谷粒商城

版本：V 2.0

我的购物车

# 一 购物车业务简介

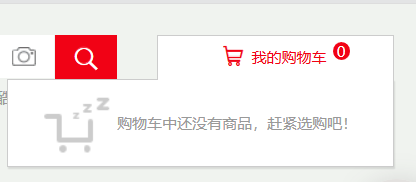


**当前的位置**

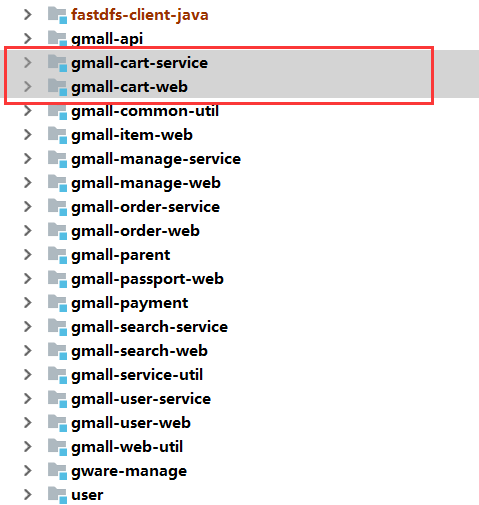
购物车模块要能过存储顾客所选的的商品，记录下所选商品，还要能随时更新，当用户决定购买时，用户可以选择决定购买的商品进入结算页面。

功能要求：

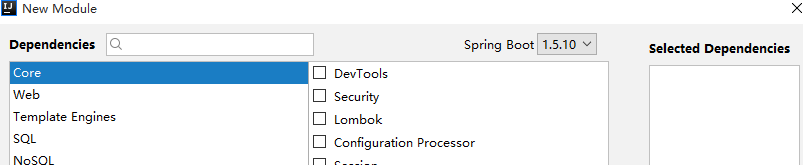
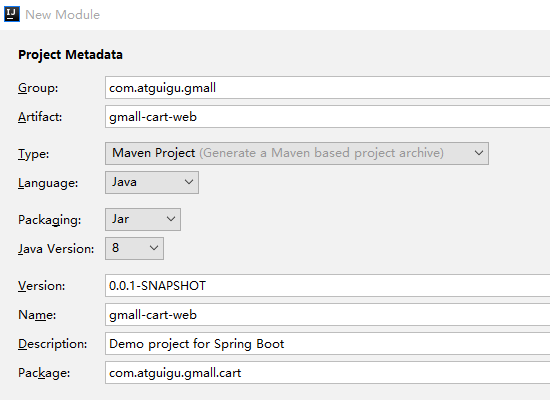
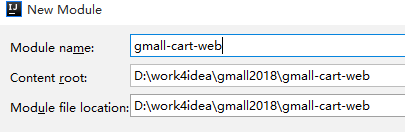
1. 要持久化，保存到数据库中。
2. 利用缓存提高性能。
3. 未登录状态也可以存入购物车，一旦用户登录要进行合并操作。



# 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=8087 spring.thymeleaf.cache=false spring.thymeleaf.mode=LEGACYHTML5  spring.dubbo.application.name=cart-web spring.dubbo.registry.protocol=zookeeper spring.dubbo.registry.address=192.168.67.163:2181 spring.dubbo.base-package=com.atguigu.gmall spring.dubbo.protocol.name=dubbo spring.dubbo.consumer.timeout=100000 spring.dubbo.consumer.check=false |

