**品优购电商系统开发**

**第18章**

**购物车解决方案**

传智播客.黑马程序员

# 0.商品详细页跨域请求

## 0.1需求分析

从商品详细页点击“加入购物车”按钮，将当前商品加入购物车，并跳转到购物车页面。

## 0.2 JS跨域请求

这里说的js跨域是指通过js在不同的域之间进行数据传输或通信，比如用ajax向一个不同的域请求数据，或者通过js获取页面中不同域的框架中(iframe)的数据。只要协议、域名、端口有任何一个不同，都被当作是不同的域。

## 0.3跨域调用测试

修改page-service 的itemController.js ，引入$http服务，修改addToCart方法

|  |
| --- |
| //添加商品到购物车  $scope.addToCart=**function**(){  $http.get('http://localhost:8080/cart/addGoodsToCartList.do?itemId='  + $scope.sku.id +'&num='+$scope.num).success(  **function**(response){  **if**(response.flag){  location.href='http://localhost:8080/cart.html';//跳转到购物车页面  }**else**{  alert(response.message);  }  }  );  } |

测试后发现无法跨域调用

XMLHttpRequest cannot load

http://localhost:9107/cart/addGoodsToCartList.do?itemId=112344&num=1.

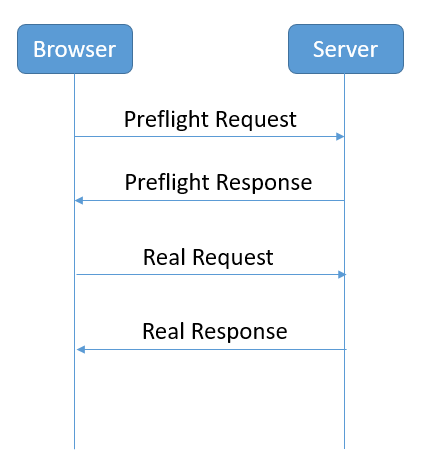
No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:9100' is therefore not allowed access. The response had HTTP status code 400.

## 0.4跨域解决方案CORS

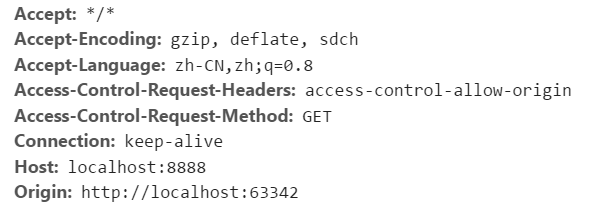
CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。CORS需要浏览器和服务器同时支持。目前，所有浏览器都支持该功能，IE浏览器不能低于IE10。

它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。

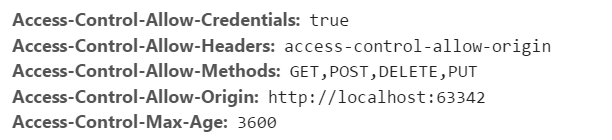
请求过程如下图:



Preflight Request：



然后服务器端给我们返回一个Preflight Response



下面我们就开始动手，让我们的购物车工程能够接收跨域请求

（1）首先修改portal-web 的CartController.java的addGoodsToCartList方法，添加下面两句代码, 下面地址配置的是page-service项目的地址

|  |
| --- |
| response.setHeader("Access-Control-Allow-Origin", "http://localhost:9003");  response.setHeader("Access-Control-Allow-Credentials", "true"); |

**Access-Control-Allow-Origin**

Access-Control-Allow-Origin是[**HTML5**](http://lib.csdn.net/base/html5)中定义的一种解决资源跨域的策略。

他是通过服务器端返回带有Access-Control-Allow-Origin标识的Response header，用来解决资源的跨域权限问题。

 使用方法，在response添加 Access-Control-Allow-Origin，例如

|  |
| --- |
| Access-Control-Allow-Origin:www.google.com |

也可以设置为 \* 表示该资源谁都可以用

（2）修改page-service的itemController.js

|  |
| --- |
| //添加商品到购物车  $scope.addToCart=**function**(){  $http.get('http://localhost:9107/cart/addGoodsToCartList.do?itemId='  + $scope.sku.id +'&num='+$scope.num,{'withCredentials':**true**}).success(  **function**(response){  .......  }  );  } |

