# 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

```
<dependencies>
    <!--mybatis依赖-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.4</version>
    </dependency>
    <!--spring-web依赖-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.2.4.RELEASE</version>
    </dependency>
    <!--mybatis-spring依赖-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.4</version>
    </dependency>
    <!--bonecp连接池配置 -->
    <dependency>
        <groupId>com.jolbox</groupId>
        <artifactId>bonecp</artifactId>
        <version>0.8.0.RELEASE</version>
    </dependency>
    <!--mysql驱动-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.48</version>
    </dependency>
    <!--spring-jdbc驱动-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.2.4.RELEASE</version>
    </dependency>
     <!--加入日志log4j -->
     <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>
    <!--slf4j-nop依赖-->
    <!--导入jstl依赖 -->
```

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

## B、配置文件

### a、spring-config.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:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        https://www.springframework.org/schema/aop/spring-aop.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
        https://www.springframework.org/schema/tx/spring-tx.xsd">

   <!--配置数据源-->
   <bean id="dataSource" class="com.jolbox.bonecp.BoneCPDataSource"
   p:driverClass="com.mysql.jdbc.Driver" p:jdbcUrl="jdbc:mysql://localhost:3306/ms"
   p:username="root" p:password="admin"/>

   <!--配置SqlSeesionFactory对象 -->
   <bean id="sqlFactroy" class="org.mybatis.spring.SqlSessionFactoryBean"
    p:dataSource-ref="dataSource" p:configLocation="classpath:mybatis-config.xml"/>

    <!--注入映射器-->
    <!--basePackage指定了扫描的基准包，批量产生映射器实现类 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"
    p:basePackage="com.ssm.dao"/>

    <!--配置基于注解扫描的Bean 将pojo实例化到spring容器中-->
    <context:component-scan base-package="com.ssm"/>

    <!--配置事务管理器 -->
    <bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
    p:dataSource-ref="dataSource"/>

    <!--配置使用注解声明式事务处理-->
    <tx:annotation-driven transaction-manager="txManager"/>
```

```
 </beans>
```

## b、springmvc-config.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:p="http://www.springframework.org/schema/p"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!--扫描组件 -->
    <!--告诉Spring 到哪里去找标记为@Controller 的Controller控制器-->
    <context:component-scan base-package="com.ssm.controller"/>
    <!--启用注解驱动-mvc配置 -->
    <mvc:annotation-driven/>
    <!--配置视图解析器 -->
    <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
    p:prefix="/WEB-INF/jsp/" p:suffix=".jsp"/>

  <!--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-->
  <!-- <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver"
  p:maxUploadSize="802400" p:defaultEncoding="utf-8"/> -->
</beans>
```

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

```
    </mappers>
</configuration>
```

### d、log4j.properties

```
### \u8BBE\u7F6ELogger\u8F93\u51FA\u7EA7\u522B\u548C\u8F93\u51FA\u76EE\u7684\u5730
###
log4j.rootLogger=debug,stdout,logfile

### \u628A\u65E5\u5FD7\u4FE1\u6O6F\u8F93\u51FA\u5230\u63A7\u5236\u53F0 ###
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.SimpleLayout

### \u628A\u65E5\u5FD7\u4FE1\u6O6F\u8F93\u51FA\u5230\u6587\u4EF6\uFF1Ajbit.log ###
log4j.appender.logfile=org.apache.log4j.FileAppender
log4j.appender.logfile.File=jbit.log
log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
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

```
public interface StudentMapper {
    @Select("select * from student")
     List<Student> getStulist();
}
```

### c、业务逻辑层

```
public interface StudentService { //业务接口

    List<Student> getStulist();
}
@Service
public class StudentServiceImpl implements StudentService {//业务接口实现类

    @Autowired //或者@Resource进行自动装配
    private StudentMapper sMapper;

    @Override
    public List<Student> getStulist() {
        return sMapper.getStulist();
    }
}
```

### d、控制层

```java
@Controller
public class StudentController {

    @Autowired
    private StudentService stuService;

