1. **关于maven创建Web项目方式**
2. 执行如下命令：

mvn archetype:create -DarchetypeCatalog=local -DgroupId=cn.xzg -DartifactId=spring4\_mybatis3 -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

解释：mvn archetype:generate 这样就可以根据提示来建立一个maven项目

      后面可以接很多参数，常用的有：

        -DgroupId= cn.xzg 组id

        -DartifactId= spring4\_mybatis3项目名称，maven会根据这个名称在当前目录下新建一个名为该名称的目录用于建立项目

        -DinteractiveMode=false  是否已交互模式进行，如果是false的话就会采用默认设置建立项目

但是：

用maven构建项目骨架时，速度非常慢

原因：

要了解其中的原因，需要知道maven是如何运作的。构建过程中，我们只是向maven提供了一些groupId、artifactId、archetypeArtifactId等信息，只有这些信息的情况下maven是如何构建出整个项目的呢？它的知识来源于哪里呢？答案是：

**[html]** [view plain](http://blog.csdn.net/codercaicai/article/details/51917522) [copy](http://blog.csdn.net/codercaicai/article/details/51917522)

1. http://repo1.maven.org/maven2/archetype-catalog.xml

这里记录了，所有archetype构建信息，maven通过archetype插件构建项目骨架时会远程访问页面，通过maven的-X参数可以证明这一点。复制文件到编辑器可知，文件长达13W行，文件大加上网络原因，导致访问速度很慢。

解决方案：

如上分析可知，是由于achetype-catalog.xml文件的访问问题，导致了整个构建过程的缓慢，所以是否能够将文件保存到本地，成为一种解决思路。翻阅Maven官方文档可以找到，确实是可以的。官方链接如下：

1. http://maven.apache.org/archetype/maven-archetype-plugin/specification/archetype-catalog.html

按照如下步骤：

      1）准备archetype-catalog.xml文件

            方式1：通过浏览器查看http://repo1.maven.org/maven2/archetype-catalog.xml页源码，复制到本地，命名为archetype-catalog.xml。

            方式2：通过命令行  curl http://repo1.maven.org/maven2/archetype-catalog.xml > archetype-catalog.xml

      2）将上述文件放置到maven的默认路径下

            注意，这不是指的安装路径，而是mvn运行时默认的存放repository的路径，一般在用户根目录下的一个隐藏目录，~/.m2。如果做过更改，可以在maven的设置文件中查看具体在哪个位置，设置文件在maven安装目录/config/settings.xml中。

      3）在构建时，在archetype:generate后加上 -DarchetypeCatalog=local参数。（IDE如eclipse和idea，都是可以在创建maven工程时设置构建参数的，这里不再截图说明了）

