**创建SSM框架Maven项目**

**详细流程可查看CSDN：https://blog.csdn.net/Eternal\_Blue/article/details/90400895**

**开发工具：Eclipse**

**框架选型：SpringMVC、Spring、Mybatis**

**数据库：MySQL**

**Jar包依赖方式：apache-maven-3.6.0**

**JDK版本：jdk1.8.0\_201**

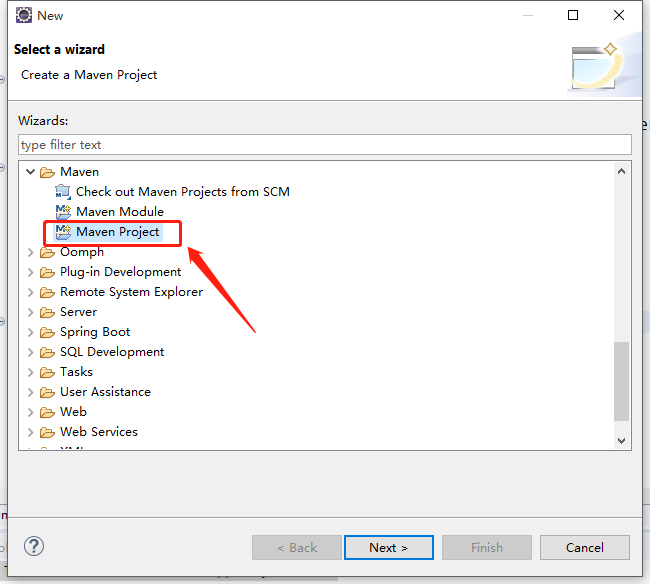
**Tomcat容器：apache-tomcat-8.5.24**

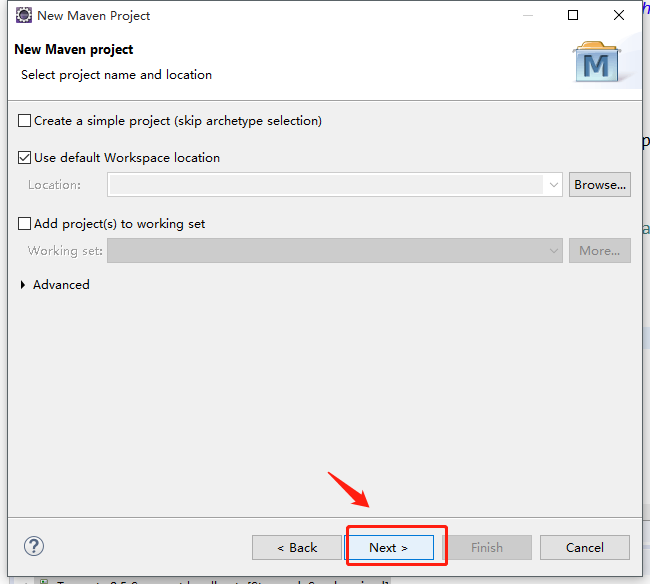
**流程：**

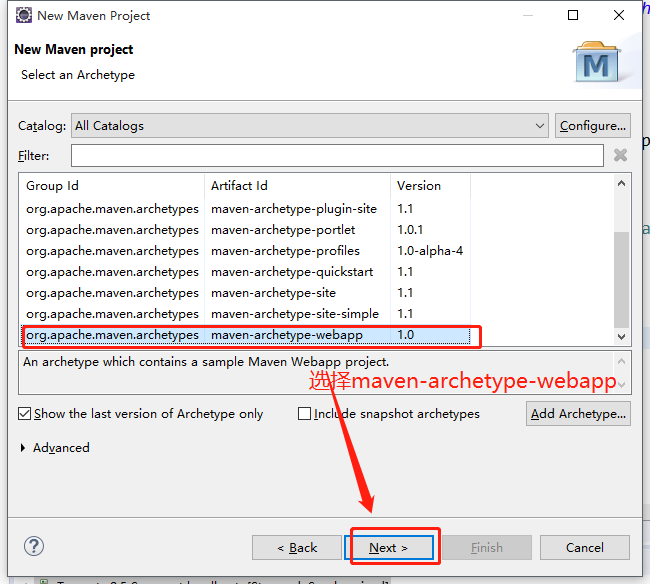
**搭建maven项目 > 搭建mvc > 搭建spring**

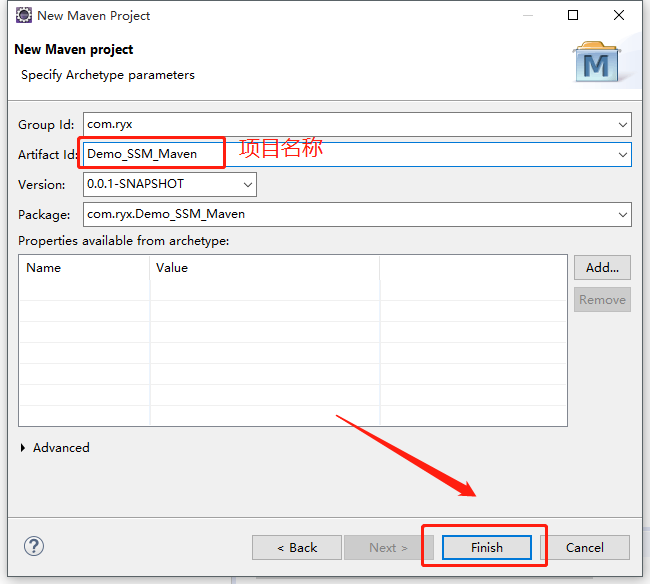
**https://blog.csdn.net/bigname22/article/details/79380238**

