11-书城项目第五阶段-图书模块

讲师：王振国

今日任务

尚硅谷书城项目

第五阶段：

1、MVC 概念

MVC 全称：Model 模型、 View 视图、 Controller 控制器。

MVC 最早出现在 JavaEE 三层中的 Web 层，它可以有效的指导 Web 层的代码如何有效分离，单独工作。

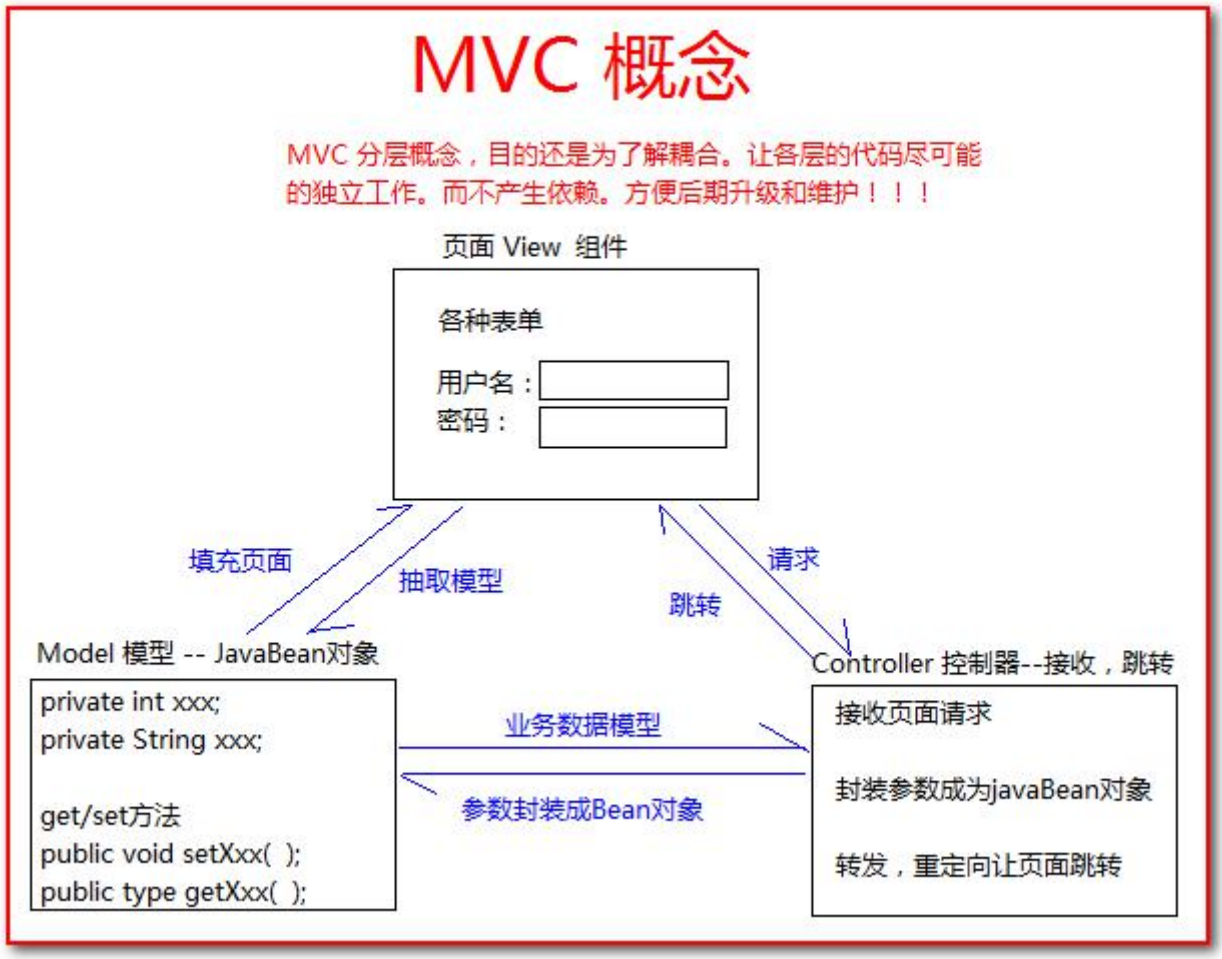
View 视图：只负责数据和界面的显示，不接受任何与显示数据无关的代码，便于程序员和美工的分工合作——JSP/HTML。

