第8章 购物车解决方案

学习目标：

- 完成购物车列表功能
- 完成购物车勾选逻辑
- 完成满减优惠金额计算

1. 购物车列表

1.1 需求分析

（1）实现购物车列表的显示。购物车必须登录后才可以访问。

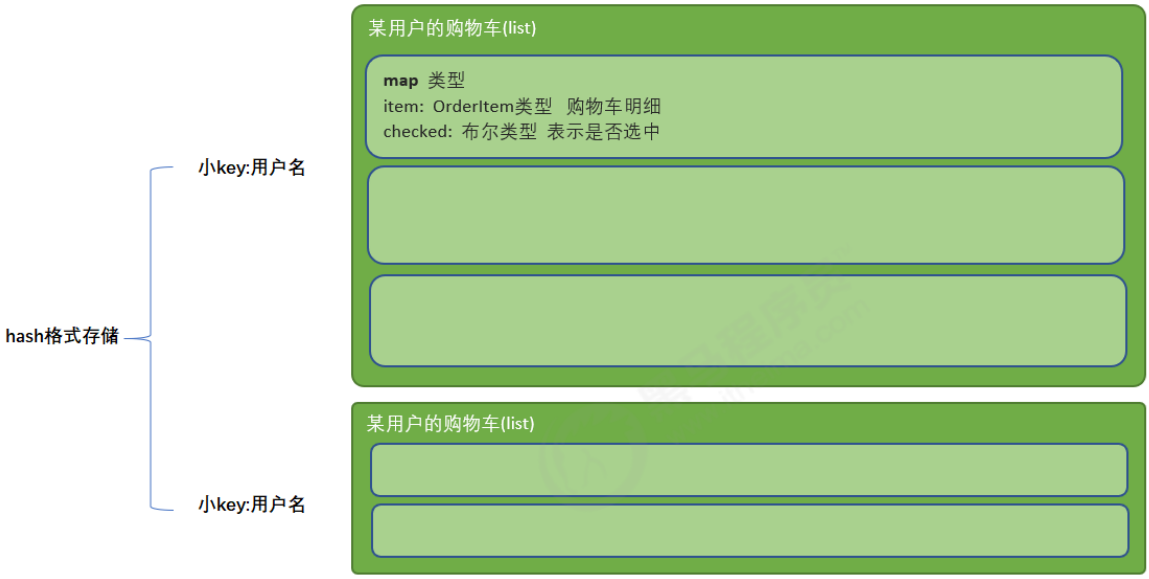
（2）从商品详细页点击“加入购物车按钮”，将商品添加到购物车。

（3）点击购物车列表项中数量的加减按钮实现对数量的修改。

（4）删除购物车中某条记录。

1.2 实现思路

使用redis（hash类型数据）实现购物车的存储。大key为 CacheKey.CART_LIST，当前登录名作为小key。购物车存储结构如下图：



每个用户的购物车就是一个list集合，集合中是map类型的数据，有两个属性，一个是item，表示购物车明细数据，另一个是checked，用于存储购物车是否被选中。

每个用户的购物车类型是 List> 。

购物车列表采用前端渲染（vue.js）的方式。

1.3 后端代码

1.3.1 购物车列表

（1）创建服务接口

```
/**
 * 购物车服务
 */
public interface CartService {

    /**
     * 从redis中提取购物车
     * @param username
     * @return
     */
    public List<Map<String,Object>> findCartList(String username);
}
```

（2）CacheKey枚举增加CART_LIST用于存储购物车列表， 创建服务实现类

```
@Service
public class CartServiceImpl implements CartService {

    @Autowired
    private RedisTemplate redisTemplate;

    @Override
    public List<Map<String, Object>> findCartList(String username) {
        System.out.println("从redis中提取购物车"+username);
        List<Map<String,Object>> cartList = (List<Map<String,Object>>)
redisTemplate.boundHashOps(CacheKey.CART_LIST).get(username);
        if(cartList==null){
            return new ArrayList<>();
        }
        return cartList;
    }
}
```

（3）在qingcheng_portal工程添加CartController

```
@RestController
@RequestMapping("/cart")
public class CartController {

    @Reference
    private CartService cartService;

    /**
     * 从redis中提取购物车
     * @return
     */
    @GetMapping("/findCartList")
    public List<Map<String, Object>> findCartList(){
        String username =
SecurityContextHolder.getContext().getAuthentication().getName();
        List<Map<String, Object>> cartList =
cartService.findCartList(username);
        return cartList;
    }

}
```

1.3.2 添加商品到购物车

实现思路：遍历购物车，如果购物车列表中不存在该商品则添加，存在则累加数量。

（1）CartService接口新增方法定义

```
    /**
     * 添加商品到购物车
     * @param username
     * @param skuId
     * @param num
     */
    public void addItem(String username, String  skuId, Integer num);
```

（2）CartServiceImpl实现此方法

```
    @Reference
    private SkuService skuService;

    @Reference
    private CategoryService categoryService;

    @Override
    public void addItem(String username, String skuId, Integer num) {
        //实现思路:  遍历购物车，如果购物车中存在该商品则累加数量，如果不存在则添加购物车项
        //获取购物车
        List<Map<String, Object>> cartList = findCartList(username);

        boolean flag=false;//是否在购物车中存在
        for( Map map:cartList  ){
            OrderItem orderItem= (OrderItem)map.get("item");
            if(orderItem.getSkuId().equals(skuId)){  //购物车存在该商品
                if(orderItem.getNum()<=0){ //如果数量小于等于0
                    cartList.remove(map);//购物车项删除
                    break;
                }
                int weight = orderItem.getWeight() / orderItem.getNum();//单个商品重量

                orderItem.setNum( orderItem.getNum()+num ); //数量累加
                orderItem.setMoney( orderItem.getNum()*orderItem.getPrice() );//金额计算
                orderItem.setWeight( weight*orderItem.getNum() );//重量计算

                if(orderItem.getNum()<=0){ //如果数量小于等于0
                    cartList.remove(map);//购物车项删除
                }

                flag=true;
                break;
            }
        }
        //如果购物车中没有该商品，则添加

        if(flag==false){
```

```
            Sku sku = skuService.findById(skuId);
            if(sku==null){
                throw new RuntimeException("商品不存在");
            }
            if(!"1".equals(sku.getStatus())){
                throw new RuntimeException("商品状态不合法");
            }
            if(num<=0){  //数量不能为0或负数
                throw new RuntimeException("商品数量不合法");
            }

            OrderItem orderItem=new OrderItem();


            orderItem.setSkuId(skuId);
            orderItem.setSpuId(sku.getSpuId());
            orderItem.setNum(num);
            orderItem.setImage(sku.getImage());
            orderItem.setPrice(sku.getPrice());
            orderItem.setName(sku.getName());
            orderItem.setMoney( orderItem.getPrice()*num );//金额计算
            if(sku.getWeight()==null){
                sku.setWeight(0);
            }
            orderItem.setWeight(sku.getWeight()*num); //重量计算


            //商品分类
            orderItem.setCategoryId3(sku.getCategoryId());
            Category category3 =
(Category)redisTemplate.boundHashOps(CacheKey.CATEGORY).get(sku.getCatego
ryId());
            if(category3==null){
                category3 =
categoryService.findById(sku.getCategoryId()); //根据3级分类id查2级

redisTemplate.boundHashOps(CacheKey.CATEGORY).put(sku.getCategoryId(),
category3 );
            }
            orderItem.setCategoryId2(category3.getParentId());
```

```
            Category category2 =
(Category)redisTemplate.boundHashOps(CacheKey.CATEGORY).get(category3.get
ParentId());
            if(category2==null ){
                category2 =
categoryService.findById(category3.getParentId()); //根据2级分类id查询1级

redisTemplate.boundHashOps(CacheKey.CATEGORY).put(category3.getParentId()
, category2 );
            }
            orderItem.setCategoryId1(category2.getParentId());

            Map map=new HashMap();
            map.put("item",orderItem);
            map.put("checked",true);//默认选中

            cartList.add(map);
        }

redisTemplate.boundHashOps(CacheKey.CART_LIST).put(username,cartList);
    }
```

（3）qingcheng_web_portal的CartController新增方法

```
    /**
     * 添加商品到购物车
     * @param skuId 商品id
     * @param num 数量
     * @return
     */
    @GetMapping("/addItem")
    public Result addItem(String skuId,Integer num){
        String
username=SecurityContextHolder.getContext().getAuthentication().getName()
;
        cartService.addItem(username,skuId,num);
        return new Result();
    }
```

1.4 前端代码

1.4.1 购物车列表

（1）拷贝cart.html到qingcheng_web_portal

（2）cart.html增加js脚本

```
<script src="/js/vue.js"></script>
<script src="/js/axios.js"></script>
<script>
    new Vue({
        el: '#app',
        data(){
            return {
                cartList: {}
            }
        },
        created(){
            this.findCartList();//查询列表
        },
        methods:{
            findCartList(){
                axios.get(`/cart/findCartList.do`).then(response => {
                    this.cartList=response.data;
                });
            }
        }
    })
</script>
```

（3）为购物车的div增加id

```
<div id="app">
```

修改cart.html购物车列表部分代码

```
<div class="cart-list" v-for="cart in cartList">
    <ul class="goods-list yui3-g">
        <li class="yui3-u-1-24">
            <input type="checkbox" name="chk_list"  v-
model="cart.checked" />
        </li>
        <li class="yui3-u-6-24">
            <div class="good-item">
                <div class="item-img"><img :src="cart.item.image" />
</div>
                <div class="item-msg">{{cart.item.title}}</div>
            </div>
        </li>
        <li class="yui3-u-5-24">
            <div class="item-txt"></div>
        </li>
        <li class="yui3-u-1-8"><span class="price">
{{(cart.item.price/100).toFixed(2)}}</span></li>
        <li class="yui3-u-1-8">
            <a  class="increment mins" >-</a>
            <input autocomplete="off" type="text" v-model="cart.item.num"
minnum="1" class="itxt" />
            <a  class="increment plus" >+</a>
        </li>
        <li class="yui3-u-1-8"><span class="sum">
{{(cart.item.price*cart.item.num/100).toFixed(2)}}</span></li>
        <li class="yui3-u-1-8">
            <a href="#none">删除</a><br />
            <a href="#none">移到收藏</a>
        </li>
    </ul>
</div>
```

1.4.2 更改数量与删除

（1）cart.html 增加方法

```
addItem(skuId,num){
    axios.get(`/cart/addItem.do?skuId=${skuId}&num=${num}`).then(response
=> {
        this.findCartList();
    });
}
```

（2）加减号调用方法

```
<a  class="increment mins" @click="addItem(cart.item.skuId,-1)">-</a>
<input autocomplete="off" type="text" v-model="cart.item.num" minnum="1"
class="itxt" />
<a  class="increment plus" @click="addItem(cart.item.skuId,1)">+</a>
```

（3）修改删除链接 。传递数量的相反数就是实现删除的功能。

```
<a @click="addItem(cart.item.skuId,-cart.item.num)" >删除</a><br />
```

1.5 与商品详细页对接

（1）在CartController新增方法

```
    @RequestMapping("/buy")
    public void buy(HttpServletResponse response,String skuId,Integer
num) throws Exception{
        String
username=SecurityContextHolder.getContext().getAuthentication().getName()
;
        cartService.addItem(username,skuId,num);
        response.sendRedirect("/cart.html");
    }
```

这个方法实现了添加商品到购物车并重定向到购物车页面的功能。

在浏览器上测试 http://localhost:9102/cart/buy.do?skuId=100000030326&num=1

（2）修改item.html模板 ，js代码如下：

```
    new Vue({
        el:'#app',
        data(){
            return {
                skuId:/*[[${sku.id}]]*/,
                price:0,
                num:1
            }
        },
        created(){
            //读取价格
            axios.get('http://localhost:9102/sku/price.do?
id='+this.skuId).then(response=>{
                this.price=(response.data/100).toFixed(2);
            })
        },
        methods:{
            addNum(num){
                this.num=this.num+num;
                if(this.num<=0){
                  this.num=1;
                }
            },
            toCartList(){
                //添加商品到购物车
                location.href=`http://localhost:9102/cart/buy.do?
skuId=${this.skuId}&num=${this.num}`;
            }

        }

    });
```

html部分

```
<div class="summary-wrap">
    <div class="fl title">
        <div class="control-group">
            <div class="controls">
                <input autocomplete="off"  v-model="num"  minnum="1"
class="itxt" />
                <a @click="addNum(1)" class="increment plus">+</a>
                <a @click="addNum(-1)" class="increment mins">-</a>
            </div>
        </div>
    </div>
    <div class="fl">
        <ul class="btn-choose unstyled">
            <li>
                <a @click="toCartList()"  class="sui-btn  btn-danger
addshopcar">加入购物车</a>
            </li>
        </ul>
    </div>
</div>
```

2. 购物车勾选逻辑

2.1 需求分析

点击购物车前面的复选框即选中该购物车项，选中后需要保存该选中状态。

购物车下方显示选择的商品数量以及合计金额。

点击删除按钮，实现对选中购物车的删除。

2.2 代码实现

2.2.1 保存勾选状态

实现思路：在页面点击复选框后向后端提交ajax请求，提交的参数为当前行的skuId和选中状态，后端接受到请求后更新redis中的购物车状态。

代码实现：

（1）CartService新增方法定义

```
/**
 * 更新选中状态
 * @param username
 * @param skuId
 * @param checked
 */
public boolean updateChecked(String username, String skuId,boolean
checked);
```

