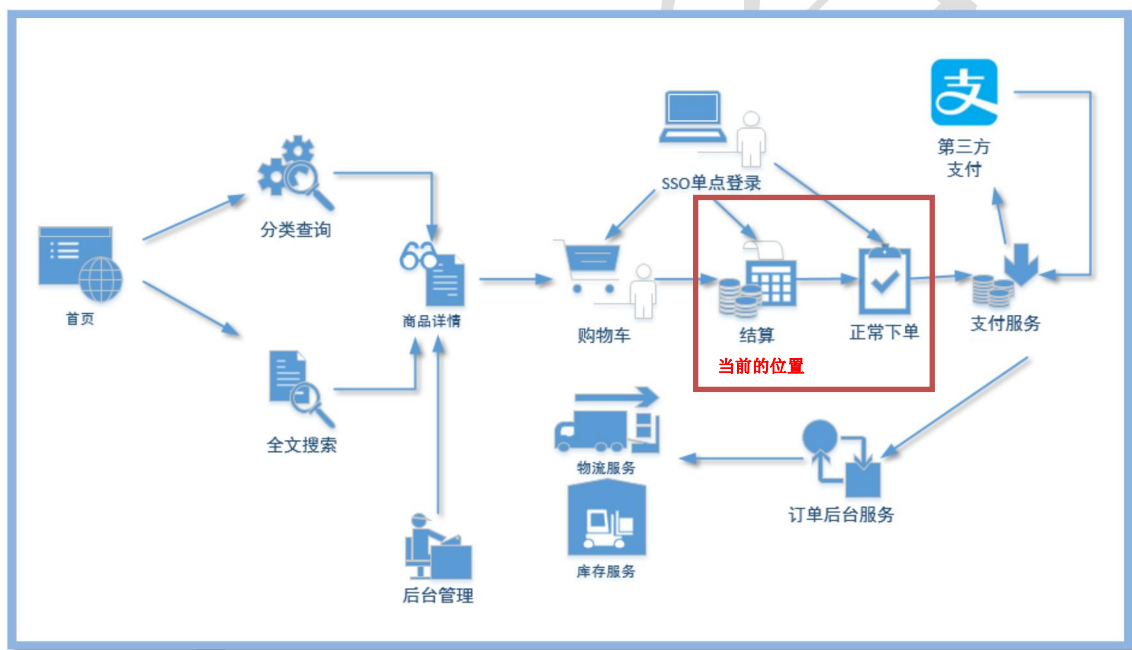


订 单

版本：V 1.0

www.atguigu.com

一、业务介绍



订单业务在整个电商平台中处于核心位置，也是比较复杂的一块业务。是把“物”变为“钱”的一个中转站。

整个订单模块一共分四部分组成：

1. 结算
2. 下单
3. 对接支付服务
4. 对接库存管理系统

二、结算页

入口：购物车点击计算按钮



1 搭建模块

1.1 gmall-order-web（模块已添加）

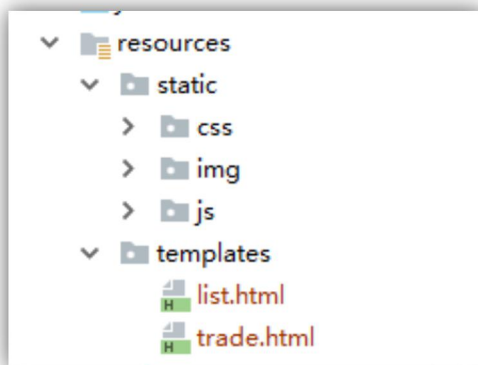
application.properties

```
spring.dubbo.application.name=order-web
spring.dubbo.registry.protocol=zookeeper
spring.dubbo.registry.address=192.168.67.163:2181
spring.dubbo.base-package=com.atguigu.gmall
spring.dubbo.protocol.name=dubbo
spring.dubbo.consumer.timeout=10000
spring.dubbo.consumer.check=false

server.port=8088
```

```
spring.thymeleaf.cache=false  
spring.thymeleaf.mode=LEGACYHTML5
```

拷贝静态资源文件和 html



host 文件

```
# gmall  
192.168.67.163    cart.gmall.com    passport.atguigu.com    item.gmall.com    list.gmall.com  
manage.gmall.com www.gmall.com    resource.gmall.com    order.gmall.com
```

nginx.conf

```
upstream order.gmall.com{  
    server 192.168.67.1:8088;  
}  
server {  
    listen 80;  
    server_name order.gmall.com;  
    location / {  
        proxy_pass http://order.gmall.com;  
        proxy_set_header X-forwarded-for $proxy_add_x_forwarded_for;  
    }  
}
```

2 分析

分析页面需要的数据:

- 1、用户地址列表（已完成）

3

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可访问百度: [尚硅谷官网](#)

