1.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

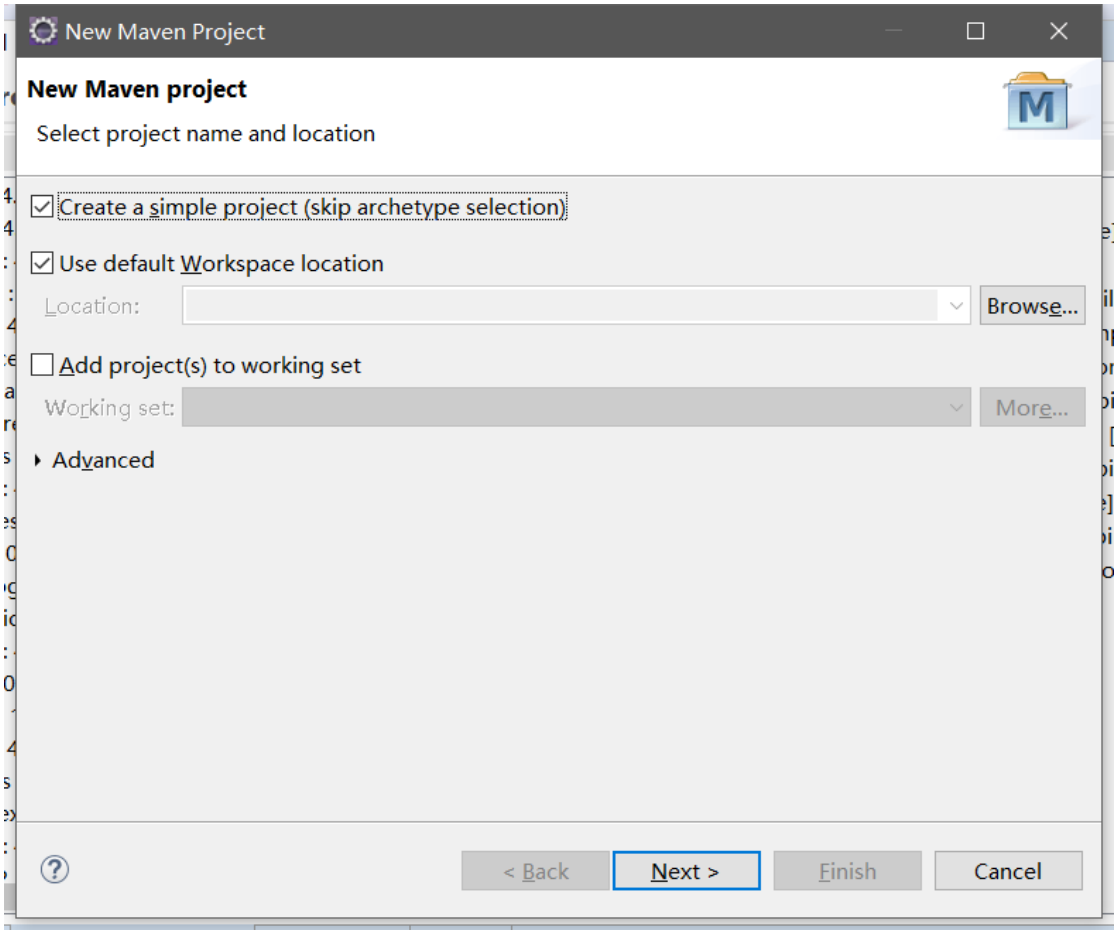
控制层：springmvc

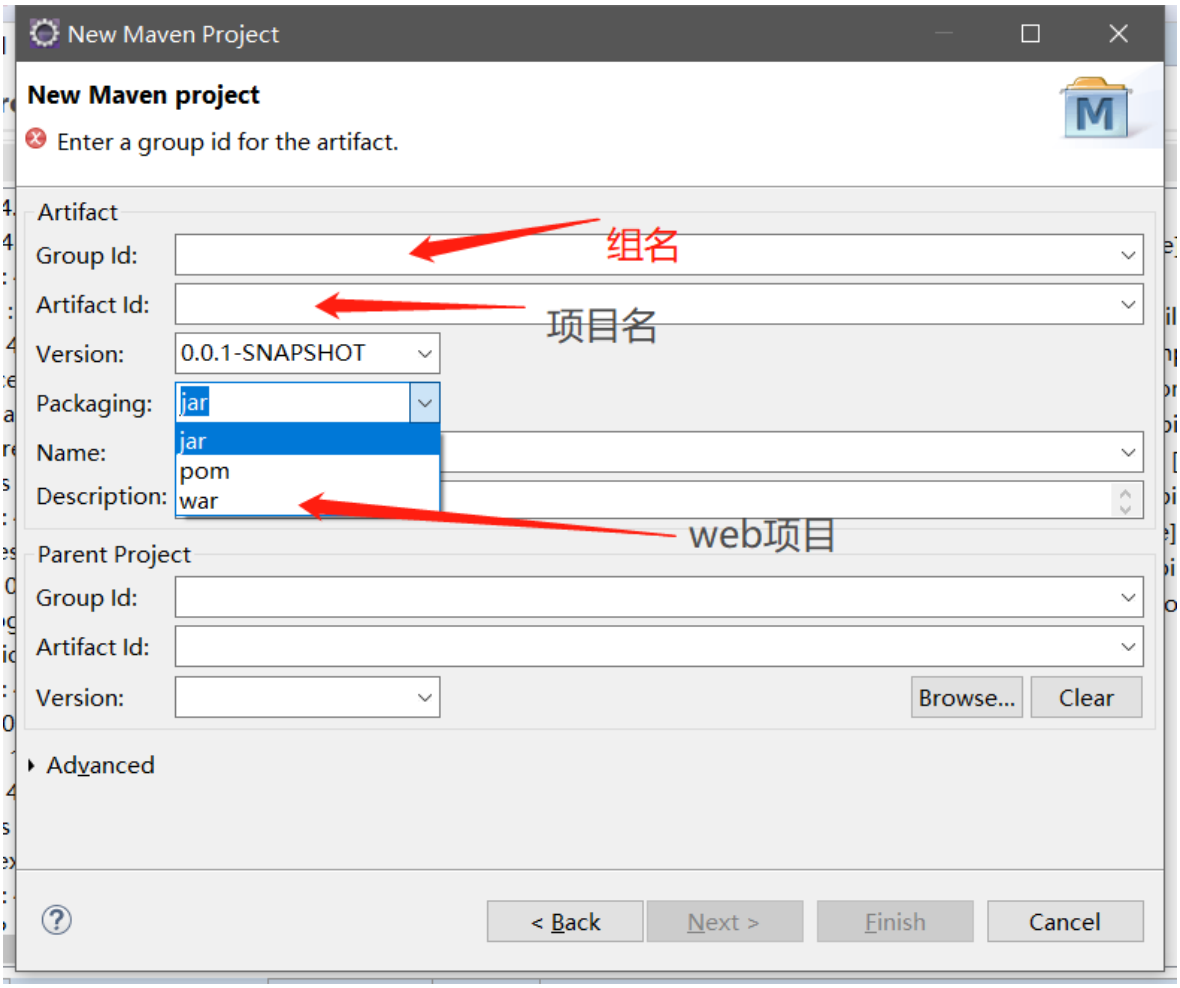
业务层：service 组件

持久层：dao 组件

2.项目搭建

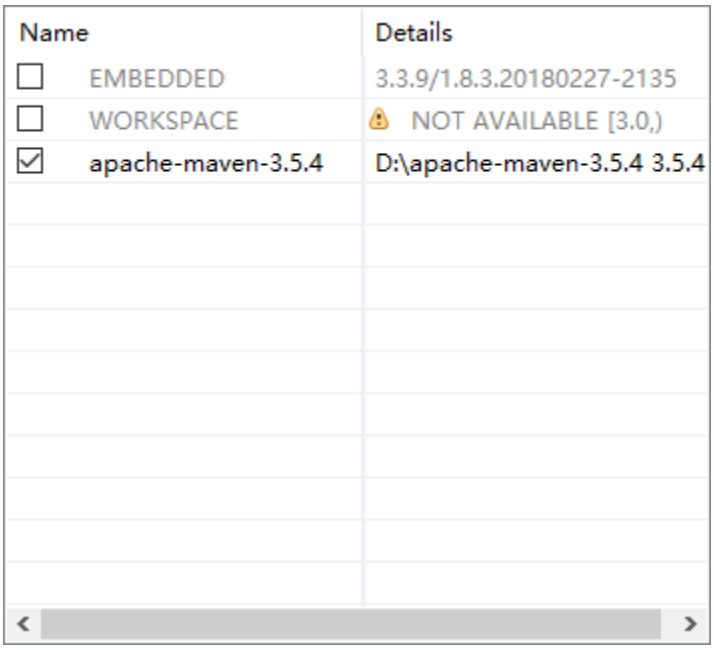
1. 创建 Maven 工程

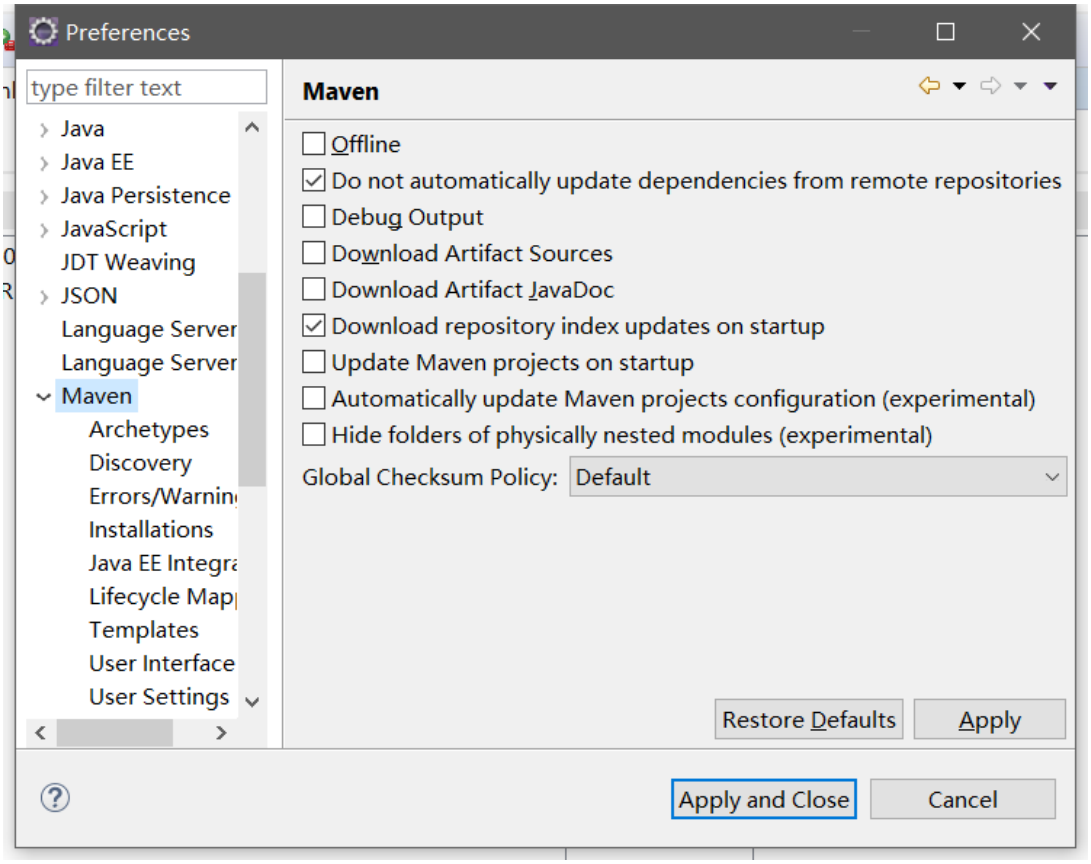


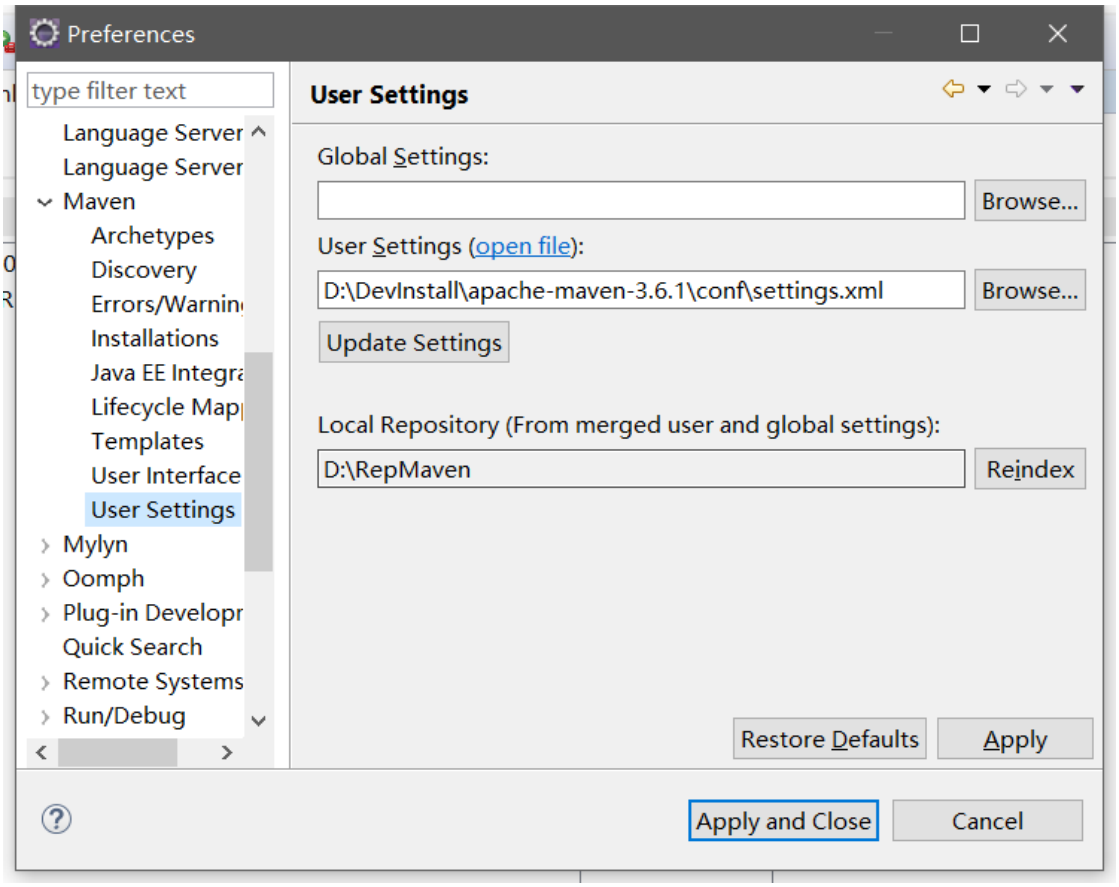


3.导入 jar 包

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


Name
Details
EMBEDDED
3.3.9/1.8.3.20180227-2135
WORKSPACE
NOT AVAILABLE [3.0,)
apache-maven-3.5.4
D:\apache-maven-3.5.4 3.5.4


Preferences
type filter text
Java
Java EE
Java Persistence
JavaScript
JDT Weaving
JSON
Language Server
Language Server
Maven
Archetypes
Discovery
Errors/Warnin
Installations
Java EE Integra
Lifecycle Map
Templates
User Interface
User Settings
Maven
Offline
Do not automatically update dependencies from remote repositories
Debug Output
Download Artifact Sources
Download Artifact JavaDoc
Download repository index updates on startup
Update Maven projects on startup
Automatically update Maven projects configuration (experimental)
Hide folders of physically nested modules (experimental)
Global Checksum Policy:
Default
Restore Defaults
Apply
Apply and Close
Cancel



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

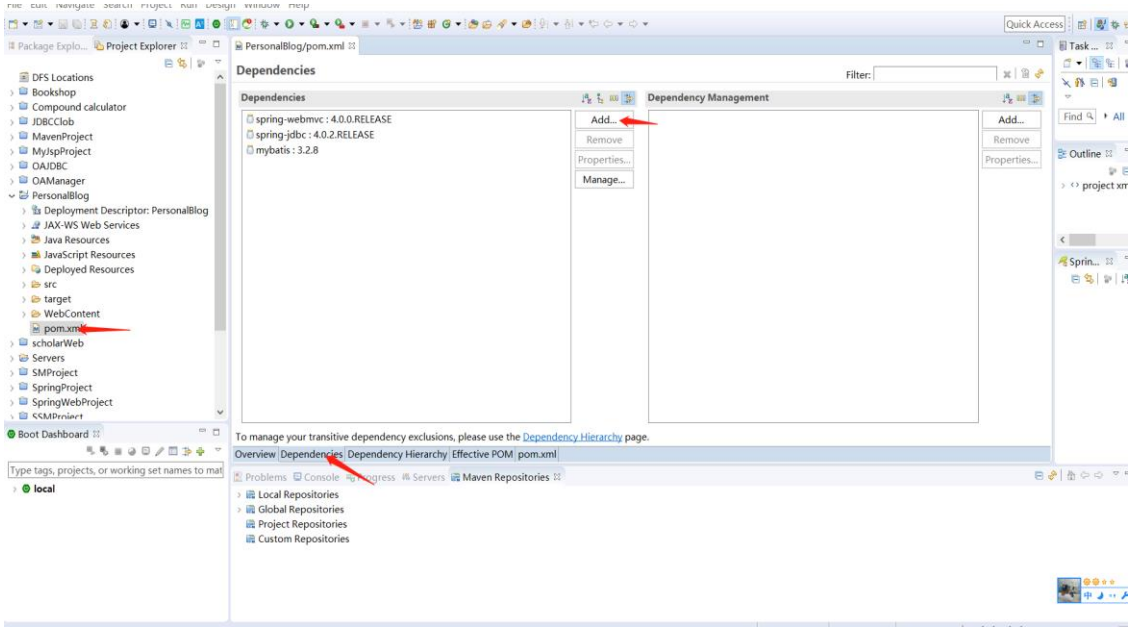
```
<mirror>
    <id>Nexus</id>
    <name>Nexus Public Mirror</name>
    <url>http://121.42.166.202:8081/nexus/content/groups/public</ur
    <mirrorOf>central</mirrorOf>
</mirror>
```

