SSM 整合课程教案

版本号：JAVAEE1.0

密　级：受控文档

课

程

标

准

化

2018 年 **07** 月 **01** 日

1.文档属性

文档属性	内容
项目/任务名称：	
项目/任务编号：	
文档名称：	
文档编号：	
文档状态：	
作　　者：	实训部开发组
文档评审通过日期：	
评审负责人签字：	
参考模板：	

2.文档变更过程

版本	修正日期	修正人	描述

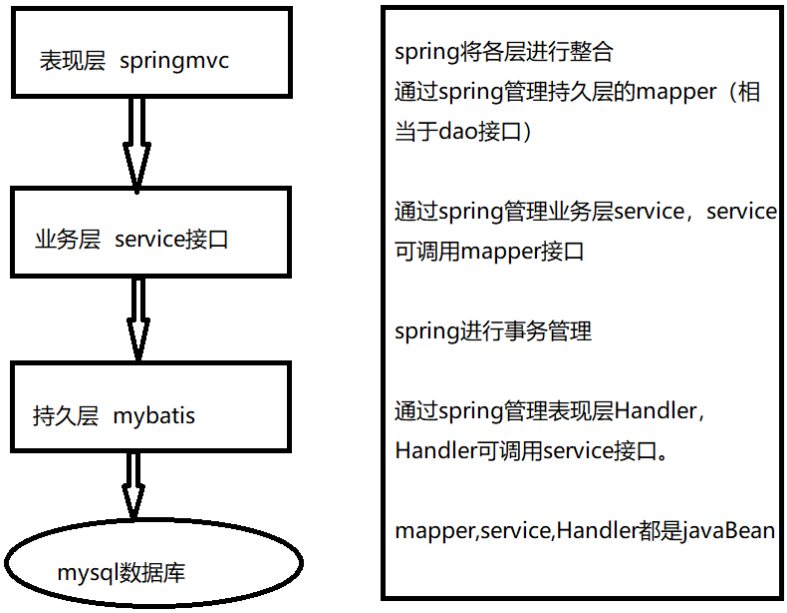
目录

1 整合环境搭建

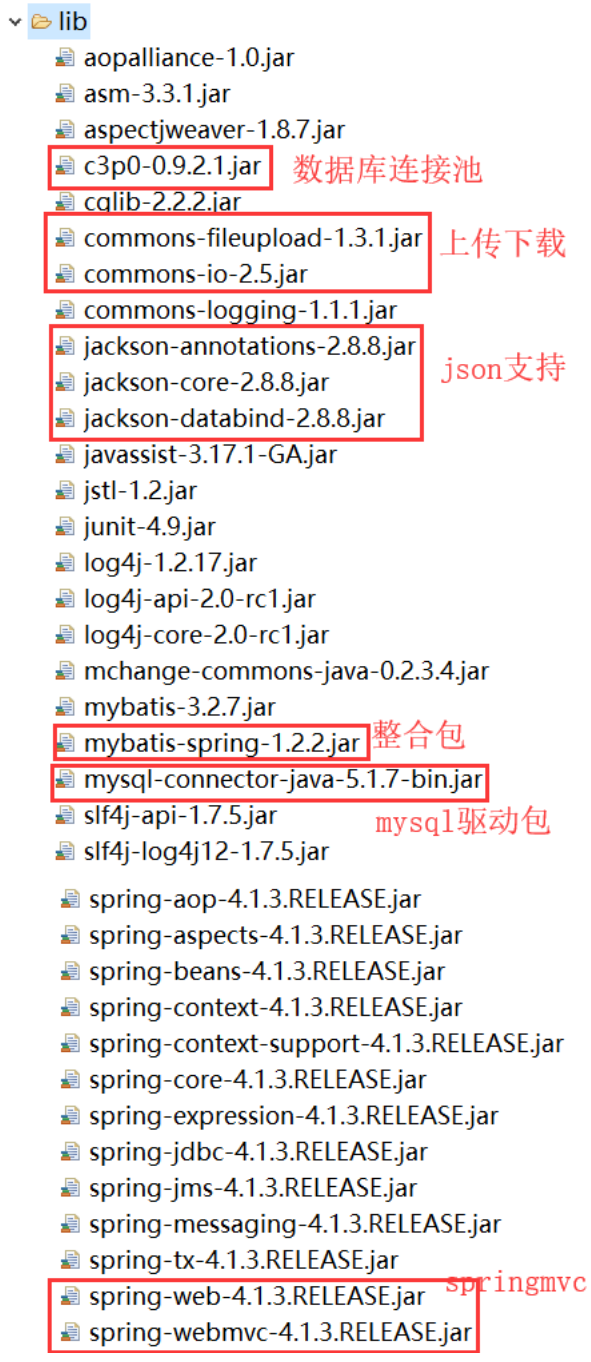
1.1 整合思想

Spring+Mybatis+SpringMVC 框架是目前流行的 Java 框架组合。
整合整体分为三层：Dao 层，Service 层和表现层。

体系架构如下图：



1.2 准备 jar

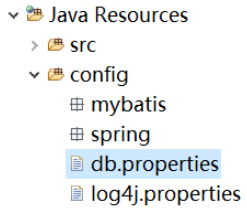


1.3 核心配置文件

- 新建一个 web 工程，将所需 jar 包添加到 lib 目录下，并且 build path

1.3.1 数据库、日志配置文件

- 根目录下创建一个放置配置文件的名为 config 的源文件夹（Source Folder）,在该文件夹下分别创建数据库常量配合文件 db.properties，日志配置文件 log4j.properties ，spring 配置文件目录以及 mybatis 配置文件目录。



db.properties:

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8
jdbc.username=root
jdbc.password=root
```

log4j.properties:

```
# Global logging configuration
log4j.rootLogger=DEBUG, stdout
# Console output...
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

1.3.2 spring 配置文件

- 在 config/spring 目录中新建配置文件 applicationContext.xml 和 springmvc.xml

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
     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
       http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
       http://www.springframework.org/schema/tx
```

```
http://www.springframework.org/schema/tx/spring-tx.xsd
        ">
    <!-- 1、加载外部资源文件 -->
    <context:property-placeholder location="classpath:db.properties" />
    <!-- 2、配置数据库连接池 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="${jdbc.driver}"></property>
        <property name="jdbcUrl" value="${jdbc.url}"></property>
        <property name="user" value="${jdbc.username}"></property>
        <property name="password" value="${jdbc.password}"></property>
    </bean>
    <!-- 3、配置sqlSessionFactory 让spring来管理sqlSessionFactory,使用mybatis-
spring包中的SqlSessionFactoryBean -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 3.1 配置数据源 -->
        <property name="dataSource" ref="dataSource"></property>
        <!-- 3.2 加载mybatis全局配置文件 -->
        <property name="configLocation" value="classpath:mybatis/mybatis-
config.xml"></property>
         <!-- 3.3 指定别名包 -->
        <!--<property name="typeAliasesPackage" value="com.hpe.po"></property> -
->
    </bean>
    <!-- 4、使用包扫描的方式  批量创建mapper的bean ， bean的id就是接口名称，首字母小写
-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 指定Mapper接口所在的包 basePackage：扫描包路径，中间可以用逗号或分号分隔
定义多个包 -->
        <property name="basePackage" value="com.hpe.mapper"></property>
    </bean>
    <!-- 5、注解扫描 -->
    <context:component-scan base-package="com.hpe.service"></context:component-
scan>
    <!-- 6.1、spring的声明式事务  事务管理器-->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 注入数据源 -->
        <property name="dataSource" ref="dataSource"></property>
    </bean>
    <!-- 6.2  注册事务注解驱动 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>
</beans>
```

1.3.3 springmvc 配置文件

springmvc.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"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    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
       http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <context:component-scan base-
package="com.hpe.controller"></context:component-scan>
    <!-- 注解驱动，会自动的注册许多bean，包括注解映射器和注解适配器 -->
    <mvc:annotation-driven></mvc:annotation-driven>
    <!-- 配置静态资源的访问映射，此配置中的文件，将不会被前端控制前拦截 -->
    <mvc:resources location="/js/" mapping="/js/**"></mvc:resources>
    <!-- 视图解析器，支持jsp -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
</beans>
```

1.3.4 mybatis 配置文件

- 在 config/mybatis 下新建 mybatis-config.xml 配置文件

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>
    <typeAliases>
        <package name="com.hpe.po" />
    </typeAliases>
</configuration>
```

1.3.5 web.xml

web.xml 文件配置如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">
    <display-name>20180724_ssm_01</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <!-- 配置加载spring文件的监听器 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/applicationContext.xml</param-value>
    </context-param>
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <servlet>
        <!-- 配置前端控制器 -->
        <servlet-name>springmvc</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!-- 初始化时加载配置文件 -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/springmvc.xml</param-value>
        </init-param>
        <!-- 表示容器在启动时 当前Servlet的加载顺序 -->
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

```
    <!-- 配置编码过滤器 -->
    <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>

</web-app>
```

1.4 po、mapper、service

在 com.hpe.po 下新建 User 对象

```
public class User {
    private int id;
    private String username;// 用户姓名
    private String sex;// 性别
    private Date birthday;// 生日
    private String address;// 地址
```

在 com.hpe.mapper 下新建 UserMapper 接口

```
public interface UserMapper {
    //根据id查询用户信息
    public User findUserById(int id);
}
```

在 com.hpe.mapper 下新建 UserMaper.xml

```
<?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.hpe.mapper.UserMapper">
    <select id="findUserById" parameterType="Integer" resultType="user">
        select * from user where id = #{id}
    </select>
```

```
</mapper>
```

注：mapper 动态代理

在 com.hpe.service 下新建 UserService 接口

```
public interface UserService {
    // 根据id查询用户信息
    public User findUserById(int id);
}
```

在 com.hpe.service.impl 下新建 UserService 接口的实现类

```
@Service("userService")
@Transactional()
public class UserServiceImpl implements UserService{
    @Resource
    private UserMapper userMapper;
    @Override
    public User findUserById(int id) {
        return userMapper.findUserById(id);
    }
}
```

注：注解声明式事务、注解 bean

1.5 控制层、页面

在 com.hpe.controller 下新建 UserController

```
@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserService userService;
    @RequestMapping("/findUserById")
    public String findUserById(int id, Model model){
        User user = userService.findUserById(id);
        model.addAttribute("user", user);
        return "user";
    }
}
```

在 WEB-INF/jsp 下新建 user.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
```

```
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>学生信息</title>
</head>
<body>
    <table border="1">
        <tr>
            <td>编号</td>
            <td>姓名</td>
            <td>性别</td>
            <td>生日</td>
            <td>地址</td>
        </tr>
        <tr>
            <td>${user.id }</td>
            <td>${user.username }</td>
            <td>${user.sex }</td>
            <td>${user.birthday }</td>
            <td>${user.address }</td>
        </tr>
    </table>
</body>
</html>
```

在 WEB-INF 下新建 index.jsp

```
<a href="${pageContext.request.contextPath }/user/findUserById?id=1">查询用户</a>
```

1.6 1.6Tomcat 启动、测试

启动项目，访问项目，会跳转到 index.jsp 欢迎页，点击 查询用户，效果如下:

编号	姓名	性别	生日	地址
1	张三	男	Mon Jan 01 00:00:00 CST 1990	山东济南