谷粒商城

版本：V 1.0

购物车

# 一、 购物车业务简介



**当前的位置**

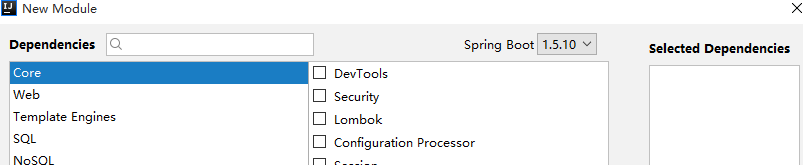
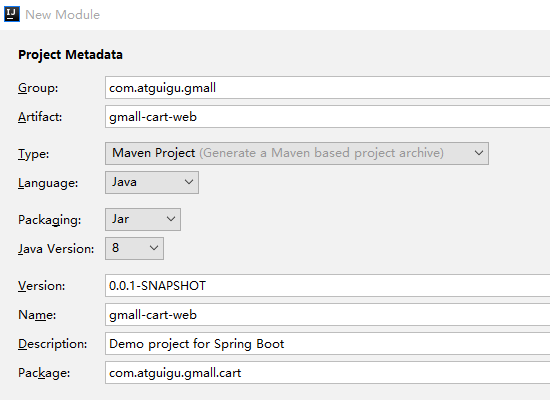
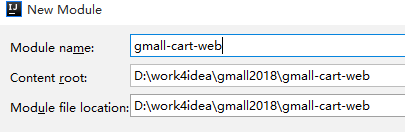
购物车模块要能过存储顾客所选的的商品，记录下所选商品，还要能随时更新，当用户决定购买时，用户可以选择决定购买的商品进入结算页面。

功能要求：

1. 要持久化，保存到数据库中。Mysql -- cartInfo
2. 利用缓存提高性能。Redis
3. 未登录状态也可以存入购物车，一旦用户登录要进行合并操作。Cookie --【Redis】

# 二、 购物车模块搭建

## 1 gmall-cart-web

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-cart-web</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>gmall-cart-web</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-web-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=8088 spring.thymeleaf.cache=false spring.thymeleaf.mode=LEGACYHTML5  spring.dubbo.application.name=cart-web spring.dubbo.registry.protocol=zookeeper spring.dubbo.registry.address=192.168.67.202:2181 spring.dubbo.base-package=com.atguigu.gmall spring.dubbo.protocol.name=dubbo spring.dubbo.consumer.timeout=100000 spring.dubbo.consumer.check=false |

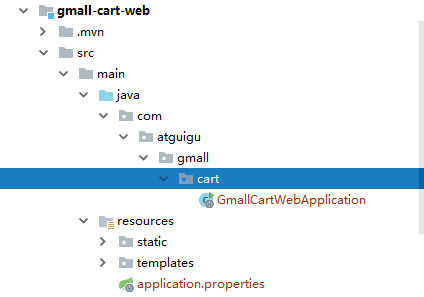
host文件

|  |
| --- |
| # 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 |

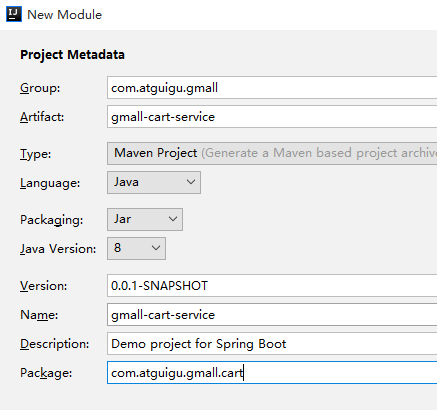
nginx.conf

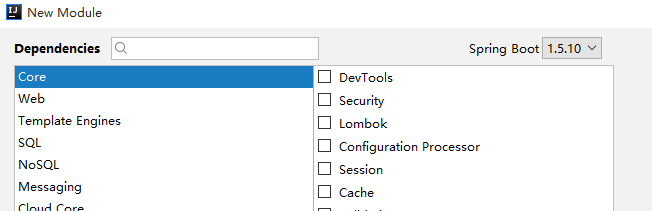
|  |
| --- |
| upstream cart.gmall.com{  server 192.168.67.1:8088;  }  server {  listen 80;  server\_name cart.gmall.com;  location / {  proxy\_pass http://cart.gmall.com;  proxy\_set\_header X-forwarded-for $proxy\_add\_x\_forwarded\_for;  }  } |