Controller 控制器：只负责接收请求，调用业务层的代码处理请求，然后派发页面，是一个“调度者”的角色——Servlet。转到某个页面。或者是重定向到某个页面。

Model 模型：将与业务逻辑相关的数据封装为具体的 JavaBean 类，其中不掺杂任何与数据处理相关的代码——JavaBean/domain/entity/pojo。

MVC 是一种思想

MVC 的理念是将软件代码拆分成为组件，单独开发，组合使用（**目的还是为了降低耦合度**）。



MVC 的作用还是为了降低耦合。让代码合理分层。方便后期升级和维护。

书城第五阶段

1、图书模块

1.1、编写图书模块的数据库表

```
create table t_book(
   `id` int primary key auto_increment,
   `name` varchar(100),
   `price` decimal(11,2),
   `author` varchar(100),
   `sales` int,
   `stock` int,
   `img_path` varchar(200)
);
```

```
## 插入初始化测试数据
insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'java 从入门到放弃' , '国哥' , 80 , 9999 , 9 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '数据结构与算法' , '严敏君' , 78.5 , 6 , 13 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '怎样拐跑别人的媳妇' , '龙伍' , 68, 99999 , 52 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '木虚肉盖饭' , '小胖' , 16, 1000 , 50 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'C++编程思想' , '刚哥' , 45.5 , 14 , 95 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '蛋炒饭' , '周星星' , 9.9, 12 , 53 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '赌神' , '龙伍' , 66.5, 125 , 535 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'Java 编程思想' , '阳哥' , 99.5 , 47 , 36 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'JavaScript 从入门到精通' , '婷姐' , 9.9 , 85 , 95 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'cocos2d-x 游戏编程入门' , '国哥' , 49, 52 , 62 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'C 语言程序设计' , '谭浩强' , 28 , 52 , 74 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'Lua 语言程序设计' , '雷丰阳' , 51.5 , 48 , 82 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '西游记' , '罗贯中' , 12, 19 , 9999 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '水浒传' , '华仔' , 33.05 , 22 , 88 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '操作系统原理' , '刘优' , 133.05 , 122 , 188 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '数据结构 java 版' , '封大神' , 173.15 , 21 , 81 , 'static/img/default.jpg');
```

```
insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'UNIX 高级环境编程' , '乐天' , 99.15 , 210 , 810 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , 'javaScript 高级编程' , '国哥' , 69.15 , 210 , 810 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '大话设计模式' , '国哥' , 89.15 , 20 , 10 , 'static/img/default.jpg');

insert into t_book(`id` , `name` , `author` , `price` , `sales` , `stock` , `img_path`)
values(null , '人月神话' , '刚哥' , 88.15 , 20 , 80 , 'static/img/default.jpg');


## 查看表内容
select id,name,author,price,sales,stock,img_path from t_book;
```

1.2、编写图书模块的 JavaBean

```
public class Book {
    private Integer id;
    private String name;
    private String author;
    private BigDecimal price;
    private Integer sales;
    private Integer stock;
    private String imgPath = "static/img/default.jpg";
```

1.3、编写图书模块的 Dao 和测试 Dao

Dao 接口

```
public interface BookDao {

    public int addBook(Book book);

    public int deleteBookById(Integer id);

    public int updateBook(Book book);

    public Book queryBookById(Integer id);

    public List<Book> queryBooks();
```

```
}
```

BookDaoImpl 实现类：

```
public class BookDaoImpl extends BaseDao implements BookDao {
    @Override
    public int addBook(Book book) {

        String sql = "insert into t_book(`name`,`author`,`price`,`sales`,`stock`,`img_path`)
values(?,?,?,?,?,?)";

        return update(sql,
book.getName(),book.getAuthor(),book.getPrice(),book.getSales(),book.getStock(),book.getImgPath());

    }

    @Override
    public int deleteBookById(Integer id) {
        String sql = "delete from t_book where id = ?";
        return update(sql, id);
    }

    @Override
    public int updateBook(Book book) {
        String sql = "update t_book set `name`=?,`author`=?,`price`=?,`sales`=?,`stock`=?,`img_path`=?
where id = ?";
        return
update(sql,book.getName(),book.getAuthor(),book.getPrice(),book.getSales(),book.getStock(),book.ge
tImgPath(),book.getId());
    }

    @Override
    public Book queryBookById(Integer id) {
        String sql = "select `id` , `name` , `author` , `price` , `sales` , `stock` , `img_path` imgPath
from t_book where id = ?";
        return queryForOne(Book.class, sql,id);
    }

    @Override
    public List<Book> queryBooks() {
        String sql = "select `id` , `name` , `author` , `price` , `sales` , `stock` , `img_path` imgPath
from t_book";
        return queryForList(Book.class, sql);
    }
}
```