# 一、创建maven项目

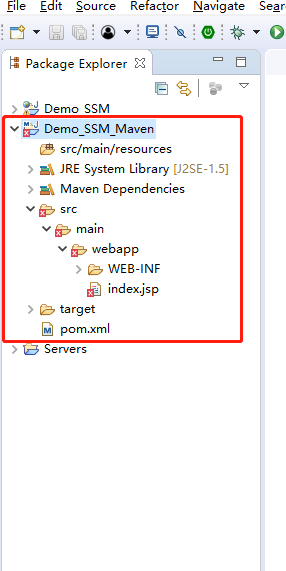






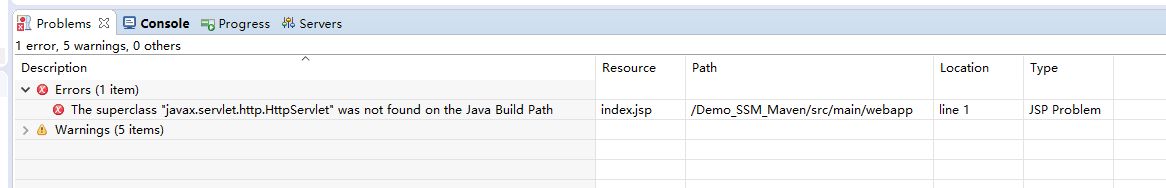


**新建的Maven项目结构为：**



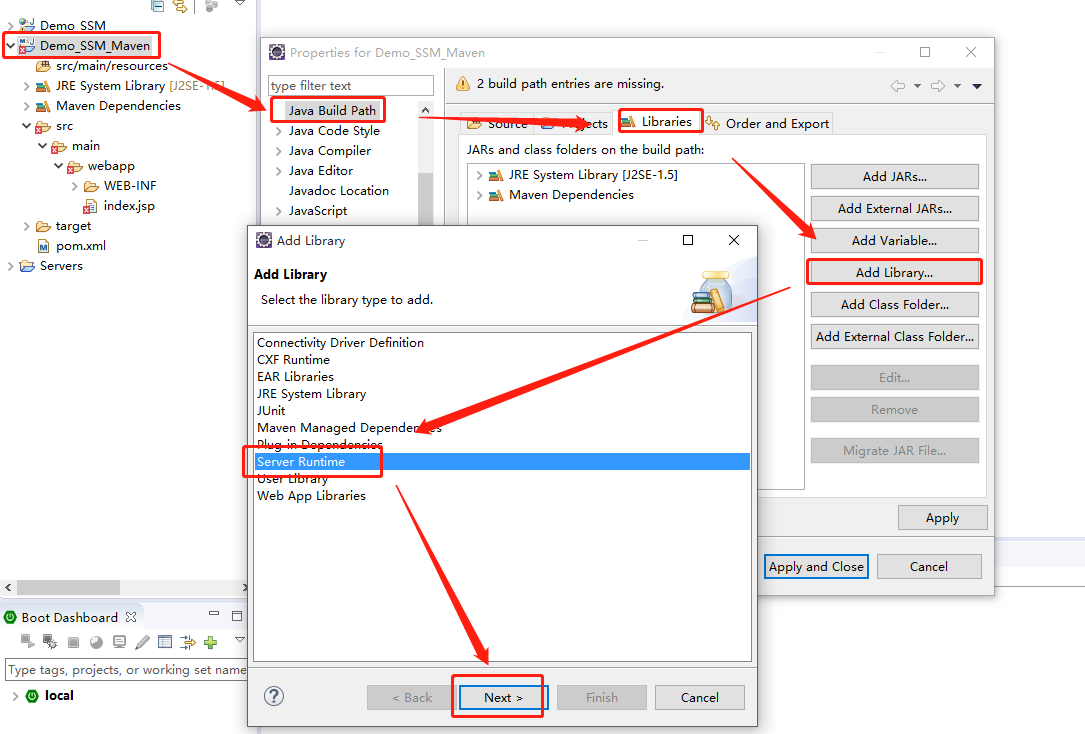
新建项目只有一个 src/main/resources，

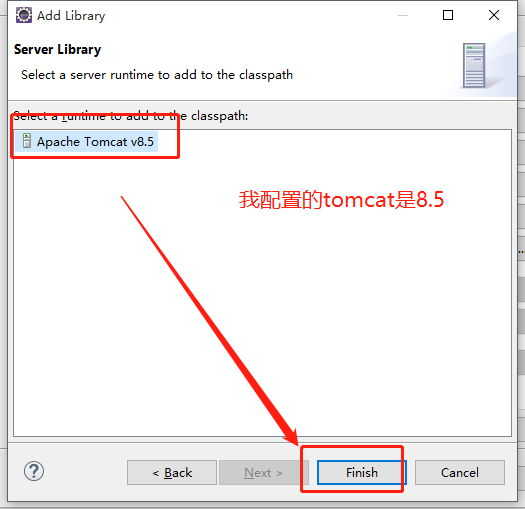
并且项目会出现错误，而且项目报下面的The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Patht 错误