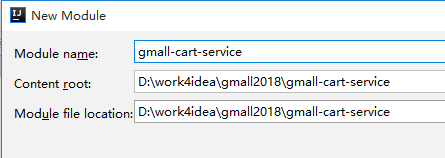
拷入静态文件和页面



## 2 gmall-cart-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-cart-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>gmall-cart-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

|  |
| --- |
| spring.datasource.url=jdbc:mysql://localhost:3306/gmall?characterEncoding=UTF-8 spring.datasource.username=root spring.datasource.password=root  logging.level.root=debug  server.port=8089  spring.dubbo.application.name=cartservice spring.dubbo.registry.protocol=zookeeper spring.dubbo.registry.address=192.168.67.202:2181 spring.dubbo.base-package=com.atguigu.gmall spring.dubbo.protocol.name=dubbo  spring.redis.host=192.168.67.202 spring.redis.port=6379 spring.redis.database=0 |

# 三、功能—添加入购物车

商品详情页面,item.html.

## 1 功能解析：

1. 根据**skuId**查询出商品详情skuInfo
2. 把skuInfo信息对应保存到购物车[购物车的实体类]
3. 返回成功页面

## 2 设计购物车的数据结构



**下单才会真正锁定价格！**

## 3 实体bean—CartInfo

|  |
| --- |
| **public class** CartInfo **implements** Serializable{   @GeneratedValue(strategy = GenerationType.***IDENTITY***)  @Id  @Column  String **id**;  @Column  String **userId**;  @Column  String **skuId**;  @Column  BigDecimal **cartPrice**;  @Column  Integer **skuNum**;  @Column  String **imgUrl**;  @Column  String **skuName**;   // 实时价格  @Transient  BigDecimal **skuPrice**;  // 下订单的时候，商品是否勾选  @Transient  String **isChecked**=**"0"**;  } |

## 4 Redis中的结构

购物车数据存放利用Hash结构存储：

String ,List,Hash,Set,ZSet

Set(key,value)

Demo:购物车中有两件商品，如果使用String 类型，要想修改第一个商品，则必须将所有的value全部进行反序列化。涉及到Io操作，性能降低。

Hset(key,field,value): field = skuId

要想修改第一个商品，hget(key,001,cartInfoJson); 需要修改哪件商品，则反序列化哪一个商品即可！

hset(user:[userId]:cart,skuId,CartInfoJson)

|  |
| --- |
| key： “user:[userId]:cart”  field: [skuId]  value: CartInfo (Json) |

## 5 gmall-cart-service 模块业务方法

### 1 思路：

1. 先检查该用户的购物车里是否已经有该商品
2. 如果有商品，只要把对应商品的数量增加上去就可以，同时更新缓存
3. 如果没有该商品，则把对应商品插入到购物车中，同时插入缓存。（可能会有缓存超时导致不一致的问题，建议直接同步购物车列表到缓存中）

## 别忘记写mapper

CartInfoMapper

### 2 CartServiceImpl

接口

|  |
| --- |
| **public interface** CartService {  **public** CartInfo addCart(String userId, String skuId, Integer num) } |