BookDao 的测试：

```
public class BookDaoTest {

    private BookDao bookDao = new BookDaoImpl();

    @Test
    public void addBook() {
        bookDao.addBook(new Book(null,"国哥为什么这么帅！", "191125", new
BigDecimal(9999),1100000,0,null
                ));
    }

    @Test
    public void deleteBookById() {
        bookDao.deleteBookById(21);
    }

    @Test
    public void updateBook() {
        bookDao.updateBook(new Book(21,"大家都可以这么帅！", "国哥", new
BigDecimal(9999),1100000,0,null
        ));
    }

    @Test
    public void queryBookById() {
        System.out.println( bookDao.queryBookById(21) );
    }

    @Test
    public void queryBooks() {
        for (Book queryBook : bookDao.queryBooks()) {
            System.out.println(queryBook);
        }
    }
}
```

1.4、编写图书模块的 Service 和测试 Service

BookService 接口

```
public interface BookService {

    public void addBook(Book book);

    public void deleteBookById(Integer id);
```

```
    public void updateBook(Book book);

    public Book queryBookById(Integer id);

    public List<Book> queryBooks();

}
```

BookServiceImpl 实现类：

```
public class BookServiceImpl implements BookService {

    private BookDao bookDao = new BookDaoImpl();

    @Override
    public void addBook(Book book) {
        bookDao.addBook(book);
    }

    @Override
    public void deleteBookById(Integer id) {
        bookDao.deleteBookById(id);
    }

    @Override
    public void updateBook(Book book) {
        bookDao.updateBook(book);
    }

    @Override
    public Book queryBookById(Integer id) {
        return bookDao.queryBookById(id);
    }

    @Override
    public List<Book> queryBooks() {
        return bookDao.queryBooks();
    }
}
```

