

# SSM整合

首先我们需要明确的一点是：我们要用 `Spring` 来整合其他的两个框架

## 环境要求

- `IDEA`
- `MySQL5.7`
- `Tomcat`
- `Maven`

## 数据库环境

- `ssmbuild`：一个存放书籍数据的数据库

```
1  CREATE DATABASE `ssmbuild`;  
2  
3  USE `ssmbuild`;  
4  
5  DROP TABLE IF EXISTS `books`;  
6  
7  CREATE TABLE `books` (  
8      `bookID` INT(10) NOT NULL 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     KEY `bookID` (`bookID`)  
13 ) ENGINE=INNODB DEFAULT CHARSET=utf8  
14  
15 INSERT INTO `books` (`bookID`, `bookName`, `bookCounts`, `detail`) VALUES  
16 (1, 'Java', 1, '从入门到放弃'),  
17 (2, 'MySQL', 10, '从删库到跑路'),  
18 (3, 'Linux', 5, '从入门到进牢');
```

一条一条执行，否则可能会有错误

## 基本环境搭建

1. 新建一个 `maven` 项目 `ssmbuild`，添加 `web` 支持
2. 导入相关的 `pom` 依赖
3. `maven` 资源过滤设置

#### 4. 建立基本结构和框架配置

- `com.bean.pojo`
- `com.bean.dao`
- `com.bean.service`
- `com.bean.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-beans.xsd">
6
7 </beans>
```

---

## Mybatis 层的编写

### 编写数据库配置文件

- `database.properties`

```
1 jdbc.driver=com.mysql.jdbc.Driver
2 # 假如使用的是MySQL 8.0以上, 需要再加一个时区的配置: &serverTimezone=Asia/Shanghai
3 jdbc.url=jdbc:mysql://localhost:3306/ssmbuild?
4   useSSL=true&useUnicode=true&characterEncoding=utf8
5 jdbc.username=root
6 jdbc.password=root
```

### IDEA 关联数据库

### 编写Mybatis 的核心配置文件

- `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          <!--数据源这里不需要在使用mybatis去配置了，Spring会搞定-->
7
8          <!--取别名-->
9          <typeAliases>
10             <package name="com.bean.pojo"></package>
11          </typeAliases>
12
13          <!--配置映射，找到各级目录下的Mapper-->
14          <mappers>
15             <package name="com.bean.dao"></package>
16          </mappers>
17      </configuration>

```

## 编写数据库对应类 `com.bean.pojo.Books`

- 在 `maven` 中添加使用 `lombok` 插件（可以自动补全构造函数，`getter` 和 `setter`，`toString`，`hashCode` 等）

```

1      <dependency>
2          <groupId>org.projectlombok</groupId>
3          <artifactId>lombok</artifactId>
4          <version>1.16.10</version>
5      </dependency>

```

- `Books`

```

1      package com.bean.pojo;
2
3
4      import lombok.AllArgsConstructor;
5      import lombok.Data;
6      import lombok.NoArgsConstructor;
7
8      import java.io.Serializable;
9
10
11     @Data
12     @AllArgsConstructor
13     @NoArgsConstructor
14     public class Books implements Serializable {
15         //注意写Books，因为java中有一个叫做Book的库
16
17         private Integer bookID;
18         private String bookName;
19         private int bookCounts;
20         private String detail;
21
22     }

```

## 编写 `Dao` 层的 `Mapper` 接口

```

1  package com.bean.dao;
2
3  import com.bean.pojo.Books;
4
5  import java.util.List;
6
7
8  public interface BookMapper {
9
10     //增加一本书
11     int addBook(Books books);
12
13     //删除一本书
14     int deleteBookById(int id);
15
16     //更新一本书
17     int updateBook(Books books);
18
19     //查询一本书
20     Books queryBookById(int id);
21
22     //查询全部书
23     List<Books> queryAllBook();
24 }

```

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 Config 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5  <mapper namespace="com.bean.dao.BookMapper">
6
7      <insert id="addBook" parameterType="Books">
8          insert into ssmbuild.books(bookName, bookCounts, detail)
9          values (#{bookName},#{bookCounts},#{detail});
10     </insert>
11
12     <delete id="deleteBookById" parameterType="Integer">
13         delete from ssmbuild.books where bookID = #{bookID}
14     </delete>
15
16     <update id="updateBook" parameterType="Books">
17         update ssmbuild.books set bookName=#{bookName},bookCounts=#{bookCounts},detail=#{
18 detail}
19     </update>
20
21     <select id="queryBookById" parameterType="Integer" resultType="Books">
22         select * from ssmbuild.books where bookID=#{bookID};
23     </select>
24
25     <select id="queryAllBook" resultType="Books">
26         select * from ssmbuild.books;
27     </select>
28 </mapper>

```

## 编写 Service 层的接口和实现类

```
1 package com.bean.service;
2
3 import com.bean.pojo.Books;
4
5 import java.util.List;
6
7 public interface IBookService {
8
9     //增加一个Book
10    int addBook(Books books);
11
12    //根据id删除一个Book
13    int deleteBookById(int id);
14
15    //更新Book
16    int updateBook(Books books);
17
18    //根据id查询Book
19    Books queryBookById(int id);
20
21    //查询所有Book
22    List<Books> queryAllBook();
23 }
```

```
1 package com.bean.service.impl;
2
3 import com.bean.dao.BookMapper;
4 import com.bean.pojo.Books;
5 import com.bean.service.IBookService;
6
7 import java.util.List;
8
9 public class BookServiceImpl implements IBookService {
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
14    public void setBookMapper(BookMapper bookMapper) {
15        this.bookMapper = bookMapper;
16    }
17
18    @Override
19    public int addBook(Books books) {
20        return bookMapper.addBook(books);
21    }
22
23    @Override
24    public int deleteBookById(int id) {
25        return bookMapper.deleteBookById(id);
26    }
27
28    @Override
29    public int updateBook(Books books) {
```

```

30         return bookMapper.updateBook(books);
31     }
32
33     @Override
34     public Books queryBookById(int id) {
35         return bookMapper.queryBookById(id);
36     }
37
38     @Override
39     public List<Books> queryAllBook() {
40         return bookMapper.queryAllBook();
41     }
42 }

```

## Spring 层的编写

### 配置 Spring 整合 Mybatis，数据源使用 c3p0 连接池

- spring-dao.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      xmlns:context="http://www.springframework.org/schema/context"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6          http://www.springframework.org/schema/beans/spring-beans.xsd
7          http://www.springframework.org/schema/context
8          https://www.springframework.org/schema/context/spring-context.xsd">
9      <!--整合Spring和Mybatis的配制文件-->
10
11      <!--1. 关联数据库文件-->
12      <context:property-placeholder location="classpath:database.properties"/>
13
14      <!--2. 数据库连接池-->
15      <!--数据库连接池: c3p0, c3p0的好处就是自动加载配置文件并设置到对象里面-->
16      <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
17          <property name="driverClass" value="${jdbc.driver}"/>
18          <property name="jdbcUrl" value="${jdbc.url}"/>
19          <property name="user" value="${jdbc.username}"/>
20          <property name="password" value="${jdbc.password}"/>
21          <!-- c3p0连接池的私有属性 -->
22          <property name="maxPoolSize" value="30"/>
23          <property name="minPoolSize" value="10"/>
24          <!-- 关闭连接后不自动commit -->
25          <property name="autoCommitOnClose" value="false"/>
26          <!-- 获取连接超时时间 -->
27          <property name="checkoutTimeout" value="10000"/>
28          <!-- 当获取连接失败重试次数 -->
29          <property name="acquireRetryAttempts" value="2"/>
30      </bean>
31
32      <!--3. 配置SqlSessionFactory对象-->

```

```

33     <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
34         <!--注入数据库连接池-->
35         <property name="dataSource" ref="dataSource"/>
36         <!--配置Mybatis的全局文件: mybatis-config.xml-->
37         <property name="configLocation" value="classpath:mybatis-config.xml"/>
38     </bean>
39
40
41
42     <!--4. 配置扫描Dao接口包, 动态实现Dao接口注入到Spring容器中-->
43     <!--解释: https://www.cnblogs.com/jpfss/p/7799806.html-->
44     <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
45         <!-- 注入sqlSessionFactory -->
46         <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
47         <!-- 给出需要扫描Dao接口包 -->
48         <property name="basePackage" value="com.bean.dao"/>
49     </bean>
50
51 </beans>

