

综合案例struts2+hibernate+datagrid

Book 实体类及注解

Java代码

```
1. package cn.entity;
2.
3. import java.util.Date;
4.
5. import javax.persistence.Column;
6. import javax.persistence.Entity;
7. import javax.persistence.GeneratedValue;
8. import javax.persistence.GenerationType;
9. import javax.persistence.Id;
10. import javax.persistence.SequenceGenerator;
11. import javax.persistence.Table;
12. /**
13.  * BOOK 实体类
14.  */
15. @Entity
16. @Table(name="BOOK")
17. public class Book {
18.     @Id
19.     @GeneratedValue(generator="BOOK_SEQ",strategy=GenerationType.SEQUENCE)
20.     @SequenceGenerator(name="BOOK_SEQ",sequenceName="BOOK_SEQ",allocationSize=1,initialValue=1)
21.     private int id;//编号
22.     @Column
23.     private String isbn;//书号
24.     @Column
25.     private String title;//标题
26.     @Column
27.     private double price;//价格
28.     @Column
29.     private Date pubdate;//出版日期
30.     @Column
31.     private String intro;//简介
32.     public Book() {
33.     }
34.     public Book(String isbn, String title, double price, Date pubdate,
35.         String intro) {
36.         this.isbn = isbn;
37.         this.title = title;
38.         this.price = price;
39.         this.pubdate = pubdate;
40.         this.intro = intro;
41.     }
42.
43.     public Book(int id, String isbn, String title, double price, Date pubdate,
44.         String intro) {
45.         this.id = id;
46.         this.isbn = isbn;
47.         this.title = title;
48.         this.price = price;
49.         this.pubdate = pubdate;
50.         this.intro = intro;
51.     }
52.     public int getId() {
53.         return id;
54.     }
55.     public void setId(int id) {
56.         this.id = id;
57.     }
58.     public String getIsbn() {
59.         return isbn;
60.     }
61.     public void setIsbn(String isbn) {
62.         this.isbn = isbn;
63.     }
64.     public String getTitle() {
65.         return title;
66.     }
67.     public void setTitle(String title) {
```

```

68.         this.title = title;
69.     }
70.     public double getPrice() {
71.         return price;
72.     }
73.     public void setPrice(double price) {
74.         this.price = price;
75.     }
76.     public Date getPubdate() {
77.         return pubdate;
78.     }
79.     public void setPubdate(Date pubdate) {
80.         this.pubdate = pubdate;
81.     }
82.     public String getIntro() {
83.         return intro;
84.     }
85.     public void setIntro(String intro) {
86.         this.intro = intro;
87.     }
88. }

```

BookDao 数据访问层

Java代码

```

1. package cn.dao;
2.
3. import java.util.List;
4.
5. import cn.entity.Book;
6. /**
7.  * Book 数据访问层接口
8.  * */
9. public interface BookDao {
10.     /**
11.      * 查询共有多少记录
12.      *
13.      * @return
14.      */
15.     public long findTotal();
16.
17.     /**
18.      * 查询一页的数据
19.      *
20.      * @param begin 从哪条开始 0
21.      * @param end 得到多少条
22.      * @param sort 排序的列
23.      * @param order 排序的方式 desc/asc
24.      * @return
25.      */
26.     public List<Book> findPageBooks(final int begin, final int end, final String sort, final String order);
27.
28.     /**
29.      * 增加一条数据
30.      *
31.      * @param book 传来的参数不包括 id
32.      * @return
33.      */
34.     public int addBook(Book book);
35.
36.     /**
37.      * 删除一条数据
38.      *
39.      * @param id 根据 id 作删除
40.      * @return
41.      */
42.     public int deleteBook(int id);
43.
44.     /**
45.      * 修改一条数据
46.      *
47.      * @param book 传来的参数包括 id
48.      * @return

