**零基础搭建最新环境下的struts2+spring4+mybatis3框架**

本文档仅限于初级开发者使用，使用者可按部就班依照介绍一次性搭建起框架学习。文中仅提供**编写本文档时的官方最新资源下载地址**，使用者可去官网自行下载。

[1 开发环境 2](#_Toc389917461)

[2 资源准备 2](#_Toc389917462)

[2.1环境设置 2](#_Toc389917463)

[2.2下载资源包 7](#_Toc389917464)

[3 搭建步骤 9](#_Toc389917465)

[3.1 新建WEB工程 9](#_Toc389917466)

[3.1.1 创建项目 9](#_Toc389917467)

[3.1.2 创建目录 14](#_Toc389917468)

[3.1.3 新建服务器 17](#_Toc389917469)

[3.2 集成struts2 25](#_Toc389917470)

[3.2.1 添加资源包 25](#_Toc389917471)

[3.2.2 创建package目录 26](#_Toc389917472)

[3.2.3 代码部分 27](#_Toc389917473)

[3.2.4 新建模块struts文件 36](#_Toc389917474)

[3.3 集成spring4 39](#_Toc389917475)

[3.3.1 添加资源包 39](#_Toc389917476)

[3.3.2 创建spring配置文件 40](#_Toc389917477)

[3.3.3 代码部分 43](#_Toc389917478)

[3.4 集成MyBatis3 48](#_Toc389917479)

[3.4.1 添加资源包 48](#_Toc389917480)

[3.4.2 添加配置 50](#_Toc389917481)

[3.4.3 排错过程 53](#_Toc389917482)

[3.4.4 代码部分 55](#_Toc389917483)

[3.4.5 最终结果 60](#_Toc389917484)

# 开发环境

|  |  |
| --- | --- |
| **操作系统** | Windows 7 (x86) |
| **IDE** | Eclipse 3.7 |
| **JDK** | SUN JDK1.6 |
| **WEB服务器** | Tomcat 7.0 |
| **数据库** | MySQL 5.5.31 |

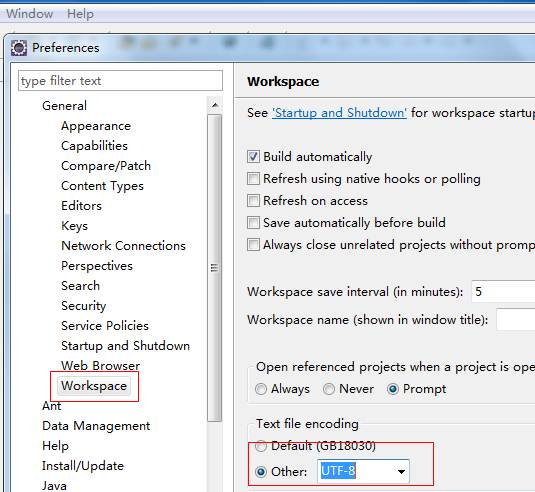
# 资源准备

## 2.1环境设置

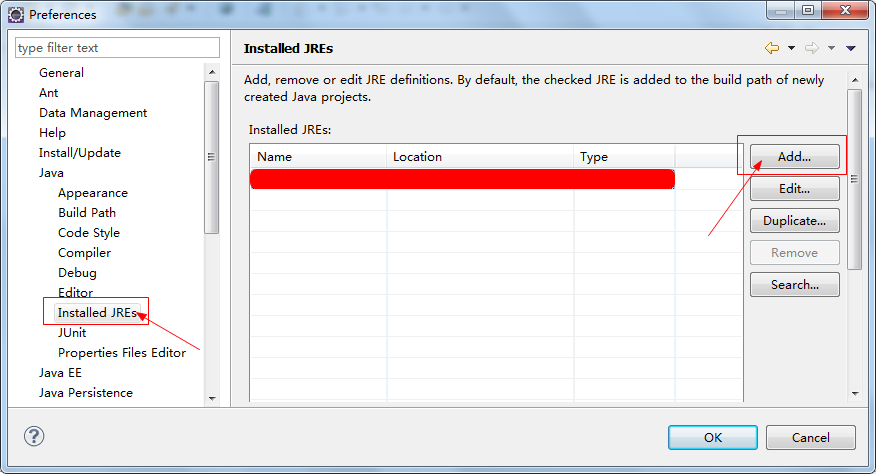
**设置Eclipse**

Window 🡪 Preferences 🡪 General

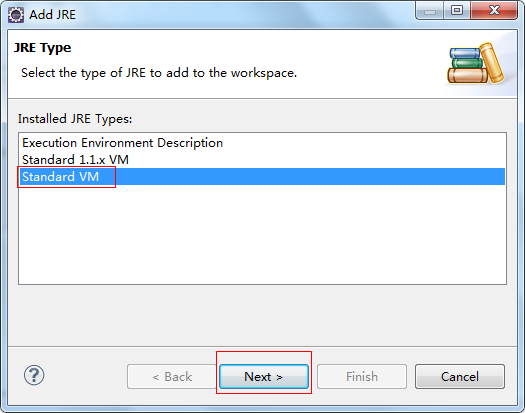
设置工作空间编码格式 General 🡪 Workspace 设置为UTF-8



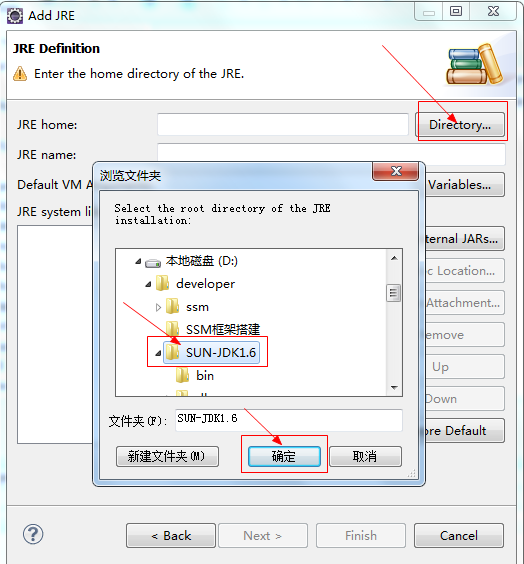
设置Java 🡪 Installed JREs 设置为jdk1.6

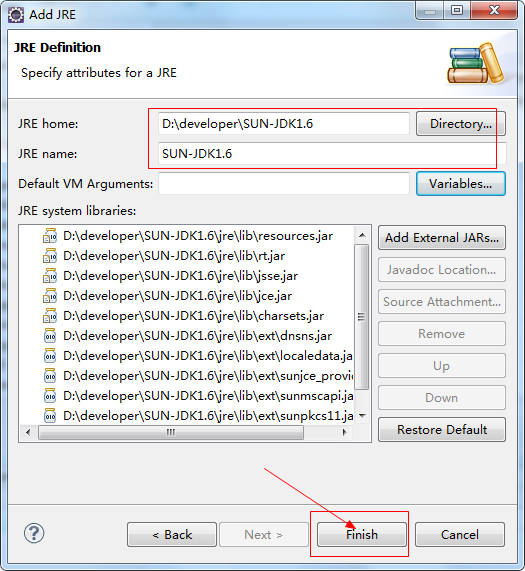


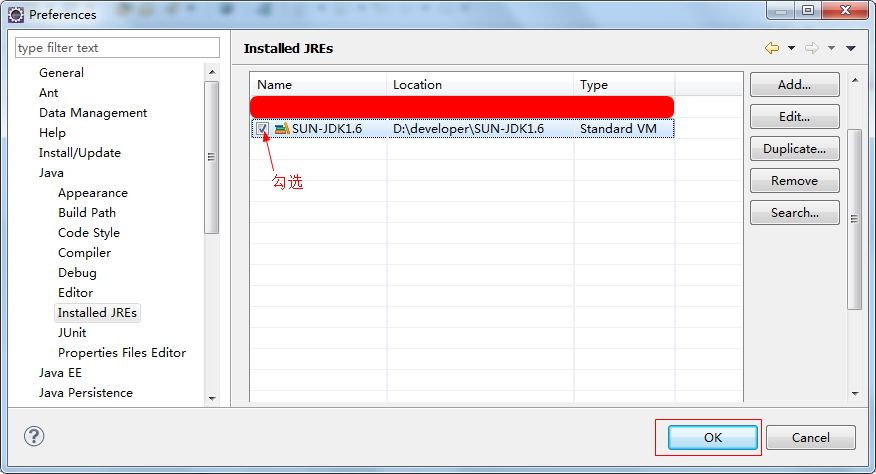
点击add



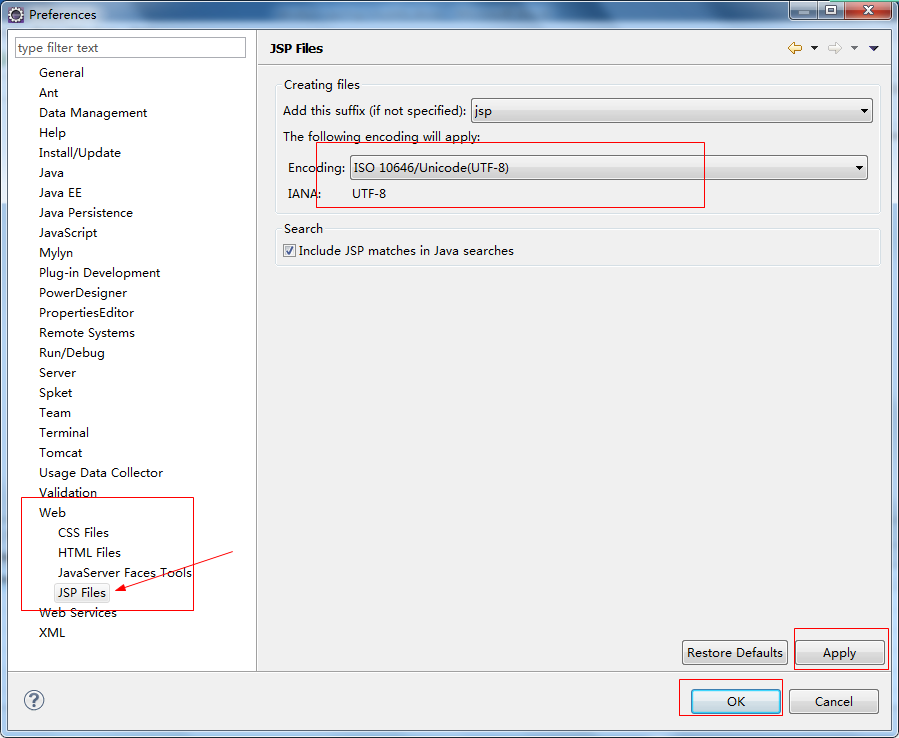
找到你的JDK目录确定







设置服务器新建JSP编码格式为UTF-8



自行建数据库，本项目数据库名：new\_ssm，表名：M\_USER,字段分别为：USERID、USERNAME、USERPASS。

## 2.2下载资源包

本文档所用资源包仅限于当前官网最新

Struts2访问地址：

<http://struts.apache.org/download.cgi%23struts23163>

资源包下载地址：

[struts-2.3.16.3-all.zip](http://mirrors.hust.edu.cn/apache/struts/binaries/struts-2.3.16.3-all.zip)