调用测试，可以实现跨域了。

CORS请求默认不发送Cookie和HTTP认证信息。如果要把Cookie发到服务器，一方面要服务器同意，指定Access-Control-Allow-Credentials字段。另一方面，开发者必须在AJAX请求中打开withCredentials属性。否则，即使服务器同意发送Cookie，浏览器也不会发送。或者，服务器要求设置Cookie，浏览器也不会处理。

## 0.5 SpringMVC跨域注解

springMVC的版本在4.2或以上版本，可以使用注解实现跨域, 我们只需要在需要跨域的方法上添加注解@CrossOrigin即可, 下面地址配置的是page-service项目的地址

|  |
| --- |
| @CrossOrigin(origins="http://localhost:8088",allowCredentials="true") |

allowCredentials="true" 可以缺省

# 1.购物车需求分析与解决方案

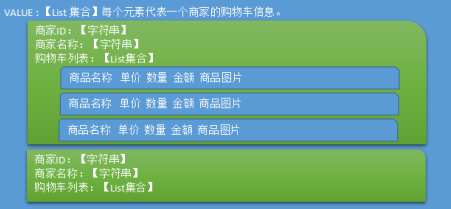
## 1.1需求分析

用户在商品详细页点击加入购物车，提交商品SKU编号和购买数量，添加到购物车。购物车展示页面如下：



## 1.2购物车实体类

购物车数据的存储结构如下：



在pojo项目的cn.itcast.core.pojogroup中创建购物车实体类

|  |
| --- |
| **public** **class** Cart **implements** Serializable{  **private** String sellerId;//商家ID  **private** String sellerName;//商家名称  **private** List<OrderItem> orderItemList;//购物车明细  //getter and setter ......  } |

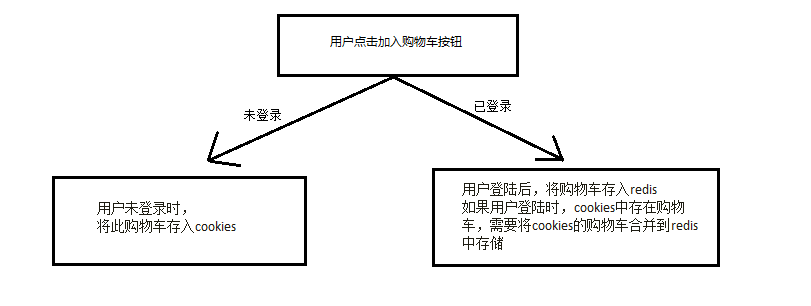
这个类是对每个商家的购物车进行的封装

## 1.3实现思路

用户未登录时，将此购物车存入cookies

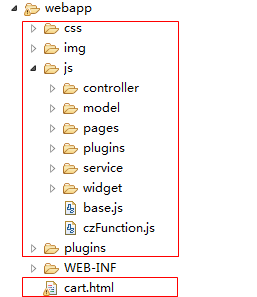
用户登陆后，将购物车存入redis

如果用户登陆时，cookies中存在购物车，需要将cookies的购物车合并到redis中存储.



## 1.4工程搭建

1. 搭建buyer-service工程结构, 并配置好配置文件
2. 在portal-web工程中配置, springsecurity 、CAS 等。添加web.xml 与spring配置文件（参照其他web工程），拷贝UserDetailServiceImpl.java , 拷贝页面资源



# 2.Cookie存储购物车

## 2.1需求分析

使用cookie存储购物车数据。

## 2.3服务实现层

代码实现 buyer-service 工程创建CartServiceImpl.java

|  |
| --- |
| /\*\*  \* 购物车管理  \* **@author** lx  \*  \*/  @Service  @Transactional  **public** **class** CartServiceImpl **implements** CartService {  //根据ItemID查询    @Autowired  **private** ItemDao itemDao;  **public** Item findItemById(Long id){  **return** itemDao.selectByPrimaryKey(id);  }  //将购物车集合装满  **public** List<Cart> findCartList(List<Cart> cartList){  **for** (Cart cart : cartList) {  List<OrderItem> orderItemList = cart.getOrderItemList();  **for** (OrderItem orderItem : orderItemList) {  Item item = itemDao.selectByPrimaryKey(orderItem.getItemId());  orderItem.setGoodsId(item.getGoodsId());  orderItem.setPicPath(item.getImage());  orderItem.setPrice(item.getPrice());  orderItem.setTitle(item.getTitle());  orderItem.setTotalFee(**new** BigDecimal(orderItem.getPrice().floatValue()\*orderItem.getNum()));  cart.setSellerName(item.getSeller());  }  }  **return** cartList;  }  } |