|  |
| --- |
| @Service **public class** CartServiceImpl **implements** CartService {   @Autowired  RedisUtil **redisUtil**;   @Autowired  CartInfoMapper **cartInfoMapper**;   @Reference  ManageService **manageService**;  @Override **public** CartInfo addCart(String userId, String skuId, Integer num) {  *// 为了防止 更新购物车前 缓存过期* loadCartCacheIfNotExists( userId) ;  *// 加数据库  // 尝试取出已有的数据 如果有 把数量更新 update 如果没有insert* CartInfo cartInfoQuery=**new** CartInfo();  cartInfoQuery.setSkuId(skuId);  cartInfoQuery.setUserId(userId);  CartInfo cartInfoExists=**null**;  cartInfoExists = **cartInfoMapper**.selectOne(cartInfoQuery);  SkuInfo skuInfo = **manageService**.getSkuInfo(skuId);  **if**(cartInfoExists!=**null**){  cartInfoExists.setSkuName(skuInfo.getSkuName());  cartInfoExists.setCartPrice(skuInfo.getPrice());  cartInfoExists.setSkuNum(cartInfoExists.getSkuNum()+num);  cartInfoExists.setImgUrl(skuInfo.getSkuDefaultImg());  **cartInfoMapper**.updateByPrimaryKeySelective(cartInfoExists);  }**else**{  CartInfo cartInfo = **new** CartInfo();  cartInfo.setSkuId(skuId);  cartInfo.setUserId(userId);  cartInfo.setSkuNum(num);  cartInfo.setImgUrl(skuInfo.getSkuDefaultImg());  cartInfo.setSkuName(skuInfo.getSkuName());  cartInfo.setCartPrice(skuInfo.getPrice());  cartInfo.setSkuPrice(skuInfo.getPrice());   **cartInfoMapper**.insertSelective(cartInfo);  cartInfoExists=cartInfo;  }  **return** cartInfoExists; }  } |
| @Override **public** List<CartInfo> cartList(String userId) {  *//先查缓存* Jedis jedis = **redisUtil**.getJedis();  String cartKey=**"cart:"**+userId+**":info"**;  List<String> cartJsonList = jedis.hvals(cartKey);  List<CartInfo> cartList=**new** ArrayList<>();  **if**(cartJsonList!=**null**&&cartJsonList.size()>0){ *//缓存命中* **for** (String cartJson : cartJsonList) {  CartInfo cartInfo = JSON.*parseObject*(cartJson, CartInfo.**class**);  cartList.add(cartInfo);  }   cartList.sort(**new** Comparator<CartInfo>() {  @Override  **public int** compare(CartInfo o1, CartInfo o2) {  **return** o2.getId().compareTo(o1.getId());  }  });   **return** cartList;  }**else** {  *//缓存未命中 //缓存没有查数据库 ，同时加载到缓存中* **return** loadCartCache(userId);  }  } |
| **public void** loadCartCacheIfNotExists(String userId){  String cartkey=**"cart:"**+userId+**":info"**;  Jedis jedis = **redisUtil**.getJedis();  Long ttl = jedis.ttl(cartkey);  **int** ttlInt = ttl.intValue();  jedis.expire(cartkey,ttlInt+10);  Boolean exists = jedis.exists(cartkey);  jedis.close();  **if**( !exists){  loadCartCache( userId);  }  } |

## 6 Web模块业务方法（gmall-cart-web）

### 1 思路：

1. 获得参数：skuId 、skuNum
2. 判断该用户是否登录，用userId判断
3. 如果登录则调用后台的service的业务方法
4. 如果未登录，利用cookie保存临时账号（token），利用该临时账号保存购物车

### 2 CartController

|  |
| --- |
| @Controller **public class** CartController {   @Reference  CartService **cartService**;    @PostMapping(**"addToCart"**)  @LoginRequire(autoRedirect = **false**)  **public** String addCart(@RequestParam(**"skuId"**) String skuId, @RequestParam(**"num"**) **int** num, HttpServletRequest request, HttpServletResponse response){  String userId =(String) request.getAttribute(**"userId"**);   **if**(userId==**null**){  *//如果用户未登录 检查cookie用户是否有token 如果有token 用token 作为id 加购物车 如果没有生成一个新的token放入cookie* userId = CookieUtil.*getCookieValue*(request, **"user\_tmp\_id"**, **false**);  **if**(userId==**null**){  userId = UUID.*randomUUID*().toString();  CookieUtil.*setCookie*(request,response,**"user\_tmp\_id"**,userId,60\*60\*24\*7,**false**);  }   }  CartInfo cartInfo = **cartService**.addCart(userId, skuId, num);  request.setAttribute(**"cartInfo"**,cartInfo);  request.setAttribute(**"num"**,num);   **return "success"**;  }  } |