BookService 的测试：

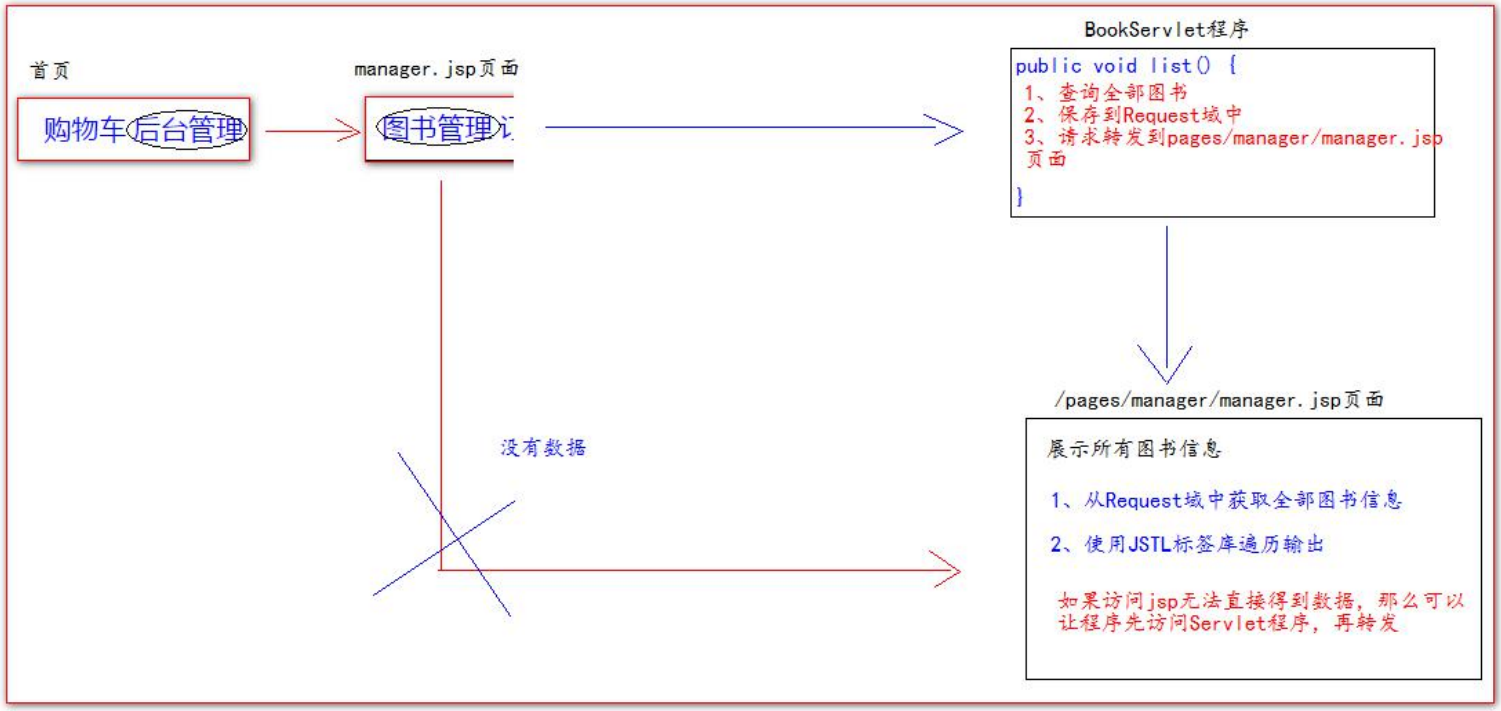
```
public class BookServiceTest {

    private BookService bookService = new BookServiceImpl();

    @Test
    public void addBook() {
        bookService.addBook(new Book(null,"国哥在手，天下我有！", "1125", new BigDecimal(1000000),
100000000, 0, null));
```

```
    }

    @Test
    public void deleteBookById() {
        bookService.deleteBookById(22);
    }

    @Test
    public void updateBook() {
        bookService.updateBook(new Book(22,"社会我国哥，人狠话不多！", "1125", new BigDecimal(999999),
10, 111110, null));
    }

    @Test
    public void queryBookById() {
        System.out.println(bookService.queryBookById(22));
    }

    @Test
    public void queryBooks() {
        for (Book queryBook : bookService.queryBooks()) {
            System.out.println(queryBook);
        }
    }
}
```

1.5、编写图书模块的 Web 层，和页面联调测试

1.5.1、图书列表功能的实现

1、图解列表功能流程：



2、BookServlet 程序中添加 list 方法

```
protected void list(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    //1 通过 BookService 查询全部图书
    List<Book> books = bookService.queryBooks();
    //2 把全部图书保存到 Request 域中
    req.setAttribute("books", books);
    //3、请求转发到/pages/manager/book_manager.jsp 页面
    req.getRequestDispatcher("/pages/manager/book_manager.jsp").forward(req,resp);
}
```

3、修改【图书管理】请求地址

```
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java"
<div>
    <a href="manager/bookServlet?action=list">图书管理</a>
    <a href="order_manager.jsp">订单管理</a>
    <a href="../../index.jsp">返回商城</a>
</div>
```