（2）CartServiceImpl实现此方法

```
@Override
public boolean updateChecked(String username, String skuId, boolean
checked) {
    //获取购物车
    List<Map<String,Object>> cartList= findCartList(username);
    //判断缓存中是含有已购物商品
    boolean isOk=false;
    for (Map map:cartList) {
        OrderItem orderItem=(OrderItem)map.get("item");
        if(orderItem.getSkuId().equals(skuId)){
            map.put("checked",checked);
            isOk=true;//执行成功
            break;
        }
    }
    if(isOk){

redisTemplate.boundHashOps(CacheKey.CART_LIST).put(username,cartList);//
存入缓存
    }
    return isOk;
}
```

（3）CartController新增方法

```
/**
 * 更改购物车项选中状态
 * @param skuId
 * @param checked
 * @return
 */
@GetMapping("/updateChecked")
public Result updateChecked(String skuId, boolean checked){
    String username =
SecurityContextHolder.getContext().getAuthentication().getName();
    cartService.updateChecked(username,skuId,checked);
    return new Result();
}
```

（4）cart.html新增方法

```
updateChecked(skuId,checked){
    axios.get(`/cart/updateChecked.do?
skuId=${skuId}&checked=${checked}`).then(
      response => {});
}
```

（5）修改cart.html，为复选框绑定点击事件

```
<input type="checkbox" name="chk_list"  v-model="cart.checked"
@click="updateChecked(cart.item.skuId,!cart.checked)"  />
```

2.2.2 合计金额计算

实现思路：使用前端的代码实现合计金额

代码实现：

（1）修改cart.html，新增属性

```
totalNum:0,//总数量
totalMoney:0//总金额
```

（2）新增方法

```
count(){
    this.totalNum=0;  //数量
    this.totalMoney=0; //金额
    //循环购物车
    for(let i=0;i<this.cartList.length;i++){
        if(this.cartList[i].checked){ //判断如果选中，数量和金额累加
            this.totalNum+= this.cartList[i].item.num;
            this.totalMoney+= this.cartList[i].item.money;
        }
    }
}
```

（3）在findCartList和updateChecked方法的回调处调用count方法。

（4）修改数量和金额的显示

```
<div class="chosed">已选择<span>{{totalNum}}</span>件商品</div>
<div class="sumprice">
    <span><em>总价（不含运费）：</em><i class="summoney">¥
{{(totalMoney/100).toFixed(2)}}</i></span>
    <span><em>已节省：</em><i></i></span>
</div>
```

2.2.3 删除勾选项

实现思路：使用 stream 获取没选择的购物车列表，覆盖redis 现有的购物车。

代码实现：

（1）CartService新增方法

```
    /**
     * 删除选中的购物车
     * @param username
     */
    public void deleteCheckedCart(String username);
```

（2）CartServiceImpl实现方法

```
    @Override
    public void deleteCheckedCart(String username) {
        //获取未选中的购物车
        List<Map<String, Object>> cartList =
findCartList(username).stream()
                .filter(cart -> (boolean) cart.get("checked") == false)
                .collect(Collectors.toList());

redisTemplate.boundHashOps(CacheKey.CART_LIST).put(username,cartList);//
存入缓存
    }
```

（3）CartController新增方法

```
    /**
     * 删除选中的购物车
     */
    @GetMapping("/deleteCheckedCart")
    public Result deleteCheckedCart(){
        String username =
SecurityContextHolder.getContext().getAuthentication().getName();
        cartService.deleteCheckedCart(username);
        return new Result();
    }
```

（4）删除选中的购物车

```
deleteCheckedCart(){ //删除选择的商品
    axios.get(`/cart/deleteCheckedCart.do`).then(response => {
        this.findCartList();
    });
}
```

（5）删除选中的商品

```
<a @click="deleteCheckedCart()">删除选中的商品</a>
```

3. 满减优惠金额计算

3.1 需求分析

我们采用满减的规则进行优惠金额的计算。例如：某品类满XX元减XX元 。

我们需要对购物车的商品按照品类进行分组，某个品类如果达到了优惠设置的消费额度，则按照设置的优惠金额进行减免。

优惠计算有翻倍和不翻倍两种。例如 ，满100元减30元。如果消费者花了320元，不翻倍是减30元，翻倍是减90元。

3.2 实现思路

首先我们先看一下满减优惠规则表 tb_preferential

字段名称	字段含义	字段类型	字段长度	备注
id	ID	INT		
buy_money	消费金额	INT		
pre_money	优惠金额	INT		
category_id	品类ID	BIGINT		
start_time	活动开始日期	DATE		
end_time	活动截至日期	DATE		
state	状态	VARCHAR		1 有效 0无效
type	类型	VARCHAR		1不翻倍 2翻倍