```

```

49.     */
50.     public int updateBook(Book book);
51. }

```

Java代码

```

1. package cn.dao.impl;
2.
3. import java.util.List;
4.
5. import org.hibernate.Criteria;
6. import org.hibernate.HibernateException;
7. import org.hibernate.Session;
8. import org.hibernate.Transaction;
9. import org.hibernate.criterion.Order;
10.
11. import cn.dao.BookDao;
12. import cn.entity.Book;
13. import cn.util.HibernateSessionFactory;
14.
15. /**
16.  * Book 数据访问层实现
17.  */
18. public class BookDaoImpl implements BookDao {
19.
20.     /**
21.      * 查询共有多少记录
22.      *
23.      * @return
24.      */
25.     public long findTotal(){
26.         Session session=HibernateSessionFactory.getSession();
27.         Long count = (Long)session.createQuery("select count(*) from Book").uniqueResult();
28.         return count;
29.     }
30.
31.     /**
32.      * 查询一页的数据
33.      *
34.      * @param begin 从哪条开始 0
35.      * @param end 得到多少条
36.      * @param sort 排序的列
37.      * @param order 排序的方式 desc/asc
38.      * @return
39.      */
40.     public List<Book> findPageBooks(final int begin, final int end, final String sort, final String order){
41.         Session session=HibernateSessionFactory.getSession();
42.         Criteria criteria = session.createCriteria(Book.class);
43.         if("desc".equals(order)){
44.             criteria.addOrder(Order.desc(sort));
45.         }else{
46.             criteria.addOrder(Order.asc(sort));
47.         }
48.         criteria.setFirstResult(begin).setMaxResults(end);
49.         List<Book> books = criteria.list();
50.         HibernateSessionFactory.closeSession();
51.         return books;
52.     }
53.
54.     /**
55.      * 增加一条数据
56.      *
57.      * @param book 传来的参数不包括 id
58.      * @return
59.      */
60.     public int addBook(Book book){
61.         Session session=HibernateSessionFactory.getSession();
62.         Transaction tx=null;
63.         int id = 0;
64.         try {
65.             tx=session.beginTransaction();
66.             id = (Integer)session.save(book);
67.             tx.commit();
68.         } catch (HibernateException e) {

```

```

69.         e.printStackTrace();
70.         tx.rollback();
71.     }finally{
72.         HibernateSessionFactory.closeSession();
73.     }
74.     return id;
75. }
76.
77. /**
78.  * 删除一条数据
79.  *
80.  * @param id 根据 id 作删除
81.  * @return
82.  */
83. public int deleteBook(int id){
84.     Session session=HibernateSessionFactory.getSession();
85.     Transaction tx = null;
86.     int num = 0;
87.     try {
88.         tx=session.beginTransaction();
89.         Book book = new Book();
90.         book.setId(id);
91.         session.delete(book);
92.         tx.commit();
93.         num = 1;
94.     } catch (HibernateException e) {
95.         e.printStackTrace();
96.         tx.rollback();
97.     }finally{
98.         HibernateSessionFactory.closeSession();
99.     }
100.    return num;
101. }
102.
103. /**
104.  * 修改一条数据
105.  *
106.  * @param book 传来的参数包括 id
107.  * @return
108.  */
109. public int updateBook(Book book){
110.     Session session=HibernateSessionFactory.getSession();
111.     Transaction tx = null;
112.     int id = 0;
113.     try {
114.         tx=session.beginTransaction();
115.         session.update(book);
116.         tx.commit();
117.         id = 1;
118.     } catch (HibernateException e) {
119.         e.printStackTrace();
120.         tx.rollback();
121.     }finally{
122.         HibernateSessionFactory.closeSession();
123.     }
124.     return id;
125. }
126.
127. }

```

BookBiz 业务逻辑层

Java代码

```

1. package cn.biz;
2.
3. import java.util.List;
4.
5. import cn.entity.Book;
6.
7. /**
8.  * Book 业务逻辑层接口
9.  */
10. 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,
String order) {
31.         int begin = (pageIndex - 1)*pageSize;
32.         return bookDao.findPageBooks(begin, pageSize, sort, order);
33.     }
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

```
1. package cn.action;
2.
3. import java.util.HashMap;
4. import java.util.Map;
5.
6. import cn.biz.BookBiz;
7. import cn.biz.impl.BookBizImpl;
8. import cn.entity.Book;
9.
10. import com.opensymphony.xwork2.ActionSupport;
11.
12. /**
13.  * Book 控制器
14.  */
15. public class BookAction extends ActionSupport {
16.
17.     private BookBiz bookBiz = new BookBizImpl(); //业务类
18.     private Book book; //一本书
19.     private int page; //当前第几页
20.     private Map<String, Object> data = new HashMap<String, Object>(); //封装数据
21.     private int rows; //页面大小
22.     private String order; //排序方向 desc 和 asc
23.     private String sort; //排序属性名, 如: price
24.
25.     public void setOrder(String order) {
26.         this.order = order;
27.     }
28.
29.     public void setSort(String sort) {
30.         this.sort = sort;
31.     }
32.
33.     public void setRows(int rows) {
34.         this.rows = rows;
35.     }
36.
37.     public void setPage(int page) {
38.         this.page = page;
39.     }
40.
41.     public int getPage() {
42.         return page;
43.     }
44.
45.     public Map<String, Object> getData() {
46.         return data;
47.     }
48.
49.     private boolean operateSuccess;
50.
51.     public boolean isOperateSuccess() {
52.         return operateSuccess;
53.     }
54.
55.     public void setOperateSuccess(boolean operateSuccess) {
56.         this.operateSuccess = operateSuccess;
57.     }
58.
59.     public Book getBook() {
60.         return book;
61.     }
62.
63.     public void setBook(Book book) {
64.         this.book = book;
65.     }
66.
67.     public void setBookBiz(BookBiz bookBiz) {
68.         this.bookBiz = bookBiz;
69.     }
70.     /**
71.      * 查询某一页的书籍
72.      */
```