2、购物车中选择的商品列表

3 购物车中选择的商品列表

3.1 在 CartServiceImpl.java

```
public List<CartInfo> getCartCheckedList(String userId){
    // 优先从缓存中取值
    Jedis jedis = redisUtil.getJedis();
    List<String> skuJsonlist = jedis.hvals("user:" + userId + ":cartChecked");
    List<CartInfo> cartInfoList=new ArrayList<>();
    if(skuJsonlist!=null &&skuJsonlist.size()!=0){
        // 序列化
        for (String skuJson : skuJsonlist) {
            CartInfo cartInfo = JSON.parseObject(skuJson, CartInfo.class);
            cartInfoList.add( cartInfo);
        }
    }
    return cartInfoList;
}
```

要把查询出来的 cartInfoList 装配到 orderDetailList 中

3.2 订单明细的数据结构:

order_detail(订单明细表)	
 id	编号
 order_id	订单编号
 sku_id	
 sku_name	sku名称(冗余)
 img_url	图片名称(冗余)
 order_price	购买价格(下单时sku价格)
 sku_num	购买个数

在 bean 下建立 order_detail

```
public class OrderDetail implements Serializable {
```

```
@Id
@Column
private String id;
@Column
private String orderId;
@Column
private String skuld;
@Column
private String skuName;
@Column
private String imgUrl;
@Column
private BigDecimal orderPrice;
@Column
private Integer skuNum;

@Transient
private String hasStock;
}
```

其中 hasStock 是一个非持久化属性，用户传递【是否还有库存】的标志。

3.3 OrderController

加入 tradeInit 方法

```
@RequestMapping(value = "trade", method = RequestMethod.GET)
@loginRequire
public String tradeInit(HttpServletRequest request, Model model){
    String userId=(String)request.getAttribute("userId");

    //查询用户地址信息
    List<UserAddress> addresssList = userManagerService.getUserAddressList( userId );
    request.setAttribute("addressList",addresssList);

    //查询商品清单
    List<CartInfo> cartInfoList=cartService.getCartCheckedList(userId);

    List<OrderDetail> orderDetailList=new ArrayList();
    Integer totalNum=0;
    BigDecimal totalAmount=new BigDecimal(0);
    for (CartInfo cartInfo : cartInfoList) {
        OrderDetail orderDetail=new OrderDetail();

        orderDetail.setSkuld(cartInfo.getSkuld());
        orderDetail.setSkuName(cartInfo.getSkuName());
        orderDetail.setOrderPrice(cartInfo.getSkuPrice());
        orderDetail.setSkuNum(cartInfo.getSkuNum());
        orderDetail.setImgUrl(cartInfo.getImgUrl());
    }
}
```

```

        totalAmount=
        BigDecimal(cartInfo.getSkuNum())) );//总金额
        orderDetailList.add(orderDetail);
    }

    model.addAttribute("orderDetailList",orderDetailList);
    model.addAttribute("totalAmount",totalAmount);
    return "trade";
}

```

3.4 结算页面

```

<!--地址-->
<div class="top-3">
    <ul>
        <li class=".address default selected" th:each="address:${addressList}">
            <input name="deliveryAddress" type="radio" th:value="${address.userAddress}"
            th:checked="${address.isDefault=='1'}">
            <span th:text="${address.consignee}"> </span><span
            th:text="${address.userAddress}"> </span>
        </li>
    </ul>
</div>

```

商品清单部分

```

<div class="to_right">
    <h5>商家： 自营</h5>
    <!--图片-->
    <div class="yun1" th:each="detail:${orderDetailList}">
        
        <div class="mi">
            <div><p style="width: 500px;" th:text="${detail.skuName}"> </p> <span
            style="float: right"> <span align="center" style="color: red" th:text="'¥'+${detail.orderPrice}">
            </span> <span th:text="'x'+${detail.skuNum}"> </span> </div>
        </div>
    </div>
</div>

```

总金额

```

<div class="yfze">
    <p class="yfze a"><span class="z"> 应 付 总 额 : </span><span class="hq" th:text="
    ¥'+${totalAmount}"> </span></p>

```

```
<button id="submitButton" class="tijiao">提交订单</button>
</div>
```

点击提交

增加一个表单，用于提交订单

```
<form action="/confirm_order" method="post" id="orderForm">
  <input name="consignee" id="consignee" type="hidden"/>
  <input name="deliveryAddress" id="deliveryAddress" type="hidden"/>
  <input name="orderDesc" id="orderDesc" type="hidden"/>
  <span th:each="detail:${orderDetailList}">
    <input name="skulds" type="hidden" th:value="${detail.skuld}" />
    <input name="skuNums" type="hidden" th:value="${detail.skuNum}" />
  </span>
</form>
```

提交按钮的 js

```
$("#submitButton").click(function () {
  $("#consignee").val($("#input[type='radio']:checked").next().text());
  $("#deliveryAddress").val($("#input[type='radio']:checked").next().next().text());
  $("#orderDesc").val($("#order_desc_page").val());
  console.log($("#orderForm").html());
  $("#orderForm").submit();
});
```