### 3 添加成功后的展示页面

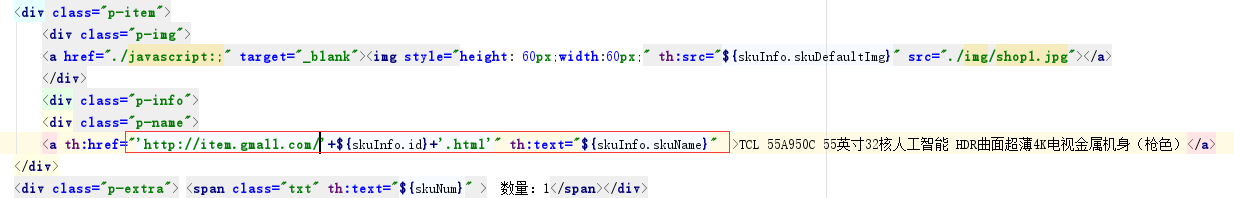
#### succuess.html

|  |
| --- |
| <div class="p-item">  <div class="p-img">  <a href="/javascript:;" target="\_blank"><img style="height: 60px;width:60px;" th:src="${skuInfo.skuDefaultImg}"></a>  </div>  <div class="p-info">  <div class="p-name">  <a th:href="'http://item.gmall.com/'+${skuInfo.id}+'.html'" th:title="${skuInfo.skuName}" th:text="${skuInfo.skuName}"> 商品名称</a> </div> <div class="p-extra"> <span class="txt" th:text="'数量:'+${num}"> 数量：XX</span></div> </div> |

需要注意的是：item.html 中表单提交需要给skuId 添加一个值

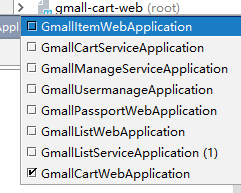
<**form id="itemForm" action="http://cart.gmall.com/addToCart" method="post"**>  
<**input type="text" name="skuNum" id="" value="1"** />  
<**input type="hidden" name="skuId" th:value="${skuInfo.id}"** />

</form>

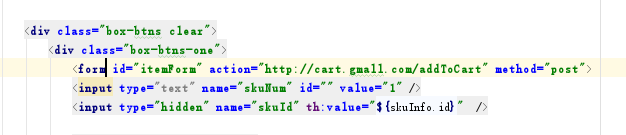


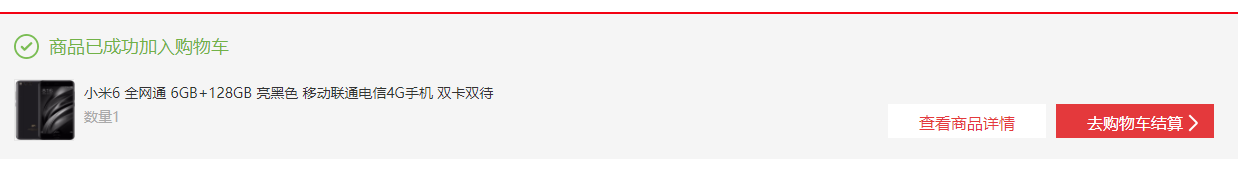
## 7测试效果

List,manage,item,cart服务+页面都要启动



点击添加购物车的时候，修改表单中的值 item.html





# 四、功能—展示购物车列表

## 1 功能解析

1. 展示购物中的信息
2. 如果**用户已登录**从缓存中取值，如果缓存没有，加载数据库。
3. 如果**用户未登录**从cookie中取出token，根据token取购物车。

## 2 CartController

|  |
| --- |
| @GetMapping(**"cartList"**) @LoginRequire(autoRedirect = **false**) **public** String cartList(HttpServletRequest request){  String userId =(String) request.getAttribute(**"userId"**);  List<CartInfo> cartList=**null**;  **if**(userId!=**null**){  cartList = **cartService**.cartList( userId);  }   String userTmpId=CookieUtil.*getCookieValue*(request, **"user\_tmp\_id"**, **false**);;  List<CartInfo> cartTempList=**null**;  **if**(userTmpId!=**null**){  cartTempList = **cartService**.cartList( userTmpId);  cartList=cartTempList;  }  request.setAttribute(**"cartList"**,cartList);   **return "cartList"**;  } |

