武汉纺织大学

Web应用开发课程设计

**学 院： 数学与计算机学院**

**班 级： 软件11804**

**姓 名： 熊天天**

**学 号： 1804240903**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

目 录

[1 需求分析 1](#_Toc59222734)

[1.1查询 1](#_Toc59222735)

[2 系统设计 1](#_Toc59222736)

[2.1UML时序图（Sequence Diagram） 1](#_Toc59222737)

[2.2表结构 2](#_Toc59222738)

[3 系统实现 3](#_Toc59222739)

[3.1 项目结构 3](#_Toc59222740)

[3.2 控制器QueryProduct 3](#_Toc59222741)

[3.3 VO类Product.java 5](#_Toc59222742)

[3.4 DAO接口类ProductDAO.java 7](#_Toc59222743)

[3.5 工具包tools 10](#_Toc59222744)

[4 系统测试 11](#_Toc59222745)

[5 系统总结 12](#_Toc59222746)

# 1 需求分析



## 1.1查询

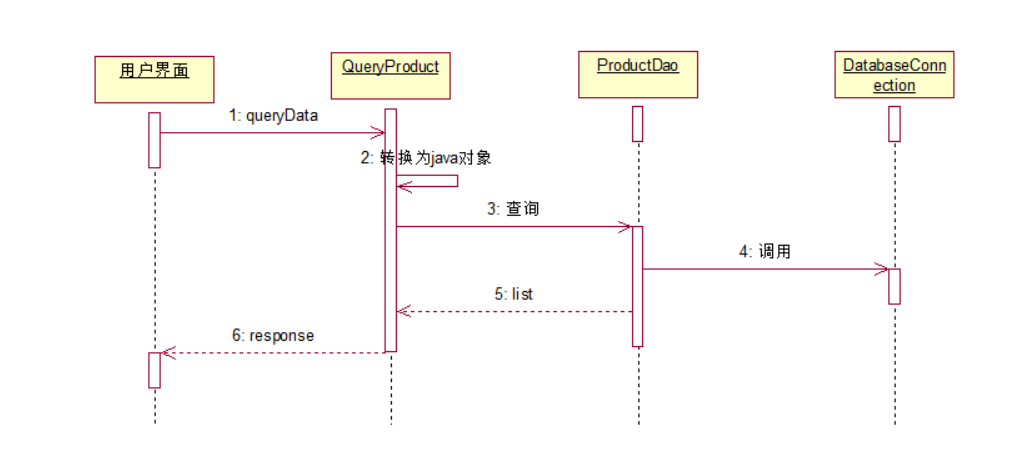
要求：

1. 数据来自于数据库
2. 可进行组合查询
3. 某些字段支持模糊查询
4. 分页显示

输入商品货号或条形码和公司，点查询后在列表中显示查询出的信息。

# 2 系统设计

## 2.1UML时序图（Sequence Diagram）

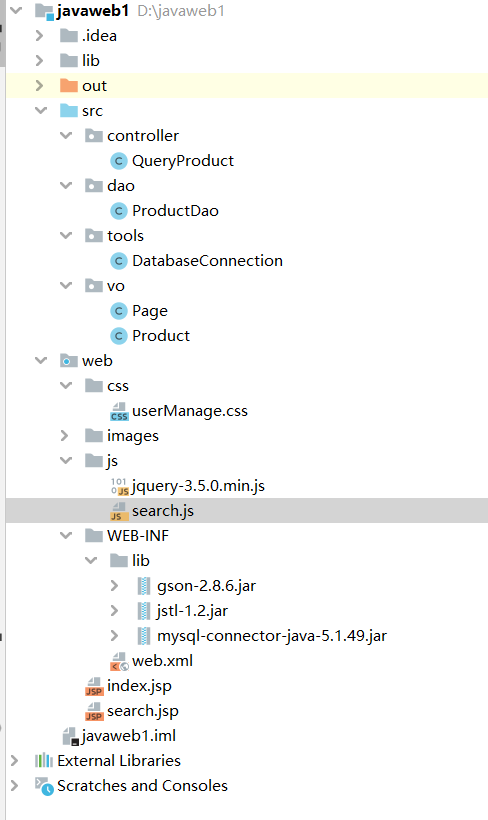


## 2.2表结构



# 3 系统实现

## 3.1 项目结构



## 3.2 控制器QueryProduct

控制器通过getParameter(),接受从前端传来的查询参数和页面参数，转换为java对象，

调用ProductDao的query()和count() 方法查询存到动态数组和一个变量上，转为json格式，再通过字符流输出。

**public class QueryProduct extends javax.servlet.http.HttpServlet {**

**protected void doPost(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException, IOException {**

**//接收客户端参数**

**String queryParams = request.getParameter("queryParams");**

**String pageParams = request.getParameter("pageParams");**

**System.out.println("查询参数："+pageParams);**

**System.out.println("分页参数："+queryParams);**

**//将json字符串参数值转换为java对象**

**Gson gson = new GsonBuilder().serializeNulls().create();**

**HashMap<String, Object> mapPage = gson.fromJson(pageParams, HashMap.class);**

**Page page = Page.getPageParams(mapPage); //转换为page对象**

**Product product = new Product();**

**if (queryParams != null) {**

**product = gson.fromJson(queryParams, Product.class);**

**}**

**//用dao查**

**ProductDao dao = new ProductDao();**

**ArrayList<Product> rows = dao.query(product, page); //查询**

**int total = dao.count(product, page); //查询记录总数**

**//转换为json数据**

**HashMap<String, Object> mapReturn = new HashMap<String, Object>();**

**mapReturn.put("rows", rows);**

**mapReturn.put("total", total);**

**String jsonStr = gson.toJson(mapReturn);**

**// 字符流输出**

**response.setContentType("text/html;charset=utf-8");**

**PrintWriter out = response.getWriter();**

**System.out.println(jsonStr);**

**out.print(jsonStr);**

**out.flush();**

**out.close();**

**}**

## 3.3 VO类Product.java

### 3.3.1 Product类

Product类，属性为数据库中product表中所有的列。有基本的getter,setter方法