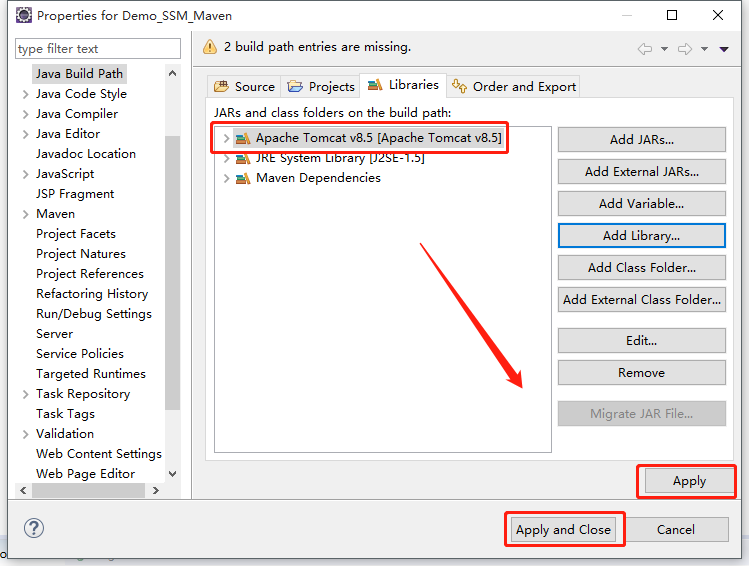


**解决方案：**

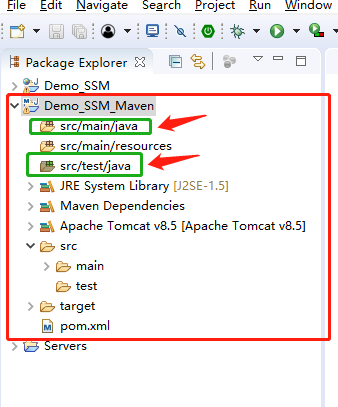
右击项目>点击Properties > 点击Java Build Path >选择Library > 点击Add Library > 选择Server Runtime > Apache Tomcat7 服务器（Tomcat8 需要 web版本3.0）





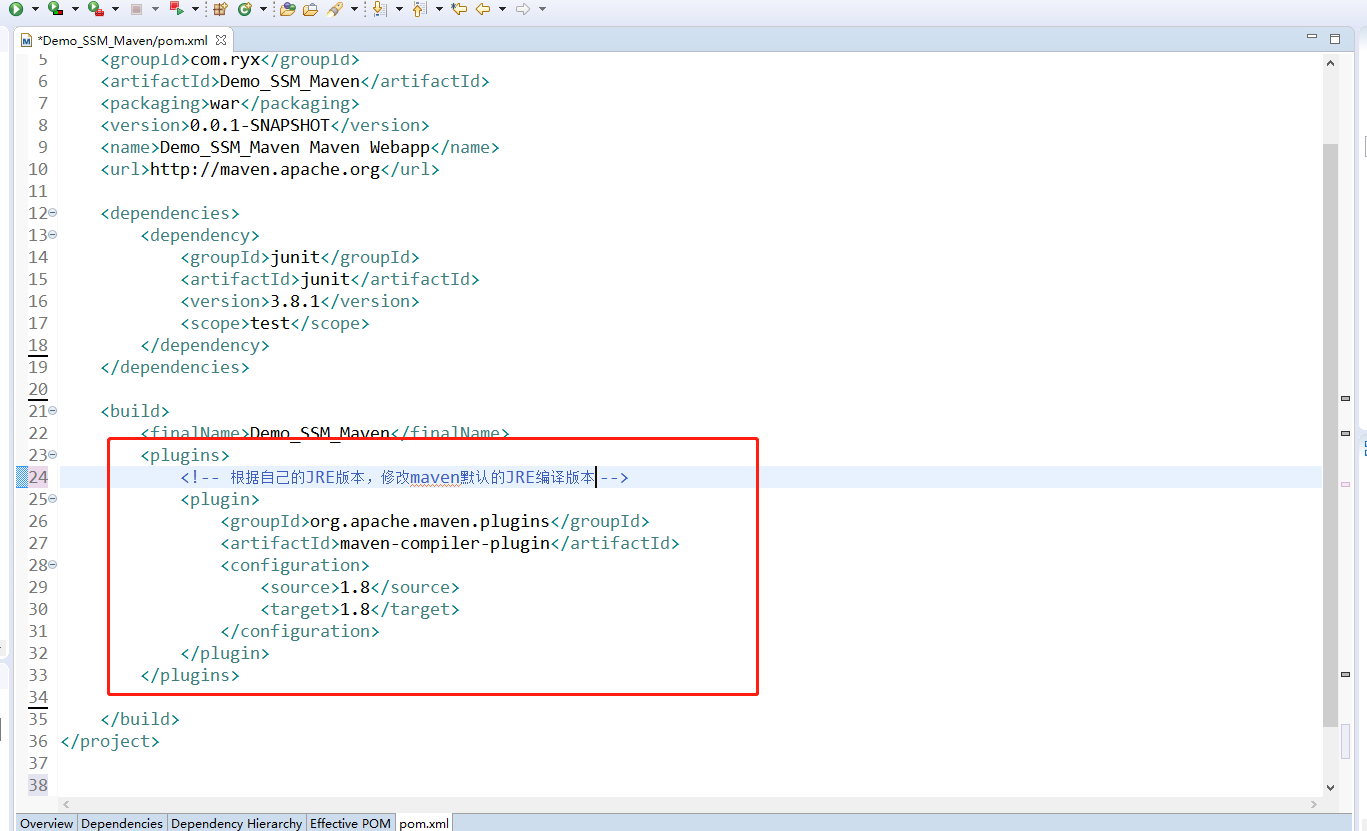


**点击Apply，干掉项目错误后，项目结构发生了改变，自动生成了src/main/java和src/test/java目录**



**此时，新建项目的jre版本是1.5，web.xml版本为2.3。我们需要对版本进行修改**

修改pom.xml中的配置



<build>

<finalName>Demo\_SSM\_Maven</finalName>

<plugins>

<!-- 修改maven默认的JRE编译版本，1.8代表JRE编译的版本，根据自己的安装版本选择1.7或1.8 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