## 3 CartServiceImpl

注意点：

1、redis中取出来要进行反序列化

2、redis的hash结构是无序的，要进行排序（可以用时间戳或者主键id，倒序排序）

3、如果redis中没有要从数据库中查询，要连带把最新的价格也取出来，默认要显示最新价格而不是当时放入购物车的价格，如果考虑用户体验可以把两者的差价提示给用户。

4、加载入缓存时一定要设定失效时间，保证和用户信息的失效时间一致即可。

接口

|  |
| --- |
| **// 缓存中没有数据，则从 数据库中加载**  **public** List<CartInfo> loadCartCache(String userId);  // 查询购物车集合列表 **public** List<CartInfo> getCartList(String userId); |

实现类

|  |
| --- |
| @Override **public** List<CartInfo> cartList(String userId) {  *//先查缓存* Jedis jedis = **redisUtil**.getJedis();  String cartKey=**"cart:"**+userId+**":info"**;  List<String> cartJsonList = jedis.hvals(cartKey);  List<CartInfo> cartList=**new** ArrayList<>();  **if**(cartJsonList!=**null**&&cartJsonList.size()>0){ *//缓存命中* **for** (String cartJson : cartJsonList) {  CartInfo cartInfo = JSON.*parseObject*(cartJson, CartInfo.**class**);  cartList.add(cartInfo);  }   cartList.sort(**new** Comparator<CartInfo>() {  @Override  **public int** compare(CartInfo o1, CartInfo o2) {  **return** o2.getId().compareTo(o1.getId());  }  });   **return** cartList;  }**else** {  *//缓存未命中 //缓存没有查数据库 ，同时加载到缓存中* **return** loadCartCache(userId);  }  } |
| */\*\*  \* 购物车查询，在数据库中查找  \** ***@param userId*** *\** ***@return*** *\*/* **public** List<CartInfo> loadCartCache(String userId){  *// 读取数据库* List<CartInfo> cartInfoList = **cartInfoMapper**.selectCartListWithSkuPrice(userId);  *//加载到缓存中  //为了方便插入redis 把list --> map* **if**(cartInfoList!=**null**&&cartInfoList.size()>0) {  Map<String, String> cartMap = **new** HashMap<>();  **for** (CartInfo cartInfo : cartInfoList) {  cartMap.put(cartInfo.getSkuId(), JSON.*toJSONString*(cartInfo));  }  Jedis jedis = **redisUtil**.getJedis();  String cartKey = **"cart:"** + userId + **":info"**;  jedis.del(cartKey);  jedis.hmset(cartKey, cartMap); *// hash* jedis.expire(cartKey, 60 \* 60 \* 24);  jedis.close();  }  **return** cartInfoList;  } |
| **Mapper接口**  **package** com.atguigu.gmall.cart.mapper;  **import** com.atguigu.gmall.bean.CartInfo; **import** tk.mybatis.mapper.common.Mapper;  **import** java.util.List;  **public interface** CartInfoMapper **extends** Mapper<CartInfo> {   **public** List<CartInfo> selectCartListWithSkuPrice(String userId);   **public void** mergeCartList(@Param(**"userIdDest"**) String userIdDest,@Param(**"userIdOrig"**) String userIdOrig); } |

CartInfoMapper.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>*** <**mapper namespace="com.atguigu.gmall0401.cart.mapper.CartInfoMapper"**>  <**select id="selectCartListWithSkuPrice" resultMap="cartMap"**>  SELECT ci.id,ci.user\_id,ci.sku\_id,ci.cart\_price ,ci.sku\_num ,ci.img\_url,ci.sku\_name, si.price sku\_price  FROM cart\_info ci JOIN sku\_info si ON ci.sku\_id=si.id  WHERE ci.user\_id=#{userId}   </**select**>   <**resultMap id="cartMap" type="com.atguigu.gmall0401.bean.CartInfo" autoMapping="true"**>  </**resultMap**>   <**insert id="mergeCartList"**>  INSERT INTO cart\_info (user\_id ,sku\_id,cart\_price ,sku\_num,img\_url,sku\_name)  SELECT #{userIdDest} ,sku\_id,cart\_price ,sku\_num,img\_url,sku\_name FROM cart\_info cc WHERE cc.user\_id=#{userIdOrig}  ON DUPLICATE KEY UPDATE cart\_info.`sku\_num`=cart\_info.`sku\_num`+cc.sku\_num  </**insert**>  </**mapper**> |