Spring4发布版镜像地址：

<http://maven.springframework.org/release/org/springframework/spring/>

资源包最新发布版4.0.5下载地址：

[spring-framework-4.0.5.RELEASE-dist.zip](http://maven.springframework.org/release/org/springframework/spring/4.0.5.RELEASE/spring-framework-4.0.5.RELEASE-dist.zip)

最新spring-framework-4.1.0.BUILD-SNAPSHOT-dist.zip版暂时不在此次下载。

MyBatis3镜像地址：

<https://github.com/mybatis/mybatis-3/releases>

资源包下载地址：

[mybatis-3.2.7.zip](https://github.com/mybatis/mybatis-3/releases/download/mybatis-3.2.7/mybatis-3.2.7.zip)

Spring 和MyBatis集成插件镜像地址：

<http://repo1.maven.org/maven2/org/mybatis/mybatis-spring/>

插件下载地址：

[mybatis-spring-1.2.2.jar](http://repo1.maven.org/maven2/org/mybatis/mybatis-spring/1.2.2/mybatis-spring-1.2.2.jar)

Apache Commons DBCP资源包

<http://commons.apache.org/proper/commons-dbcp/download_dbcp.cgi>

下载地址：

本项目JDK环境是1.6，取1.4版本的。

[commons-dbcp-1.4-bin.zip](http://mirror.bit.edu.cn/apache/commons/dbcp/binaries/commons-dbcp-1.4-bin.zip)

Apache Commons POOL资源包

<http://commons.apache.org/proper/commons-pool/download_pool.cgi>

下载地址：

[commons-pool-1.6-bin.zip](http://apache.fayea.com/apache-mirror/commons/pool/binaries/commons-pool-1.6-bin.zip)

本项目取1.6版本的pool资源包，2.2包不适用。

MySql的驱动包自行下载，不提供下载地址。

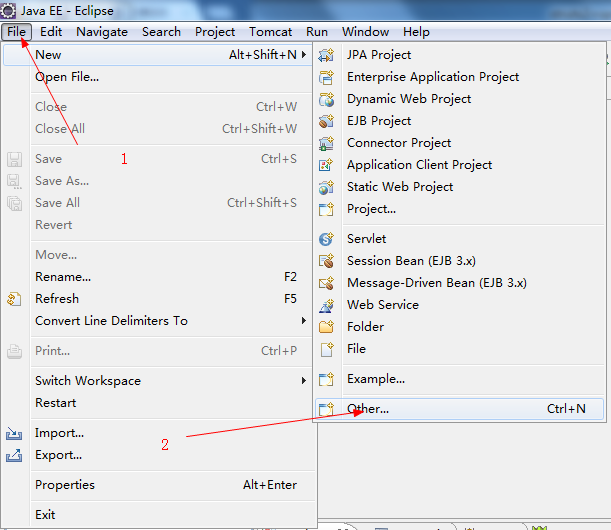
# 搭建步骤

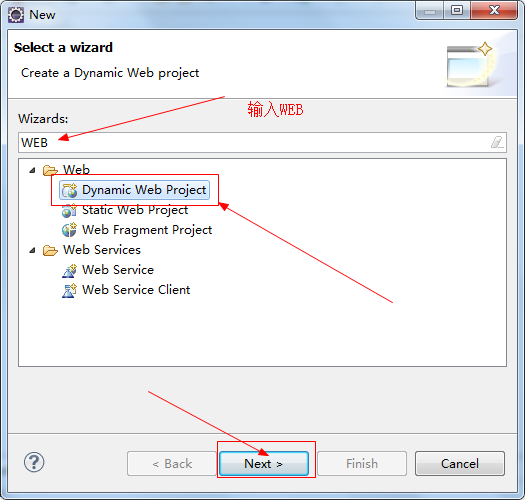
总的步骤为：新建WEB工程🡪集成struts🡪集成spring🡪集成mybatis

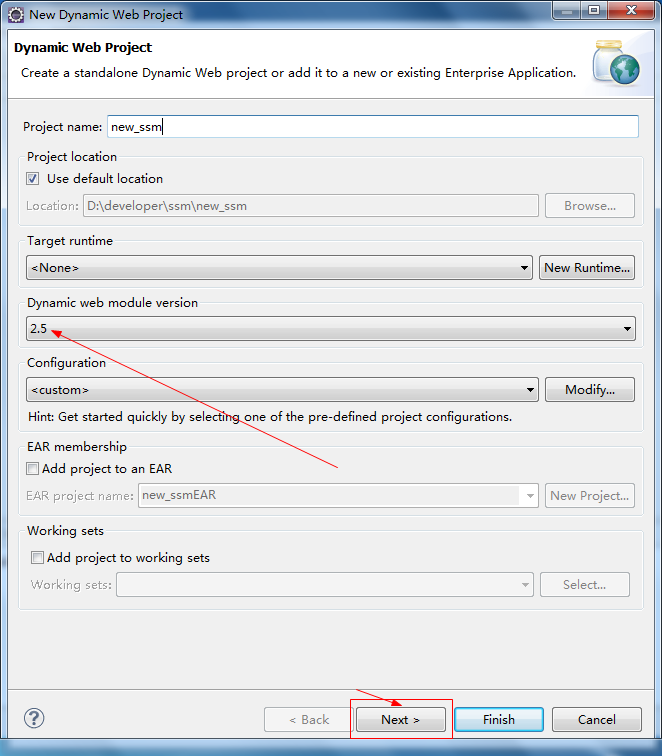
全过程手工搭建，每一步出现问题的解决方案会截图显示。

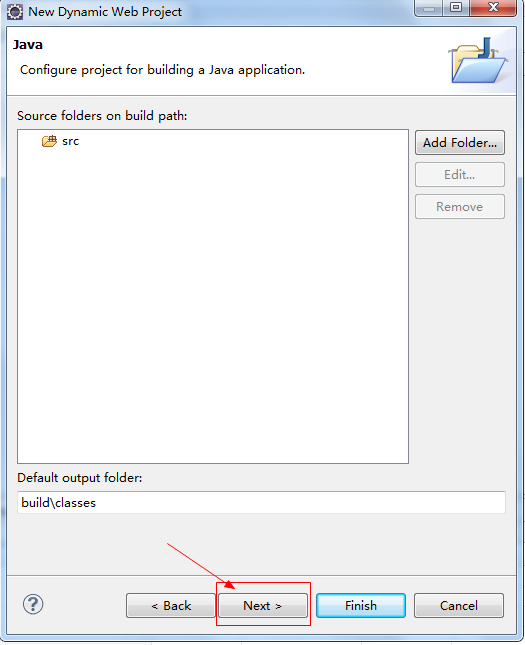
## 新建WEB工程

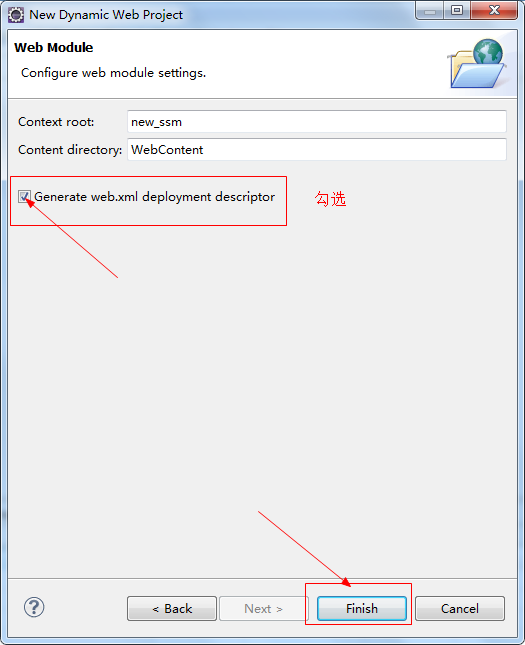
### 创建项目

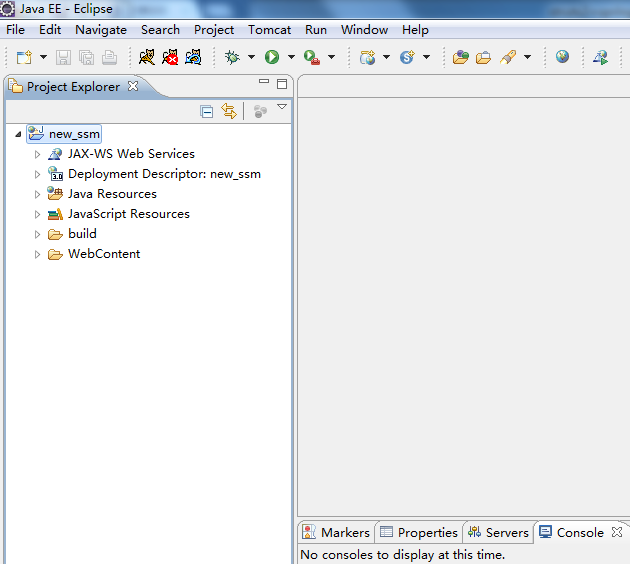








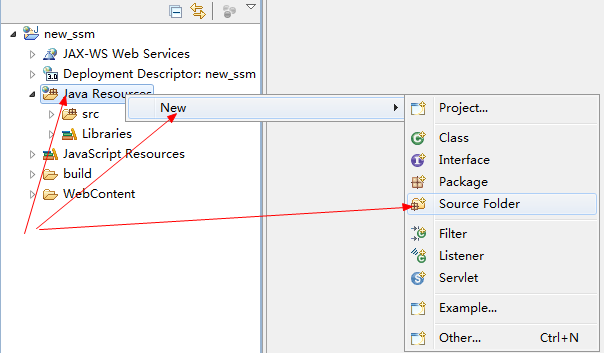


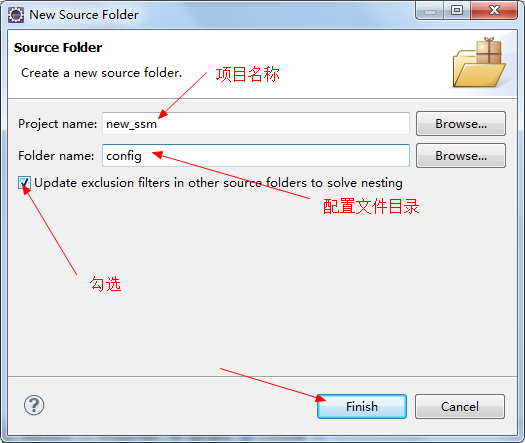


### 创建目录

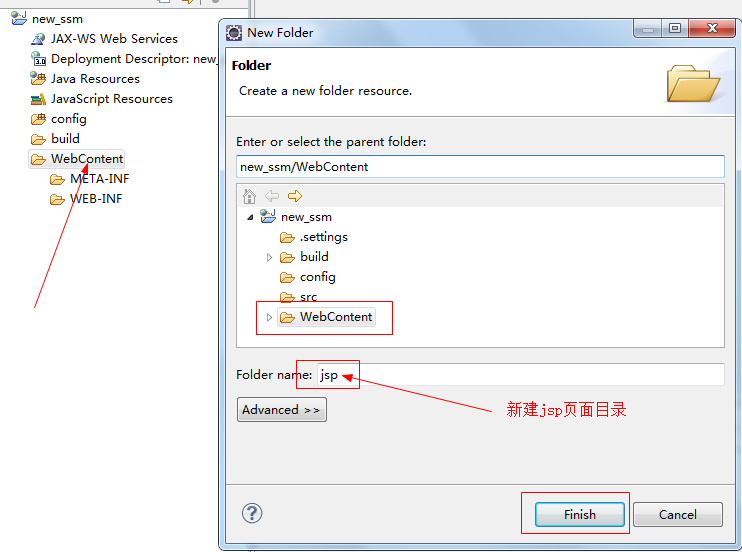
#### 配置文件目录

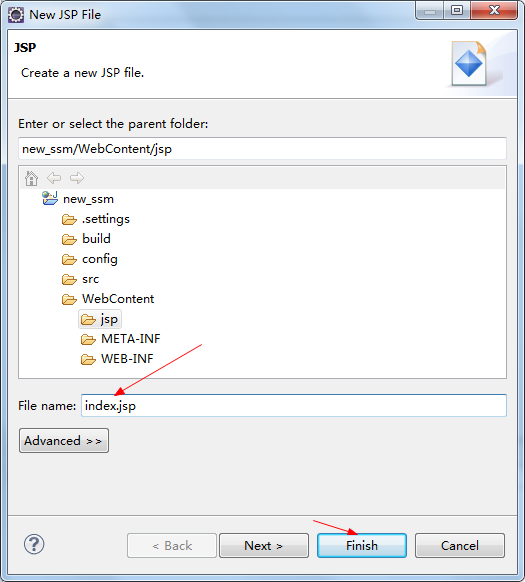
为了便于管理各个配置文件，创建配置文件源文件目录。项目编译时会自动编译到根目录下。





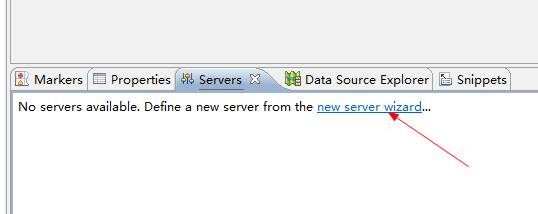
#### jsp页面目录

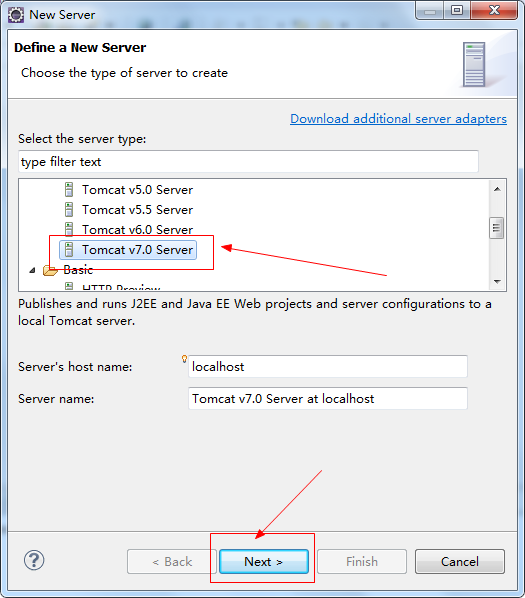


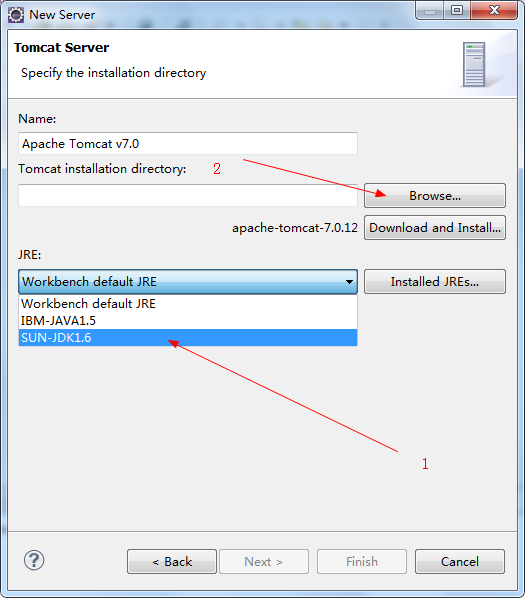


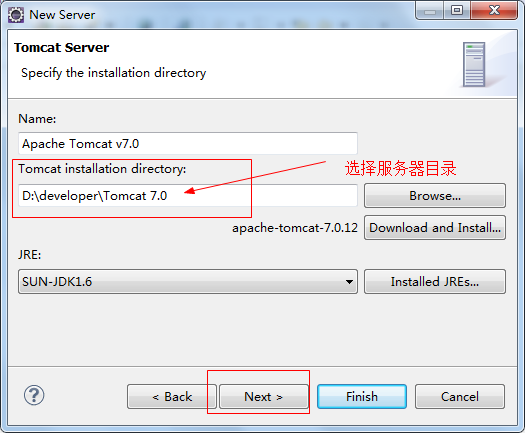
在首页中加上一句话测试看看能否正常显示。

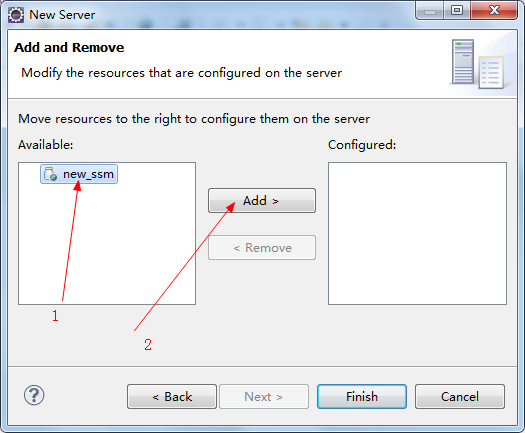
### 新建服务器

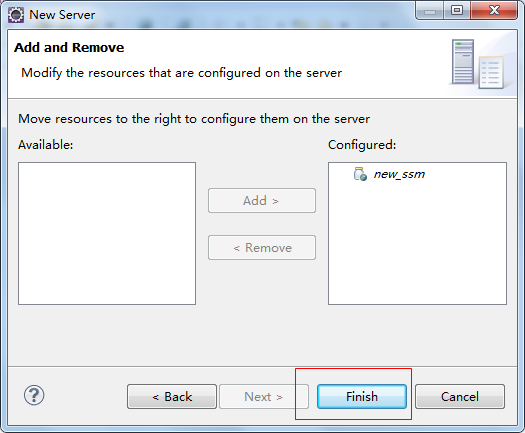


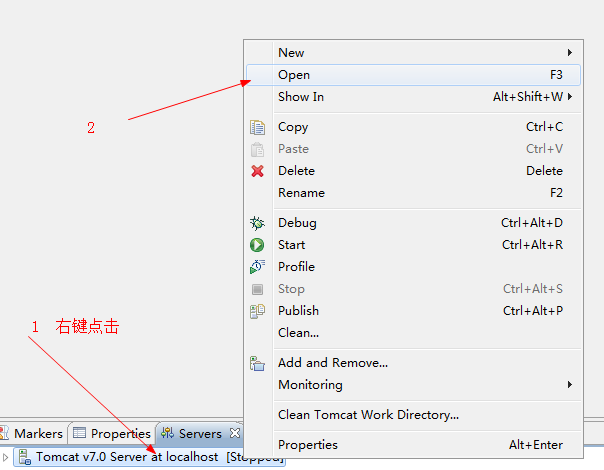




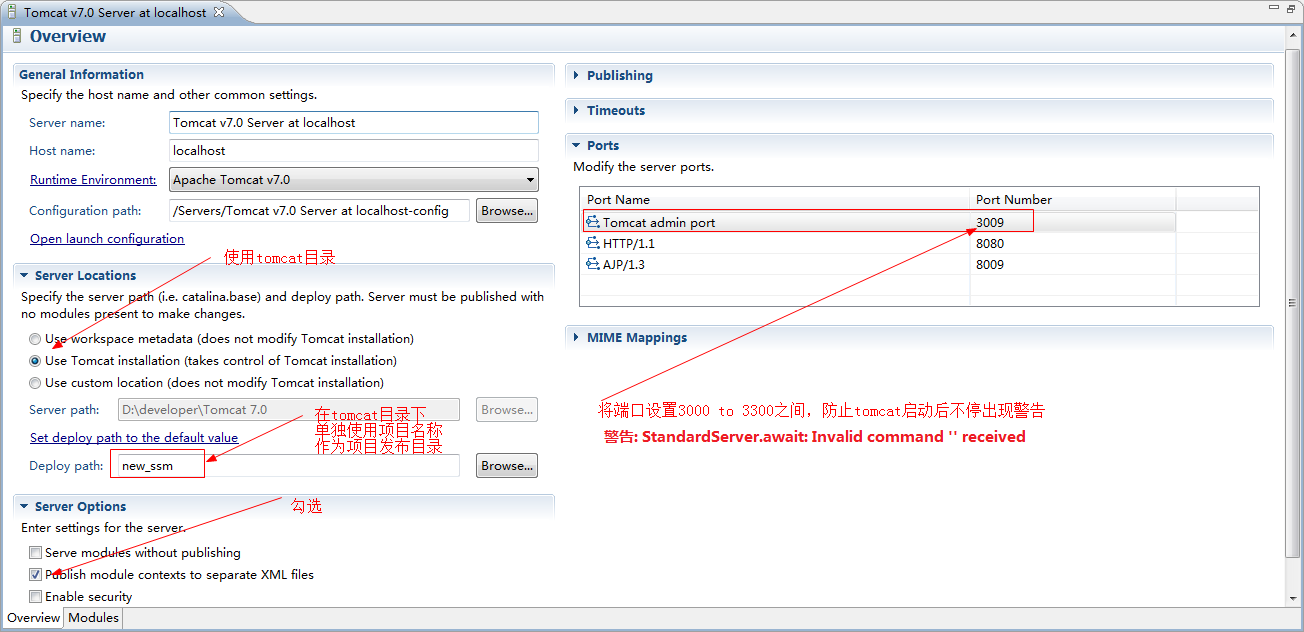


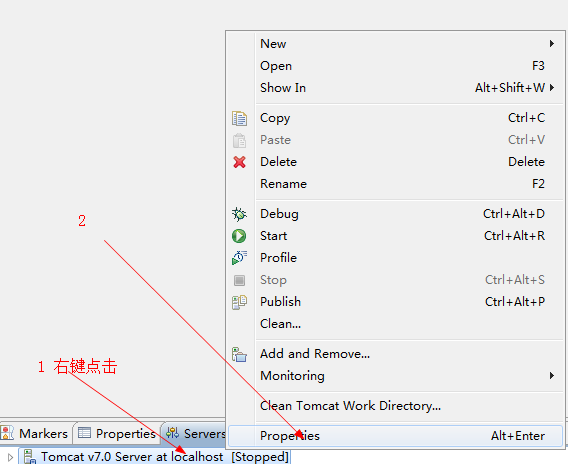


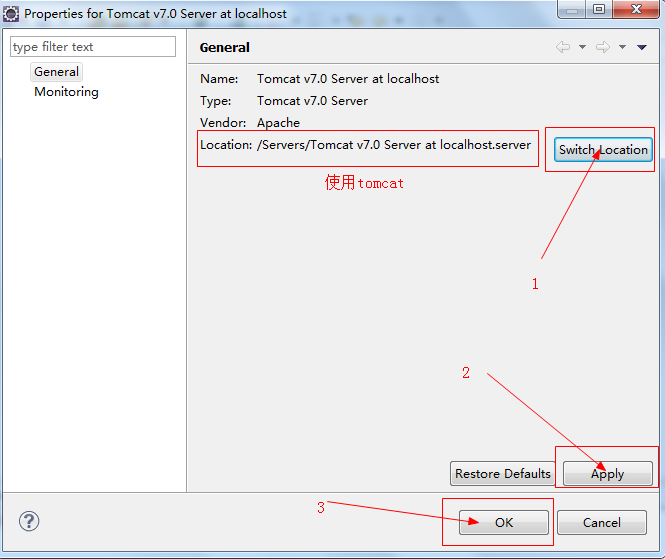




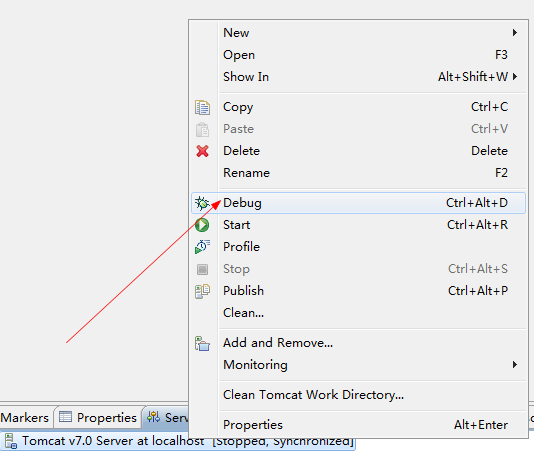
设置一下服务器

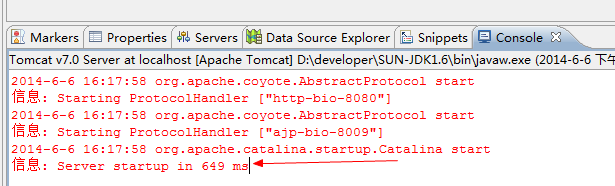






右键debug模式下启动服务器

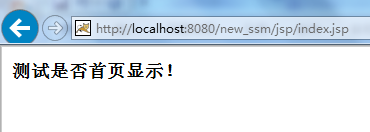




启动成功，测试创建web工程是否成功。

浏览器输入一下地址测试（使用index.jsp的绝对路径）

http://localhost:8080/new\_ssm/jsp/index.jsp



创建web工程并且发布到tomcat成功。

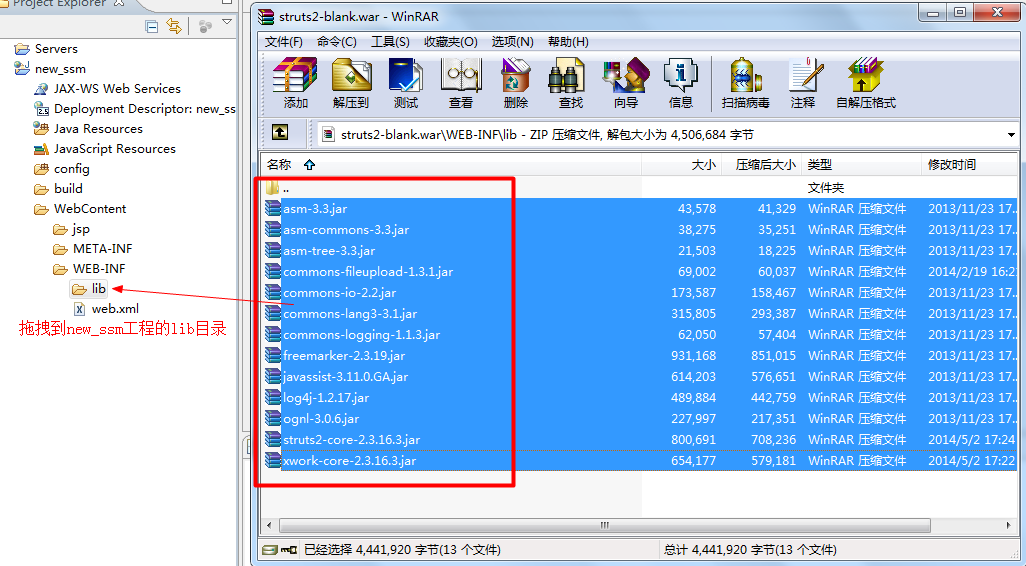
## 集成struts2

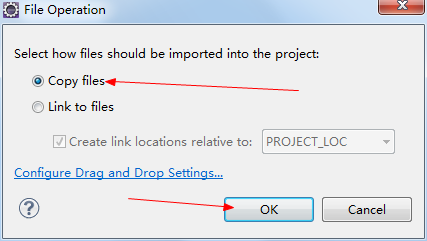
### 添加资源包

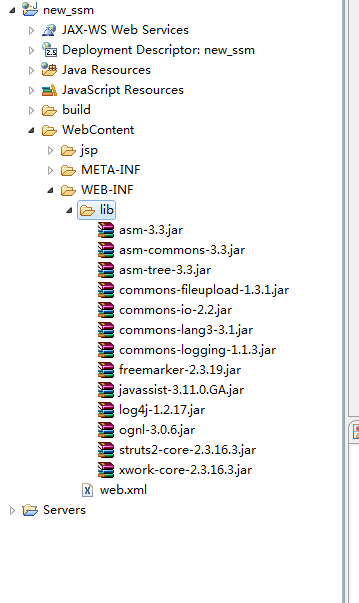
打开struts-2.3.16.3-all.zip压缩包进入

struts-2.3.16.3-all.zip\struts-2.3.16.3\apps\struts-2.3.16.3\apps目录将struts2-blank.war文件解压到当前目录

打开struts2-blank.war文件，将该文件struts2-blank.war\WEB-INF\lib目录下所有文件加入web工程的lib目录内。



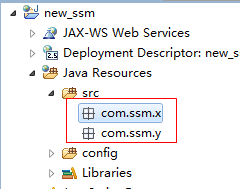




### 创建package目录

建议根据功能模块创建package

本次项目假设有个两个模块分别是x、y模块则创建com.ssm.x和com.ssm.y包

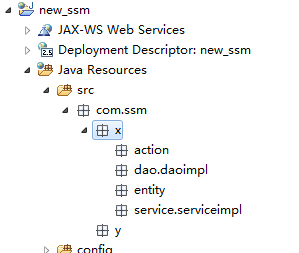


每个模块中分别建立不同的控制层package存放java文件

以x模块为例分别建立action包、service包、dao包、entity包，也可根据实际项目建立用于校验表单的validator包等等。

action存放动作类、service存放业务处理类、dao存放数据处理类、entity存放业务对象类。

service包内再建立serviceimpl包存储service接口实现类，dao包内再建立daoimpl包存储dao接口实现类。



### 代码部分

创建一个用户user的实体类

|  |
| --- |
| **package** com.ssm.x.entity;  **import** java.io.Serializable;  /\*\*  \* **@author** cjz  \*  \*/  **public** **class** UserEntity **implements** Serializable {  /\*\*  \*  \*/  **private** **static** **final** **long** *serialVersionUID* = 1L;    **private** Integer userId;    **private** String userName;    **private** String userPass;  **public** Integer getUserId() {  **return** userId;  }  **public** **void** setUserId(Integer userId) {  **this**.userId = userId;  }  **public** String getUserName() {  **return** userName;  }  **public** **void** setUserName(String userName) {  **this**.userName = userName;  }  **public** String getUserPass() {  **return** userPass;  }  **public** **void** setUserPass(String userPass) {  **this**.userPass = userPass;  }  } |

创建一个首页动作类，不和业务层交互仅是测试集成struts使用。

|  |
| --- |
| **package** com.ssm.x.action;  **import** com.opensymphony.xwork2.ActionSupport;  /\*\*  \* **@author** cjz  \*  \*/  **public** **class** IndexAction **extends** ActionSupport {  /\*\*  \*  \*/  **private** **static** **final** **long** *serialVersionUID* = 1L;  **public** String execute() **throws** Exception{  **return** *SUCCESS*;  }  } |

编辑工程的web.xml文件增加struts过滤器，修改首页文件位置。

打开struts2-blank.war文件struts2-blank.war\WEB-INF\web.xml文件复制如下内容粘贴到new\_ssm工程\WEB-INF\web.xml内

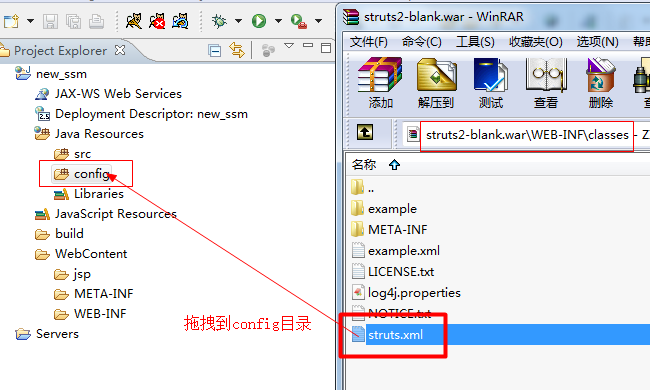
|  |
| --- |
| <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

如下图：



在config文件目录内增加struts.xml文件

拖拽struts2-blank.war\WEB-INF\classes目录下的struts.xml文件到config目录中，并进行修改。

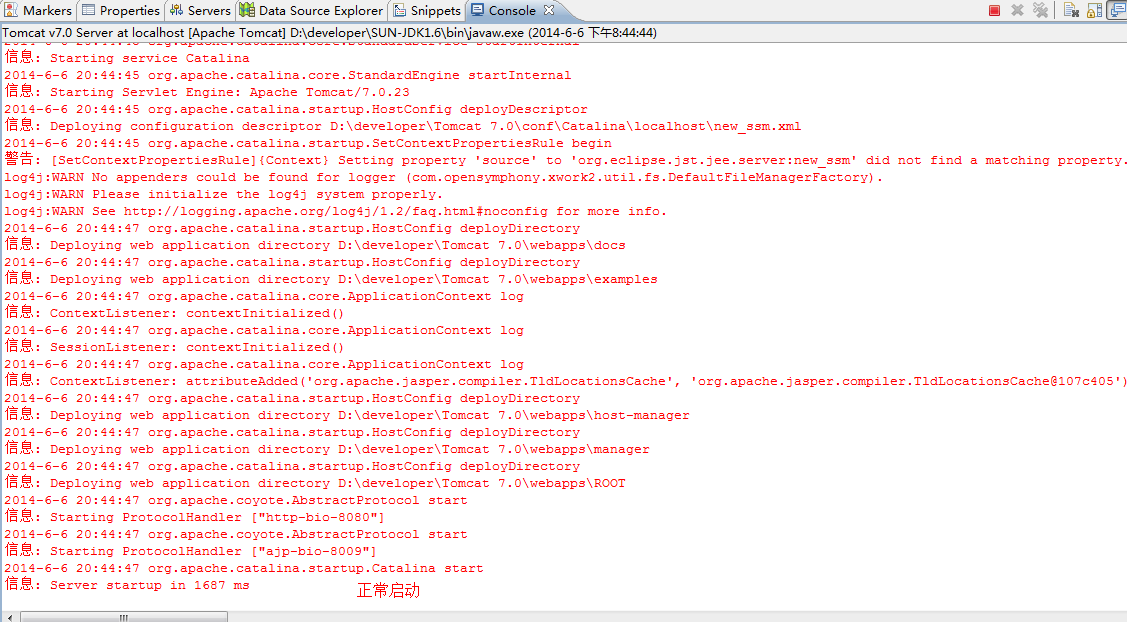




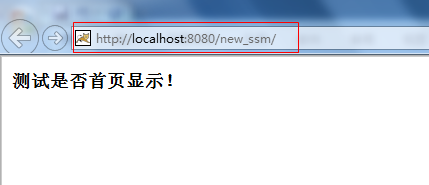
在jsp目录下增加一个error.jsp



启动tomcat



浏览器输入<http://localhost:8080/new_ssm/>看看能否打开首页。



这次没有输入首页jsp的绝对路径打开首页。

修改首页，增加超链到登录窗口。

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>首页</title>  <%  //获取上下文  String path = request.getContextPath();  String basePath = request.getScheme() + "://"  + request.getServerName() + ":" + request.getServerPort()  + path + "/";  %>  </head>  <body>  <h3>首页显示OK！</h3>  <h4>  当前页面上下文路径：<%=basePath%></h4>  <!-- 直接转跳到登录页面 -->  <a target=*"\_self"* href=*"*<%=basePath%>*jsp/login.jsp"*>登录</a>  </body>  </html> |

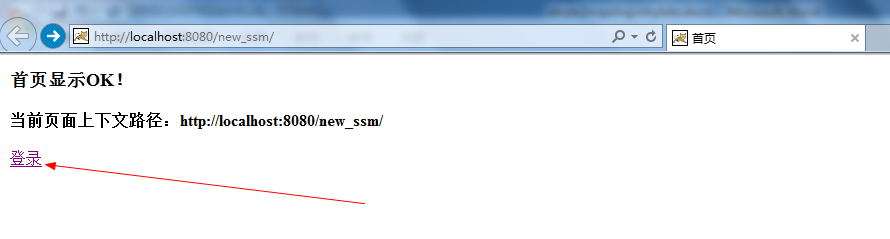
增加登录页面login.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <%@ taglib prefix=*"s"* uri=*"/struts-tags"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>登录</title>  </head>  <body>  <!-- namespace="/x" 以功能模块名称区分 命名空间 -->  <s:form action=*"login"* namespace=*"/x"*>  <table align=*"center"* cellpadding=*"0"* cellspacing=*"0"* border=*"1"*>  <tr>  <s:textfield label=*"用户名"* />  </tr>  <tr>  <s:password label=*"密码"* />  </tr>  <tr>  <s:submit value=*"登录"* />  </tr>  </table>  </s:form>  </body>  </html> |

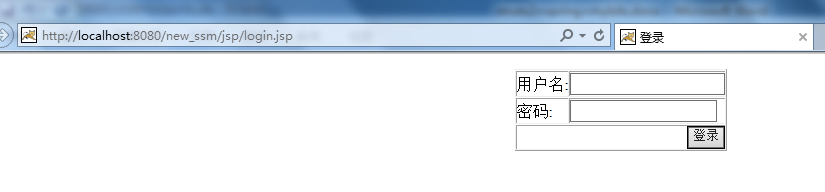
增加登录成功页面ok.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>OK</title>  </head>  <body>  <h3>欢迎您登录</h3>  </body>  </html> |

测试登录页面是否正常打开，打开后因为没写loginAction所以登录不能成功。



打开登录页面



增加LoginAction类

|  |
| --- |
| package com.ssm.x.action;  import com.opensymphony.xwork2.ActionSupport;  import com.ssm.x.entity.UserEntity;  public class LoginAction extends ActionSupport {  //构造方法，测试是否拦截器起作用，在eclipse控制台可以看见打印结果  public LoginAction(){  System.out.println("LoginAction类被调用，struts拦截器起作用，集成struts初步成功！");  }    /\*\* \*/  private static final long serialVersionUID = 1L;    //增加用户对象，生成get、set方法，login页面直接可以使用  private UserEntity entity;    public String execute() throws Exception{  System.out.println("登录界面用户名输入的是："+entity.getUserName());  System.out.println("登录界面密码输入的是："+entity.getUserPass());  //判断登录信息  if (entity.getUserName().equals("ssm") && entity.getUserPass().equals("123")) {  return SUCCESS;  }  return INPUT;  }    public UserEntity getEntity() {  return entity;  }  public void setEntity(UserEntity entity) {  this.entity = entity;  }  } |

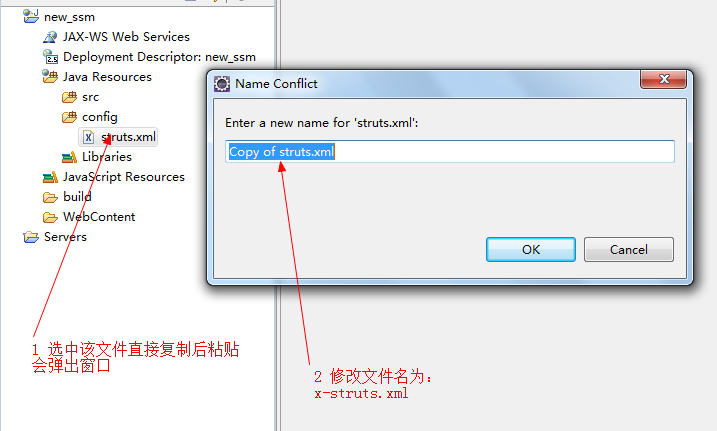
修改login.jsp页面，增加用户名和密码name属性。增加form提交空间名称，后续x模块的struts文件命名空间也以x命名。



### 新建模块struts文件

新建x模块的struts文件

选中struts.xml文件后直接复制粘贴



然后修改x-struts.xml文件，这时的Action类要用绝对路径，package中namespace="/x"本项目以功能模块命名,登录成功跳转到OK页面，否则跳转到错误页面。

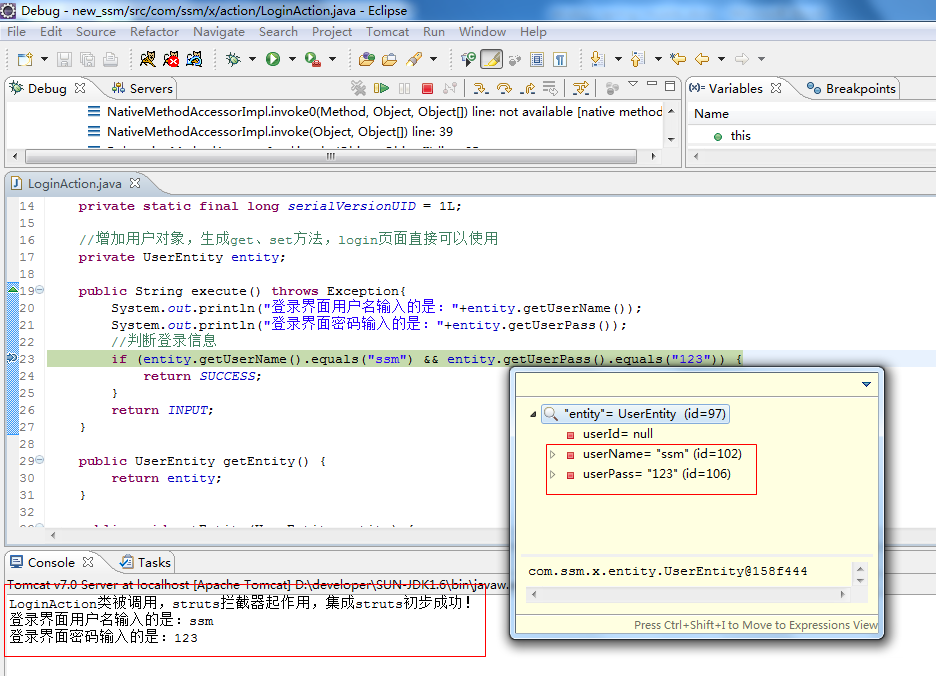
|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"  "http://struts.apache.org/dtds/struts-2.3.dtd">  <struts>  <package name=*"x"* namespace=*"/x"* extends=*"struts-default"*>    <action name=*"login"* class=*"com.ssm.x.action.LoginAction"*>  <result name=*"success"*>/jsp/ok.jsp</result>  <result name=*"input"*>/jsp/error.jsp</result>  </action>  </package>  </struts> |