```

73.     * @return
74.     */
75.     public String list(){
76.         data.clear();
77.         if(sort == null){
78.             sort = "title";
79.         }
80.         if(order == null){
81.             order = "asc";
82.         }
83.         data.put("rows", bookBiz.findPageBooks(page, rows, sort, order));
84.         data.put("total", bookBiz.findTotal());
85.         return SUCCESS;
86.     }
87.
88.     /**
89.     * 添加书籍
90.     *
91.     * @return
92.     */
93.     public String addBook(){
94.         operateSuccess = (bookBiz.addBook(book)>0);
95.         return SUCCESS;
96.     }
97.
98.     /**
99.     * 更新书籍
100.    *
101.    * @return
102.    */
103.    public String updateBook(){
104.        operateSuccess = (bookBiz.updateBook(book)>0);
105.        return SUCCESS;
106.    }
107.
108.    /**
109.    * 删除书籍
110.    *
111.    * @return
112.    */
113.    public String deleteBook(){
114.        operateSuccess = (bookBiz.deleteBook(book.getId())>0);
115.        return SUCCESS;
116.    }
117. }

```

hibernate.cfg.xml 配置

Java代码

```

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

```

```

25.     </property>
26.     <property name="hbm2ddl.auto">update</property>
27.     <mapping class="cn.entity.Book" />
28. </session-factory>
29. </hibernate-configuration>

```

struts.xml 配置

Java代码

```

1. <?xml version="1.0" encoding="UTF-8" ?>
2. <!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
   "http://struts.apache.org/dtds/struts-2.1.dtd">
3. <struts>
4.     <constant name="struts.devMode" value="true"></constant>
5.     <package name="default" namespace="/" extends="json-default">
6.         <!-- 显示所有的书籍 -->
7.         <action name="list" class="cn.action.BookAction" method="list">
8.             <result type="json">
9.                 <!-- 指定的属性做为根元素输出 -->
10.                <param name="root">data</param>
11.            </result>
12.        </action>
13.        <!-- 添加书籍 -->
14.        <action name="addBook" class="cn.action.BookAction" method="addBook">
15.            <result type="json">
16.                <param name="root">operateSuccess</param>
17.            </result>
18.        </action>
19.        <!-- 删除书籍 -->
20.        <action name="deleteBook" class="cn.action.BookAction" method="deleteBook">
21.            <result type="json">
22.                <param name="root">operateSuccess</param>
23.            </result>
24.        </action>
25.        <!-- 更新书籍 -->
26.        <action name="updateBook" class="cn.action.BookAction" method="updateBook">
27.            <result type="json">
28.                <param name="root">operateSuccess</param>
29.            </result>
30.        </action>
31.    </package>
32. </struts>

```

index.jsp 页面

Java代码

```

1. <%@ page language="java" pageEncoding="UTF-8"%>
2. <%
3.     String path = request.getContextPath();
4.     String basePath = request.getScheme() + "://" + request.getServerName() + ":"
5.         + request.getServerPort() + path + "/";
6. %>
7. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
8. <html xmlns="http://www.w3.org/1999/xhtml">
9.     <head>
10.        <base href="<%=basePath%>" />
11.        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12.        <!-- easyui配置 -->
13.        <link rel="stylesheet" href="script/themes/default/easyui.css" type="text/css" />
14.        <link rel="stylesheet" href="script/themes/icon.css" type="text/css" />
15.        <script type="text/javascript" src="script/jquery.min.js"></script>
16.        <script type="text/javascript" src="script/jquery.easyui.min.js"></script>
17.        <script type="text/javascript" src="script/easyui-lang-zh_CN.js"></script>
18.        <!-- book的css和js -->
19.        <link rel="stylesheet" href="style/book.css" type="text/css" />
20.        <script type="text/javascript" src="script/book.js"></script>
21.        <title>书籍操作</title>
22.    </head>
23.
24.    <body>

```