三、下单

1 数据结构



id	主键。自动生成
consignee	收货人名称。页面获取
consignee_tel	收货人电话。页面获取
deliveryAddress	收货地址。页面获取
total_amount	总金额。计算
order_status	订单状态，用于显示给用户查看。设定初始值。
userId	用户 id。从拦截器已放到请求属性中。
payment_way	支付方式（网上支付、货到付款）。页面获取
orderComment	订单状态。页面获取
out_trade_no	第三方支付编号。按规则生成
create_time	创建时间。设当前时间
expire_time	默认当前时间+1 天
process_status	订单进度状态，程序控制、 后台管理查看。设定初始值，
tracking_no	物流编号,初始为空，发货后补充
parent_order_id	拆单时产生，默认为空

id	主键，自动生成
order_id	订单编号，主表保存后给从表
sku_id	商品 id 页面传递
sku_name	商品名称，后台添加
img_url	图片路径，后台添加
order_price	商品单价，从页面中获取，并验价。
sku_num	商品个数，从页面中获取

增加实体 Bean

```
public class OrderInfo implements Serializable {  
    @Column  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private String id;  
  
    @Column  
    private String consignee;  
  
    @Column  
    private String consigneeTel;  
  
    @Column  
    private BigDecimal totalAmount;  
  
    @Column  
    private OrderStatus orderStatus;  
  
    @Column  
    private ProcessStatus processStatus;  
  
    @Column  
    private String userId;  
  
    @Column  
    private PaymentWay paymentWay;  
  
    @Column  
    private Date expectDeliveryTime;  
  
    @Column
```

```
private String deliveryAddress;

@Column
private String orderComment;

@Column
private Date createTime;

@Column
private String parentOrderId;

@Column
private String trackingNo;

@Transient
private List<OrderDetail> orderDetailList;

@Transient
private List<OrderInfo> orderSubList;

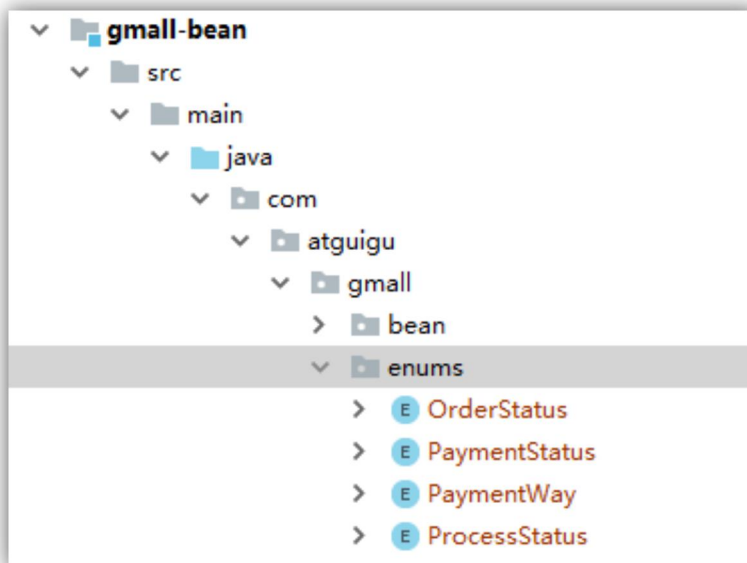
@Transient
private String wareId;

@Column
private String outTradeNo;

public void sumTotalAmount(){
    BigDecimal totalAmount=new BigDecimal("0");
    for (OrderDetail orderDetail : orderDetailList) {
        totalAmount= totalAmount.add(orderDetail.getOrderPrice().multiply(new
BigDecimal(orderDetail.getSkuNum())));
    }
    this.totalAmount= totalAmount;
}
}
```

