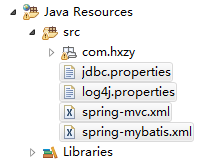
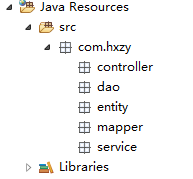
# 配置文件：



# 项目目录：



# 查询操作：

## 1、创建Page、book、user实体bean：

**private** **int** offset;//查询起始位置：0[无限制]

**private** **int** limit;//偏移量：0[无限制]

**private** **int** count;//总条数

**public** Page(){}

**public** Page(**int** offset, **int** limit) {

**super**();

**this**.offset = offset;

**this**.limit = limit;

}

Get、set……

## 2、创建DAO接口：数据持久层

**public** **interface** BookDataBaseDao {

**public** List<Book> queryBookListByPage(Page page);

}

不需要实现类

## 3、创建mapper配置文件：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"com.hxzy.dao.BookDataBaseDao"*>

<!-- 对象参数传递到了语句中，id、username 和 password 属性将会被查找，然后将它们的值传入预处理语句的参数中 -->

<select id=*"queryBookListByPage"* resultType=*"com.hxzy.entity.Book"*

parameterType=*"com.hxzy.entity.Page"*>

select `id`,`name`,`author`,`publish`,left(publish\_date,10) as publishDate,`page`,`price`,`content`

from `book`

<if test=*"offset>=0 and limit!=0"*>

limit #{offset},#{limit}

</if>

</select>

</mapper>

## 4、创建Service（接口）用来申明增删改查得方法：

**public** **interface** BookDataBaseService {

**public** List<Book> queryBookListByPage(Page page);

}

## 5、实现Service接口：

//表示的Service层，如果不写Spring就找不到

@Service

**public** **class** BookDataBaseServiceImpl **implements** BookDataBaseService {

//表示Spring的自动注入

//依赖注入（DI）：依赖【需要某个类对象】

//,注入【Spring会自动创建并且注入到这个类对象文件中】

@Resource

**private** BookDataBaseDao dao;

/\*\*

\* 请求一页数据，根据page对象里面设置的起始位置和偏移量

\* **@return** 查询到的数据列表

\*/

**public** List<Book> queryBookListByPage(Page page) {

//第一次查询，只获取总条数

List<Book> bs = dao.queryBookListByPage(**new** Page());

page.setCount(bs.size());

//第二次查询，查询指定的N条

**return** dao.queryBookListByPage(page);

}

}

## 6、创建Controller：相当于servlet：

/\*\*

\* spring mvc处理方法支持的返回方式：ModelAndView, Model, ModelMap, Map,View, String, void

\* ModelAndView构造方法可以指定返回的页面名称，也可以通过setViewName()方法跳转到指定的页面 ,

\* 使用addObject()设置需要返回的值，addObject()有几个不同参数的方法，可以默认和指定返回对象的名字。

\* 调用addObject()方法将值设置到一个名为ModelMap的类属性，ModelMap是LinkedHashMap的子类

\* **@author** STAR

\*

\*/

@ResponseBody

@Controller

@Scope("prototype")

**public** **class** BookDataBaseController {

@Resource

**private** BookDataBaseService service;

/\*\*

\* 获取一页数据

\* **@param** session

\* **@param** myoffset 起始页

\* **@param** mylimit 请求条数

\* **@param** map

\* **@return**

\*/

@RequestMapping(value = "/booksPage.do")

**public** Map<String,Object> toBookListPage(HttpSession session, Page page, ModelMap map) {

List<Book> list = **null**;

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

list = service.queryBookListByPage(page);

result.put("total",page.getCount());

result.put("rows",list);

**return** result;

}

}

# 删除操作：

## 2、创建持久层

**public** **interface** BookDataBaseDao {

**public** List<Book> queryBookListByPage(Page page);

**public** **int** deleteBook(Integer id);

}

## 3、编写Mapper

<mapper namespace=*"com.hxzy.dao.BookDataBaseDao"*>

<!-- 对象参数传递到了语句中，id、username 和 password 属性将会被查找，然后将它们的值传入预处理语句的参数中 -->

<select id=*"queryBookListByPage"* resultType=*"com.hxzy.entity.Book"*

parameterType=*"com.hxzy.entity.Page"*>

select `id`,`name`,`author`,`publish`,left(publish\_date,10) as publishDate,`page`,`price`,`content`

