# 基础知识

###### 框架概述

###### 1.SSM(Spring+SpringMVC+MyBatis) 框架由Spring、SpringMVC、MyBatis三个开源框架整合而成，作为数据源较简单的web项目的框架

###### 2.Spring

###### 一个轻量级的控制反转(IOC) 和面向切面（AOP）的容器框架

###### 3.SpringMVC

###### 分离了控制器、模型对象、分配器以及处理程序对象的角色，这种分离让它们更容器进行定制

###### 4.MyBatis

###### 一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架

###### 5.MVC设计思想

###### 表现层：html+css+Jquery+ajax

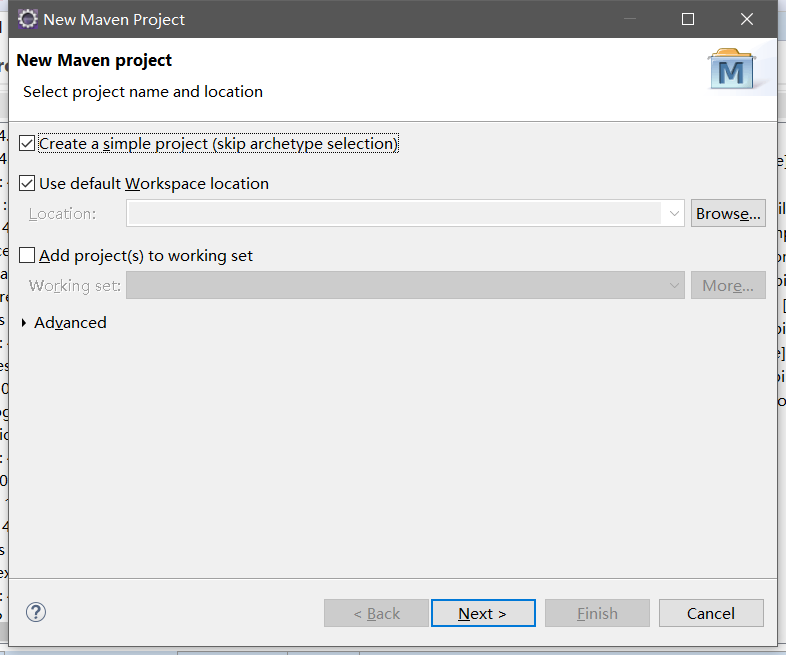
###### 控制层：springmvc

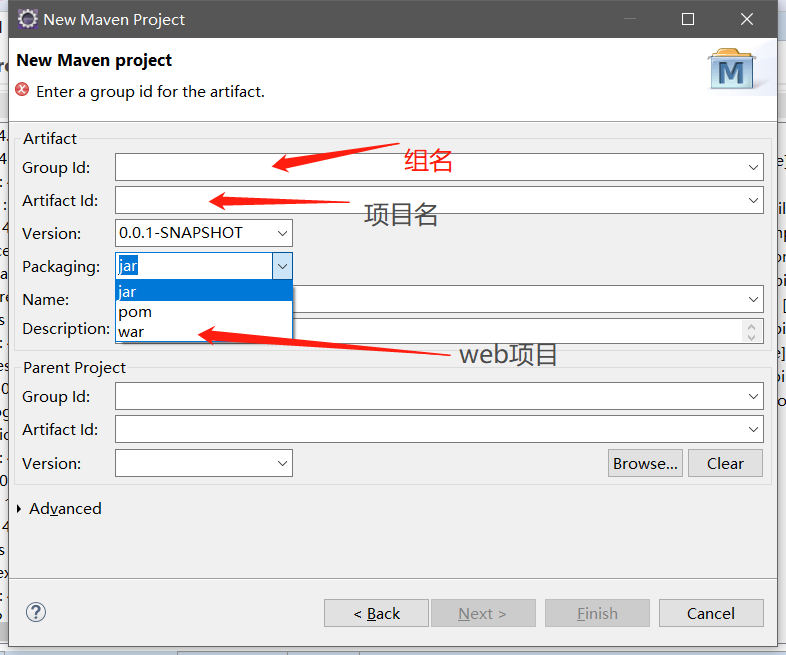
###### 业务层：service组件

###### 持久层：dao组件

# 2.项目搭建

###### 创建Maven工程





# 3.导入jar包

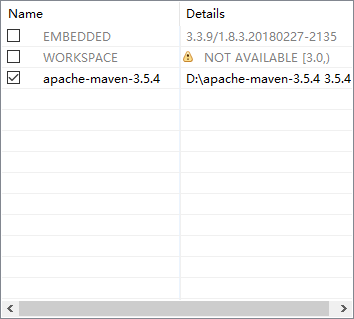
导包方式有两种：一种是ecplise直接选择或下载jar包，自动生成xml配置

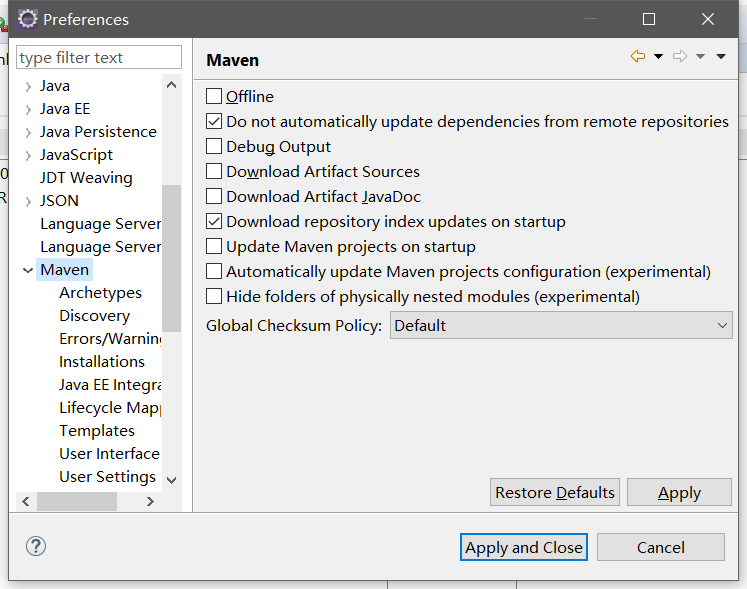
另一种就是复制黏贴xml配置，当然手打也行，然后ecplise会根据xml的配置自动下载jar包

