

SSM大整合及常见增删改查操作

一、整合关键点

Spring：负责对象的创建、维护、管理及对象依赖资源的注入

SpringMVC：负责请求的处理相当于(Servlet)

MyBatis：负责与数据库进行交互

二、基本步骤

1、pom.xml添加依赖

mybatis|spring-webmvc|mybatis-spring|spring-jdbc|mysql|bonecp数据源|log4j日志|jstl依赖

2、编写配置文件

spring-config.xml|springmvc-config.xml|mybatis-config.xml|log4j.properties

A、spring-config.xml

数据源|sqlSessionFactory|映射器|事务|基于注解的组件扫描

B、springmvc-config.xml

组件扫描(controller)|注解驱动|视图解析器|文件上传等

C、mybatis-config.xml

日志记录工具|批量取别名|批量加载映射文件

D、log4j.properties

定义输出级别|日志输出位置-控制台和文件

3、定义项目结构和代码

com.ssm.entity|com.ssm.dao|com.ssm.service| com.ssm.service.impl|com.ssm.controller

4、配置web.xml

加载spring-config.xml|springmvc-config.xml|中文乱码

5、测试功能

三、代码展示

1、查询所有学生

A、导入依赖

```
1 <dependencies>
2   <!--mybatis依赖-->
3   <dependency>
4     <groupId>org.mybatis</groupId>
5     <artifactId>mybatis</artifactId>
6     <version>3.5.4</version>
7   </dependency>
8   <!--spring-web依赖-->
9   <dependency>
10    <groupId>org.springframework</groupId>
11    <artifactId>spring-webmvc</artifactId>
12    <version>5.2.4.RELEASE</version>
13  </dependency>
14  <!--mybatis-spring依赖-->
15  <dependency>
16    <groupId>org.mybatis</groupId>
17    <artifactId>mybatis-spring</artifactId>
18    <version>2.0.4</version>
19  </dependency>
20  <!--bonecp连接池配置 -->
21  <dependency>
22    <groupId>com.jolbox</groupId>
23    <artifactId>bonecp</artifactId>
24    <version>0.8.0.RELEASE</version>
25  </dependency>
26  <!--mysql驱动-->
27  <dependency>
28    <groupId>mysql</groupId>
29    <artifactId>mysql-connector-java</artifactId>
30    <version>5.1.48</version>
31  </dependency>
32  <!--spring-jdbc驱动-->
33  <dependency>
34    <groupId>org.springframework</groupId>
35    <artifactId>spring-jdbc</artifactId>
36    <version>5.2.4.RELEASE</version>
37  </dependency>
38  <!--加入日志log4j -->
39  <dependency>
40    <groupId>log4j</groupId>
41    <artifactId>log4j</artifactId>
42    <version>1.2.17</version>
43  </dependency>
44  <!--slf4j-nop依赖-->
45  <!--导入jstl依赖 -->
```

```

46     <dependency>
47         <groupId>javax.servlet</groupId>
48         <artifactId>jstl</artifactId>
49         <version>1.2</version>
50     </dependency>
51 </dependencies>

```

B、配置文件

a、spring-config.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:p="http://www.springframework.org/schema/p"
5      xmlns:c="http://www.springframework.org/schema/c"
6      xmlns:aop="http://www.springframework.org/schema/aop"
7      xmlns:context="http://www.springframework.org/schema/context"
8      xmlns:tx="http://www.springframework.org/schema/tx"
9      xsi:schemaLocation="http://www.springframework.org/schema/beans
10      http://www.springframework.org/schema/beans/spring-beans.xsd
11      http://www.springframework.org/schema/aop
12      http://www.springframework.org/schema/aop/spring-aop.xsd
13      http://www.springframework.org/schema/context
14      http://www.springframework.org/schema/context/spring-context.xsd
15      http://www.springframework.org/schema/tx
16      http://www.springframework.org/schema/tx/spring-tx.xsd">
17
18      <!--配置数据源-->
19      <bean id="dataSource" class="com.jolbox.bonecp.BoneCPDataSource"
20          p:driverClass="com.mysql.jdbc.Driver" p:jdbcUrl="jdbc:mysql://localhost:3306/ms"
21          p:username="root" p:password="admin"/>
22
23      <!--配置SqlSessionFactory对象 -->
24      <bean id="sqlFactroy" class="org.mybatis.spring.SqlSessionFactoryBean"
25          p:dataSource-ref="dataSource" p:configLocation="classpath:mybatis-config.xml"/>
26
27      <!--注入映射器-->
28      <!--basePackage指定了扫描的基准包，批量产生映射器实现类 -->
29      <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"
30          p:basePackage="com.ssm.dao"/>
31
32      <!--配置基于注解扫描的Bean 将pojo实例化到spring容器中-->
33      <context:component-scan base-package="com.ssm"/>
34
35      <!--配置事务管理器 -->
36      <bean id="txManager"
37          class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
38          p:dataSource-ref="dataSource"/>
39
40      <!--配置使用注解声明式事务处理-->
41      <tx:annotation-driven transaction-manager="txManager"/>

```

b、springmvc-config.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:p="http://www.springframework.org/schema/p"
5      xmlns:mvc="http://www.springframework.org/schema/mvc"
6      xmlns:context="http://www.springframework.org/schema/context"
7      xsi:schemaLocation="http://www.springframework.org/schema/mvc
8          http://www.springframework.org/schema/mvc/spring-mvc.xsd
9          http://www.springframework.org/schema/context
10             http://www.springframework.org/schema/context/spring-context.xsd
11             http://www.springframework.org/schema/beans
12             http://www.springframework.org/schema/beans/spring-beans.xsd">
13      <!--扫描组件 -->
14      <!--告诉Spring 到哪里去找标记为@Controller 的Controller控制器-->
15      <context:component-scan base-package="com.ssm.controller"/>
16      <!--启用注解驱动-mvc配置 -->
17      <mvc:annotation-driven/>
18      <!--配置视图解析器 -->
19      <bean id="viewResolver"
20          class="org.springframework.web.servlet.view.InternalResourceViewResolver"
21          p:prefix="/WEB-INF/jsp/" p:suffix=".jsp"/>
22      <!--添加multipartResolver的bean支持文件上传 maxUploadSize 上传文件的最大大小,单位为字节-->
23      <!-- <bean id="multipartResolver"
24          class="org.springframework.web.multipart.commons.CommonsMultipartResolver"
25          p:maxUploadSize="802400" p:defaultEncoding="utf-8"/> -->
26  </beans>