from `book`

<if test=*"offset>=0 and limit!=0"*>

limit #{offset},#{limit}

</if>

</select>

<!-- 删除一本书 -->

<delete id=*"deleteBook"* parameterType=*"int"*>

DELETE FROM `book` WHERE id = #{id}

</delete>

</mapper>

## 4、编写service层接口

**public** **interface** BookDataBaseService {

**public** List<Book> queryBookListByPage(Page page);

**public** **int** deleteBook(Integer id);

}

## 5、实现service层接口

/\*\*

\* 删除指定ID的book

\*/

@Override

**public** **int** deleteBook(Integer id) {

**return** dao.deleteBook(id);

}

## 6、编写controller层

/\*\*

\* 删除数据

\* **@param** session

\* **@param** book

\* **@param** map

\* **@return**

\*/

@RequestMapping(value = "/bookDelete.do")

**public** String delete(HttpSession session, HttpServletRequest resq, ModelMap map) {

Object obj = session.getAttribute("loginUser");

**if** (obj == **null**) {

map.put("message", "登入已过期，请重新登入！");

//return "book\_db/index";

}

String[] ids = resq.getParameterValues("id");

**for** (String id : ids) {

**if** (service.deleteBook(**new** Integer(id)) == 1) {

System.***out***.println("数据删除成功：ID="+id);

**return** "{\"status\":\"success\",\"data\":\"数据删除成功\"}";

}

}

**return** "{\"status\":\"\",\"data\":\"\"}";

}

# 插入操作：

## 2、在BookDataBaseDao.java文件中增加方法申明：

**public** **int** insertBook(Book book);

## 3、在BookDataBaseMapper.xml文件中增加插入的SQL语句

<!-- 插入一本书 -->

<insert id=*"insertBook"* parameterType=*"com.hxzy.entity.Book"*>

insert into `book` ( `name`,`author`,`publish`,`publish\_date`,`page`,`price`,`content`

)

values

( #{name},#{author},#{publish},#{publishDate},#{page},#{price},#{content}

)

</insert>

## 4、在BookDataBaseService.java文件中增加方法申明

**public** **int** insertBook(Book book);

## 5、在BookDataBaseServiceImpl.java文件中增加实现方法

/\*\*

\* 插入一条数据

\*/

@Override

**public** **int** insertBook(Book book) {

**return** dao.insertBook(book);

}

## 6、在BookDataBaseController.java文件中增加插入方法

/\*\*

\* 插入一条数据

\* **@param** session

\* **@param** book

\* **@param** map

\* **@return**

\*/

@RequestMapping(value = "/bookInsert.do",produces="text/html;charset=UTF-8")

**public** String bookUpdate(HttpSession session, Book book, ModelMap map) {

Object obj = session.getAttribute("loginUser");

**if** (obj == **null**) {

map.put("message", "登入已过期，请重新登入！");

//return "book\_db/index";

}

**if**(service.insertBook(book) == 1){

System.***out***.println("数据增加成功："+book.toString());

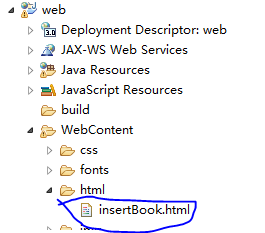
**return** "<h2>数据增加成功</h2>";

}

**return** "{\"status\":\"success\",\"data\":\"\"}";

}

## 修改前端访问地址：



如上图文件，打开



把action="/m/bookUpdate.do"改成：action="/web/bookInsert.do"

# 更新操作：

## 2、在BookDataBaseDao.java文件中增加方法申明：

## public int updateBook(Book book);

## 3、在BookDataBaseMapper.xml文件中增加插入的SQL语句

<!-- 更新字段 -->

<update id=*"updateBook"* parameterType=*"com.hxzy.entity.Book"*>

update `book`

<set>

<if test=*"name!=null"*>

`name`=#{name},

</if>

<if test=*"author!=null"*>

`author`=#{author},

</if>

<if test=*"publish!=null"*>

`publish`=#{publish},

</if>

<if test=*"publishDate!=null"*>

`publish\_date`=#{publishDate},

</if>

<if test=*"page!=null"*>

`page`=#{page},

</if>

<if test=*"price!=null"*>

`price`=#{price},

</if>

<if test=*"content!=null"*>

`content`=#{content},

</if>

</set>

where id = #{id}

</update>

## 4、在BookDataBaseService.java文件中增加方法申明

**public** **int** updateBook(Book book);

## 5、在BookDataBaseServiceImpl.java文件中增加实现方法

/\*\*

\* 更新字段

\*/

@Override

**public** **int** updateBook(Book book) {

**return** dao.updateBook(book);

}

## 6、在BookDataBaseController.java文件中增加插入方法

/\*\*

\* 更新一条数据

\* **@param** session

\* **@param** book

\* **@param** map

\* **@return**

\*/

@RequestMapping(value = "/bookUpdate.do",produces="text/html;charset=UTF-8")