4、修改 pages/manager/book_manager.jsp 页面的数据遍历输出

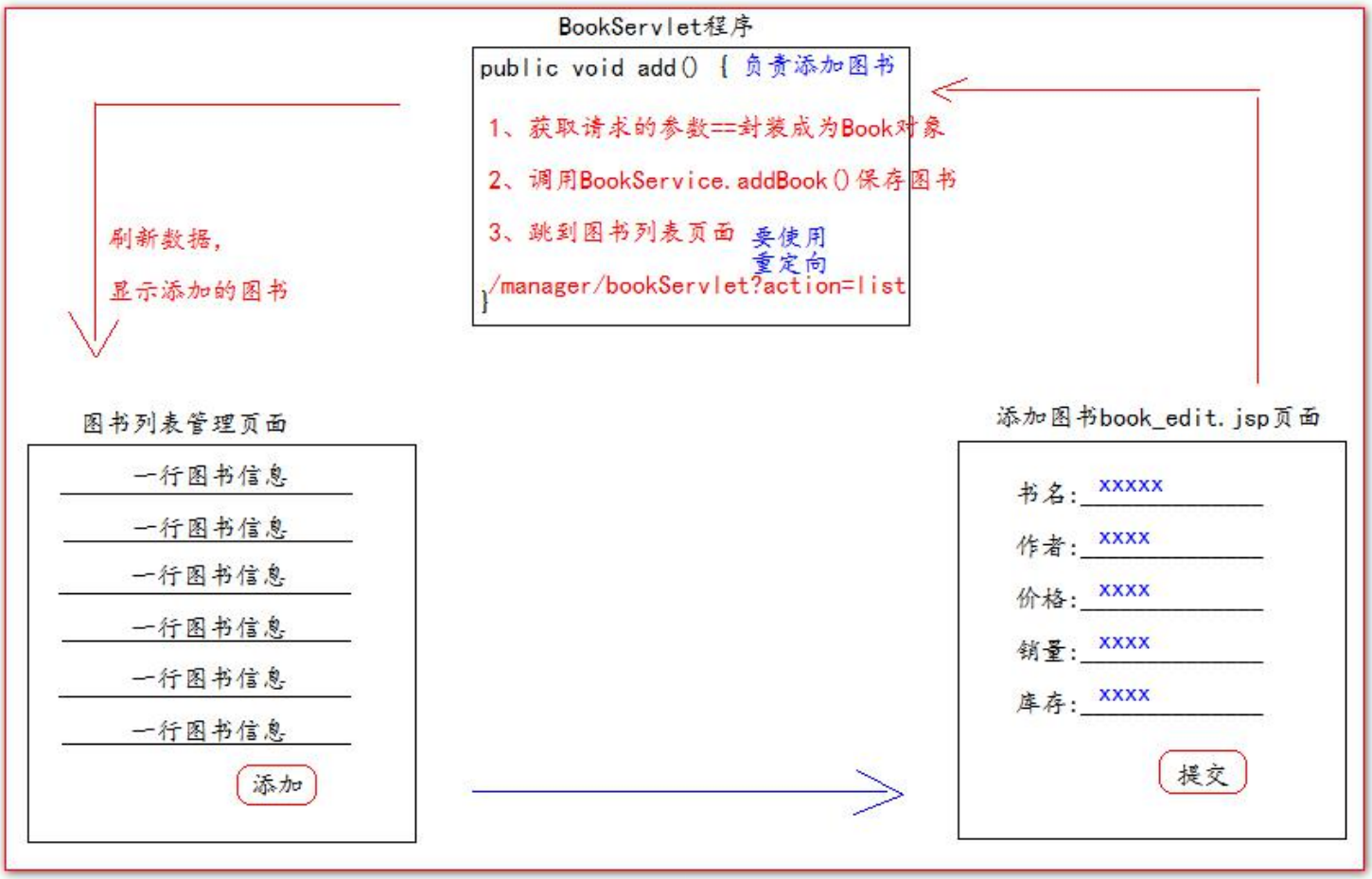
```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>图书管理</title>

   <%-- 静态包含 base 标签、css 样式、jQuery 文件 --%>
   <%@ include file="/pages/common/head.jsp"%>


</head>
<body>

   <div id="header">
         <img class="logo_img" alt="" src="../../static/img/logo.gif" >
         <span class="wel_word">图书管理系统</span>

      <%-- 静态包含 manager 管理模块的菜单  --%>
      <%@include file="/pages/common/manager_menu.jsp"%>


   </div>

   <div id="main">
      <table>
         <tr>
            <td>名称</td>
            <td>价格</td>
            <td>作者</td>
            <td>销量</td>
            <td>库存</td>
            <td colspan="2">操作</td>
         </tr>

         <c:forEach items="${requestScope.books}" var="book">
            <tr>
               <td>${book.name}</td>
               <td>${book.price}</td>
               <td>${book.author}</td>
               <td>${book.sales}</td>
               <td>${book.stock}</td>
               <td><a href="book_edit.jsp">修改</a></td>
               <td><a href="#">删除</a></td>
```

```
            </tr>
        </c:forEach>

        <tr>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td><a href="book_edit.jsp">添加图书</a></td>
        </tr>
    </table>
</div>
<%--静态包含页脚内容--%>
<%@include file="/pages/common/footer.jsp"%>
</body>
</html>
```