```

c、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      <!--配置日志实现 -->
7      <settings>
8          <setting name="logImpl" value="LOG4J"/>
9      </settings>
10     <!--给实体类取别名-->
11     <typeAliases>
12         <!--批量取别名 默认为实体类首字母小写 -->
13         <package name="com.ssm.entity"/>
14     </typeAliases>
15     <!--映射器-告诉MyBatis去哪里找到sql映射文件 -->
16     <mappers>
17         <!--批量加载映射文件 -->
18         <package name="com.ssm.dao"/>

```

```
19     </mappers>
20 </configuration>
```

d、log4j.properties

```
1  ### \u8bbe\u7f6ELogger\u8f93\u51fa\u7ea7\u522b\u548c\u8f93\u51fa\u76ee\u7684\u5730
   ###
2  log4j.rootLogger=debug,stdout,logfile
3
4  ### \u628a\u65e5\u5fd7\u4fe1\u606f\u8f93\u51fa\u5230\u63a7\u5236\u53f0 ###
5  log4j.appender.stdout=org.apache.log4j.ConsoleAppender
6  log4j.appender.stdout.Target=System.out
7  log4j.appender.stdout.layout=org.apache.log4j.SimpleLayout
8
9  ### \u628a\u65e5\u5fd7\u4fe1\u606f\u8f93\u51fa\u5230\u6587\u4ef6\u533b\u5bfc\u6587\u4ef6 ###
10 log4j.appender.logfile=org.apache.log4j.FileAppender
11 log4j.appender.logfile.File=jbit.log
12 log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
13 log4j.appender.logfile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}%l %F %p %m%n
```

C、代码编写

a、实体类编写

b、数据访问层

```
1 public interface StudentMapper {
2     @Select("select * from student")
3     List<Student> getStuList();
4 }
```

c、业务逻辑层

```
1 public interface StudentService { //业务接口
2
3     List<Student> getStuList();
4 }
5 @Service
6 public class StudentServiceImpl implements StudentService { //业务接口实现类
7
8     @Autowired //或者@Resource进行自动装配
9     private StudentMapper sMapper;
10
11     @Override
12     public List<Student> getStuList() {
13         return sMapper.getStuList();
14     }
15 }
```

d、控制层

```

1  @Controller
2  public class StudentController {
3
4      @Autowired
5      private StudentService stuService;
6
7      @RequestMapping("/toIndex.do")
8      public String toIndex(){
9          return "index";
10     }
11
12     @RequestMapping("/stulist.do")
13     public ModelAndView selectAllStu(){
14         List<Student> slist=stuService.getStulist();
15         if(slist!=null){
16             return new ModelAndView("index","slist",slist);
17         }
18         return null;
19     }
20 }

```

e、首页列表

```

1  <body>
2  <div align="center">
3      <table align="center" border="1" cellpadding="0" cellspacing="0" width="50%">
4          <tr>
5              <th>学号</th>
6              <th>姓名</th>
7              <th>性别</th>
8              <th>年龄</th>
9              <th>操作</th>
10         </tr>
11         <c:forEach items="${slist}" var="stu">
12             <tr class="base">
13                 <th>${stu.sno}</th>
14                 <th>${stu.sname}</th>
15                 <th>${stu.sex}</th>
16                 <th>${stu.age}</th>
17                 <th>操作</th>
18             </tr>
19         </c:forEach>
20     </table>
21 </body>

```

D、配置web.xml

```

1  <context-param>
2      <param-name>contextConfigLocation</param-name>
3      <param-value>classpath:spring-config.xml</param-value>
4  </context-param>

```

