# 整体架构设计

# 实体类设计

# 数据存储设计

# 用户管理模块DAO层开发

## DAO层接口定义

## DAO层实现开发

# 用户管理模块SERVICE层开发

## SERVICE层接口定义

## SERVICE层实现开发

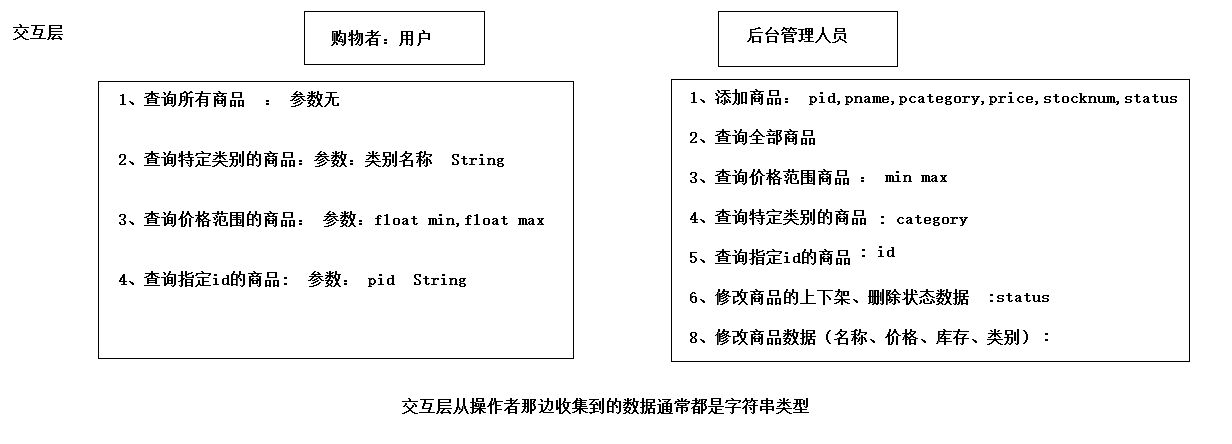
# 商品管理模块DAO层开发

## DAO层接口定义

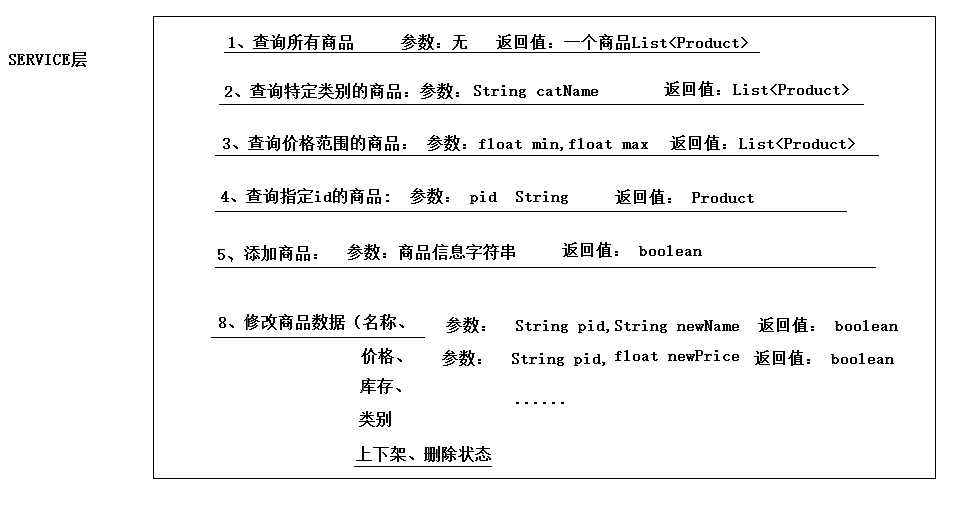
## DAO层实现开发

# 商品管理模块功能设计

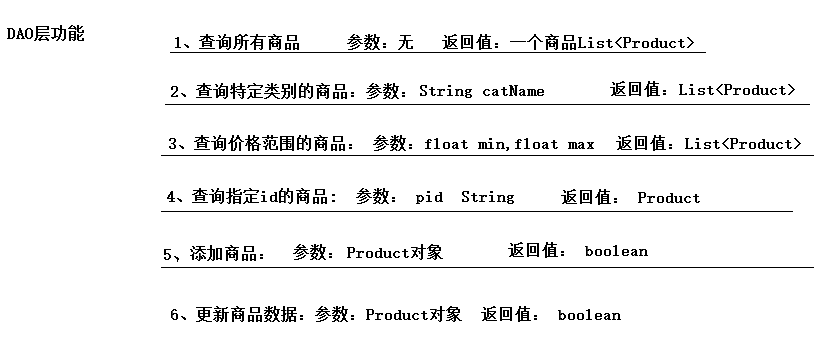