解决，Have fun!再次尝试构建项目骨架，飞一般的感觉。

部分maven命令：

通过archetype插件构建Maven项目骨架目录

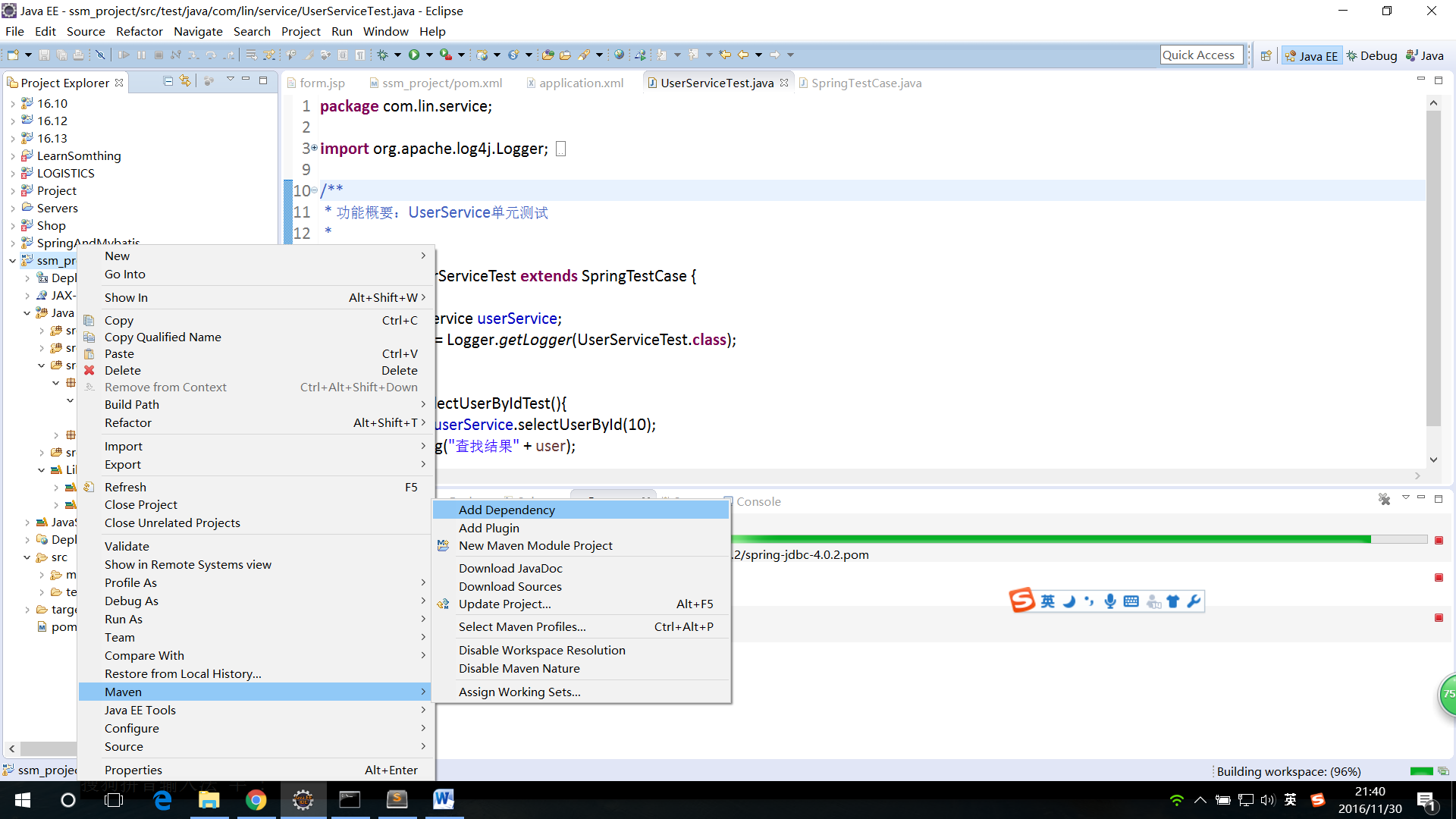
（4）Maven的常用命令

Maven常用命令：   
1. 创建Maven的普通java项目：   
mvn archetype:create -DgroupId=packageName -DartifactId=projectName   
2. 创建Maven的Web项目：   
mvn archetype:create -DgroupId=packageName -DartifactId=webappName-DarchetypeArtifactId=maven-archetype-webapp   
3. 编译源代码： mvn compile   
4. 编译测试代码：mvn test-compile   
5. 运行测试：mvn test   
6. 产生site：mvn site   
7. 打包：mvn package   
8. 在本地Repository中安装jar：mvn install   
9. 清除产生的项目：mvn clean   
10. 生成eclipse项目：mvn eclipse:eclipse   
11. 生成idea项目：mvn idea:idea   
12. 组合使用goal命令，如只打包不测试：mvn -Dtest package   
13. 编译测试的内容：mvn test-compile   
14. 只打jar包: mvn jar:jar   
15. 只测试而不编译，也不测试编译：mvn test -skipping compile -skipping test-compile   
( -skipping 的灵活运用，当然也可以用于其他组合命令)   
16. 清除eclipse的一些系统设置:mvn eclipse:clean

Eclipse使用Maven创建Web时找不到maven-archetype-webap或报  
Could not resolve archetype org.apache.maven.archetypes:maven-archetype-webapp:RELEASE from any of the configured repositories  
解决方法：  
window-->preferences-->Maven-->Archetypes-->Add Remote Catalog中添加  
catalog File:<http://repo1.maven.org/maven2/archetype-catalog.xml>  
Description: 随便填写  
  