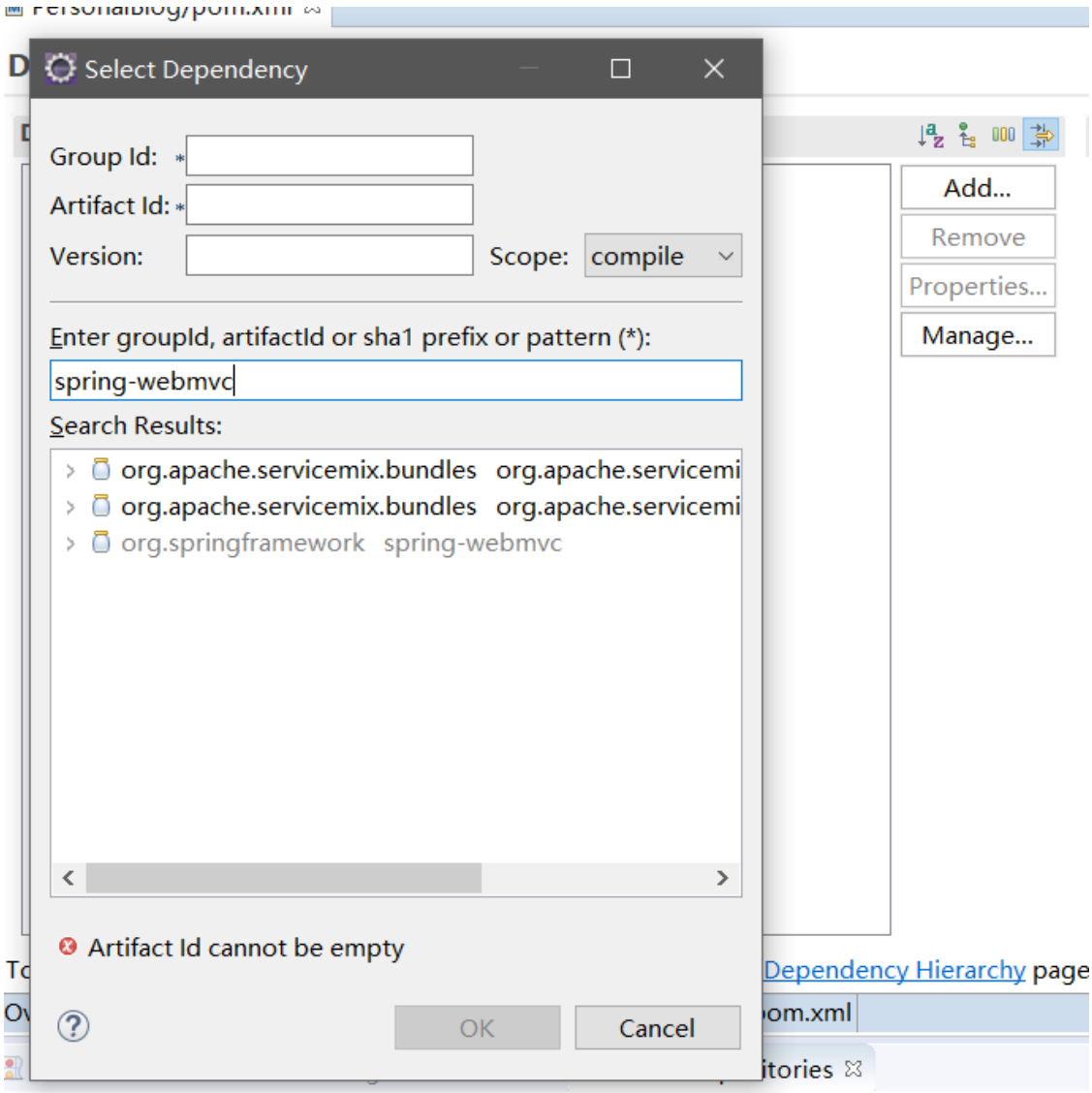
```
<profile>
    <id>dev</id>
    <repositories>
        <repository>
            <id>Nexus</id>
            <url>http://121.42.166.202:8081/nexus/content/groups/public<
            <releases>
                <enabled>true</enabled>
                <updatePolicy>always</updatePolicy>
                <checksumPolicy>warn</checksumPolicy>
            </releases>
            <snapshots>
                <enabled>true</enabled>
                <updatePolicy>never</updatePolicy>
                <checksumPolicy>fail</checksumPolicy>
            </snapshots>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>Nexus</id>
            <url>http://121.42.166.202:8081/nexus/content/groups/public<
            <releases>
                <enabled>true</enabled>
                <checksumPolicy>warn</checksumPolicy>
            </releases>
            <snapshots>
                <enabled>true</enabled>
                <checksumPolicy>fail</checksumPolicy>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
```

```
            <pluginRepository>
                <id>Nexus</id>
                <url>http://121.42.166.202:8081/nexus/content/groups/public</url>
                <releases>
                    <enabled>true</enabled>
                    <checksumPolicy>warn</checksumPolicy>
                </releases>
                <snapshots>
                    <enabled>true</enabled>
                    <checksumPolicy>fail</checksumPolicy>
                </snapshots>
            </pluginRepository>
        </pluginRepositories>
        <!-- <properties> <environment.type>prod</environment.type> </properties> -->
    </profile>
</profiles>
<activeProfiles>
    <activeProfile>dev</activeProfile>
</activeProfiles>
```

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


Project Explorer
DFS Locations
Bookshop
Compound calculator
JDBCClob
MavenProject
MyJspProject
OAJDBC
OAManager
PersonalBlog
Deployment Descriptor: PersonalBlog
JAX-WS Web Services
Java Resources
JavaScript Resources
Deployed Resources
src
target
WebContent
pom.xml
scholarWeb
Servers
SMProject
SpringProject
SpringWebProject
Boot Dashboard
local
PersonalBlog/pom.xml
Dependencies
Filter:
Dependencies
spring-webmvc : 4.0.0.RELEASE
spring-jdbc : 4.0.2.RELEASE
mybatis : 3.2.8
Add...
Remove
Properties...
Manage...
Dependency Management
Add...
Remove
Properties...
To manage your transitive dependency exclusions, please use the Dependency Hierarchy page.
Overview
Dependencies
Dependency Hierarchy
Effective POM
pom.xml
Problems
Console
Servers
Maven Repositories
Local Repositories
Global Repositories
Project Repositories
Custom Repositories
Quick Access
Task...
Outline
Find
All

