[一、数据库](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "_15" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[二、动态Web工程](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "Web_35" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[1、实现bean--->Book](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "1beanBook_37" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[2、分页Page 业务逻辑Bean（用于实现分页的类）](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "2Page_Bean_128" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[3、创建Druid的数据库连接工具JDBCTools.java](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "3DruidJDBCToolsjava_202" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[4、创建带泛型的BasicDAO类](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "4BasicDAO_296" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[5、创建BookDAO接口](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "5BookDAO_428" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[6、创建BookDAO接口实现类BookDAOImpl类](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "6BookDAOBookDAOImpl_470" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[7、创建BookService接口](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "7BookService_549" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[8、创建BookService接口实现类BookServiceImpl类](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "8BookServiceBookServiceImpl_570" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[9、创建基础BaseServlet](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "9BaseServlet_648" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[10、创建BaseServlet的子类](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "10BaseServlet_696" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[三、JSP页面的编写](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "JSP_774" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[1、编写index.jsp](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "1indexjsp_776" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[2、重用页面](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "2_786" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[3、编写显示查询结果的页面book\_manager.jsp](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "3book_managerjsp_839" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[（1）页码导航栏的编写【普通的页码导航栏】](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "1_841" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[4、编写带有价格的查询页面book\_client.jsp](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "4book_clientjsp_977" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[（2）页码导航栏的编写【动态的页码导航栏】](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "2_979" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

[四、最终实现的效果](https://blog.csdn.net/weixin_41043145/article/details/100012100?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522160828259516780308314197%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=160828259516780308314197&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-1-100012100.pc_search_result_no_baidu_js&utm_term=%E5%A4%9A%E6%9D%A1%E4%BB%B6%E7%BB%84%E5%90%88%E6%9F%A5%E8%AF%A2%E5%88%86%E9%A1%B5%E6%98%BE%E7%A4%BA%E5%AE%9E%E9%AA%8C%E6%8A%A5%E5%91%8A" \l "_1145" \t "https://blog.csdn.net/weixin_41043145/article/details/_self)

# **说明**

这个demo包括用户登录、注册、注销和分页，主要介绍分页，所以下面主要展示关于分页的说明

分页（两种）：

第一种：简单查询分页

第二种：带价格的查询分页

## **一、数据库**

关于数的表格结构：

CREATE TABLE `books` (

`id` int(11) NOT NULL AUTO\_INCREMENT, */\* 编号\*/*

`title` varchar(100) NOT NULL,*/\*书名 \*/*

`author` varchar(100) NOT NULL,*/\*作者 \*/*

`price` double(11,2) NOT NULL,*/\* 价格\*/*

`sales` int(11) NOT NULL,*/\* 销售量\*/*

`stock` int(11) NOT NULL,*/\* 库存量\*/*

`img\_path` varchar(100) NOT NULL,*/\* 图片路径\*/*

PRIMARY KEY (`id`)*/\* 主键\*/*) ENGINE=InnoDB AUTO\_INCREMENT=46 DEFAULT CHARSET=utf8;

## **二、动态Web工程**

### **1、实现bean—>Book**

package com.xxj.bean;

public class Book {

private Integer id;

private String title;

private String author;

private Double price;

private Integer sales;

private Integer stock;

private String imgPath ="static/img/default.jpg"; *//这个地方给了一个默认的图片路径*

public Book(Integer id, String title, String author, Double price, Integer sales, Integer stock, String imgPath) {

super();

this.id = id;

this.title = title;

this.author = author;

this.price = price;

this.sales = sales;

this.stock = stock;

this.imgPath = imgPath;

}

public Book(Integer id, String title, String author, Double price, Integer sales, Integer stock) {

super();

this.id = id;

this.title = title;

this.author = author;

this.price = price;

this.sales = sales;

this.stock = stock;

}

public Book() {

super();

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public Integer getSales() {

return sales;

}

public void setSales(Integer sales) {

this.sales = sales;

}

public Integer getStock() {

return stock;

}

public void setStock(Integer stock) {

this.stock = stock;

}

public String getImgPath() {

return imgPath;

}

public void setImgPath(String imgPath) {

this.imgPath = imgPath;

}

@Override

public String toString() {

return "Book [id=" + id + ", title=" + title + ", author=" + author + ", price=" + price + ", sales=" + sales

+ ", stock=" + stock + ", imgPath=" + imgPath + "]";