## 2.4后端控制层

实现思路：

（1）从cookie中取出购物车

（2）向购物车添加商品

（3）将购物车存入cookie

portal-web工程新建CartController.java

|  |
| --- |
| /\*\*  \* 购物车管理  \*  \* **@author** lx  \*  \*/  @RestController  @RequestMapping("/cart")  **public** **class** CartController {  @Reference  **private** CartService cartService;  // 加入购物车  @RequestMapping("/addGoodsToCartList")  @CrossOrigin(origins = "http://localhost:9004",allowCredentials = "true")  **public** Result addGoodsToCartList(HttpServletRequest request, HttpServletResponse response, Long itemId, Integer num) {    **try** {  List<Cart> cartList = **null**;  //未登陆时  Cookie[] cookies = request.getCookies();  **if** (**null** != cookies && cookies.length > 0) {  **for** (Cookie cookie : cookies) {  **if**("CART".equals(cookie.getName())){  cartList = JSON.*parseArray*(cookie.getValue(), Cart.**class**);  } }  }  **if** (**null** == cartList) {  cartList = **new** ArrayList<Cart>();  }  // 追加当前款  Item item = cartService.findItemById(itemId);  **if** (cartList.size() > 0) {  Boolean flag = **true**;  // 根据item的ID查询Item对象  **for** (Cart cart : cartList) {  // 判断商家ID  **if** (cart.getSellerId().equals(item.getSellerId())) {  Boolean f = **true**;  List<OrderItem> orderItemList = cart.getOrderItemList();  **for** (OrderItem orderItem : orderItemList) {  **if** (orderItem.getItemId().equals(itemId)) {  // 相同款 不加新款 只加数量  orderItem.setNum(orderItem.getNum() + num);  f = **false**;  }  }  **if**(f){  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  }  flag = **false**;  }  }  **if**(flag){  // 追加当前款  Cart cart = **new** Cart();  List<OrderItem> orderItemList = **new** ArrayList<>();  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  cart.setOrderItemList(orderItemList);  cart.setSellerId(item.getSellerId());  cartList.add(cart);  }  } **else** {  // 追加当前款  Cart cart = **new** Cart();  List<OrderItem> orderItemList = **new** ArrayList<>();  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  cart.setOrderItemList(orderItemList);  cart.setSellerId(item.getSellerId());  cartList.add(cart);  }  //保存到Cookie中  Cookie cookie = **new** Cookie("CART",JSON.*toJSONString*(cartList));  cookie.setPath("/");  cookie.setMaxAge(60\*60\*24);  response.addCookie(cookie);    **return** **new** Result(**true**,"加入购物车成功");  } **catch** (Exception e) {  // **TODO**: handle exception  **return** **new** Result(**true**,"加入购物车失败");  }    }  // 去购物车结算  @RequestMapping("/findCartList")  **public** List<Cart> findCartList(HttpServletRequest request, HttpServletResponse response){  List<Cart> cartList = **null**;  // 不登陆  Cookie[] cookies = request.getCookies();  **if** (**null** != cookies && cookies.length > 0) {  **for** (Cookie cookie : cookies) {  **if**("CART".equals(cookie.getName())){  cartList = JSON.*parseArray*(cookie.getValue(), Cart.**class**);  }  }  }  //如果购物车中有商品 将商品的信息装满  **if**(**null** != cartList && cartList.size() > 0){  cartList = cartService.findCartList(cartList);  }  **return** cartList;  } |