## 交互层功能设计



## SERVICE层功能设计



## DAO层功能设计



# 商品管理模块SERVICE层开发

## SERVICE层接口定义

## SERVICE层实现开发

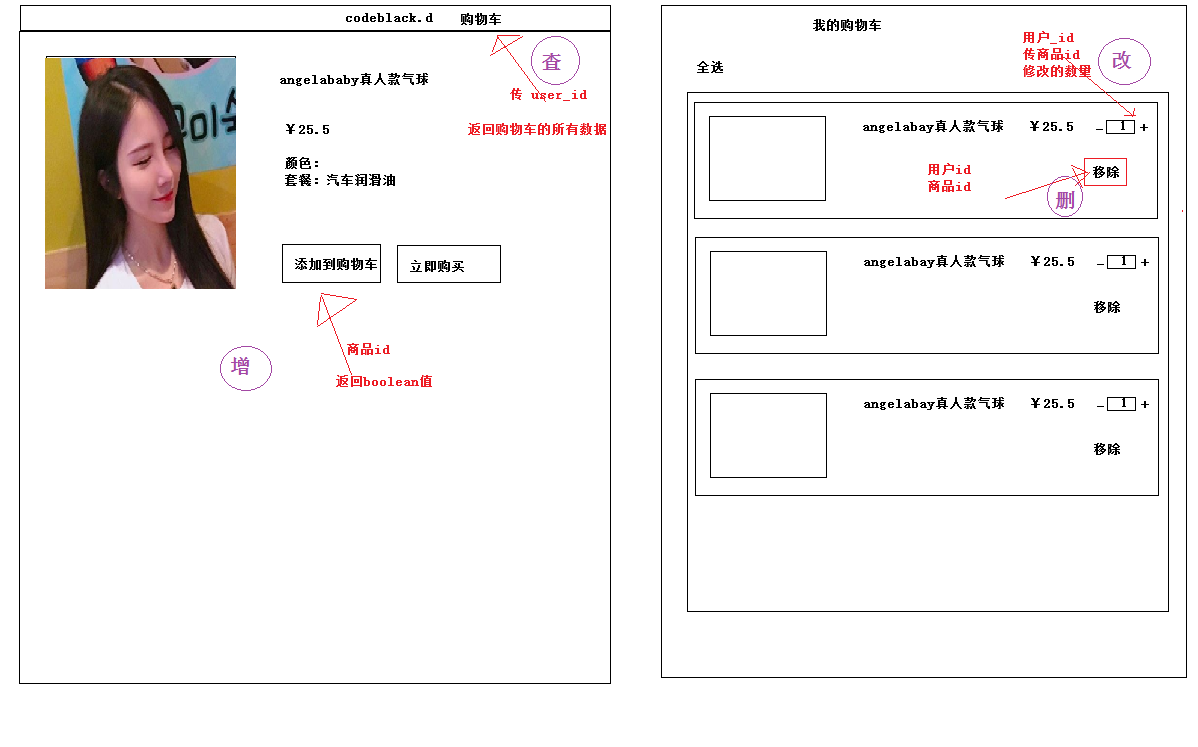
# 商品管理模块DAO层开发

## DAO接口定义

## DAO 实现开发

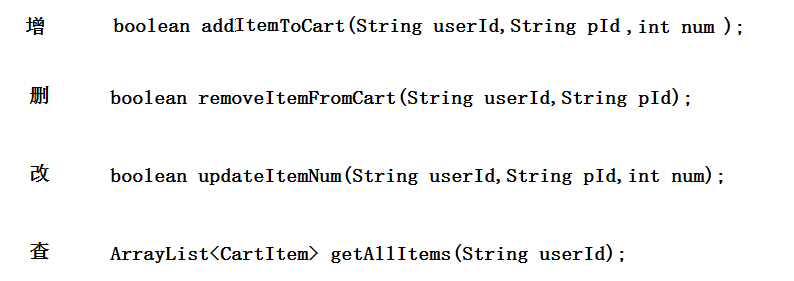
# 购物车模块功能分析设计

## 购物车模块交互层功能设计



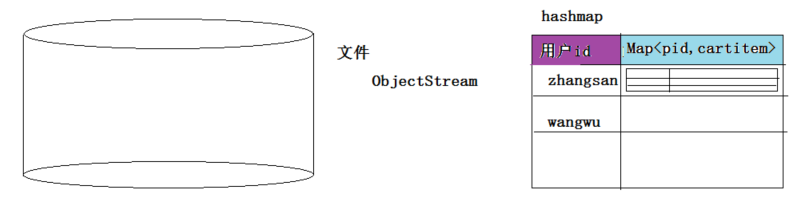
## 购物车模块SERVICE层接口设计

由于购物车中所装的商品数据 跟 商品管理模块中的Product类所封装的数据有所不同，因此在购物车模块中专门设计一个实体类CartItem来封装购物车中的商品数据



