的号码列表，必填。

smsRequest.set\_PhoneNumbers(phone\_numbers)

# 调用短信发送接口，返回json

smsResponse = acs\_client.do\_action\_with\_exception(smsRequest)

# TODO 业务处理

return smsResponse

在发送验证码视图中根据签名调用链接方法发送短信:

\_\_business\_id = uuid.uuid1()

params ="{\"code\":\"%s\",\"product\":\"Supermarket\"}" % random\_code

# print(params)

rs = send\_sms(\_\_business\_id, phone, "注册验证", "SMS\_2245271", params)

print(rs.decode('utf-8'))

验证码验证:

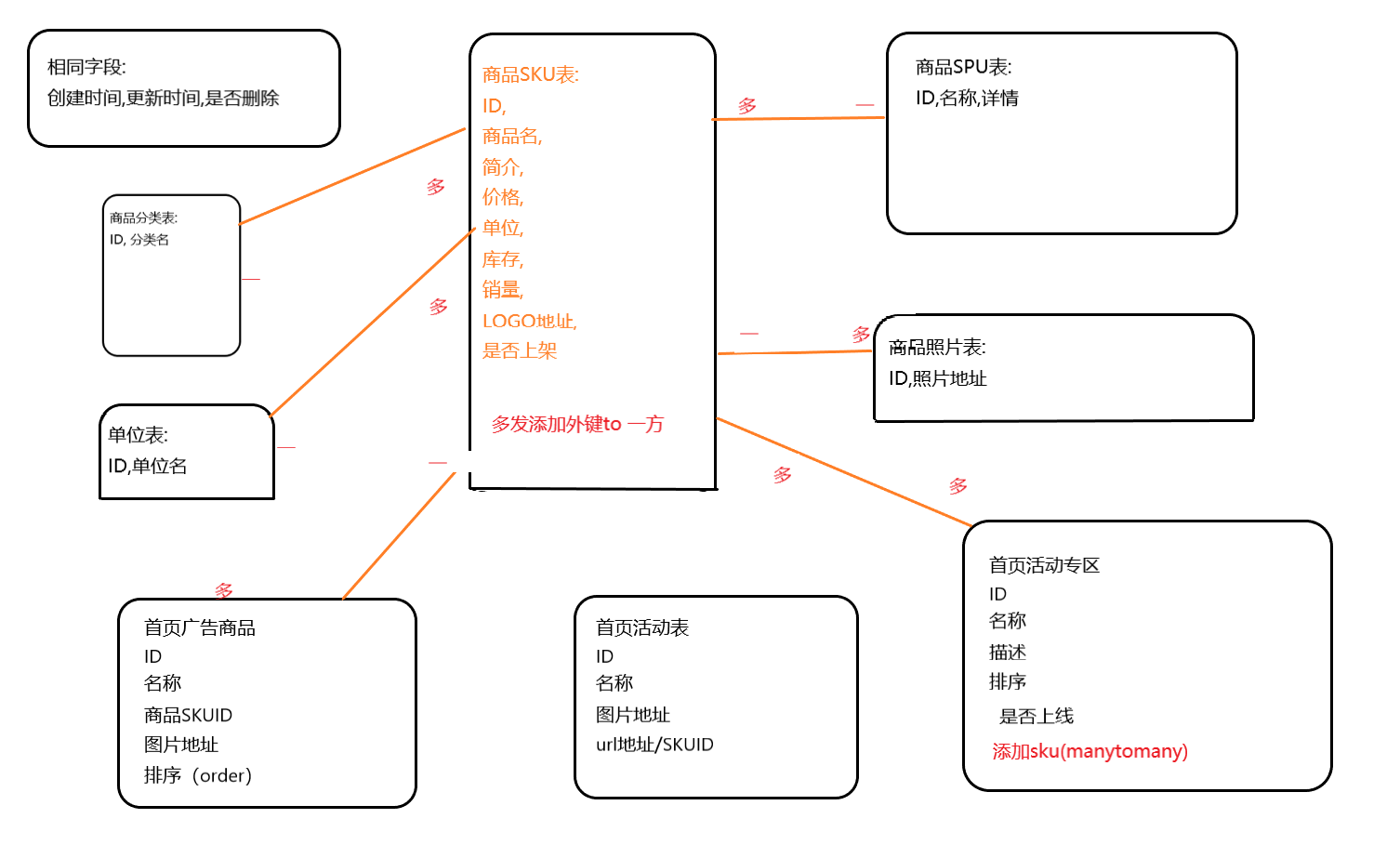
在form中进行综合验证:

1.添加captcha验证字段

2.要在验证密码一致后验证:

3.在redis中获取的值为二进制转码

1. 商品功能模块
   1. 主页、列表页、详情
      1. 需求
         1. 展示主页:城市定位，轮播，广告专区，跳转超市页面
         2. 超市:分类列表；按分类商品展示，
         3. 商品详情:展示指定商品的详情信息
      2. 流程
         1. 超市主页----->超市---->详情
      3. 设计要点（数据库和页面交互）
         1. 数据库设计导图



* + - 1. 创建模块
      2. 提出相同字段:创建时间,更新时间,是否删除
      3. 商品属性:

SPU = Standard Product Unit (标准产品单位)

SPU 是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息的集合，该集合描述 了一个产品的特性。通俗点讲，属性值、特性相同的商品就可以称为一个 SPU。

例如:iphone7 就是一个 SPU，与商家，与颜色、款式、套餐都无关。

SKU=stock keeping unit(库存量单位)

SKU 即库存进出计量的单位， 可以是以件、盒、托盘等为单位。

SKU 是物理上不可分割的最小存货单元。在使用时要根据不同业态，不同管理模式来处理。 在服装、鞋类商品中使用最多最普遍。

例如:纺织品中一个 SKU 通常表示:规格、颜色、款式。

* + 1. 要点难点及解决方案
       1. 一对多,在多方添加外键关联一方
       2. \_\_str\_\_中只能返回字符串
       3. 在class Meta中使用:

db\_table=’表名’

verbose\_name=’后台名’

verbose\_name\_plural(后台名复数)=verbose\_name

* + - 1. 创建完模块需要在admin设置添加到后台:

admin.site.register(模块名)

* + - 1. 可以在子应用的app文件里使用verbose\_name=’应用名’,汉化
      2. 查询时正确使用正向查询和你想查询:

一对多:

正向查询(模型类变为小写):少的一方模型对象.多的一方模型类\_set.all()

逆向查询:多一方模型对象.关联属性

过滤查询:模型类.objects.filter(关联模型类名\_\_字段\_\_条件助记符="值")

多对多(图片相册):

有关联字段的一方为正向!

* + - 1. 在商品详情中会有多种格式文件和文本编辑,后台没有此功能,所以要使用富文本编辑器:

1. 编辑器有多种,选好后要在镜像环境中安装编辑器中安装包:

pip install django-ckeditor

1. 安装完成后配置编辑器设置:

https://www.zybuluo.com/yqcdwg/note/1244768

* 1. 列表页排序
     1. 需求
        1. 按指定需求对分类产品进行排序，点击指定按钮时变色，其他保持不变
     2. 要点
        1. 在url在增加一个接收参数order：

url('^category/(?P<id>\d\*)\_{1}(?P<order>\d?)$',CateGoryView.as\_view(),name='商品分类'),

* + - 1. 设置传入对应排序字段:｛0:综合,1:销量,2:价格(升),3:价格(降),4:添加时间｝
      2. 将字段放进列表中,参数为其所对应的索引（-字段为降序）：

order\_rule=['pk','-sales','price','-price','-create\_time']

* + - 1. 增加一个接收参数odrer作为order\_rule索引查询排序字段
      2. 主页面跳转时url为：{% url 'goods:商品分类' '' ''%}，所以判断当接收参数

为空字符串时默认为分类中的第一个类别：

if id =='':  
 #正则中\*接收0或多个,当为0时返回空字符串  
 cate=categorys.first()  
 #默认为分类中的第一个  
 id=cate.id

* + - 1. 参数接收时为字符串，在转化为整数
      2. 根据类别和排序查询对应商品
      3. 使用if判断order值，改变颜色：

<li{% if order == 2 **or** order == 3 %} class="current" {% endif %}>  
 <a id="price" href="{% url 'goods:商品分类' id 2 %}">  
 <i class="list-price">价格</i>  
 </a>  
</li>

* + - 1. 使用JS改变价格升序降序：

$(**function** () {  
 {% if order == 2 %}  
 $('#price').attr('href', '{% url 'goods:商品分类' id 3 %}');  
 {% elif order == 3 %}  
 $('#price').attr('href', '{% url 'goods:商品分类' id 2 %}');  
 {% endif %}  
  
})

注意：要在界面加载执行$(function){}

* 1. 搜索功能
     1. 需求
        1. 在搜索框输入关键字，展示所有带有关键字并且没有下架的商品
     2. 简介：

MySQL中的索引：主键ID，唯一键，普通字段，全文搜索，但全文模糊搜索（like %%）效率低下。查询sql执行效率：Expain select \* from.....