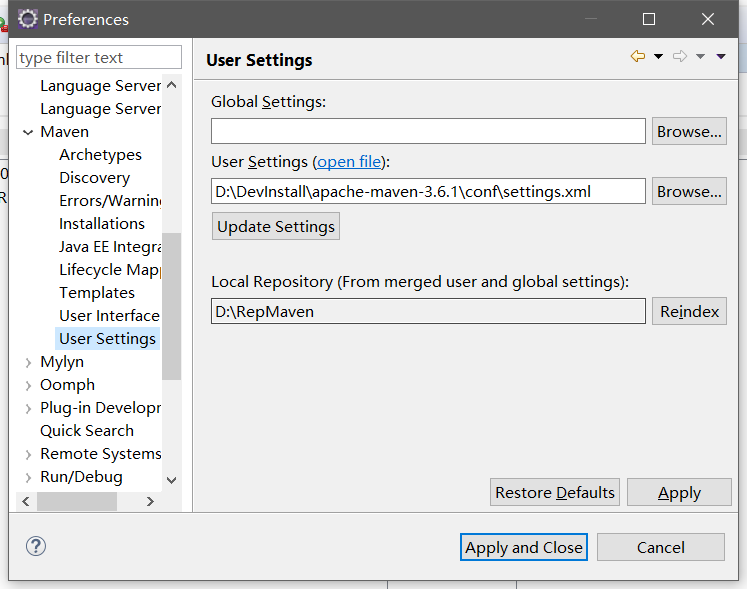
这里我用的是第一种

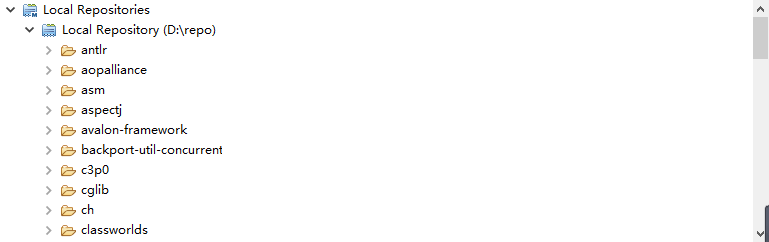
遇到的问题：由于配置完maven，配置成功了，但是搜索不到本地库的jar包

配置成功，也能在本地库里看见本地jar包，但是搜索不到









试过了重新建立了索引

在eclipse中打开菜单 window-> show view –> other –> Maven -> maven repositories