}}

### **2、分页Page 业务逻辑Bean（用于实现分页的类）**

package com.xxj.bean;

import java.util.List;

public class Page<T> {

private List<T> list; *//从数据库中查询当前页Book的集合*

*//设置为public便于后面操作*

public static final int PAGE\_SIZE = 4; *//每页显示多少条记录 设置的固定值*

private int pageNo; *//当前页 用户输入的页码*

private int totalPageNo; *//总页数 计算出来的*

private int totalRecord; *//从数据库中查询的总记录数*

public int getTotalRecord() {

return totalRecord;

}

public void setTotalRecord(int totalRecord) {

this.totalRecord = totalRecord;

}

public void setTotalPageNo(int totalPageNo) {

this.totalPageNo = totalPageNo;

}

public List<T> getList() {

return list;

}

public void setList(List<T> list) {

this.list = list;

}

public int getPageNo() {

if (pageNo < 1) {

*// 如果当前页码小于1，直接返回1*

return 1;

} else if (pageNo > getTotalPageNo()) {

*// 如果当前页码大于总页数，返回总页数*

return getTotalPageNo();

} else {

return pageNo;

}

}

public void setPageNo(int pageNo) {

this.pageNo = pageNo;

}

public static int getPageSize() {

return PAGE\_SIZE;

}

*// 总页数是由总记录数和每页显示的条数计算得到*

public int getTotalPageNo() {

if (totalRecord % PAGE\_SIZE == 0) {

return totalRecord / PAGE\_SIZE;

} else {

return totalRecord / PAGE\_SIZE + 1;

}

}

@Override

public String toString() {

return "Page [pageNo=" + pageNo + ", totalPageNo=" + totalPageNo + ", totalRecord=" + totalRecord + "]";

}

}

### **3、创建Druid的数据库连接工具JDBCTools.java**

其中使用到了ThreadLocal,介绍如下：

JDK 1.2的版本中就提供java.lang.ThreadLocal，为解决多线程程序的并发问题提供了一种新的思路。使用这个工具类可以很简洁地编写出优美的多线程程序。通常用来在在多线程中管理共享数据库连接、Session等

ThreadLocal用于保存某个线程共享变量，原因是在Java中，每一个线程对象中都有一个ThreadLocalMap<ThreadLocal, Object>，其key就是一个ThreadLocal，而Object即为该线程的共享变量。而这个map是通过ThreadLocal的set和get方法操作的。对于同一个static ThreadLocal，不同线程只能从中get，set，remove自己的变量，而不会影响其他线程的变量。

1、ThreadLocal.get: 获取ThreadLocal中当前线程共享变量的值。

2、ThreadLocal.set: 设置ThreadLocal中当前线程共享变量的值。

3、ThreadLocal.remove: 移除ThreadLocal中当前线程共享变量的值。

package com.xxj.utils;

import java.io.IOException;import java.sql.Connection;import java.sql.SQLException;import java.util.Properties;

import javax.sql.DataSource;

import com.alibaba.druid.pool.DruidDataSourceFactory;

public class JDBCTools {

private static DataSource dataSource;

private static ThreadLocal<Connection> threadLocal;

static{

try {

*//是一个Map，是Hashtable的子类，它的key和value都是String类型。*

Properties pro = new Properties();

*//如果是加载bin目录下的资源文件（因为src或config目录下的配置文件，最后编译后在bin目录下，和.class在一起），所以用类加载器去加载*

pro.load(JDBCTools.class.getClassLoader().getResourceAsStream("druid.properties"));

dataSource = DruidDataSourceFactory.createDataSource(pro);

threadLocal = new ThreadLocal<Connection>();

} catch (IOException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}

public static Connection getConnection(){

try {

Connection conn = threadLocal.get();

if(conn == null){

conn = dataSource.getConnection();

threadLocal.set(conn);

}

return conn;

} catch (SQLException e) {

*/\**

*\* SQLException编译时，*

*\* RuntimeException：运行时异常，不强制在编译期间编写throws或try...catch*

*\*/*

throw new RuntimeException(e);

}

}

public static void free(){

try {

Connection conn = threadLocal.get();

if(conn!=null){

threadLocal.remove();

conn.close();

}

} catch (SQLException e) {

throw new RuntimeException(e);

}

}}

配置文件：

url=jdbc:mysql://localhost:3306/数据库名

username=用户名

password=密码

driverClassName=com.mysql.jdbc.Driver

initialSize=10

maxActive=50

minIdle=10

maxWait=1000

### **4、创建带泛型的BasicDAO类**

下面使用到了Apache的DButils中的相关技术，介绍如下：

commons-dbutils 是 Apache 组织提供的一个开源 JDBC工具类库，它是对JDBC的简单封装，学习成本极低，并且使用dbutils能极大简化jdbc编码的工作量，同时也不会影响程序的性能。