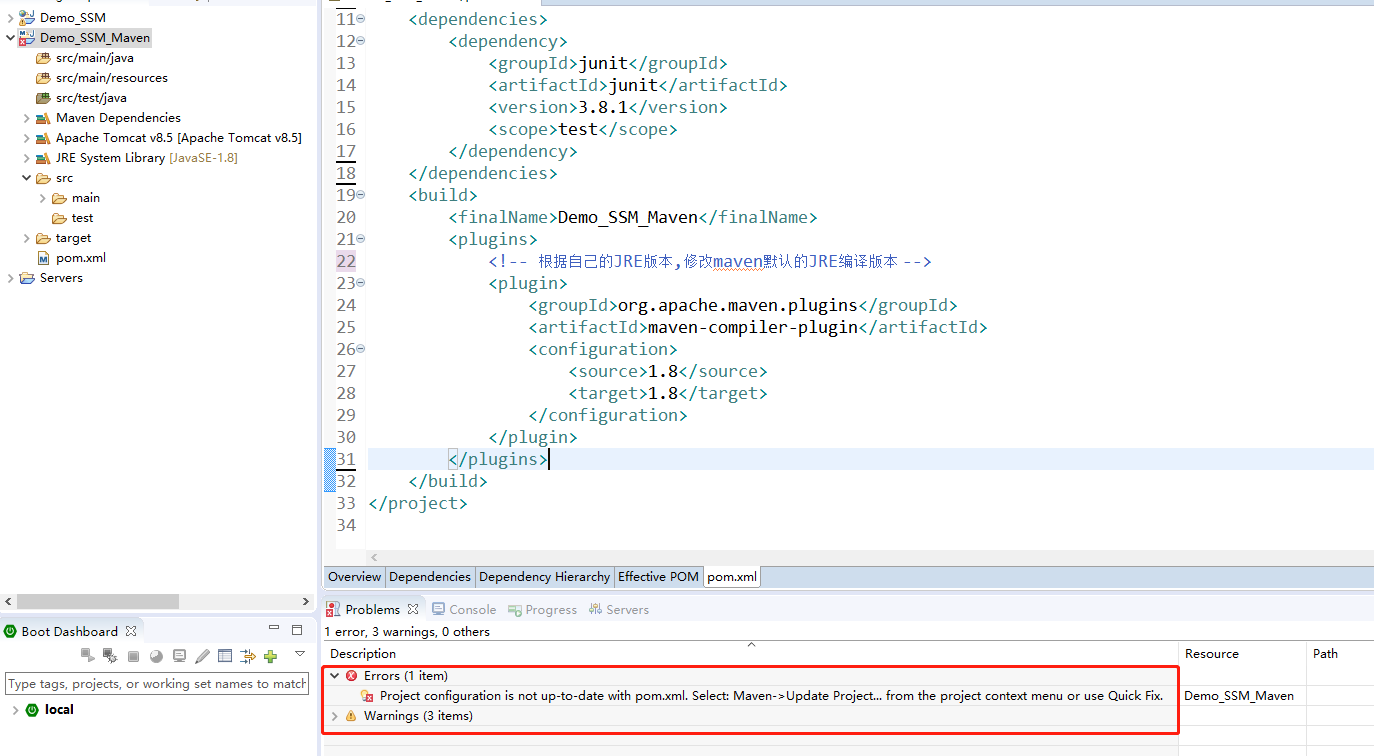
</configuration>

</plugin>

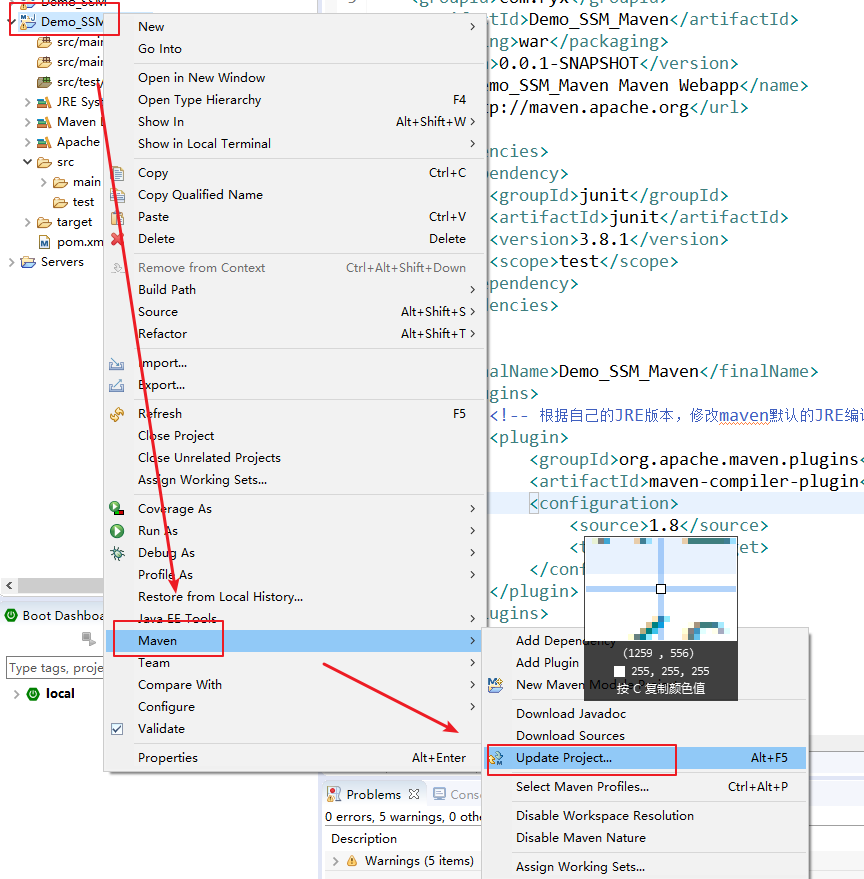
</plugins>

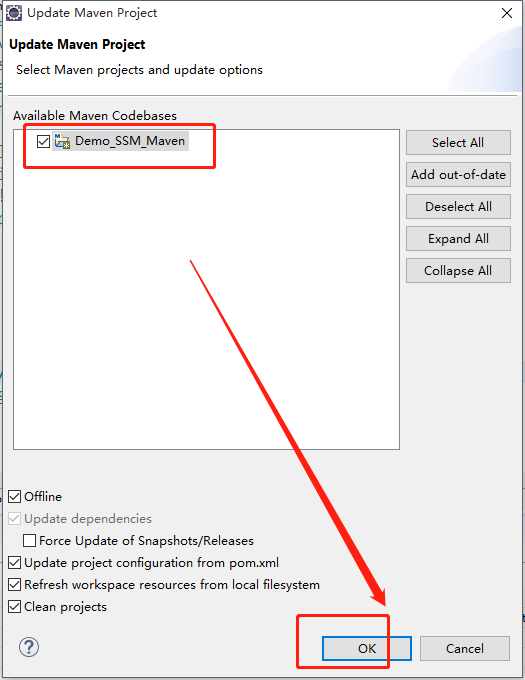
</build>

**修改完配置,会出现下面这个问题**

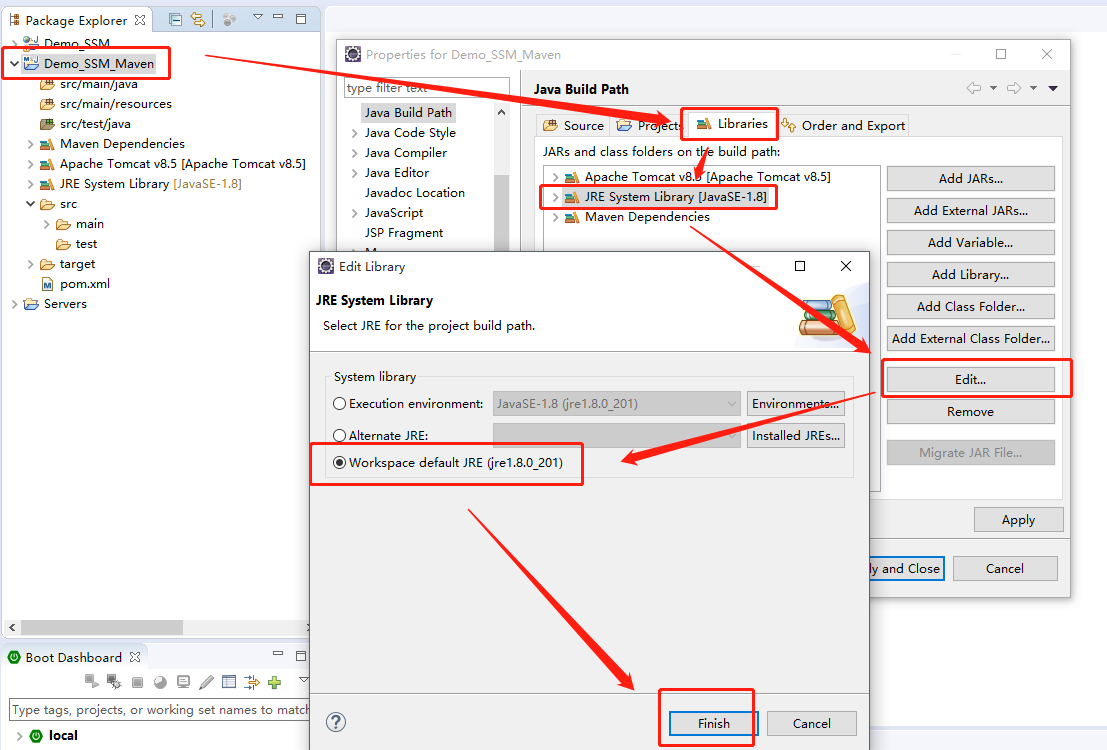


**解决：只需要更新maven。右击项目> 点击Maven >点击Update Project..