Python搜索框架：haystack，支持大部分搜索引擎

常用搜索引擎：whoosh（python）、solr（java）、Xapian、Elasticsearc（java）

因为引擎对中文分词不全（垃圾），所以要用中文分词

免费分词：jieba

* + 1. 配置
       1. 安装第三方包：

pip install django-haystack

pip install whoosh

pip install jieba

* + - 1. 在配置文件中配置全文检索框架haystack：



注意：最好添加到子应用配置前



* + - 1. 在应用目录（goods）下，创建一个文件search\_indexes.py文件在该文件中定义一个索引类，全文检索框架根据该类生成索引：

# 导入全文检索框架索引类

from haystack import indexes

from sp\_goods.models import GoodsSKU

class GoodsSKUSearchIndex(indexes.SearchIndex, indexes.Indexable):

# 设置需要检索的主要字段内容 use\_template表示字段内容在模板中

text = indexes.CharField(document=True, use\_template=True)

# 获取检索对应对的模型

def get\_model(self):

return GoodsSKU

# 设置检索需要使用的查询集

def index\_queryset(self, using=None):

"""Used when the entire index for model is updated."""

return self.get\_model().objects.all()

* + - 1. 创建模版文件，文件中添加搜索字段

例如：templates/search/indexes/应用名/模型名小写\_text.txt

# 指定搜索的字段

{{object.sku\_name}}

{{object.brief}}

注意：关联查询正向和逆向查找方式

* + - 1. 生成索引牵引文件：python manage.py rebuild\_index
      2. 在主路由中添加：url(r'^search/', include('haystack.urls')),
      3. 创建搜索表单 ：

创建一个form表单，用于搜索，关键点在于:

-- 搜索引擎默认接收一个GET方式传递q参数作为搜索关键字

-- action 请求地址就是在主路由配置的地址 /search/

<form action="/search/" method="get">

<input type="text" name="q" value="" class="cate-input" placeholder="请输入您要的搜索的产品关键词"/>

<input type="submit" class="cate-btn" value=""/>

</form>

* + - 1. 在templates/search/目录下创建search.html模板



query是接收name=‘q’的参数（关键字），page 当前页检索结果集，可用于分页

* + - 1. whoosh自带的分词对中文不友好，使用免费的中文分词jieba ：

虚拟环境下在/Lib/site-packages/haystack/backends/下，复制 whoosh\_backend.py 改名为 whoosh\_cn\_backend.py ， 打开复制的新文件，引入中文分析类，内部采用jieba分词。

导入from jieba.analyse import ChineseAnalyzer

将analyzer=StemmingAnalyzer()改为analyzer=ChineseAnalyzer()

并将配置文件中的路径改为：

'ENGINE': 'haystack.backends.whoosh\_cn\_backend.WhooshEngine',

再重新生成索引牵引文件：python manage.py rebuild\_index（每次更换搜索字段都有生成一次）

1. 购物车功能模块
   1. 需求
      * 1. 需在商品列表页，商品详情页面，用户点击可以将该商品加入到购物车
        2. 在商品列表页面，用户每次点击+ 或者 - 都在购物车中添加或者减少一个商品数量
        3. 在商品详情页面，用户可以指定添加多少商品到购物车
        4. 通过ajax提交实现，无刷新效果，提高用户体验度
        5. 在购物车管理页面，用户可以管理购物车功能，可以选中某些商品，可以全选，全不选，可以添加商品数量和减少商品数量。当商品数量为0上删除该商品数据。
        6. 由于购物车数据用户可以频繁操作，如果保存到数据库会增加数据库的负担，造成数据库宕机，所以将购物车数据添加到redis中保存，大大减少了数据库服务器的压力。
   2. 列表页与购物车
      1. 分析：



1. 在登录后才能使用，未登录点击提示登录，并返回到当前页。所以当跳转到登录页面时应到发送一个session保存当前页面url
2. 要保存用户ID，SKU\_ID，和价格，所以使用redis的哈希保存：

Key:field value

1. 要点击+号，数量增加，浮现-号和数量，数量加1或减1并且购物车数量随着变化，刚好redis中使用hincrby实现
   * 1. 要点难点
        1. 创建一个返回json格式数据函数方法，定义错误code和错误信息，数据，方便以后调用：