1.5.2、前后台的简单介绍

1.5.3、添加图书功能的实现

1.5.3.1、添加图书流程细节：



1.5.3.2、问题说明：表单重复提交：

当用户提交完请求，浏览器会记录下最后一次请求的全部信息。当用户按下功能键 F5，就会发起浏览器记录的最后一次请求。

最后一次请求。

1.5.3.3、BookServlet 程序中添加 add 方法

```
    protected void add(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        1、获取请求的参数==封装成为 Book 对象
        Book book = WebUtils.copyParamToBean(req.getParameterMap(),new Book());
//        2、调用 BookService.addBook()保存图书
        bookService.addBook(book);
//        3、跳到图书列表页面
//                /manager/bookServlet?action=list
//        req.getRequestDispatcher("/manager/bookServlet?action=list").forward(req, resp);
```

```
        resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=list");

    }
```

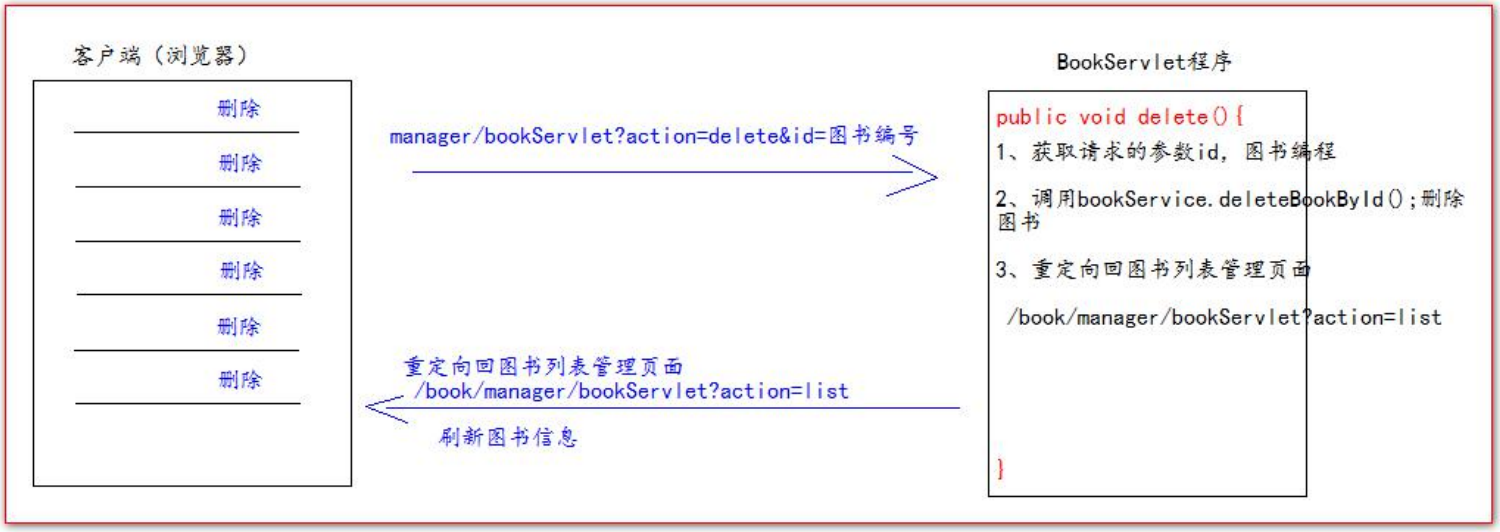
1.5.3.4、修改 book_edit.jsp 页面

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>编辑图书</title>

    <%-- 静态包含 base 标签、css 样式、jQuery 文件 --%>
    <%@ include file="/pages/common/head.jsp"%>


    <style type="text/css">
    h1 {
        text-align: center;
        margin-top: 200px;
    }

    h1 a {
        color:red;
    }

    input {
        text-align: center;
    }
</style>
</head>
<body>
        <div id="header">
            <img class="logo_img" alt="" src="../../static/img/logo.gif" >
            <span class="wel_word">编辑图书</span>

            <%-- 静态包含 manager 管理模块的菜单  --%>
            <%@include file="/pages/common/manager_menu.jsp"%>


        </div>

        <div id="main">
            <form action="manager/bookServlet" method="get">
                <input type="hidden" name="action" value="add" />
```

```
            <table>
                <tr>
                    <td>名称</td>
                    <td>价格</td>
                    <td>作者</td>
                    <td>销量</td>
                    <td>库存</td>
                    <td colspan="2">操作</td>
                </tr>
                <tr>
                    <td><input name="name" type="text" value="时间简史"/></td>
                    <td><input name="price" type="text" value="30.00"/></td>
                    <td><input name="author" type="text" value="霍金"/></td>
                    <td><input name="sales" type="text" value="200"/></td>
                    <td><input name="stock" type="text" value="300"/></td>
                    <td><input type="submit" value="提交"/></td>
                </tr>
            </table>
        </form>

