1、环境

- IDEA
- MySql 5.5.29
- Tomcat 8.5.28
- Maven 3.5.4

2、数据库环境

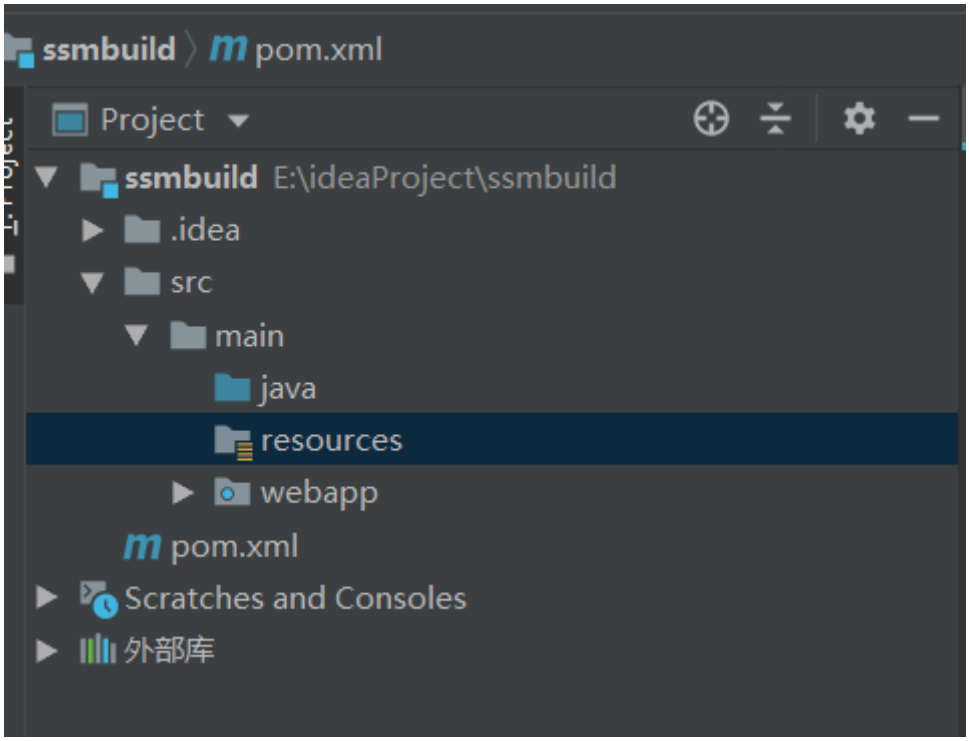
创建一个存放书籍的数据库表

```sql
create database ssmbuild;

use ssmbuild;

drop table if exists books;

create table books(
bookID int(10) primary key auto_increment comment '书id',
bookName varchar(100) not null comment '书名',
bookCounts int(11) not null comment '数量',
detail varchar(200) not null comment '描述'
)engine=innodb default charset=utf8;

insert into books(bookID,bookName,bookCounts,detail) values
(1,'Java',10,'从入门到放弃'),
(2,'MySQL',5,'从删库到跑咯'),
(3,'linux',3,'从入门到进牢');

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!

```xml
<!--依赖：junit，数据库驱动，连接池，servlet，jsp，mybatis，mybatis-spring,spring-->
<dependencies>
  <!--Junit-->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
  </dependency>
  <!--数据库驱动-->
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.47</version>
  </dependency>
  <!-- 数据库连接池 c3p0 -->
  <dependency>
    <groupId>com.mchange</groupId>
    <artifactId>c3p0</artifactId>
    <version>0.9.5.2</version>
  </dependency>

  <!--Servlet - JSP -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.2</version>
  </dependency>
  <dependency>
```

```
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>

  <!--Mybatis-->
  <dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.2</version>
  </dependency>
  <dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.0.2</version>
  </dependency>

  <!--Spring-->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.1.9.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.1.9.RELEASE</version>
  </dependency>
  <!--lombok-->
  <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.8</version>
  </dependency>
  <!--aop织入-->
  <dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.9.4</version>
  </dependency>
</dependencies>
```

3. Maven资源过滤设置

```
<build>
    <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
        <resource>
            <directory>src/main/resources</directory>
            <includes>
```

```
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
    </resources>
</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

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

</configuration>
```

   - applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-
beans.xsd">

</beans>
```

4、Mybatis层

1. 数据库配置文件 **database.properties**

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/ssmbuild?
useSSL=true&useUnicode=true&characterEncoding=utf8
jdbc.username=root
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

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

```xml
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>

    <!--配置数据源，交给spring-->

    <settings>
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>

    <typeAliases>
        <package name="com.zh.pojo"/>
    </typeAliases>

    <mappers>
        <mapper class="com.zh.mapper.BookMapper"/>
    </mappers>

</configuration>
```

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

```java
package com.zh.pojo;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Books {

    private int bookID;
    private String bookName;
    private int bookCounts;
    private String detail;

}
```

5. 编写Dao层的 Mapper接口!

```java
package com.zh.mapper;

import com.zh.pojo.Books;
import org.apache.ibatis.annotations.Param;

import java.util.List;

public interface BookMapper {

    //增加一本书
    int addBook(Books books);

```

```
    //删除
    int deleteBookById(@Param("id") int id);

    //修改
    int updateBook(Books books);

    //查询
    Books findById(@Param("id") int id);

    List<Books> findAll();

}
```

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

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.zh.mapper.BookMapper">

    <insert id="addBook" parameterType="books">
        insert into ssmbuild.books (bookName, bookCounts, detail)
        values (#{bookName}, #{bookCounts}, #{detail});
    </insert>

    <delete id="deleteBookById" parameterType="int">
        delete from ssmbuild.books where bookID = #{id}
    </delete>

    <update id="updateBook" parameterType="books">
        update ssmbuild.books set bookName = #{bookName},
        bookCounts = #{bookCounts}, detail = #{detail}
        where bookID = #{bookID};
    </update>

    <select id="findById" parameterType="int" resultType="books">
        select * from ssmbuild.books where bookID = #{id};
    </select>

    <select id="findAll" resultType="books">
        select * from ssmbuild.books;
    </select>
</mapper>
```

7. 实现类service接口

```
package com.zh.service;

import com.zh.pojo.Books;
import org.apache.ibatis.annotations.Param;

import java.util.List;

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() {
```

```
        return bookMapper.findAll();
    }
}
```

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

```
<!--Spring整合mapper层-->
<!--1.关联数据库配置文件-->
<context:property-placeholder
location="classpath:database.properties"/>

<!--2.c3p0连接池-->
<bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="${jdbc.driver}"/>
    <property name="jdbcUrl" value="${jdbc.url}"/>
    <property name="user" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>

    <!-- c3p0连接池的私有属性 -->
    <property name="maxPoolSize" value="30"/>
    <property name="minPoolSize" value="10"/>
    <!-- 关闭连接后不自动commit -->
    <property name="autoCommitOnClose" value="false"/>
    <!-- 获取连接超时时间 -->
    <property name="checkoutTimeout" value="10000"/>
    <!-- 当获取连接失败重试次数 -->
    <property name="acquireRetryAttempts" value="2"/>
</bean>

<!--3.sqlSessionFactory-->
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <!--绑定mybatis配置文件-->
    <property name="configLocation" value="classpath:mybatis-
config.xml"/>
</bean>

<!--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-->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <!--注入sqlSessionFactory-->
    <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory"/>
    <!--要扫描的mapper包-->
    <property name="basePackage" value="com.zh.mapper"/>
</bean>
```

3. **Spring整合service层**

```xml
<!--spring整合service层-->
<!--1.扫描service下的包-->
<context:component-scan base-package="com.zh.service"/>

<!--2.将所有业务注入到spring中，可以通过配置和注解-->
<bean id="bookService" class="com.zh.service.impl.BookServiceImpl">
    <property name="bookMapper" ref="bookMapper"/>
</bean>

<!--3.声明式事务配置-->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager
">
    <!--注入数据源-->
    <property name="dataSource" ref="dataSource"/>
</bean>
<!--4.aop事务支持-->
<!--结合aop实现事务的织入-->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*" propagation="REQUIRED"/>
    </tx:attributes>
</tx:advice>
<!--配置事务切入-->
<aop:config>
    <aop:pointcut id="txPointcut" expression="execution(*
com.zh.mapper.*.*(..))"/>
    <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut"/>
</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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

  <display-name>Archetype Created Web Application</display-name>

  <!--DispatcherServlet-->
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:applicationContext.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
```

```
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <!--乱码过滤-->
  <filter>
    <filter-name>encodingFilert</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>utf-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>encodingFilert</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!--session-->
  <session-config>
    <session-timeout>15</session-timeout>
  </session-config>
</web-app>
```

2. **applicationContext.xml**

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

10、Controller层 和 视图层编写

1. 编写首页 **index.jsp**。进入查询全部

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

```
        a {
            text-decoration: none;
            color: black;
            font-size: 18px;
        }
        h3 {
            width: 180px;
            height: 38px;
            margin: 100px auto;
            text-align: center;
            line-height: 38px;
            background: deepskyblue;
            border-radius: 4px;
        }
    </style>
</head>
<body>

<h3>
    <a href="${pageContext.request.contextPath}/book/allBook">点击进入列表页</a>
</h3>
</body>
</html>
```

2. BookController 类编写， 方法一：查询全部书籍

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

    //controller 调 service层
    @Autowired
    @Qualifier("bookService")
    private BookService bookService;

    @RequestMapping("/allBook")
    public String allBook(Model model){

        List<Books> list = bookService.findAll();

        model.addAttribute("list",list);

        return "allBook";
    }
}
```