增加枚举类

枚举类路径放到 gmall-bean 模块中和 bean 同级目录。



```
public enum OrderStatus {  
    UNPAID("未支付"),  
    PAID("已支付"),  
    WAITING_DELEVER("待发货"),  
    DELEVERED("已发货"),  
    CLOSED("已关闭"),  
    FINISHED("已完结"),  
    SPLIT("订单已拆分");  
  
    private String comment;  
  
    OrderStatus(String comment){  
        this.comment=comment;  
    }  
  
    public String getComment(){  
        return comment;  
    }  
  
    public void setComment(String comment){  
        this.comment = comment;  
    }  
}
```

```
public enum ProcessStatus {  
    UNPAID("未支付",OrderStatus.UNPAID),  
    PAID("已支付",OrderStatus.PAID),
```

```
NOTIFIED_WARE("已通知仓储",OrderStatus.PAID),
WAITING_DELEVER("待发货",OrderStatus.WAITING_DELEVER),
STOCK_EXCEPTION("库存异常",OrderStatus.PAID),
DELEVERED("已发货",OrderStatus.DELEVERED),
CLOSED("已关闭",OrderStatus.CLOSED),
FINISHED("已完结",OrderStatus.FINISHED) ,
PAY_FAIL("支付失败",OrderStatus.UNPAID),
SPLIT("订单已拆分",OrderStatus.SPLIT);
```

```
private String comment ;
private OrderStatus orderStatus;
```

```
ProcessStatus(String comment, OrderStatus orderStatus){
    this.comment=comment;
    this.orderStatus=orderStatus;
}
```

```
public String getComment() {
    return comment;
}
```

```
public void setComment(String comment) {
    this.comment = comment;
}
```

```
public OrderStatus getOrderStatus() {
    return orderStatus;
}
```

```
public void setOrderStatus(OrderStatus orderStatus) {
    this.orderStatus = orderStatus;
}
```

```
public enum PaymentWay {
    ONLINE("在线支付"),
    OUTLINE("货到付款");
```

```
private String comment ;
```

```
PaymentWay(String comment ){
    this.comment=comment;
}
```

```
public String getComment() {
    return comment;
}
```

```
public void setComment(String comment) {
    this.comment = comment;
}
```

```
}
```

