综合案例struts2+hibernate+datagrid

Book 实体类及注解

Java代码

```java
package cn.entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
/**
 * BOOK 实体类
 */
@Entity
@Table(name="BOOK")
public class Book {
    @Id
    @GeneratedValue(generator="BOOK_SEQ",strategy=GenerationType.SEQUENCE)
    @SequenceGenerator(name="BOOK_SEQ",sequenceName="BOOK_SEQ",allocationSize=1,initialValue=1)
    private int id;//编号
    @Column
    private String isbn;//书号
    @Column
    private String title;//标题
    @Column
    private double price;//价格
    @Column
    private Date pubdate;//出版日期
    @Column
    private String intro;//简介
    public Book() {
    }
    public Book(String isbn, String title, double price, Date pubdate,
            String intro) {
        this.isbn = isbn;
        this.title = title;
        this.price = price;
        this.pubdate = pubdate;
        this.intro = intro;
    }

    public Book(int id, String isbn, String title, double price, Date pubdate,
            String intro) {
        this.id = id;
        this.isbn = isbn;
        this.title = title;
        this.price = price;
        this.pubdate = pubdate;
        this.intro = intro;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getIsbn() {
        return isbn;
    }
    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
```

```
        this.title = title;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public Date getPubdate() {
        return pubdate;
    }
    public void setPubdate(Date pubdate) {
        this.pubdate = pubdate;
    }
    public String getIntro() {
        return intro;
    }
    public void setIntro(String intro) {
        this.intro = intro;
    }
}
```

BookDao 数据访问层

Java代码

```
package cn.dao;

import java.util.List;

import cn.entity.Book;
/**
 * Book 数据访问层接口
 * */
public interface BookDao {
    /**
     * 查询共有多少记录
     *
     * @return
     */
    public long findTotal();

    /**
     * 查询一页的数据
     *
     * @param begin 从哪条开始 0
     * @param end 得到多少条
     * @param sort 排序的列
     * @param order 排序的方式 desc/asc
     * @return
     */
    public List<Book> findPageBooks(final int begin, final int end, final String sort, final String order);

    /**
     * 增加一条数据
     *
     * @param book 传来的参数不包括 id
     * @return
     */
    public int addBook(Book book);

    /**
     * 删除一条数据
     *
     * @param id 根据 id 作删除
     * @return
     */
    public int deleteBook(int id);

    /**
     * 修改一条数据
     *
     * @param book 传来的参数包括 id
     * @return
```

```
     */
    public int updateBook(Book book);
}
```

Java代码

```
package cn.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.criterion.Order;

import cn.dao.BookDao;
import cn.entity.Book;
import cn.util.HibernateSessionFactory;

/**
 * Book 数据访问层实现
 * */
public class BookDaoImpl implements BookDao {

    /**
     * 查询共有多少记录
     *
     * @return
     */
    public long findTotal(){
        Session session=HibernateSessionFactory.getSession();
        Long count = (Long)session.createQuery("select count(*) from Book").uniqueResult();
        return count;
    }

    /**
     * 查询一页的数据
     *
     * @param begin 从哪条开始 0
     * @param end 得到多少条
     * @param sort 排序的列
     * @param order 排序的方式 desc/asc
     * @return
     */
    public List<Book> findPageBooks(final int begin, final int end, final String sort, final String order){
        Session session=HibernateSessionFactory.getSession();
        Criteria criteria = session.createCriteria(Book.class);
        if("desc".equals(order)){
            criteria.addOrder(Order.desc(sort));
        }else{
            criteria.addOrder(Order.asc(sort));
        }
        criteria.setFirstResult(begin).setMaxResults(end);
        List<Book> books = criteria.list();
        HibernateSessionFactory.closeSession();
        return books;
    }

    /**
     * 增加一条数据
     *
     * @param book 传来的参数不包括 id
     * @return
     */
    public int addBook(Book book){
        Session session=HibernateSessionFactory.getSession();
        Transaction tx=null;
        int id = 0;
        try {
            tx=session.beginTransaction();
            id = (Integer)session.save(book);
            tx.commit();
        } catch (HibernateException e) {
```

```
            e.printStackTrace();
            tx.rollback();
        }finally{
            HibernateSessionFactory.closeSession();
        }
        return id;
    }

    /**
     * 删除一条数据
     *
     * @param id 根据 id 作删除
     * @return
     */
    public int deleteBook(int id){
        Session session=HibernateSessionFactory.getSession();
        Transaction tx = null;
        int num = 0;
        try {
            tx=session.beginTransaction();
            Book book = new Book();
            book.setId(id);
            session.delete(book);
            tx.commit();
            num = 1;
        } catch (HibernateException e) {
            e.printStackTrace();
            tx.rollback();
        }finally{
            HibernateSessionFactory.closeSession();
        }
        return num;
    }

    /**
     * 修改一条数据
     *
     * @param book 传来的参数包括 id
     * @return
     */
    public int updateBook(Book book){
        Session session=HibernateSessionFactory.getSession();
        Transaction tx = null;
        int id = 0;
        try {
            tx=session.beginTransaction();
            session.update(book);
            tx.commit();
            id = 1;
        } catch (HibernateException e) {
            e.printStackTrace();
            tx.rollback();
        }finally{
            HibernateSessionFactory.closeSession();
        }
        return id;
    }

}
```

