# 第一章 SpringMVC 框架（一）

## 课程目标

1、 SpringMVC 概述

2、 SpringMVC 入门

3、 Spring 框架架构、运行原理分析

4、 SSM 整合-实战案例

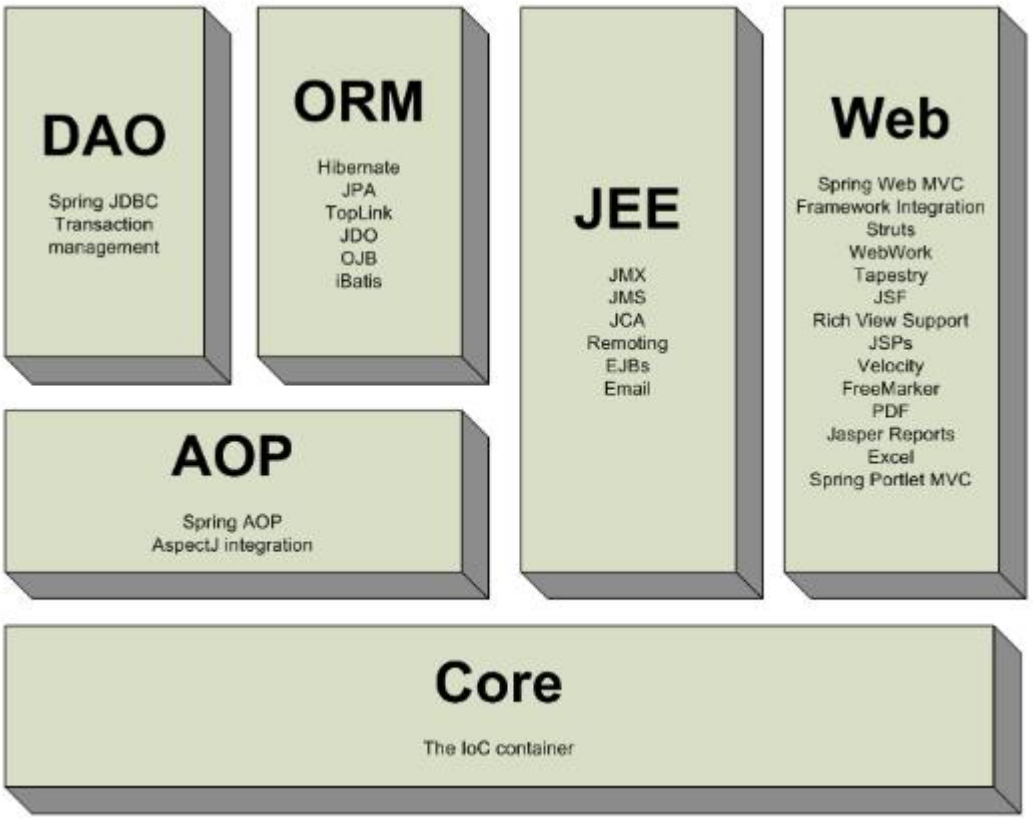
## 课程内容

## 1、 SpringMVC 概述

### Springmvc 是什么

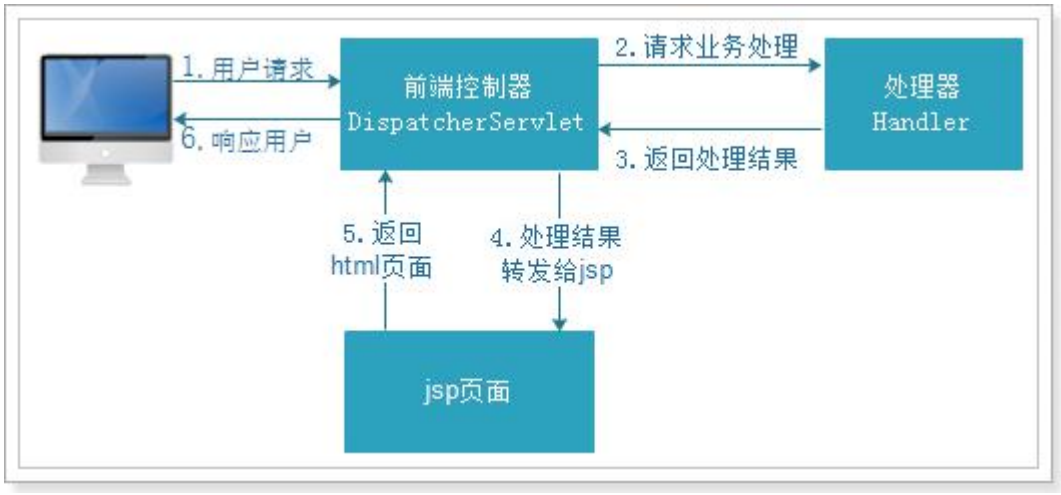
Spring web mvc 和 Struts2 都属于表现层的框架,它是 Spring 框架的一部分,我们可以从 Spring 的整体结构中看得出来,如下图：