由于涉及枚举类所以 `application.properties` 中要加入

```
mapper.enum-as-simple-type=true
```

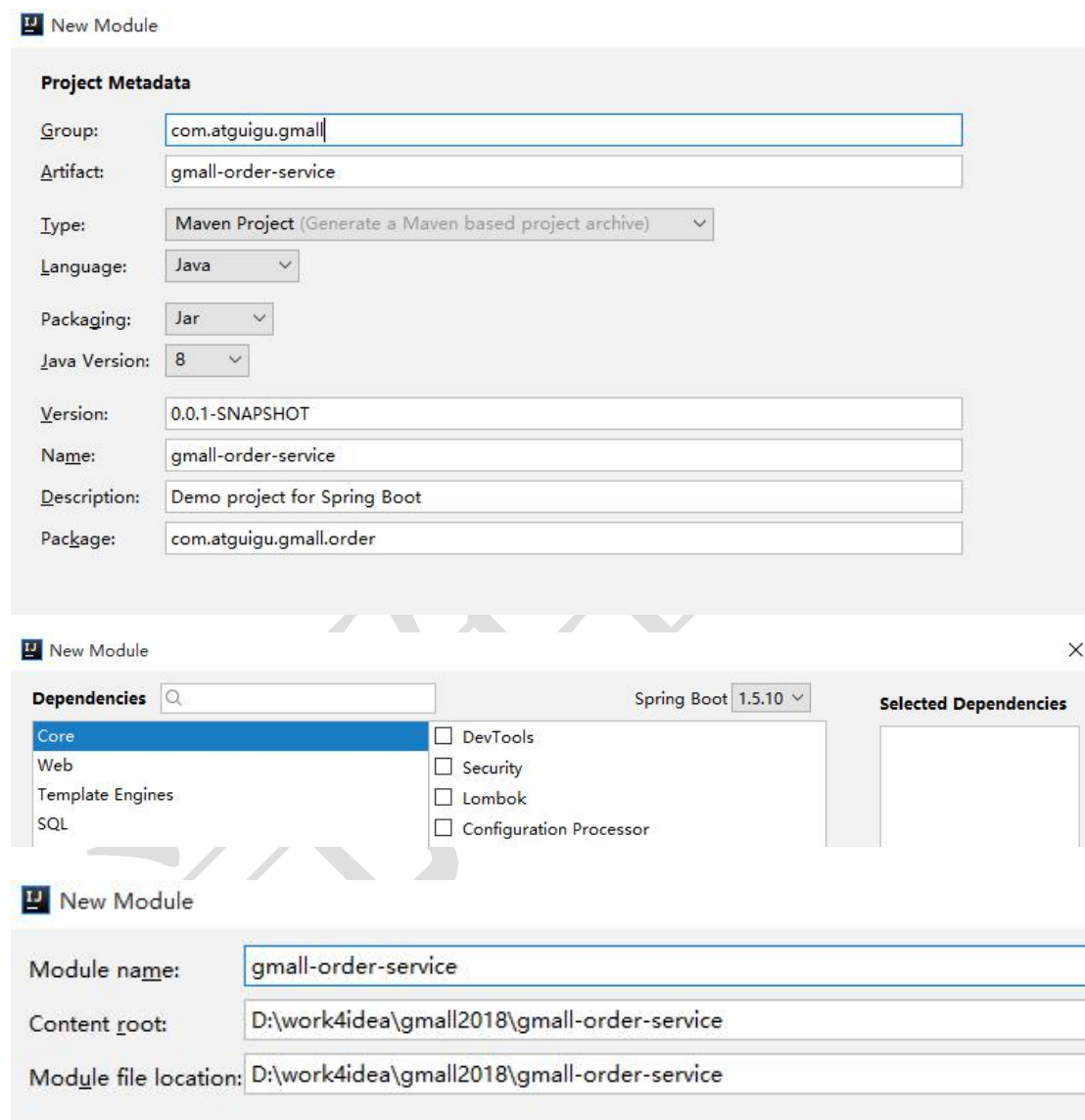
这个配置会把枚举类当成字符串处理。

2 分析下单：

1. 保存单据前要做交易：验库存、验价
2. 保存单据。
3. 保存以后把购物车中的商品删除
4. 重定向到支付页面。

3 订单后台服务模块

3.1 搭建 gmall-order-service



The image shows three screenshots of the IntelliJ IDEA 'New Module' dialog. The top screenshot shows the 'Project Metadata' tab with the following fields: Group (com.atguigu.gmall), Artifact (gmall-order-service), Type (Maven Project), Language (Java), Packaging (Jar), Java Version (8), Version (0.0.1-SNAPSHOT), Name (gmall-order-service), Description (Demo project for Spring Boot), and Package (com.atguigu.gmall.order). The middle screenshot shows the 'Dependencies' tab with a search bar, a list of dependencies (Core, Web, Template Engines, SQL), and a list of selected dependencies (DevTools, Security, Lombok, Configuration Processor). The bottom screenshot shows the 'New Module' dialog with the Module name (gmall-order-service), Content root (D:\work4idea\gmall2018\gmall-order-service), and Module file location (D:\work4idea\gmall2018\gmall-order-service).

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>

<groupId>com.atguigu.gmall</groupId>
<artifactId>gmall-order-service</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>gmall-order-service</name>
<description>Demo project for Spring Boot</description>

<parent>
  <groupId>com.atguigu.gmall</groupId>
  <artifactId>gmall-parent</artifactId>
  <version>1.0-SNAPSHOT</version>
</parent>
<dependencies>

  <dependency>
    <groupId>com.atguigu.gmall</groupId>
    <artifactId>gmall-interface</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>

  <dependency>
    <groupId>com.atguigu.gmall</groupId>
    <artifactId>gmall-service-util</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

application.properties

```
server.port=8078
```

```
logging.level.root=debug
```

```
spring.dubbo.application.name=order-service
```

```
spring.dubbo.registry.protocol=zookeeper
spring.dubbo.registry.address=192.168.67.163:2181
spring.dubbo.base-package=com.atguigu.gmall
spring.dubbo.protocol.name=dubbo
spring.dubbo.consumer.timeout=100000
spring.dubbo.consumer.check=false

spring.datasource.url=jdbc:mysql://59.110.141.236:3306/gmall?characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=123456

mybatis.mapper-locations=classpath:mapper/*Mapper.xml
mybatis.configuration.mapUnderscoreToCamelCase=true

spring.redis.host=192.168.67.163
spring.redis.port=6379
spring.redis.database=0
```

GmallOrderServiceApplication

```
@SpringBootApplication
@MapperScan(basePackages = "com.atguigu.gmall.order.mapper")
@ComponentScan(basePackages = "com.atguigu.gmall")
public class GmallOrderServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(GmallOrderServiceApplication.class, args);
    }
}
```

3.2 增加 OrderServiceImpl

```
@Service
public class OrderServiceImpl implements OrderService{

    @Autowired
    OrderInfoMapper orderInfoMapper;

    @Autowired
    OrderDetailMapper orderDetailMapper;

    @Override
    @Transactional
    public void saveOrder(OrderInfo orderInfo){
        //增加时间、状态、生成外部流水号
    }
}
```



```
orderInfo.setCreateTime(new Date());
if(orderInfo.getOrderStatus()==null){
    orderInfo.setOrderStatus(ProcessStatus.UNPAID.getOrderStatus());
}
if(orderInfo.getProcessStatus()==null){
    orderInfo.setProcessStatus(ProcessStatus.UNPAID);
}

orderInfo.setOutTradeNo(OrderConst.out_trade_no_prefix+System.currentTimeMillis()+"000"+new Random().nextInt(1000));

orderInfoMapper.insertSelective(orderInfo);
System.out.println(orderInfo);

List<OrderDetail> orderDetailList=orderInfo.getOrderDetailList();
for (OrderDetail orderDetail : orderDetailList) {
    orderDetail.setOrderId(orderInfo.getId());
    orderDetailMapper.insertSelective(orderDetail);
}
}
```

3.3 CartServiceImpl 中增加删除购物车方法

```
public void delCartCheckedList(String userId,List<String> skulds){

    Example example=new Example(CartInfo.class);
    example.createCriteria().andIn("skuld",skulds);
    cartInfoMapper.deleteByExample(example);

    Jedis jedis=redisUtil.getJedis();
    jedis.pipelined();
    for (String skuld : skulds) {
        jedis.hdel("user:" + userId + ":cartChecked",skuld);
        jedis.hdel("user:" + userId + ":cart",skuld);
    }
    jedis.sync();
    jedis.close();
}
```