BookBiz 业务逻辑层

Java代码

```
package cn.biz;

import java.util.List;

import cn.entity.Book;

/**
 * Book 业务逻辑层接口
 * */
public interface BookBiz {
```

```
11.
12.     public long findTotal();
13.
14.     /**
15.      * 查询一页的数据
16.      *
17.      * @param pageIndex 从哪页开始
18.      * @param pageSize 得到的页数
19.      * @param sort 排序的列
20.      * @param order 排序的方式 desc/asc
21.      * @return
22.      */
23.     public List<Book> findPageBooks(final int pageIndex, final int pageSize, final String sort, final String
    order);
24.
25.     public int addBook(Book book);
26.
27.     public int deleteBook(int id);
28.
29.     public int updateBook(Book book);
30. }
```

Java代码

```
1.  package cn.biz.impl;
2.
3.  import java.util.List;
4.
5.  import cn.biz.BookBiz;
6.  import cn.dao.BookDao;
7.  import cn.dao.impl.BookDaoImpl;
8.  import cn.entity.Book;
9.
10. /**
11.  * Book 业务逻辑层实现
12.  * */
13. public class BookBizImpl implements BookBiz {
14.
15.     private BookDao bookDao = new BookDaoImpl();
16.
17.     public long findTotal() {
18.         return bookDao.findTotal();
19.     }
20.
21.     /**
22.      * 查询一页的数据
23.      *
24.      * @param pageIndex 当前页号
25.      * @param pageSize 页面大小
26.      * @param sort 排序的列
27.      * @param order 排序的方式 desc/asc
28.      * @return
29.      */
30.     public List<Book> findPageBooks(int pageIndex, int pageSize, String sort,
31.             String order) {
32.         int begin = (pageIndex - 1)*pageSize;
33.         return bookDao.findPageBooks(begin, pageSize, sort, order);
34.     }
35.
36.     public int addBook(Book book) {
37.         return bookDao.addBook(book);
38.     }
39.
40.     public int deleteBook(int id) {
41.         return bookDao.deleteBook(id);
42.     }
43.
44.     public int updateBook(Book book) {
45.         return bookDao.updateBook(book);
46.     }
47.
48. }
```

BookAction 控制器

Java代码

```java
package cn.action;

import java.util.HashMap;
import java.util.Map;

import cn.biz.BookBiz;
import cn.biz.impl.BookBizImpl;
import cn.entity.Book;

import com.opensymphony.xwork2.ActionSupport;

/**
 * Book 控制器
 * */
public class BookAction extends ActionSupport {

    private BookBiz bookBiz = new BookBizImpl();//业务类
    private Book book;//一本书
    private int page;//当前第几页
    private Map<String, Object> data = new HashMap<String, Object>();//封装数据
    private int rows;//页面大小
    private String order;//排序方向 desc 和 asc
    private String sort;//排序属性名，如：price

    public void setOrder(String order) {
        this.order = order;
    }

    public void setSort(String sort) {
        this.sort = sort;
    }

    public void setRows(int rows) {
        this.rows = rows;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public int getPage() {
        return page;
    }

    public Map<String, Object> getData() {
        return data;
    }

    private boolean operateSuccess;

    public boolean isOperateSuccess() {
        return operateSuccess;
    }

    public void setOperateSuccess(boolean operateSuccess) {
        this.operateSuccess = operateSuccess;
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    public void setBookBiz(BookBiz bookBiz) {
        this.bookBiz = bookBiz;
    }
    /**
     * 查询某一页的书籍
     *
```

```
     * @return
     */
    public String list(){
        data.clear();
        if(sort == null){
            sort = "title";
        }
        if(order == null){
            order = "asc";
        }
        data.put("rows", bookBiz.findPageBooks(page, rows, sort, order));
        data.put("total", bookBiz.findTotal());
        return SUCCESS;
    }

    /**
     * 添加书籍
     *
     * @return
     */
    public String addBook(){
        operateSuccess = (bookBiz.addBook(book)>0);
        return SUCCESS;
    }

    /**
     * 更新书籍
     *
     * @return
     */
    public String updateBook(){
        operateSuccess = (bookBiz.updateBook(book)>0);
        return SUCCESS;
    }

    /**
     * 删除书籍
     *
     * @return
     */
    public String deleteBook(){
        operateSuccess = (bookBiz.deleteBook(book.getId())>0);
        return SUCCESS;
    }
}
```

hibernate.cfg.xml 配置

Java代码

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>
<session-factory>
    <!-- 数据库URL -->
    <property name="connection.url">
        jdbc:oracle:thin:@localhost:1521:oracle11
    </property>
    <!-- 数据库用户 -->
    <property name="connection.username">A_hr</property>
    <!-- 数据库用户密码 -->
    <property name="connection.password">123456</property>
    <!-- 数据库 JDBC 驱动 -->
    <property name="connection.driver_class">
        oracle.jdbc.driver.OracleDriver
    </property>
    <!-- 是否将运行期生成的 SQL 输出到日志以供调试  -->
    <property name="show_sql">true</property>
    <!-- 每个数据库都有其对应的 Dialect 以匹配其平台特征 -->
    <property name="dialect">
        org.hibernate.dialect.OracleDialect
```

```
    </property>
    <property name="hbm2ddl.auto">update</property>
    <mapping class="cn.entity.Book" />
</session-factory>
</hibernate-configuration>
```

struts.xml 配置

Java代码

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
"http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
    <constant name="struts.devMode" value="true"></constant>
    <package name="default" namespace="/" extends="json-default">
        <!-- 显示所有的书籍 -->
        <action name="list" class="cn.action.BookAction" method="list">
            <result type="json">
                <!-- 指定的属性做为根元素输出 -->
                <param name="root">data</param>
            </result>
        </action>
        <!-- 添加书籍 -->
        <action name="addBook" class="cn.action.BookAction" method="addBook">
            <result type="json">
                <param name="root">operateSuccess</param>
            </result>
        </action>
        <!-- 删除书籍 -->
        <action name="deleteBook" class="cn.action.BookAction" method="deleteBook">
            <result type="json">
                <param name="root">operateSuccess</param>
            </result>
        </action>
        <!-- 更新书籍 -->
        <action name="updateBook" class="cn.action.BookAction" method="updateBook">
            <result type="json">
                <param name="root">operateSuccess</param>
            </result>
        </action>
    </package>
</struts>
```

index.jsp 页面

Java代码

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://" + request.getServerName() + ":"
            + request.getServerPort() + path + "/";
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.ddd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <base href="<%=basePath%>" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <!-- easyui配置 -->
        <link rel="stylesheet" href="script/themes/default/easyui.css" type="text/css" />
        <link rel="stylesheet" href="script/themes/icon.css" type="text/css" />
        <script type="text/javascript" src="script/jquery.min.js"></script>
        <script type="text/javascript" src="script/jquery.easyui.min.js"></script>
        <script type="text/javascript" src="script/easyui-lang-zh_CN.js"></script>
        <!-- book的Css和JS -->
        <link rel="stylesheet" href="style/book.css" type="text/css" />
        <script type="text/javascript" src="script/book.js"></script>
        <title>书籍操作</title>
    </head>

    <body>
```

```
        <div id="main">
            <table id="bookbody">
            </table>
        </div>
        <!-- 编辑数据的div，默认看不到 -->
        <div id="divEdit">
                <div id="tabEdit">
                    <form id="frmEdit" method="post">
                    <input type="hidden" id="id" name="book.id" />
                    <dl>
                        <dd>
                            ISBN:
                        </dd>
                        <dd>
                            <input type="text" size="15" id="isbn" name="book.isbn" />
                        </dd>
                    </dl>
                    <dl>
                        <dd>
                            书名:
                        </dd>
                        <dd>
                            <input type="text" size="40" id="title" name="book.title" />
                        </dd>
                    </dl>
                    <dl>
                        <dd>
                            价格￥:
                        </dd>
                        <dd>
                            <input type="text" size="10" id="price" name="book.price" />
                        </dd>
                    </dl>
                    <dl>
                        <dd>
                            出版日期:
                        </dd>
                        <dd>
                            <input type="text" style="width: 150px" id="pubdate" name="book.pubdate" />
                        </dd>
                    </dl>
                    <dl>
                        <dd>
                            简介:
                        </dd>
                        <dd>
                            <textarea cols="45" rows="3" id="intro" name="book.intro"></textarea>
                        </dd>
                    </dl>
                    </form>
                </div>
        </div>
    </body>
</html>
```

book.css 样式

Java代码

```
/* CSS Document */

#divEdit {
    display: none;
}

* {
    font: 12px Arial;
}

div#main {
    margin: 0px auto;
    width: 600px;
}

```

```
#tabEdit input[type="text"],#tabEdit textarea {
    border: solid #66ccff 1px;
}

#tabEdit dl {
    padding-right: 35px;
}
```

book.js 脚本

Java代码

```
// jQuery 起点
$(function(){
    //加载所有的书籍
    listBook();
    //日期加上日期控件
    $("#pubdate").datebox({
        required:true
    });
    //给文本框加上验证器
    $("#isbn").validatebox({
        required:true
    });
    //书名的验证
    $("#title").validatebox({
        required:true,
        missingMessage:'书名不能为空'
    });
    //价格用货币验证框
    $("#price").numberbox({
        required:true,
        min:5.5,
        max:9999,
        precision:2,
        missingMessage:'请输入价格'
    });
    //简介加验证
    $("#intro").validatebox({
        required:true
    });
});
//显示所有的书籍列表
function listBook(){
    $("#bookbody").datagrid({
        whidth:600,
        height:"auto",
        iconsCls:'icon-help',//表格左上角的图标样式
        title:'显示所有书籍',//表格标题
        url:'list.action',//访问服务器的地址，要求返回 JSON 对象
        rownumbers:true,//在最前面显示行号
        fitColumns:true,//自动适应列宽
        pagination:true,//在底部显示分页工具栏
        striped:true,//隔行变色
        singleSelect:true,//每次只选中一行
        loadMsg:'加载书籍列表，请稍候...',
        pageSize:5,//指定每页的大小，服务器要加上 page 属性和 total 属性
        remoteSort:true,//从服务器端排序，默认 true
        pageList:[3,5,10],//可以设置每页记录数的列表，服务器要加上 rows 属性
        idField:'id',//主键属性

        toolbar:[{//工具栏
            text:'添加',
            iconCls:'icon-add',//图标
            handler:function(){//处理函数
                addBook();
            }
        },{
            text:'删除',
            iconCls:'icon-cancel',//图标
            handler:function(){//处理函数
                deleteBook();
            }
```