```

25.     <div id="main">
26.         <table id="bookbody">
27.             </table>
28.     </div>
29.     <!-- 编辑数据的div，默认看不到 -->
30.     <div id="divEdit">
31.         <div id="tabEdit">
32.             <form id="frmEdit" method="post">
33.                 <input type="hidden" id="id" name="book.id" />
34.                 <dl>
35.                     <dd>
36.                         ISBN:
37.                     </dd>
38.                     <dd>
39.                         <input type="text" size="15" id="isbn" name="book.isbn" />
40.                     </dd>
41.                 </dl>
42.                 <dl>
43.                     <dd>
44.                         书名:
45.                     </dd>
46.                     <dd>
47.                         <input type="text" size="40" id="title" name="book.title" />
48.                     </dd>
49.                 </dl>
50.                 <dl>
51.                     <dd>
52.                         价格¥:
53.                     </dd>
54.                     <dd>
55.                         <input type="text" size="10" id="price" name="book.price" />
56.                     </dd>
57.                 </dl>
58.                 <dl>
59.                     <dd>
60.                         出版日期:
61.                     </dd>
62.                     <dd>
63.                         <input type="text" style="width: 150px" id="pubdate" name="book.pubdate" />
64.                     </dd>
65.                 </dl>
66.                 <dl>
67.                     <dd>
68.                         简介:
69.                     </dd>
70.                     <dd>
71.                         <textarea cols="45" rows="3" id="intro" name="book.intro"></textarea>
72.                     </dd>
73.                 </dl>
74.             </form>
75.         </div>
76.     </div>
77. </body>
78. </html>

```

book.css 样式

Java代码

```

1.  /* CSS Document */
2.
3.  #divEdit {
4.      display: none;
5.  }
6.
7.  * {
8.      font: 12px Arial;
9.  }
10.
11. div#main {
12.     margin: 0px auto;
13.     width: 600px;
14. }
15.

```

```

16.
17. #tabEdit input[type="text"],#tabEdit textarea {
18.     border: solid #66ccff 1px;
19. }
20.
21. #tabEdit dl {
22.     padding-right: 35px;
23. }

```

book.js 脚本

Java代码

```

1. // jQuery 起点
2. $(function(){
3.     //加载所有的书籍
4.     listBook();
5.     //日期加上日期控件
6.     $("#pubdate").datebox({
7.         required:true
8.     });
9.     //给文本框加上验证器
10.    $("#isbn").validatebox({
11.        required:true
12.    });
13.    //书名的验证
14.    $("#title").validatebox({
15.        required:true,
16.        missingMessage: '书名不能为空'
17.    });
18.    //价格用货币验证框
19.    $("#price").numberbox({
20.        required:true,
21.        min:5.5,
22.        max:9999,
23.        precision:2,
24.        missingMessage: '请输入价格'
25.    });
26.    //简介加验证
27.    $("#intro").validatebox({
28.        required:true
29.    });
30. });
31. //显示所有的书籍列表
32. function listBook(){
33.     $("#bookbody").datagrid({
34.         whidth:600,
35.         height:"auto",
36.         iconsCls:'icon-help',//表格左上角的图标样式
37.         title:'显示所有书籍',//表格标题
38.         url:'list.action',//访问服务器的地址, 要求返回 JSON 对象
39.         rownumbers:true,//在最前面显示行号
40.         fitColumns:true,//自动适应列宽
41.         pagination:true,//在底部显示分页工具栏
42.         striped:true,//隔行变色
43.         singleSelect:true,//每次只选中一行
44.         loadMsg:'加载书籍列表, 请稍候...',
45.         pageSize:5,//指定每页的大小, 服务器要加上 page 属性和 total 属性
46.         remoteSort:true,//从服务器端排序, 默认 true
47.         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
48.         idField:'id',//主键属性
49.
50.         toolbar:[{//工具栏
51.             text:'添加',
52.             iconCls:'icon-add',//图标
53.             handler:function(){//处理函数
54.                 addBook();
55.             }
56.         },{
57.             text:'删除',
58.             iconCls:'icon-cancel',//图标
59.             handler:function(){//处理函数
60.                 deleteBook();
61.             }

```