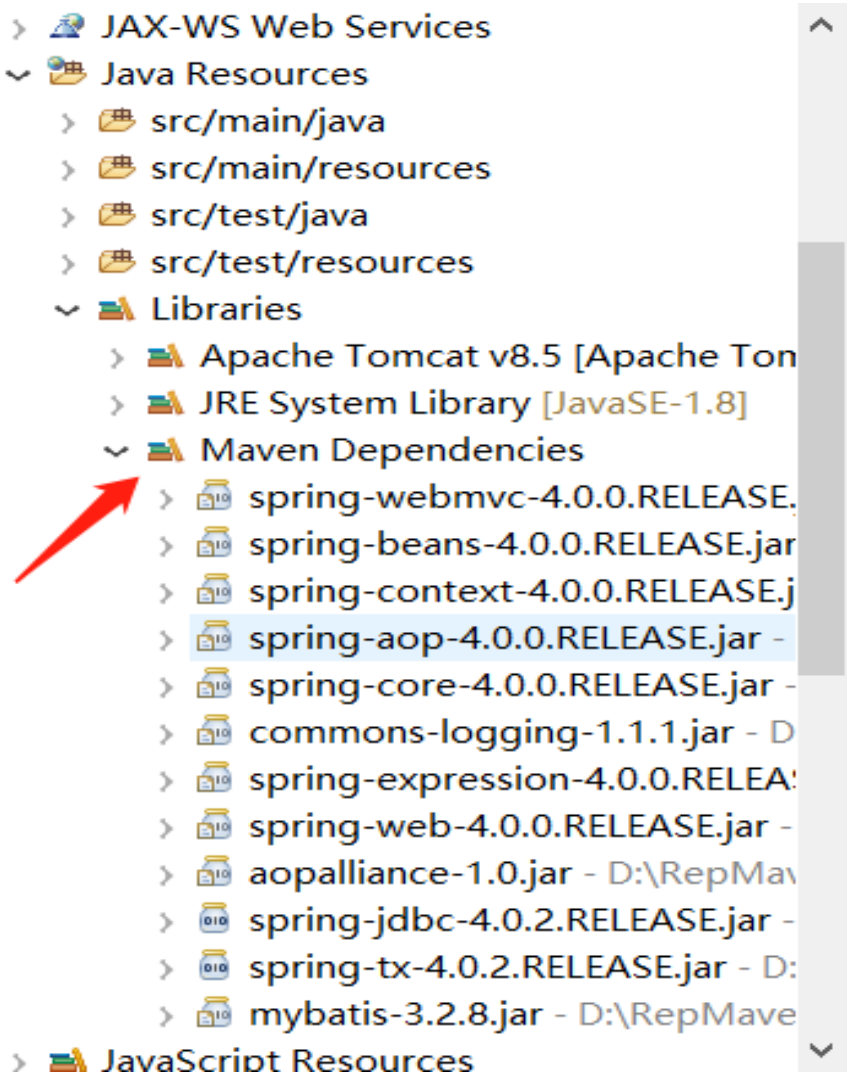


导完之后 pom.xml 自动添加导完 jar 包的信息配置

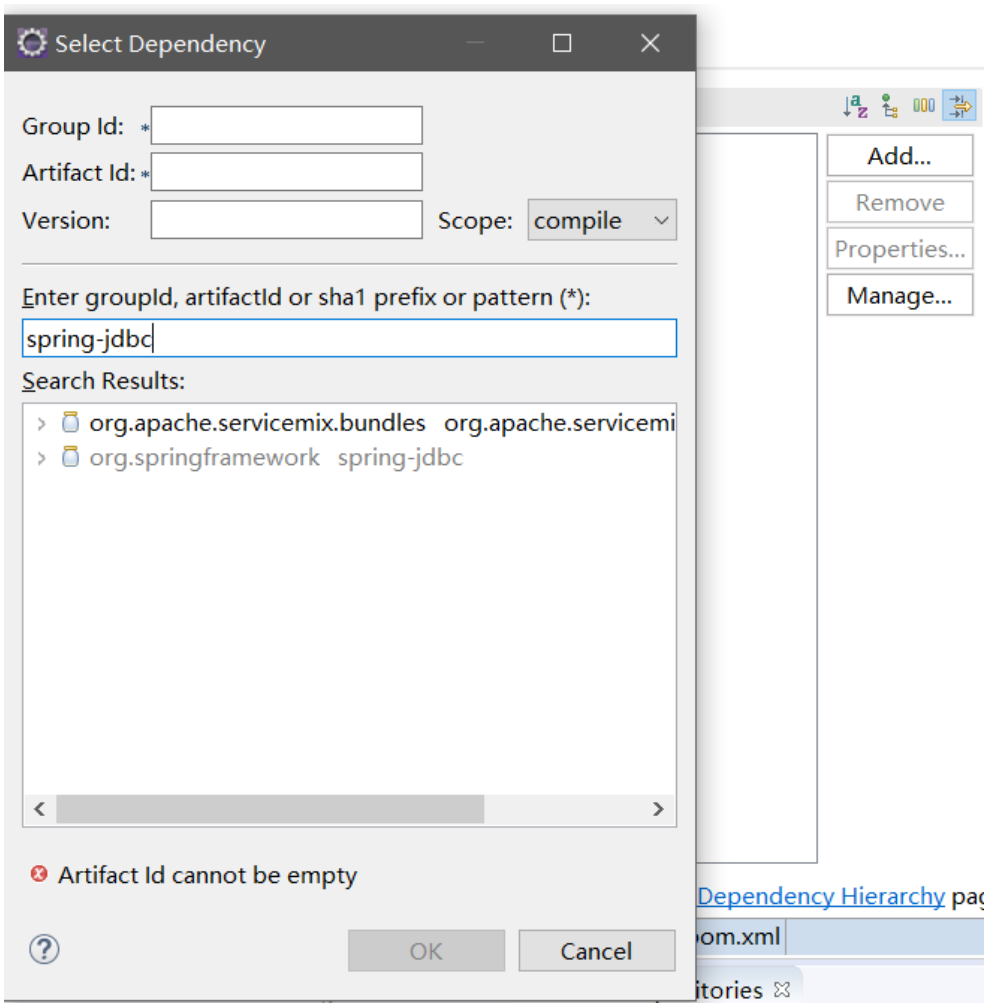
PersonalBlog/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javablog</groupId>
  <artifactId>PersonalBlog</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>4.0.0.RELEASE</version>
    </dependency>
```

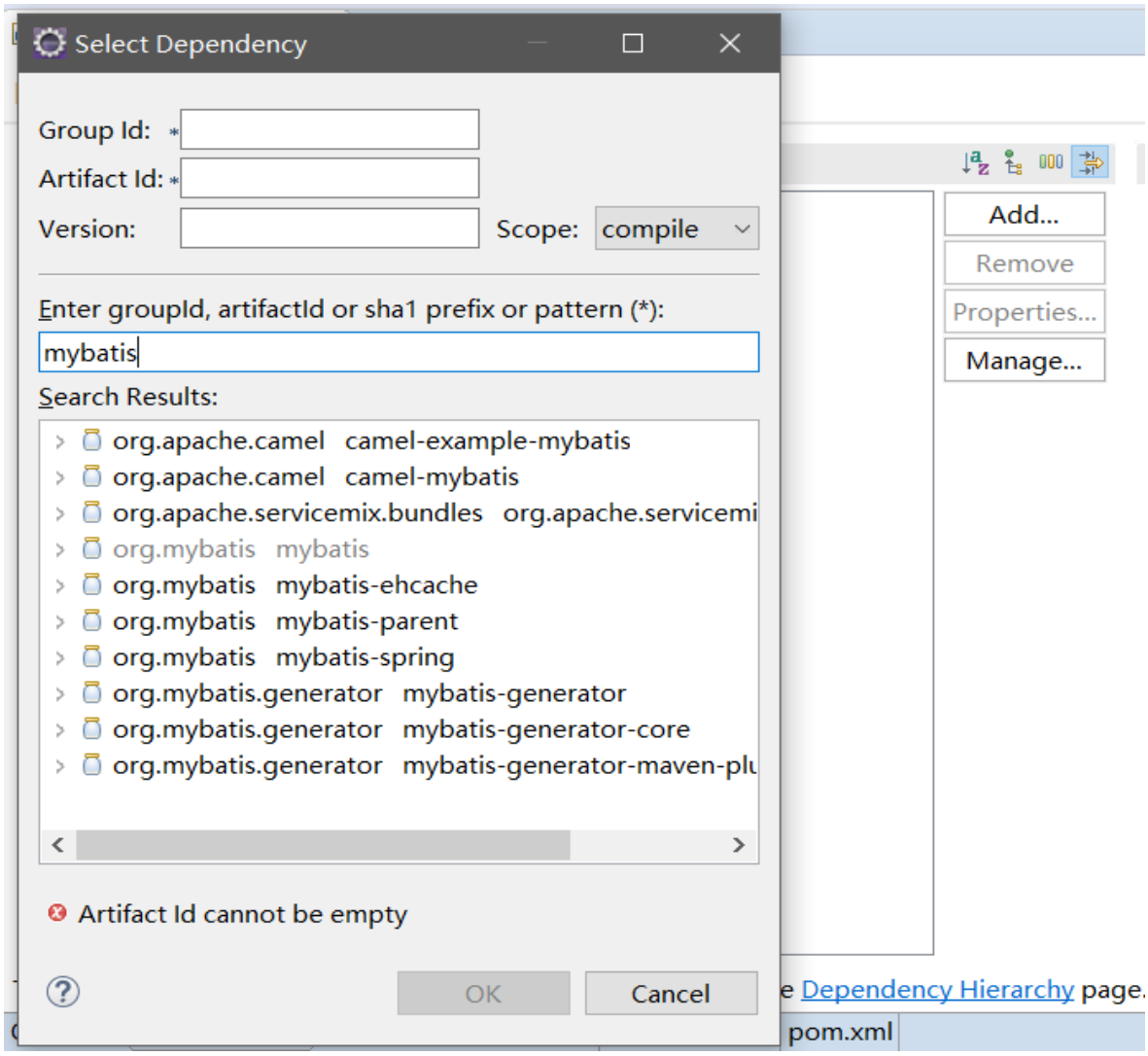
在项目里也自动导入了 spring-webMVC 的相关 jar 包(maven 的好处之一)


JAX-WS Web Services
Java Resources
src/main/java
src/main/resources
src/test/java
src/test/resources
Libraries
Apache Tomcat v8.5 [Apache Ton
