1. 创建父项目

|  |
| --- |
| 1.1建立一个maven project作为父项目Parent(不要使用以及选择archetype)  1.2删除父项目中的src文件夹，在pom.xml中配置打包方式<packaging>pom</packaging> |

1. 创建子项目

|  |
| --- |
| 2.1选中父项目创建module，选择maven项目，勾选Create from archetype，  选择archetype-quickstart模板，填写airfactId。  依次建立子项目commons\_utils、domain、dao、service  2.2选中父项目创建module，选择maven项目，勾选Create from archetype，  选择archetype-webapp模板，填写airfactId。  建立子项目web |

1. 配置父项目的pom.xml中的共享依赖(ssm的15个基本依赖+1个文件上传的依赖

|  |
| --- |
| <!-------------------声明依赖的版本------------------->  <properties>  <mybatis.version>3.2.8</mybatis.version>  <mybatis-spring.version>1.3.1</mybatis-spring.version>  ...  ...  ...  </properties>  <dependencyManagement>  <dependencies>  <!-------------------数据库访问层------------------->  <!--mybatis -->  <!-- mybatis-spring-->  <!--dataSource(阿里的德鲁伊连接池druid) -->  <!--mysql -->  <!-- pageHelper-->  <!----------------------service层----------------------->  <!-- spring-beans（depend spring-core）-->  <!-- spring-context（depend spring-core）-->  <!--spring-jdbc（depend spring-tx） -->  <!--spring-aspects -->  <!-- aspectj-weaver-->  <!------------------------web层------------------------->  <!--springmvc（depend spring-web） -->  <!--jsp -->  <!-- servlet-->  <!--jstl -->  <!--json(jackbind denpends jackson-core and jackson-annotation) -->  <!--------------------文件上传下载-------------------->  <!-- commons-fileupload（depend commons-io）-->  </dependencies>  </dependencyManagement> |

1. 打包时拷贝资源文件

intelliJ idea中必须在build标签中如下配置

|  |
| --- |
| <!-- 资源文件拷贝 --> <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <!--<include>\*\*/\*.properties</include>-->  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  <resource>  <directory>src/main/resources</directory>  </resource>  </resources> |

1. 配置tomcat插件

|  |
| --- |
| <plugins>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <version>2.2</version>  <configuration>  <!-- 访问项目的context（即tomcat安装目录中webapps下面的文件夹名称） -->  <path>/car</path>  <!-- 上传war，需要访问的tomcat的路径（即通过该路径上传war包） -->  <url>http://192.168.30.10:8080/manager/text</url>  <!-- 设置tomcat服务器默认编码 -->  <uriEncoding>UTF-8</uriEncoding>  <!-- maven的settings.xml中配置的tomcat账户的id -->  <server>tomcat7</server>  </configuration>  </plugin>  </plugins> |

1. 配置子项目之间的依赖

|  |
| --- |
| · [一domain模块](https://blog.csdn.net/qiwenmingshiwo/article/details/78945234" \l "一bos-domain模块" \t "https://blog.csdn.net/qiwenmingshiwo/article/details/_self)  · [二utils模块](https://blog.csdn.net/qiwenmingshiwo/article/details/78945234" \l "二bos-utils模块" \t "https://blog.csdn.net/qiwenmingshiwo/article/details/_self)  配置domain依赖  · [三DAO模块](https://blog.csdn.net/qiwenmingshiwo/article/details/78945234" \l "三bos-dao模块" \t "https://blog.csdn.net/qiwenmingshiwo/article/details/_self)  配置utils依赖，配置DAO层的相关依赖  · [四service模块](https://blog.csdn.net/qiwenmingshiwo/article/details/78945234" \l "四bos-service模块" \t "https://blog.csdn.net/qiwenmingshiwo/article/details/_self)  配置mapper依赖，配置service层的相关依赖  · [五web模块](https://blog.csdn.net/qiwenmingshiwo/article/details/78945234" \l "五bos-web模块" \t "https://blog.csdn.net/qiwenmingshiwo/article/details/_self)  配置service依赖，配置web层的相关依赖 |