**package vo;**

**public class Product {**

**private String id;**

**private String barcode;**

**private String name;**

**public Product(String id, String barcode, String name) {**

**this.id = id;**

**this.barcode = barcode;**

**this.name = name;**

**}**

**private String category;**

**private String reclassify;**

**private Integer price;**

**private Integer minsaleprice;**

**private Integer maxsaleprice;**

**private String specifications;**

**private String origin;**

**private String company;**

**public Product() {**

**}**

//构造方法、getter,setter，toString方法,自行补充

}

### 3.3.2 Page类

Page类

属性：

pagesize页面尺寸（一面放多少行数据）

pageNumber当前所在的页数

sort 排序的列

sortOder排序顺序asc或者desc

方法：

getter setter方法

getPageParams()方法

根据传入的HashMap返回一个page对象（在控制器QueryProduct.java中被调用）

public class Page {

private int pageSize;

private int pageNumber;

private String sort;

private String sortOrder;

public int getPageSize() {

return pageSize;

}

public void setPageSize(int pageSize) {

this.pageSize = pageSize;

}

public int getPageNumber() {

return pageNumber;

}

public void setPageNumber(int pageNumber) {

this.pageNumber = pageNumber;

}

public String getSort() {

return sort;

}

public void setSort(String sort) {

this.sort = sort;

}

public String getSortOrder() {

return sortOrder;

}

public void setSortOrder(String sortOrder) {

this.sortOrder = sortOrder;

}

public static Page getPageParams(HashMap<String,Object> map){

Page page=new Page();

if(map.get("pageNumber")!=null || !"".equals((String)map.get("pageNumber"))){

page.setPageNumber(Integer.parseInt((String)map.get("pageNumber")));

}

else{

page.setPageNumber(1);

}

if(map.get("pageSize")!=null || !"".equals((String)map.get("pageSize"))){

page.setPageSize(Integer.parseInt((String)map.get("pageSize")));

}

else{

page.setPageSize(10);