JRE System Library [JavaSE-1.8]
Maven Dependencies
spring-webmvc-4.0.0.RELEASE.
spring-beans-4.0.0.RELEASE.jar
spring-context-4.0.0.RELEASE.j
spring-aop-4.0.0.RELEASE.jar -
spring-core-4.0.0.RELEASE.jar -
commons-logging-1.1.1.jar - D
spring-expression-4.0.0.RELEA
spring-web-4.0.0.RELEASE.jar -
aopalliance-1.0.jar - D:\RepMav
spring-jdbc-4.0.2.RELEASE.jar -
spring-tx-4.0.2.RELEASE.jar - D:
mybatis-3.2.8.jar - D:\RepMave
JavaScript Resources

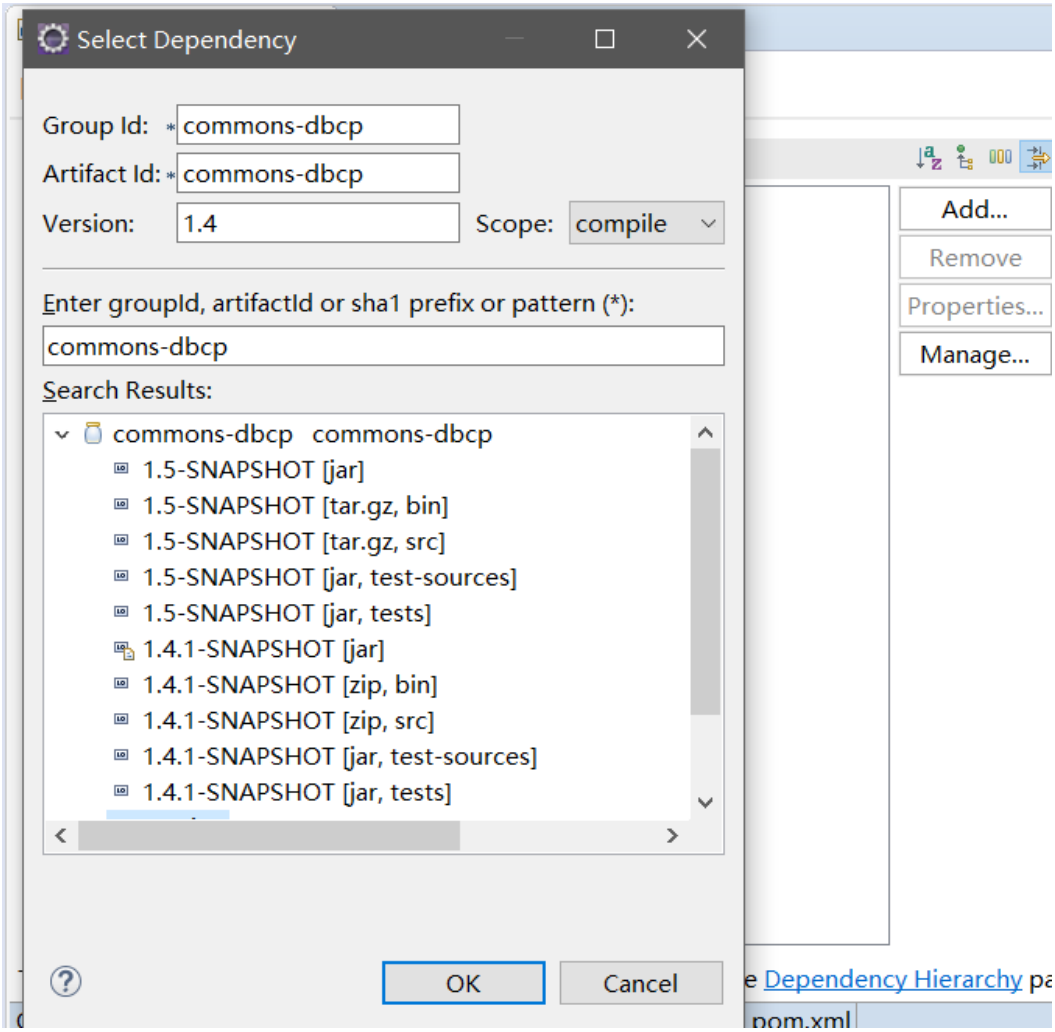
2.spring



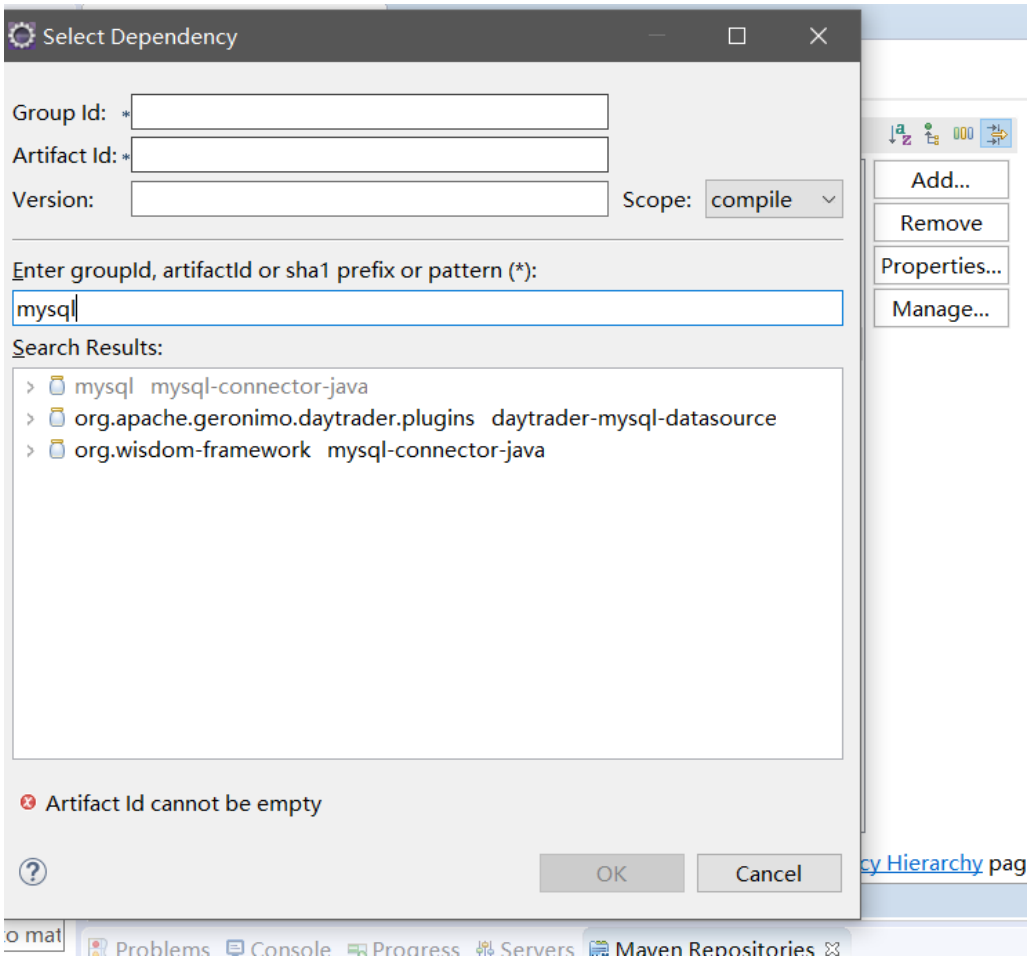