其中QueryRunner类封装了SQL的执行，是线程安全的。

（1）可以实现增、删、改、查、批处理、

（2）考虑了事务处理需要共用Connection。

（3）该类最主要的就是简单化了SQL查询，它与ResultSetHandler组合在一起使用可以完成大部分的数据库操作，能够大大减少编码量。

****（1）更新****

public int update(Connection conn, String sql, Object… params) throws SQLException:用来执行一个更新（插入、更新或删除）操作。

…

****（2）插入****

public T insert(Connection conn,String sql,ResultSetHandler rsh, Object… params) throws SQLException：只支持INSERT语句，其中 rsh - The handler used to create the result object from the ResultSet of auto-generated keys. 返回值: An object generated by the handler.即自动生成的键值

…

****（3）批处理****

public int[] batch(Connection conn,String sql,Object[][] params)throws SQLException： INSERT, UPDATE, or DELETE语句

public T insertBatch(Connection conn,String sql,ResultSetHandler rsh,Object[][] params)throws SQLException：只支持INSERT语句

…

****（4）使用QueryRunner类实现查询**** 重点

public Object query(Connection conn, String sql, ResultSetHandler rsh,Object… params) throws SQLException：执行一个查询操作，在这个查询中，对象数组中的每个元素值被用来作为查询语句的置换参数。该方法会自行处理 PreparedStatement 和 ResultSet 的创建和关闭。

…

ResultSetHandler接口用于处理 java.sql.ResultSet，将数据按要求转换为另一种形式。ResultSetHandler 接口提供了一个单独的方法：Object handle (java.sql.ResultSet rs)该方法的返回值将作为QueryRunner类的query()方法的返回值。

该接口有如下实现类可以使用：

* BeanHandler：将结果集中的第一行数据封装到一个对应的JavaBean实例中。
* BeanListHandler：将结果集中的每一行数据都封装到一个对应的JavaBean实例中，存放到List里。
* ScalarHandler：查询单个值对象
* MapHandler：将结果集中的第一行数据封装到一个Map里，key是列名，value就是对应的值。
* MapListHandler：将结果集中的每一行数据都封装到一个Map里，然后再存放到List
* ColumnListHandler：将结果集中某一列的数据存放到List中。
* KeyedHandler(name)：将结果集中的每一行数据都封装到一个Map里，再把这些map再存到一个map里，其key为指定的key。
* ArrayHandler：把结果集中的第一行数据转成对象数组。
* ArrayListHandler：把结果集中的每一行数据都转成一个数组，再存放到List中。

package com.xxj.dao.basic;

import java.lang.reflect.ParameterizedType;import java.lang.reflect.Type;import java.sql.SQLException;import java.util.List;import java.util.Map;

import org.apache.commons.dbutils.QueryRunner;import org.apache.commons.dbutils.handlers.BeanHandler;import org.apache.commons.dbutils.handlers.BeanListHandler;import org.apache.commons.dbutils.handlers.MapListHandler;import org.apache.commons.dbutils.handlers.ScalarHandler;

import com.xxj.utils.JDBCTools;

public abstract class BasicDAO<T> {

*//clazz代表BasicDAO的子类的泛型实参类型*

*//例如：class BookDAOImpl extends BasicDAO<Book>，clazz代表<Book>的类型*

private Class clazz;

*//QueryRunner是Apache给封装的类*

private QueryRunner queryRunner = new QueryRunner();

*/\**

*\* 子类继承父类时，不会继承父类的构造器。*

*\* 但是创建子类对象时，一定调用父类的构造器，默认调用父类的无参构造。*

*\*/*

protected BasicDAO(){

*//当创建子类对象时，以下代码会被执行*

*//(1)获取BasicDAO的子类的Class对象*

*//例如：class BookDAOImpl extends BasicDAO<Book>，clazz代表<Book>的类型*

*//subType代表BookDAOImpl的类型*

Class<? extends BasicDAO> subType = this.getClass();

*//(2)获取泛型父类的类型*

*//例如：class BookDAOImpl extends BasicDAO<Book>，clazz代表<Book>的类型*

*//获取BasicDAO<Book>的类型*

Type superType = subType.getGenericSuperclass();