```

## spring整合 service 层

- spring-service.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      xmlns:context="http://www.springframework.org/schema/context"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6          http://www.springframework.org/schema/beans/spring-beans.xsd
7          http://www.springframework.org/schema/context
8          http://www.springframework.org/schema/context/spring-context.xsd">
9
10     <!--1. 扫描service下面的包-->
11     <context:component-scan base-package="com.bean.service"/>
12
13     <!--2. 所有的业务类放到Spring, 可以通过注解或者配置-->
14     <bean id="bookServiceImpl" class="com.bean.service.impl.BookServiceImpl">
15         <!--注意这里可能会报错, 原因是这里的和dao配置没有关联起来,
16             关联方式有两种:
17             1. <import resource="classpath:spring-dao.xml"/>
18             2. 通过idea自动关联, 就是当此页面的最上面出现黄色条幅的时候直接点击`Configure
19             application context`, 然后加入到一起
20             (去File->Project Structure->Module->Spring->ApplicationContext 里面查看是否
21             关连到了一起)
22             还爆红重启
23             -->
24         <property name="bookMapper" ref="bookMapper"/>
25     </bean>
26
27     <!--3. 声明式事务配置-->
28     <bean id="transactionManager"
29         class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
30         <property name="dataSource" ref="dataSource"/>
31     </bean>

```

```
31      <!--AOP暂时先不写，因为AOP的包没导-->
32
33  </beans>
```

---

## SpringMVC 层的编写

---

### 1. web.xml

```
1  <!DOCTYPE web-app PUBLIC
2  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
3  "http://java.sun.com/dtd/web-app_2_3.dtd" >
4
5  <web-app>
6      <display-name>Archetype Created Web Application</display-name>
7
8      <servlet>
9          <servlet-name>dispatcherServlet</servlet-name>
10         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
11         <init-param>
12             <param-name>contextConfigLocation</param-name>
13             <param-value>classpath:spring-mvc.xml</param-value>
14         </init-param>
15         <load-on-startup>1</load-on-startup>
16     </servlet>
17     <servlet-mapping>
18         <servlet-name>dispatcherServlet</servlet-name>
19         <url-pattern>/</url-pattern>
20     </servlet-mapping>
21
22     <!--乱码过滤-->
23     <filter>
24         <filter-name>encodingFilter</filter-name>
25         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
26         <init-param>
27             <param-name>encoding</param-name>
28             <param-value>utf-8</param-value>
29         </init-param>
30     </filter>
31     <filter-mapping>
32         <filter-name>encodingFilter</filter-name>
33         <url-pattern>/*</url-pattern>
34     </filter-mapping>
35
36     <session-config>
37         <session-timeout>15</session-timeout>
38     </session-config>
39 </web-app>
```

### 1. spring-mvc.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      xmlns:mvc="http://www.springframework.org/schema/mvc"
5      xmlns:context="http://www.springframework.org/schema/context"
6      xsi:schemaLocation="http://www.springframework.org/schema/beans
7          http://www.springframework.org/schema/beans/spring-beans.xsd
8          http://www.springframework.org/schema/cache
9          http://www.springframework.org/schema/cache/spring-cache.xsd
10         http://www.springframework.org/schema/context
11         https://www.springframework.org/schema/context/spring-context.xsd">
12
13      <!--1. 注解驱动-->
14      <mvc:annotation-driven/>
15
16      <!--2. 静态资源过滤-->
17      <mvc:default-servlet-handler/>
18
19      <!--3. 扫描包-->
20      <context:component-scan base-package="com.bean.controller"/>
21
22      <!--4. 视图解析器-->
23      <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
24          <property name="prefix" value="/WEB-INF/jsp/" />
25          <property name="suffix" value=".jsp" />
26      </bean>
27
28 </beans>

```

1. Spring 配置整合文件， applicationContext.xml

配置文件结束

```

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-beans.xsd">
6
7      <import resource="spring-dao.xml" />
8      <import resource="spring-service.xml" />
9      <import resource="spring-mvc.xml" />
10
11 </beans>

```

整合到此结束，剩下的就是测试了