3.mybatis



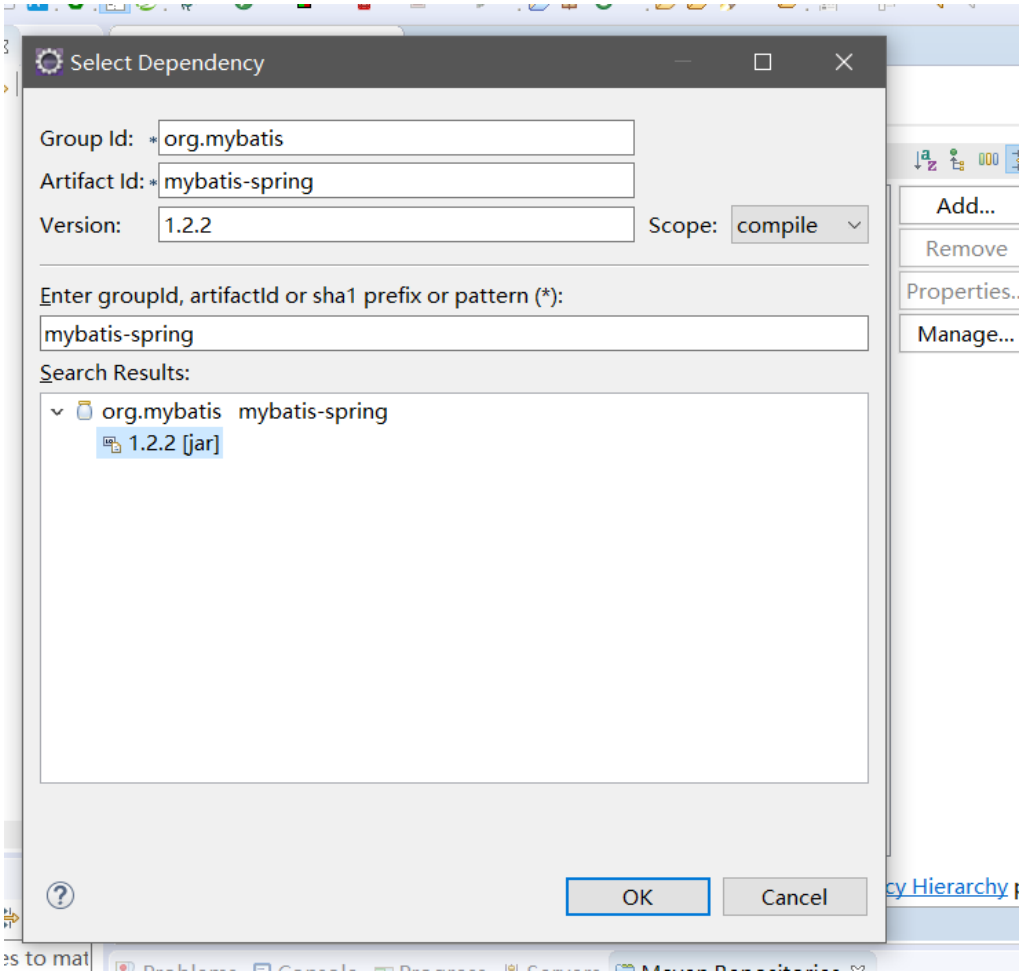
4.连接池(commons-dbcp)



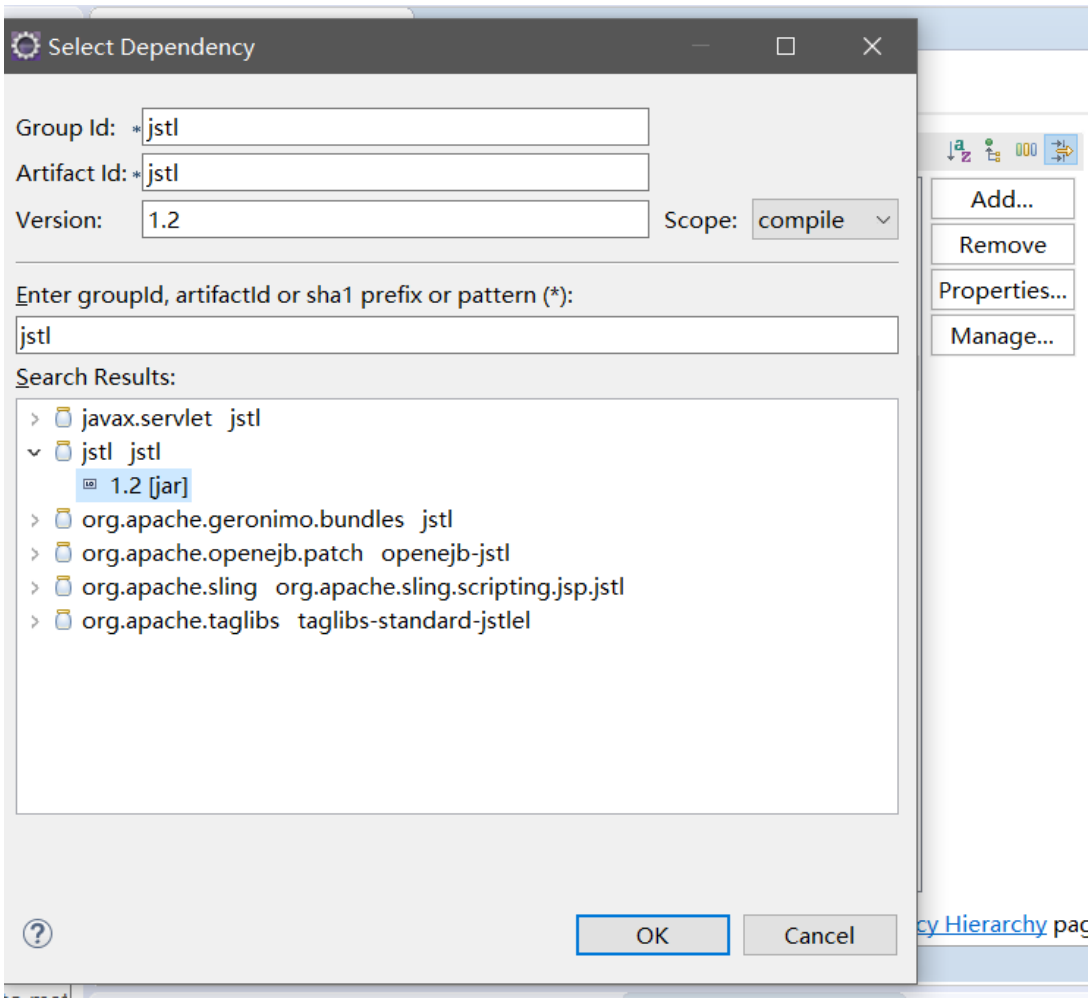
5.数据库驱动（ 我用的是 mysql ）



6.Mybatis-spring(整合 spring 和 mybatis)

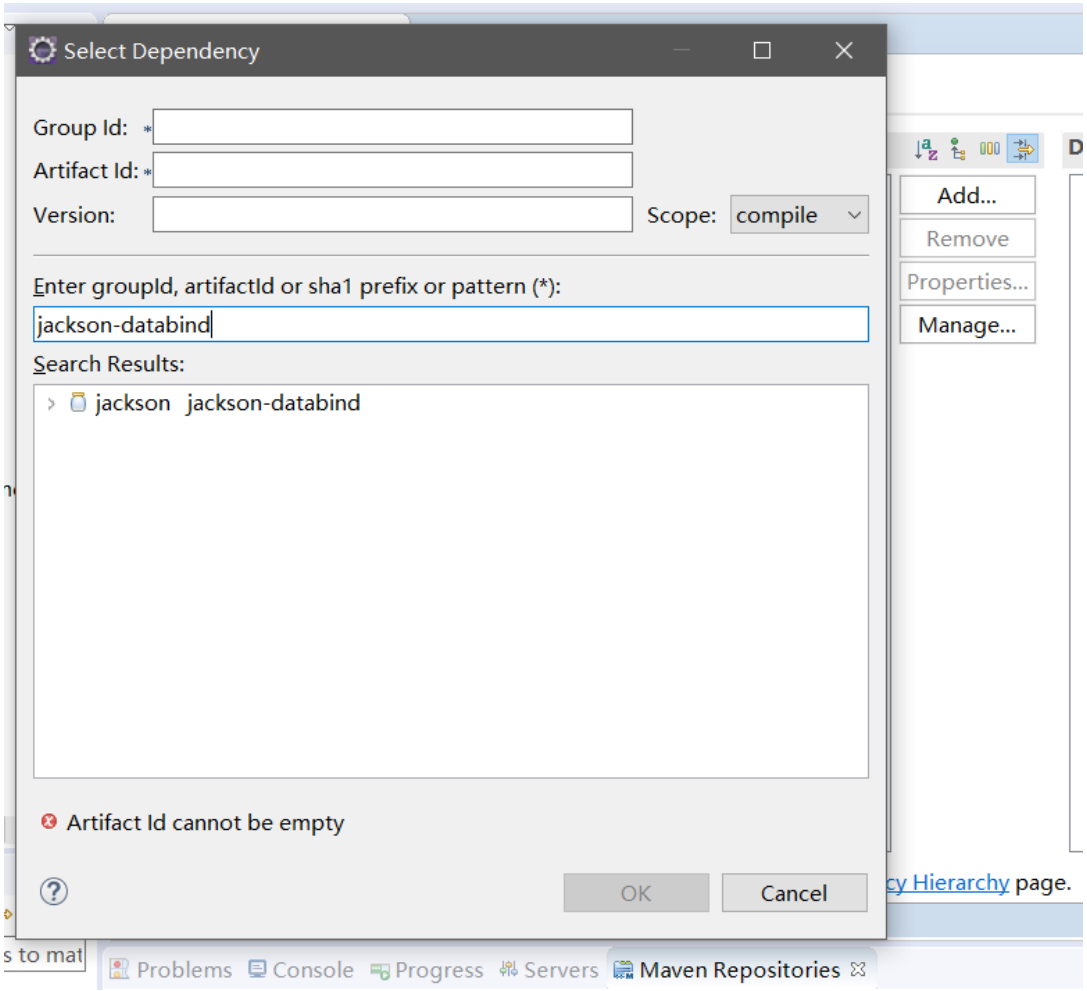


7.Jstl（jsp 开发要用到的 jstl 标签）



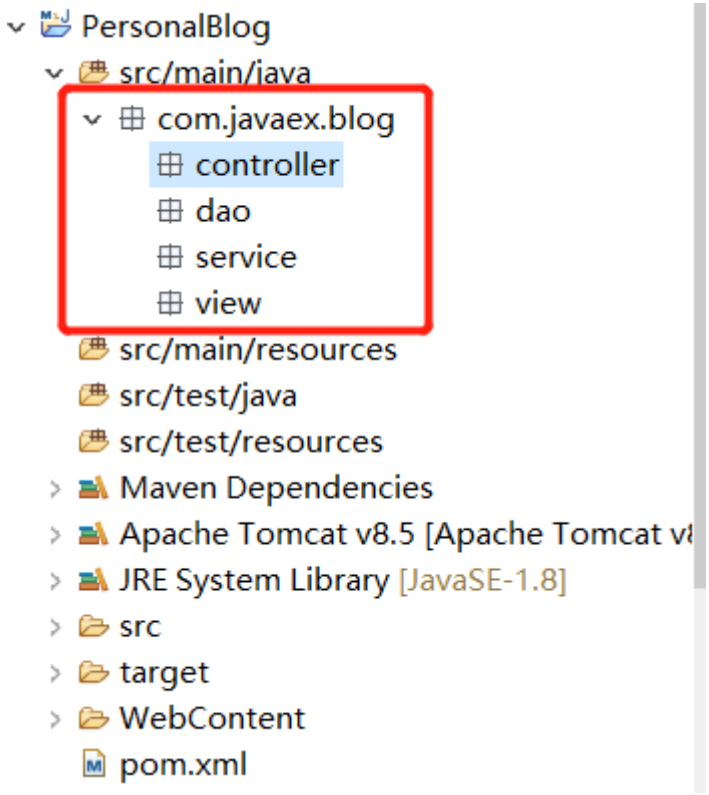
8.Json(用于返回 json 数据)

Databind:数据绑定，序列化


Select Dependency
Group Id: *
Artifact Id: *
Version:
Scope: compile
Enter groupId, artifactId or sha1 prefix or pattern (*):
jackson-databind
Search Results:
jackson jackson-databind
Artifact Id cannot be empty