3. 书籍列表页面 **allbook.jsp**

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

```
    <!-- 引入 Bootstrap -->
    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

<div class="container">

    <div class="row clearfix">
        <div class="col-md-12 column">
            <div class="page-header">
                <h1>
                    <small>书籍列表 -- 显示所有书籍</small>
                </h1>
            </div>
        </div>
    </div>

    <div class="row">
        <div class="col-md-4 column">
            <a class="btn btn-primary" href="${pageContext.request.contextPath}/book/toAddBook">新增</a>
            <a class="btn btn-primary" href="${pageContext.request.contextPath}/book/allBook">显示全部</a>
        </div>
        <div class="col-md-8 column">
            <%--查询书籍--%>
            <form action="${pageContext.request.contextPath}/book/findByName" style="float:right" class="form-inline" method=post>
                <span style="color: red;font-weight: bold" >${msg}</span>
                <input type="text" name="bookName" class="form-control" placeholder="请输入要查询的书名" >
                <input type="submit" value="查询" class="btn btn-primary">
            </form>
        </div>
    </div>

    <div class="row clearfix">
        <div class="col-md-12 column">
            <table class="table table-hover table-striped">
                <thead>
                <tr>
                    <th>书籍编号</th>
                    <th>书籍名字</th>
                    <th>书籍数量</th>
                    <th>书籍详情</th>
                    <th>操作</th>
                </tr>
                </thead>

                <tbody>
                <c:forEach var="book" items="${requestScope.get('list')}">
                    <tr>
```

```
                        <td>${book.getBookID()}</td>
                        <td>${book.getBookName()}</td>
                        <td>${book.getBookCounts()}</td>
                        <td>${book.getDetail()}</td>
                        <td>
                            <a href="${pageContext.request.contextPath}/book/toUpdateBook?id=${book.getBookID()}">更改</a> |
                            <a href="${pageContext.request.contextPath}/book/del/${book.getBookID()}">删除</a>
                        </td>
                    </tr>
                </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>
```

4. BookController 类编写， 方法二：去添加书籍页面

```
@RequestMapping("/toAddBook")
public String toAddBook(){

    return "addBook";
}
```

5. 添加书籍页面：**addBook.jsp**

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>
<head>
    <title>新增书籍</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 引入 Bootstrap -->
    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container">

    <div class="row clearfix">
        <div class="col-md-12 column">
            <div class="page-header">
                <h1>
                    <small>新增书籍</small>
                </h1>
            </div>
        </div>
    </div>
```

```
    <form action="${pageContext.request.contextPath}/book/addBook" method="post">
        <div class="form-group">
            <label>书籍名称：</label>
            <input type="text" class="form-control" name="bookName" required>
        </div>
        <div class="form-group">
            <label>书籍数量：</label>
            <input type="text" class="form-control" name="bookCounts" required>
        </div>
        <div class="form-group">
            <label>书籍详情：</label>
            <input type="text" class="form-control" name="detail" required>
        </div>
        <div class="form-group">
            <input type="submit" class="form-control" value="添加">
        </div>
    </form>

</div>
```

6. BookController 类编写， 方法三：添加书籍

```
@RequestMapping("/addBook")
public String addBook(Books books){

    bookService.addBook(books);

    //添加完成后重定向到查询全部的请求
    return "redirect:/book/allBook";
}
```

7. BookController 类编写， 方法四：删除书籍

```
@RequestMapping("/del/{id}")
public String del(@PathVariable("id") int id){

    bookService.deleteBookById(id);

    return "redirect:/book/allBook";
}
```

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

```java
@RequestMapping("/toUpdateBook")
public String toUpdateBook(int id,Model model){

    Books books = bookService.findById(id);

    model.addAttribute("book",books);

    return "updateBook";
}
```

9. 修改书籍页面 **updateBook.jsp**

```jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>修改信息</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 引入 Bootstrap -->
    <link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container">

    <div class="row clearfix">
        <div class="col-md-12 column">
            <div class="page-header">
                <h1>
                    <small>修改书籍</small>
                </h1>
            </div>
        </div>
    </div>

    <form action="${pageContext.request.contextPath}/book/updateBook" method="post">
        <input type="hidden" name="bookID" value="${book.getBookID()}"/>
        <div class="form-group">
            <label>书籍名称：</label>
            <input type="text" value="${book.getBookName()}" class="form-control" name="bookName" required>
        </div>
        <div class="form-group">
            <label>书籍数量：</label>
            <input type="text" value="${book.getBookCounts()}" class="form-control" name="bookCounts" required>
        </div>
        <div class="form-group">
            <label>书籍详情：</label>
            <input type="text" value="${book.getDetail() }" class="form-control" name="detail" required>
        </div>
```