## 购物车模块DAO层接口设计

数据模型：



# 购物车模块DAO层开发

## 购物车模块DAO层接口定义

|  |
| --- |
| **public** **interface** CartDao {  // 此方法，用于购物车数据库中已有该用户的购物项  **public** **boolean** addItemToCart(String userId,CartItem item) **throws** Exception ;    // 此方法，用于这个用户第一次购物  **public** **boolean** addNewUserItemToCart(String userId, CartItem item);    **public** **boolean** removeItemFromCart(String userId,String pId) **throws** Exception ;    **public** **boolean** updateItemBuyNum(String userId,String pId,**int** buyNum) **throws** Exception ;    **public** List<CartItem> getAllItemsFromCart(String userId) **throws** Exception ;    // 判断数据库中是否有某个人的购物车  **public** **boolean** checkIfUserExist(String userId);    // 判断某个人的购物车中是否有指定的商品  **public** **boolean** checkIfItemExsit(String userId,String pId);  // 根据指定的userid和商品id从购物车中取出购物项  **public** CartItem getItemById(String userId, String pId);  } |

## 购物车模块DAO层实现开发

# 购物车模块SERVICE层开发

## 购物车模块SERVICE层接口开发

|  |
| --- |
| **public** **interface** CartService {  /\*\*  \* 增  \*/  **public** **boolean** addItemToCart(String userId,String pId,**int** buyNum) **throws** Exception ;    /\*\*  \* 删  \*/  **public** **boolean** removeItemFromCart(String userId,String pId) **throws** Exception ;      /\*\*  \* 改  \*/  **public** **boolean** updateItemBuyNum(String userId,String pId,**int** buyNum) **throws** Exception ;      /\*\*  \* 查  \*/  **public** List<CartItem> getAllItems(String userId) **throws** Exception ;  } |

## 购物车模块SERVICE层实现开发

### 添加商品到购物车

|  |
| --- |
| **public** **class** CartServiceImpl **implements** CartService {  CartDao cartDao = **null**;  ProductDao productDao = **new** ProductDaoImpl();  /\*\*  \* 当页面上的用户点击添加商品到购物车时，  \* 1、判断购物车数据库中是否已经有该用户的数据  \* 2、如果该用户存在，继续判断该商品是否在他的购物项hashmap中  \* 3、如果用户和购物项都存在，本方法只做叠加商品数量的事  \*  \* 4、如果用户存在，但商品不存在，要做的事： 添加这个商品到该用户的购物项hashmap中  \*  \* 5、如果用户不存在，为这个用户构造一个购物项hashmap，添加到购物车数据库中  \*  \*/  @Override  **public** **boolean** addItemToCart(String userId, String pId, **int** buyNum) **throws** Exception {  **boolean** addItemToCart;  // 先判断购物车数据库中是否有该用户  **boolean** checkIfUserExist = cartDao.checkIfUserExist(userId);  // 如果该用户确实已存在于购物车数据库  **if** (checkIfUserExist) {  // 再判断该商品是否存在与购物项中  **boolean** checkIfItemExsit = cartDao.checkIfItemExsit(userId, pId);  **if** (checkIfItemExsit) {  // 先取出这个人已有的这个购物项数据  CartItem item = cartDao.getItemById(userId, pId);  // 然后把之前的数量+本次的数量，通过dao,写入数据库  addItemToCart = cartDao.updateItemBuyNum(userId, pId, buyNum + item.getBuyNum());  } **else** {  // 先根据pId从商品管理模块的dao中获取商品信息  Product productsById = productDao.getProductsById(pId);  // 然后构造一个购物项实体对象，并填入商品数据  CartItem item = **new** CartItem();  item.set(pId, productsById.getpName(), productsById.getPrice(), buyNum);  // 再利用购物车dao将这个购物项添加到用户已有的购物车数据中  addItemToCart = cartDao.addItemToCart(userId, item);  }  // 如果该用户在购物车数据库中不存在  } **else** {  // 先根据pId从商品管理模块的dao中获取商品信息  Product productsById = productDao.getProductsById(pId);  // 然后构造一个购物项实体对象，并填入商品数据  CartItem item = **new** CartItem();  item.set(pId, productsById.getpName(), productsById.getPrice(), buyNum);  // 再利用购物车dao将这个购物项新增到购物车中  addItemToCart = cartDao.addNewUserItemToCart(userId, item);  }  **return** addItemToCart;  }  ..................... |

# 订单管理模块开发设计

## 订单模块交互层功能设计

## 订单模块SERVICE层功能设计

## 订单模块SERVICE层功能设计

# 订单模块DAO层开发

## 订单模块DAO层接口定义

## 订单模块DAO层实现开发

# 订单模块SERVICE层开发

## 订单模块SERVICE层接口定义

## 订单模块SERVICE层实现开发