# 今日任务

1. 购物车服务端实现
2. 购物车同步方案
3. 代码实现

# 购物车（登录状态）

1. 把当前选择的商品添加到本地cookie中（同无登录状态）
2. 由于是登录状态，需要去服务器上查询是否有购物车信息
3. 如果服务器里有，就把服务器的购物车和本地的进行合并后，再更新服务器购物车。
4. 如果服务器没有，就把本地的购物车添加至服务器中。

## 前端合并问题？？

如果本地数据和服务器冲突了，怎么办？？ 我们以本地为准。

## 后端更新问题？？

合并时，以前端最新合并结果为准进行更新，但是请注意，重复商品的替换问题。

# 购物车同步

## 发布接口

|  |
| --- |
| @PostMapping("/cart/sync")  public @ResponseBody Object sync(long userId,String cart) {  if(userId==0) {  return ServiceResponse.illegalArgs();  }  try {  List<Cart> jsonToList = null;  if(StringUtils.isEmpty(cart)) {  jsonToList = new ArrayList<>();  }else {  jsonToList = JsonUtils.jsonToList(cart, new TypeReference<List<Cart>>() {});  }  List<Cart> list = cartService.sync(jsonToList, userId);  return ServiceResponse.ok(list);  } catch (Exception e) {  e.printStackTrace();  return ServiceResponse.error();  }  }    @PostMapping("/cart/remove")  public @ResponseBody Object remove(long userId,String ids) {  if(userId==0 || StringUtils.isEmpty(ids)) {  return ServiceResponse.illegalArgs();  }  try {  List<Cart> list = cartService.remove(ids, userId);  return ServiceResponse.ok(list);  } catch (Exception e) {  e.printStackTrace();  return ServiceResponse.error();  }  }    @GetMapping("/cart/list")  public @ResponseBody Object getCart(long userId) {  try {  List<Cart> list = cartService.get(userId);  return ServiceResponse.ok(list);  } catch (Exception e) {  e.printStackTrace();  return ServiceResponse.error();  }  } |

## 业务代码

|  |
| --- |
| package com.go9.mall.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import com.go9.mall.cart.CartCache;  import com.go9.mall.pojo.Cart;  @Service  public class CartServiceImpl implements CartService {  @Autowired  private CartCache cartCache;    //最新的客户端合并之后的购物车，更新服务器的购物车数据  @Override  public List<Cart> sync(List<Cart> carts,long userId) {  //同步购物车，如没有就添加新的商品，如有，结果以客户端为准  List<Cart> cartss = cartCache.getCarts().get(userId);  if(cartss==null || cartss.size()==0) {  cartss = carts;  }else {  //清空购物车，客户端为准  if(carts.size()==0) {  cartss.clear();  }else {  for(Cart cart : carts){  boolean find = false;  long id = cart.getId();  for(int i=0;i<cartss.size();i++) {  if(cartss.get(i).getId()==id) {  find = true;  //替换  cartss.set(i, cart);  break;  }  }  if(!find) {  //add  cartss.add(cart);  }  }  }  }  //一定别忘了要put进去  cartCache.getCarts().put(userId, cartss);  return cartss;  }  @Override  public List<Cart> get(long userId) {  return cartCache.getCarts().get(userId);  }  @Override  public List<Cart> remove(String ids, long userId) {  String[] carts = ids.split(",");  List<Cart> cartss = cartCache.getCarts().get(userId);  for(int i=0;i<carts.length;i++) {  long id = new Long(carts[i]);  for(int k=0;k<cartss.size();k++) {  if(id == cartss.get(k).getId()) {  cartss.remove(k);  break;  }  }  }  //一定要put进去  cartCache.getCarts().put(userId, cartss);  return cartss;  }  } |