3.4 OrderController

```
@RequestMapping(value = "submitOrder", method = RequestMethod.POST)
@loginRequire(debugUserId = "8")
public String submitOrder(OrderInfo orderInfo, HttpServletRequest request){
    String userId = (String) request.getAttribute("userId");

    List<OrderDetail> orderDetailList = orderInfo.getOrderDetailList();
    for (OrderDetail orderDetail : orderDetailList) {
        SkuInfo skuInfo = manageService.getSkuInfo(orderDetail.getSkuld());
        //验价
        if(!orderDetail.getOrderPrice().equals(skuInfo.getPrice())){
            cartService.loadCartCache(userId);
            request.setAttribute("errMsg", "商品[" + skuInfo.getSkuName() + "]价格已发生变化, 请在购物车页面重新进行结算");
            return "tradeFail";
        }

        orderDetail.setImgUrl(skuInfo.getSkuDefaultImg());
        orderDetail.setSkuName(skuInfo.getSkuName());
    }

    orderInfo.setUserId(userId);
    orderInfo.sumTotalAmount();

    orderService.saveOrder(orderInfo);

    List<String> skuldList = new ArrayList<>();
    for (OrderDetail orderDetail : orderDetailList) {
        skuldList.add(orderDetail.getSkuld());
    }
    cartService.delCartCheckedList(userId, skuldList);
    return "redirect://payment.gmall.com/index";
}
```

3.5 页面 trade.html

```
<form action="/submitOrder" method="post" id="orderForm">
    <input name="consignee" id="consignee" type="hidden"/>
    <input name="deliveryAddress" id="deliveryAddress" type="hidden"/>
    <input name="paymentWay" id="paymentWay" type="hidden"/>
    <input name="orderComment" id="orderComment" type="hidden"/>
    <span th:each="detail, stat: ${orderDetailList}">
        <input th:name="'orderDetailList[${stat.index}+'].skuId'"
```

```
type="hidden" th:value="${detail.skuId}" />
    <input th:name="'orderDetailList['+${stat.index}+'].skuNum'"
type="hidden" th:value="${detail.skuNum}" />
    <input th:name="'orderDetailList['+${stat.index}+'].orderPrice'"
type="hidden" th:value="${detail.orderPrice}" />
</span>

</form>
```

```
$("#submitButton").click(function () {
    $("#consignee").val($("#input[type='radio']:checked").next().text());

    $("#deliveryAddress").val($("#input[type='radio']:checked").next().next().text());
    $("#paymentWay").val("ONLINE");
    $("#orderComment").val($("#orderCommentPage").val());
    console.log($("#orderForm").html());
    $("#orderForm").submit();

});
```

4 如何解决用户利用浏览器刷新和回退重复提交订单？

在进入结算页面时，生成一个结算流水号，然后保存到结算页面的隐藏元素中，每次用户提交都检查该流水号与页面提交的是否相符，订单保存以后把后台的流水号删除掉。那么第二次用户用同一个页面提交的话流水号就会匹配失败，无法重复保存订单。

4.1 修改结算页增加流水号的生成。

4.1.1 OrderServiceImpl

```
//生成流水号
@Override
public String genTradeNo(String userId){
    Jedis jedis = redisUtil.getJedis();
    String userTradeNoKey="user:"+userId+":tradeNo";
    UUID uuid = UUID.randomUUID();
    jedis.setex(userTradeNoKey,OrderConst.tradeExpire,uuid.toString());
    return uuid.toString();
}

//验证流水号
@Override
public boolean checkTradeNo(String userId,String trackNo){
    Jedis jedis = redisUtil.getJedis();
    String userTradeNoKey="user:"+userId+":tradeNo";
    String uuid = jedis.get(userTradeNoKey);
    if(trackNo!=null&&trackNo.equals(uuid)){
        return true;
    }
    return false;
}

//删除流水号
@Override
public void delTradeNo(String userId ){
    Jedis jedis = redisUtil.getJedis();
    String userTradeNoKey="user:"+userId+":tradeNo";
    jedis.del(userTradeNoKey);
    return ;
}
```

4.1.2 常量类

```
public interface OrderConst {

    public static final String out_trade_no_prefix="ATGUGU";

    public static final int tradeExpire=10*60;

}
```

4.2 OrderController 中的 trade 方法中

```
String tradeNo = orderService.genTradeNo(userId);
request.setAttribute("tradeNo", tradeNo);
```