*//BasicDAO<Book>属于ParameterizedType类型*

*//(3)强转为ParameterizedType类型*

ParameterizedType pt = (ParameterizedType) superType;

*//(4)获取实际类型参数*

*//例如：class BookDAOImpl extends BasicDAO<Book>，clazz代表<Book>的类型*

*//获取BasicDAO<Book>的类型中<Book>类型*

Type[] actualTypeArguments = pt.getActualTypeArguments();

*//(5)因为这里只有一个<T>，所以我取数组的[0]元素为clazz赋值*

clazz = (Class) actualTypeArguments[0];

}

protected List<T> getBeanList(String sql, Object... args){

try {

return queryRunner.query(JDBCTools.getConnection(), sql, new BeanListHandler<T>(clazz), args);

} catch (SQLException e) {

throw new RuntimeException(e);

}

}

*//例如：查询总记录数，查询最高工资，查询平均工资等*

@SuppressWarnings("unchecked")

protected Object getSingleValue(String sql, Object... args){

try {

return queryRunner.query(JDBCTools.getConnection(), sql, new ScalarHandler(), args);

} catch (SQLException e) {

throw new RuntimeException(e);

}

}

}

**5、创建BookDAO接口**

package com.xxj.dao;

import java.util.List;

import com.xxj.bean.Book;import com.xxj.bean.Page;

public interface BookDAO {

*/\*\**

*\* 用于查询符合条件的所有的Book对象的数量*

*\* @param args 可以传入最低价格和最高价格*

*\* @return 一共有多少符合规则的记录*

*\*/*

Object getValue(Object... args); *//用于查找符合条件的总记录数*

*/\*\**

*\* 获取带分页的所有图书信息*

*\* @param page 传入的page对象是一个带页码的page对象*

*\* @return 返回的page对象是包含所有属性的page对象*

*\*/*

public Page<Book> getBooksByPage(Page<Book> page);

*/\*\**

*\* 根据价格范围获取带分页的图书信息*

*\* @param page 携带有页码的page*

*\* @param minPrice 最小值*

*\* @param maxPrice 最大值*

*\* @return*

*\*/*

public Page<Book> getPageBooksByPrice(Page<Book> page, double minPrice, double maxPrice);

}

### **6、创建BookDAO接口实现类BookDAOImpl类**

package com.xxj.dao.impl;

import java.util.List;

import com.xxj.bean.Book;import com.xxj.bean.Page;import com.xxj.dao.BookDAO;import com.xxj.dao.basic.BasicDAO;

public class BookDAOImpl extends BasicDAO<Book> implements BookDAO {

*/\**

*\* 根据价钱获取数据库中符合条件的数据总数*

*\*/*

public Object getValue(Object... args) {

String sql = "select count(\*) from books where price between ? and ?";

return getSingleValue(sql, args[0],args[1]);

}

*/\**

*\* 查询指定页码的Book信息*

*\* @param page 一个只有当前页码属性的Page对象（作用是：将PageNo封装为Page对象）*

*\* @return 返回一个List属性为【当前页码的所有的Book对象】的Page对象*

*\*/*

@Override

public Page<Book> getBooksByPage(Page<Book> page) {

*//这里注意一个小细节：当我们查询的时候，如果表的字段名和bean中的不一致需要取别名*

String sql = "SELECT id, title, author, price,`sales`,`stock`,img\_path imgPath FROM books LIMIT ?, ?";

*// 这里将最低价格赋值为0，最高价格为Integer的最大值。*

*//这样就能查询出所有的Book对象*

*//这里需要注意的是如果不转化为Long，则会报类型转换异常且必须强转为Long*

long totalNum = (Long)getValue(0,Integer.MAX\_VALUE);

*// 设置数据库中一共有多少条记录*

page.setTotalRecord((int)totalNum);

int totalPageNo = (int) (totalNum / Page.PAGE\_SIZE );

page.setTotalPageNo(totalPageNo);

*// 查询出当前页码中的所有的Book对象*

List<Book> bookList = getBeanList(sql,(page.getPageNo()-1)\*Page.PAGE\_SIZE,Page.PAGE\_SIZE);

*// 赋值给page对象的List属性*

page.setList(bookList);

*//返回page对象*

return page;

}

@Override

public Page<Book> getPageBooksByPrice(Page<Book> page, double minPrice, double maxPrice) {

String sql = "SELECT id, title, author, price,sales,stock,img\_path imgPath FROM books WHERE price BETWEEN ? AND ? LIMIT ?, ?";