**浏览器测试：**

查看购物车：http://localhost:8080/cart/findCartList.do

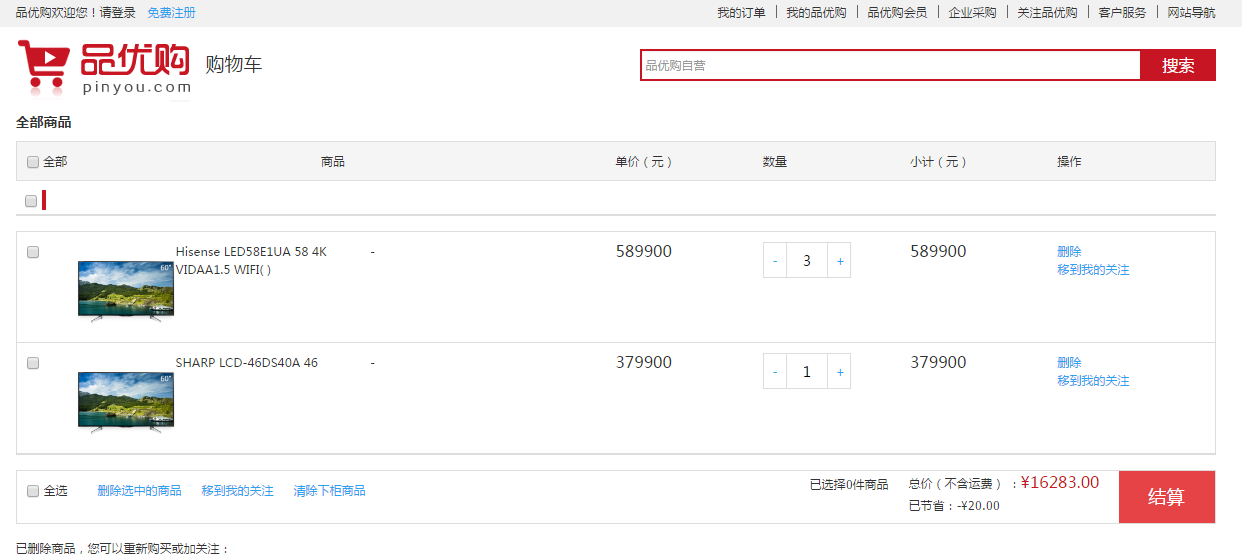
添加商品到购物车 ：

http://localhost:8080/cart/addGoodsToCartList.do?itemId=1369280&num=100

# 3.购物车前端代码

## 3.1需求分析

实现购物车页面的展示与相关操作



可以实现购物车列表、数量的增减与移除以及合计数统计

## 3.2购物车列表

### 3.2.1前端服务层

portal-web增加cartService.js

|  |
| --- |
| //购物车服务层  app.service('cartService',**function**($http){  //购物车列表  **this**.findCartList=**function**(){  **return** $http.get('cart/findCartList.do');  }  }); |

### 3.2.2前端控制层

portal-web增加cartController.js

|  |
| --- |
| //购物车控制层  app.controller('cartController',**function**($scope,cartService){  //查询购物车列表  $scope.findCartList=**function**(){  cartService.findCartList().success(  **function**(response){  $scope.cartList=response;  }  );  }  }); |

### 3.2.3页面

修改cart.html 引入js

|  |
| --- |
| <script type=*"text/javascript"* src=*"plugins/angularjs/angular.min.js"*> </script>  <script type=*"text/javascript"* src=*"js/base.js"*> </script>  <script type=*"text/javascript"* src=*"js/service/cartService.js"*> </script>  <script type=*"text/javascript"* src=*"js/controller/baseController.js"*> </script>  <script type=*"text/javascript"* src=*"js/controller/cartController.js"*> </script> |

添加相关指令，指定控制器，调用初始化方法

|  |
| --- |
| <body ng-app=*"pinyougou"* ng-controller=*"cartController"* ng-init=*"findCartList()"*> |

循环显示购物车列表

|  |
| --- |
| <div class=*"cart-item-list"* ng-repeat=*"cart in cartList"*>  <div class=*"cart-shop"*>  <input type=*"checkbox"* name=*""* id=*""* value=*""* />  <span class=*"shopname self"*>{{cart.sellerName}}【商家ID:{{cart.sellerId}}】</span>  </div>  <div class=*"cart-body"*>  <div class=*"cart-list"* ng-repeat=*"orderItem in cart.orderItemList"*>  <ul class=*"goods-list yui3-g"*>  <li class=*"yui3-u-1-24"*>  <input type=*"checkbox"* name=*""* id=*""* value=*""* />  </li>  <li class=*"yui3-u-11-24"*>  <div class=*"good-item"*>  <div class=*"item-img"*><img src=*"{{orderItem.picPath}}"* /></div>  <div class=*"item-msg"*>  {{orderItem.title}}  </div>  </div>  </li>  <li class=*"yui3-u-1-8"*><span class=*"price"*>{{orderItem.price.toFixed(2)}}</span></li>  <li class=*"yui3-u-1-8"*>  <a href=*"javascript:void(0)"* class=*"increment mins"*>-</a>  <input autocomplete=*"off"* type=*"text"* ng-model=*"orderItem.num"* minnum=*"1"* class=*"itxt"* />  <a href=*"javascript:void(0)"* class=*"increment plus"*>+</a>  </li>  <li class=*"yui3-u-1-8"*><span class=*"sum"*>{{orderItem.totalFee.toFixed(2)}}</span></li>  <li class=*"yui3-u-1-8"*>  <a href=*"#none"*>删除</a><br />  <a href=*"#none"*>移到我的关注</a>  </li>  </ul>  </div>  </div>  </div> |