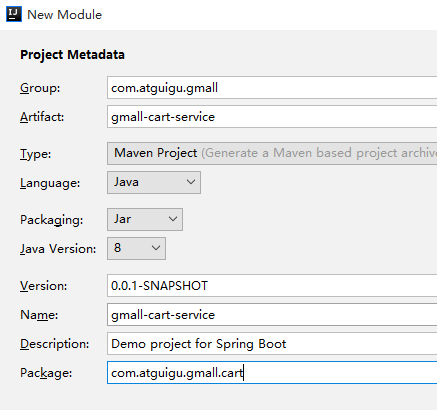
host文件

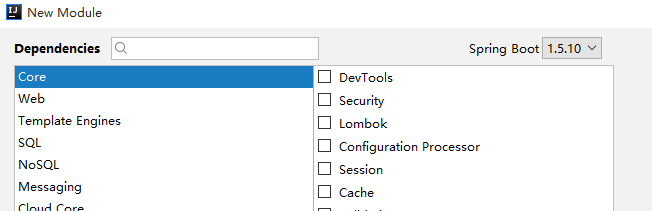
|  |
| --- |
| # gmall  192.168.67.163 cart.gmall.com passport.atguigu.com item.gmall.com list.gmall.com manage.gmall.com www.gmall.com resource.gmall.com |

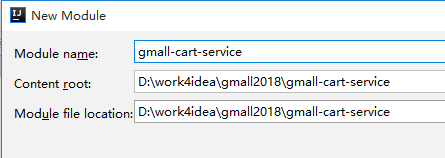
\*GmallCartWebApplication提到跟cart目录同级

(为了让购物车模块能够扫描到util中的工具类)

## 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://59.110.141.236:3306/gmall?characterEncoding=UTF-8 spring.datasource.username=root spring.datasource.password=123456  logging.level.root=debug logging.level.com.atguigu.gmall.usermanage.mapper=debug  server.port=8077  spring.dubbo.application.name=usermanage 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.redis.host=192.168.67.163 spring.redis.port=6379 spring.redis.database=0 |

GmallCartServiceApplication提到跟cart目录同级(让购物车模块能够扫描到redis)

# 三 添加入购物车

## 1 功能解析：

1 通过商品列表的添加按钮可以添加购物车

2 在商品详情页面添加



3 流程，一般在用户未登录是我们也可以让用户使用购物车功能，但是某些系统会要求用户必须登录才能使用购物车功能，比如淘宝

浏览器cookie

未登录

添加

DB

登录

## 2 Redis中的结构

利用Hash结构存储：

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

## 3 gmall-cart-service模块业务方法

### 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。

## 4 gmall-cart-web模块业务方法

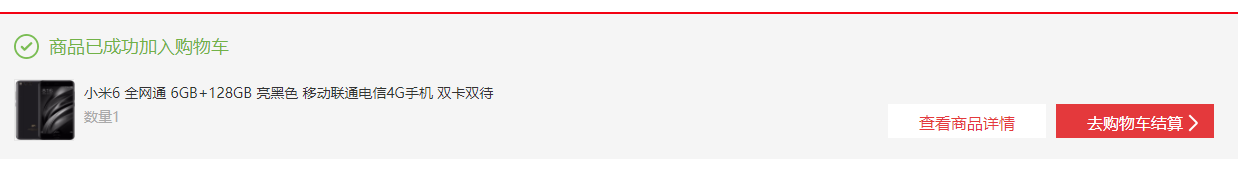
### 思路：

1. 获得参数：skuId 、num
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，要把购物车信息暂存到cookie中。
5. 实现利用cookie保存购物车的方法。

