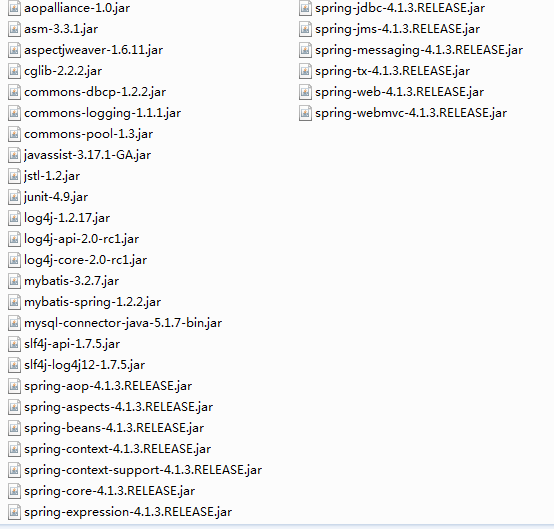
# 整合

依赖的jar包：



整合JSON需要的jar包：



SSM整合：

1. Dao层

Pojo和映射文件以及接口使用逆向工程生成。

SqlMapConfig.xml MyBatis的核心配置文件。

ApplicatonContext-dao.xml整合后Spring在dao层的配置：配置数据源，配置会话工厂，扫描Mapper。

1. Service层

事物 ApplicationContext-trans.xml

@Service注解扫描 ApplicationContext-service.xml

1. Controller层

SpringMVC.xml

配置注解扫描：扫描@Controller注解。

配置注解驱动：配置最新版本的处理器映射器和处理器适配器。

配置视图解析器。

1. web.xml

SpringMVC前端控制器（DispatcherServlet）。

Spring监听器。

SqlMapConfig.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>    <settings>  <!--使用jdbc的getGeneratedKeys获取数据库自增主键值 -->  <setting name="useGeneratedKeys" value="true" />  <!--使用列标签代替列别名 -->  <setting name="useColumnLabel" value="true" />  <!--开启驼峰命名转换 -->  <setting name="mapUnderscoreToCamelCase" value="true" />  </settings>  </configuration> |

ApplicatonContext-dao.xm：

|  |
| --- |
| <?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-3.1.xsd ">    <!--引入jdbc的属性文件 -->  <context:property-placeholder location="classpath:jdbc.properties"/>  <!--配置数据库连接池 -->  <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="${jdbc.user}"/>  <property name="password" value="${jdbc.password}"/>    <!--c3p0私有属性 -->  <property name="maxPoolSize" value="30"/>  <property name="minPoolSize" value="10"/>  <property name="autoCommitOnClose" value="false"/>  <property name="checkoutTimeout" value="10000"/>  <!--当获取连接失败后重试的次数 -->  <property name="acquireRetryAttempts" value="2"/>    </bean>    <!--配置SqlSessionFactory对象 -->  <bean id="sqlSessionFactoryName" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource" /><!--注入数据库连接池 -->  <property name="configLocation" value="classpath: SqlMapConfig.xml"/><!--配置mybatis全局配置文件-->  <property name="typeAliasesPackage" value="code.lsh.entity"/><!--扫描pojo包 使用别名 -->  <property name="mapperLocations" value="classpath:mapper/\*.xml"/>  </bean>    <!--配置扫描Dao接口包，动态实现dao接口，注入到spring容器中 -->  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="sqlSessionFactoryBeanName" value="sqlSessionFactoryName"/>  <property name="basePackage" value="code.lsh.dao" />  </bean>  </beans> |

ApplicationContext-service.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:aop="http://www.springframework.org/schema/aop"  xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.2.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx-4.2.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.2.xsd">  <!--扫描service包下所有使用注解的类型 -->  <context:component-scan base-package="code.lsh.service" />  <!--配置事务管理器 -->  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property name="dataSource" ref="dataSource" /><!--注入数据库连接池 -->  </bean>  <!--配置基于注解的事务声明方式 -->  <tx:annotation-driven transaction-manager="transactionManager"/>  </beans> |

SpringMVC.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:p="http://www.springframework.org/schema/p"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:aop="http://www.springframework.org/schema/aop"  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-4.2.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">  <!--开启SpringMVC注解模式 -->  <mvc:annotation-driven />  <!--静态资源默认servlet配置 允许/作为整体映射-->  <mvc:resources location="/resources/" mapping="\*/resources/\*\*"></mvc:resources>  <mvc:default-servlet-handler/>  <!--定义视图解析器 -->  <bean id="" class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/html/" />  <property name="suffix" value=".html" />  </bean>    <!--文件上传解析器 -->  <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  <property name="defaultEncoding" value="utf-8"></property>  <!-- 1024\*1024\*20=20M -->  <property name="maxUploadSize" value="20971520"></property>  <property name="maxInMemorySize" value="20971520"></property>  </bean>    <!-- 扫描web相关的bean -->  <context:component-scan base-package="code.lsh.web" />  </beans> |

Web.xml

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

# 配置SqlSessionFactoryBean

SqlSessionFactory是产生SqlSession的基础，在MyBatis-Spring项目中提供了SqlSessionFactoryBean去支持SqlSessionFactory的配置。

# 配置MapperFactoryBean

<bean id="" class="org.mybatis.spring.mapper.MapperFactoryBean">  
 <property name="mapperInterface" value=""></property>  
 <property name="sqlSessionFactory" ref="sqlSessionFactory"></property>  
 <property name="sqlSessionTemplate" ref=""></property>  
</bean>

mapperInterface：映射器的接口

如果同时配置SqlSessionFactory和SqlSessionTemplate，那么它就会启用SqlSessionTemplate，而SqlSessionFactory就作废。

# 配置MapperScannerConfigurer

这是一个通过扫描的形式进行配置Mapper的类，如果一个个配置Mapper，显然工作量时非常大的，并且导致配置泛滥，而通过该配置就能够生成大量的Mapper

主要配置项：

basePackage：指定让Spring自动扫描什么包，它会逐层深入扫描，如果遇到多个包可以使用半角号分隔。

annotationClass：表示如果类被这个注解标识的时候，才进行扫描。对于开发而言，笔者建议使用这个方式进行注册对应的Mapper

SqlSessionFactoryBeanName：指定在Spring中定义SqlSessionFactory的Bean名称。如果sqlSessionTemplateBeanName被定义，那么它将失去作用。

markerInterface：指定实现了什么接口就分为它是Mapper，我们需要提供一个公共的接口去标记。