1. 在web子项目的src/main/resources中配置spring相关xml

【以两个系统erp、sys为例】

（xml中的schema一般都是指结构/概要，

xsd是指XML结构定义 ( XML Schemas Definition ) XML Schema 是DTD的替代品）

7.1配置mybatis.cfg.xml和log4j.properties

|  |
| --- |
| <?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>  <setting name="logImpl" value="LOG4J"/>  </settings>  <plugins>  <plugin interceptor="com.github.pagehelper.PageInterceptor"/>  </plugins>  </configuration> |
| 以下为log4j.properties |
| # Global logging configuration  log4j.rootLogger=DEBUG, stdout  # MyBatis logging configuration...  log4j.logger.org.mybatis.example.BlogMapper=TRACE  # 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 |

7.2配置application-dao.xml

DAO层包结构com.sxt.sys.dao.mapper：dao包中为mapper接口，mapper包中为.\*mapper.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  ”>  <!--配置数据源dataSource-->  <bean id=”dataSource” class=”xxx.xxx.DruidDataSource”>  <property name=”driverClassName” value=”com.mysql.jdbc.Driver”/>  <property name=”url” value=”jdbc:mysql://192.168.30.10:3306/schema”/>  <property name=”username” value=”root”/>  <property name=”password” value=”root”/>  </bean>  <!--配置SqlSessionFactoryBean并注入dataSource-->  <bean id=”sqlSessionFactory” class=”xxx.xxx.SqlSessionFactoryBean”>  <property name=”dataSource” ref=”dataSource”/>  <property name=”configLocation” value=”classPath:mybatis.cfg.xml”/>  <!--只有一个系统时直接在mapperLocations属性中配置value的唯一值-->  <property name=”mapperLocations”>  <list>  <value>classpath:com/sxt/sys/dao/mapper/\*.xml</value>  <value>classpath:com/sxt/erp/dao/mapper/\*.xml</value>  </list>  </property>  </bean>  <!--扫描mapper接口，产生mapper接口的代理对象-->  <bean class=”xxx.xxx.MapperScannerConfigurer”>  <property name=”basePackage” value=”com.sxt.erp.dao,com.sxt.sys.dao”/>  <property name=”sqlSessionFactoryBeanName” value=”sqlSessionFactory”/>  </bean>  </beans> |

7.3配置application-service.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:tx=”http://www.springframework.org/schema/tx”  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/context/spring-context.xsd](http://www.springframework.org/schema/beans/spring-beans.xsd)  [http://www.springframework.org/schema/t](http://www.springframework.org/schema/beans)x  http://www.springframework.org/schema/tx/spring-tx.xsd  [http://www.springframework.org/schema/a](http://www.springframework.org/schema/beans)op  http://www.springframework.org/schema/aop/spring-aop.xsd  ”>  <!--扫描service层,纳入spring容器管理-->  <context:component-scan base-package=”com.sxt.erp.service.impl,com.sxt.sys.service.impl”/>  <!--实例化事物管理器transactionManager-->  <bean id=”transactionManager” class=”xxx.xxx.DataSourceTransactionManager”>  <!--  只有当web.xml配置了classPath\*:application-\*.xml全扫描后  这里才能注入在application-dao.xml中配置的dataSource  -->  <property name=”dataSource” ref=”dataSource”/>  </bean>  <!--声明事物切面-->  <tx:advice id=”txAdvice” transaction-manager=”transactionManager”>  <tx:attributes>  <tx:method name=”add” isolation=”DEFAULT” propagation=”REQUIRED”>  <tx:method name=”save” isolation=”DEFAULT” propagation=”REQUIRED”>  <tx:method name=”insert” isolation=”DEFAULT” propagation=”REQUIRED”>  <tx:method name=”update” isolation=”DEFAULT” propagation=”REQUIRED”>  <tx:method name=”delete” isolation=”DEFAULT” propagation=”REQUIRED”>  <tx:method name=”load” isolation=”DEFAULT” propagation=”REQUIRED”  read-only=”true”>  <tx:method name=”get” isolation=”DEFAULT” propagation=”REQUIRED”  read-only=”true”>  <tx:method name=”\*” isolation=”DEFAULT” propagation=”REQUIRED”  read-only=”true”>  </tx:attributes>  </tx:advice>  <!--进行aop织入-->  <aop:config>  <aop:pointcut expression=”execution(\* com.sxt.erp.service.impl.\*.\*(..))” id=”pc1”/>  <aop:pointcut expression=”execution(\* com.sxt.sys.service.impl.\*.\*(..))” id=”pc2”/>  <aop:advisor advice-ref=”txAdvice” pointcut-ref=”pc1”/>  <aop:advisor advice-ref=”txAdvice” pointcut-ref=”pc2”/>  </aop:cofig>  </beans> |