}

if(map.get("sort")!=null || !"".equals((String)map.get("sort"))){

page.setSort(String.valueOf(map.get("sort")));

}

else{

page.setSort("id");

}

if(map.get("sortOrder")!=null || !"".equals((String)map.get("sortOrder")) ){

page.setSortOrder(String.valueOf(map.get("sortOrder")));

}

else{

page.setSortOrder("asc");

}

return page;

}

}

## 3.4 DAO接口类ProductDAO.java

两个方法：

1.query()

传入product（包含有查询的参数id、barcode，也可能为null）和page（需要其pageSize、 pageNumber、sortOrder、sort来完成分页）

List存放查询的结果，condition存放查询条件

判断id、barcode是否为空，不为空就以”and id like %xx%”形式先加入到condition中，然后再拼接到sql语句中，以实现模糊查询。

根据当前页面的值和大小，在sql语句中拼接limit子句实现分页查询。根据sort在sql语句中拼接order by语句，实现排序。

再将每一条结果用一个Product对象存放，然后加入到List中去。

2.count()

生成条件 condition的操作与query()的一致，根据这个condition返回匹配的行数。

public class ProductDao {

public Product get(String id) {

Product product = null;

public ArrayList<Product> query(Product product, Page page) {

ArrayList<Product> list = new ArrayList<Product>(); // 存放查询结果的集合

StringBuffer condition = new StringBuffer();// 查询条件

if (product.getId() != null&& !"".equals(product.getId())) { // 判断是否有该查询条件

condition.append(" and id like '%")

.append(product.getId()).append("%'");

}

if (product.getBarcode() != null&& !"".equals(product.getBarcode())) { //

condition.append(" and barcode like '%").append(product.getBarcode())

.append("%'");

}

int begin = page.getPageSize() \* (page.getPageNumber() - 1);

String sql = "select id,barcode,name,category,reclassify,price,minsaleprice,maxsaleprice,specifications,origin,company";

sql = sql + " from product ";

sql = sql + " where 1=1 ";

sql = sql + condition + " order by " + page.getSort() + " "

+ page.getSortOrder() + " limit " + begin + ","

+ page.getPageSize();

System.out.println(sql);

try {

DatabaseConnection databaseConnection=new DatabaseConnection();

Connection con = databaseConnection.getConnection();

PreparedStatement pst = con.prepareStatement(sql);

ResultSet rs = pst.executeQuery();

while (rs.next()) {

Product productResult = new Product();

productResult.setId(rs.getString("id"));

productResult.setBarcode(rs.getString("barcode"));

productResult.setCategory(rs.getString("category"));

productResult.setCategory(rs.getString("reclassify"));

productResult.setCompany(rs.getString("company"));

productResult.setMaxsaleprice(rs.getInt("maxsaleprice"));

productResult.setMinsaleprice(rs.getInt("minsaleprice"));

productResult.setName(rs.getString("name"));

productResult.setOrigin(rs.getString("origin"));

productResult.setPrice(rs.getInt("price"));

productResult.setReclassify(rs.getString("reclassify"));

productResult.setSpecifications(rs.getString("specifications"));

list.add(productResult);

}

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return list;

}

//根据传入的参数进行查询

public ArrayList<Product> query(Product product, Page page) {

ArrayList<Product> list = new ArrayList<Product>(); // 存放查询结果的集合

StringBuffer condition = new StringBuffer();// 查询条件

if (product.getId() != null && !"".equals(product.getId())) { // 判断是否有该查询条件

condition.append(" and id like '%")

.append(product.getId()).append("%'");

}

if (product.getBarcode() != null && !"".equals(product.getBarcode())) { //

condition.append(" and barcode like '%").append(product.getBarcode())

.append("%'");

}

int begin = page.getPageSize() \* (page.getPageNumber() - 1);

String sql = "select id,barcode,name,category,reclassify,price,minsaleprice,maxsaleprice,specifications,origin,company";

sql = sql + " from product ";

sql = sql + " where 1=1 ";

sql = sql + condition + " order by " + page.getSort() + " "

+ page.getSortOrder() + " limit " + begin + ","