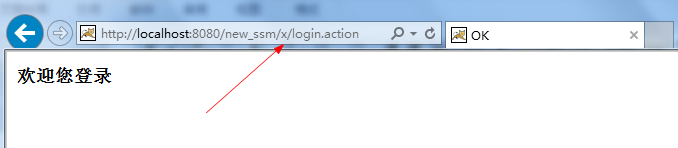
修改struts.xml文件，添加引入x-struts.xml文件代码。



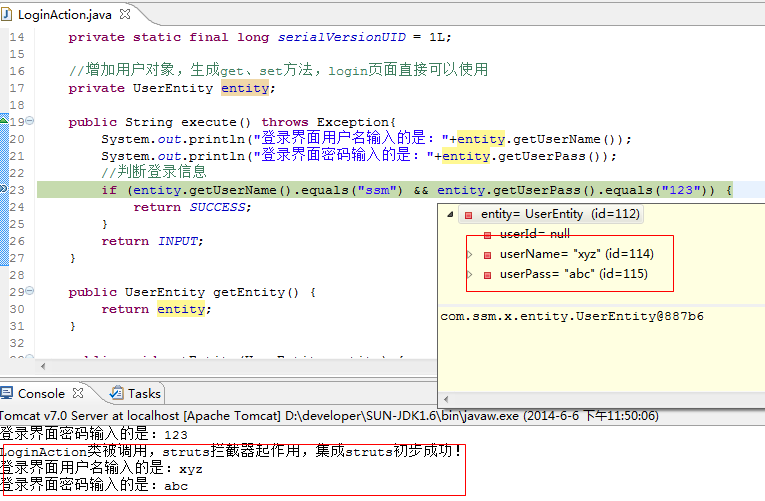
启动tomcat，测试是否集成struts2成功。

断点处看到如下

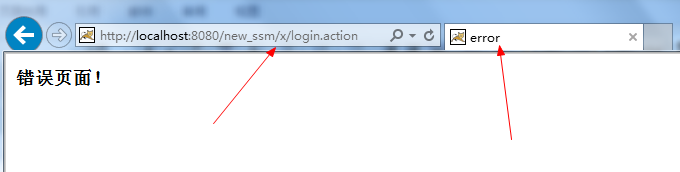




登录成功。再测试错误用户名密码



浏览器地址没变，已经跳转到错误页面。



## 集成spring4

### 添加资源包

打开

spring-framework-4.0.5.RELEASE-dist.zip\spring-framework-4.0.5.RELEASE\libs目录

每种资源包都包含三类文件分别是编译好的jar文件，分别是工具jar，API帮助文档jar,源码jar。项目引入的必须是工具jar文件。

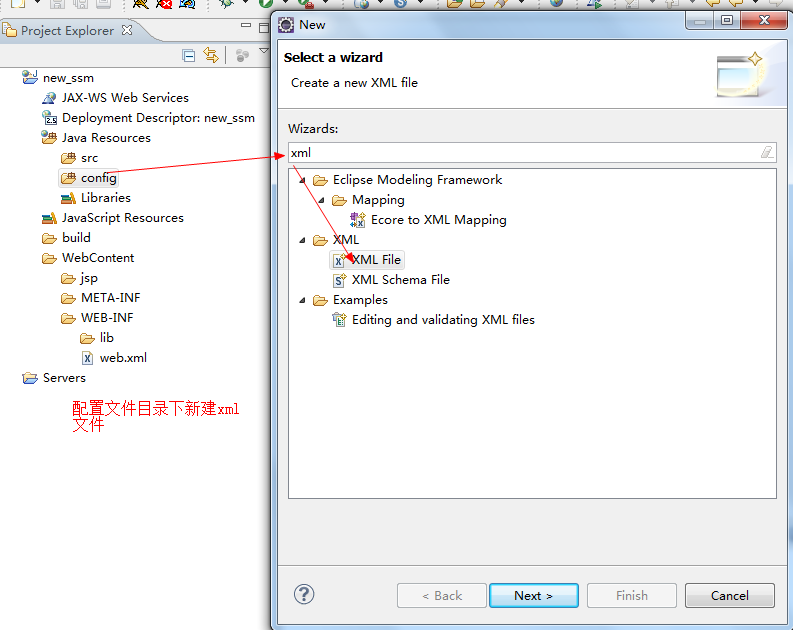
* spring-context-4.0.5.RELEASE.jar
* spring-beans-4.0.5.RELEASE.jar
* spring-context-4.0.5.RELEASE.jar
* spring-web-4.0.5.RELEASE.jar
* spring-tx-4.0.5.RELEASE.jar
* spring-jdbc-4.0.5.RELEASE.jar
* spring-aop-4.0.5.RELEASE.jar
* spring-expression-4.0.5.RELEASE.jar
* 解压struts-2.3.16.3-all.zip

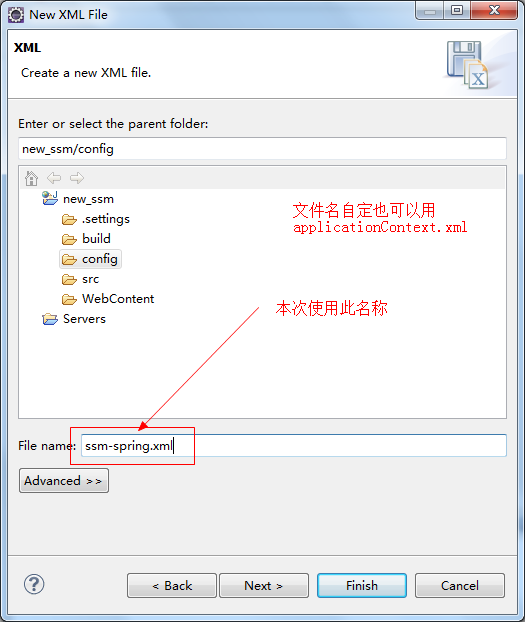
struts-2.3.16.3-all.zip\struts-2.3.16.3\apps中的struts2-showcase.war文件，在struts2-showcase.war\WEB-INF\lib目录下提取struts集成spring的插件包

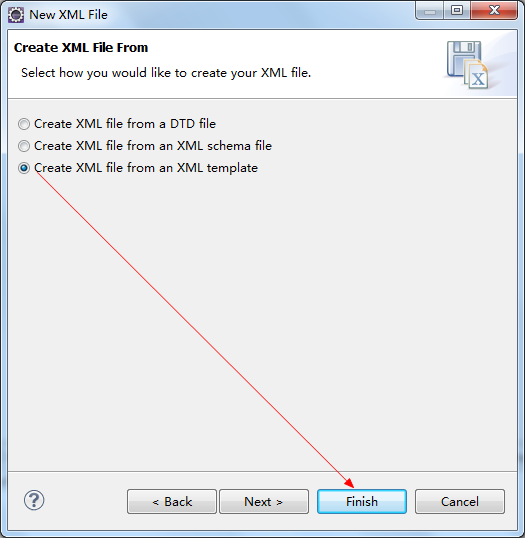
* struts2-spring-plugin-2.3.16.3.jar

### 创建spring配置文件

新建spring的配置文件，名称可以自定。







编写spring配置文件



### 代码部分

添加自动扫描

|  |
| --- |
| <?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:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc.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"*>    <!-- 自动扫描注入 注解 扫描com.ssm目录下所有文件 -->  <!-- @Repository：数据访问层组件，@Service：业务层组件，@Controller：控制层组件 -->  <context:component-scan base-package=*"com.ssm"*/>      </beans> |

修改web.xml,添加spring监听和上下文配置文件位置，将下列框中代码加入web.xml文件中

|  |
| --- |
| <!-- spring监听器 -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>    <listener>  <listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>  </listener>    <!-- spring配置文件的位置 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:ssm-spring.xml</param-value>  </context-param> |

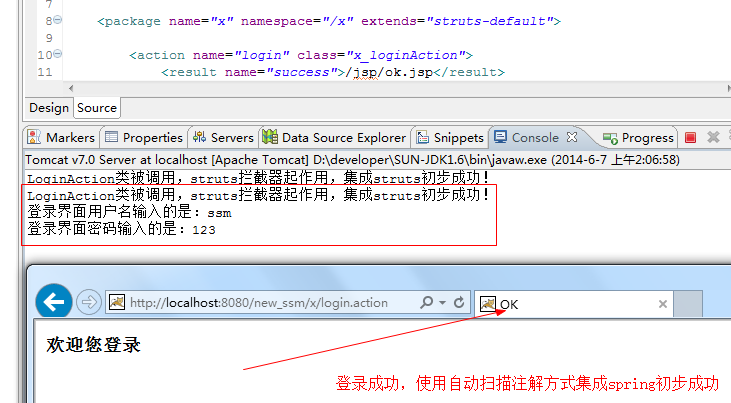
修改LoginAction类，增加注解。



修改x-struts.xml文件中action的class的名称



启动tomcat，登录测试LoginAction类是否被spring自动扫描后托管创建



创建业务处理service层，由spring托管。

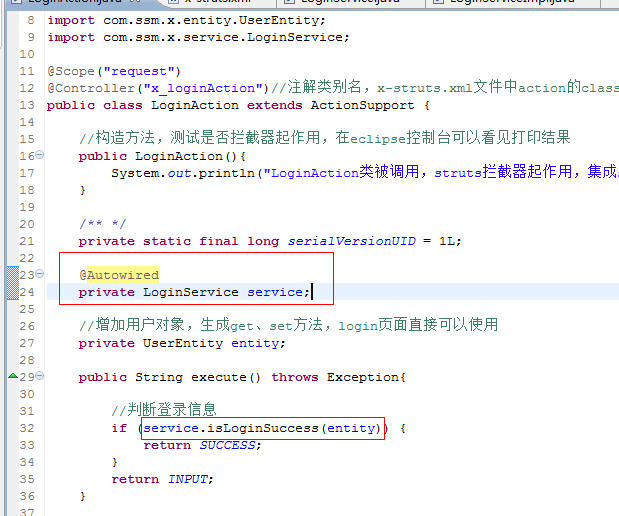
新建LoginService.java接口类

|  |
| --- |
| **package** com.ssm.x.service;  **import** org.springframework.stereotype.Service;  **import** com.ssm.x.entity.UserEntity;  @Service  **public** **interface** LoginService {  **public** **boolean** isLoginSuccess(UserEntity entity);  } |

新建LoginServic.java接口实现类LoginServiceImpl.java

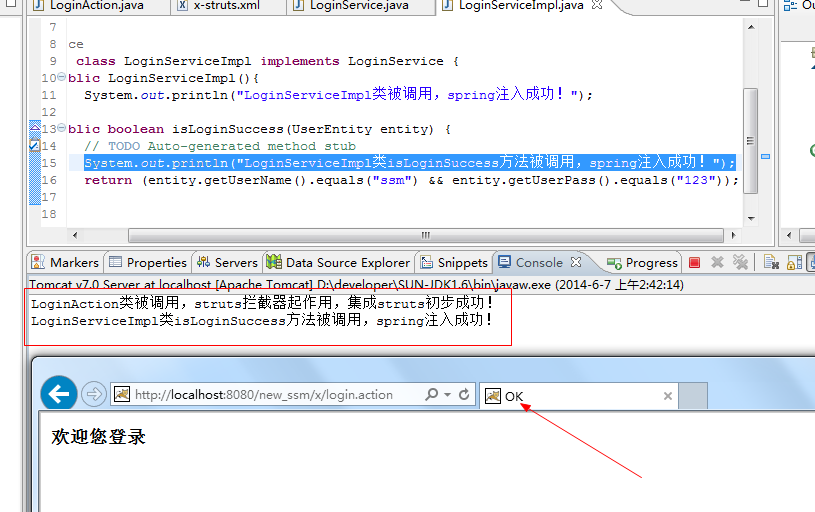
|  |
| --- |
| **package** com.ssm.x.service.serviceimpl;  **import** org.springframework.stereotype.Service;  **import** com.ssm.x.entity.UserEntity;  **import** com.ssm.x.service.LoginService;  @Service  **public** **class** LoginServiceImpl **implements** LoginService {  **public** LoginServiceImpl(){  System.*out*.println("LoginServiceImpl类被调用，spring注入成功！");  }  **public** **boolean** isLoginSuccess(UserEntity entity) {  // **TODO** Auto-generated method stub  System.out.println("LoginServiceImpl类isLoginSuccess方法被调用，spring注入成功！");  **return** (entity.getUserName().equals("ssm") && entity.getUserPass().equals("123"));  }  } |

修改LoginAction.java，注入LoginService接口，由LoginService接口实现业务处理。



启动tomcat，这时容器已经将业务注入

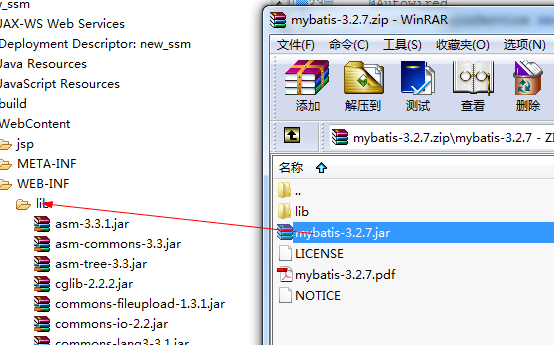




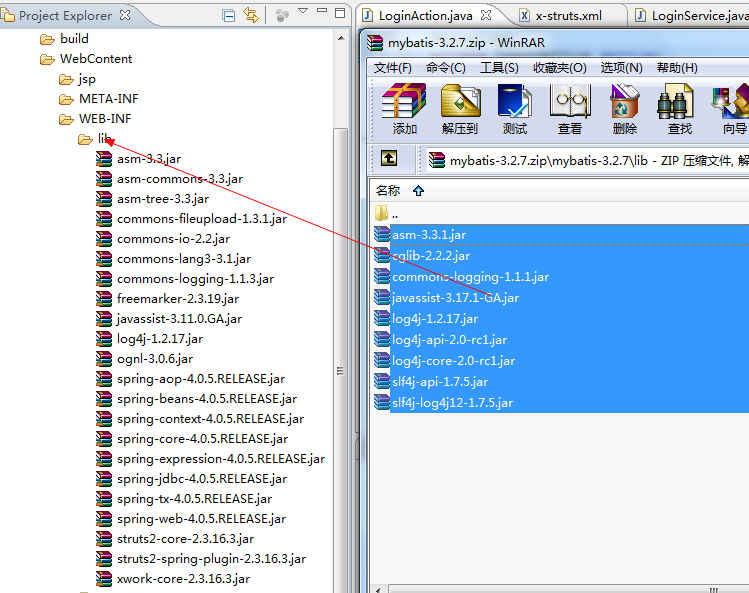
## 集成MyBatis3

### 添加资源包

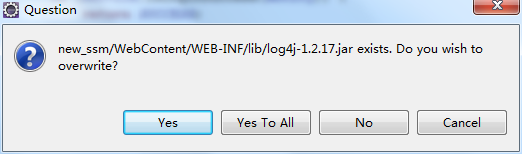
打开mybatis-3.2.7.zip\mybatis-3.2.7目录导入mybatis-3.2.7.jar



