# 购物车分析

## 购物车设计

1. 用户未登录状态下：在不登陆的情况下把购物车信息写入cookie

优点：

1、不占用服务端存储空间

2、代码实现简单。

3、用户体验好

缺点：

1、cookie中保存的容量有限。最大4k

2、把购物车信息保存在cookie中，更换设备购物车信息不能同步。

2. 用户已登录状态下：把购物车信息保存到服务端的 Redis 中

优点：

1、更换设备购物车信息可以同步

缺点：

1、占用服务端存储空间

### **创建usian\_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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_cart\_service</artifactId>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_redis</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_pojo</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

application.yml:

|  |
| --- |
| server:  port: 8100  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8761/eureka/  spring:  application:  name: usian-cart-service  datasource:  driver-class-name: com.mysql.jdbc.Driver  url: jdbc:mysql://127.0.0.1:3306/usian?characterEncoding=UTF-8  username: root  password: 1111  type: com.alibaba.druid.pool.DruidDataSource  redis:  cluster:  nodes:  - 192.168.233.131:7001  - 192.168.233.131:7002  - 192.168.233.131:7003  - 192.168.233.131:7004  - 192.168.233.131:7005  - 192.168.233.131:7006  jedis:  pool:  max-active: 20 #连接池最大连接数  max-idle: 10 #连接池中的最大空闲连接  min-idle: 5 # 连接池中的最小空闲连接 |

Logback.xml:

启动类：

### **创建usian\_cart\_feign**

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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_cart\_feign</artifactId>  <dependencies>  <!--Spring Cloud OpenFeign Starter -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-openfeign</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_pojo</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

### **创建usian\_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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_cart\_web</artifactId>  <dependencies>  <dependency>  <groupId>com.usian</groupId>  <artifactId>usian\_cart\_feign</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_utils</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>usian\_item\_feign</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

application.yml：

|  |
| --- |
| spring:  application:  name: usian-cart-web  server:  port: 8101  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8761/eureka/ |