+ page.getPageSize();

System.out.println(sql);

// DatabaseConnection类封装了数据库驱动加载和连接

DatabaseConnection dbc = new DatabaseConnection();

Connection con = dbc.getConnection();

try {

Statement stm = con.createStatement();

ResultSet rs = stm.executeQuery(sql);

while (rs.next()) {

Product productResult = new Product();

productResult.setId(rs.getString("id"));

productResult.setBarcode(rs.getString("barcode"));

productResult.setCategory(rs.getString("category"));

productResult.setCompany(rs.getString("company"));

productResult.setMaxsaleprice(rs.getInt("maxsaleprice"));

productResult.setMinsaleprice(rs.getInt("minsaleprice"));

productResult.setName(rs.getString("name"));

productResult.setOrigin(rs.getString("origin"));

productResult.setPrice(rs.getInt("price"));

productResult.setReclassify(rs.getString("reclassify"));

productResult.setSpecifications(rs.getString("specifications"));

list.add(productResult);

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

dbc.close();// 6.关闭连接

}

return list;

}

//查询记录数

public int count(Product product, Page page){

int total=0;

StringBuffer condition = new StringBuffer();// 查询条件

if (product.getId() != null && !"".equals(product.getId())) { // 判断是否有该查询条件

condition.append(" and id like '%")

.append(product.getId()).append("%'");

}

if (product.getBarcode() != null && !"".equals(product.getBarcode())) { //

condition.append(" and barcode like '%").append(product.getBarcode())

.append("%'");

}

String sql = "select count(\*) from product";

sql = sql + " where 1=1 ";

sql = sql + condition;

System.out.println(sql);

DatabaseConnection dbc = new DatabaseConnection();

Connection con = dbc.getConnection();

try {

Statement stm = con.createStatement();

ResultSet rs = stm.executeQuery(sql);

if (rs.next()) {

total = rs.getInt(1);

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

// 6.关闭连接

dbc.close();

}

return total;

}

}

## 3.5 工具包tools

以个用来进行数据库连接的工具

public class DatabaseConnection {

private static final String JDBCPROPERTY = "jdbc.properties";

private static String DBDRIVER = "com.mysql.jdbc.Driver";

private static String DBURL = "jdbc:mysql://localhost:3306/product";

private static String DBUSER = "root";

private static String PASSWORD = "123";

private Connection conn;

/\*\*

\*/

static {

try {

Properties property = new Properties();

InputStream is = DatabaseConnection.class.getResourceAsStream("/"+JDBCPROPERTY);

property.load(new InputStreamReader(is, "utf-8"));

is.close();

DBDRIVER = property.getProperty("DBDRIVER");

DBURL = property.getProperty("DBURL");

DBUSER = property.getProperty("DBUSER");

PASSWORD = property.getProperty("PASSWORD");

Class.forName(DBDRIVER);

} catch (Exception e) {

e.printStackTrace();

}

}

public DatabaseConnection() {

try {

this.conn = DriverManager.getConnection(DBURL, DBUSER, PASSWORD);

} catch (Exception e) {

e.printStackTrace();

}

}

public Connection getConnection() {

return this.conn;

}

public void close() {

if (this.conn != null) {

try {

this.conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

## 3.6 js search.js

用queryParms和queryParams保存参数，转换为Json字符串存到queryData中



ajax() 方法发送数据queryData，请求成功后的回调函数把查询到的数据遍历到表中。



分页操作函数：

1.翻页，attr()获取被选元素的值，分别对应，然后将pageNumber改为相应的值。调用reload()重新加载。



2．排序



默认的就是asc，根据其class值确定sortOder值



# 4 系统测试





# 5 系统总结

这次课程设计没有使用框架，就用的初步的时候教的结构和方法。前端通过ajax传入数据，QueryProduct接受客户端数据，将json字符串参数值转换为java对象，用dao查，再把查出来的数据字符流输出。

这门课让我学到了很多关于 java web的很多知识，虽然对于框架还不熟悉，但是算是有一些了解，对基本的原理明白了，以后自己学习心里也有谱。