    </div>

    <%--静态包含页脚内容--%>
    <%@include file="/pages/common/footer.jsp"%>

</body>
</html>
```

1.5.4、删除图书功能的实现

1.5.4.1、图解删除流程：



1.5.4.2、BookServlet 程序中的 delete 方法：

```
    protected void delete(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        1、获取请求的参数 id，图书编程
        int id = WebUtils.parseInt(req.getParameter("id"), 0);
//        2、调用 bookService.deleteBookById();删除图书
        bookService.deleteBookById(id);
//        3、重定向回图书列表管理页面
//                /book/manager/bookServlet?action=list
        resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=list");
    }
```

1.5.4.3、给 WebUtils 工具类添加转换 int 类型的工具方法

```
/**
 * 将字符串转换成为 int 类型的数据
 * @param strInt
 * @param defaultValue
 * @return
 */
public static int parseInt(String strInt,int defaultValue) {
    try {
        return Integer.parseInt(strInt);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return defaultValue;
}
```

1.5.4.4、修改删除的连接地址：

```
        <td>${book.stock}</td>
        <td><a href="book_edit.jsp">修改</a></td>
        <td><a class="deleteClass" href="manager/bookServlet?action=delete&id=${book.id}">删除</a></td>
    </tr>
</c:forEach>
```

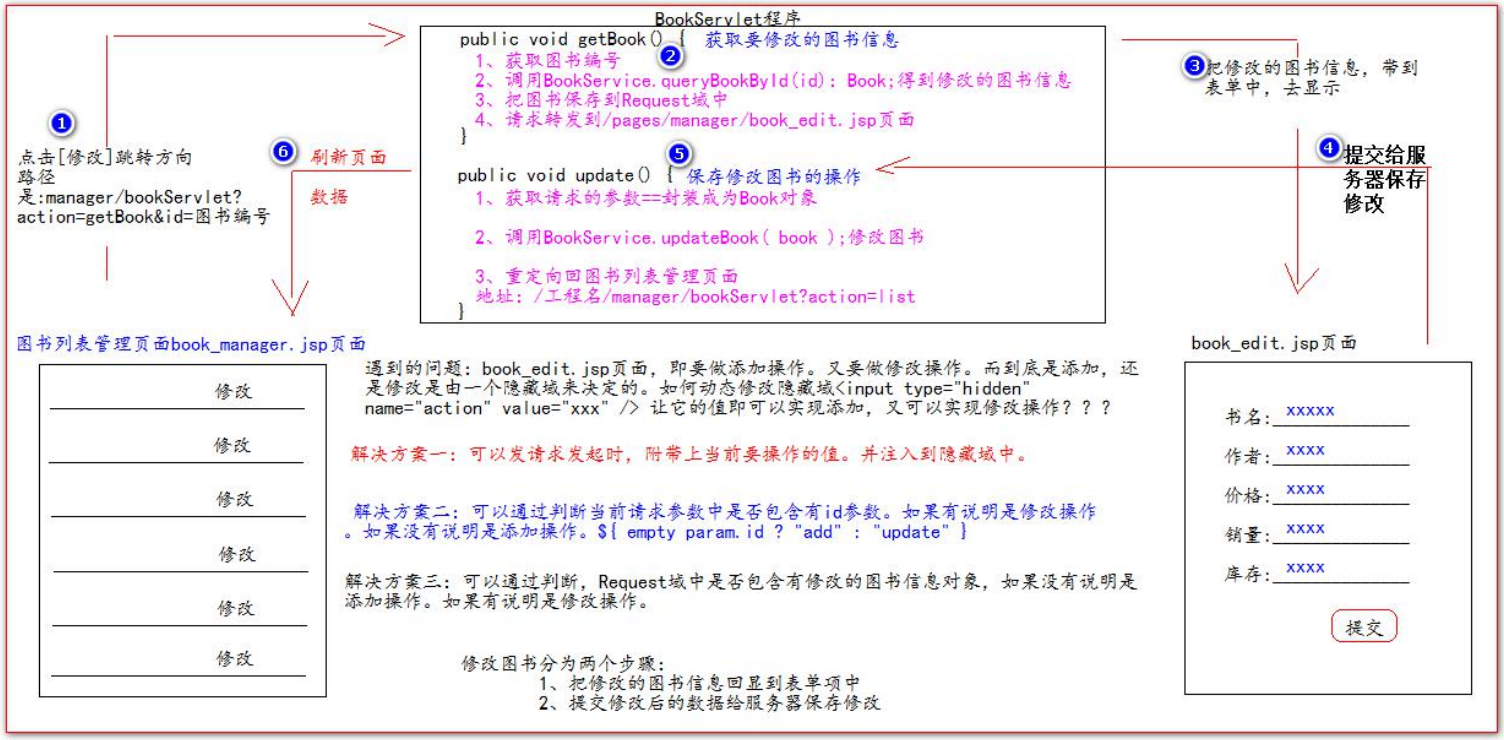
1.5.4.5、给删除添加确认提示操作：