```
        },{
            text:'编辑',
            iconCls:'icon-edit',//图标
            handler:function(){//处理函数
                editBook();
            }
        }],

        columns:[[{//注意 2 个中括号
            field:'isbn',//实体类属性
            title:'ISBN',//显示的列标题
            width:70
        },{
            field:'title',
            title:'书籍名称',
            //可以排序，但服务器也完成相应的代码，要加入 sort 和 order 属性
            sortable:true
        },{
            field:'price',
            title:'价格',
            align:'right',
            width:60,
            sortable:true,
            formatter:function(value){
                return "$"+value;
            },
            styler:function(value,row,index){//自定义单元格样式
                if(value<80){
                    return 'color:red;';
                }
            }
        },{
            field:'pubdate',
            title:'出版日期',
            sortable:true,
            formatter:function(value){
                return value.substring(0,10);
            }
        }]]
    });
};
//显示添加/编辑窗口
function showEditForm(){
    $("#tabEdit").dialog({
        modal:true,//模式窗口
        title:'书籍操作',
        iconCls:'icon-save',
        buttons:[{
            text:'确认',
            handler:function(){
                //进行表单字段验证，当全部字段都有效时返回 true 和 validatebox 一起使用
                if($("#tabEdit").form('validate')){
                    //提交到服务器并写入数据库
                    dealSave();
                    //关闭窗口
                    colseForm();
                }else{
                    $.messager.alert('验证','书籍信息有误或不完整','error');
                }
            }
        },{
            text:'取消',
            handler:function(){
                colseForm();
            }
        }]
    });
}

//关闭窗口
function colseForm(){
    $("#frmEdit").form('clear');
    $("#tabEdit").dialog('close');
}
```

```
//添加的函数
function addBook(){
    //清空原有的数据
    $("#frmEdit").form('clear');
    //显示添加对话
    showEditForm();
}

//删除书籍
function deleteBook(){
    var book = $("#bookbody").datagrid('getSelected');//得到选中的一行数据
    //如果没有选中记录
    if(book == null){
        $.messager.alert('删除','请先选中要删除的书籍','info');
        return;
    }
    $.messager.confirm('确认','真的要删除选中的记录吗',function(r){
        if(r){
            var url='deleteBook.action?book.id='+book.id;
            //试一下 get 方法（地址，回调函数）
            $.get(url,function(operateSuccess){
                if(operateSuccess){
                    $.messager.alert('删除','选中的书籍成功删除','info');
                    //清除选中
                    $("#bookbody").datagrid('unselectAll');
                    //重新加载
                    $("#bookbody").datagrid('reload');
                }else{
                    $.messager.alert('删除','删除失败','warning');
                }
            });
        }
    });
}

//在增加和更新时点确定按钮的处理函数
function dealSave(){
    //表单数据序列化一个字符串用 & 拼接
    var params = $("#frmEdit").serialize();
    //得到 ID 值，为空串表示添加
    if($("#id").val() == ""){
        $.post("addBook.action",params,function(operateSuccess){
            if(operateSuccess){
                $("#bookbody").datagrid("reload");//重新加载
                $.messager.alert('添加','添加成功','info');
            }else{
                $.messager.alert('添加','添加出现问题！','info');
            }
        });
    }else{
        //表示更新
        $.post("updateBook.action",params,function(operateSuccess){
            if(operateSuccess){
                $("#bookbody").datagrid("reload");//重新加载
                $.messager.alert('更新','更新成功','info');
            }else{
                $.messager.alert('更新','更新出现问题！','info');
            }
        });
    }
}

//编辑按钮的操作
function editBook(){
    var book = $("#bookbody").datagrid('getSelected');//得到选中的一行数据
    //如果没有选中记录
    if(book == null){
        $.messager.alert('书籍','请先选中要编辑的书籍','info');
        return;
    }
    $("#frmEdit").form('clear');
    //填充数据
    $("#id").val(book.id);
    $("#isbn").val(book.isbn);
    $("#title").val(book.title);
```

```
212.     //赋值方法不同
213.     $("#price").numberbox("setValue",book.price);
214.     //给默认方法不同
215.     $("#pubdate").datebox("setValue",book.pubdate.substring(0,10));
216.     //给默认值
217.     $("#intro").val(book.intro);
218.     //显示编辑页面
219.     showEditForm();
220. }
```

效果图：