logback.xml：

启动类：

# 未登录状态操作购物车

## 业务逻辑：

1、从cookie中查询商品列表：Map<itemId,TbItem> 商品购买数量使用TbItem的num保存

购物车已存在则直接返回

购物车不存在则创建空的购物车并返回

2、添加商品到购物车：

如果购物车存在该商品，商品数量相加。

如果购物车不存在该商品，根据商品id查询商品信息并添加到购车列表

3、把购车商品列表写入cookie。

读写cookie可以使用CookieUtils工具类实现

## 添加购物车：

Usian\_cart\_web

|  |
| --- |
| /\*\*  \* 加入购物车  \* @param userId  \* @param itemId  \* @param num  \* @param request  \* @param response  \* @return  \*/  @RequestMapping("/addItem")  public Result addItem(String userId, Long itemId,  @RequestParam(defaultValue = "1") Integer num, HttpServletRequest request,  HttpServletResponse response){  try {  /\*\*\*\*\*\*\*\*\*\*\*在用户未登录的状态下\*\*\*\*\*\*\*\*\*\*/  if(StringUtils.isBlank(userId)){  //获取购物车  Map<String, TbItem> cart = getCartFromCookie(request);  //添加商品到购物车  addItemToCart(itemId,num,cart);  //将购物车加入cookie  addClientCookie(response,cart,request);  }  /\*\*\*\*在用户已登录的状态\*\*\*\*/  else{    }  return Result.ok();  }catch (Exception e){  e.printStackTrace();  return Result.error("error");  }  }  private Boolean addItemToRedis(String userId, Map<String, TbItem> cart) {  return cartServiceFeign.insertCart(userId,cart);  }  /\*\*  \* 从Redis获取购物车  \* @param userId  \* @return  \*/  private Map<String, TbItem> getCartFromRedis(String userId) {  Map<String, TbItem> cart = cartServiceFeign.getCartFromRedis(userId);  if(cart!=null && cart.size()>0){  return cart;  }  return new HashMap<String, TbItem>();  }  /\*\*  \* 将购物车加入cookie  \* @param response  \* @param cart  \* @param request  \*/  private void addClientCookie(HttpServletResponse response, Map<String, TbItem> cart,  HttpServletRequest request) {  String cartJson = JsonUtils.objectToJson(cart);  CookieUtils.setCookie(request,response,CART\_COOKIE\_KEY,cartJson,  CART\_COOKIE\_EXPIRE,true);  }  /\*\*  \* 添加商品到购物车  \* @param itemId  \* @param num  \* @param cart  \*/  private void addItemToCart(Long itemId, Integer num, Map<String, TbItem> cart) {  //查询购物有无该商品  TbItem tbItem = cart.get(itemId.toString());  //如果有修改库存数量  if(tbItem!=null){  tbItem.setNum(tbItem.getNum()+num);  }  //如果没有在数据库查，设置库存量，  else{  tbItem = itemServiceFeign.selectItemInfo(itemId);  tbItem.setNum(num);  }  //添加到购物车  cart.put(itemId.toString(),tbItem);  }  /\*\*  \* 从cookie获取购物车  \* @param request  \* @return  \*/  private Map<String, TbItem> getCartFromCookie(HttpServletRequest request) {  String cookieValue = CookieUtils.getCookieValue(request, CART\_COOKIE\_KEY, true);  if(StringUtils.isNotBlank(cookieValue)){  Map<String, TbItem> map = JsonUtils.jsonToMap(cookieValue, TbItem.class);  return map;  }  return new HashMap<String,TbItem>();  } |

## 查看购物车:

Usian\_cart\_web

|  |
| --- |
| /\*\*  \* 查看购物车  \* @param userId  \* @param request  \* @return  \*/  @RequestMapping("/showCart")  public Result showCart(String userId,HttpServletRequest request){  try {  List<TbItem> itemList = new ArrayList<>();  /\*\*\*\*\*\*\*\*\*\*\*在用户未登录的状态下\*\*\*\*\*\*\*\*\*\*/  if(StringUtils.isBlank(userId)){  //未登录  Map<String,TbItem> cart = getCartFromCookie(request);  Set<String> keys = cart.keySet();  for (String key : keys) {  TbItem tbItem = cart.get(key);  itemList.add(tbItem);  }  }else{  //登录  }  return Result.ok(itemList);  }catch (Exception e){  e.printStackTrace();  return Result.error("error");  }  } |

## 修改购物车:

Usian\_cart\_web

|  |
| --- |
| /\*\*  \* 修改购物车  \* @param itemId  \* @param userId  \* @param num  \* @param request  \* @param response  \* @return  \*/  @RequestMapping("/updateItemNum")  public Result updateItemNum(Long itemId,String userId,Integer num,  HttpServletRequest request,HttpServletResponse response){  try{  /\*\*\*\*\*\*\*\*\*\*\*在用户未登录的状态下\*\*\*\*\*\*\*\*\*\*/  if(StringUtils.isBlank(userId)){  Map<String, TbItem> cart = getCartFromCookie(request);  TbItem tbItem = cart.get(itemId.toString());  tbItem.setNum(num);  addClientCookie(response,cart,request);  }else{  }  return Result.ok();  }catch (Exception e){  e.printStackTrace();  return Result.error("error");  }  } |

## 删除购物车:

|  |
| --- |
| /\*\*  \* 购物车删除  \* @param userId  \* @param itemId  \* @param request  \* @return  \*/  @RequestMapping("/deleteItemFromCart")  public Result deleteItemFromCart(String userId,Long itemId,  HttpServletRequest request,HttpServletResponse response){  try {  if(StringUtils.isBlank(userId)){  Map<String, TbItem> cart = getCartFromCookie(request);  cart.remove(itemId.toString());  addClientCookie(response,cart,request);  }else{  }  return Result.ok();  }catch (Exception e){  e.printStackTrace();  return Result.error("error");  }  } |