```
<script type="text/javascript">
  $(function () {
```

```
    // 给删除的a标签绑定单击事件，用于删除的确认提示操作
    $("a.deleteClass").click(function () {
        // 在事件的function函数中，有一个this对象。这个this对象，是当前正在响应事件的dom对象。
        /**
         * confirm是确认提示框函数
         * 参数是它的提示内容
         * 它有两个按钮，一个确认，一个是取消。
         * 返回true表示点击了，确认，返回false表示点击取消。
         */
        return confirm("你确定要删除【" + $(this).parent().parent().find("td:first").text() + "】?");
        // return false// 阻止元素的默认行为===不提交请求
    });
  });
</script>
```

1.5.5、修改图书功能的实现

1.5.5.1：图解修改图书细节：



1.5.5.2、更新【修改】的请求地址：

```
        <td>${book.author}</td>
        <td>${book.sales}</td>
        <td>${book.stock}</td>
        <td><a href="manager/bookServlet?action=getBook&id=${book.id}">修改</a></td>
        <td><a class="deleteClass" href="manager/bookServlet?action=delete&id=${book.id}">删除</a></td>
    </tr>
```

1.5.5.3、BookServlet 程序中添加 getBook 方法：

```
protected void getBook(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    //1 获取请求的参数图书编号
    int id = WebUtils.parseInt(req.getParameter("id"), 0);
    //2 调用 bookService.queryBookById 查询图书
    Book book = bookService.queryBookById(id);
    //3 保存到图书到 Request 域中
    req.setAttribute("book", book) ;
    //4 请求转发到。pages/manager/book_edit.jsp 页面
    req.getRequestDispatcher("/pages/manager/book_edit.jsp").forward(req,resp);
}
```

1.5.5.4、在 book_edit.jsp 页面中显示修改的数据

```
<div id="main">
  <form action="manager/bookServlet" method="get">
    <input type="hidden" name="action" value="add" />
    <table>
      <tr>
        <td>名称</td>
        <td>价格</td>
        <td>作者</td>
        <td>销量</td>
        <td>库存</td>
        <td colspan="2">操作</td>
      </tr>
      <tr>
        <td><input name="name" type="text" value="${requestScope.book.name}"/></td>
        <td><input name="price" type="text" value="${requestScope.book.price}"/></td>
        <td><input name="author" type="text" value="${requestScope.book.author}"/></td>
        <td><input name="sales" type="text" value="${requestScope.book.sales}"/></td>
        <td><input name="stock" type="text" value="${requestScope.book.stock}"/></td>
```

```
            <td><input type="submit" value="提交"/></td>
        </tr>
    </table>
</form>
</div>
```

1.5.5.5、在 BookServlet 程序中添加 update 方法：

```
    protected void update(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        1、获取请求的参数==封装成为 Book 对象
        Book book = WebUtils.copyParamToBean(req.getParameterMap(),new Book());
//        2、调用 BookService.updateBook( book );修改图书
        bookService.updateBook(book);
//        3、重定向回图书列表管理页面
//        地址：/工程名/manager/bookServlet?action=list
        resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=list");
    }
```

1.5.5.6、解决 book_edit.jsp 页面，即要实现添加，又要实现修改操作。

```
<div id="main">
    <form action="manager/bookServlet" method="get">
        <input type="hidden" name="action" value="${ empty param.id ? "add" : "update" }" />
        <input type="hidden" name="id" value="${ requestScope.book.id }" />
        <table>
            <tr>
```