*// 获取数据库中的记录数*

long totalRecord = (Long) getValue(minPrice, maxPrice);

*// 给一共有多少记录属性赋值*

page.setTotalRecord((int)totalRecord);

*// 得到LIMIT的第一个参数*

int a = (page.getPageNo() - 1) \* Page.PAGE\_SIZE;

*//防止当page.getPageNo()为0时，a是一个负值，造成SQL异常*

if(a < 0) {

a = 0;

}

List<Book> books = getBeanList(sql, minPrice, maxPrice, a, Page.PAGE\_SIZE);

*// 将查询出结果赋予Page对象*

page.setList(books);

return page;

}

}

### **7、创建BookService接口**

package com.xxj.service;

import java.util.List;

import com.xxj.bean.Book;import com.xxj.bean.Page;

public interface BookService {

*//分页查找图书的方法*

public Page<Book> getBooksByPage(String pageNo);

*//通过价钱分页查赵图书*

public Page<Book> getPageBooksByPrice(String pageNo, String minPrice, String maxPrice);}

### **8、创建BookService接口实现类BookServiceImpl类**

package com.xxj.service.impl;

import java.util.List;

import com.xxj.bean.Book;import com.xxj.bean.Page;import com.xxj.dao.BookDAO;import com.xxj.dao.impl.BookDAOImpl;import com.xxj.service.BookService;

public class BookServiceImpl implements BookService {

private BookDAO bookDAO = new BookDAOImpl();

@Override

public Page<Book> getBooksByPage(String pageNo) {

*//当用户什么都不输入，默认为1，为首页*

int currentPageNo = 1;

*//接收当前页并转为整型。当用户输入的不合法。如：abs，捕获处理异常*

try {

currentPageNo = Integer.parseInt(pageNo);

}catch (NumberFormatException e) {

}

*// 创建一个Page对象*

Page<Book> page = new Page<Book>();

*// 只给当前页码属性赋值*

page.setPageNo(currentPageNo);

*// 返回一个包含了所有的属性值的Page对象*

return bookDAO.getBooksByPage(page);

}

@Override

public Page<Book> getPageBooksByPrice(String pageNoStr, String minPrice, String maxPrice) {

*// 默认页码*

int pageNo = 1;

*// 默认最低价格*

Double min = 0.0;

*// 默认最高价格*

Double max = Double.MAX\_VALUE;

*//以防出现异常，捕获到异常也不打印，让它们失效，使用默认值*

try {

pageNo = Integer.parseInt(pageNoStr);

min = Double.parseDouble(minPrice);

max = Double.parseDouble(maxPrice);

} catch (NullPointerException e1) {

} catch (NumberFormatException e2) {

}

*// 如果最低价格大于最高价格，交换，以防用户最大值和最小值填写错误位置*

if (min > max) {

min += max;

max = min - max;

min = min - max;

}

Page<Book> page = new Page<Book>();

*// 赋予当前页码值*

page.setPageNo(pageNo);

return bookDAO.getPageBooksByPrice(page, min, max);

}

}

其就是一个普通的继承自HttpServlet的Servlet，doPost调用doGet。在doGet中，获取方法名的请求参数，通过反射来调用继承自BaseServlet的子Servlet的相应方法

因为每个servlet都要执行doPost，doGet方法处理相关请求，我们就使用动态反射简化代码。之后我们只用指定相关的方法处理我们的请求即可。

package com.xxj.servlet;

import java.io.IOException;import java.lang.reflect.Method;

import javax.servlet.ServletException;import javax.servlet.http.HttpServlet;import javax.servlet.http.HttpServletRequest;import javax.servlet.http.HttpServletResponse;

public class BaseServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

*//设置编码字符集*

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("text/html;charset=UTF-8");

*//获取方法名*

String method = request.getParameter("method");

try {

*//获取方法的对象*

Method methodObj = this.getClass().getDeclaredMethod(method, HttpServletRequest.class,HttpServletResponse.class);

*//通过方法的对象调用具体的方法*

methodObj.invoke(this, request,response);

} catch (Exception e) {

e.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}}

### **10、创建BaseServlet的子类**

package com.xxj.servlet;import java.io.IOException;import java.util.List;import javax.servlet.Servlet;import javax.servlet.ServletException;import javax.servlet.http.HttpServletRequest;import javax.servlet.http.HttpServletResponse;import com.xxj.bean.Book;import com.xxj.bean.Page;import com.xxj.service.BookService;import com.xxj.service.impl.BookServiceImpl;*/\*\**

*\*这个是用来处理普通查询*

*\*/*

public class BookManagerServlet extends BaseServlet{

private static final long serialVersionUID = 1L;

private BookService bookService = new BookServiceImpl();

public void getPageBook(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{

*//获取当前页的值*

String pageNo = request.getParameter("pageNo");

*//拿到当前页的所有对象*

Page<Book> pageBook = bookService.getBooksByPage(pageNo);

*//放到request域中*

request.setAttribute("pageBook", pageBook);

*//进行转发到显示页面*

request.getRequestDispatcher("/pages/manager/book\_manager.jsp").forward(request, response);

}}

package com.xxj.servlet;

import java.io.IOException;import javax.servlet.Servlet;import javax.servlet.ServletException;import javax.servlet.http.HttpServletRequest;import javax.servlet.http.HttpServletResponse;import com.xxj.bean.Book;import com.xxj.bean.Page;import com.xxj.service.BookService;import com.xxj.service.impl.BookServiceImpl;*/\*\**

*\*这个是用来通过价格查询进行分页*

*\*/*public class BookClientServlet extends BaseServlet{

private static final long serialVersionUID = 1L;

private BookService bookService = new BookServiceImpl();

*//通过价钱查询页面的book对象*

public void getClientBooksByPageAndPrice(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{

*//设置字符集*

request.setCharacterEncoding("UTF-8");

*//取值*

String pageNoStr = request.getParameter("pageNo");

String minPrice = request.getParameter("min");

String maxPrice = request.getParameter("max");

*//调用service层的方法获取page对象*

Page<Book> page = bookService.getPageBooksByPrice(pageNoStr, minPrice, maxPrice);

*//加入域中*

request.setAttribute("page", page);

*//转发*

request.getRequestDispatcher("/pages/client/book\_client.jsp").forward(request, response);

}}

1. **JSP页面的编写**

**1、编写index.jsp**

里面调用JSP的动作标签，转发到BookClientServlet中，并携带方法的请求参数

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<jsp:forward page="/BookClientServlet?method=getClientBooksByPageAndPrice"></jsp:forward>

### **2、重用页面**

base.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%-- <base href="<%=request.getScheme()%>://<%=request.getServerName()%>:<%=request.getServerPort()%><%=request.getContextPath() %>/"> --%>

<!--获取的项目根路径-->

<base href="<%=request.getContextPath() %>/">

<!--引入样式-->

<link type="text/css" rel="stylesheet" href="static/css/style.css" >

<!--引入js库-->

<script type="text/javascript" src="static/script/jquery-1.7.2.js"></script>

<!--导入标签库-->

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

header.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<div>

<a href="BookManagerServlet?method=getPageBook">图书管理</a>

<a href="index.jsp">返回商城</a>

</div>

welcome.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:if test="${empty sessionScope.user }">

<div>

<a href="pages/user/login.jsp">登录</a> |

<a href="pages/user/regist.jsp">注册</a> &nbsp;&nbsp;

<a href="pages/manager/manager.jsp">后台管理</a>

</div>

</c:if>

<c:if test="${not empty sessionScope.user }">

<div>

<span>欢迎<span class="um\_span">${sessionScope.user.username }</span>光临书城</span>

<a href="UserServlet?method=logout">注销</a>&nbsp;&nbsp;

<a href="index.jsp">返回</a>

</div>

</c:if>

### **3、编写显示查询结果的页面book\_manager.jsp**

#### **（1）页码导航栏的编写【普通的页码导航栏】**

① 需要有首页，前一页，当前页 - 1，当前页，当前页 + 1，下一页，末页。这些都是a标签  
② 需要提示用户一共有多少条记录，一共有几页。并提供输入框，用于输入要跳转的页码。并提供一个跳转的按钮

分析：  
⒈ 首页的href可以直接写方法的请求参数  
⒉ 前一页 和 当前页 - 1，需要判断当前页码是否为1：如果当前页码为1，则需要将这两个a标签变为文字显示；否则就显示a标签  
⒊ 同理：下一页 和 当前页 + 1，的显示和隐藏替换和上面类似  
⒋ 末页的href的pageNo请求参数就是一共有多少页码  
⒌ 一共有多少条记录，和一共有多少页。都可以通过page对象来获取  
⒍ 提供一个输入框，添加一个id，方便获取其value值  
⒎ 提供一个按钮，添加一个id，方便添加点击事件  
⒏ 按钮的点击事件：首先获取输入跳转页码的输入框value属性值，然后调用window.location.href = 要跳转的页面，来实现要跳转的页面.可以直接转化为location=要跳转的页面，来实现要跳转的页面