2.为Eclipse添加自己的maven环境  
 选择window-->preferences-->Maven-->Installations-->Add  
选择本地maven安装目录，-->Apply  
选择User Settings-->选择本地maven安装目录下conf-->settings.xml-->ok  
重启Eclipse。  
  
3.修改本地maven库地址  
在settings.xml中<settings ...></settings>添加<localRepository>E:/localMaven/</localRepository>  
  
4.启用jetty-run命令  
settings.xml中添加<pluginGroup>org.mortbay.jetty</pluginGroup>  
  
5.规范maven库  
settings.xml中的<profiles></profiles>增加<repositories><repository><id></id><url>可以为局域网地址</url></repository></repositoories>  
  
6.安装jar包到本地maven库  
mvn install:install-file -Dfile=E:\jodconverter-2.2.2.jar -DgroupId=com.arto  
fsolving -DartifactId=jodconverter -Dversion=2.2.2 -Dpackaging=jar -DgeneratePom  
=true -DcreateChecksum=true  
  
7.maven 关联本地项目jar包  
<dependency>  
 <groupId>com.artofsolving</groupId>  
 <artifactId>jodconverter-core</artifactId>  
 <version>3.0-beta-4</version>  
 <scope>system</scope>  
 <systemPath>${basedir}/lib/jodconverter-core-3.0-beta-4.jar</systemPath>   
 </dependency>  
lib目录必须与pom.xml同级  
   
8.can not calculate build plan:  
选中项目右键—》maven-->Update Dependencies.

9、有时候编写在pom文件中的jar包斌没有加载到工程中去，这是就需要手动

添加，方法如下：手动添加



**使用maven创建简单ssm项目**

1. 使用mysql创建数据表结构

链接数据库D:\wamp\bin\mysql\mysql5.7.11\bin>mysql -h localhost -u xzg -p123

CREATE TABLE t\_user (

USER\_ID int(11) NOT NULL AUTO\_INCREMENT,

USER\_NAME char(30) NOT NULL,

USER\_PASSWORD char(10) NOT NULL,

USER\_EMAIL char(30) NOT NULL,

PRIMARY KEY (USER\_ID),

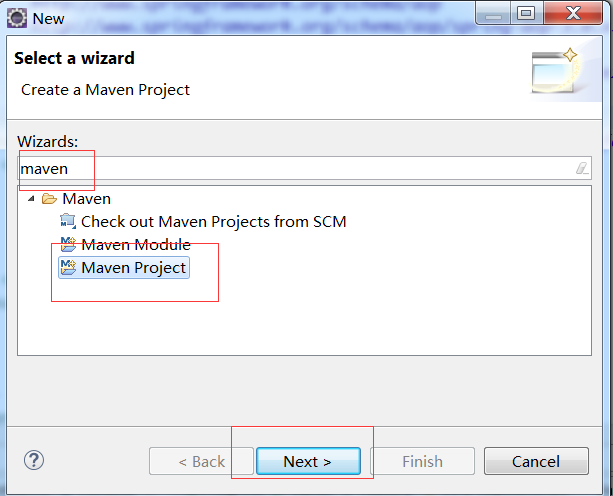
KEY IDX\_NAME (USER\_NAME)

) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8

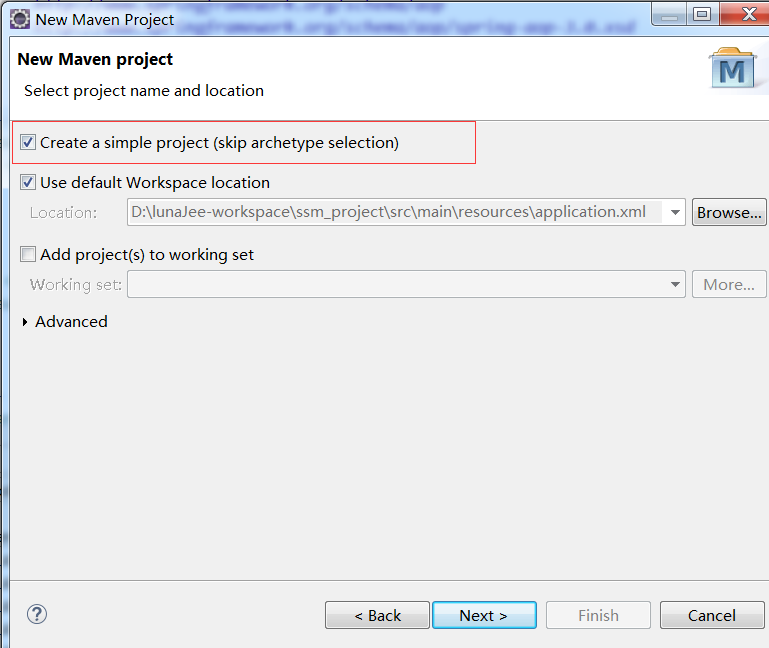
# 2、二、工程创建

1、Maven工程创建

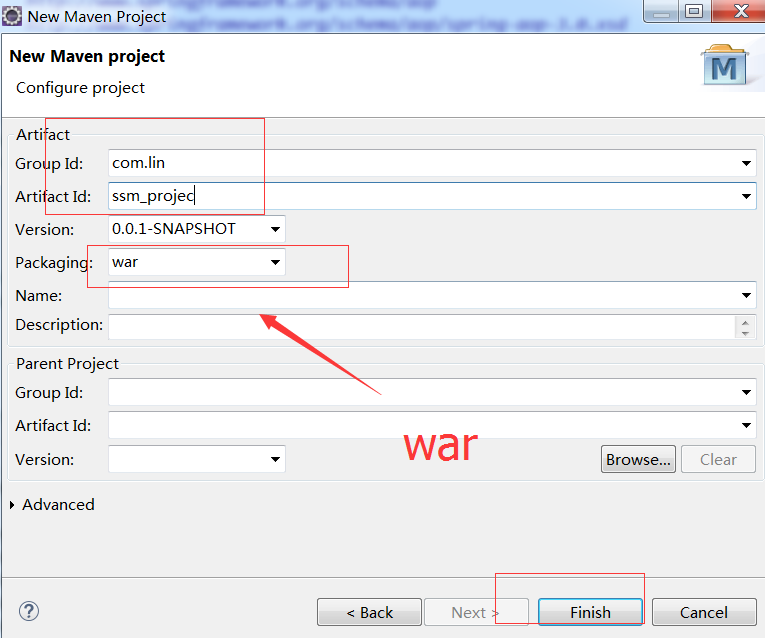
（1）新建

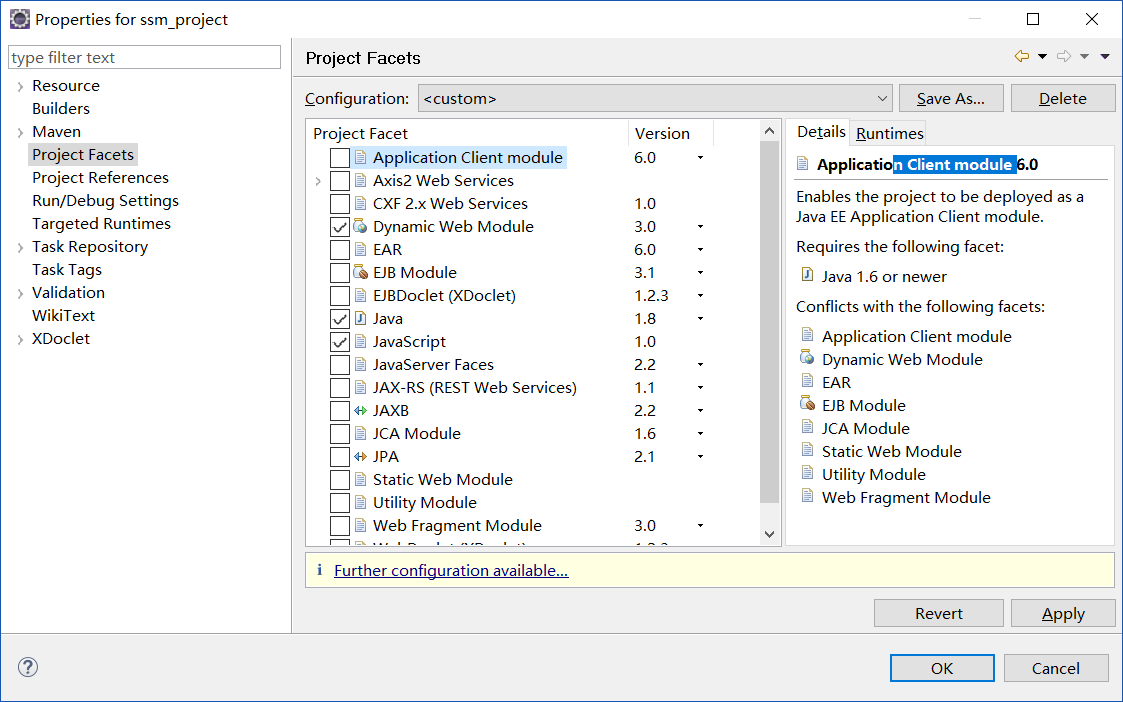


（2）选择快速框架



（3）输出项目名，包，记得选war(表示web项目，以后可以spingMVC连起来用)