## 5 用来操作cookie的工具类(cookieUtil)

|  |
| --- |
| import javax.servlet.http.Cookie;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import java.io.UnsupportedEncodingException;  import java.net.URLDecoder;  import java.net.URLEncoder;  /\*\*  \* @param  \* @return  \*/  public class CookieUtil {  /\*\*\*  \* 获得cookie中的值，默认为主ip：www.gmall.com  \* @param request  \* @param cookieName  \* @param isDecoder  \* @return  \*/  public static String getCookieValue(HttpServletRequest request, String cookieName, boolean isDecoder) {  Cookie[] cookies = request.getCookies();  if (cookies == null || cookieName == null){  return null;  }  String retValue = null;  try {  for (int i = 0; i < cookies.length; i++) {  if (cookies[i].getName().equals(cookieName)) {  if (isDecoder) {//如果涉及中文  retValue = URLDecoder.decode(cookies[i].getValue(), "UTF-8");  } else {  retValue = cookies[i].getValue();  }  break;  }  }  } catch (UnsupportedEncodingException e) {  e.printStackTrace();  }  return retValue;  }  /\*\*\*  \* 设置cookie的值  \* @param request  \* @param response  \* @param cookieName  \* @param cookieValue  \* @param cookieMaxage  \* @param isEncode  \*/  public static void setCookie(HttpServletRequest request, HttpServletResponse response, String cookieName, String cookieValue, int cookieMaxage, boolean isEncode) {  try {  if (cookieValue == null) {  cookieValue = "";  } else if (isEncode) {  cookieValue = URLEncoder.encode(cookieValue, "utf-8");  }  Cookie cookie = new Cookie(cookieName, cookieValue);  if (cookieMaxage >= 0)  cookie.setMaxAge(cookieMaxage);  if (null != request)// 设置域名的cookie  cookie.setDomain(getDomainName(request));  // 在域名的根路径下保存  cookie.setPath("/");  response.addCookie(cookie);  } catch (Exception e) {  e.printStackTrace();  }  }  /\*\*\*  \* 获得cookie的主域名，本系统为gmall.com，保存时使用  \* @param request  \* @return  \*/  private static final String getDomainName(HttpServletRequest request) {  String domainName = null;  String serverName = request.getRequestURL().toString();  if (serverName == null || serverName.equals("")) {  domainName = "";  } else {  serverName = serverName.toLowerCase();  serverName = serverName.substring(7);  final int end = serverName.indexOf("/");  serverName = serverName.substring(0, end);  final String[] domains = serverName.split("\\.");  int len = domains.length;  if (len > 3) {  // www.xxx.com.cn  domainName = domains[len - 3] + "." + domains[len - 2] + "." + domains[len - 1];  } else if (len <= 3 && len > 1) {  // xxx.com or xxx.cn  domainName = domains[len - 2] + "." + domains[len - 1];  } else {  domainName = serverName;  }  }  if (domainName != null && domainName.indexOf(":") > 0) {  String[] ary = domainName.split("\\:");  domainName = ary[0];  }  System.out.println("domainName = " + domainName);  return domainName;  }  /\*\*\*  \* 将cookie中的内容按照key删除  \* @param request  \* @param response  \* @param cookieName  \*/  public static void deleteCookie(HttpServletRequest request, HttpServletResponse response, String cookieName) {  setCookie(request, response, cookieName, null, 0, false);  }  } |

## 6 返回页面效果



# 四 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中取值。

## 2 业务方法CartServiceImpl提示

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。

## 3 显示购物车列表页面

|  |
| --- |
| <div class="One\_ShopCon">  <ul>  <li th:each="cartInfo:${cartList}">  <div> </div>  <div>  <ol>  <li><input type="checkbox" class="check" th:value="${cartInfo.skuId}" onchange="*checkSku*(this)" th:checked="(${cartInfo.isChecked}=='1')?'true':'false'"/></li>  <li>  <dt><img th:src="${cartInfo.imgUrl}" src="img\shop1.jpg" alt=""></dt>  <dd th:onclick="'toItem('+${cartInfo.skuId}+')'">  <p>  <span th:text="${cartInfo.skuName}"> 商品名称</span>  </p>  </dd>  </li>  <li>  <p class="dj" th:text="'￥'+${cartInfo.skuPrice}">￥钱</p>  </li>  <li>  <p>  <span>-</span>  <span th:text="${cartInfo.skuNum}">5</span>  <span>+</span>  </p>  </li>  <li style="font-weight:bold"><p class="zj" th:text="'￥'+${cartInfo.totalAmount}">￥22995.00</p></li>  <li>  <p>删除</p>  </li>  </ol>  </div>  </li>  </ul> </div> |