**public** String bookUpdate(HttpSession session, Book book, ModelMap map) {

Object obj = session.getAttribute("loginUser");

**if** (obj == **null**) {

map.put("message", "登入已过期，请重新登入！");

//return "book\_db/index";

}

**if**(service.updateBook(book) == 1){

System.***out***.println("数据更新成功："+book.toString());

**return** "<h2>数据更新成功</h2>";

}

**return** "{\"status\":\"success\",\"data\":\"\"}";

}

# 登陆：

## 2、在UserDataBaseDao.java文件中增加方法申明：

**public** User queryUser(User user);

## 3、在UserDataBaseMapper.xml文件中增加插入的SQL语句

<select id=*"queryUser"* resultType=*"com.hxzy.entity.User"*

parameterType=*"com.hxzy.entity.User"*>

select `id`,`name`,`password`,`age`,`sex`,`nickname`,`usertype`

from `user`

<where>

<if test=*"name!=null and password!=null"*>

and `name` = #{name}

and `password` = #{password}

</if>

</where>

</select>

## 4、在UserDataBaseService.java文件中增加方法申明

**public** User queryUser(User user);

## 5、在USerDataBaseServiceImpl.java文件中增加实现方法

@Override

**public** User queryUser(User user) {

**return** dao.queryUser(user);

}

## 6、在USerDataBaseController.java文件中增加插入方法

@RequestMapping(value = "/login.do")

**public** String updateUserPassword(HttpSession session, User user, ModelMap map) {

User u = service.queryUser(user);

**if** (u == **null**) {

map.put("name", user.getName());

map.put("message", "用户名或密码错误！");

**return** "book\_db/index";

}

session.setAttribute("loginUser", u);

**return** "redirect:/web/index.jsp";

}

## 登陆跳转：

**return** "forward:index.jsp";

## 前端重定向：

<%@page contentType=*"text/html"*%>

<%@page pageEncoding=*"UTF-8"*%>

<%@page session=*"true"*%>

<%@page import=*"com.hxzy.entity.User"* %>

<%

String path = request.getContextPath();

String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort() + path + "/";

%>

<%

User user= (User)session.getAttribute("loginUser");

**if**(user==**null** || user.getName() == **null**)

{

response.sendRedirect(basePath+"jsp/login.jsp");

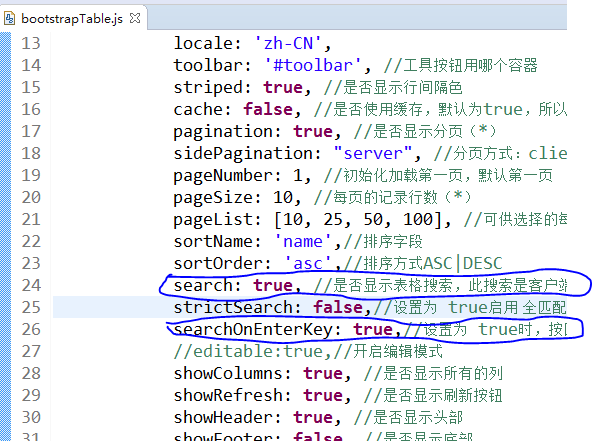
**return**;