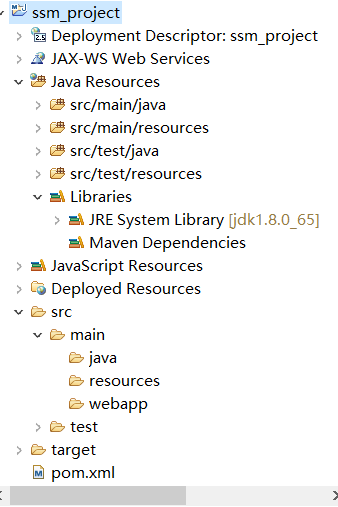
  
修改项目,且保持jdk版本保持一致：





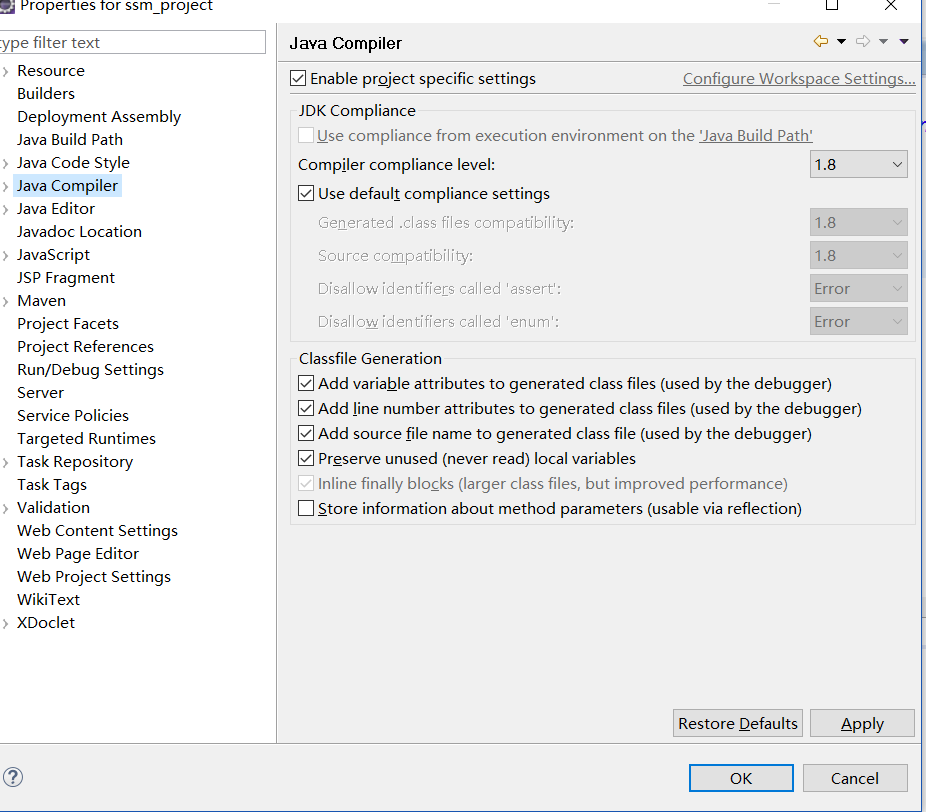
（4）创建好之后

目录如下：

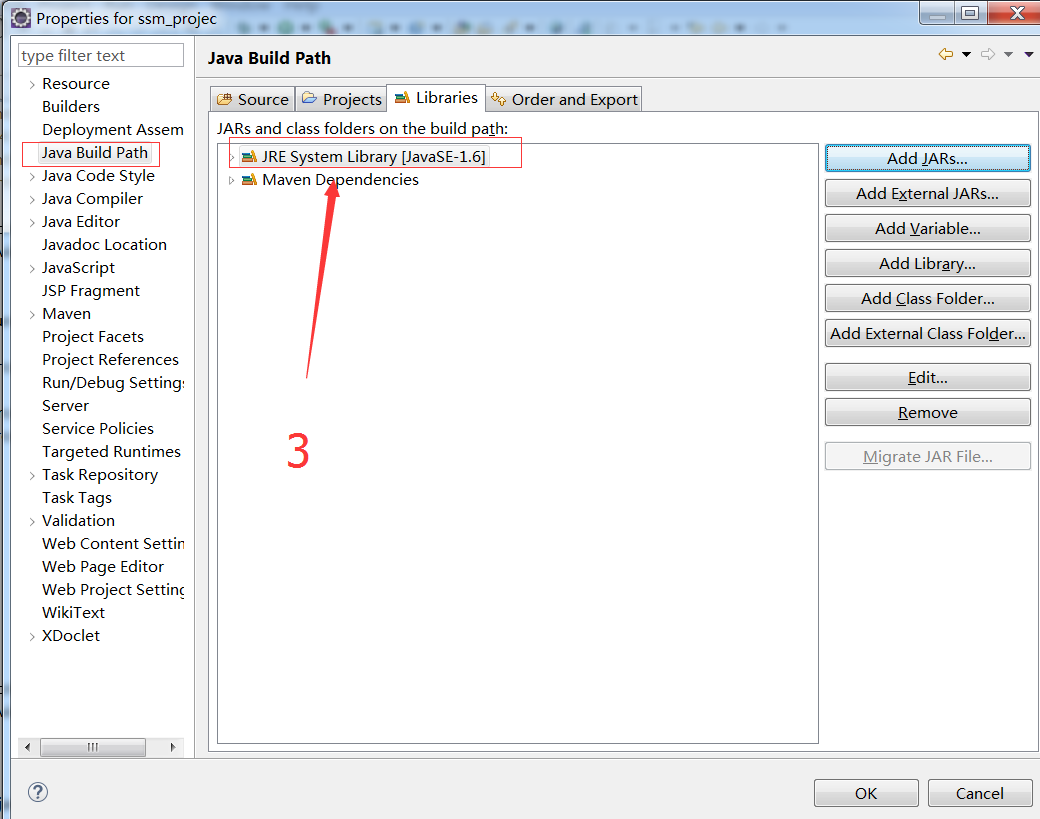


（5）检查下

这三个地方JDK的版本一定要一样！！！！

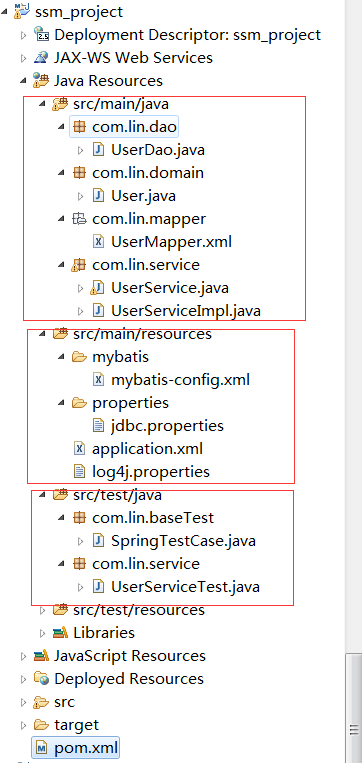






# 三、sping+mybatis配置

1、整个工程目录如下：



2、POM文件

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.li</groupId>

<artifactId>ssm\_project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<project.build.sourceEncoding>