```
        <div class="form-group">
            <input type="submit" class="form-control" value="修改">
        </div>
    </form>

</div>
```

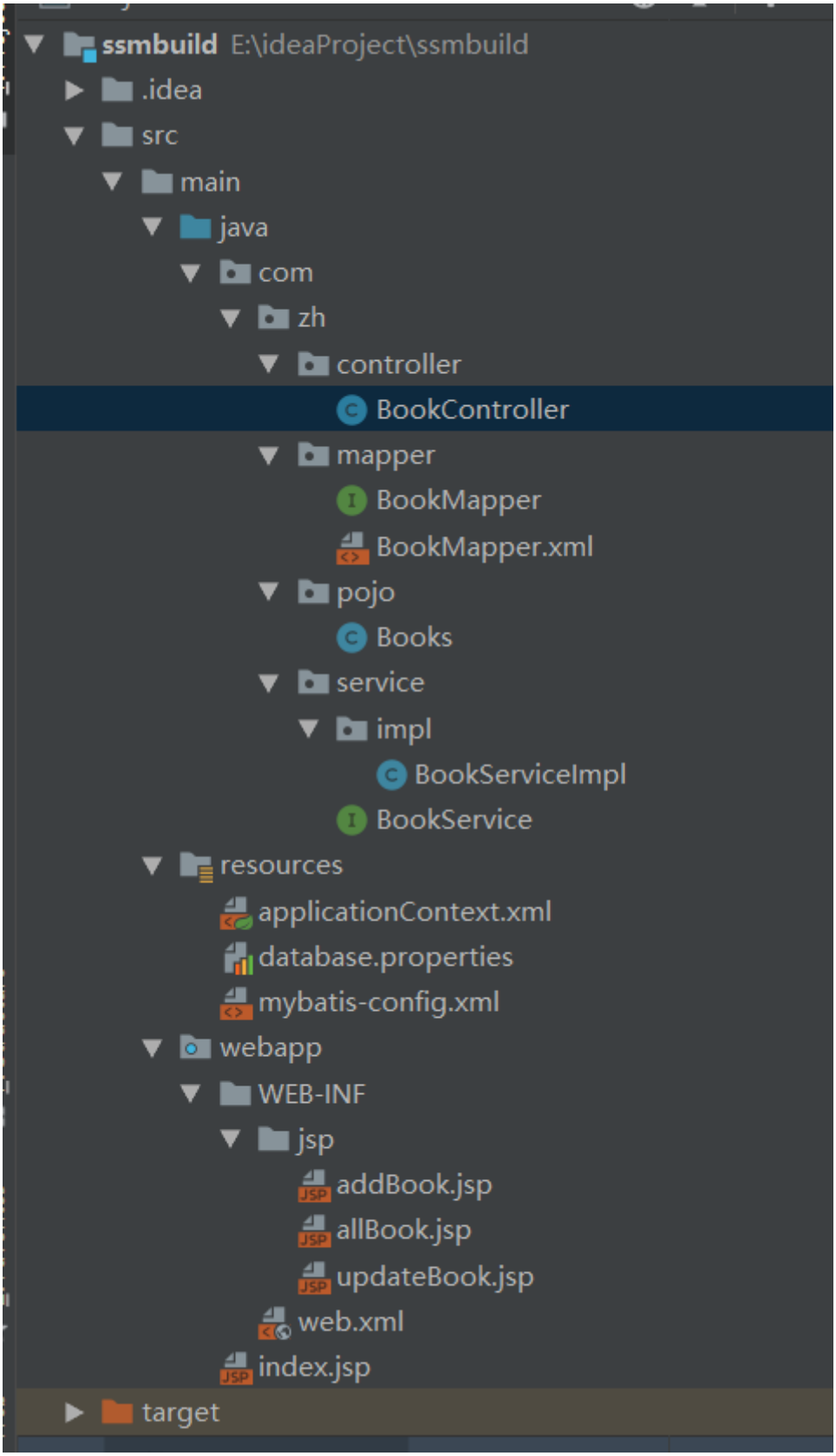
10. BookController 类编写， 方法六：修改书籍

```
@RequestMapping("/updateBook")
public String updateBook(Books books){

    bookService.updateBook(books);

    return "redirect:/book/allBook";
}
```

11. BookController 类编写， 方法七：查询书籍

```
@RequestMapping("/findByName")
public String findByName(String bookName,Model model){

    List<Books> list = bookService.findByName(bookName);

    System.out.println("list=================="+list);

    if (list.size() == 0){

        System.out.println("list空");

        list = bookService.findAll();

        model.addAttribute("msg","没有此书籍");
    }

    model.addAttribute("list",list);

    return "allBook";
}
```

项目结构



页面展示

index.jsp

点击进入列表页

allBook.jsp

addBook.jsp

updateBook.jsp