# **购物车分析**

未登录状态操作购物车

登录状态操作购物车

合并购物车

## **购物车功能**

1.添加购物车商品

2.展示购物车列表页面

3.修改购物车商品数量

4.删除购物车商品

## **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**

<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>

**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**

@SpringBootApplication

@EnableDiscoveryClient

public class CartServiceApp {

public static void main(String[] args) {

SpringApplication.run(CartServiceApp.class, args);

}

}

### **创建usian\_cart\_feign**

#### **pom.xml**

<!--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>

### **创建usian\_cart\_web**

#### **pom.xml**

<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>

#### **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**

@SpringBootApplication

@EnableDiscoveryClient

@EnableFeignClients

public class CartWebApp {

public static void main(String[] args) {

SpringApplication.run(CartWebApp.class, args);

}

}

## **添加购物车**

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

## **查看购物车**

### **usian\_cart\_web**

#### **controller**

/\*\*

\* 查看购物车

\*/

@RequestMapping("/showCart")

public Result showCart(String userId,HttpServletRequest

request,HttpServletResponse response){

try{

if(StringUtils.isBlank(userId)){

//在用户未登录的状态下

List<TbItem> list = new ArrayList<TbItem>();

Map<String,TbItem> cart = this.getCartFromCookie(request);

Set<String> keys = cart.keySet();

for(String key :keys){

list.add(cart.get(key));

}

}else{

// 在用户已登录的状态

}

return Result.ok(list);

}catch(Exception e){

e.printStackTrace();

}

return Result.error("error");

}

## **修改购物车**

/\*\*

\* 修改购物车

\*/

@RequestMapping("/updateItemNum")

public Result updateItemNum(String userId,Long itemId,Integer num,

HttpServletRequest request,HttpServletResponse response){

try {

if(StringUtils.isBlank(userId)){

//未登录

//1、获得cookie中的购物车

Map<String, TbItem> cart = getCartFromCookie(request);

//2、修改购物车中的商品

TbItem tbItem = cart.get(itemId.toString());

tbItem.setNum(num);

cart.put(itemId.toString(),tbItem);

//3、把购物车写到cookie

addClientCookie(cart,request,response);

}else{

//登录

}

return Result.ok();

}catch (Exception e){

e.printStackTrace();

return Result.error("修改失败");

}

}

## **删除购物车**

/\*\*

\* 删除购物车中的商品

\*/

@RequestMapping("/deleteItemFromCart")

public Result deleteItemFromCart(Long itemId, String userId, HttpServletRequest

request, HttpServletResponse response) {

try {

if (StringUtils.isBlank(userId)) {

//在用户未登录的状态下

Map<String,TbItem> cart = this.getCartFromCookie(request);

cart.remove(itemId.toString());

this.addClientCookie(request,response,cart);

} else {

// 在用户已登录的状态

}

return Result.ok();

} catch (Exception e) {

e.printStackTrace();

}

return Result.error("删除失败");

}