# 项目介绍

* 1. 项目描述简介

通过完成一个仿唯品会的电商，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。

用户端采用前后端分离开发模式，前端通过调用后端API接口来获取和提交数据。掌握项目中前后端分离的开发模式。

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
     + 1. 用户模块
       2. 商品模块
       3. 订单模块
       4. 购物车模块
  2. 开发环境和技术

|  |  |
| --- | --- |
| 开发环境 | Window |
| 开发工具 | Ptcharm |
| 相关技术 | Js html python Django jQuery |

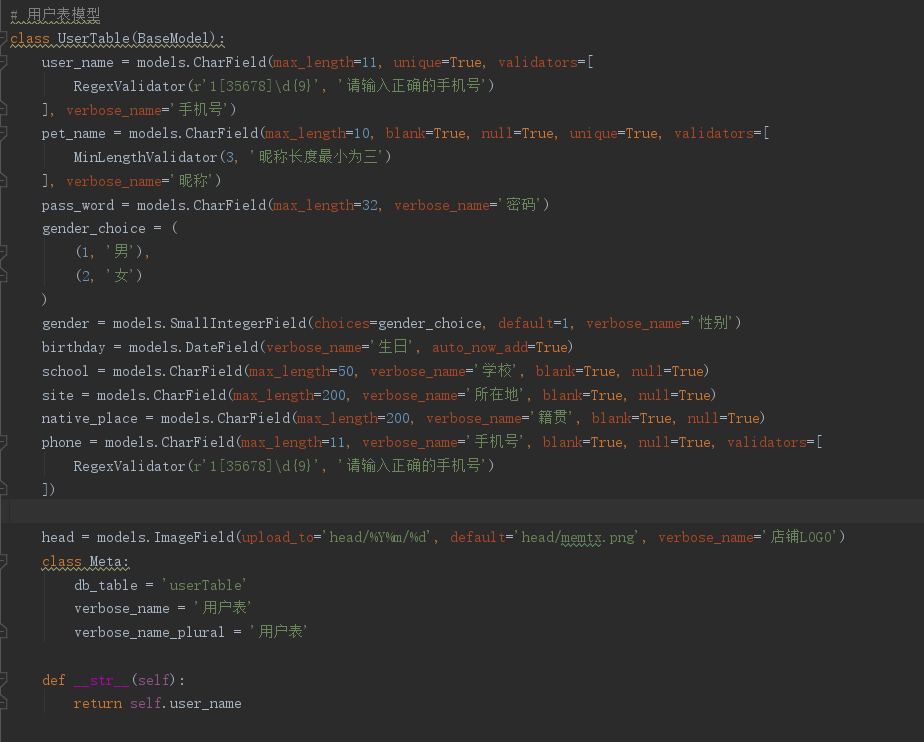
* 1. 项目人员组成周期成本
     1. 人员组成

|  |  |  |
| --- | --- | --- |
| 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. 用户功能模块
   1. 需求
      * 1. 用户根据账号和密码进行登陆
        2. 用户忘记密码时可以使用忘记密码进行找回
        3. 用户通过手机号注册账号
        4. 注册和修改密码及找回密码时需要短信验证码
        5. 用户登陆过后的个人中心
        6. 用户个人信息的修改
   2. 流程
      * 1. 分析静态网页，将其渲染出来
        2. 创建用户的数据表，将用户信息存储起来
        3. 完成注册功能
        4. 完成登陆功能
        5. 完成找回密功能
        6. 完成用户修改个人信息功能
   3. 设计要点（数据库和页面交互）
      * 1. 用户表



* + - 1. 交互设计

用户在注册及登陆时如果输入不合法就会回显不合法的原因在对应框框的下面

完成对应的操作回跳到对应的页面，对应的输入框内都有对应的文字提示

* 1. 要点难点及解决方案
     + 1. 对用户的密码进行加密

使用md5加密方式，每次等陆加密进行比对

* + - 1. 用户等路后对用户登陆状态的验证

用户登陆成功后将对应的信息保存到session中每次进去需要登陆查看的页面，先检查器session看登陆状态，没有登陆的话就跳转到登陆页面，可以将验证方法写成一个装饰器，如下

# 检查登陆状态  
def check\_login(func):  
 def verify\_login(request, \*args, \*\*kwargs):  
 if request.session.get('ID') is None:  
 return redirect('user:login')  
 else:  
 return func(request, \*args, \*\*kwargs)  
 return verify\_login

@method\_decorator方法可以实现对类中方法的装饰，可以定义一个基础视图类然后之后需要验证的类继承改基础视图类就好了，基础视图类中将装饰器妆饰在dispatch上，重写该方法就可以了

* + - 1. 将session保存到redis中减轻服务器压力

先pip install Django-redis

再在主应用中的settings中配置如下

* CACHES = {
* "default": {
* "BACKEND": "django\_redis.cache.RedisCache",
* "LOCATION": "redis://127.0.0.1:6379/1",
* "OPTIONS": {
* "CLIENT\_CLASS": "django\_redis.client.DefaultClient",
* }
* }
* }

在配session存储引擎

配置session的存储引擎

* SESSION\_ENGINE = "django.contrib.sessions.backends.cache"
* SESSION\_CACHE\_ALIAS = "default"
  + - 1. 发送短信验证码

绑定发送验证码的单击事件，单击发送ajax请求，发送请求到发送验证码的视图，设置定时器，使按钮60秒才可以点击一次，并作样式的改变，视图里生成随机验证码然后接入阿里云的短信服务即可

代码如下

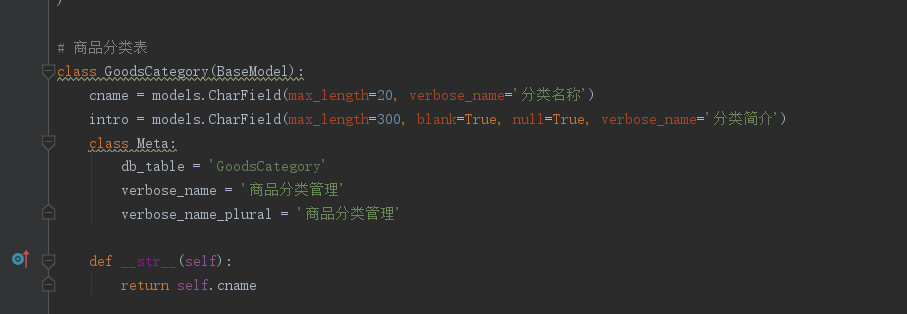
Js部分

<script type="application/javascript">  
{# js 代码网页加载完毕才能执行 #}  
 $(**function** () {  
 //网页加载完毕才执行  
 //1. 绑定点击事件  
 $('#yzm-hq').on('click',**function**(){  
 //测试  
 {#console.debug(111)#}  
 //2. 获取手机号码, 手机号码上添加一个id  
 **var** user\_name = $('#user\_name').val();  
 **var** reg = /^1[3-9]\d{9}$/;  
 //验证手机号码  
 **if** (!reg.test(user\_name)){  
 //验证失败  
 alert("手机号码格式错误!");  
 **return**;  
 }  
  
 **var** self = **this**;  
 //3. 发送ajax请求  
 $.ajax({  
 type:'post',  
 url:"{% url 'user:sendmsg' %}",  
 data:{'user\_name':user\_name,'csrfmiddlewaretoken':'{{ csrf\_token }}'},  
 dataType:"json",  
 success:**function** (data) {  
 //4. 判断是否发送成功  
 **if**(data.error == 0){  
  
 //1. 按钮颜色变灰  
 $(self).css("background-color","gray");  
 //2. 按钮不能再点击  
 $(self).prop('disabled',**true**);  
 //3. 开始倒计时 周期型执行函数  
 **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. 按钮颜色变绿  
 $(self).css("background-color","#76bb2a");  
 //2. 按钮能再点击  
 $(self).prop('disabled',**false**);  
 //3. 返回原来的提示文字  
 $(self).val("获取验证码");  
 //4. 清除定时  
 window.clearInterval(timer);  
 //返回  
 **return**;  
 }  
 msg = time + "秒后重新发送";  
 $(self).val(msg);  
 },1000);  
  