**





**把jre 指定到Workspace default JRE**



**修改web.xml版本**

默认的Dynamic Web Module为2.3，使用Tomcat 8.5，需要修改为3.0

**方法一：**修改maven工程所在目录下org.eclipse.wst.common.project.facet.core.xml

<?xml version="1.0" encoding="UTF-8"?>

<faceted-project>

<fixed facet="wst.jsdt.web"/>

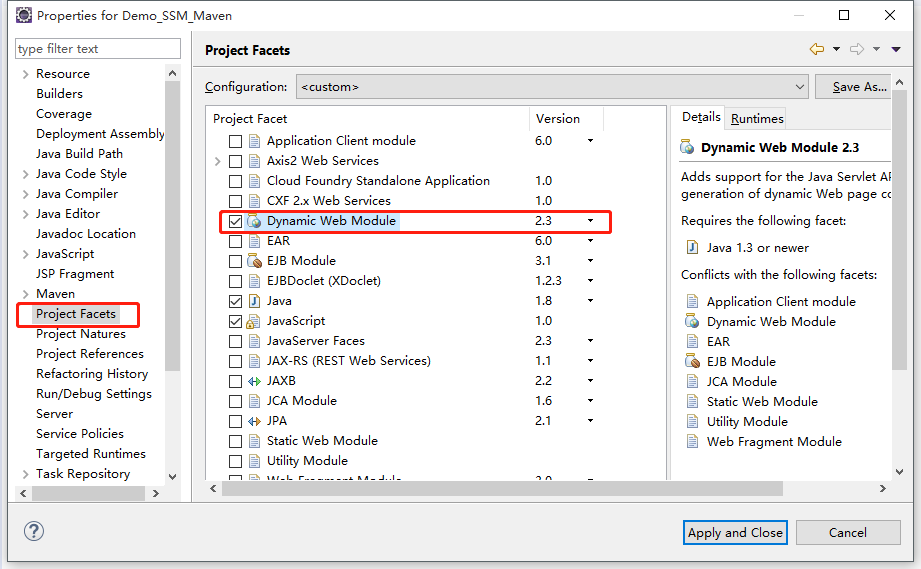
<installed facet="java" version="1.8"/>

<installed facet="jst.web" version="2.3"/> <!—2.3修改为3.0 -->

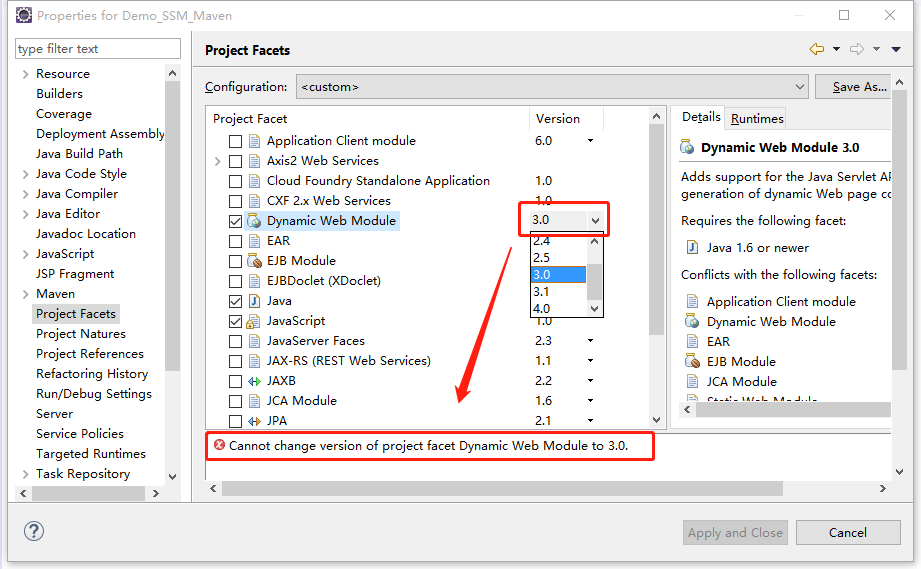
<installed facet="wst.jsdt.web" version="1.0"/>

</faceted-project>

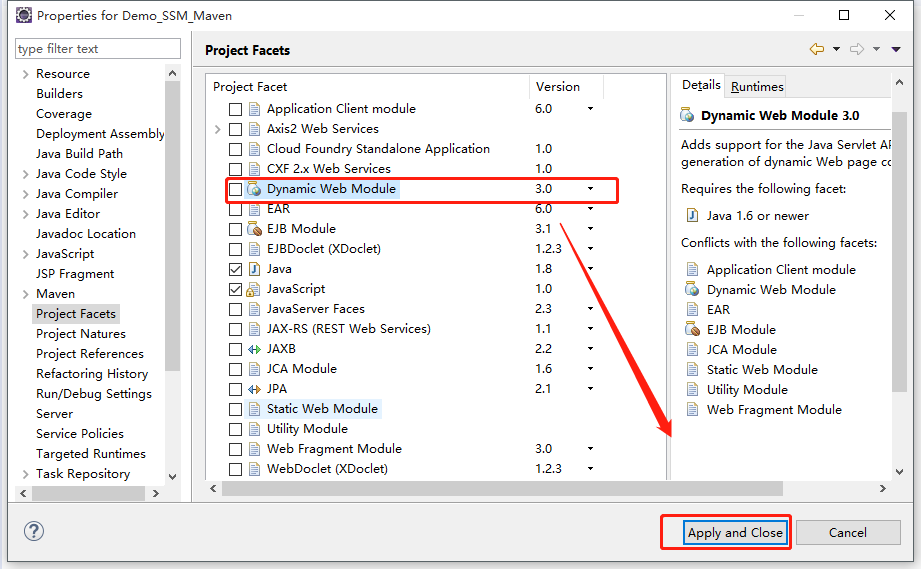
**方法二：**右击项目>点击Properties >点击Project Facets(项目模板)，如下图，可以看到Dynamic Web Module版本为 2.3