实现思路：

（1）创建一个方法，可以根据品类id和消费金额计算优惠金额

（2）对购物车按商品品类进行分组，统计每个品类的消费额，调用第1步的方法得到优惠金额，将每个品类的优惠金额累加起来，得到总的优惠金额。

3.3 代码实现

3.3.1 根据分类和消费额查询优惠金额

（1）PreferentialService新增方法定义

```
/**
 * 根据分类和消费额查询优惠金额
 * @param categoryId
 * @param money
 * @return
 */
public int findPreMoneyByCategoryId(Long categoryId,int money);
```

（2）PreferentialServiceImpl实现此方法

```
@Override
public int findPreMoneyByCategoryId(Long categoryId, int money) {

    Example example=new Example(Preferential.class);
    Example.Criteria criteria = example.createCriteria();
    criteria.andEqualTo("state","1");//状态
    criteria.andEqualTo("categoryId",categoryId);
    criteria.andLessThanOrEqualTo("buyMoney",money);//小于等于优惠金额
    criteria.andGreaterThanOrEqualTo("endTime",new Date());//截至日期大于等于当前日期
    criteria.andLessThanOrEqualTo("startTime",new Date());//开始日期小于等于当前日期
    example.setOrderByClause("buy_money desc");//按照购买金额降序排序
    List<Preferential> preferentials = preferentialMapper.selectByExample(example);
    if(preferentials.size()>=1){
        Preferential preferential = preferentials.get(0);
        if("1".equals(preferential.getType())){//不翻倍
            return preferential.getPreMoney();//返回优惠的金额
        }else{ //翻倍
            int multiple=  money/preferentials.get(0).getBuyMoney();
            return preferential.getPreMoney()*multiple;
        }
    }else{
        return 0;
    }
}
```

3.3.2 计算当前购物车优惠金额

（1）修改CartService，新增计算优惠金额的方法定义

```
/**
 * 计算当前选中的购物车的优惠金额
 * @param
 * @return
 */
public int preferential(String username);
```

（2）CartServiceImpl实现该方法

```
@Autowired
private PreferentialService preferentialService;

@Override
public int preferential(String username) {
    //获取选中的购物车  List<OrderItem>  List<Map>
    List<OrderItem> orderItemList = findCartList(username).stream()
            .filter(cart -> (boolean) cart.get("checked") == true)
            .map(cart -> (OrderItem) cart.get("item"))
            .collect(Collectors.toList());
    //按分类聚合统计每个分类的金额    group by
    Map<Integer, IntSummaryStatistics> cartMap = orderItemList.stream()
            .collect(Collectors.groupingBy(OrderItem::getCategoryId3,
Collectors.summarizingInt(OrderItem::getMoney)));
    //循环结果，统计每个分类的优惠金额，并累加
    int allPreMoney=0;//累计优惠金额
    for( Integer categoryId:  cartMap.keySet() ){
        // 获取品类的消费金额
        int money = (int)cartMap.get(categoryId).getSum();
        int preMoney =
preferentialService.findPreMoneyByCategoryId(categoryId, money); //获取优
惠金额
        System.out.println("分类："+categoryId+"  消费金额："+ money+  " 优
惠金额：" + preMoney );
        allPreMoney+=   preMoney;
    }
    return allPreMoney;
}
```

（3）CartController新增方法

```
/**
 * 计算当前选中的购物车的优惠金额
 * @param
 * @return
 */
@GetMapping("/preferential")
public Map preferential(){
    String username =
SecurityContextHolder.getContext().getAuthentication().getName();
    int preferential = cartService.preferential(username);
    Map map=new HashMap();
    map.put("preferential",preferential);
    return map;
}
```

（4）修改cart.html，新增属性

```
preferential:0//优惠金额
```

修改count方法，追加以下代码，实现优惠金额的计算

```
//计算优惠金额
axios.get(`/cart/preferential.do`).then(response => {
    this.preferential= response.data.preferential;
    this.totalMoney=this.totalMoney-this.preferential;//最后的总金额
});
```