## Springmvc 处理流程

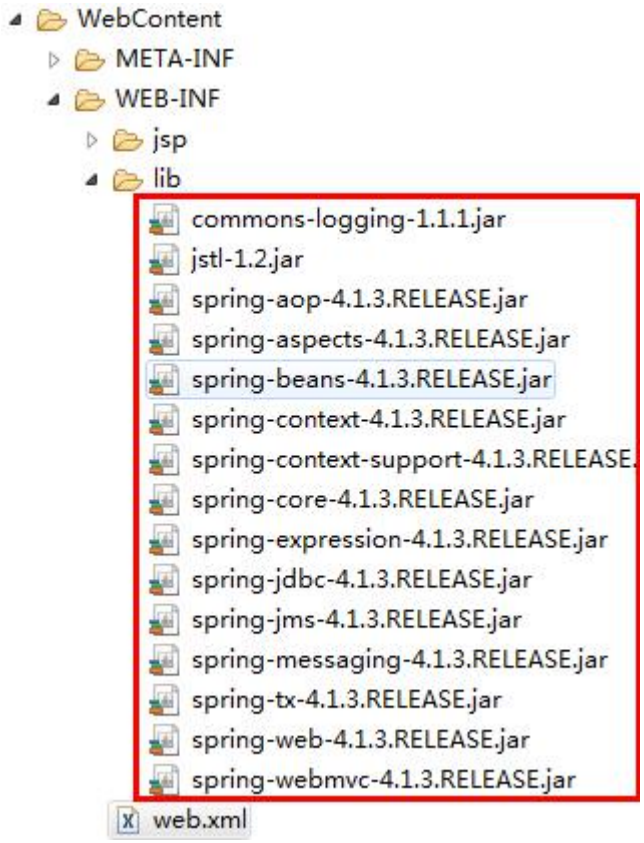
如下图所示：



# 2、 SpringMVC 入门

## 第一步：添加 jar 包：




## 第二步：在 web.xml 文件中配置 spring 的 DispatcherServlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID"
version="3.0">
  <display-name>SpringMVC-day01</display-name>
  <!-- 配置 springmvc 的 dispatcher Servlet 中央控制器 -->
  <servlet>
    <servlet-name>springmvc1</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servl
et-class>
    <!-- 配置 springmvc 的配置文件，默认情况下，springmvc 的配置文件名为：
```

```
servlet-name+"-servlet".xml
        并且放到/WEB-INF/目录下，例如本例，文件名必须定义为：
springmvc1-servlet.xml,为了便于管理所有的
        配置文件，将该配置文件放到类路径下的 config 中的 spring 目录下
    -->
    <init-param>
        <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring/applicationContext-*.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc1</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

## 第三步：在 config 类路径下定义 springmvc 的配置文件：


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.1.xsd">
<context:component-scan base-package="com.zelin.web.controller"/>
```

```
<!-- 1、配置 springmvc 的处理器映射器 -->
<!-- <bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMapping
HandlerMapping"/>
2、配置 springmvc 的处理器适配器
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMapping
HandlerAdapter"/> -->

<!-- 注意：上面 1、2、两步可以用下面的代替 -->
<mvc:annotation-driven/>
<!-- 3、配置视图解析器 -->
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <!-- 3.1)配置视图解析器的前缀部分 -->
    <property name="prefix" value="/WEB-INF/jsp/"/>
    <!-- 3.2)配置视图解析器的后缀部分 -->
    <property name="suffix" value=".jsp"/>
</bean>
```

逻辑视图名需要在controller中返回ModelAndView指定，比如逻辑视图名为bookList，则最终返回的jsp视图地址：

“WEB-INF/jsp/bookList.jsp”

最终 jsp 物理地址：前缀+**逻辑视图名**+后缀

## 第四步：定义后台处理器（控制器）：

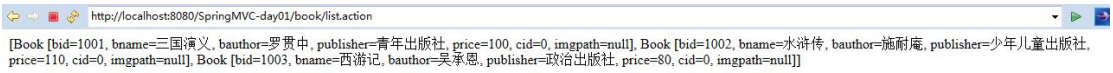
```
@Controller
@RequestMapping("/book")
public class BookController {
    @RequestMapping("/list")
    //列表图书
    public ModelAndView list(){
        List<Book> books = new ArrayList<>();
        books.add(new Book(1001,"三国演义","罗贯中","青年出版社",100));
        books.add(new Book(1002,"水浒传","施耐庵","少年儿童出版社",110));
        books.add(new Book(1003,"西游记","吴承恩","政治出版社",80));
        ModelAndView mv = new ModelAndView("listBooks", "books", books);
        return mv;
    }
}
```

## 第五步：定义视图/WEB-INF/jsp/listBooks.jsp：

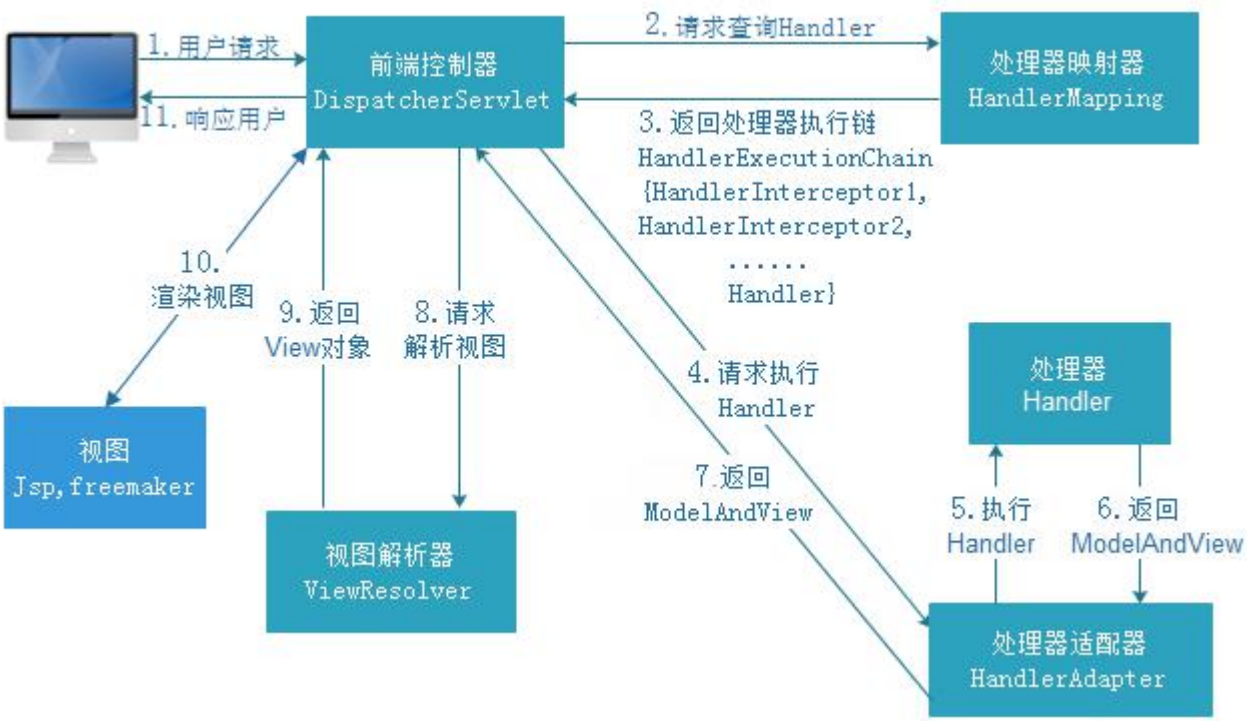
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    ${books }
</body>
</html>
```

## 第六步：运行效果：




# 3、 Spring 框架架构、运行原理分析

## 3.1）架构原理图如下：



## 3.2）架构流程说明：

1、 用户发送请求至前端控制器DispatcherServlet
2、 DispatcherServlet收到请求调用HandlerMapping处理器映射器。
3、 处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。
4、 DispatcherServlet通过HandlerAdapter处理器适配器调用处理器
5、 执行处理器(Controller，也叫后端控制器)。
6、 Controller执行完成返回ModelAndView
7、 HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet
8、 DispatcherServlet将ModelAndView传给ViewReslover视图解析器
9、 ViewReslover解析后返回具体View
10、DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）。
11、DispatcherServlet响应用户

## 3.3）组件功能说明：

以下组件通常使用框架提供实现：

◆ DispatcherServlet：前端控制器

用户请求到达前端控制器，它就相当于mvc模式中的c，dispatcherServlet是整个流程控制的中心，由它调用其它组件处理用户的请求，dispatcherServlet的存在降低了组件之间的耦合性。

◆ HandlerMapping：处理器映射器

HandlerMapping负责根据用户请求url找到Handler即处理器，springmvc提供了不同的映射器实现不同的映射方式，例如：配置文件方式，实现接口方式，注解方式等。

◆ Handler：处理器

Handler 是继DispatcherServlet前端控制器的后端控制器，在DispatcherServlet的控制下Handler对具体的用户请求进行处理。

由于Handler涉及到具体的用户业务请求，所以一般情况需要程序员根据业务需求开发Handler。

◆ HandlAdapter：处理器适配器

通过HandlerAdapter对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理器进行执行。

下图是许多不同的适配器，最终都可以使用usb接口连接

◆ ViewResolver：视图解析器

View Resolver负责将处理结果生成View视图，View Resolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象，最后对View进行渲染将处理结果通过页面展示给用户。

◆ View：视图

springmvc框架提供了很多的View视图类型的支持，包括：jstlView、freemarkerView、pdfView等。我们最常用的视图就是jsp。

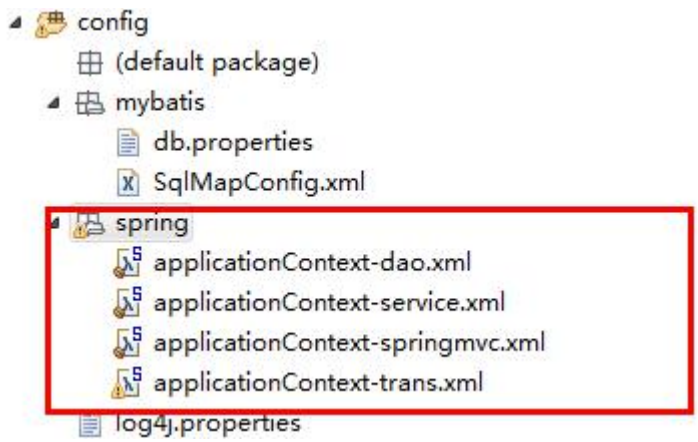
一般情况下需要通过页面标签或页面模版技术将模型数据通过页面展示给用户，需要由程序员根据业务需求开发具体的页面。

说明：在 springmvc 的各个组件中，处理器映射器、处理器适配器、视图解析器称为 springmvc 的三大组件。
需要用户开发的组件有 handler、view

# 4、 SSM 整合-实战案例

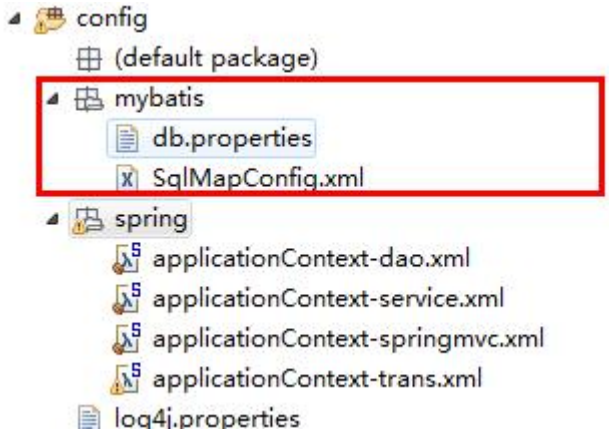
## 4.1）展示所有图书（含有图书类别）：

### 第一步：配置 Spring 相关的配置文件及 SpringMVC 的相关文件，结构如下：



## 第二步：配置 MyBatis 相关的配置结构如下：



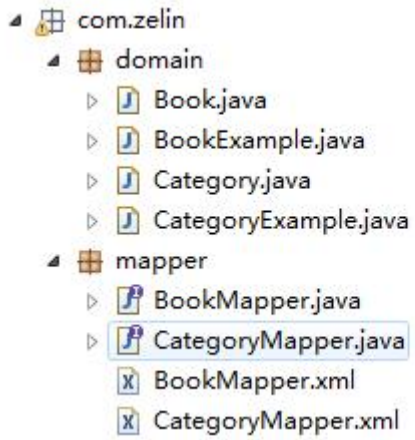

## 第三步：在 web.xml 文件中配置 spring 监听器及 SpringMVC 的 DispatcherServlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>SpringMVC-day01</display-name>
  <!-- 第一部分：配置 spring 相关 -->
  <!-- 0：加载 spring 的配置文件 -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring/applicationContext-*.xml</param-value>
  </context-param>
    <!-- 配置 spring 的监听器 -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener
     </listener-class>
  </listener>

  <!-- 第二部分：配置 springmvc 的 dispatcher Servlet 中央控制器 -->
  <servlet>
    <servlet-name>springmvc1</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
    <!-- 配置 springmvc 的配置文件，默认情况下，springmvc 的配置文件名为：
servlet-name+"servlet".xml
          并且放到/WEB-INF/目录下，例如本例，文件名必须定义为：springmvc1-servlet.xml,为
了便于管理所有的
          配置文件，将该配置文件放到类路径下的 config 中的 spring 目录下
```

```
     -->
    <init-param>
        <param-name>contextConfigLocation</param-name>

    <param-value>classpath:spring/applicationContext-springmvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc1</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>



  <!-- 第三部分：配置处理中文乱码的过滤器（Spring 自带） -->
  <filter>
    <filter-name>characterEncoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-cl
ass>
  </filter>
  <filter-mapping>
    <filter-name>characterEncoding</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

## 第四步：让 MyBatis 的自动生成工具生成相应的 mapper、domain 及 xml 文件：




注意：如何生成，请参见 mybatis《第三章讲义》

## 第五步：定义 BookService 及其实现类：

```
@Service
public class BookServiceImpl implements BookService {
    @Autowired
    private BookMapper bookMapper;
    @Override
    public List<Book> findAll() throws SQLException {
        return bookMapper.selectByExample(null);
    }
}
```

## 第六步：定义 BookController 处理器类：

```
@Controller
@RequestMapping("/book")
public class BookController {

    @Autowired
    private BookService bookService;
    @RequestMapping("/list")
    //列表图书
    public ModelAndView list() throws SQLException{
        //1.查询数据库
        List<Book> books = bookService.findAll();
        //2.构造 ModelAndView 对象
        ModelAndView mv = new ModelAndView("listBooks", "books", books);
        //3.返回 ModelAndView 对象
        return mv;
    }
```

## 第七步：定义/WEB-INF/jsp/bookList.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!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>列表所有学生</title>
    <jsp:include page="/bs.jsp"></jsp:include>
    <style>
    tr{
        text-align:center;
    }
    .first{
        font-weight:bold;
    }
    </style>
</head>
<body>

    <div class="container">
         <h1 class="page-header">BootStrap 框架+SSM 整合开发-<small>列表图书页面
</small> </h1>
         <div class="panel panel-primary">
          <div class="panel-heading">
            <div class="h4"><span class=" glyphicon
glyphicon-list-alt"></span> 列表所有图书信息</div>
          </div>
          <table class="table table-striped table-bordered">
            <!-- 模糊查询 -->
            <tr>
                <td colspan=7 style="text-align:left">
                        <%-- <form class="form-inline"

action="${pageContext.request.contextPath }/Student?method=search"
                              method="post">
                            <div class="form-group">
                                <label for="sname">姓名:</label>
                                <input type="text" class="form-control"
id="sname" name="sname" value="${param.sname }">
                            </div>
```

```
                        <div class="form-group">
                            <label for="addr">住址:</label>
                            <input type="text" class="form-control"
id="addr" name="addr" value="${param.addr }">
                        </div>
                        <div class="form-group">
                            <label for="cid">所在班级：</label>
                            <select name="cid" class="form-control">
                                <option value="0">所有班级</option>
                                <c:forEach items="${classes }" var="c">
                                    <option value="${c.cid }"
${c.cid==param.cid?'selected':'' }>${c.cname }</option>
                                </c:forEach>

                            </select>
                        </div>
                        <button type="submit" class="glyphicon
glyphicon-search btn btn-default"> 查询</button>
                    </form> --%>
                </td>
            </tr>
            <tr class="first">
            <td>编号</td><td>图书名称 </td><td>作者</td><td>出版社</td>
            <td>单价</td><td>类别</td><td>操作</td>
            </tr>
            <c:forEach items="${books }" var="book">
            <tr>
                <td>${book.bid }</td>
                <td>${book.bname}</td>
                <td>${book.bauthor }</td>
                <td>${book.publisher}</td>
                <td>${book.price}</td>
                <td>${book.cid }</td>
                <td>
                <a
href="${pageContext.request.contextPath }/Book?method=toupdate&bid=${book.bi
d}" class="btn btn-info glyphicon glyphicon-edit">修改</a>
                <a
href="${pageContext.request.contextPath }/Book?method=delete&bid=${book.bid}
" class="btn btn-info glyphicon glyphicon-remove-circle"
                    onclick="return confirm('真的要删除吗?')">删除</a>
                </td>
            </tr>
            </c:forEach>
```

```
            <tr>
            <td colspan=7 style="text-align:right">
                <a
href="${pageContext.request.contextPath }/Book?method=toadd" class="btn
btn-default glyphicon glyphicon-plus-sign">添加图书</a>
            </td>
            </tr>
          </table>
        </div>
    </div>
</body>
</html>
```

## 第八步：测试并运行：


## 第九步：查询关联的图书类别，修改 BookMapper.xml 文件：

```
<mapper namespace="com.zelin.mapper.BookMapper" >
  <resultMap id="BaseResultMap" type="com.zelin.domain.Book" >
    <id column="bid" property="bid" jdbcType="INTEGER" />
    <result column="bname" property="bname" jdbcType="VARCHAR" />
    <result column="bauthor" property="bauthor" jdbcType="VARCHAR" />
    <result column="publisher" property="publisher" jdbcType="VARCHAR" />
    <result column="price" property="price" jdbcType="INTEGER" />
   <!--  <result column="cid" property="cid" jdbcType="INTEGER" /> -->
    <result column="imgpath" property="imgpath" jdbcType="VARCHAR" />
    <association property="category" column="cid" javaType="category" select="findCategoryByCid"/>
  </resultMap>
  <select id="findCategoryByCid" resultType="category" parameterType="int">
        select * from category where cid = #{cid}
  </select>
```

查询关联表的数据，用一个子查询实现

第十步：最终页面效果如下：


## 4.2）模糊查询：

第一步：在 BookService 接口中定义条件查询方法并实现：

```
public interface BookService {

    /**
     * 查询所有图书
     * @return
     * @throws SQLException
     */
    public List<Book> findAll() throws SQLException;

    /**
     * 根据查询条件查询图书
     */
    public List<Book> findBooksByWords(Book book) throws SQLException;
}
```

```
@Service
public class BookServiceImpl implements BookService {

    @Autowired
    private BookMapper bookMapper;
    @Override
    public List<Book> findAll() throws SQLException {
        return bookMapper.selectByExample(null);
    }
    @Override
    public List<Book> findBooksByWords(Book book) throws SQLException {
        BookExample example = new BookExample();
         Criteria criteria = example.createCriteria()
                             .andBauthorLike("%"+book.getBauthor()+"%")
                             .andPublisherLike("%"+book.getPublisher()+"%");
         if(book.getCategory().getCid() != 0){
             criteria.andCidEqualTo(book.getCategory().getCid());
         }
        return bookMapper.selectByExample(example);

    }
}
```

## 第二步：定义 BookController 中的 search 方法：

```
//获取请求数据方法一（参数使用HttpServletRequest）
/*@RequestMapping("/search")
public ModelAndView search(HttpServletRequest request) throws Exception{
    //1.收集表单数据
    Book book = new Book();
    BeanUtils.populate(book, request.getParameterMap());

    return null;
}*/
//获取请求参数方法二（传递过来的参数会根据参数对象的属性进行一一对应赋值，
//比如：请求参数bname,会找到此时方法中的形参Book类中的bname并调用setBname()进行赋值）
@RequestMapping("/search")
public ModelAndView search(Book book) throws Exception{
    List<Book> books = bookService.findBooksByWords(book);
    return new ModelAndView("book/list", "books", books);
}
```

## 第三步：定义/WEB-INF/jsp/book/list.jsp 视图：

```
<td colspan=7 style="text-align:left">
                    <form class="form-inline"
     action="${pageContext.request.contextPath }/book/search.action"
                          method="post">
                        <div class="form-group">
                            <label for="bauthor">图书作者:</label>
                            <input type="text" class="form-control"
id="bauthor" name="bauthor" value="${param.bauthor }">
                        </div>
                        <div class="form-group">
                            <label for="publisher">出版社:</label>
```

```
                        <input type="text" class="form-control"
id="publisher" name="publisher" value="${param.publisher }">
                    </div>
                    <div class="form-group">
                        <label for="cid">图书类别：</label>
                        <select name="category.cid"
class="form-control">
                            <option value="0">所有类别</option>
                            <c:forEach items="${cates }" var="c">
                                <option value="${c.cid }"
${c.cid==param.cid?'selected':'' }>${c.cname }</option>
                            </c:forEach>

                        </select>
                    </div>
                    <button type="submit" class="glyphicon
glyphicon-search btn btn-default"> 查询</button>
                </form>
            </td>
```

第四步：运行效果展示：


## 4.3）添加：

第一步：在 BookServiceImpl.java 中完成添加图书的方法的定义：

```
@Override
public void insertBook(Book book) throws SQLException {
    bookMapper.insert(book);
}
```

## 第二步：在 BookController 控制器层完成 add（）方法的定义：

```
//添加图书
@RequestMapping("/add")
public String add(Book book) throws Exception{
    bookService.insertBook(book);
    return "redirect:/book/list.action";
}
```

注意：添加完成后需使用重定向功能。

## 第三步：定义/WEB-INF/jsp/book/add.jsp 页面：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!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>添加图书页面</title>
<jsp:include page="/bs.jsp"></jsp:include>
</head>
<body>
    <div class="container">
        <h1 class="page-header">
            图书管理系统-<small>添加图书功能</small>
        </h1>
        <div class="col-md-6">
            <div class="panel panel-primary">
                <div class="panel-heading">
                    <div class="h4">图书管理-添加图书</div>
                </div>
                <div class="panel-body">
                    <form

    action="${pageContext.request.contextPath }/book/add.action"
                        method="post" class="form-horizontal">
                        <div class="form-group">
                            <label for="bname" class="col-sm-2 control-label">
图书名称 </label>
                            <div class="col-sm-10">
                                <input type="text" name="bname"
class="form-control" id="bname">
                            </div>
                        </div>
                        <div class="form-group">
```

```
                        <label for="bauthor" class="col-sm-2
control-label">图书作者 </label>
                        <div class="col-sm-10">
                            <input type="text" name="bauthor"
class="form-control" id="bauthor">
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="publisher" class="col-sm-2
control-label">出版社 </label>
                        <div class="col-sm-10">
                            <input type="text" name="publisher"
class="form-control" id="publisher">
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="price" class="col-sm-2 control-label">
图书单价 </label>
                        <div class="col-sm-10">
                            <input type="text" name="price"
class="form-control" id="price">
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="cid" class="col-sm-2 control-label">
图书类别 </label>
                        <div class="col-sm-10">
                            <select name="category.cid"
class="form-control">
                                <c:forEach items="${cates }" var="c">
                                    <option
value="${c.cid }">${c.cname }</option>
                                </c:forEach>
                            </select>
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="imgpath" class="col-sm-2
control-label">图书封面 </label>
                        <div class="col-sm-10">
                            <input type="text" name="imgpath"
class="form-control" id="imgpath">
                        </div>
                    </div>
```

```
                    <div class="form-group">
                        <input type="submit" value="添加" class="btn
btn-primary col-md-offset-1 col-md-4 ">
                        <input type="reset" value="清空" class="btn
btn-primary col-md-offset-1 col-md-4">
                    </div>
                </form>
            </div>

        </div>
    </div>
</div>
</body>
</html>
```

## 第四步：修改 BookMapper.xml 文件：

```
<insert id="insert" parameterType="com.zelin.domain.Book" >
  insert into book (bid, bname, bauthor,
    publisher, price, cid,
    imgpath)
  values (#{bid,jdbcType=INTEGER}, #{bname,jdbcType=VARCHAR}, #{bauthor,jdbcType=VARCHAR},
    #{publisher,jdbcType=VARCHAR}, #{price,jdbcType=INTEGER}, #{category.cid,jdbcType=INTEGER},
    #{imgpath,jdbcType=VARCHAR})
</insert>
```

第五步：页面效果图：


## 4.4）删除：

第一步：定义 BookServiceImpl.java，实现删除方法。

```
@Override
    public void deleteBook(int bid) throws SQLException {
        bookMapper.deleteByPrimaryKey(bid);
    }
```

第二步：定义 BookController 控制器中的删除方法。

```
    //删除图书
    @RequestMapping("/delete")
    public String delete(int bid) throws Exception{
        bookService.deleteBook(bid);
        return "redirect:/book/list.action";
    }
```

第三步：定义 list.jsp 页面中的删除功能：

```
<c:forEach items="${books }" var="book">
```

```
                <tr>
                    <td>${book.bid }</td>
                    <td>${book.bname}</td>
                    <td>${book.bauthor }</td>
                    <td>${book.publisher}</td>
                    <td>${book.price}</td>
                    <td>${book.category.cname }</td>
                    <td>
                    <a
href="${pageContext.request.contextPath }/book/toupdate.action?bid=${boo
k.bid}" class="btn btn-info glyphicon glyphicon-edit">修改</a>
                    <a
href="${pageContext.request.contextPath }/book/delete.action?bid=${book.
bid}" class="btn btn-info glyphicon glyphicon-remove-circle"
                        onclick="return confirm('真的要删除吗?')">删除</a>
                    </td>
                </tr>
            </c:forEach>
            <tr>
                <td colspan=7 style="text-align:right">
                    <a
href="${pageContext.request.contextPath }/book/addUI.action" class="btn
btn-default glyphicon glyphicon-plus-sign">添加图书</a>
                </td>
            </tr>
          </table>
        </div>
```