因为要取mapper.xml所以

application.properties中要加入

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

## 4 显示购物车列表页面

|  |
| --- |
| <li th:each=" cartInfo:${cartList}">  <div> </div>   <div>  <ol>  <li><input type="checkbox" class="check" onchange="*checkSku*(this)" /></li>  <li>  <dt><img th:src="${cartInfo.imgUrl}" src="img\shop1.jpg" alt=""></dt>  <dd th:onclick="'toItem('+${cartInfo.skuId}+')'" onclick="*toItem*(1)">  <p>  <span th:text="${cartInfo.skuName}" >TCL 55A950C 55英寸32核</span>  </p>  </dd>  </li>  <li>  <p class="dj" th:utext="((${#numbers.formatDecimal(cartInfo.cartPrice,1,2)}==${#numbers.formatDecimal(cartInfo.skuPrice,1,2)})?${cartInfo.cartPrice}:('<del>'+${cartInfo.cartPrice}+'</del>('+${cartInfo.skuPrice}+')'))" >￥4599.00</p>  </li>  <li>  <p>  <span>-</span>  <span th:text="${cartInfo.skuNum}">5</span>  <span>+</span>  </p>  </li>  <li style="font-weight:bold"><p class="zj" th:text="${cartInfo.skuNum}\*${#numbers.formatDecimal(cartInfo.skuPrice,1,2)}" >￥22995.00</p></li>  <li>  <p>删除</p>  </li>  </ol>  </div> </li>  </li> |

## 5 测试效果



# 五、功能--合并购物车

由于加入购物车时，用户可能存在登录和未登录两种情况，登录前在cookie中保存了一部分购物车信息，如果用户登录了，那么对应的要把临时的购物车合并到数据库中，并且刷新缓存。

## 1 CartController

增加判断如果用户是登录状态的，但是cookie里却还有购物车，说明需要把cookie中的购物车合并进来，同时把cookie中的清空。

|  |
| --- |
| @GetMapping(**"cartList"**) @LoginRequire(autoRedirect = **false**) **public** String cartList(HttpServletRequest request){  String userId =(String) request.getAttribute(**"userId"**); *//查看用户登录id* **if**(userId!=**null**){ *//有登录* List<CartInfo> cartList=**null**; *//如果登录前（未登录）时，存在临时购物车 ，要考虑合并* String userTmpId=CookieUtil.*getCookieValue*(request, **"user\_tmp\_id"**, **false**); *//取临时id* **if**(userTmpId!=**null**){  List<CartInfo> cartTempList = **cartService**.cartList( userTmpId); *//如果有临时id ，查是否有临时购物车* **if**( cartTempList!=**null**&&cartTempList.size()>0){  cartList= **cartService**.mergeCartList(userId,userTmpId); *// 如果有临时购物车 ，那么进行合并 ，并且获得合并后的购物车列表* }  }  **if**(cartList==**null**||cartList.size()==0){  cartList = **cartService**.cartList( userId); *//如果不需要合并 ，再取登录后的购物车* }  request.setAttribute(**"cartList"**,cartList);  }**else** { *//未登录 直接取临时购物车* String userTmpId=CookieUtil.*getCookieValue*(request, **"user\_tmp\_id"**, **false**);  **if**(userTmpId!=**null**) {  List<CartInfo> cartTempList = **cartService**.cartList(userTmpId);  request.setAttribute(**"cartList"**,cartTempList);  }   }   **return "cartList"**; } |

## 2 CartServiceImpl

|  |
| --- |
| */\*\*  \* 合并购物车  \** ***@param userIdDest*** *\** ***@param userIdOrig*** *\** ***@return*** *\*/* @Override **public** List<CartInfo> mergeCartList(String userIdDest, String userIdOrig) {  *//1 先做合并* **cartInfoMapper**.mergeCartList(userIdDest,userIdOrig);  *// 2 合并后把临时购物车删除* CartInfo cartInfo = **new** CartInfo();  cartInfo.setUserId(userIdOrig);  **cartInfoMapper**.delete(cartInfo);  *// 3 重新读取数据 加载缓存* List<CartInfo> cartInfoList = loadCartCache(userIdDest);   **return** cartInfoList; } |