打开之后，选择local repositories –> local repository ，右击，选择Build index

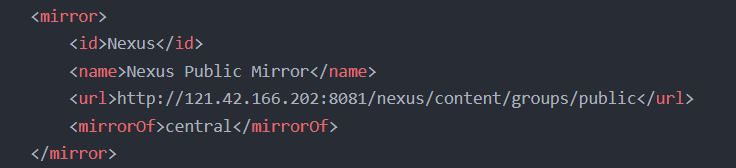
还是搜索不到

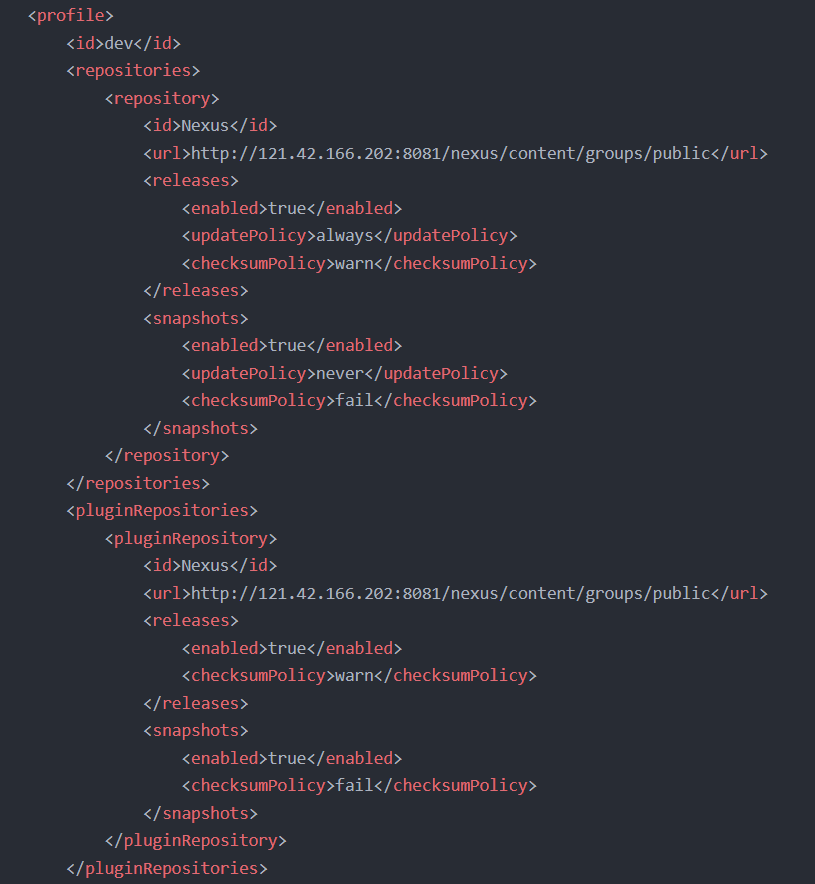
解决方法：

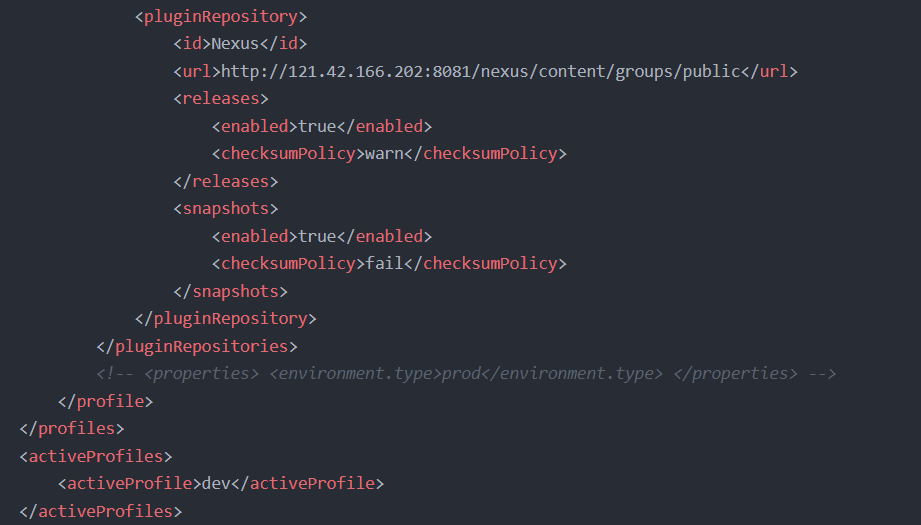
1.发现桌面->用户->.m2文件夹里含有repository，而且里面也有本地jar包，删除了repository文件夹后(网上方法)