```

62.     },{
63.         text: '编辑',
64.         iconCls: 'icon-edit', // 图标
65.         handler: function() { // 处理函数
66.             editBook();
67.         }
68.     }],
69.
70.     columns: [[ // 注意 2 个中括号
71.         {
72.             field: 'isbn', // 实体类属性
73.             title: 'ISBN', // 显示的列标题
74.             width: 70
75.         }, {
76.             field: 'title',
77.             title: '书籍名称',
78.             // 可以排序, 但服务器也完成相应的代码, 要加入 sort 和 order 属性
79.             sortable: true
80.         }, {
81.             field: 'price',
82.             title: '价格',
83.             align: 'right',
84.             width: 60,
85.             sortable: true,
86.             formatter: function(value) {
87.                 return "$"+value;
88.             },
89.             styler: function(value, row, index) { // 自定义单元格样式
90.                 if (value < 80) {
91.                     return 'color:red;';
92.                 }
93.             }
94.         }, {
95.             field: 'pubdate',
96.             title: '出版日期',
97.             sortable: true,
98.             formatter: function(value) {
99.                 return value.substring(0, 10);
100.            }
101.        }]]
102.    });
103.    // 显示添加/编辑窗口
104.    function showEditForm() {
105.        $("#tabEdit").dialog({
106.            modal: true, // 模式窗口
107.            title: '书籍操作',
108.            iconCls: 'icon-save',
109.            buttons: [{
110.                text: '确认',
111.                handler: function() {
112.                    // 进行表单字段验证, 当全部字段都有效时返回 true 和 validatebox 一起使用
113.                    if ($("#tabEdit").form('validate')) {
114.                        // 提交到服务器并写入数据库
115.                        dealSave();
116.                        // 关闭窗口
117.                        colseForm();
118.                    } else {
119.                        $.messager.alert('验证', '书籍信息有误或不完整', 'error');
120.                    }
121.                }
122.            }, {
123.                text: '取消',
124.                handler: function() {
125.                    colseForm();
126.                }
127.            }
128.        ]});
129.    }
130.
131.    // 关闭窗口
132.    function colseForm() {
133.        $("#frmEdit").form('clear');
134.        $("#tabEdit").dialog('close');
135.    }
136.

```

```

137. //添加的函数
138. function addBook(){
139.     //清空原有的数据
140.     $("#frmEdit").form('clear');
141.     //显示添加对话
142.     showEditForm();
143. }
144.
145. //删除书籍
146. function deleteBook(){
147.     var book = $("#bookbody").datagrid('getSelected');//得到选中的一行数据
148.     //如果没有选中记录
149.     if(book == null){
150.         $.messager.alert('删除','请先选中要删除的书籍','info');
151.         return;
152.     }
153.     $.messager.confirm('确认','真的要删除选中的记录吗',function(r){
154.         if(r){
155.             var url='deleteBook.action?book.id='+book.id;
156.             //试一下 get 方法（地址，回调函数）
157.             $.get(url,function(operateSuccess){
158.                 if(operateSuccess){
159.                     $.messager.alert('删除','选中的书籍成功删除','info');
160.                     //清除选中
161.                     $("#bookbody").datagrid('unselectAll');
162.                     //重新加载
163.                     $("#bookbody").datagrid('reload');
164.                 }else{
165.                     $.messager.alert('删除','删除失败','warning');
166.                 }
167.             });
168.         }
169.     });
170. }
171.
172. //在增加和更新时点确定按钮的处理函数
173. function dealSave(){
174.     //表单数据序列化一个字符串用 & 拼接
175.     var params = $("#frmEdit").serialize();
176.     //得到 ID 值，为空串表示添加
177.     if($("#id").val() == ""){
178.         $.post("addBook.action",params,function(operateSuccess){
179.             if(operateSuccess){
180.                 $("#bookbody").datagrid("reload");//重新加载
181.                 $.messager.alert('添加','添加成功','info');
182.             }else{
183.                 $.messager.alert('添加','添加出现问题!','info');
184.             }
185.         });
186.     }else{
187.         //表示更新
188.         $.post("updateBook.action",params,function(operateSuccess){
189.             if(operateSuccess){
190.                 $("#bookbody").datagrid("reload");//重新加载
191.                 $.messager.alert('更新','更新成功','info');
192.             }else{
193.                 $.messager.alert('更新','更新出现问题!','info');
194.             }
195.         });
196.     }
197. }
198.
199. //编辑按钮的操作
200. function editBook(){
201.     var book = $("#bookbody").datagrid('getSelected');//得到选中的一行数据
202.     //如果没有选中记录
203.     if(book == null){
204.         $.messager.alert('书籍','请先选中要编辑的书籍','info');
205.         return;
206.     }
207.     $("#frmEdit").form('clear');
208.     //填充数据
209.     $("#id").val(book.id);
210.     $("#isbn").val(book.isbn);
211.     $("#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:

