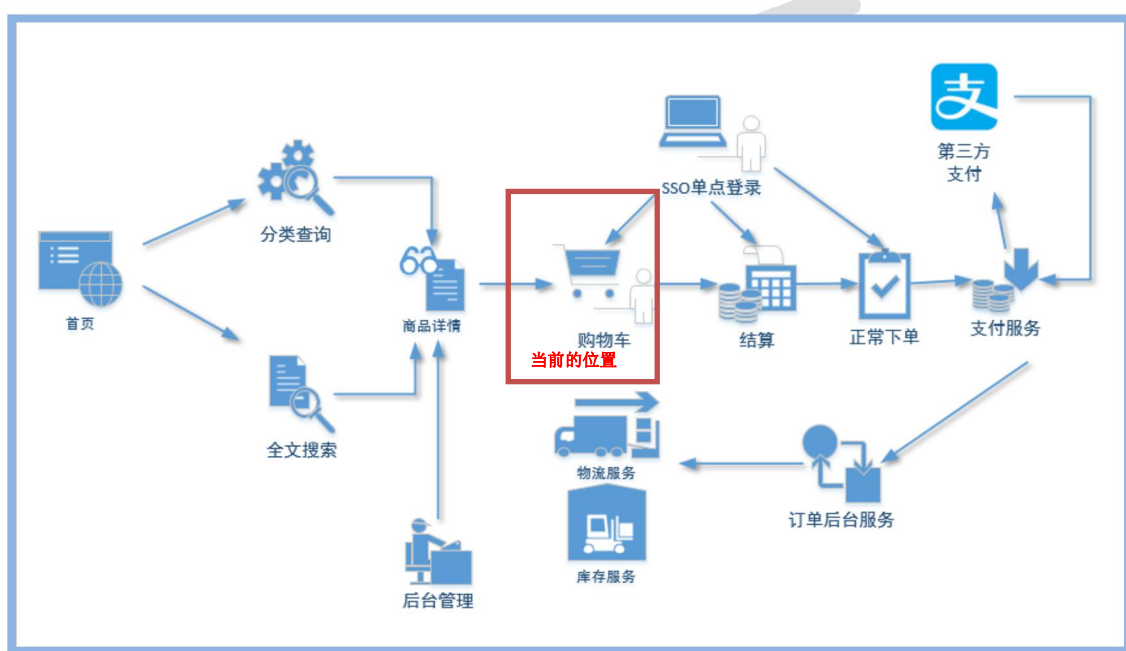


购物车

版本: V 1.0

www.atguigu.com

一、购物车业务简介



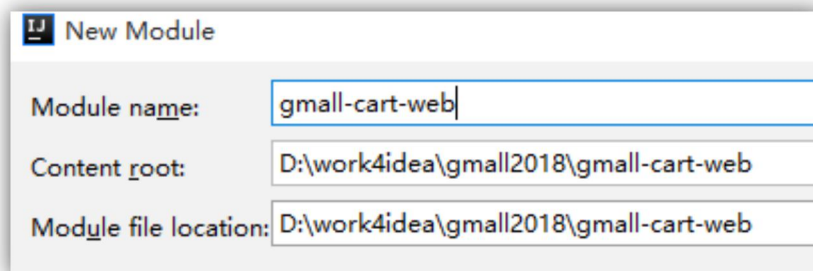
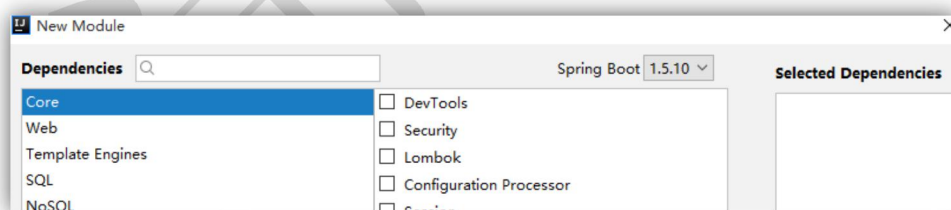
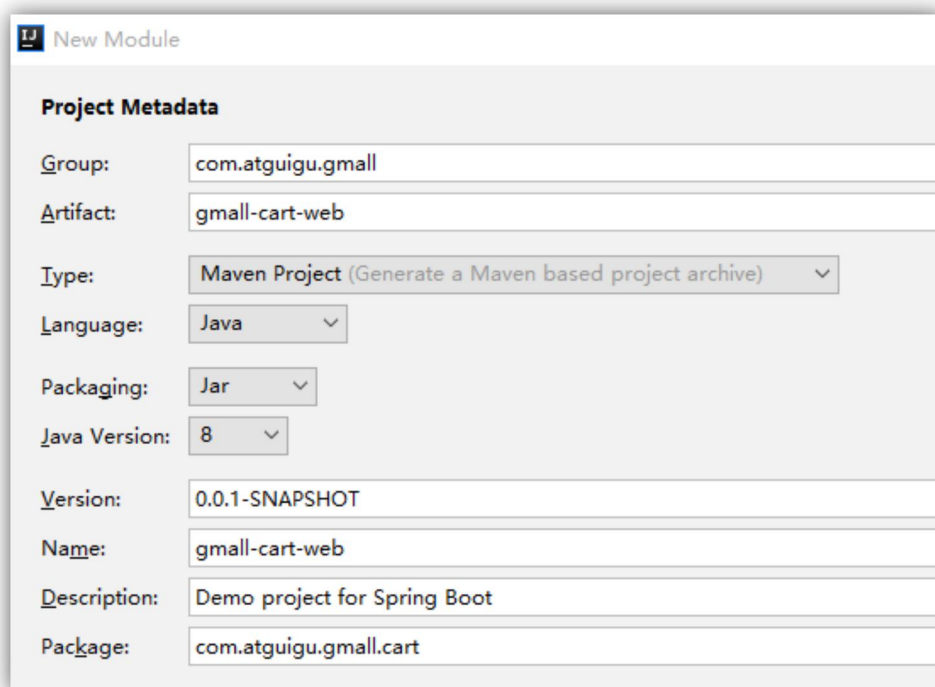
购物车模块要能过存储顾客所选的的商品，记录下所选商品，还要能随时更新，当用户决定购买时，用户可以选择决定购买的商品进入结算页面。

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
```

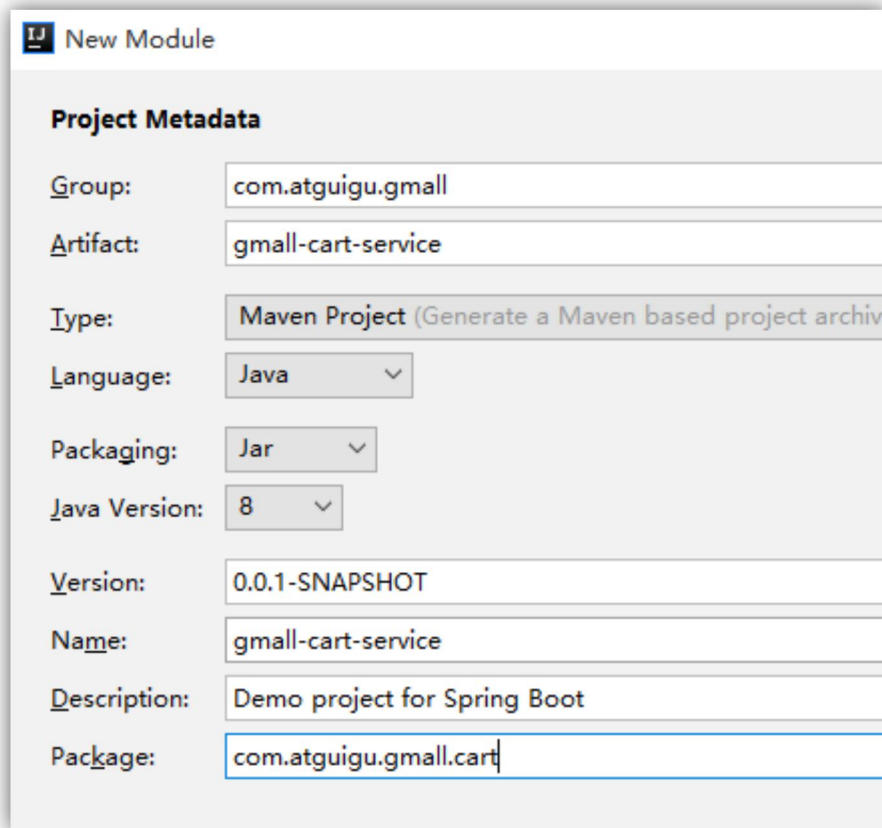
nginx.conf

```
upstream cart.gmall.com{
    server 192.168.67.1:8087;
}
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;
    }
}
```

GmallCartWebApplication 提到跟 cart 目录同级

拷入静态文件和页面

2 small-cart-service



New Module

Project Metadata

Group: com.atguigu.gmall

Artifact: small-cart-service

Type: Maven Project (Generate a Maven based project archive)

Language: Java

Packaging: Jar

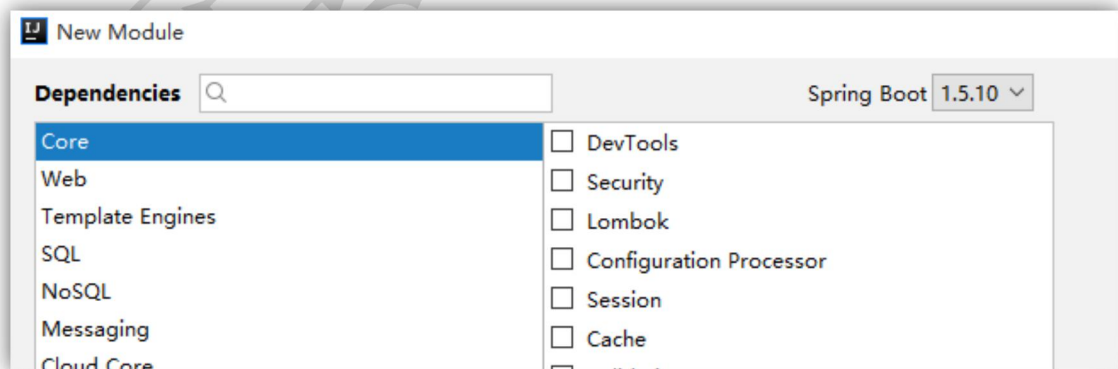
Java Version: 8

Version: 0.0.1-SNAPSHOT

Name: small-cart-service

Description: Demo project for Spring Boot

Package: com.atguigu.gmall.cart

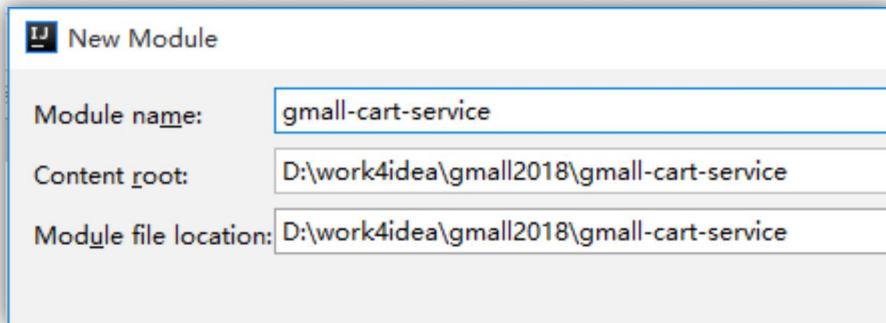


New Module

Dependencies

Spring Boot 1.5.10

Core	<input type="checkbox"/> DevTools
Web	<input type="checkbox"/> Security
Template Engines	<input type="checkbox"/> Lombok
SQL	<input type="checkbox"/> Configuration Processor
NoSQL	<input type="checkbox"/> Session
Messaging	<input type="checkbox"/> Cache
Cloud Core	<input type="checkbox"/> ...



The image shows a 'New Module' dialog box from an IDE. It contains three input fields: 'Module name' with the value 'gmall-cart-service', 'Content root' with the value 'D:\work4idea\gmall2018\gmall-cart-service', and 'Module file location' with the same value 'D:\work4idea\gmall2018\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>
</project>
```

```
<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

三、功能一添加入购物车

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
- 3、返回成功页面

2 设计购物车的数据结构

cart_info(购物车表)	
id	编号
user_id	用户id
sku_id	skuid
cart_price	放入购物车时价格
sku_num	数量
img_url	图片文件
sku_name	sku名称 (冗余)

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;  
  
@Transient  
BigDecimal skuPrice;  
  
@Transient  
String isChecked="0";  
}
```

4 Redis 中的结构

利用 Hash 结构存储:

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。

2 CartServiceImpl

```
public void addToCart(String userId, SkuInfo skuInfo, Integer skuNum){
```

```
String userCartKey="user:"+userId+":cart";
//插入前先检查
CartInfo cartInfoQuery=new CartInfo();
cartInfoQuery.setSkuld(skuInfo.getId());
cartInfoQuery.setUserId(userId);
CartInfo cartInfoExist = cartInfoMapper.selectOne(cartInfoQuery);
if(cartInfoExist!=null) {
    cartInfoExist.setSkuPrice(skuInfo.getPrice());
    cartInfoExist.setSkuNum(cartInfoExist.getSkuNum() + skuNum);
    cartInfoMapper.updateByPrimaryKeySelective(cartInfoExist);
    //更新缓存
    Jedis jedis = redisUtil.getJedis();
    jedis.hset(userCartKey,skuInfo.getId(),JSON.toJSONString(cartInfoExist));
    jedis.close();
}else{
    //插入数据库
    CartInfo cartInfo=new CartInfo();
    cartInfo.setSkuld(skuInfo.getId());
    cartInfo.setCartPrice(skuInfo.getPrice());
    cartInfo.setImgUrl(skuInfo.getSkuDefaultImg());
    cartInfo.setSkuName(skuInfo.getSkuName());
    cartInfo.setUserId(userId);
    cartInfo.setSkuNum(skuNum);
    cartInfo.setSkuPrice(skuInfo.getPrice());
    cartInfoMapper.insertSelective(cartInfo);

    //更新缓存
    Jedis jedis = redisUtil.getJedis();
    jedis.hset("user:"+userId+":cart",skuInfo.getId(),JSON.toJSONString(cartInfo));
    Long ttl = jedis.ttl("user:"+userId+":info");
    jedis.expire(userCartKey, ttl.intValue());
    jedis.close();
}
}
```

6 Web 模块业务方法（gmall-cart-web）

1 思路：

- 1、获得参数：skuld 、 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。

2 CartController

```
@Controller
public class CartController {

    @Reference
    ManageService manageService;

    @Reference
    CartService cartService;

    @Autowired
    CartCookieHandler cartCookieHandler;

    @RequestMapping(value = "addToCart", method = RequestMethod.POST)
    @LoginRequire(autoRedirect = false)
    public String addCart(HttpServletRequest request, HttpServletResponse response){
        String skuld = request.getParameter("skuld");
        Integer num=Integer.parseInt( request.getParameter("num"));

        SkuInfo skuInfo = manageService.getSkuInfo(skuld);
        String userId=(String)request.getAttribute("userId");
        if(userId!=null){
            cartService.addToCart(userId,skuInfo,num);
        }else{
            cartCookieHandler.addToCart(skuInfo, num ,request,response);
        }
        request.setAttribute("skuInfo",skuInfo);
        request.setAttribute("num",num);
        return "success";
    }
}
```

7 实现利用 cookie 保存购物车的 CartCookieHandler

1 思路：

和 service 模块的方法类似

- 1、先查询出来在 cookie 中的购物车，反序列化成列表。

- 2、通过循环比较有没有该商品
- 3、如果有，增加数量
- 4、如果没有，增加商品
- 5、然后把列表反序列化，利用之前最好的 CookieUtil 保存到 cookie 中。

2 CartCookieHandler

```
@Component
public class CartCookieHandler {

    String cartCookieName="CART";

    public void addToCart(SkuInfo cartSkuInfo, Integer num, HttpServletRequest request,
        HttpServletResponse response) {

        try {
            List<CartInfo> cartList = getCartList(request);

            CartInfo cart = null;
            if (cartList != null && cartList.size() > 0) {
                for (CartInfo c : cartList) {
                    // 判断购物车中是否存在该商品
                    if (c.getSkuld().equals(cartSkuInfo.getId())) {
                        cart = c;
                        break;
                    }
                }
            }

            if (cart == null) {
                // 当前的购物车没有该商品
                cart = new CartInfo();
                cart.setSkuld(cartSkuInfo.getId());
                cart.setSkuName(cartSkuInfo.getSkuName());
                // 设置商品主图
                cart.setImgUrl(cartSkuInfo.getSkuDefaultImg());
                cart.setSkuPrice(cartSkuInfo.getPrice());
                cart.setCartPrice(cartSkuInfo.getPrice());
                cart.setSkuNum(num);

                cartList.add(cart);
            } else {
                // 在购物车中存在该商品
                cart.setSkuNum(cart.getSkuNum() + num);
            }
            // 设置购物车的商品，过期时间 7 天
            CookieUtil.setCookie(request, response, cartCookieName, JSON.toJSONString(cartList),
                WebConst.cookieMaxAge,true);
        }
    }
}
```

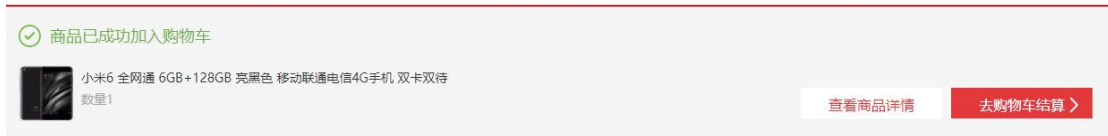
```
} catch (Exception e) {  
    e.printStackTrace();  
}  
  
}  
  
public List<CartInfo> getCartList(HttpServletRequest request) {  
  
    try {  
        String cartListJson = CookieUtil.getCookieValue(request, cartCookieName, true);  
        if (cartListJson != null) {  
            List<CartInfo> cartList = JSON.parseArray(cartListJson, CartInfo.class);  
            return cartList;  
        }  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return new ArrayList<>();  
}  
  
}
```

3 添加成功后的展示页面

success.html

```
<div class="p-item">  
    <div class="p-img">  
        <a href="/javascript:;" target="_blank"></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>
```

8 测试效果



四、功能一展示购物车列表

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 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。

```
public List<CartInfo> getCartList(String userId){  
    //优先从缓存中取值  
    Jedis jedis = redisUtil.getJedis();  
    List<String> skuJsonlist = jedis.hvals("user:" + userId + ":cart");  
    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);
}
//缓存中的值取出来是没有序的 用id 进行排序
cartInfoList.sort(new Comparator<CartInfo>() {
    @Override
    public int compare(CartInfo o1, CartInfo o2) {
        return Long.compare(Long.parseLong(o2.getId()), Long.parseLong(o1.getId()));
    }
});
return cartInfoList;
}else{
    //如果缓存没有就总数据库中加载
    cartInfoList = loadCartCache( userId);
    return cartInfoList;
}
}

//从数据中加载
public List<CartInfo> loadCartCache(String userId){
    //
    List<CartInfo> cartlist = cartInfoMapper.selectCartListWithCurPrice(Long.parseLong(userId));
    if(cartlist==null || cartlist.size()==0){
        return null;
    }
    Jedis jedis = redisUtil.getJedis();
    String userCartKey="user:"+userId+":cart";
    String userInfoKey="user:"+userId+":info";
    Map cartMap =new HashMap(cartlist.size());
    for (CartInfo cartInfo : cartlist) {
        cartMap.put(cartInfo.getSkuld(),JSON.toJSONString(cartInfo));
    }
    jedis.hmset(userCartKey, cartMap);
    Long ttl = jedis.ttl(userInfoKey);
    jedis.expire(userCartKey, ttl.intValue());
    return cartlist;
}
```

CartInfoMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper SYSTEM "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.atguigu.gmall.cart.mapper.CartInfoMapper">
    <select id="selectCartListWithCurPrice" parameterType="long" resultMap="cartMap">
        SELECT c.*,s.price FROM cart_info c
        INNER JOIN sku_info s ON c.sku_id=s.id WHERE c.user_id=#{userId}
```

```
        order by c.id desc
    </select>
    <resultMap id="cartMap" type="com.atguigu.gmall.bean.CartInfo" autoMapping="true">
        <result property="id" column="id" ></result>
        <result property="skuPrice" column="price" ></result>
    </resultMap>
</mapper>
```

CartInfoMapper.java

```
public interface CartInfoMapper extends Mapper<CartInfo> {

    public List<CartInfo> selectCartListWithCurPrice(long userId);
}
```

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
```

