

1、环境

- IDEA
- MySql 5.5.29
- Tomcat 8.5.28
- Maven 3.5.4

2、数据库环境

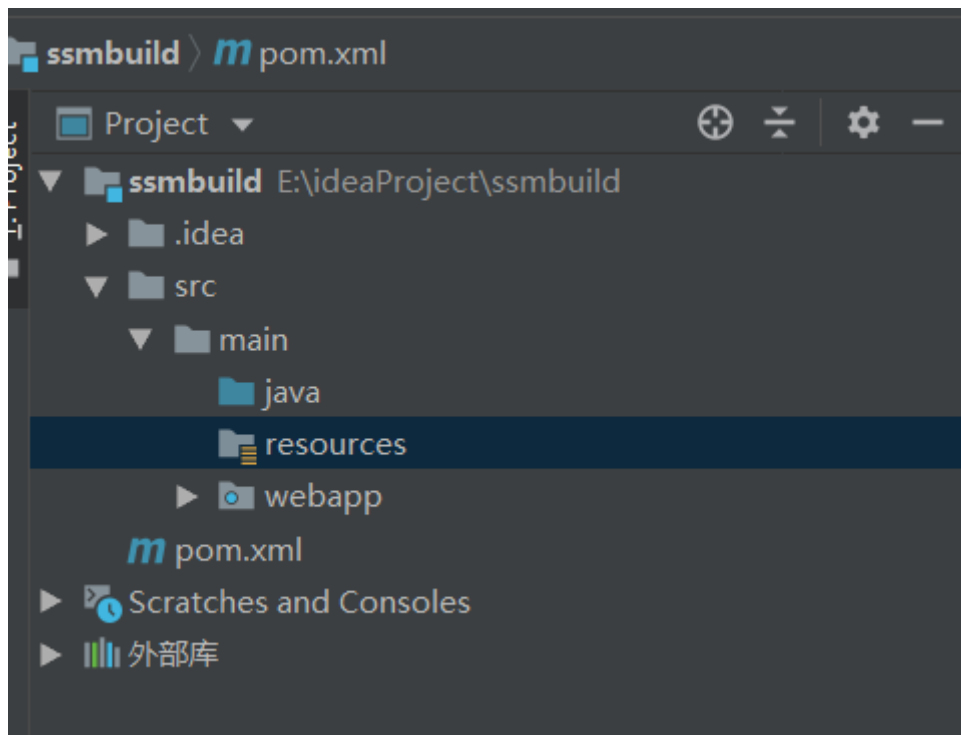
创建一个存放书籍的数据库表

```
1  create database ssmbuild;
2
3  use ssmbuild;
4
5  drop table if exists books;
6
7  create table books(
8  bookID int(10) primary key auto_increment comment '书id',
9  bookName varchar(100) not null comment '书名',
10 bookCounts int(11) not null comment '数量',
11 detail varchar(200) not null comment '描述'
12 )engine=innodb default charset=utf8;
13
14 insert into books(bookID,bookName,bookCounts,detail) values
15 (1,'Java',10,'从入门到放弃'),
16 (2,'MySQL',5,'从删库到跑咯'),
17 (3,'linux',3,'从入门到进牢');
18
19 select * from books;
```

3、基本环境搭建

1. 新建一Maven-web项目！ ssmbuild

创建java文件夹标记为源，resources标记为Resources根



2. 导入相关的pom依赖!

```
1  <!--依赖: junit, 数据库驱动, 连接池, servlet, jsp, mybatis, mybatis-  
    spring,spring-->  
2  <dependencies>  
3      <!--Junit-->  
4      <dependency>  
5          <groupId>junit</groupId>  
6          <artifactId>junit</artifactId>  
7          <version>4.12</version>  
8      </dependency>  
9      <!--数据库驱动-->  
10     <dependency>  
11         <groupId>mysql</groupId>  
12         <artifactId>mysql-connector-java</artifactId>  
13         <version>5.1.47</version>  
14     </dependency>  
15     <!-- 数据库连接池 c3p0 -->  
16     <dependency>  
17         <groupId>com.mchange</groupId>  
18         <artifactId>c3p0</artifactId>  
19         <version>0.9.5.2</version>  
20     </dependency>  
21  
22     <!--Servlet - JSP -->  
23     <dependency>  
24         <groupId>javax.servlet</groupId>  
25         <artifactId>servlet-api</artifactId>  
26         <version>2.5</version>  
27     </dependency>  
28     <dependency>  
29         <groupId>javax.servlet.jsp</groupId>  
30         <artifactId>jsp-api</artifactId>  
31         <version>2.2</version>  
32     </dependency>  
33     <dependency>
```

```

34     <groupId>javax.servlet</groupId>
35     <artifactId>jstl</artifactId>
36     <version>1.2</version>
37 </dependency>
38
39 <!--Mybatis-->
40 <dependency>
41     <groupId>org.mybatis</groupId>
42     <artifactId>mybatis</artifactId>
43     <version>3.5.2</version>
44 </dependency>
45 <dependency>
46     <groupId>org.mybatis</groupId>
47     <artifactId>mybatis-spring</artifactId>
48     <version>2.0.2</version>
49 </dependency>
50
51 <!--Spring-->
52 <dependency>
53     <groupId>org.springframework</groupId>
54     <artifactId>spring-webmvc</artifactId>
55     <version>5.1.9.RELEASE</version>
56 </dependency>
57 <dependency>
58     <groupId>org.springframework</groupId>
59     <artifactId>spring-jdbc</artifactId>
60     <version>5.1.9.RELEASE</version>
61 </dependency>
62 <!--lombok-->
63 <dependency>
64     <groupId>org.projectlombok</groupId>
65     <artifactId>lombok</artifactId>
66     <version>1.18.8</version>
67 </dependency>
68 <!--aop织入-->
69 <dependency>
70     <groupId>org.aspectj</groupId>
71     <artifactId>aspectjweaver</artifactId>
72     <version>1.9.4</version>
73 </dependency>
74 </dependencies>

```

3. Maven资源过滤设置

```

1 <build>
2     <resources>
3         <resource>
4             <directory>src/main/java</directory>
5             <includes>
6                 <include>**/*.properties</include>
7                 <include>**/*.xml</include>
8             </includes>
9             <filtering>>false</filtering>
10        </resource>
11        <resource>
12            <directory>src/main/resources</directory>
13            <includes>

```

```

14         <include>**/*.properties</include>
15         <include>**/*.xml</include>
16     </includes>
17     <filtering>>false</filtering>
18 </resource>
19 </resources>
20 </build>

```

4. 建立基本结构和配置框架!

- com.zh.pojo
- com.zh.mapper
- com.zh.service
- com.zh.controller
- mybatis-config.xml

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration
3     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6
7 </configuration>

```

- applicationContext.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans
5                           http://www.springframework.org/schema/beans/spring-
6                           beans.xsd">
7 </beans>

```

4、Mybatis层

1. 数据库配置文件 **database.properties**

```

1 jdbc.driver=com.mysql.jdbc.Driver
2 jdbc.url=jdbc:mysql://localhost:3306/ssmbuild?
  useSSL=true&useUnicode=true&characterEncoding=utf8
3 jdbc.username=root
4 jdbc.password=123456

```

2. IDEA关联数据库

3. 编写MyBatis的核心配置文件mybatis-config.xml

注意：这里的配置也可以配置在applicationContext.xml中

```

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

```

```

2  <!DOCTYPE configuration
3      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-config.dtd">
5  <configuration>
6
7      <!--配置数据源，交给spring-->
8
9      <settings>
10         <setting name="logImpl" value="STDOUT_LOGGING"/>
11     </settings>
12
13     <typeAliases>
14         <package name="com.zh.pojo"/>
15     </typeAliases>
16
17     <mappers>
18         <mapper class="com.zh.mapper.BookMapper"/>
19     </mappers>
20
21 </configuration>

```

4. 编写数据库对应的实体类 com.kuang.pojo.Books
使用lombok插件!

```

1  package com.zh.pojo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  @Data
8  @AllArgsConstructor
9  @NoArgsConstructor
10 public class Books {
11
12     private int bookID;
13     private String bookName;
14     private int bookCounts;
15     private String detail;
16
17 }

```

5. 编写Dao层的 Mapper接口!

```

1  package com.zh.mapper;
2
3  import com.zh.pojo.Books;
4  import org.apache.ibatis.annotations.Param;
5
6  import java.util.List;
7
8  public interface BookMapper {
9
10     //增加一本书
11     int addBook(Books books);
12

```

```

13 //删除
14 int deleteBookById(@Param("id") int id);
15
16 //修改
17 int updateBook(Books books);
18
19 //查询
20 Books findById(@Param("id") int id);
21
22 List<Books> findAll();
23
24 }

```

6. 编写接口对应的 Mapper.xml 文件。需要导入MyBatis的包;

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.zh.mapper.BookMapper">
7
8     <insert id="addBook" parameterType="books">
9         insert into ssmbuild.books (bookName, bookCounts, detail)
10         values (#{bookName}, #{bookCounts}, #{detail});
11     </insert>
12
13     <delete id="deleteBookById" parameterType="int">
14         delete from ssmbuild.books where bookID = #{id}
15     </delete>
16
17     <update id="updateBook" parameterType="books">
18         update ssmbuild.books set bookName = #{bookName},
19         bookCounts = #{bookCounts}, detail = #{detail}
20         where bookID = #{bookID};
21     </update>
22
23     <select id="findById" parameterType="int" resultType="books">
24         select * from ssmbuild.books where bookID = #{id};
25     </select>
26
27     <select id="findAll" resultType="books">
28         select * from ssmbuild.books;
29     </select>
30 </mapper>

```

7. 实现类service接口

```

1 package com.zh.service;
2
3 import com.zh.pojo.Books;
4 import org.apache.ibatis.annotations.Param;
5
6 import java.util.List;
7
8 public interface BookService {

```

```

9
10 //增加一本书
11 int addBook(Books books);
12
13 //删除
14 int deleteBookById(int id);
15
16 //修改
17 int updateBook(Books books);
18
19 //查询
20 Books findById(int id);
21
22 List<Books> findAll();
23
24 }

```

8. service实现方法

```

1 package com.zh.service.impl;
2
3 import com.zh.mapper.BookMapper;
4 import com.zh.pojo.Books;
5 import com.zh.service.BookService;
6
7 import java.util.List;
8
9 public class BookServiceImpl implements BookService{
10
11     //调用dao层的操作，设置一个set接口，方便Spring管理
12     private BookMapper bookMapper;
13     public void setBookMapper(BookMapper bookMapper) {
14         this.bookMapper = bookMapper;
15     }
16
17     @Override
18     public int addBook(Books books) {
19         return bookMapper.addBook(books);
20     }
21
22     @Override
23     public int deleteBookById(int id) {
24         return bookMapper.deleteBookById(id);
25     }
26
27     @Override
28     public int updateBook(Books books) {
29         return bookMapper.updateBook(books);
30     }
31
32     @Override
33     public Books findById(int id) {
34         return bookMapper.findById(id);
35     }
36
37     @Override
38     public List<Books> findAll() {

```

```

39         return bookMapper.findAll();
40     }
41 }

```

8、Spring层

注意：applicationContext.xml自动导入的头文件不全，需要手动补全

1. 配置Spring整合MyBatis，我们这里数据源使用c3p0连接池；
2. 我们去编写Spring整合Mybatis的相关的配置文件

```

1  <!--Spring整合mapper层-->
2  <!--1.关联数据库配置文件-->
3  <context:property-placeholder
4      location="classpath:database.properties"/>
5
6  <!--2.c3p0连接池-->
7  <bean id="dataSource"
8      class="com.mchange.v2.c3p0.ComboPooledDataSource">
9      <property name="driverClass" value="${jdbc.driver}"/>
10     <property name="jdbcUrl" value="${jdbc.url}"/>
11     <property name="user" value="${jdbc.username}"/>
12     <property name="password" value="${jdbc.password}"/>
13
14     <!-- c3p0连接池的私有属性 -->
15     <property name="maxPoolSize" value="30"/>
16     <property name="minPoolSize" value="10"/>
17     <!-- 关闭连接后不自动commit -->
18     <property name="autoCommitOnClose" value="false"/>
19     <!-- 获取连接超时时间 -->
20     <property name="checkoutTimeout" value="10000"/>
21     <!-- 当获取连接失败重试次数 -->
22     <property name="acquireRetryAttempts" value="2"/>
23 </bean>
24
25 <!--3.sqlSessionFactory-->
26 <bean id="sqlSessionFactory"
27     class="org.mybatis.spring.SqlSessionFactoryBean">
28     <property name="dataSource" ref="dataSource"/>
29     <!--绑定mybatis配置文件-->
30     <property name="configLocation" value="classpath:mybatis-
31         config.xml"/>
32 </bean>
33
34 <!--4.配置mapper接口扫描包，动态实现了Mapper接口可以注入到spring容器中-->
35 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
36     <!--注入sqlSessionFactory-->
37     <property name="sqlSessionFactoryBeanName"
38         value="sqlSessionFactory"/>
39     <!--要扫描的mapper包-->
40     <property name="basePackage" value="com.zh.mapper"/>
41 </bean>

```

3. Spring整合service层


```

1 <!--spring整合service层-->
2 <!--1.扫描service下的包-->
3 <context:component-scan base-package="com.zh.service"/>
4
5 <!--2.将所有业务注入到spring中，可以通过配置和注解-->
6 <bean id="bookService" class="com.zh.service.impl.BookServiceImpl">
7     <property name="bookMapper" ref="bookMapper"/>
8 </bean>
9
10 <!--3.声明式事务配置-->
11 <bean id="transactionManager"
12     class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
13     >
14     <!--注入数据源-->
15     <property name="dataSource" ref="dataSource"/>
16 </bean>
17 <!--4.aop事务支持-->
18 <!--结合aop实现事务的织入-->
19 <tx:advice id="txAdvice" transaction-manager="transactionManager">
20     <tx:attributes>
21         <tx:method name="*" propagation="REQUIRED"/>
22     </tx:attributes>
23 </tx:advice>
24 <!--配置事务切入-->
25 <aop:config>
26     <aop:pointcut id="txPointcut" expression="execution(*
    com.zh.mapper.*.*(..))"/>
27     <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut"/>
28 </aop:config>

```

9、SpringMVC层

1. web.xml

注意：maven创建的web项目，需要更改web.xml的头文件，要不配置filter标签报错

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5     http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
6     version="4.0">
7
8     <display-name>Archetype Created Web Application</display-name>
9
10    <!--DispatcherServlet-->
11    <servlet>
12        <servlet-name>springmvc</servlet-name>
13        <servlet-
14        class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
15        <init-param>
16            <param-name>contextConfigLocation</param-name>
17            <param-value>classpath:applicationContext.xml</param-value>
18        </init-param>
19        <load-on-startup>1</load-on-startup>
20    </servlet>

```

```

20     <servlet-mapping>
21         <servlet-name>springmvc</servlet-name>
22         <url-pattern>/</url-pattern>
23     </servlet-mapping>
24
25     <!--乱码过滤-->
26     <filter>
27         <filter-name>encodingFilter</filter-name>
28         <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
29         <init-param>
30             <param-name>encoding</param-name>
31             <param-value>utf-8</param-value>
32         </init-param>
33     </filter>
34     <filter-mapping>
35         <filter-name>encodingFilter</filter-name>
36         <url-pattern>/*</url-pattern>
37     </filter-mapping>
38
39     <!--session-->
40     <session-config>
41         <session-timeout>15</session-timeout>
42     </session-config>
43 </web-app>

```

2. applicationContext.xml

```

1  <!--SpringMVC-->
2  <!--1.注解驱动-->
3  <mvc:annotation-driven/>
4  <!--2.静态资源过滤-->
5  <mvc:default-servlet-handler/>
6  <!--3.扫描包:controller-->
7  <context:component-scan base-package="com.zh.controller" />
8  <!--4.视图解析器-->
9  <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolve
r">
10     <property name="prefix" value="/WEB-INF/jsp/" />
11     <property name="suffix" value=".jsp" />
12 </bean>

```

10、Controller层 和 视图层编写

1. 编写首页 index.jsp。进入查询全部

```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
2  <!DOCTYPE HTML>
3  <html>
4  <head>
5      <title>首页</title>
6      <style type="text/css">

```

```

7         a {
8             text-decoration: none;
9             color: black;
10            font-size: 18px;
11        }
12        h3 {
13            width: 180px;
14            height: 38px;
15            margin: 100px auto;
16            text-align: center;
17            line-height: 38px;
18            background: deepskyblue;
19            border-radius: 4px;
20        }
21    </style>
22 </head>
23 <body>
24
25 <h3>
26     <a href="${pageContext.request.contextPath}/book/allBook">点击进入列
    表页</a>
27 </h3>
28 </body>
29 </html>

```

2. BookController 类编写， 方法一： 查询全部书籍

```

1  @Controller
2  @RequestMapping("/book")
3  public class BookController {
4
5      //controller 调 service层
6      @Autowired
7      @Qualifier("bookService")
8      private BookService bookService;
9
10     @RequestMapping("/allBook")
11     public String allBook(Model model){
12
13         List<Books> list = bookService.findAll();
14
15         model.addAttribute("list",list);
16
17         return "allBook";
18     }
19 }

```

3. 书籍列表页面 allbook.jsp

```

1  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
3  <html>
4  <head>
5      <title>书籍列表</title>
6      <meta name="viewport" content="width=device-width, initial-
    scale=1.0">

```

```

7      <!-- 引入 Bootstrap -->
8      <link
href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
9  </head>
10 <body>
11
12 <div class="container">
13
14     <div class="row clearfix">
15         <div class="col-md-12 column">
16             <div class="page-header">
17                 <h1>
18                     <small>书籍列表 -- 显示所有书籍</small>
19                 </h1>
20             </div>
21         </div>
22     </div>
23
24     <div class="row">
25         <div class="col-md-4 column">
26             <a class="btn btn-primary"
href="${pageContext.request.contextPath}/book/toAddBook">新增</a>
27             <a class="btn btn-primary"
href="${pageContext.request.contextPath}/book/allBook">显示全部</a>
28         </div>
29         <div class="col-md-8 column">
30             <!-- 查询书籍 -->
31             <form
action="${pageContext.request.contextPath}/book/findByName"
style="float:right" class="form-inline" method=post>
32                 <span style="color: red;font-weight: bold" >${msg}
</span>
33                 <input type="text" name="bookName" class="form-control"
placeholder="请输入要查询的书名" >
34                 <input type="submit" value="查询" class="btn btn-
primary">
35             </form>
36         </div>
37     </div>
38
39     <div class="row clearfix">
40         <div class="col-md-12 column">
41             <table class="table table-hover table-striped">
42                 <thead>
43                     <tr>
44                         <th>书籍编号</th>
45                         <th>书籍名字</th>
46                         <th>书籍数量</th>
47                         <th>书籍详情</th>
48                         <th>操作</th>
49                     </tr>
50                 </thead>
51
52                 <tbody>
53                     <c:forEach var="book"
items="${requestScope.get('list')}">
54                         <tr>

```

```

55         <td>${book.getBookID()}</td>
56         <td>${book.getBookName()}</td>
57         <td>${book.getBookCounts()}</td>
58         <td>${book.getDetail()}</td>
59         <td>
60             <a
href="${pageContext.request.contextPath}/book/toUpdateBook?
id=${book.getBookID()}">更改</a> |
61             <a
href="${pageContext.request.contextPath}/book/del/${book.getBookID()}">
删除</a>
62         </td>
63     </tr>
64 </c:forEach>
65 </tbody>
66 </table>
67 </div>
68 </div>
69 </div>

```

4. BookController 类编写，方法二：去添加书籍页面

```

1 @RequestMapping("/toAddBook")
2 public String toAddBook(){
3
4     return "addBook";
5 }

```

5. 添加书籍页面：addBook.jsp

```

1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2 <%@ page contentType="text/html;charset=UTF-8" language="java" %>
3
4 <html>
5 <head>
6     <title>新增书籍</title>
7     <meta name="viewport" content="width=device-width, initial-
scale=1.0">
8     <!-- 引入 Bootstrap -->
9     <link
href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
10 </head>
11 <body>
12 <div class="container">
13
14     <div class="row clearfix">
15         <div class="col-md-12 column">
16             <div class="page-header">
17                 <h1>
18                     <small>新增书籍</small>
19                 </h1>
20             </div>
21         </div>
22     </div>