4.3 OrderController 中的 submitOrder 方法中

```
@RequestMapping(value = "submitOrder", method = RequestMethod.POST)
@loginRequire(debugUserId = "8")
public String submitOrder(OrderInfo orderInfo, HttpServletRequest request) {
    String userId = (String) request.getAttribute("userId");

    String tradeNo = request.getParameter("tradeNo");
    Boolean hasTradeNo = orderService.checkTradeNo(userId, tradeNo);
    if (!hasTradeNo) {
        request.setAttribute("errMsg", "结算页面过期或已失效，请重新结算。");
        return "tradeFail";
    }

    List<OrderDetail> orderDetailList = orderInfo.getOrderDetailList();
    for (OrderDetail orderDetail : orderDetailList) {
        SkuInfo skuInfo = manageService.getSkuInfo(orderDetail.getSkuld());
        // 验价
        if (!orderDetail.getOrderPrice().equals(skuInfo.getPrice())) {
            cartService.loadCartCache(userId);
            request.setAttribute("errMsg", "商品[" + skuInfo.getSkuName() + "]价格已发生变化，请在购物车页面重新进行结算");
            return "tradeFail";
        }

        orderDetail.setImgUrl(skuInfo.getSkuDefaultImg());
        orderDetail.setSkuName(skuInfo.getSkuName());
    }

    orderInfo.setUserId(userId);
    orderInfo.sumTotalAmount();

    orderService.saveOrder(orderInfo);
}
```

```
List<String> skuldList =new ArrayList<>();
for (OrderDetail orderDetail : orderDetailList) {
    skuldList.add(orderDetail.getSkuld());
}
cartService.delCartCheckedList(userId,skuldList);
orderService.delTradeNo(userId);
return "redirect://payment.gmall.com/index";
}
```

5 验库存

通过 restful 接口查询商品是否有库存

一般电商系统的商品库存，都不由电商系统本身来管理，由另外一套仓库管理系统，或者进销存系统来管理，电商系统通过第三方接口调用该系统。

由于库管系统可能是异构的系统，所以不在 dubbo 的分布式体系之内。只支持 restful 风格的 webservice 调用和消息队列的调用。

详见《库存管理系统手册》

根据手册中的接口文档，编写调用代码。

1、 查询库存

接口	/hasStock
请求参数	skuld : 商品 skuld num: 商品数量
请求方式	get
例	/hasStock?skuld=10221&num=2

5.1 OrderController 中注入 application.properties 的值

```
@Value("${ware_sys_url}")
public String ware_sys_url;
```

增加方法

```
private String hasStorkBySkuidAndNum(String skuid,String num) {  
    String url=ware_sys_url+"/hasStock";  
    url=url+"?skuId="+skuid+"&num="+num;  
    String result = HttpClientUtil.doGet(url);  
    return result;  
}
```

5.2 submitOrder 中增加 方法

```
@RequestMapping(value = "submitOrder",method = RequestMethod.POST)  
@LoginRequire(debugUserId = "8")  
public String submitOrder(OrderInfo orderInfo , HttpServletRequest request){  
    String userId = (String) request.getAttribute("userId");  
  
    String tradeNo= request.getParameter("tradeNo");  
    Boolean hasTradeNo=orderService.checkTradeNo(userId,tradeNo);  
    if(!hasTradeNo){  
        request.setAttribute("errMsg","结算页面过期或已失效，请重新结算。");  
        return "tradeFail";  
    }  
  
    List<OrderDetail> orderDetailList = orderInfo.getOrderDetailList();  
    for (OrderDetail orderDetail : orderDetailList) {  
        SkuInfo skuInfo = manageService.getSkuInfo(orderDetail.getSkuld());  
        //验价  
        if(!orderDetail.getOrderPrice().equals(skuInfo.getPrice())){  
            cartService.loadCartCache(userId);  
            request.setAttribute("errMsg","商品["+skuInfo.getSkuName()+"]价格已发生变化，请在购物车页面重新进行结算");  
            return "tradeFail";  
        }  
        String hasStock = hasStorkBySkuidAndNum(orderDetail.getSkuld(),  
String.valueOf(orderDetail.getSkuNum()));  
        if (hasStock==null || hasStock.equals("0")){  
            request.setAttribute("errMsg","商品["+skuInfo.getSkuName()+"]暂时缺货。");  
            return "tradeFail";  
        }  
  
        orderDetail.setImgUrl(skuInfo.getSkuDefaultImg());  
        orderDetail.setSkuName(skuInfo.getSkuName());  
    }  
}
```

```
}

orderInfo.setUserId(userId);
orderInfo.sumTotalAmount();

orderService.saveOrder(orderInfo);

List<String> skuldList =new ArrayList<>();
for (OrderDetail orderDetail : orderDetailList) {
    skuldList.add(orderDetail.getSkuld());
}
cartService.delCartCheckedList(userId,skuldList);
orderService.delTradeNo(userId);
return "redirect://payment.gmall.com/index";
}
```

我的订单(不做讲解)

OrderInfoMapper

```
public interface OrderInfoMapper extends Mapper<OrderInfo>{
    public List<OrderInfo> selectOrderListByUser(Long userId);
}
```

OrderInfoMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.atguigu.gmall.order.mapper.OrderInfoMapper">
    <select id="selectOrderListByUser" parameterType="long"
    resultMap="orderInfoListMap">
        SELECT o.id,
        o.consignee,
        o.consignee_tel,
        o.total_amount,
        o.order_status,
        o.user_id ,
        payment_way,
        o.delivery_address,
        o.order_comment,
        o.out_trade_no,
        o.create_time ,
```



```
o.expire_time ,
o.process_status,
o.tracking_no,
o.parent_order_id,
od.order_id,
od.order_price,
od.sku_num,
od.id order_detail_id,
od.img_url,
od.sku_id,
od.sku_name
FROM   order_info o INNER JOIN order_detail od ON o.id=od.order_id
WHERE user_id=#{userId} order by create_time desc

</select>
<resultMap      id="orderInfoListMap"      type="com.atguigu.gmall.bean.OrderInfo"
autoMapping="true">
  <result property="id" column="id" ></result>

  <collection property="orderDetailList" ofType="com.atguigu.gmall.bean.OrderDetail"
autoMapping="true">
    <result property="id" column="order_detail_id" ></result>
  </collection>
</resultMap>
</mapper>
```

OrderServiceImpl

```
@Override
public List<OrderInfo> getOrderListByUser(String userId) {
    // 优先去查缓存
    // 缓存未命中 去查库

    List<OrderInfo> orderInfoList =
    orderInfoMapper.selectOrderListByUser(Long.parseLong(userId));
    return orderInfoList;
}
```

OrderController

```
@RequestMapping(value = "list", method = RequestMethod.GET)
@loginRequire
public String getOrderList(HttpServletRequest httpServletRequest, Model model) {
    String userId = (String) httpServletRequest.getAttribute("userId");
    List<OrderInfo> orderList = orderService.getOrderListByUser(userId);
    model.addAttribute("orderList", orderList);
    return "list";
}
```

list.html (涉及渲染的片段)

```
<table class="table" th:each="orderInfo:${orderList}">
    <tr>
        <td colspan="7" style="background:#F7F7F7" class="order-header" >
            <span style="color:#AAAAAA"
th:text="${#dates.format(orderInfo.createTime,'yyyy-mm-dd HH:MM:SS')}"> </span>
            <span><ruby style="color:#AAAAAA"> 订 单 号 :</ruby><span
th:text="${orderInfo.id}"></span> </span>
            <span th:text="${orderInfo.orderStatus.getComment()}">已拆分 </span>
        </td>
    </tr>

    <!-- 不拆单情况----->
    <!-- 不拆单情况----->
    <!-- 不拆单情况----->
    <tr
th:each="orderDetail,state:${orderInfo.orderDetailList}">
        <td colspan="3" class="item">
            
            <div>
                <p th:text="${orderDetail.skuName}"></p>
            </div>
            <div style="margin-left:15px;" th:text="'x'+${orderDetail.skuNum}"></div>
        </td>

        <td
th:if="${state.index==0}" th:rowspan="${state.size}" ><span
th:text="${orderInfo.consignee}"></span> </td>
        <td th:if="${state.index==0}" th:rowspan="${state.size}">
            <div style="margin-left:15px;" th:text="'总额 ￥'+${orderInfo.totalAmount}"> </div>
            <hr style="width:90%;">
            <p th:text="${orderInfo.paymentWay.getComment()}">在线支付</p>
        </td>
        <td th:if="${state.index==0}" th:rowspan="${state.size}">
            <ul>
                <li style="color:#71B247;" th:text="${orderInfo.orderStatus.getComment()}">等待收货
```

```
</li>
<li style="margin:4px 0;" class="hide"><i class="table_i2"></i> 跟 踪 <i
class="table_i3"></i>
  <div class="hi">
    <div class="p-tit">
      普通快递 运单号:390085324974
    </div>
    <div class="hideList">
      <ul>
        <li>
          [北京市] 在北京昌平区南口公司进行签收扫描,快件已被拍照(您
          的快件已签收,感谢您使用韵达快递)签收
        </li>
        <li>
          [北京市] 在北京昌平区南口公司进行签收扫描,快件已被拍照(您
          的快件已签收,感谢您使用韵达快递)签收
        </li>
        <li>
          [北京昌平区南口公司] 在北京昌平区南口公司进行派件扫描
        </li>
        <li>
          [北京市] 在北京昌平区南口公司进行派件扫描;派送业务员:业务员;联系
          电话:17319268636
        </li>
      </ul>
    </div>
  </div>
</li>
<li class="tdLi">订单详情</li>
</ul>
</td>
<td th:if="${state.index==0}" th:rowspan="${state.size}">
  <button>确认收货</button>
  <p style="margin:4px 0;">取消订单</p>
  <p>催单</p>
</td>
</tr>
</table>
```