3 CartController

```
@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);
    }
}

return "cartList";
}

```

cartCookieHandler 中 getCartList 在前面的做添加购物车时已完成。

4 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.skuld}" onchange="checkSku(this)"
th:checked="(${cartInfo.isChecked}=='1')?'true':'false'"/></li>
                    <li>
                        <dt></dt>
                        <dd th:onclick="toItem('${cartInfo.skuld}+')">
                            <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>
  </li>
</ul>
</div>
```

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 `cookie` 中的购物车合并到数据库中，并且刷新缓存。

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.getSkuld().equals(cartInfoExist.getSkuld())){
                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";  
}
```

在 CartCookieHandler 中加入

```
public void deleteCartCookie(HttpServletRequest request,  
                             HttpServletResponse response){  
    CookieUtil.deleteCookie(request,response, cartCookieName);  
}
```

在 web-util 的 CookieUtil 中增加

```
public static void deleteCookie(HttpServletRequest request,  
                                 HttpServletResponse response, String cookieName) {  
    setCookie(request, response, cookieName, null, 0, false);  
}
```

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 CartServiceImpl

把对应 skuld 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。

```
public void setCheckedCart(String userId,String skuld,String isChecked){
    Jedis jedis = redisUtil.getJedis();
    String userCartKey="user:"+userId+":cart";
    String cartInfoJson = jedis.hget(userCartKey, skuld);
    CartInfo cartInfo = JSON.parseObject(cartInfoJson, CartInfo.class);
    cartInfo.setIsChecked(isChecked);
    String newCartInfoJson=JSON.toJSONString(cartInfo);
    jedis.hset(userCartKey, skuld,newCartInfoJson);

    String cartCheckedKey="user:"+userId+":cartChecked";
    if(isChecked.equals("1")){
        jedis.hset(cartCheckedKey,skuld,newCartInfoJson);
        jedis.expire(cartCheckedKey, jedis.ttl(userCartKey).intValue());
    }else{
        jedis.hdel(cartCheckedKey,skuld );
    }
}
```

合并的时候要把未登录前 cookie 里的勾选附上。

```
public List<CartInfo> mergeToCart(String userId ,List<CartInfo> cartInfoCookieList){
    //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 : cartInfoCookieList) {
        for (CartInfo cartInfoExist : cartInfoExistList) {
            if( cartInfo.getSkuld().equals(cartInfoExist.getSkuld())){
                cartInfoExist.setSkuNum(cartInfoExist.getSkuNum()+cartInfo.getSkuNum());
                cartInfoMapper.updateByPrimaryKey(cartInfoExist);
            }
        }
        cartInfo.setUserId(userId);
        cartInfoMapper.insertSelective(cartInfo);
    }

    List<CartInfo> newCartInfoList = loadCartCache(userId);

    for (CartInfo cartInfo : cartInfoCookieList) {
        if(cartInfo.getIsChecked().equals("1")){
            setCheckedCart( userId,  cartInfo.getSkuld(), "1");
        }
    }

    return newCartInfoList;
}
```

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 skuld = request.getParameter("skuld");
    String userId =(String) request.getAttribute("userId");
    String isChecked =(String) request.getParameter("isChecked");
    if(userId!=null){
        cartService.setCheckedCart(userId,skuld,isChecked);
    }else{
        cartCookieHandler.setCheckCartCookie(request,response,skuld,isChecked);
    }
    return ;
}
```

3 CartCookieHandler(修改 cookie 数据)

从 cookie 的购物车列表中依次匹配 skuld，匹配中则设入 isChecked 标志。

```
public void setCheckCartCookie(HttpServletRequest request,HttpServletResponse response,String skuld,String isChecked) {
    try {
        List<CartInfo> cartList = getCartList(request);
        CartInfo cart = null;
        if (cartList != null && cartList.size() > 0) {
            for (CartInfo c : cartList) {
                // 判断购物车中是否存在该商品
                if (c.getSkuld().equals(skuld)) {
                    cart = c;
                    break;
                }
            }
            cart.setIsChecked(isChecked);
        }

        CookieUtil.setCookie(request, response, cartCookieName, JSON.toJSONString(cartList),
            WebConst.cookieMaxAge, true);
    }
}
```

```
} catch (Exception e) {  
    e.printStackTrace();  
}  
}
```

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。

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

5 测试效果

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";
}
```