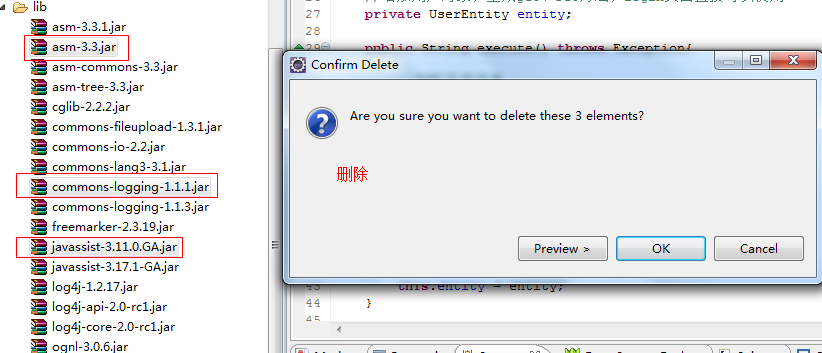
打开mybatis-3.2.7.zip\mybatis-3.2.7\lib目录导入该目录下所有包



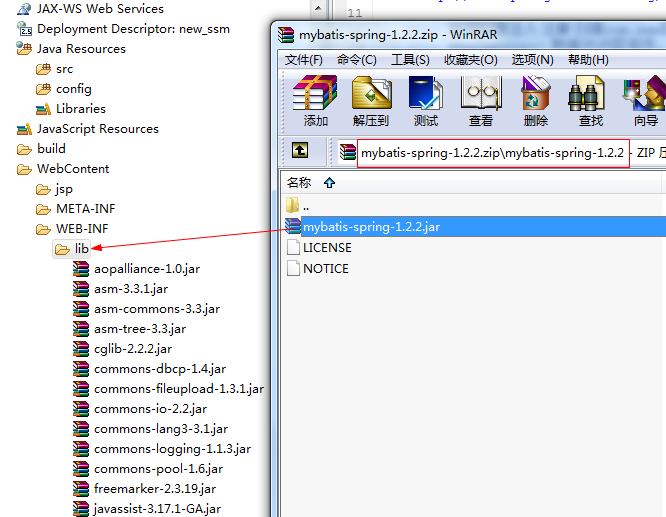
发现已经导入项目lib目录下有重复包



确认覆盖，然后在项目lib目录中删除低版本资源包。



导入mybatis-spring-1.2.2.zip\mybatis-spring-1.2.2目录下的插件包mybatis-spring-1.2.2.jar

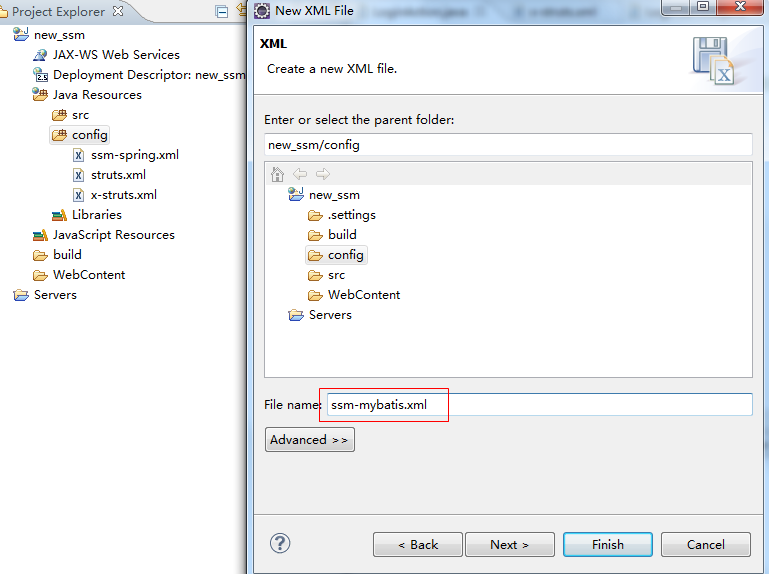


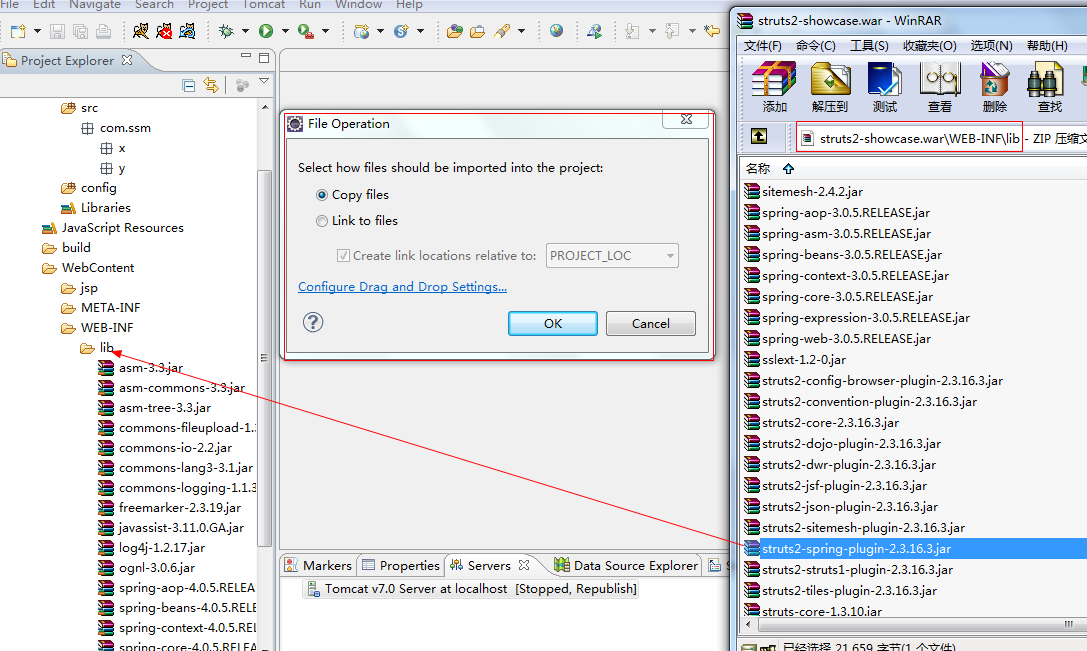
导入mysql驱动包到项目lib目录

mysql-connector-java-5.1.18-bin.jar **这个驱动包使用者自行下载**。

### 添加配置

配置目录下新建MyBatis配置文件ssm-mybatis.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>  <typeAliases>  <!-- 给用户实体映射类起个别名，在SQL映射配置文件里方便使用 -->  <typeAlias alias=*"userEntity"* type=*"com.ssm.x.entity.UserEntity"* />  </typeAliases>  </configuration> |

修改spring的配置文件，增加数据库连接配置，定义扫描MyBatis映射文件。

|  |
| --- |
| <?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:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc.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"*>    <!-- 自动扫描注入 注解 扫描com.ssm目录下所有文件 -->  <!-- @Repository：数据访问层组件，@Service：业务层组件，@Controller：控制层组件 -->  <context:component-scan base-package=*"com.ssm"*/>    <!-- 定义数据源连接 -->  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*>  <property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"url"* value=*"jdbc:mysql://localhost:3306/new\_ssm"* />  <property name=*"username"* value=*"root"* />  <property name=*"password"* value=*"root123"* />  <property name=*"maxActive"* value=*"100"*/>  <property name=*"maxIdle"* value=*"30"*/>  <property name=*"maxWait"* value=*"500"*/>  <property name=*"defaultAutoCommit"* value=*"true"*/>  </bean>  <!-- 定义全局的事务控制 -->  <bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <property name=*"dataSource"* ref=*"dataSource"* />  </bean>  <!-- 开启注解方式声明事务 -->  <tx:annotation-driven />  <!-- 定义SqlSessionFactory 使用mapperLocations，MyBatis-Spring会自动地扫描该路径下的所有Mapper文件-->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"* />  <property name=*"configLocation"* value=*"classpath:ssm-mybatis.xml"* />  <property name=*"mapperLocations"* value=*"classpath:\*-SQLMapper.xml"* />  </bean>    </beans> |

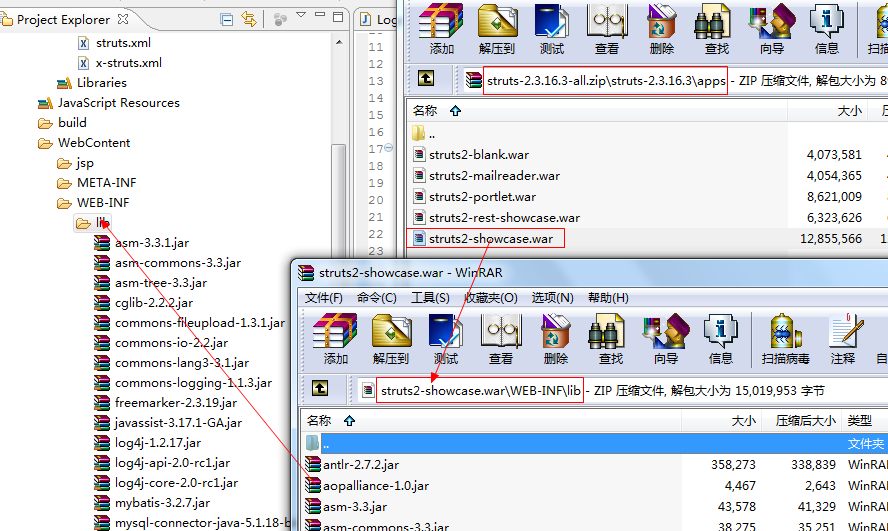
### 排错过程

此时启动tomcat会发现启动报错，开发者可根据错误提示一步一步解决问题。

|  |
| --- |
| 严重: Exception sending context initialized event to listener instance of class org.springframework.web.context.ContextLoaderListener  org.springframework.beans.factory.BeanDefinitionStoreException: Unexpected exception parsing XML document from class path resource [ssm-spring.xml]; nested exception is java.lang.NoClassDefFoundError: org/aopalliance/intercept/MethodInterceptor |