OK
Cancel
Add...
Remove
Properties...
Manage...
cy Hierarchy page.
Problems
Console
Progress
Servers
Maven Repositories

4.构建基本目录

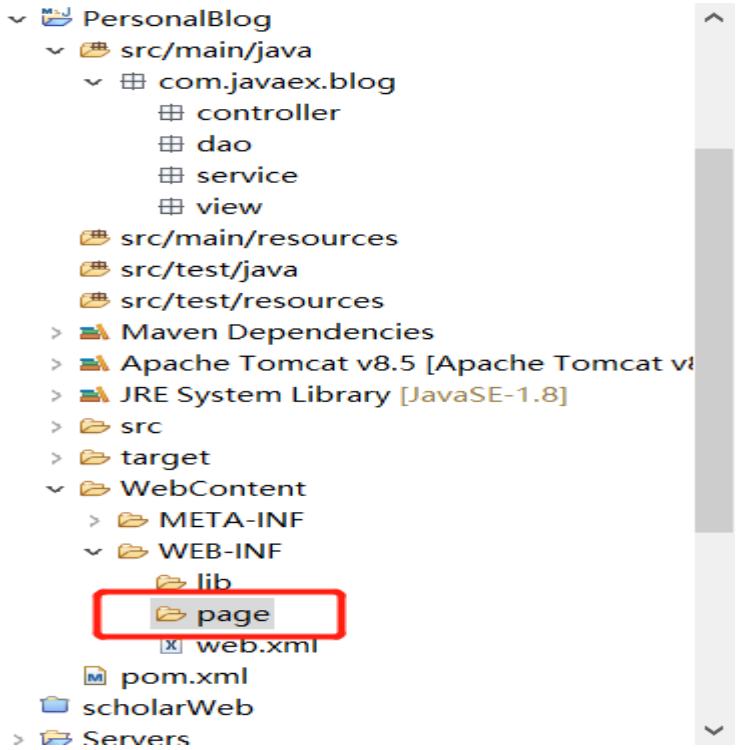
1.包目录



View:视图层
Service：业务层
Dao：与数据库打交道，持久层
Controller：控制层

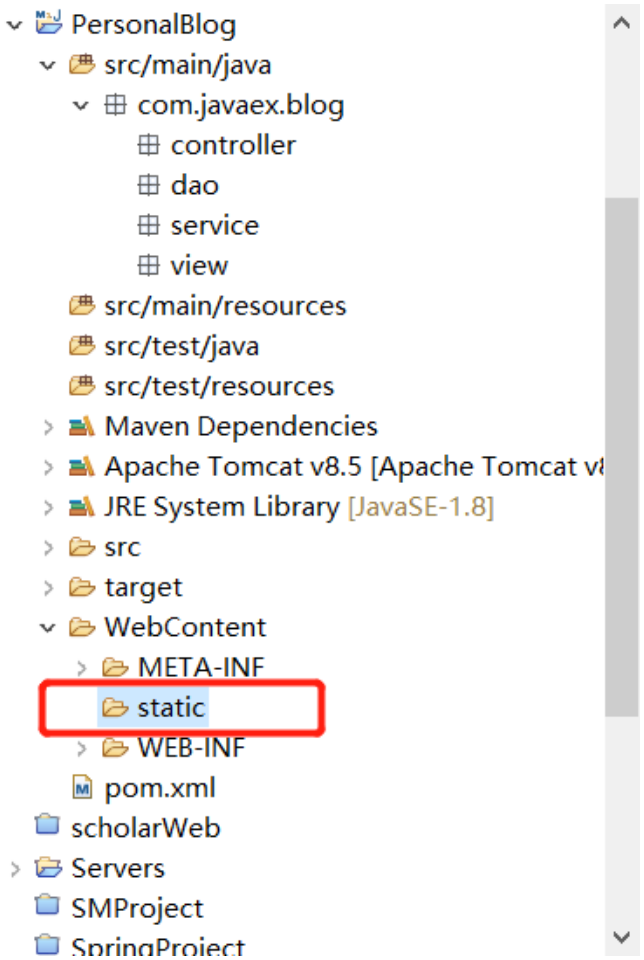
2.后台页面目录

在 web-inf 里面是安全的，无法直接访问



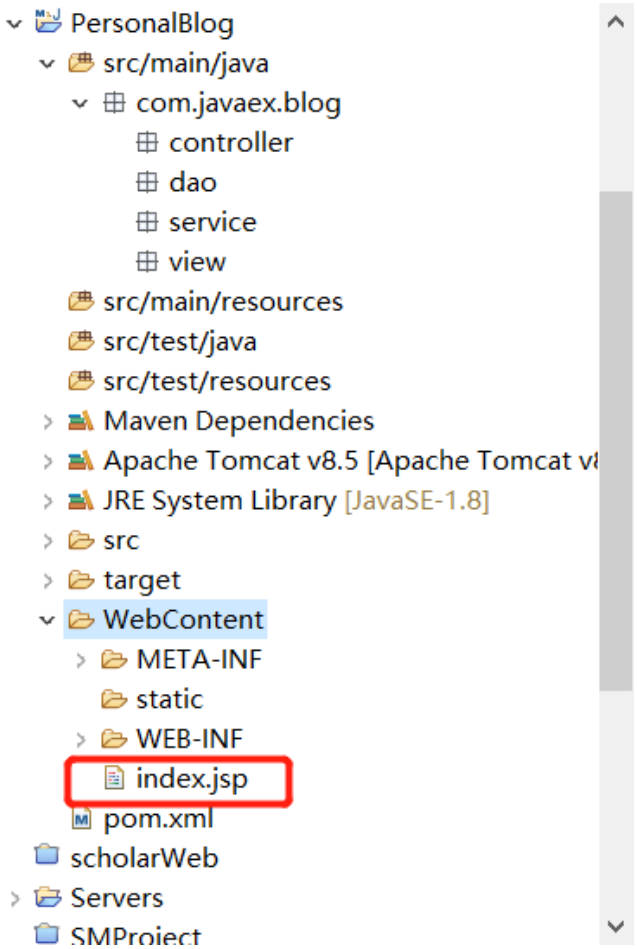
2. 静态资源目录

建在 WebContent（Webapp）下，可以直接访问



3. 默认首页

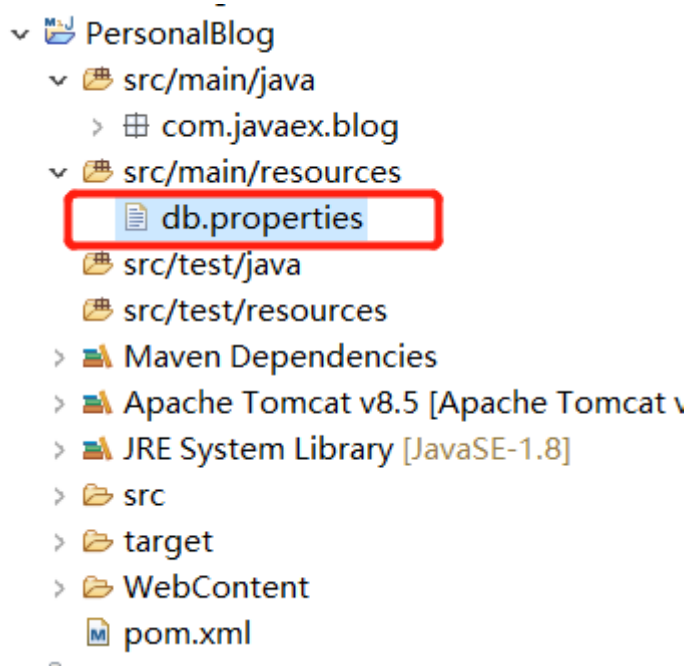
创建默认首页，建在 WebContent（Webapp）下，可以直接访问



5.文件配置

1. 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