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>图书管理</title>

<%@ include file="/WEB-INF/include/base.jsp" %>

<script type="text/javascript">

$(function(){

$(".delbook").click(function(){

var title = $(this).attr("name");

var result = confirm("确认删除[ "+title+" ]吗?");

if(result==false){

return false;

}

});

$("#goBtn").click(function(){

var goPageNo = $("#goPageNo").val();

location = "BookManagerServlet?method=getPageBook&pageNo=" + goPageNo;

});

});

</script>

</head>

<body>

<div id="header">

<span class="wel\_word">图书管理系统</span>

<%@ include file="/WEB-INF/include/header.jsp" %>

</div>

<div id="main" style="position: relative;">

<table>

<tr>

<td>名称</td>

<td>价格</td>

<td>作者</td>

<td>销量</td>

<td>库存</td>

<td>图片路径</td>

<td colspan="2">操作</td>

</tr>

<!--遍历list中的book-->

<c:forEach items="${requestScope.pageBook.list}" var="book">

<tr>

<td>${book.title}</td>

<td>${book.price}</td>

<td>${book.author}</td>

<td>${book.sales}</td>

<td>${book.stock}</td>

<td>${book.imgPath}</td>

<td><a href="BookManagerServlet?method=editBook&&id=${book.id}">修改</a></td>

<td><a name="${book.title}" class="delbook" href="BookManagerServlet?method=deleteBook&&id=${book.id}" >删除</a></td>

</tr>

</c:forEach>

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td colspan="2" align="center"><a href="pages/manager/book\_edit.jsp" >添加图书</a></td>

</tr>

</table>

<br><br>

<!--分页的导航栏-->

<div style="position:absolute; bottom: 10px;left:290px;">

<%-- 当前页码大于1 --%>

<c:if test="${requestScope.pageBook.pageNo > 1}">

<a href="BookManagerServlet?method=getPageBook">首页</a>&emsp;

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.pageNo - 1}">上一页</a>&emsp;

</c:if>

<%-- 当前页码等于1 --%>

<c:if test="${requestScope.pageBook.pageNo == 1}">

<a href="BookManagerServlet?method=getPageBook">首页</a>&emsp;

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.pageNo - 1}">上一页</a>&emsp;

</c:if>

<a style="color:red; text-decoration:none;" href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.pageNo}">【${requestScope.pageBook.pageNo}/${requestScope.pageBook.totalPageNo}】</a>&emsp;

<%-- 当前页码小于总页码 --%>

<c:if test="${requestScope.pageBook.pageNo < requestScope.pageBook.pageNo}">

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.pageNo + 1}">下一页</a>&emsp;

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.totalPageNo}">末页</a>&emsp;

</c:if>

<%-- 当前页码等于总页码 --%>

<c:if test="${requestScope.pageBook.pageNo == requestScope.pageBook.pageNo}">

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.pageNo + 1}">下一页</a>&emsp;

<a href="BookManagerServlet?method=getPageBook&pageNo=${requestScope.pageBook.totalPageNo}">末页</a>&emsp;

</c:if>

<span>共<strong>&nbsp;${requestScope.pageBook.totalPageNo}&nbsp;</strong>页</span>&emsp;

<span>共<strong>&nbsp;${requestScope.pageBook.totalRecord}&nbsp;</strong>条记录</span>&emsp;

<span>跳转到第&nbsp;<input style="text-align: center;color:green;" id="goPageNo" type="text" name="pageNo" value="${requestScope.pageBook.pageNo}" size="1"/>&nbsp;页</span>&emsp;

<input type="button" id="goBtn" value="确定"/>

</div>

</div>

<div id="bottom" >

<span>

书城.Copyright &copy;2019

</span>

</div>

</body>

</html>

### **4、编写带有价格的查询页面book\_client.jsp**

#### **（2）页码导航栏的编写【动态的页码导航栏】**

最终实现的页码导航栏的效果是：指定一次显示5个页码，尽量让当前页码在中间位置

分析：

totalPageNo<5

begin:1

end:totalPageNo

totalPageNo>=5

pageNo<=3

begin:1

end:5

【1】2345

1【2】345

12【3】45

pageNo>3&&pageNo<=totalPageNo-2

begin:pageNo-2

end:pageNo+2

23【4】56

34【5】67

45【6】78

pageNo>totalPageNo-2

begin:totalPageNo-4

end:totalPageNo

456【7】8

4567【8】

实现上述的集中情况要使用到jstl的标签库：

<c:choose>

<c:when test="当前页码所属的范围">

<c:set var="起始页码" value="起始页码值"></c:set>

<c:set var="结束页码" value="结束页码值"></c:set>

</c:when>

</c:choose>

代码：

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>书城首页</title>

<%@include file="/WEB-INF/include/base.jsp" %>

<script type="text/javascript">

$(function(){

$("#page\_nav :button,.book\_cond :button").click(function(){

var pageNo = $("#pn\_input").val();