```

5   <listener>
6     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
7   </listener>
8   <servlet>
9     <servlet-name>smvc</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:springmvc-config.xml</param-value>
14    </init-param>
15    <load-on-startup>1</load-on-startup>
16  </servlet>
17  <servlet-mapping>
18    <servlet-name>smvc</servlet-name>
19    <url-pattern>*.do</url-pattern>
20  </servlet-mapping>
21  <filter>
22    <filter-name>encoding</filter-name>
23    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
24    <init-param>
25      <param-name>encoding</param-name>
26      <param-value>utf-8</param-value>
27    </init-param>
28  </filter>
29  <filter-mapping>
30    <filter-name>encoding</filter-name>
31    <url-pattern>/*</url-pattern>
32  </filter-mapping>

```

E、测试功能

```
1 | http://localhost:8080/06-SpringSpringMVCMYbatis/stulist.do
```

2、查询学生详情

A、页面链接

```
1 | index页面操作中:<a href="toDetail.do?sno=${stu.sno}">详情</a>
```

B、数据层

```

1 | @Select("select * from student where sno=#{sno}")
2 | Student selectBySno(Integer sno);

```

C、业务层

```

1  Student selectBySno(Integer sno);//业务接口
2  @Service
3  public class StudentServiceImpl implements StudentService { //业务实现类
4      @Autowired //或者@Resource进行自动装配
5      private StudentMapper sMapper;
6      @Override
7      public Student selectBySno(Integer sno) {
8          return sMapper.selectBySno(sno);
9      }
10 }

```

D、控制层

```

1  @RequestMapping("/toDetail.do")
2  public ModelAndView selectBySno(Integer sno){
3      Student stu=stuService.selectBySno(sno);
4      if(stu!=null){
5          return new ModelAndView("detail","stu",stu);
6      }
7      return null;
8  }

```

3、删去某个学生

A、页面链接

```

1  <a href="deleteStu.do?sno=${stu.sno}">删去</a>

```

B、数据层

```

1  @Delete("delete from student where sno=#{sno}")
2  int deleteStu(Integer sno);

```

C、业务层

```

1  int deleteStu(Integer sno);//业务接口
2
3  @Service
4  public class StudentServiceImpl implements StudentService { //业务实现类
5      @Autowired //或者@Resource进行自动装配
6      private StudentMapper sMapper;
7      @Transactional
8      public int deleteStu(Integer sno) {
9          return sMapper.deleteStu(sno);
10     }
11 }

```


D、控制层

```
1 @RequestMapping("/deleteStu.do")
2 public ModelAndView deleteStu(Integer sno){
3     int num = stuService.deleteStu(sno);
4     if(num>0){
5         return new ModelAndView("forward:stulist.do","msg","删去成功!");
6     }
7     return null;
8 }
```

4、修改某个学生

A、页面链接

```
1 <a href="toUpdate.do?sno=${stu.sno}">修改</a>
```

B、数据层

```
1 int updateStu(Student stu);
```

```
1 <!--映射文件中-->
2 <update id="updateStu" parameterType="student">
3     update student
4     <set>
5         <if test="sname!=null and sname!='">sname=#{sname},</if>
6         <if test="sex!=null and sex!='">sex=#{sex},</if>
7         <if test="age!=null">age=#{age},</if>
8         <if test="gid!=null">gid=#{gid},</if>
9         <if test="phone!=null and phone!='">phone=#{phone},</if>
10        <if test="address!=null and address!='">address=#{address}</if>
11    </set>
12    where sno=#{sno}
13 </update>
14
```

C、业务层

```
1 int updateStu(Student stu); //业务接口
2
3 @Transactional
4 public int updateStu(Student stu) { //接口实现类
5     return smapper.updateStu(stu);
6 }
```

D、控制层

```

1 @RequestMapping("/updateStu.do")
2 public ModelAndView update(Student stu){
3     int num = stuService.updateStu(stu);
4     if(num>0){
5         return new ModelAndView("forward:stuList.do","msg","更新成功!");
6     }
7     return null;
8 }

```

5、增加单个学生

A、页面链接

```

1 <h2><a href="toAdd.do">增加学生</a></h2>

```

B、数据层

```

1 @Insert("insert into student(sname,sex,age,gid,phone,address) values(#{sname},#{sex},#{age},#{gid},#{phone},#{address})")
2 int addStu(Student stu);

```

C、业务层

```

1 int addStu(Student stu); //业务接口
2
3 @Transactional
4 public int addStu(Student stu) { //业务接口实现类
5     return sMapper.addStu(stu);
6 }

```

D、控制层

```

1 @RequestMapping("/toAdd.do")
2 public String toAdd(){
3     return "add";
4 }
5
6 @RequestMapping("/addStu.do")
7 public ModelAndView addStu(Student stu){
8     int num = stuService.addStu(stu);
9     if(num>0){
10         return new ModelAndView("forward:stuList.do","msg","增加成功!");
11     }
12     return null;
13 }

```

四、课堂总结

五、更多视频扩展资源

今日头条：<https://www.toutiao.com/c/user/1736832959129566/> 进去后-选择视频-在线点击观看

哔哩哔哩：<https://space.bilibili.com/425233867>