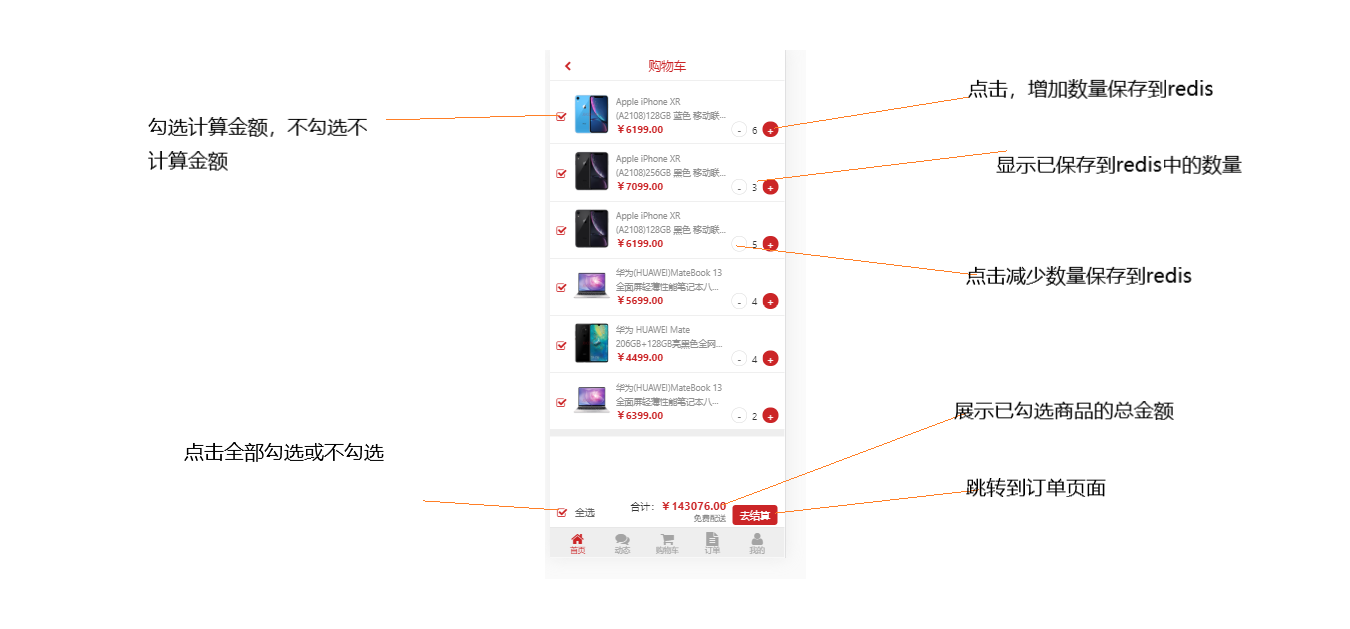
#设置Json格式数据  
def json\_msg(code,msg=None,data=None):  
 *"""* ***:param*** *code:0 为正确，其他为错误* ***:param*** *msg: 错误信息* ***:param*** *data: 提交数据  
 """* return {'code':code,'errmsg':msg,'data':data}

* + - 1. 在求情头中有个Referer含有当前url,用is\_ajax()判断是否是ajax请求，重新设置检查session装饰器：

# 检查session  
def check\_session(func):  
 def new(request, \*args, \*\*kwargs):  
 # 判断有没有session  
 if request.session.get('id') is None:  
 # 获取当前地址  
 referer = request.META.get('HTTP\_REFERER', None)  
 if referer:  
 # 如果有就添加到session中  
 request.session['referer'] = referer  
 # 判断是否是ajax请求is\_ajax()返回布尔值  
 if request.is\_ajax():  
 return JsonResponse(json\_msg(4, '未登录'))  
 else:  
 return redirect('user:登录')  
 else:  
 # 如果有session调用原函数  
 return func(request, \*args, \*\*kwargs)  
  
 # 返回新函数  
 return new

* + - 1. Request得到的参数为str，要转化为int，从redis得到的数据为二进制，整数可以直接用int（）直接转码，str要用decode（utf-8）转码
      2. 定一个获取商品总数量函数方法，方便以后调用：

#获取 当前用户购物车中的总数量  
def get\_count(request):  
  
 user\_id = request.session.get("id")  
 if user\_id is None:  
 return 0  
 else:  
 # redis  
 r = get\_redis\_connection()  
 # 准备键  
 cart\_key = f"cart\_{user\_id}"  
 # 获取字段值（多个）  
 values = r.hvals(cart\_key)  
 # 准备一个总数量  
 total\_count = 0  
 for v in values:  
 total\_count += int(v)  
 return total\_count

* + - 1. 在goods列表页中要查出用户已添加购物车商品总数量，展示到页面；在add和minus中也要查出实时总数量提交返回给ajax更新页面总数量信息展示
  1. 详情页与购物车
     1. 分析
        1. 
        2. 与列表页不同，点击加入购物车后再添加或购买（确认后添加，没有减少）
        3. 数量不能超过库存数量
        4. 点击加入购物车或购买要判断是否登录：
        5. 未登录：登录后跳转回当前页
        6. 登录：点击加入购物车后再将数据（use\_id，sku\_id，count）存到数据库
        7. 添加成功后提示添加回复信息（成功，失败）：
        8. 成功：询问是否结算，是：跳转到购物车；不是：跳转回当前页
        9. 失败：提示错误信息
     2. 要点
        1. 用confirm提示选择
        2. 用location.href=’URL’，跳转页面
  2. 购物车
     1. 分析
        1. 

1. 常见面试问题