## 3.3购物车数量增减与移除

### 3.3.1前端服务层

portal-web的 cartService.js

|  |
| --- |
| //添加商品到购物车  **this**.addGoodsToCartList=**function**(itemId,num){  **return** $http.get('cart/addGoodsToCartList.do?itemId='+itemId+'&num='+num);  } |

### 3.3.2前端控制层

portal-web的 cartController.js

|  |
| --- |
| //添加商品到购物车  $scope.addGoodsToCartList=**function**(itemId,num){  cartService.addGoodsToCartList(itemId,num).success(  **function**(response){  **if**(response.flag){  $scope.findCartList();//刷新列表  }**else**{  alert(response.message);//弹出错误提示  }  }  );  } |

### 3.3.3页面

修改portal-web的cart.html 实现数量增减

|  |
| --- |
| <li class=*"yui3-u-1-8"*>  <a href=*"javascript:void(0)"* ng-click="addGoodsToCartList(or*derItem.itemId,-1)"* class=*"increment mins"*>-</a>  <input autocomplete=*"off"* type=*"text"* ng-model=*"orderItem.num"* minnum=*"1"* class=*"itxt"* />  <a href=*"javascript:void(0)"* ng-click="addGoodsToCartList(*orderItem.itemId,1)"* class=*"increment plus"*>+</a> </li> |

实现删除功能

|  |
| --- |
| <a href=*"#none"* ng-click=*"*addGoodsToCartList*(orderItem.itemId,-orderItem.num)"* >删除</a><br /> |

## 3.4合计数

### 3.4.1前端服务层

修改cartService.js

|  |
| --- |
| //求合计  **this**.sum=**function**(cartList){  **var** totalValue={totalNum:0, totalMoney:0.00 };//合计实体  **for**(**var** i=0;i<cartList.length;i++){  **var** cart=cartList[i];  **for**(**var** j=0;j<cart.orderItemList.length;j++){  **var** orderItem=cart.orderItemList[j];//购物车明细  totalValue.totalNum+=orderItem.num;  totalValue.totalMoney+= orderItem.totalFee;  }  }  **return** totalValue;  } |

### 3.4.2前端控制层

修改cartController.js ，调用服务层方法

|  |
| --- |
| //查询购物车列表  $scope.findCartList=**function**(){  cartService.findCartList().success(  **function**(response){  $scope.cartList=response;  $scope.totalValue=cartService.sum($scope.cartList);//求合计数  }  );  } |

### 3.4.3页面

|  |
| --- |
| <div class=*"chosed"*>已选择<span>{{totalValue.totalNum}}</span>件商品</div>  <div class=*"sumprice"*>  <span><em>总价（不含运费） ：  </em><i class=*"summoney"*>¥{{totalValue.totalMoney}}</i></span>  </div> |

# 4.Redis存储购物车

## 4.1需求分析

1. 判断当前用户是否登陆

1.1 如果未登录采用Cookie存储

1.2 如果登录则采用Redis存储

2. 登录后要进行Cookie购物车与Redis购物车的合并操作

3. 清除Cookie中的购物车。

## 4.2获取当前登录人账号

### 4.2.1配置文件

spring-security.xml 更改配置:

去掉

|  |
| --- |
| <http pattern=*"/cart/\*.do"* security=*"none"*></http> |

添加

|  |
| --- |
| <http use-expressions=*"false"* entry-point-ref=*"casProcessingFilterEntryPoint"*>  <intercept-url pattern=*"/cart/\*.do"* access=*"IS\_AUTHENTICATED\_ANONYMOUSLY"*/>  <intercept-url pattern=*"/\*\*"* access=*"ROLE\_USER"*/>  <custom-filter position=*"CAS\_FILTER"* ref=*"casAuthenticationFilter"* />  <custom-filter ref=*"requestSingleLogoutFilter"* before=*"LOGOUT\_FILTER"*/>  <custom-filter ref=*"singleLogoutFilter"* before=*"CAS\_FILTER"*/>  </http> |

access="IS\_AUTHENTICATED\_ANONYMOUSLY" 用于设置资源在不登陆时可以访问。

此配置与 security="none"的区别在于当用户未登陆时获取登陆人账号的值为anonymousUser ，而security="none"的话，无论是否登陆都不能获取登录人账号的值。

### 4.2.2代码实现

在portal-web的findCartList和addGoodsToCartList方法中，获取用户名

|  |
| --- |
| //得到登陆人账号,判断当前是否有人登陆  String username = SecurityContextHolder.*getContext*().getAuthentication().getName(); |

测试：当用户未登陆时，username的值为anonymousUser

## 4.3远程购物车存取

### 4.3.1服务接口层

Interface项目中CartService.java定义方法

|  |
| --- |
| //保存购物车到缓存中  **public** **void** insertCartListToRedis(List<Cart> cartList,String name);  //从缓存中获取购物车  **public** List<Cart> findCartListFromRedis(String name); |

### 4.3.2服务实现层

buyer-service项目中CartServiceImpl.java实现方法

|  |
| --- |
| @Autowired  **private** RedisTemplate redisTemplate;  //保存购物车到缓存中  **public** **void** insertCartListToRedis(List<Cart> newCartList,String name){  List<Cart> oldCartList = (List<Cart>) redisTemplate.boundHashOps("CART").get(name);  redisTemplate.boundHashOps("CART").put(name, mergeCartList(oldCartList,newCartList));  }  //从缓存中获取购物车  **public** List<Cart> findCartListFromRedis(String name){  **return** (List<Cart>) redisTemplate.boundHashOps("CART").get(name);  } |

### 4.3.3控制层

修改CartController.java的findCartList方法

|  |
| --- |
| // 去购物车结算  @RequestMapping("/findCartList")  **public** List<Cart> findCartList(HttpServletRequest request, HttpServletResponse response) {  List<Cart> cartList = **null**;  Cookie[] cookies = request.getCookies();  **if** (**null** != cookies && cookies.length > 0) {  **for** (Cookie cookie : cookies) {  **if** ("CART".equals(cookie.getName())) {  cartList = JSON.*parseArray*(cookie.getValue(), Cart.**class**);  **break**;  }  }  }  // 获取用户名  String name = SecurityContextHolder.*getContext*().getAuthentication().getName();  **if** (!"anonymousUser".equals(name)) {  // 已登陆  **if** (**null** != cartList) {  cartService.insertCartListToRedis(cartList, name);  // 清空Cookie  Cookie cookie = **new** Cookie("CART", **null**);  cookie.setPath("/");  cookie.setMaxAge(0);  response.addCookie(cookie);  }  cartList = cartService.findCartListFromRedis(name);  }  // 如果购物车中有商品 将商品的信息装满  **if** (**null** != cartList && cartList.size() > 0) {  cartList = cartService.findCartList(cartList);  }  **return** cartList;  } |