UTF-8

</project.build.sourceEncoding>

<!-- spring版本号 -->

<spring.version>4.0.2.RELEASE</spring.version>

<!-- log4j日志文件管理包版本 -->

<slf4j.version>1.7.5</slf4j.version>

<log4j.version>1.2.17</log4j.version>

<!-- mybatis版本号 -->

<mybatis.version>3.2.6</mybatis.version>

<!-- activiti版本号-->

<activiti.version>5.17.0</activiti.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<!-- 表示开发的时候引入，发布的时候不会加载此包 -->

<scope>test</scope>

</dependency>

<!-- spring核心包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- mybatis核心包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<!-- mybatis/spring包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<!-- 导入Mysql数据库链接jar包 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.35</version>

</dependency>

<!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.2.2</version>

</dependency>

<!-- JSTL标签类 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- 日志文件管理包 -->

<!-- log start -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<!-- 格式化对象，方便输出日志 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.1.41</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- log end -->

<!--spring单元测试依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

<scope>test</scope>

</dependency>

<!-- 映入JSON -->

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<!-- activiti依赖包 -->

<dependency>

<groupId>org.activiti</groupId>

<artifactId>activiti-engine</artifactId>

<version>${activiti.version}</version>

</dependency>

<!-- activiti 与 Spring 集成 -->

<dependency>

<groupId>org.activiti</groupId>

<artifactId>activiti-spring</artifactId>

<version>${activiti.version}</version>

</dependency>

<!-- 1.1 使用REST的好处

简单化：利用现有模块（activiti-rest.war）代替直接API调用

标准化：各个系统根据rest模块的接口规范访问REST资源，统一处理；对于工作流平台来说此特性尤为突出

扩展性：如果官方提供的REST接口还不能满足可以继续在其基础上进行扩展以满足业务系统（平台）的需求

1.2 不适合使用REST的场景

业务数据与流程数据分离:就像kft-activiti-demo中普通表单的演示一样，

业务数据保存在一张单独设计的表中，而不是把表单数据保存在引擎的变量表中，

所以对于这样的场景中需要联合事务管理的就不能使用REST了，例如：启动流程、任务完成、业务与流程数据联合查询。 -->

<dependency>

<groupId>org.activiti</groupId>

<artifactId>activiti-rest</artifactId>

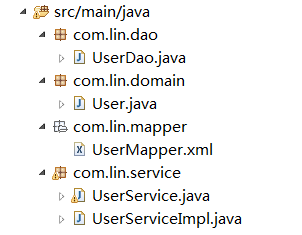
<version>${activiti.version}</version>

</dependency> </dependencies>

</project>

3、java代码-------src/main/java

目录如下：



（1）User.[**Java**](http://lib.csdn.net/base/javaee)

对应**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**中表的字段，放在src/main/java下的包com.lin.domain

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **package** com.lin.domain;
3. /\*\*
4. \* User映射类
5. \*/
6. **public** **class** User {
7. **private** Integer userId;
8. **private** String userName;
9. **private** String userPassword;
10. **private** String userEmail;
12. **public** Integer getUserId() {
13. **return** userId;
14. }
16. **public** **void** setUserId(Integer userId) {
17. **this**.userId = userId;
18. }
20. **public** String getUserName() {
21. **return** userName;
22. }
24. **public** **void** setUserName(String userName) {
25. **this**.userName = userName;
26. }
28. **public** String getUserPassword() {
29. **return** userPassword;
30. }
32. **public** **void** setUserPassword(String userPassword) {
33. **this**.userPassword = userPassword;
34. }
36. **public** String getUserEmail() {
37. **return** userEmail;
38. }
40. **public** **void** setUserEmail(String userEmail) {
41. **this**.userEmail = userEmail;
42. }
44. @Override
45. **public** String toString() {
46. **return** "User [userId=" + userId + ", userName=" + userName
47. + ", userPassword=" + userPassword + ", userEmail=" + userEmail
48. + "]";
49. }
51. }

