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

```sql
CREATE DATABASE `ssmbuild`;

USE `ssmbuild`;

DROP TABLE IF EXISTS `books`;

CREATE TABLE `books` (
  `bookID` INT(10) NOT NULL 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',
  KEY `bookID` (`bookID`)
) ENGINE=INNODB DEFAULT CHARSET=utf8

INSERT  INTO `books`(`bookID`,`bookName`,`bookCounts`,`detail`)VALUES
(1,'Java',1,'从入门到放弃'),
(2,'MySQL',10,'从删库到跑路'),
(3,'Linux',5,'从进门到进牢');
```

> 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`

      ```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
      <?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>
      ```

`Mybatis` 层的编写

编写数据库配置文件

- `database.properties`

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

`IDEA` 关联数据库

`编写Mybatis` 的核心配置文件

- `mybatis-config.xml`

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

```xml
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!--数据源这里不需要在使用mybatis去配置了，Spring会搞定-->

    <!--取别名-->
    <typeAliases>
        <package name="com.bean.pojo"></package>
    </typeAliases>

    <!--配置映射，找到各级目录下的Mapper-->
    <mappers>
        <package name="com.bean.dao"></package>
    </mappers>
</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）

```xml
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.16.10</version>
    </dependency>
```

- `Books`

```java
package com.bean.pojo;


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

import java.io.Serializable;


@Data
@AllArgsConstructor
@NoArgsConstructor
public class Books implements Serializable {
    //注意写Books，因为java中有一个叫做Book的库

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

}
```

编写 `Dao` 层的 `Mapper` 接口

```java
package com.bean.dao;

import com.bean.pojo.Books;

import java.util.List;


public interface BookMapper {

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

    //删除一本书
    int deleteBookById(int id);

    //更新一本书
    int updateBook(Books books);

    //查询一本书
    Books queryBookById(int id);

    //查询全部书
    List<Books> queryAllBook();
}
```

编写接口对应的 `Mapper.xml` 文件，需要导入 `Mybatis` 的包

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

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

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

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

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

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

编写Service层的接口和实现类

```java
package com.bean.service;

import com.bean.pojo.Books;

import java.util.List;

public interface IBookService {

    //增加一个Book
    int addBook(Books books);

    //根据id删除一个Book
    int deleteBookById(int id);

    //更新Book
    int updateBook(Books books);

    //根据id查询Book
    Books queryBookById(int id);

    //查询所有Book
    List<Books> queryAllBook();
}
```

```java
package com.bean.service.impl;

import com.bean.dao.BookMapper;
import com.bean.pojo.Books;
import com.bean.service.IBookService;

import java.util.List;

public class BookServiceImpl implements IBookService {

    //调用dao层的操作，设置一个set接口方便Spring管理
    private BookMapper bookMapper;

    public void setBookMapper(BookMapper bookMapper) {
        this.bookMapper = bookMapper;
    }

    @Override
    public int addBook(Books books) {
        return bookMapper.addBook(books);
    }

    @Override
    public int deleteBookById(int id) {
        return bookMapper.deleteBookById(id);
    }

    @Override
    public int updateBook(Books books) {
        return bookMapper.updateBook(books);
    }

    @Override
    public Books queryBookById(int id) {
        return bookMapper.queryBookById(id);
    }

    @Override
    public List<Books> queryAllBook() {
        return bookMapper.queryAllBook();
    }
}
```

Spring 层的编写

配置 Spring 整合 Mybatis，数据源使用 c3p0 连接池

- spring-dao.xml

```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"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/context
            https://www.springframework.org/schema/context/spring-context.xsd">
    <!--整合Spring和Mybatis的配制文件-->

    <!--1. 关联数据库文件-->
    <context:property-placeholder location="classpath:database.properties"/>

    <!--2. 数据库连接池-->
        <!--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-->
        <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. 配置SqlSessionFactory对象-->
```

```xml
        <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
            <!--注入数据库连接池-->
            <property name="dataSource" ref="dataSource"/>
            <!--配置Mybatis的全局文件：mybatis-config.xml-->
            <property name="configLocation" value="classpath:mybatis-config.xml"/>
        </bean>



    <!--4. 配置扫描Dao接口包，动态实现Dao接口注入到Spring容器中-->
        <!--解释 ： https://www.cnblogs.com/jpfss/p/7799806.html-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 注入sqlSessionFactory -->
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
        <!-- 给出需要扫描Dao接口包 -->
        <property name="basePackage" value="com.bean.dao"/>
    </bean>

</beans>
```

`spring` 整合 `service` 层

- `spring-service.xml`

```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"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="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.xsd">

    <!--1. 扫描service下面的包-->
    <context:component-scan base-package="com.bean.service"/>

    <!--2. 将所有的业务类放到Spring，可以通过注解或者配置-->
    <bean id="bookServiceImpl" class="com.bean.service.impl.BookServiceImpl">
        <!--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，
            关联方式有两种：
                1. <import resource="classpath:spring-dao.xml"/>
                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
application context`，然后加入到一起
                    (去File->Project Stucture->Module->Spring->ApplicationContext 里面查看是否
关连到了一起)
            还爆红重启
        -->
        <property name="bookMapper" ref="bookMapper"/>
    </bean>


    <!--3. 声明式事务配置-->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!--AOP暂时先不写，因为AOP的包没导-->

</beans>
```

SpringMVC层的编写

1. web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

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

  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring-mvc.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <!--乱码过滤-->
  <filter>
    <filter-name>encodingFilter</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>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

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

1. spring-mvc.xml

```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"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/cache
        http://www.springframework.org/schema/cache/spring-cache.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd">

    <!--1. 注解驱动-->
    <mvc:annotation-driven/>

    <!--2. 静态资源过滤-->
    <mvc:default-servlet-handler/>
    <!--3. 扫描包-->
    <context:component-scan base-package="com.bean.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>

</beans>
```

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

> 配置文件结束

```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">

    <import resource="spring-dao.xml"/>
    <import resource="spring-service.xml"/>
    <import resource="spring-mvc.xml"/>

</beans>
```

> 整合到此结束，剩下的就是测试了