```
driverName=com.mysql.jdbc.Driver
jdbcUrl=jdbc:mysql://localhost:3306/baidu?useUnicode=true&characterE
userName=root
password=123
```

2. spring-mybatis

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

```
<!-- 加载db.properties文件 -->
<bean  id="config" class="org.springframework.beans.factory.config.Preference
    <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.DataSourceTransactionManag
     <property name="dataSource" ref="dataSource"></property>
    </bean>
</beans>
```

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/sch
ema/tx"
   xmlns:util="http://www.springframework.org/s
chema/util"
   xmlns:jee="http://www.springframework.org/sc
hema/jee"
   xsi:schemaLocation="http://www.springframewo
rk.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spri
ng-jdbc-4.0.xsd
      http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/sprin
g-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/sprin
g-jee-4.0.xsd
```

```
   http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spr
ing-beans.xsd

   http://www.springframework.org/schema/contex
t
http://www.springframework.org/schema/context/s
pring-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/spri
ng-util-4.0.xsd">



   <!-- 加载db.properties文件 -->
   <bean  id="config"
class="org.springframework.beans.factory.config
.PreferencesPlaceholderConfigurer">
```

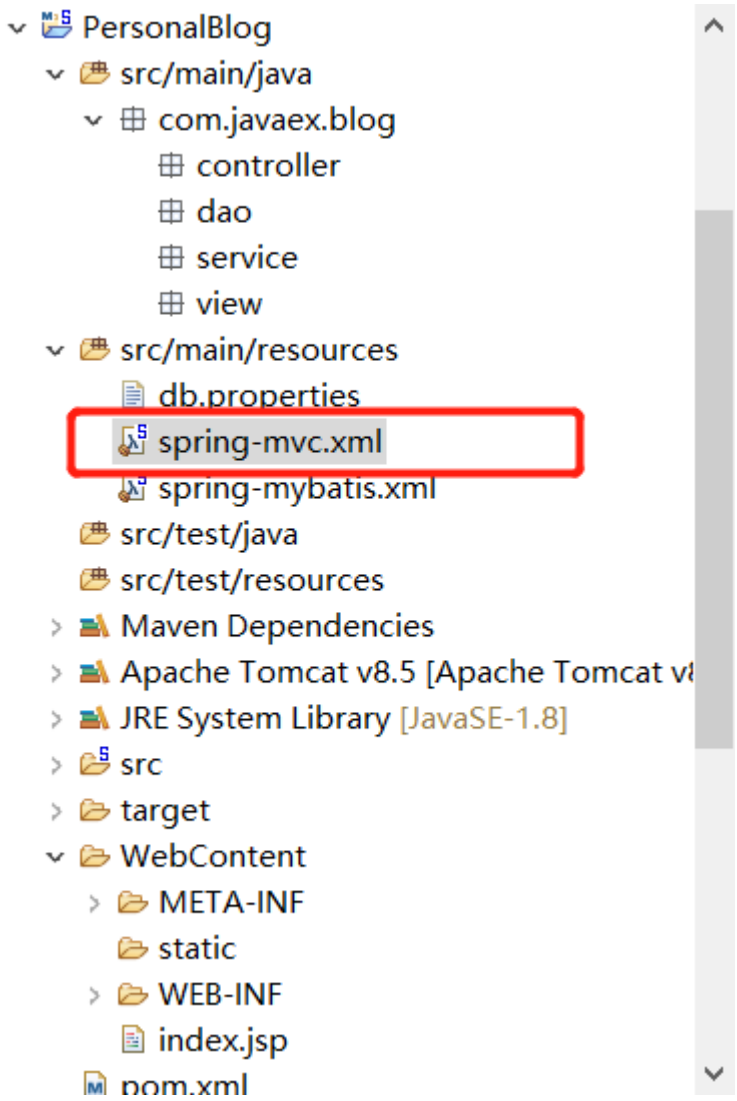
```
      <property name="locations">
        <array>

   <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"></p
roperty>
  </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.MapperScannerC
onfigurer">
```

```
        <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.Data
SourceTransactionManager">
      <property name="dataSource"
ref="dataSource"></property>
    </bean>
</beans>
```