 }**else** {  
 alert(data.errmsg)  
 }  
 }  
 })  
 })  
 })  
  
</script>

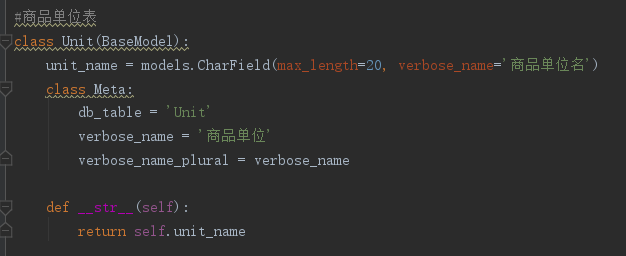
视图部分：

# 发送验证码信息  
class SendMsg(View):  
 def get(self, request):  
 pass  
  
 def post(self, request):  
 user\_name = request.POST.get('user\_name', '')  
 # 验证数据的合法性  
 rs = re.search('^1[3-9]\d{9}$', user\_name)  
 if rs is None:  
 return JsonResponse({'error': 1, 'errmsg': '电话号码格式错误!'})  
  
 # 生成随机验证码  
 random\_code = "".join([str(random.randint(0, 9)) for \_ in range(6)])  
 print("=============随机验证码为==={}==============".format(random\_code))  
  
 # >>>2. 保存验证码到redis中  
 # 获取连接  
 r = get\_redis\_connection()  
 # 保存手机号码对应的验证码  
 r.set(user\_name, random\_code)  
 r.expire(user\_name, 60) # 设置60秒后过期  
  
 # 首先获取当前手机号码的发送次数  
 key\_times = "{}\_times".format(user\_name)  
 now\_times = r.get(key\_times) # 从redis获取的二进制,需要转换  
 # print(int(now\_times))  
 if now\_times is None or int(now\_times) < 5:  
 # 保存手机发送验证码的次数, 不能超过5次  
 r.incr(key\_times)  
 # 设置一个过期时间  
 r.expire(key\_times, 3600) # 一个小时后再发送  
 else:  
 # 返回,告知用户发送次数过多  
 return JsonResponse({"error": 1, "errmsg": "发送次数过多"})  
  
 # >>>3. 接入运营商  
 \_\_business\_id = uuid.uuid1()  
 params = "{\"code\":\"%s\",\"product\":\"测试测试\"}" % random\_code  
 # print(params)  
 rs = send\_sms(\_\_business\_id, user\_name, "注册验证", "SMS\_2245271", params)  
 print(rs.decode('utf-8'))  
  
 # 3. 合成响应  
 return JsonResponse({'error': 0})

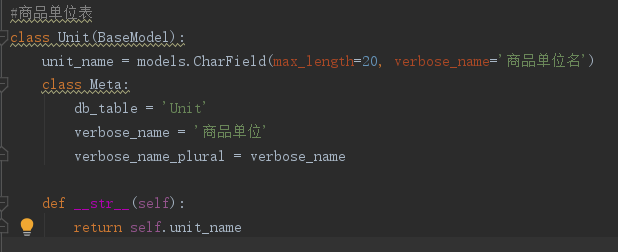
1. 商品功能模块
   1. 需求
2. 展示商品分类
3. 展示商品列表，可以根据销量，综合，价格，新品进行排序
4. 展示商品详情页
5. 展示商品主页，主页包括一些小功能，还有多个专区
6. 可以根据用户的id定位到对应的城市
   1. 流程
      * 1. 拿到网页的静态文件先分析网页
        2. 渲染每个网页，保证每个网页都可以正确进入
        3. 设置数据模型
        4. 后台管理添加数据
        5. 把数据渲染进网页内
   2. 设计要点（数据库和页面交互）
      * 1. 数据库设计

商品分类

商品spu表



商品单位表



商品sku表



商品相册表



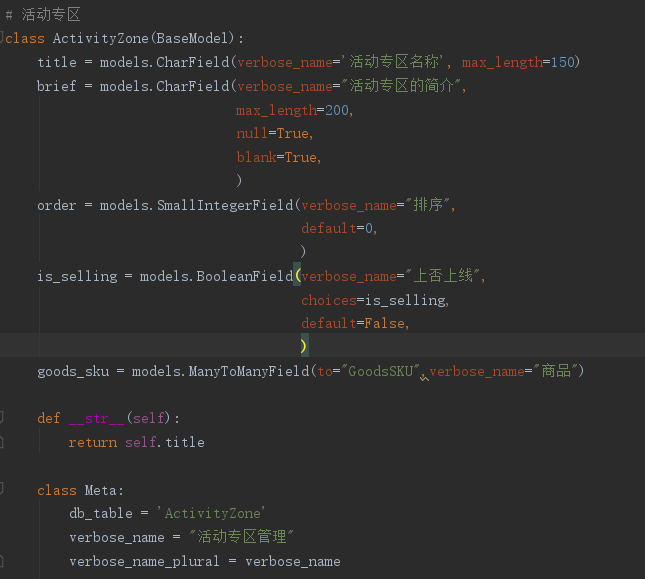
首页轮播



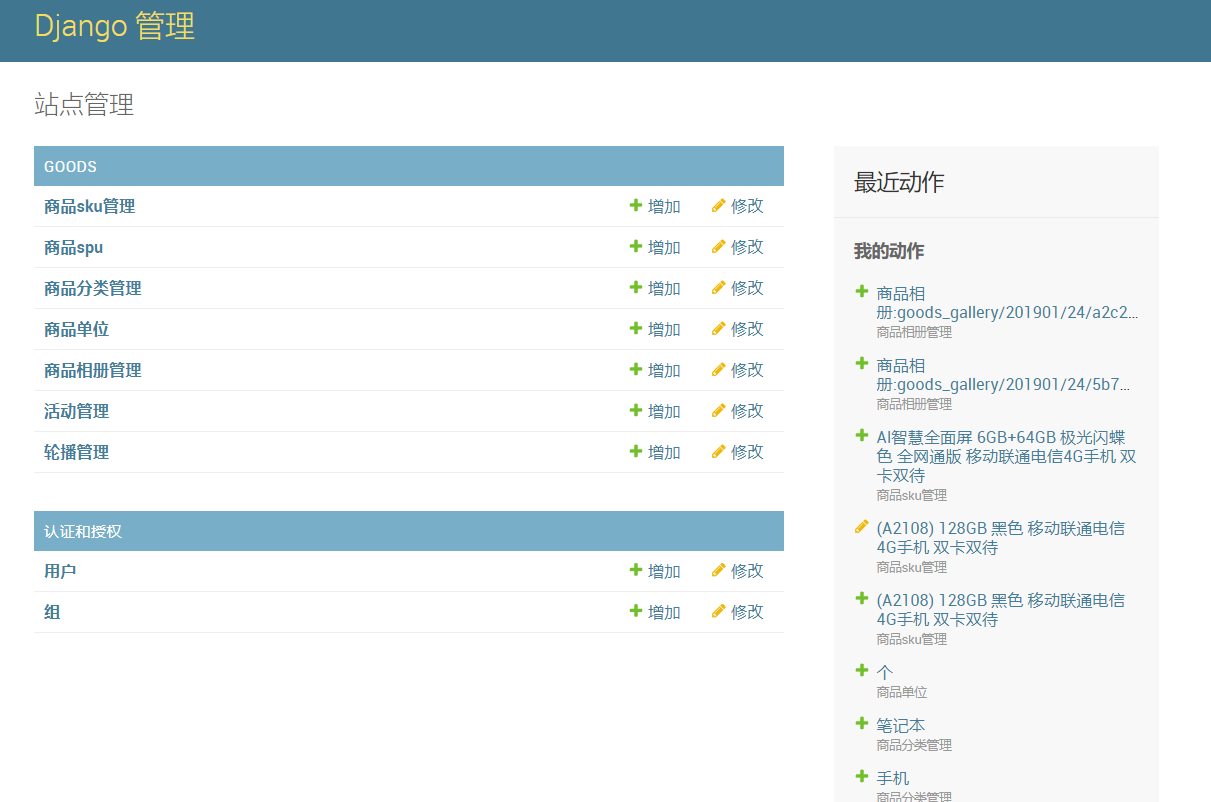
首页活动



活动专区



在后台数据库中添加数据





将数据渲染进网页

点击每个商品都可以看到对应的详细数据

在分类管理中可以根据分类加载对应数据，还可以按销量，综合，价格，时间等字段排序

* 1. 要点难点及解决方案
     + 1. 建立数据模型表

分析清除网页，搞清楚数据之间的联系，注意对应字段之间的细节

* + - 1. 一对多查询

主要是搞清楚正向和逆行查询。

正向查询语法

stu = Student.objects.get(pk=7)

print(stu.classinfo)

逆向查询语法

classinfo = ClassInfo.objects.get(pk=1)

student\_list = classinfo.student\_set.all().values()

print(student\_list)

* + - 1. 将数据渲染进页面

注意查询语句的使用，分清楚正向查询和逆向查询

逆向查询时多的一方的模型类要小写，语法规定

* + - 1. 如何点击对应的商品查询出对应的数据渲染进网页

传入对应的主键进行查询即可

* + - 1. 富文本框的编辑使用
  1. 在您的python路径中安装或添加django-ckeditor

Pip install Django-ckeditor

* 1. 将ckeditor添加到您的INSTALLED\_APPS设置中。
  2. 配置静态文件路径

STATIC\_ROOT = os.path.join(BASE\_DIR, "static")

* 1. 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

python.exe manage.py collectstatic

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

CKEDITOR\_UPLOAD\_PATH = "uploads/"

* 1. 将ckeditor\_uploader添加到您的INSTALLED\_APPS设置中
* 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urlpatterns = [
* url(r'^admin/', admin.site.urls),
* # 上传部件自动调用的上传地址
* url(r'^ckeditor/', include("ckeditor\_uploader.urls")),
* # 用户模块
* url(r'^user/', include("sp\_user.urls", namespace="sp\_user")),
* # 其他模块
* ]

7.

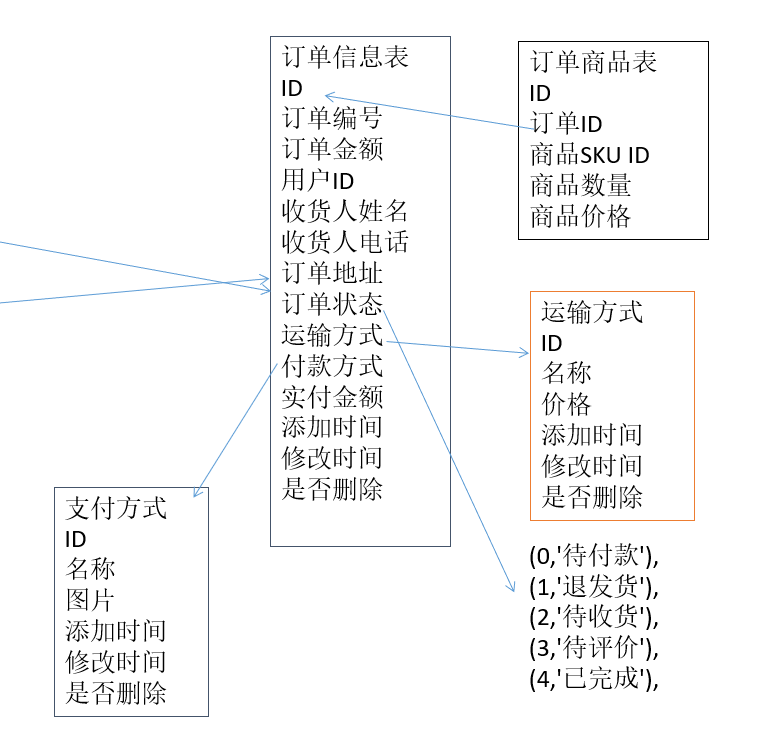
* 设置ckeditor在后台显示的样式（显示效果）   
  使用默认配置就行，具体看文档修改
* # 编辑器样式配置
* CKEDITOR\_CONFIGS = {
* 'default': {
* 'toolbar': 'full',
* },
* }

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

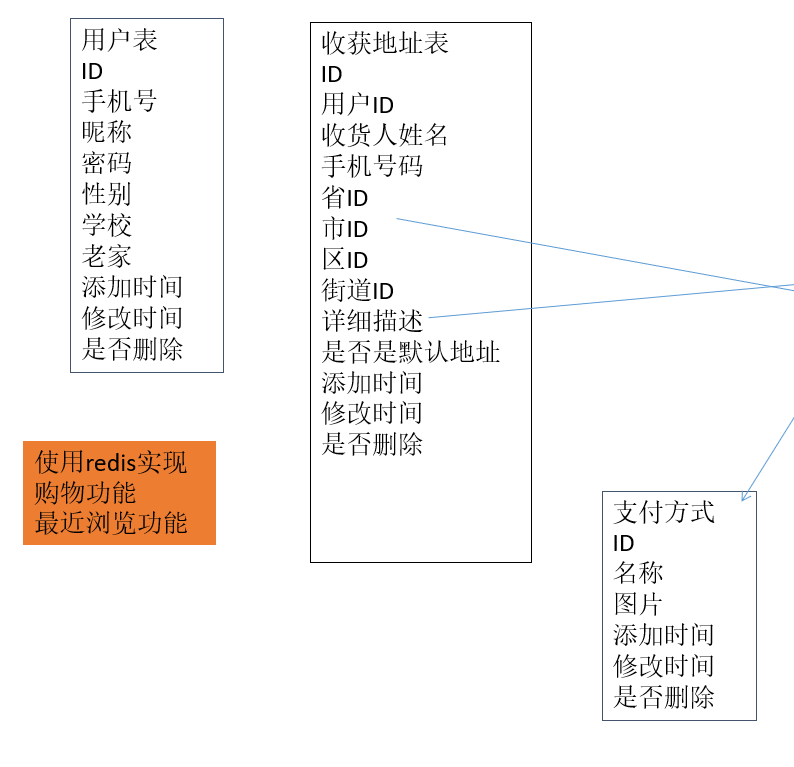
9

前台模板显示   
{{content|safe}}

1. 购物车模块
   1. 需求
      * 1. 将用户选中的商品及数量放入购物车内
        2. 访问购物车是展示数据库中的商品
        3. 添加以及减少数据通过ajax实现
   2. 流程
      * 1. 分析静态网页，将其渲染出来
        2. 将用户提交的数据存入redis中
        3. 将数据展示在购物车页面中
        4. 完成添加商品功能
        5. 完成减少商品功能
   3. 设计要点
      * 1. 将数据存入redis中
        2. 通过redis中存入的商品sku\_id,查询出对应的商品展示在页面上
   4. 要点难点及解决方案
      * 1. Redis的使用
        2. 对数据进行判断
2. 订单模块
   1. 需求
      * 1. 用户点击购物车中的去结算按钮后，跳转到确认订单页面
        2. 添加用户的收获地址
        3. 展示用户的购买的商品，计算价格
        4. 点击提交订单后，创建订单，将订单保存到数据库中
        5. 跳转到确认支付页面
   2. 流程
      * 1. 渲染确认订单页面
        2. 创建用户收获地址表
        3. 实现用户收获地址的添加及删除和展示
        4. 将用户的收获地址展示在确认订单页面
        5. 将商品展示在确认订单页面
        6. 创建运输方式表，并将数据展示在确认订单页面
        7. 计算总价格，提交订单
        8. 创建订单表，将数据保存到数据库中
        9. 跳转到确认支付页面
   3. 设计要点
      * 1. 设计用户订单表



* + - 1. 创建收获地址表



* 1. 要点及解决方案
     + 1. 订单是要确保正确的数据，要将下单时的重要信息都保存下来，而不是引用别的表的
       2. 确认订单时可以使用ajax先去创建订单，创建成功后在跳转。

1. 常见面试问题