# 六、选中状态的变更

用户每次勾选购物车的多选框，都要把当前状态保存起来。由于可能会涉及更频繁的操作，所以这个勾选状态不必存储到数据库中。保留在缓存状态即可。

## 1 页面动作的js

注意jquery1.6版本以后要用prop(“checked”)取得多选框选中状态。

|  |
| --- |
| **function** *checkSku*( chkbox){  **console**.log(**$**(chkbox));  **var** skuId= **$**(chkbox).attr(**"value"**);  **var** checked=**$**(chkbox).prop(**"checked"**);  **var** isCheckedFlag=**"0"**;  **if**(checked){  isCheckedFlag=**"1"**;  }  **var** param=**"isChecked="**+isCheckedFlag+**"&"**+**"skuId="**+skuId;  **console**.log(param);  **$**.post(**"checkCart"**,param,**function** (data) {  *sumSumPrice*();  }); } |

在页面89行添加

<**li**><**input type="checkbox" class="check" th:value="${cartInfo.skuId}" onchange="***checkSku*(**this**)**"** /></**li**>

## 2 CartController

同样这里要区分，用户登录和未登录状态。

如果登录，修改缓存中的数据，如果未登录，修改cookie中的数据。

|  |
| --- |
| @RequestMapping(**"checkCart"**) @ResponseBody @LoginRequire(autoRedirect = **false**) **public void** checkCart(HttpServletRequest request,HttpServletResponse response){  String skuId = request.getParameter(**"skuId"**);  String isChecked = request.getParameter(**"isChecked"**);  String userId=(String) request.getAttribute(**"userId"**);  **if** (userId!=**null**){  **cartService**.checkCart(skuId,isChecked,userId);  }**else**{  **cartCookieHandler**.checkCart(request,response,skuId,isChecked);  } } |

## 3 CartServiceImpl

把对应skuId的购物车的信息从redis中取出来，反序列化，修改isChecked标志。

再保存回redis中。

同时保存另一个redis的key 专门用来存储用户选中的商品，方便结算页面使用。

|  |
| --- |
| @Override **public void** checkCart(String userId, String skuId, String isChecked) {  loadCartCacheIfNotExists(userId);*// 检查一下缓存是否存在 避免因为缓存失效造成 缓存和数据库不一致   // isCheck数据 值保存在缓存中  //保存标志* String cartKey = **"cart:"** + userId + **":info"**;  Jedis jedis = **redisUtil**.getJedis();  String cartInfoJson = jedis.hget(cartKey, skuId);  CartInfo cartInfo = JSON.*parseObject*(cartInfoJson, CartInfo.**class**);  cartInfo.setIsChecked(isChecked);  String cartInfoJsonNew = JSON.*toJSONString*(cartInfo);  jedis.hset(cartKey,skuId,cartInfoJsonNew);  *// 为了订单结账 把所有勾中的商品单独 在存放到一个checked购物车中* String cartCheckedKey = **"cart:"** + userId + **":checked"**;  **if**(isChecked.equals(**"1"**)){ *//勾中加入到待结账购物车中， 取消勾中从待结账购物车中删除* jedis.hset(cartCheckedKey,skuId,cartInfoJsonNew);  jedis.expire(cartCheckedKey,60\*60);  }**else**{  jedis.hdel(cartCheckedKey,skuId);  }  jedis.close();  } |

CartCookieHandler中的checkCart

|  |
| --- |
| @PostMapping(**"checkCart"**) @LoginRequire(autoRedirect = **false**) @ResponseBody **public void** checkCart(@RequestParam(**"isChecked"**) String isChecked ,@RequestParam(**"skuId"**) String skuId,HttpServletRequest request){  String userId =(String)request.getAttribute(**"userId"**);  **if**(userId==**null**){  userId = CookieUtil.*getCookieValue*(request, **"user\_tmp\_id"**, **false**);  }   **cartService**.checkCart(userId,skuId,isChecked);  } |

## 4 修改页面isChecked的状态

在88行添加

|  |
| --- |
| <**li**><**input type="checkbox" class="check" th:value="${cartInfo.skuId}" onchange="***checkSku*(**this**)**" th:checked="(${cartInfo.isChecked}==1)?'true':'false'"** /></**li**> |