7.4配置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:mvc=”http://www.springframework.org/schema/mvc”  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/c](http://www.springframework.org/schema/beans)ontext  [http://www.springframework.org/schema/c](http://www.springframework.org/schema/beans)ontext/spring-context.xsd  [http://www.springframework.org/schema/m](http://www.springframework.org/schema/beans)vc  [http://www.springframework.org/schema/m](http://www.springframework.org/schema/beans)vc/spring-mvc.xsd  ”>  <!--扫描controller,纳入spring容器的管理-->  <context:component-scan base-package=”com.sxt.erp.controller,com.sxt.sys.controller”/>  <!--开启注解驱动,配置注解映射器和注解适配器-->  <mvc:annotation-driven/>  <!--配置视图解析器-->  <bean class=”xxx.xxx.InternalResourceViewResolver”>  <property name=”prefix” value=”WEB-INF/jsp/”/>  <property name=”suffix” value=”.jsp”/>  </bean>  <mvc:resources mapping=”/resource/\*\*” location=”/resource/”/>  <!--配置拦截器-->  <mvc:interceptors>  <mvc:interceptor>  <!--指定需要拦截的路径-->  <mvc:mapping path=”/\*\*”/>  <!--指定不需要拦截的路径-->  <mvc:exclude-mapping path=”/user/toLogin\*”/>  <mvc:exclude-mapping path=”/user/login\*”/>  <!--实例化自定义拦截器类-->  <bean class=”com.sxt.sys.interceptor.LoginInterceptor”/>  </mvc:interceptor>  </mvc:interceptors>  <!--实例化二进制流解析器（文件上传）-->  <bean id=”multipartResolver” class=”xxx.xxx.CommonsMultipartResolver”>  <!-- 指定文件上传过程中提交的数据库的编码 --> <property name="defaultEncoding" value="UTF-8" /> <!-- 配置文件上传临时路径 --> <property name="uploadTempDir" value="/upload/temp" /> <!-- 配置文件上传的最大文件 10m --> <property name="maxUploadSize" value="1024000000" />  </bean>  </beans> |

1. 编写domain、utils、dao/mapper、vo、service/impl、controller、interceptor
2. 导入静态资源、编写jsp页面
3. 配置web.xml

|  |
| --- |
| <?xml version=”1.0” encoding=”UTF-8”>  <web-app  version=”2.5”  xmlns=”http://java.sun.com/xml/ns/javaee”  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”  xsi:schemaLocation=”  <http://java.sun.com/xml/ns/javaee>  <http://java.sun.com/xml/ns/javaee>/web-app\_2\_5.xsd  ”>  <display-name>xxx</display-name> <welcome-file-list>  <welcome-file>index.jsp</welcome-file> </welcome-file-list>    <!--加载spring IOC容器-->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classPath\*:application-\*.xml</param-value>  </context-param>  <!--配置监听器ContextLoaderListener-->  <listener>  <listener-class>xxx.xxx.ContextLoaderListener</listener-class>  </listener>  <!--配置springmvc的过滤器CharacterEncodingFilter-->  <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>xxx.xxx.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>\*.action</url-pattern>  </filter-mapping>  <!--配置springmvc的核心控制器DispatchServlet-->  <servlet>  <servlet-name>springmvc</servlet-name> <servlet-class>xxx.xxx.DispatcherServlet</servlet-class> <!-- 加载springmvc.xml --> <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath\*:springmvc.xml</param-value> </init-param> <!-- servlet在服务器启动时实例化 --> <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>\*.action</url-pattern>  </servlet-mapping>  </web-app> |