修改addGoodsToCartList方法

|  |
| --- |
| /\*\*  \* 购物车管理  \*  \* **@author** lx  \*  \*/  @RestController  @RequestMapping("/cart")  **public** **class** CartController {  @Reference  **private** CartService cartService;  // 加入购物车  @RequestMapping("/addGoodsToCartList")  @CrossOrigin(origins = "http://localhost:9004", allowCredentials = "true")  **public** Result addGoodsToCartList(HttpServletRequest request, HttpServletResponse response, Long itemId,  Integer num) {  **try** {  List<Cart> cartList = **null**;  // 未登陆时  Cookie[] cookies = request.getCookies();  **if** (**null** != cookies && cookies.length > 0) {  **for** (Cookie cookie : cookies) {  **if** ("CART".equals(cookie.getName())) {  cartList = JSON.*parseArray*(cookie.getValue(), Cart.**class**);  **break**;  }  }  }  //状态值 : 判断Cookie中是否已经含有购物车  Boolean i = **true**;  **if** (**null** == cartList) {  cartList = **new** ArrayList<Cart>();  i = **false**;  }  // 追加当前款  Item item = cartService.findItemById(itemId);  **if** (cartList.size() > 0) {  //状态值:判断商家是否相同  Boolean j = **true**;  // 根据item的ID查询Item对象  **for** (Cart cart : cartList) {  // 判断商家ID  **if** (cart.getSellerId().equals(item.getSellerId())) {  //状态值:判断商品是否含有同款  Boolean k = **true**;  List<OrderItem> orderItemList = cart.getOrderItemList();  **for** (OrderItem orderItem : orderItemList) {  **if** (orderItem.getItemId().equals(itemId)) {  // 相同款 不加新款 只加数量  orderItem.setNum(orderItem.getNum() + num);  k = **false**;  }  }  **if** (k) {  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  }  j = **false**;  }  }  **if** (j) {  // 追加当前款  Cart cart = **new** Cart();  List<OrderItem> orderItemList = **new** ArrayList<>();  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  cart.setOrderItemList(orderItemList);  cart.setSellerId(item.getSellerId());  cartList.add(cart);  }  } **else** {  // 追加当前款  Cart cart = **new** Cart();  List<OrderItem> orderItemList = **new** ArrayList<>();  OrderItem orderItem = **new** OrderItem();  orderItem.setItemId(itemId);  orderItem.setNum(num);  orderItemList.add(orderItem);  cart.setOrderItemList(orderItemList);  cart.setSellerId(item.getSellerId());  cartList.add(cart);  }  // 获取用户名  String name = SecurityContextHolder.*getContext*().getAuthentication().getName();  **if** ("anonymousUser".equals(name)) {  // 未登陆  // 保存到Cookie中  Cookie cookie = **new** Cookie("CART", JSON.*toJSONString*(cartList));  cookie.setPath("/");  cookie.setMaxAge(60 \* 60 \* 24);  response.addCookie(cookie);  } **else** {  // 已登陆  cartService.insertCartListToRedis(cartList, name);  **if** (i) {  // 清空Cookie  Cookie cookie = **new** Cookie("CART", **null**);  cookie.setPath("/");  cookie.setMaxAge(0);  response.addCookie(cookie);  }  }  **return** **new** Result(**true**, "加入购物车成功");  } **catch** (Exception e) {  // **TODO**: handle exception  **return** **new** Result(**true**, "加入购物车失败");  }  } |

### 4.3.4跳板页

（1）创建跳板页：portal-web 工程新建login.html ,页面添加脚本

|  |
| --- |
| <script type=*"text/javascript"*>  location.href="cart.html";  </script> |

1. 购物车页面链接到跳板页

|  |
| --- |
| 请<a href=*"login.html"*>登录</a> |

## 4.4购物车合并

### 4.4.1服务接口层

interface工程的CartService.java定义方法

|  |
| --- |
| //合并购物车  **public** List<Cart> mergeCartList(List<Cart> oldCartList,List<Cart> newCartList); |

### 4.4.2服务实现层

buyer-service工程CartServiceImpl.java实现方法

|  |
| --- |
| //合并购物车  **public** List<Cart> mergeCartList(List<Cart> oldCartList,List<Cart> newCartList){  //1:如果老购物车为Nulll或长度为0 直接返回新购物车  **if**(**null** != oldCartList && oldCartList.size() >0){  //2:如果新购物车为NULL 或长度为0 直接返回老购物车  **if**(**null** != newCartList && newCartList.size() >0){  //3:合并开始  **for** (Cart newCart : newCartList) {  **int** index = oldCartList.indexOf(newCart);  **if**(index == -1){  //没有同商家  oldCartList.add(newCart);  }**else**{  // 同一个商家  Cart oldCart = oldCartList.get(index);  // 同一个商家 里 的 商品集合  List<OrderItem> oldOrderItemList = oldCart.getOrderItemList();  // 新车里的商品集合  List<OrderItem> newOrderItemList = newCart.getOrderItemList();  **for** (OrderItem newOrderItem : newOrderItemList) {  //老车中商品集合  **int** i = oldOrderItemList.indexOf(newOrderItem);  **if**(i == -1){  //没有相同商品 追加新商品  oldOrderItemList.add(newOrderItem);  }**else**{  //有相同商品 合并数量  OrderItem oldOrderItem = oldOrderItemList.get(i);  oldOrderItem.setNum(oldOrderItem.getNum()+newOrderItem.getNum());  }  }    }  }  }**else**{  //老购物车  **return** oldCartList;  }  }**else**{  //新购物车  **return** newCartList;  }  //老购物车  **return** oldCartList;    } |