第四步：运行效果略。

## 4.5）修改：

第一步：定义 BookServiceImpl.java 实现类中关于修改的两个方法：

```
@Override
    public Book findBookByBid(int bid) throws SQLException {
        return bookMapper.selectByPrimaryKey(bid);
    }
    @Override
    public void updateBook(Book book) throws SQLException {
        bookMapper.updateByPrimaryKey(book);
    }
```

## 第二步：修改 BookMapper.xml 文件：

```
<update id="updateByPrimaryKey" parameterType="com.zelin.domain.Book" >
  update book
  set bname = #{bname,jdbcType=VARCHAR},
    bauthor = #{bauthor,jdbcType=VARCHAR},
    publisher = #{publisher,jdbcType=VARCHAR},
    price = #{price,jdbcType=INTEGER},
    cid = #{category.cid,jdbcType=INTEGER},
    imgpath = #{imgpath,jdbcType=VARCHAR}
  where bid = #{bid,jdbcType=INTEGER}
</update>
```

## 第三步：定义控制器方法 updateUI 和 update

```
//到修改页面
    //@RequestParam 注解，一般用于在传递的参数与方法的形参名不一样时使用，
参数含义如下：
    //defaultValue:代表没有得到外界传递的参数值时的默认值
    //required：是此参数必须有值
    //value:代表外界传递过来的参数名称
    @RequestMapping("/updateUI")
    public String
toUpdate(@RequestParam(defaultValue="1",required=true,value="bid1")
Integer bid,Model model)
            throws Exception{
        Book book = bookService.findBookByBid(bid);
        model.addAttribute("book",book);
        model.addAttribute("cates", categoryService.findAll());
        return "book/update";
    }
    //修改图书
    @RequestMapping("/update")
    public String update(Book book) throws Exception{
        bookService.updateBook(book);
        return "forward:/book/list.action";
    }
```

## 第四步：定义/WEB-INF/jsp/book/update.jsp 页面：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!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>修改图书页面</title>
```

```
<jsp:include page="/bs.jsp"></jsp:include>
</head>
<body>
    <div class="container">
        <h1 class="page-header">
            图书管理系统-<small>修改图书功能</small>
        </h1>
        <div class="col-md-6">
            <div class="panel panel-primary">
                <div class="panel-heading">
                    <div class="h4">图书管理-修改图书</div>
                </div>
                <div class="panel-body">
                    <form

    action="${pageContext.request.contextPath }/book/update.action"
                        method="post" class="form-horizontal">
                        <input type="hidden" name="bid" value="${book.bid }">
                        <div class="form-group">
                            <label for="bname" class="col-sm-2 control-label">
图书名称 </label>
                            <div class="col-sm-10">
                                <input type="text" name="bname"
class="form-control" id="bname"
                                    value="${book.bname }">
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="bauthor" class="col-sm-2
control-label">图书作者 </label>
                            <div class="col-sm-10">
                                <input type="text" name="bauthor"
class="form-control" id="bauthor"
                                    value="${book.bauthor }">
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="publisher" class="col-sm-2
control-label">出版社 </label>
                            <div class="col-sm-10">
                                <input type="text" name="publisher"
class="form-control" id="publisher"
                                    value="${book.publisher }">
                            </div>
```

```
                        </div>
                        <div class="form-group">
                            <label for="price" class="col-sm-2 control-label">
图书单价 </label>
                            <div class="col-sm-10">
                                <input type="text" name="price"
class="form-control" id="price"
                                    value="${book.price }">
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="cid" class="col-sm-2 control-label">
图书类别 </label>
                            <div class="col-sm-10">
                                <select name="category.cid"
class="form-control">
                                    <c:forEach items="${cates }" var="c">
                                        <option value="${c.cid }"
${book.category.cid==c.cid?'selected':'' }>${c.cname }</option>
                                    </c:forEach>
                                </select>
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="imgpath" class="col-sm-2
control-label">图书封面 </label>
                            <div class="col-sm-10">
                                <input type="text" name="imgpath"
class="form-control" id="imgpath"
                                    value="${book.imgpath }">
                            </div>
                        </div>
                        <div class="form-group">
                            <input type="submit" value="修改" class="btn
btn-primary col-md-offset-1 col-md-4 ">
                            <input type="reset" value="清空" class="btn
btn-primary col-md-offset-1 col-md-4">
                        </div>
                    </form>
                </div>

            </div>
        </div>
    </div>
```

```
</body>
</html>
```

第五步：运行效果如下：