直接修改会出现错误

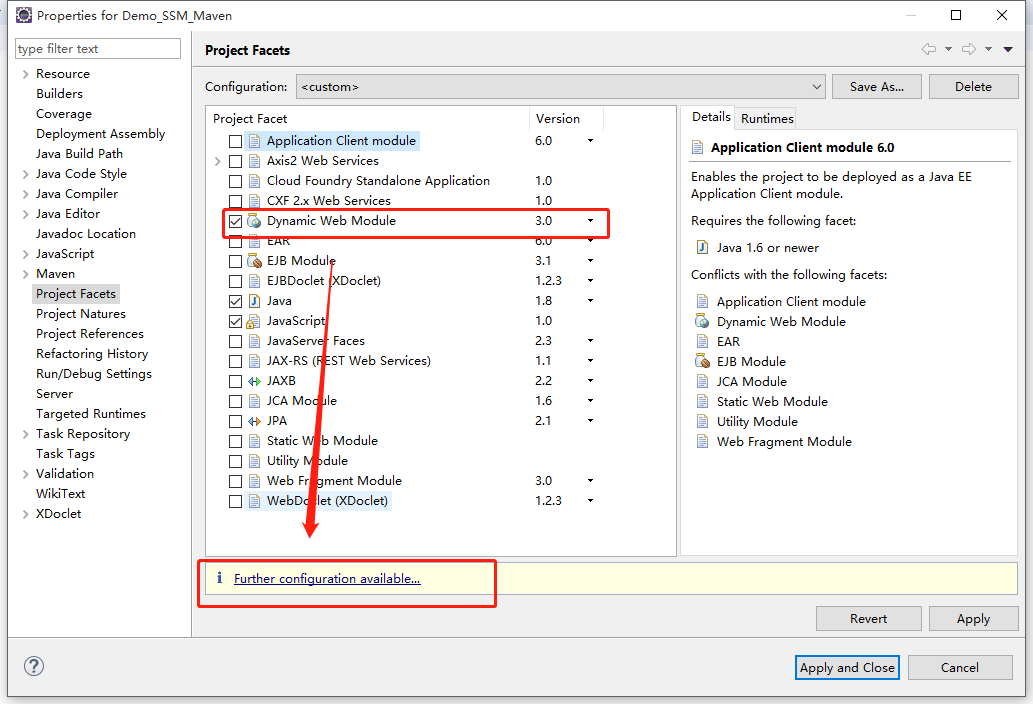


先取消Dynamic Web Module前面的勾选并选择3.0，然后Apple and Close

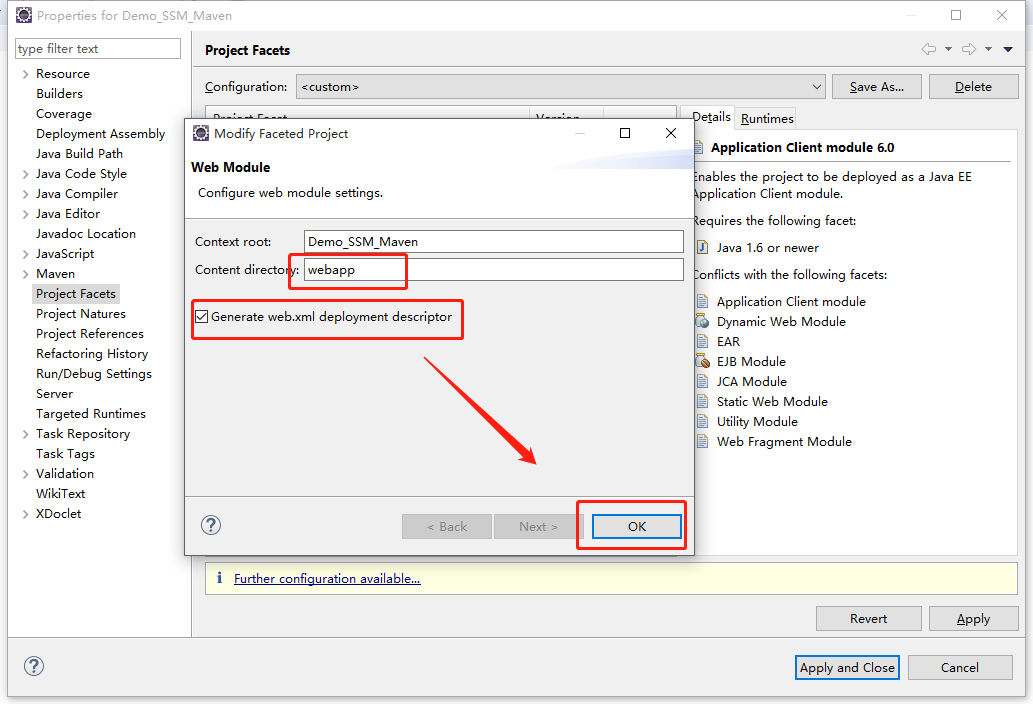


重新打开这个页面，对Dynamic Web Module进行勾选。

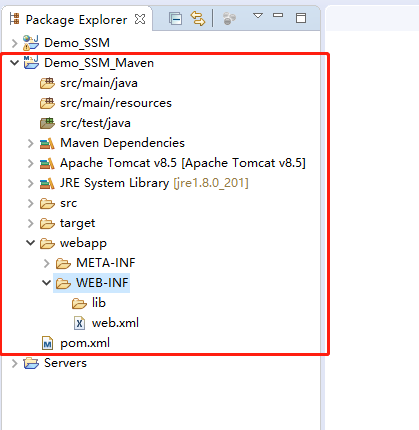
这时候先不要着急保存关闭。点击下面的Furter configuration avail…



修改Content directory为webapp，并勾选下面，如果保存保存保存



最后的项目结构是这样的



此时的web.xml是这样的

<?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>Demo\_SSM\_Maven</display-name>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

# 二、搭建SpringMVC

**Maven项目搭建好了以后，就开始搭建SpringMVC框架**

**1.pom.xml文件中添加springmvc依赖**

<!-- springMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.14.RELEASE</version>

</dependency>

**2.创建springmvc配置文件**

在src/main/resource下创建spring-mvc.xml,内容如下

<?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"*

xmlns:util=*"http://www.springframework.org/schema/util"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd*

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.0.xsd"*>

<!-- 开启注解 -->

<mvc:annotation-driven />

<!-- 让扫描spring扫描这个包下所有的类，让标注spring注解的类生效 -->

<context:component-scan base-package=*"com.ryx.demo.controller"*></context:component-scan>

<!-- 视图解析器 -->

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/views/"* />

<property name=*"suffix"* value=*".jsp"*></property>

</bean>

</beans>

**3.在web.xml中添加配置**

<!-- 定义前端控制器springmvc -->

<servlet>

<servlet-name>spring-mvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- 指定路径 -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mvc.xml</param-value>

</init-param>

<!-- 随spring启动而启动 -->

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring-mvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

**4．创建一个controller测试SpringMVC框架(可以使用PostMan接口测试工具进行测试)**

@RequestMapping("/demo")

@Controller

public class studentController {

@RequestMapping("/test")

@ResponseBody

public String test() {

return "hello,word";

}

}

# 三、搭建Spring

**SpringMVC框架搭建成功后，就开始搭建Spring框架**

**由于在依赖springmvc的时候已经添加了许多spring相关包了，所以此时不需要添加额外的包，可以直接创建配置文件了。**

1. **创建spring-context.xml配置文件**

在src/main/resource下创建spring-context.xml，内容如下

<?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"*

xmlns:util=*"http://www.springframework.org/schema/util"*

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-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd*

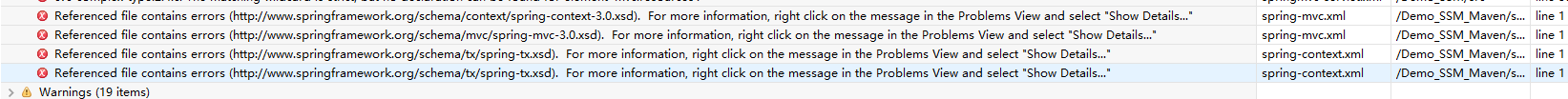
*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd*