它说是没有找到MethodInterceptor这个类，说明有spring依赖资源包aopalliance.jar未导入。

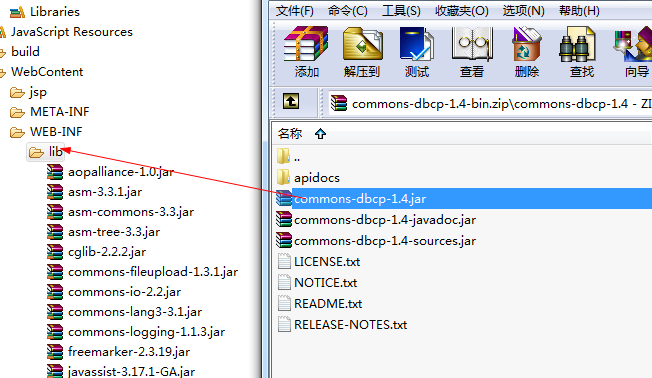
打开struts-2.3.16.3-all.zip文件的\struts-2.3.16.3\apps目录，在struts2-showcase.war文件\WEB-INF\lib目录下有该资源包，导入项目lib



再次启动tomcat报错，错误消息如下：

|  |
| --- |
| 严重: Exception sending context initialized event to listener instance of class org.springframework.web.context.ContextLoaderListener  org.springframework.beans.factory.CannotLoadBeanClassException: Cannot find class [org.apache.commons.dbcp.BasicDataSource] for bean with name 'dataSource' defined in class path resource [ssm-spring.xml]; nested exception is java.lang.ClassNotFoundException: org.apache.commons.dbcp.BasicDataSource |

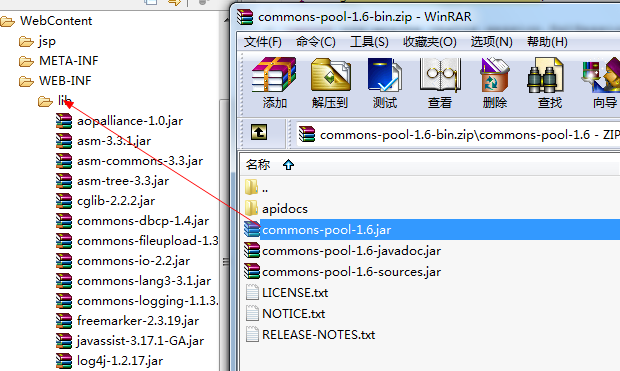
同上一步一样，有个依赖资源包未导入commons-dbcp.jar



再次启动tomcat再次报错

|  |
| --- |
| 严重: Exception sending context initialized event to listener instance of class org.springframework.web.context.ContextLoaderListener  java.lang.NoClassDefFoundError: org/apache/commons/pool/KeyedObjectPoolFactory |

依赖资源包未导入commons-pool.jar



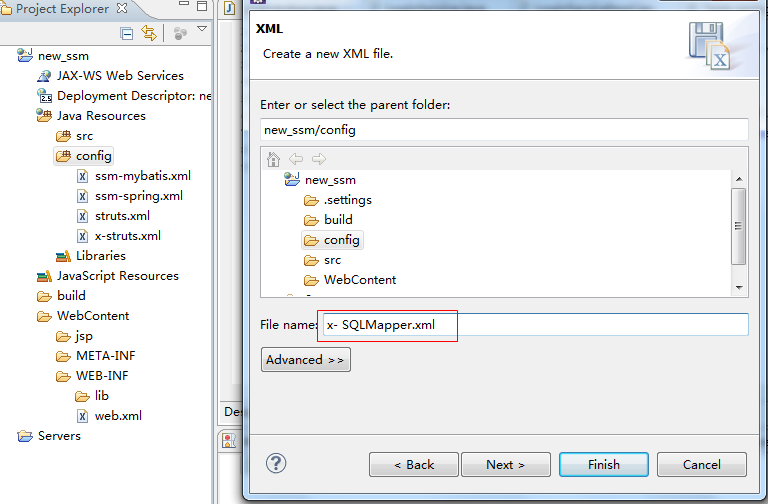
再次启动tomcat没有出现错误信息，说明ssm框架基本依赖资源包已经完成，SSM框架搭建完成，下一步编写数据处理和sql映射配置文件的使用。

### 代码部分

根据spring配置文件中的写法，编写以功能模块命名的Mybatis的SQL映射文件。ssm-spring.xml中代码片段：

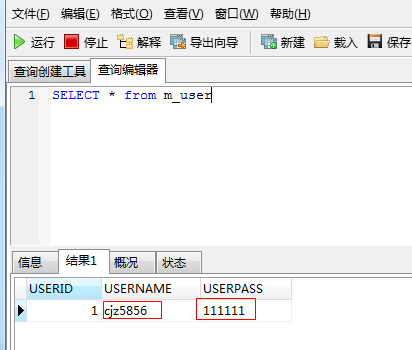
|  |
| --- |
| <!-- 定义SqlSessionFactory 使用mapperLocations，MyBatis-Spring会自动地扫描该路径下的所有Mapper文件-->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"* />  <property name=*"mapperLocations"* value=*"classpath\*:\*-SQLMapper.xml"* />  </bean> |

在config配置文件目录中新增x- SQLMapper.xml映射文件



|  |
| --- |
| <?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=*"x"*>  <select id=*"SSM\_LOGIN"* parameterType=*"userEntity"* resultType=*"userEntity"*>  select \* from m\_user  where userName = #{userName} and  userPass = #{userPass}  </select>  </mapper> |

说明：m\_user是已经建好的表，初始化有一条数据，namespace="x"按照惯例以功能模块命名，parameterType="userEntity" resultType="userEntity"参数类型和返回类型都是使用MyBatis映射配置文件中起的别名。id="SSM\_LOGIN"必须为整个项目中的唯一。



新建LoginDao.java接口类

|  |
| --- |
| **package** com.ssm.x.dao;  **import** org.springframework.stereotype.Repository;  **import** com.ssm.x.entity.UserEntity;  @Repository  **public** **interface** LoginDao {  **public** UserEntity getUserEntity(UserEntity entity);  } |

新建LoginDao.java接口实现类LoginDaoImpl.java

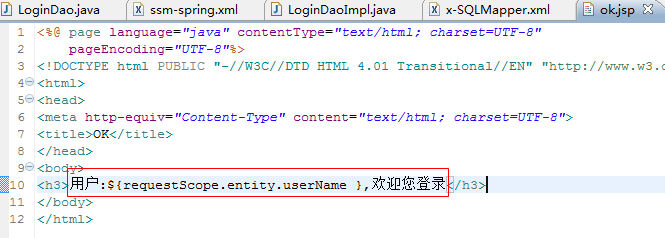
|  |
| --- |
| **package** com.ssm.x.dao.daoimpl;  **import** javax.annotation.Resource;  **import** org.apache.ibatis.session.SqlSessionFactory;  **import** org.mybatis.spring.SqlSessionTemplate;  **import** org.springframework.stereotype.Repository;  **import** com.ssm.x.dao.LoginDao;  **import** com.ssm.x.entity.UserEntity;  @Repository  **public** **class** LoginDaoImpl **implements** LoginDao {    **public** LoginDaoImpl(){  System.*out*.println("LoginDaoImpl类被调用，dao层注入spring成功");  }    //定义sql映射文件中id值  **private** **final** **static** String *SSM\_LOGIN* = "SSM\_LOGIN";    @Resource  **private** SqlSessionFactory sqlSessionFactory;    **private** SqlSessionTemplate sqlSessionTemplate;    //获取SqlSessionTemplate对象  **private** SqlSessionTemplate getSqlSessionTemplate(){  **if** (sqlSessionTemplate == **null**) {  sqlSessionTemplate = **new** SqlSessionTemplate(sqlSessionFactory);  }  **return** sqlSessionTemplate;  }    **public** UserEntity getUserEntity(UserEntity entity) {  // **TODO** Auto-generated method stub  System.*out*.println("LoginDaoImpl类getUserEntity方法被调用，用户名："+entity.getUserName()+" 密码："+entity.getUserPass());  **return** getSqlSessionTemplate().selectOne(*SSM\_LOGIN*, entity);  }  } |

修改LoginServiceImpl.java类，将

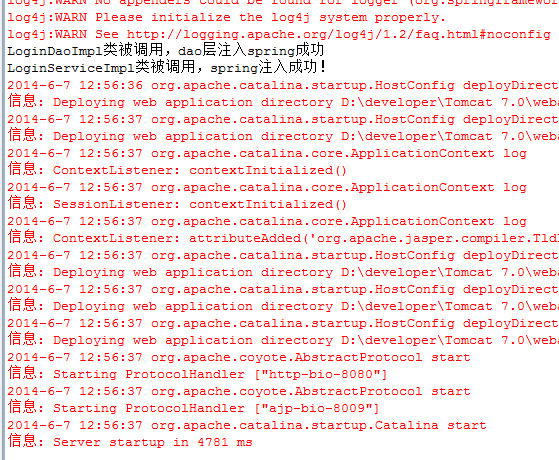
|  |
| --- |
| **package** com.ssm.x.service.serviceimpl;  **import** javax.annotation.Resource;  **import** org.springframework.stereotype.Service;  **import** com.ssm.x.dao.LoginDao;  **import** com.ssm.x.entity.UserEntity;  **import** com.ssm.x.service.LoginService;  @Service  **public** **class** LoginServiceImpl **implements** LoginService {  **public** LoginServiceImpl(){  System.*out*.println("LoginServiceImpl类被调用，spring注入成功！");  }    @Resource  **private** LoginDao loginDao;  **public** **boolean** isLoginSuccess(UserEntity entity) {  // **TODO** Auto-generated method stub  System.*out*.println("LoginServiceImpl类isLoginSuccess方法被调用，spring注入成功！");  **if** (loginDao.getUserEntity(entity) != **null**) {  **return** **true**;  } **else** {  **return** **false**;  }  }  } |

修改ok.jsp

|  |
| --- |
| <h3>用户:${requestScope.entity.userName },欢迎您登录</h3> |



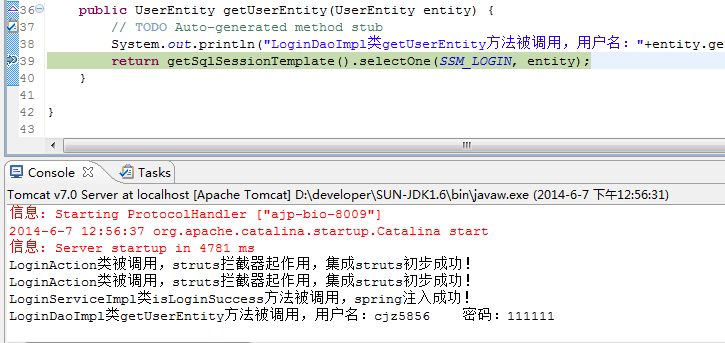
启动tomcat

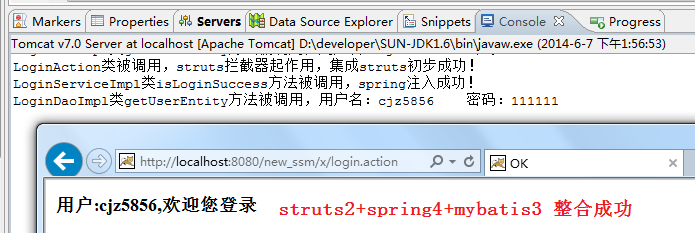


### 最终结果

登录页面测试结果

debug模式下看到控制台输出结果。





至此搭建框架成功，访问数据库也成功。增删改查 操作自行增加，本文档仅供初学者避免走弯路。

QQ：332440325

2014年6月7日