3. spring-mvc

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



```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: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.
        http://www.springframework.org/schema/beans http://www.springframewor
        http://www.springframework.org/schema/context http://www.springframew

     <!-- 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:componer

    <!-- 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.InternalResourceViewRes
       <property name="prefix" value="/WEB-INF/page/"></property>
       <property name="suffix" value=".jsp"></property>
    </bean>
</beans>
```

懒人模式：

```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: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.Int
ernalResourceViewResolver">
        <property name="prefix" value="/WEB-
INF/page/"></property>
        <property name="suffix"
value=".jsp"></property>
    </bean>
</beans>
```

4. web.xml

配置 web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
  <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>
```

```
23          <servlet-class>org.springframework.web.serv
24          <init-param>
25                  <param-name>contextConfigLocation
26                  <param-value>classpath:spring-*.x
27          </init-param>
28    </servlet>
29    <servlet-mapping>
30          <servlet-name>SpringMVC</servlet-name>
31          <url-pattern>*.action</url-pattern>
32    </servlet-mapping>
33    <servlet-mapping>
34          <servlet-name>SpringMVC</servlet-name>
35          <url-pattern>*.json</url-pattern>
36    </servlet-mapping>
37
38 </web-app>
```

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.Dispatche
rServlet</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

1. 准备数据库数据

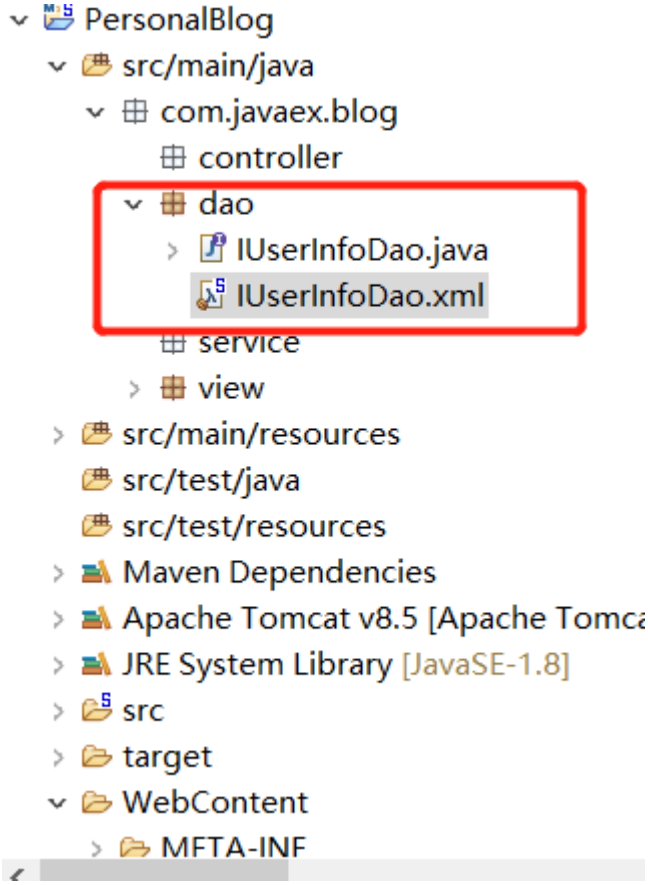
创建数据库和表和添加一条测试数据

2. 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

3. 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

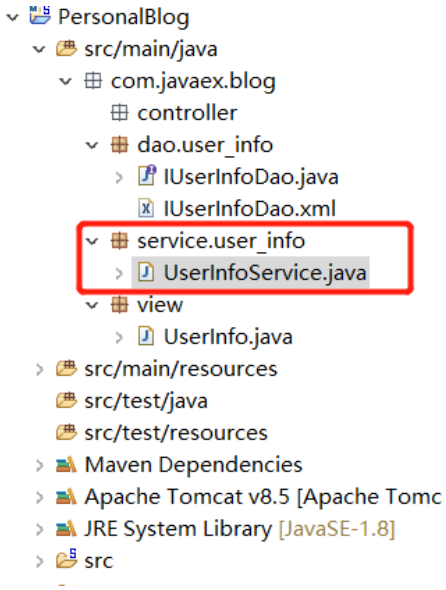
用来写 sql 语句，实现 dao 层接口，sql 的 id 对应接口的函数名

```
<?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.IUserI
nfoDao">
```

```
    <!-- 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>
```

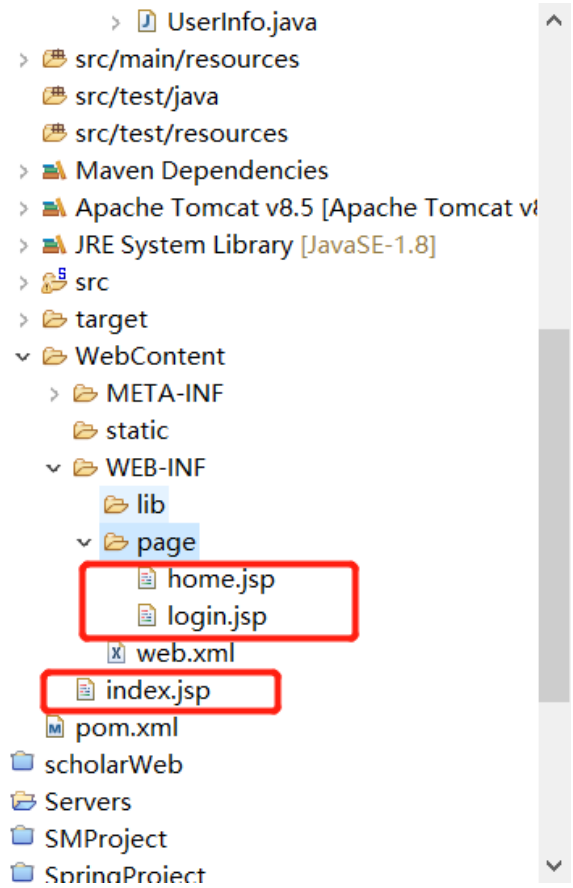
4. service 层




```
package com.javaex.blog.service.user_info;

import
org.springframework.beans.factory.annotation.Au
towired;

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);
   }
}
```

5. 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

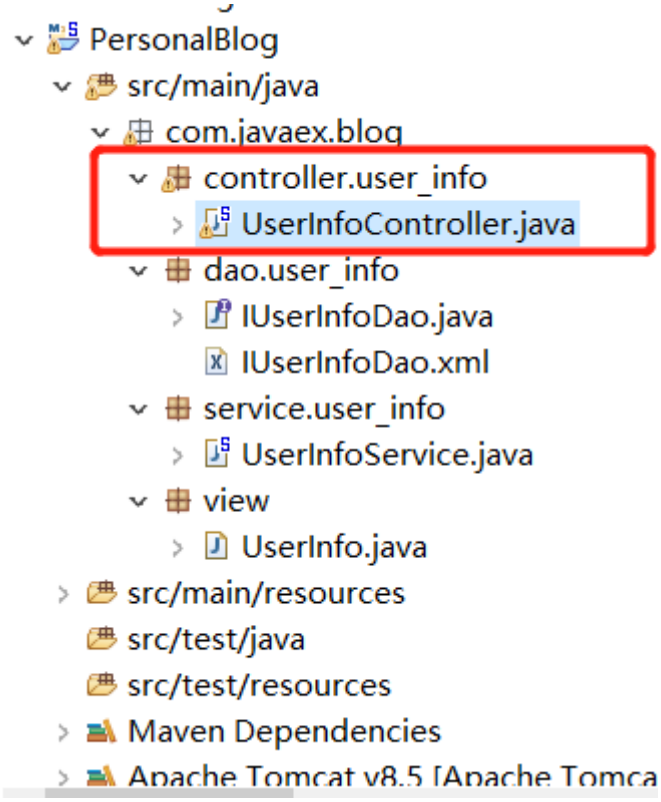
```
<%
    pageContext.setAttribute("APP_PATH",
request.getContextPath());
%>
<script>
    window.location.href
="${APP_PATH}/user_info/index.action";
</script>
```

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)

6. 控制层




```
package com.javaex.blog.controller.user_info;

import
org.springframework.beans.factory.annotation.Au
towired;
import
org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import
org.springframework.web.bind.annotation.Request
```

```
Mapping;
import
org.springframework.web.bind.annotation.Request
Param;

import
com.javaex.blog.service.user_info.UserInfoServi
ce;
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.is
Empty(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