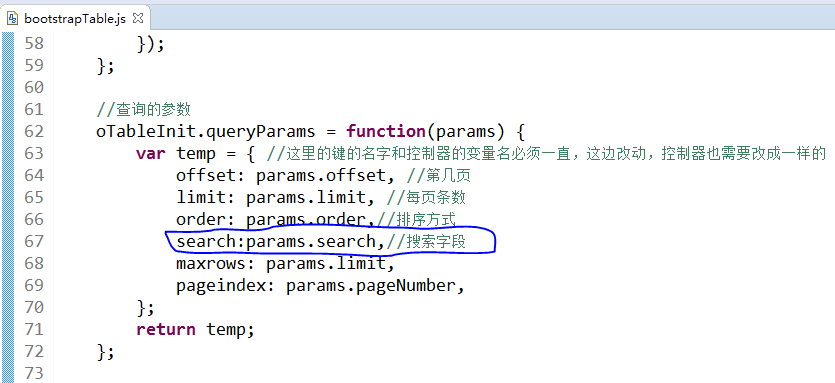
}

%>

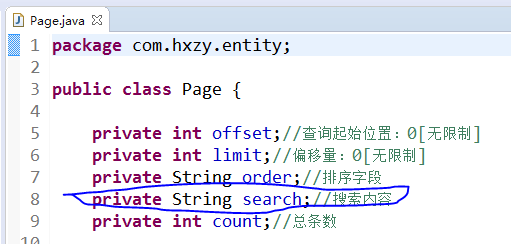
# 搜索：

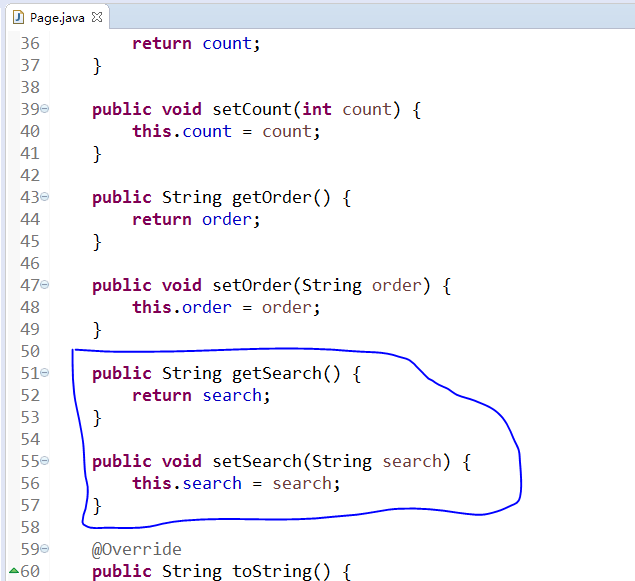
## 前端：





## 后端：



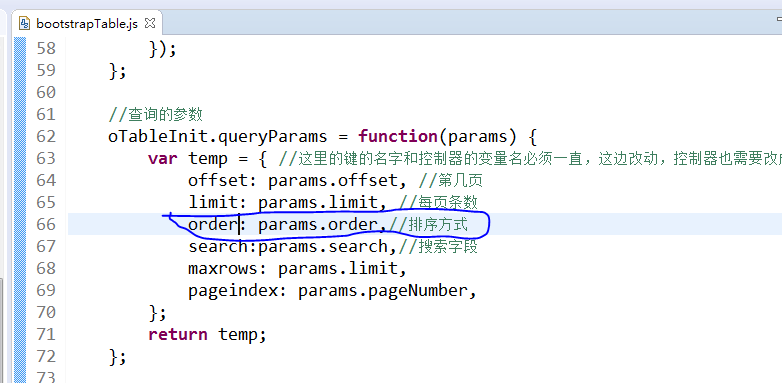


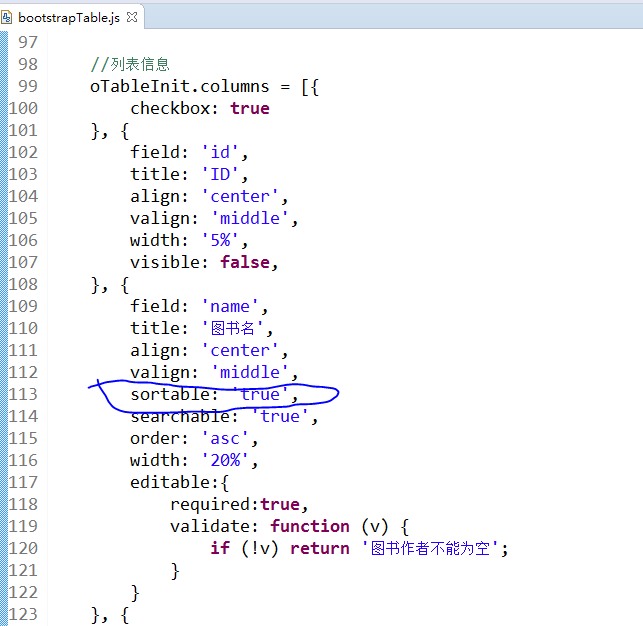




# 排序:

## 前端：





## 后端：