2.正想打算放弃本地jar库，添加私服的jar库（阿里私服）网上找的方法

在maven->conf->setting.xml添加私服



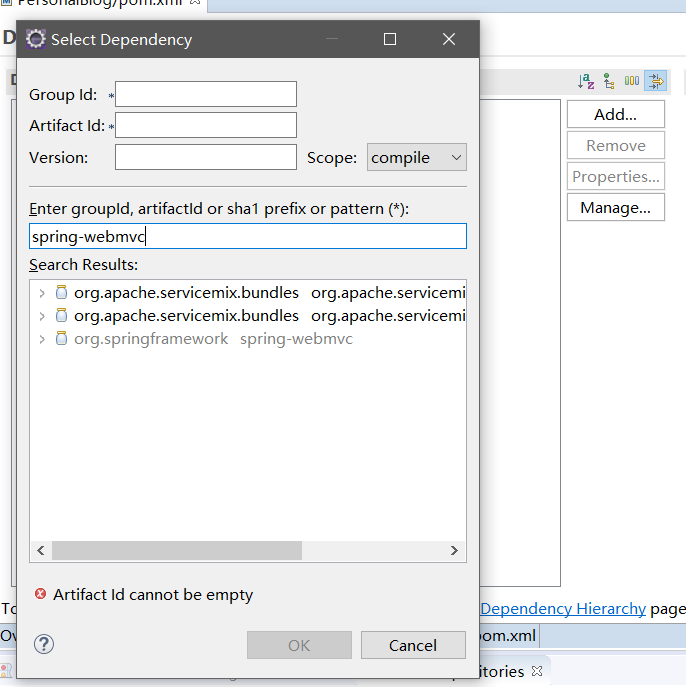




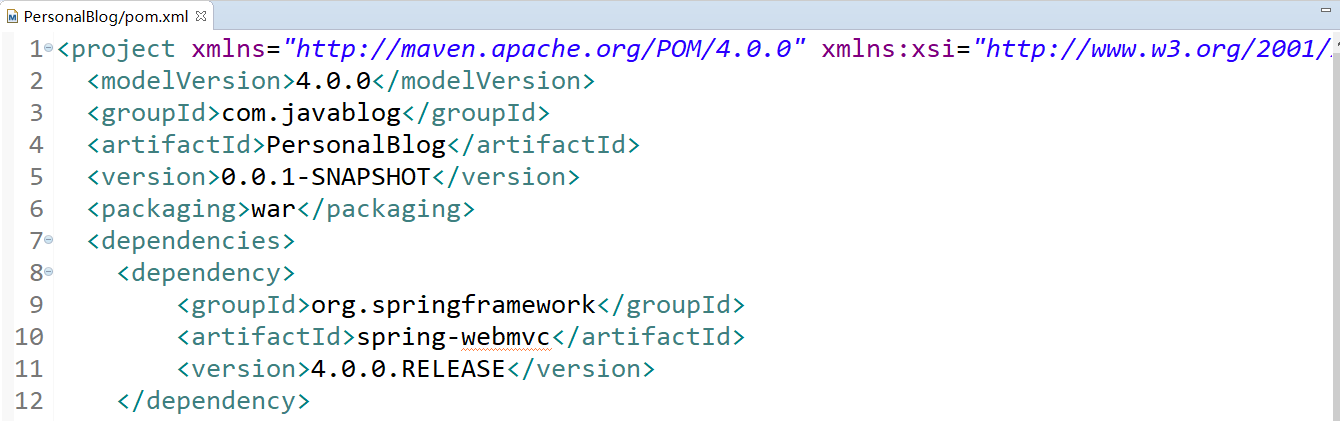
添加完之后发现既能搜索本地库也能搜索私服，搞定（应该是第一种解决方法1解决搜索本地库找不到的问题）

###### 1.springMVC

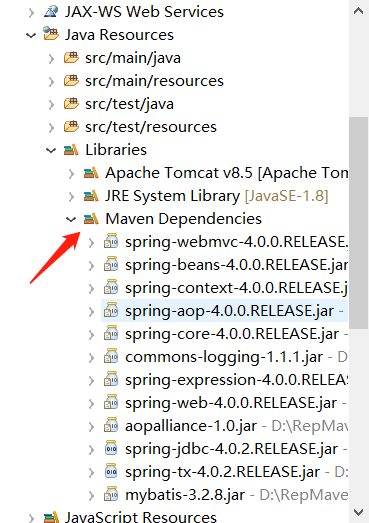
###### 



导完之后pom.xml自动添加导完jar包的信息配置



在项目里也自动导入了spring-webMVC的相关jar包(maven的好处之一)



###### 2.spring

###### 

###### 3.mybatis

###### 

###### 4.连接池( commons-dbcp )

###### 

###### 5.数据库驱动 ( 我用的是mysql )

###### 

###### 6.Mybatis-spring(整合spring和mybatis)

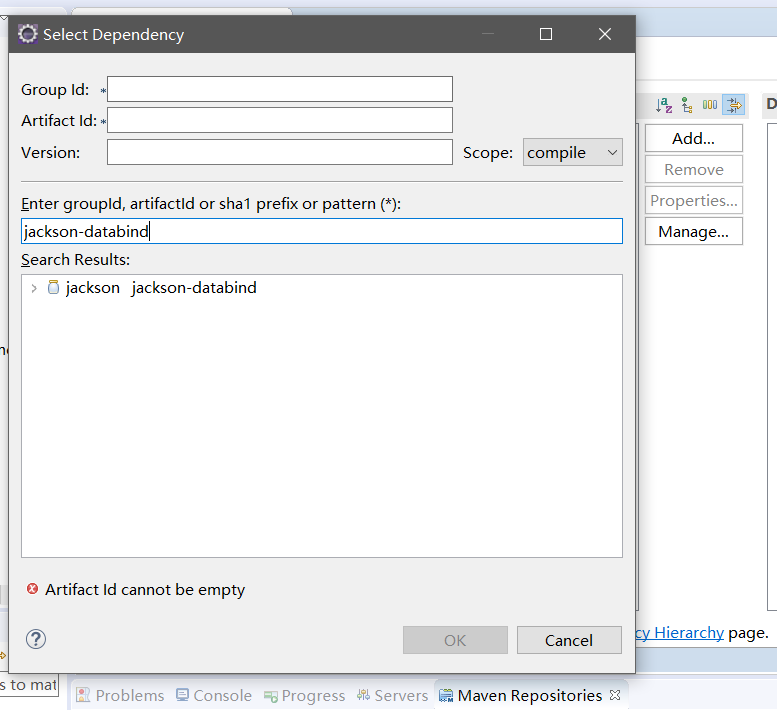
###### 

###### 7.Jstl（jsp开发要用到的jstl标签）

###### 

###### 8.Json(用于返回json数据)

Databind:数据绑定，序列化



# 4.构建基本目录

###### 1.包目录

###### 

View:视图层

Service：业务层

Dao：与数据库打交道，持久层

Controller：控制层

###### 2.后台页面目录

在web-inf里面是安全的，无法直接访问

###### 

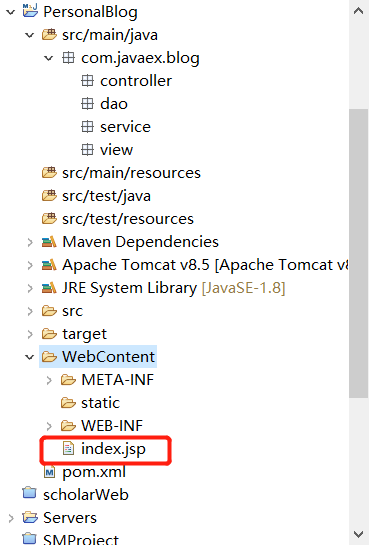
###### 静态资源目录

建在WebContent（Webapp）下，可以直接访问

###### 

###### 默认首页

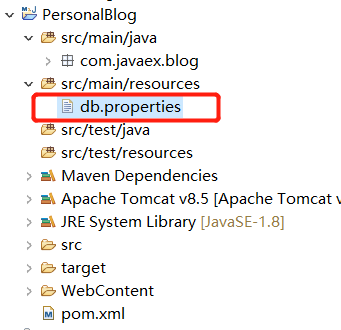
创建默认首页，建在WebContent（Webapp）下，可以直接访问



# 5.文件配置

###### 数据库

在src\main\resources目录下创建数据库连接配置文件db.properties



填写配置内容

注意：oracle、mysql、sqlserver三者的写法都不相同

变量名可以随便取，但是要注意，不能和系统内置的变量名起冲突

等号右边的内容，不需要加双引号或单引号

**SqlServer配置**

driver = com.microsoft.sqlserver.jdbc.SQLServerDriver

url = jdbc:sqlserver://数据库地址(一般写ip地址); DatabaseName=数据库名称

username = 数据库用户名

password = 数据库密码

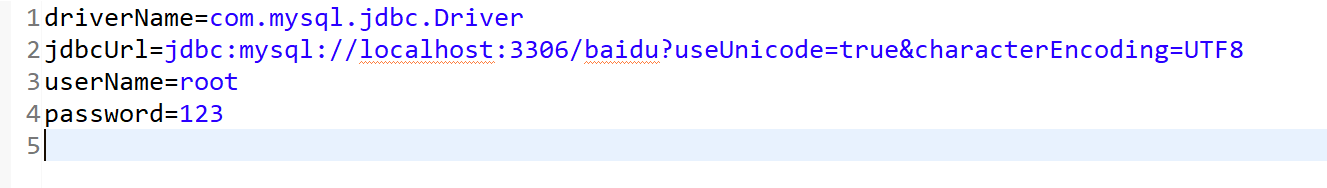
**MySql配置**

driverName=com.mysql.jdbc.Driver

jdbcUrl=jdbc:mysql://数据库地址(一般写ip地址)/数据库名useUnicode=true&characterEncoding=UTF8

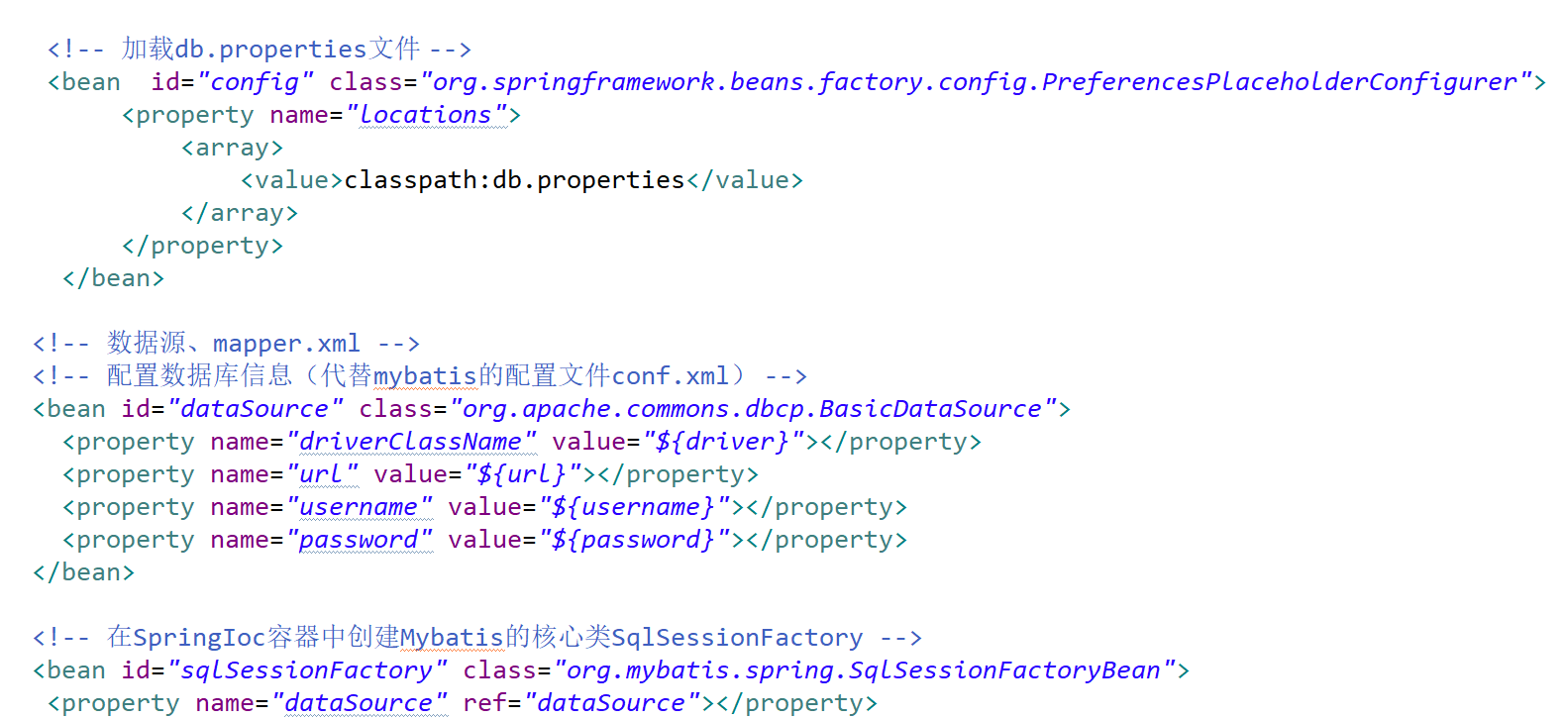
userName=数据库用户名

password=数据库密码



###### spring-mybatis

在src\main\resources目录下创建spring-mybatis整合配置文件 spring-mybatis.xml





**懒人模式：**

<?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:jdbc=*"http://www.springframework.org/schema/jdbc"*

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

xmlns:mybatis-spring=*"http://mybatis.org/schema/mybatis-spring"*

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

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

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

xsi:schemaLocation=*"http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

*http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd*

*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-4.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>

<!-- 加载db.properties文件 -->

<bean id=*"config"* class=*"org.springframework.beans.factory.config.PreferencesPlaceholderConfigurer"*>

<property name=*"locations"*>

<array>

<value>classpath:db.properties</value>

</array>

</property>

</bean>

<!-- 数据源、mapper.xml -->

<!-- 配置数据库信息（代替mybatis的配置文件conf.xml） -->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*>

<property name=*"driverClassName"* value=*"${driver}"*></property>

<property name=*"url"* value=*"${url}"*></property>

<property name=*"username"* value=*"${username}"*></property>

<property name=*"password"* value=*"${password}"*></property>

</bean>

<!-- 在SpringIoc容器中创建Mybatis的核心类SqlSessionFactory -->

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

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

<property name=*"mapperLocations"* value=*"classpath:com/javaex/blog/dao/\*.xml"*></property>

</bean>

<!-- 第三种方式生存mapper对象(批量产生多个mapper)

批量产生Mapper对在SpringIoc中的id值默认就是首字母小写接口名(首字母小写接口名=id值)

-->

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

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

<!-- 指定批量产生哪个包的mapper对象 -->

<property name=*"basePackage"* value=*"com.javaex.blog.dao"*></property>

<!-- 上面basePackage所在的property的作用：

将com.mapper包中，所有的接口 产生与之对应的动态代理对象

（对象名就是首字母小写的接口名）: studentMapper.方法-->

</bean>

<!-- 开启事务注解驱动 -->

<tx:annotation-driven/>

<!-- （事务管理） -->

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

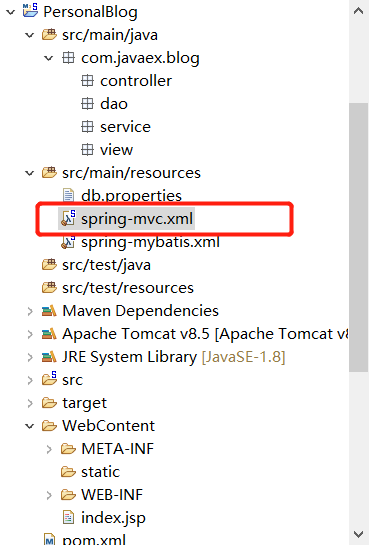
<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

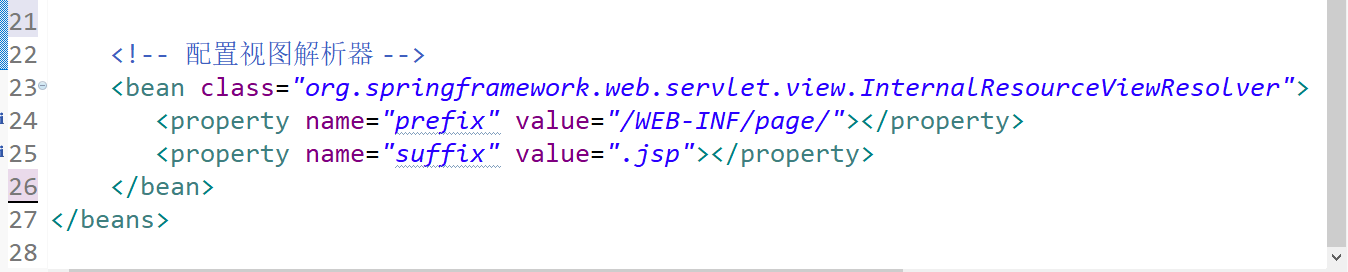
</beans>

###### spring-mvc

在src\main\resources目录下创建spring-mvc整合配置文件 spring-mvc.xml







**懒人模式：**

<?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: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-4.3.xsd*

*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-4.3.xsd"*>

<!-- 将控制器所在包 加入IOC容器 -->

<!-- 开启组件扫描 -->

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

<!-- SpringMVC基础配置、标配 -->

<!-- 启用注解驱动 -->

<mvc:annotation-driven></mvc:annotation-driven>

<!-- 处理静态资源 -->

<mvc:default-servlet-handler/>

<!-- 配置视图解析器 -->

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

<property name=*"prefix"* value=*"/WEB-INF/page/"*></property>

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

</bean>

</beans>

###### web.xml

配置web.xml





**懒人模式：**

<?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\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>blog</display-name>

<welcome-file-list>

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

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

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

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

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

</welcome-file-list>

<!-- 配置静态资源文件路径 -->

<!-- <servlet-mapping>

<servlet-name>default</servlet-name>

<url-pattern>/static/\*</url-pattern>

</servlet-mapping> -->

<!-- spring mvc请求响应 -->

<servlet>

<servlet-name>SpringMVC</servlet-name>

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

<init-param>

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

<param-value>classpath:spring-\*.xml</param-value>

</init-param>

</servlet>

<!-- 请求页面跳转 -->

<servlet-mapping>

<servlet-name>SpringMVC</servlet-name>

<url-pattern>\*.action</url-pattern>

</servlet-mapping>

<!-- 请求数据 -->

<servlet-mapping>

<servlet-name>SpringMVC</servlet-name>

<url-pattern>\*.json</url-pattern>

</servlet-mapping>

</web-app>

# 6.访问测试

###### 准备数据库数据

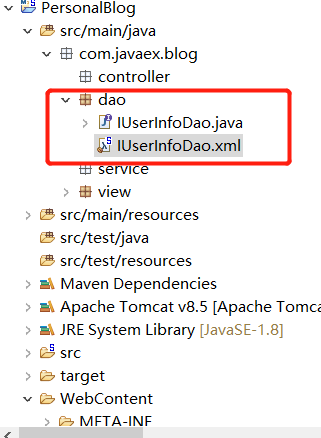
创建数据库和表和添加一条测试数据

###### 实体类

创建实体类的时候要注意，实体类名等于表名，实体类属性名等于数据库里的字段名，实体类属性类型等于数据库里的字段类型

###### dao层

创建Dao层模块功能接口和数据库方法xml文件



**IUserInfoDAO.java**

**写接口函数，如果有2个以上的参数，则必须给参数加@Param参数注解**

**package** com.javaex.blog.dao;

**import** org.apache.ibatis.annotations.Param;

**import** com.javaex.blog.view.User;

**public** **interface** IUserInfoDao {

/\*\*

\* **@param** loginName 登录名

\* **@param** passWord 登录密码

\*/

**public** User checkUser(@Param("loginName") String loginName,@Param("passWord") String passWord);

}

IUserInfoDAO.xml

用来写sql语句，实现dao层接口，sql的id对应接口的函数名

<?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.javaex.blog.dao.user\_info.IUserInfoDao"*>

<!-- 建立sql查询结果字段与实体属性的映射关系 -->

<resultMap id=*"UserInfoMap"* type=*"com.javaex.blog.view.UserInfo"*>

<result column=*"id"* property=*"id"*/>

<result column=*"login\_name"* property=*"loginName"*/>

<result column=*"pass\_word"* property=*"passWord"*/>

</resultMap>

<!-- 校验用户 -->

<select id=*"checkUser"* resultMap=*"UserInfoMap"*>

SELECT \* FROM user\_info WHERE 1=1 (这里要改哟，下面两个if就是提示，改完删了下面两句if)

<if test=*"loginName !=null and loginName != ''"*>AND login\_name=#{loginName}</if>

<if test=*"passWord !=null and passWord != ''"*>AND pass\_word=#{passWord}</if>

</select>

</mapper>

###### service层

###### 

**package** com.javaex.blog.service.user\_info;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** com.javaex.blog.dao.user\_info.IUserInfoDao;

**import** com.javaex.blog.view.UserInfo;

@Service(“UserInfoService”)

**public** **class** UserInfoService {

@Autowired

**private** IUserInfoDao iUserInfoDao;

/\*\*

\* 校验用户登录

\* **@param** loginName 登录名

\* **@param** passWord 登录密码

\* **@return**

\*/

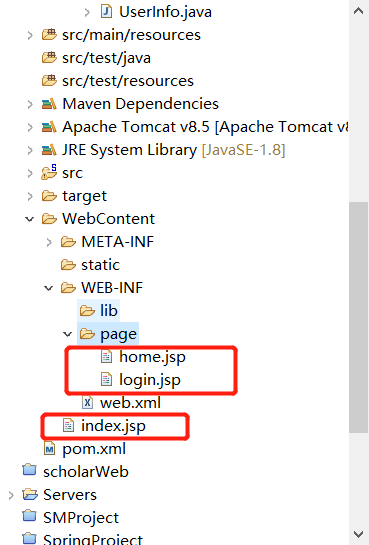
**public** UserInfo checkUser(String loginName,String passWord) {

**return** iUserInfoDao.checkUser(loginName, passWord);

}

**｝**

###### 准备页面



**Index.jsp(默认首页)**

**这个页面啥也不干，直接发送请求到后台，并跳转到登录页**

<%

pageContext.setAttribute("APP\_PATH", request.getContextPath());

%>

<script>

window.location.href ="${APP\_PATH}/user\_info/index.action";

</script>

**Login.jsp（登录页）**

<form id=*"loginForm"* action=*"login.action"* method=*"post"*>

登录名:<input type=*"text"* name=*"login\_name"*/>

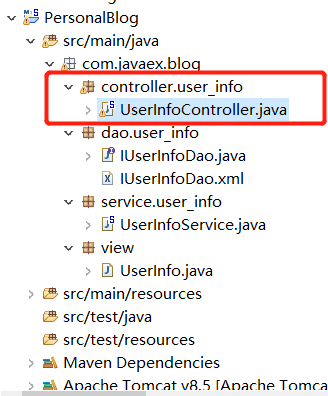
密码:<input type=*"password"* name=*"pass\_word"*/>

<input type=*"submit"* value=*"登录"*/>

</form>

Home.jsp(主页)

###### 控制层



**package** com.javaex.blog.controller.user\_info;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.javaex.blog.service.user\_info.UserInfoService;

**import** com.javaex.blog.view.UserInfo;

**import** com.mysql.jdbc.StringUtils;

@Controller

@RequestMapping("user\_info")

**public** **class** UserInfoController {

@Autowired

**private** UserInfoService userInfoService;

/\*\*

\* 首页跳转登录页面

\*/

@RequestMapping("index.action")

**public** String index() {

**return** "login";

}

/\*

\* 用户登录

\* @param loginName 登录名

\* @param passWord 登录密码

\*/

@RequestMapping("login.action")

**public** String login(ModelMap map,@RequestParam(required = **false** , value = "login\_name") String loginName,

@RequestParam(required = **false** , value ="pass\_word") String passWord) {

//如果登录名或密码未填写，直接返回登录页面

**if** (StringUtils.isEmpty(loginName)||StringUtils.isEmpty(passWord)) {

**return** "login";

}

//校验用户名，密码是否正确

UserInfo userInfo = userInfoService.checktUser(loginName, passWord);

**if** (userInfo==**null**) {

**return** "login";

}

//登录成功，进入主页

**return** "home";

}

}

###### 7.访问