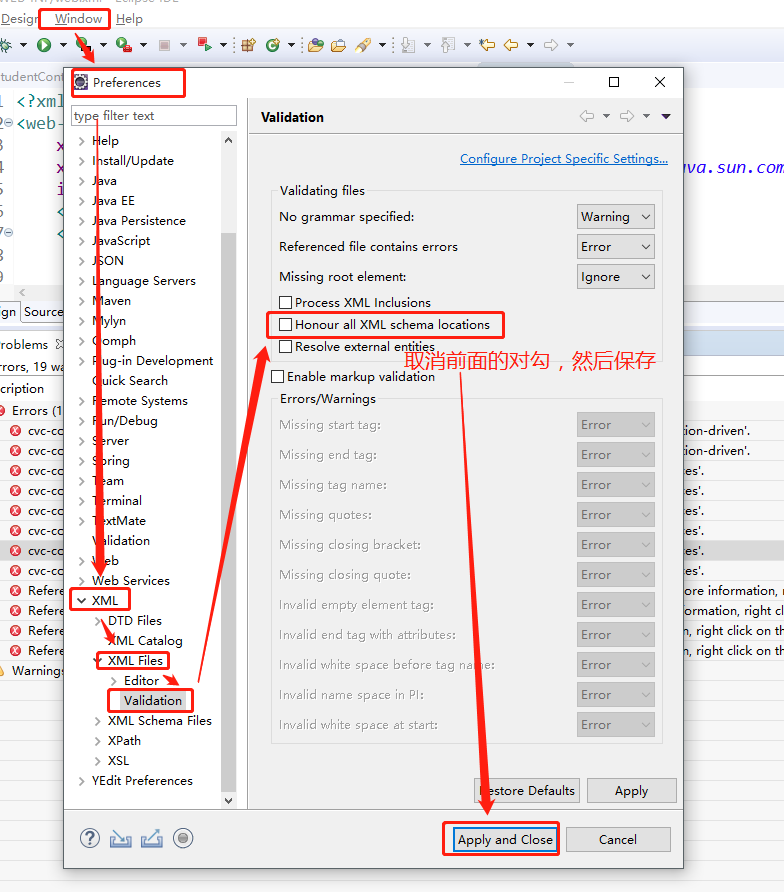
*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd"*>

</beans>

温馨提示：如果xml文件头文件报错 ，并显示一下错误



**解决方式**：window > Prefereces > XML > XML Files > Validation > 取消掉**Honour all XML schema locations** 前面的对勾



**2. web.xml中配置spring**

<!-- 配置适配器spring -->

<listener>

<description>启动spring容器</description>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-context.xml</param-value>

</context-param>

**3.pom.xml中配置 数据源+数据库 的依赖jar包(我用的数据源是C3P0)**

<!-- 数据库连接 //start -->

<!-- c3p0 数据库连接池 -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5.2</version>

</dependency>

<!-- 数据库 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>6.0.6</version>

</dependency>

<!-- 数据库连接 //end -->

**4.在spring-context.xml配置c3p0数据源**

<!-- 配置c3p0 -->

<!-- 连接池 -->

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*></property>

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/demo?characterEncoding=utf8&amp;serverTimezone=UTC"*></property>

<property name=*"user"* value=*"root"*></property>

<property name=*"password"* value=*"123456"*></property>

<property name=*"minPoolSize"* value=*"1"*></property>

<property name=*"maxPoolSize"* value=*"5"*></property>

<property name=*"initialPoolSize"* value=*"1"*></property>

<property name=*"acquireIncrement"* value=*"1"*></property>

</bean>

**5．配置spring声明式事务管理，pom.xml中添加 声明式事务的依赖jar包**

<!-- spring声明式事务管理 //start -->

<!-- spring-tx -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>4.3.14.RELEASE</version>

</dependency>

<!-- spring-jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.14.RELEASE</version>

</dependency>

<!-- spring声明式事务管理 //end -->

**6.在spring-context.xml配置事务管理器**

<!-- 配置spring声明式事务管理 -->

<!-- 配置事务管理器 -->

<bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

<!-- 使用注解来控制事务 -->

<tx:annotation-driven transaction-manager=*"transactionManager"*></tx:annotation-driven>

# 四、搭建Mybatis

**1.pom.xml中添加Mybatis的依赖jar包**

<!-- Mybatis //start -->

<!-- mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.1</version>

</dependency>

<!-- mybatis-spring//无缝整合mybatis+spring -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.0</version>

</dependency>

<!-- mybatis分页插件pagehelper -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>5.1.2</version>

</dependency>

<!-- cglib代理 -->

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib</artifactId>

<version>3.2.2</version>

</dependency>

<!-- Mybatis //end -->

**1.创建mybatis-config.xml配置文件**

在src/main/resource下创建mybatis-config.xml，内容如下

<?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=*"cacheEnabled"* value=*"true"* />

<setting name=*"defaultStatementTimeout"* value=*"3000"* />

<setting name=*"mapUnderscoreToCamelCase"* value=*"true"* />

<!-- 代理 -->

<setting name=*"proxyFactory"* value=*"CGLIB"* />

<setting name=*"lazyLoadingEnabled"* value=*"true"* />

</settings>

<!-- 分页插件 -->

<plugins>

<plugin interceptor=*"com.github.pagehelper.PageInterceptor"*>

<!-- 该参数默认为false 设置为true时，会将RowBounds第一个参数offset当成pageNum页码使用和startPage中的pageNum效果一样 -->

<property name=*"offsetAsPageNum"* value=*"true"* />

<!-- 该参数默认为false 设置为true是，使用RowBounds分页会进行count查询 -->

<property name=*"rowBoundsWithCount"* value=*"true"* />

<!-- 设置为true时,如果pageSize=0或ROwRounds.limit=0就会查询出全部的结果(相当于每一偶执行分页查询，但是返回结果仍然是page类型) -->

<property name=*"pageSizeZero"* value=*"true"* />

</plugin>

</plugins>

</configuration>

**2.** **spring-context.xml中配置mybatis**

<!-- 配置mybatis, 绑定c3p0 -->

<bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

<property name=*"configLocation"* value=*"classpath:mybatis-config.xml"*></property>

<property name=*"mapperLocations"*>

<list>

<value>classpath:mapper/\*.xml</value>

</list>

</property>

</bean>

<!-- 扫描生成所有dao层 -->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<property name=*"basePackage"* value=*"com.bigname.demo03.dao"*></property>

<property name=*"sqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*></property>

</bean>

# 五、创建entity、controller、servcie . . . .class

。

**自行解决**