```

```

23     <form action="${pageContext.request.contextPath}/book/addBook"
method="post">
24         <div class="form-group">
25             <label>书籍名称: </label>
26             <input type="text" class="form-control" name="bookName"
required>
27         </div>
28         <div class="form-group">
29             <label>书籍数量: </label>
30             <input type="text" class="form-control" name="bookCounts"
required>
31         </div>
32         <div class="form-group">
33             <label>书籍详情: </label>
34             <input type="text" class="form-control" name="detail"
required>
35         </div>
36         <div class="form-group">
37             <input type="submit" class="form-control" value="添加">
38         </div>
39     </form>
40
41 </div>

```

6. BookController 类编写，方法三：添加书籍

```

1  @RequestMapping("/addBook")
2  public String addBook(Books books){
3
4      bookService.addBook(books);
5
6      //添加完成后重定向到查询全部的请求
7      return "redirect:/book/allBook";
8  }

```

7. BookController 类编写，方法四：删除书籍

```

1  @RequestMapping("/del/{id}")
2  public String del(@PathVariable("id") int id){
3
4      bookService.deleteBookById(id);
5
6      return "redirect:/book/allBook";
7  }

```

8. BookController 类编写，方法五：查询一个书籍，跳转修改页面

```

1 @RequestMapping("/toUpdateBook")
2 public String toUpdateBook(int id,Model model){
3
4     Books books = bookService.findById(id);
5
6     model.addAttribute("book",books);
7
8     return "updateBook";
9 }

```

9. 修改书籍页面 updateBook.jsp

```

1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
3 <html>
4 <head>
5     <title>修改信息</title>
6     <meta name="viewport" content="width=device-width, initial-
7 scale=1.0">
8     <!-- 引入 Bootstrap -->
9     <link
10 href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css"
11 rel="stylesheet">
12 </head>
13 <body>
14 <div class="container">
15
16     <div class="row clearfix">
17         <div class="col-md-12 column">
18             <div class="page-header">
19                 <h1>
20                     <small>修改书籍</small>
21                 </h1>
22             </div>
23         </div>
24     </div>
25
26     <form action="${pageContext.request.contextPath}/book/updateBook"
27 method="post">
28         <input type="hidden" name="bookID"
29 value="${book.getBookID()}" />
30         <div class="form-group">
31             <label>书籍名称: </label>
32             <input type="text" value="${book.getBookName()}"
33 class="form-control" name="bookName" required>
34         </div>
35         <div class="form-group">
36             <label>书籍数量: </label>
37             <input type="text" value="${book.getBookCounts()}"
38 class="form-control" name="bookCounts" required>
39         </div>
40         <div class="form-group">
41             <label>书籍详情: </label>
42             <input type="text" value="${book.getDetail() }"
43 class="form-control" name="detail" required>
44         </div>

```

```

37         <div class="form-group">
38             <input type="submit" class="form-control" value="修改">
39         </div>
40     </form>
41
42 </div>

```

10. BookController 类编写，方法六：修改书籍

```

1  @RequestMapping("/updateBook")
2  public String updateBook(Books books){
3
4      bookService.updateBook(books);
5
6      return "redirect:/book/allBook";
7  }