    @RequestMapping("/toIndex.do")
    public String toIndex(){
        return "index";
    }

    @RequestMapping("/stulist.do")
    public ModelAndView selectAllStu(){
        List<Student> slist=stuService.getStulist();
        if(slist!=null){
            return new ModelAndView("index","slist",slist);
        }
        return null;
    }
}
```

### e、首页列表

```jsp
<body>
<div align="center">
    <table align="center" border="1" cellpadding="0" cellspacing="0" width="50%">
        <tr>
            <th>学号</th>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
            <th>操作</th>
        </tr>
        <c:forEach items="${slist}" var="stu">
            <tr class="base">
                <th>${stu.sno}</th>
                <th>${stu.sname}</th>
                <th>${stu.sex}</th>
                <th>${stu.age}</th>
                <th>操作</th>
            </tr>
        </c:forEach>
    </table>
</body>
```

## D、配置web.xml

```xml
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring-config.xml</param-value>
  </context-param>
```

```
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <servlet>
    <servlet-name>smvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>smvc</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
  <filter>
    <filter-name>encoding</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>encoding</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

### E、测试功能

```
http://localhost:8080/06-SpringSpringMVCMyBatis/stulist.do
```

## 2、查询学生详情

### A、页面链接

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

### B、数据层

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

### C、业务层

```
Student selectBySno(Integer sno);//业务接口
@Service
public class StudentServiceImpl implements StudentService {//业务实现类
    @Autowired //或者@Resource进行自动装配
    private StudentMapper sMapper;
    @Override
    public Student selectBySno(Integer sno) {
        return sMapper.selectBySno(sno);
    }
}
```

### D、控制层

```
@RequestMapping("/toDetail.do")
public ModelAndView selectBySno(Integer sno){
    Student stu=stuService.selectBySno(sno);
    if(stu!=null){
        return new ModelAndView("detail","stu",stu);
    }
    return null;
}
```

## 3、删去某个学生

### A、页面链接

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

### B、数据层

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

### C、业务层

```
int deleteStu(Integer sno);//业务接口

@Service
public class StudentServiceImpl implements StudentService {//业务实现类
    @Autowired //或者@Resource进行自动装配
    private StudentMapper sMapper;
    @Transactional
    public int deleteStu(Integer sno) {
        return sMapper.deleteStu(sno);
    }
}
```

### D、控制层

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

## 4、修改某个学生

### A、页面链接

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

### B、数据层

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

```
<!--映射文件中-->
<update id="updateStu" parameterType="student">
    update student
    <set>
        <if test="sname!=null and sname!=''">sname=#{sname},</if>
        <if test="sex!=null and sex!=''">sex=#{sex},</if>
        <if test="age!=null">age=#{age},</if>
        <if test="gid!=null">gid=#{gid},</if>
        <if test="phone!=null and phone!=''">phone=#{phone},</if>
        <if test="address!=null and address!=''">address=#{address}</if>
    </set>
    where sno=#{sno}
</update>

```

### C、业务层

```
int updateStu(Student stu);//业务接口

@Transactional
public int updateStu(Student stu) {//接口实现类
    return sMapper.updateStu(stu);
}
```

### D、控制层

```java
@RequestMapping("/updateStu.do")
public ModelAndView update(Student stu){
    int num = stuService.updateStu(stu);
    if(num>0){
        return new ModelAndView("forward:stulist.do","msg","更新成功!");
    }
    return null;
}
```

# 5、增加单个学生

## A、页面链接

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

## B、数据层

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

## C、业务层

```java
int addStu(Student stu);//业务接口

@Transactional
public int addStu(Student stu) {//业务接口实现类
    return sMapper.addStu(stu);
}
```

## D、控制层

```java
@RequestMapping("/toAdd.do")
    public String toAdd(){
       return "add";
    }

@RequestMapping("/addStu.do")
public ModelAndView addStu(Student stu){
    int num = stuService.addStu(stu);
    if(num>0){
        return new ModelAndView("forward:stulist.do","msg","增加成功!");
    }
    return null;
}
```

# 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