var min = $("input[name='min']").val();

var max = $("input[name='max']").val();

//使用location去BookClientServlet请求

location = "BookClientServlet?method=getClientBooksByPageAndPrice&pageNo="

+pageNo+"&min="+min+"&max="+max;

});

});

</script>

</head>

<body>

<!-- http://localhost:8080<br> -->

<%-- <%=request.getScheme()%>://<%=request.getServerName()%>:<%=request.getServerPort()%><%=request.getContextPath() %> --%>

<div id="header">

<span class="wel\_word">网上书城</span>

<%@include file="/WEB-INF/include/welcome.jsp" %>

</div>

<div id="main">

<div id="book">

<br>

<div class="book\_cond">

价格：<input type="text" name="min" value="${param.min }"> 元 -

<input type="text" name="max" value="${param.max }"> 元

<button>查询</button>

</div>

<br>

<c:forEach var="book" items="${requestScope.page.list }">

<div class="b\_list">

<div class="img\_div">

<img class="book\_img" alt="" src="${book.imgPath}" />

</div>

<div class="book\_info">

<div class="book\_name">

<span class="sp1">书名:</span>

<span class="sp2">${book.title }</span>

</div>

<div class="book\_author">

<span class="sp1">作者:</span>

<span class="sp2">${book.author }</span>

</div>

<div class="book\_price">

<span class="sp1">价格:</span>

<span class="sp2">￥${book.price }</span>

</div>

<div class="book\_sales">

<span class="sp1">销量:</span>

<span class="sp2">${book.sales }</span>

</div>

<div class="book\_amount">

<span class="sp1">库存:</span>

<span class="sp2">${book.stock }</span>

</div>

<div class="book\_add">

<button>加入购物车</button>

</div>

</div>

</div>

</c:forEach>

</div>

<!--导航栏动态分页的实现-->

<c:choose>

<c:when test="${page.totalPageNo<5 }">

<c:set var="begin" value="1"></c:set>

<c:set var="end" value="${page.totalPageNo }"></c:set>

</c:when>

<c:when test="${page.pageNo<=3 }">

<c:set var="begin" value="1"></c:set>

<c:set var="end" value="5"></c:set>

</c:when>

<c:when test="${page.pageNo>3&&page.pageNo<=page.totalPageNo-2 }">

<c:set var="begin" value="${page.pageNo-2 }"></c:set>

<c:set var="end" value="${page.pageNo+2 }"></c:set>

</c:when>

<c:otherwise>

<c:set var="begin" value="${page.totalPageNo-4 }"></c:set>

<c:set var="end" value="${page.totalPageNo }"></c:set>

</c:otherwise>

</c:choose>

<div id="page\_nav">

<c:forEach var="i" begin="${begin}" end="${end}" step="1">

<c:if test="${i == page.pageNo }">

<span style="color:red;">【<strong style="color: blue;">${i}</strong>】</span>

</c:if>

<c:if test="${i != page.pageNo }">

<a href="BookClientServlet?method=getClientBooksByPageAndPrice&pageNo=${i }&min=${param.min}&max=${param.max}">${i }</a>

</c:if>

</c:forEach>

&nbsp;&nbsp;共&nbsp;<strong id="pag" style="color:red;">${requestScope.page.pageNo }</strong>/<strong style="color: blue;">${requestScope.page.totalPageNo }</strong>&nbsp;页&nbsp;&nbsp;<strong style="color:red;">${requestScope.page.totalRecord }</strong> &nbsp;条记录&nbsp;&nbsp;

跳到第<input style="color: purple;" value="${requestScope.page.pageNo }" name="pn" id="pn\_input"/>页

<input type="button" value="确定">

</div>

</div>

<div id="bottom">

<span>

书城.Copyright &copy;2019

</span>

</div>

</body>

</html>

## **最终实现的效果**

## IMG_256