```

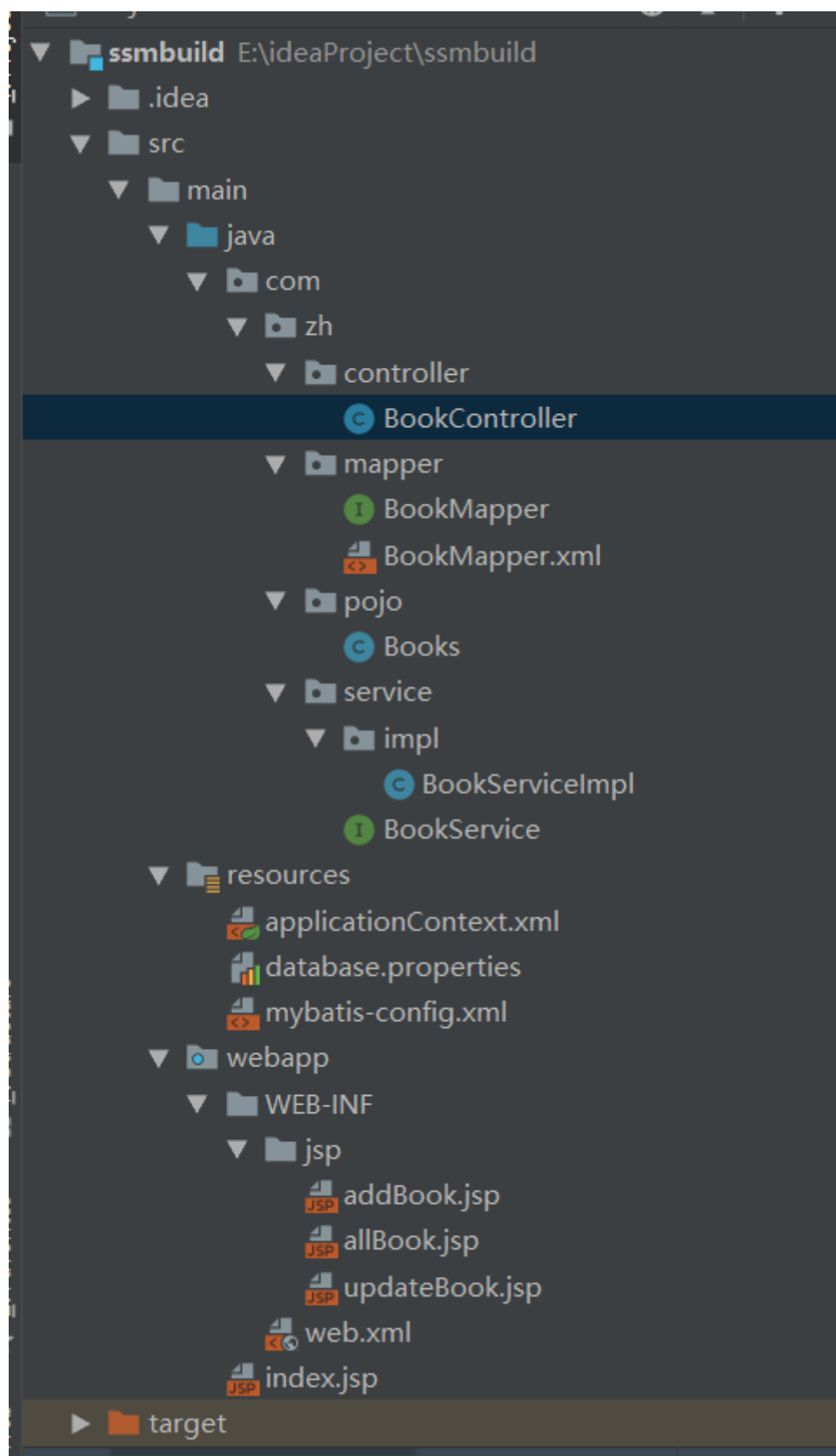
11. BookController 类编写，方法七：查询书籍

```

1  @RequestMapping("/findByName")
2  public String findByName(String bookName, Model model){
3
4      List<Books> list = bookService.findByName(bookName);
5
6      System.out.println("list=====" + list);
7
8      if (list.size() == 0){
9
10         System.out.println("list空");
11
12         list = bookService.findAll();
13
14         model.addAttribute("msg", "没有此书籍");
15     }
16
17     model.addAttribute("list", list);
18
19     return "allBook";
20 }

```

项目结构



页面展示

index.jsp

点击进入列表页

allBook.jsp

书籍列表 —— 显示所有书籍

新增显示全部

请输入要查询的书名查询

书籍编号	书籍名字	书籍数量	书籍详情	操作
2	MySQL	10	从删库到跑咯	更改 删除
3	linux	3	从入门到进牢	更改 删除
6	Java	10	从入门到放弃	更改 删除

addBook.jsp

新增书籍

书籍名称:

书籍数量:

书籍详情:

添加

updateBook.jsp

修改书籍

书籍名称:

MySQL

书籍数量:

10

书籍详情:

从删库到跑咯

修改