## 4 测试效果



# 五、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cookie中的购物车合并到数据库中，并且刷新缓存。

继续购物

用户登录

登陆时要进行额外的并行操作

合并用户的购物车信息

## 1 CartServiceImpl

思路：用数据库中的购物车列表与传递过来的cookie里的购物车列表循环匹配。

能匹配上的数量相加

匹配不上的插入到数据库中。

最后重新加载缓存

|  |
| --- |
| **public** List<CartInfo> mergeToCart(String userId ,List<CartInfo> cartInfoList){  *//查询用户名下的购物车清单* CartInfo cartInfoQuery=**new** CartInfo();  cartInfoQuery.setUserId(userId);  List<CartInfo> cartInfoExistList = **cartInfoMapper**.select(cartInfoQuery);  **for** (CartInfo cartInfo : cartInfoList) {  **for** (CartInfo cartInfoExist : cartInfoExistList) {  **if**( cartInfo.getSkuId().equals(cartInfoExist.getSkuId())){  cartInfoExist.setSkuNum(cartInfoExist.getSkuNum()+cartInfo.getSkuNum());  **cartInfoMapper**.updateByPrimaryKey(cartInfoExist);  }  }  cartInfo.setUserId(userId);  **cartInfoMapper**.insertSelective(cartInfo);  }   List<CartInfo> newCartInfoList = loadCartCache(userId);  **return** newCartInfoList; } |

## 2 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。

|  |
| --- |
| @RequestMapping(**"cartList"**) @LoginRequire(autoRedirect = **false**) **public** String cartList(HttpServletRequest request,HttpServletResponse response){  String userId =(String) request.getAttribute(**"userId"**);  List<CartInfo> cartCookieList = **cartCookieHandler**.getCartList(request);  **if**(userId==**null**){  request.setAttribute(**"cartList"**,cartCookieList );  }**else** {  **if**(cartCookieList==**null**||cartCookieList.size()==0) {  List<CartInfo> cartList = **cartService**.getCartList(userId);  request.setAttribute(**"cartList"**,cartList );  }**else** {  List<CartInfo> cartList = **cartService**.mergeToCart(userId, cartCookieList);  **cartCookieHandler**.deleteCartCookie(request,response);  request.setAttribute(**"cartList"**,cartList );   }  }   **return "cartList"**; } |

# 六、修改购物车

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。

但是在实际的项目中，有时商城的并发量并不高，有时则很高，所以，关于功能是否使用缓存，其实在后台管理系统中，可以加入一个缓存”开关”，平时操作时可以同步到缓存，在并发量高时(如节假日、双十一、618等时候)，可以关闭数据库的同步

关闭数据库和缓存的同步并不是代表不同步，在每日深夜或者一段时间后，可以定时同步一次缓存

## 1 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。

## 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(value = **"checkCart"**,method = RequestMethod.***POST***) @LoginRequire(autoRedirect = **false**) @ResponseBody **public void** checkCart(HttpServletRequest request,HttpServletResponse response){  String skuId = request.getParameter(**"skuId"**);  String userId =(String) request.getAttribute(**"userId"**);  String isChecked =(String) request.getParameter(**"isChecked"**);  **if**(userId!=**null**){  **cartService**.setCheckedCart(userId,skuId,isChecked);  }**else**{  **cartCookieHandler**.setCheckCartCookie(request,response,skuId,isChecked);  }  **return** ; } |

## 4 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。



# 七、点击结算要做的收尾工作

要解决用户在未登录且购物车中有商品的情况下，直接点击结算。

所以不能直接跳到结算页面，要让用户强制登录后，检查cookie并进行合并后再重定向到结算页面

## 1 CartController

|  |
| --- |
| @RequestMapping(**"toTrade"**) @LoginRequire(autoRedirect = **true**) **public** String toTrade(HttpServletRequest request,HttpServletResponse response){  String userId =(String) request.getAttribute(**"userId"**);  List<CartInfo> cartCookieList = **cartCookieHandler**.getCartList(request);  **if**(cartCookieList!=**null**&&cartCookieList.size()>0) {  List<CartInfo> cartList = **cartService**.mergeToCart(userId, cartCookieList);  **cartCookieHandler**.deleteCartCookie(request,response);  }  **return "redirect://order.gmall.com/trade"**; } |