（2）UserDao.java

Dao接口类，用来对应mapper文件。放在src/main/java下的包com.lin.dao,内容如下：

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **package** com.lin.dao;

4. **import** com.lin.domain.User;
6. /\*\*
7. \* 功能概要：User的DAO类
8. \*
9. \*/
10. **public** **interface** UserDao {
11. /\*\*
12. \*
13. \* @return
14. \*/
15. **public** User selectUserById(Integer userId);
17. }

（2）UserService.java和UserServiceImpl.java

service接口类和实现类，放在src/main/java下的包com.lin.service,内容如下：

UserService.java

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **package** com.lin.service;
3. **import** org.springframework.stereotype.Service;
5. **import** com.lin.domain.User;
7. /\*\*
8. \* 功能概要：UserService接口类
9. \*
10. \*/
11. **public** **interface** UserService {
12. User selectUserById(Integer userId);
14. }

UserServiceImpl.java

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **package** com.lin.service;
3. **import** org.springframework.beans.factory.annotation.Autowired;
4. **import** org.springframework.stereotype.Service;
6. **import** com.lin.dao.UserDao;
7. **import** com.lin.domain.User;
9. /\*\*
10. \* 功能概要：UserService实现类
11. \*
12. \* @author linbingwen
13. \* @since  2015年9月28日
14. \*/
15. @Service
16. **public** **class** UserServiceImpl **implements** UserService{
17. @Autowired
18. **private** UserDao userDao;
20. **public** User selectUserById(Integer userId) {
21. **return** userDao.selectUserById(userId);
23. }
25. }

（4）mapper文件

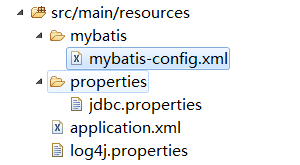
用来和dao文件对应，放在src/main/java下的com.lin.mapper包下

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3. "http://mybatis.org/dtd/mybatis-3-mapper.dtd"**>**
4. **<mapper** namespace="com.lin.dao.UserDao"**>**
5. <!--设置domain类和数据库中表的字段一一对应，注意数据库字段和domain类中的字段名称不致，此处一定要！-->
6. **<resultMap** id="BaseResultMap" type="com.lin.domain.User"**>**
7. **<id** column="USER\_ID" property="userId" jdbcType="INTEGER" **/>**
8. **<result** column="USER\_NAME" property="userName" jdbcType="CHAR" **/>**
9. **<result** column="USER\_PASSWORD" property="userPassword" jdbcType="CHAR" **/>**
10. **<result** column="USER\_EMAIL" property="userEmail" jdbcType="CHAR" **/>**
11. **</resultMap>**
12. <!-- 查询单条记录 -->
13. **<select** id="selectUserById" parameterType="int" resultMap="BaseResultMap"**>**
14. SELECT \* FROM t\_user WHERE USER\_ID = #{userId}
15. **</select>**
16. **</mapper>**

4、资源配置-------src/main/resources

目录如下：



(1)mybatis配置文件

这里没有什么内容，因为都被放到application.xml中去了,放在src/main/resources下的mybatis文件夹下

mybatis-config.xml内容如下：

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3. "http://mybatis.org/dtd/mybatis-3-config.dtd"**>**
4. **<configuration>**
5. **</configuration>**

（2）数据源配置jdbc.properties

放在src/main/resources下的propertiesy文件夹下

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

jdbc\_driverClassName=com.mysql.jdbc.Driver

jdbc\_url=jdbc:mysql://localhost:3307/xzg?useUnicode=true&characterEncoding=utf8

jdbc\_username=xzg

jdbc\_password=123

（3）Spring配置

这是最重要的：application.xml内容如下

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
4. xmlns:aop="http://www.springframework.org/schema/aop"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8. http://www.springframework.org/schema/aop
9. http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
10. http://www.springframework.org/schema/context
11. http://www.springframework.org/schema/context/spring-context-3.0.xsd"**>**
13. <!-- 引入jdbc配置文件 -->
14. **<bean** id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"**>**
15. **<property** name="locations"**>**
16. **<list>**
17. **<value>**classpath:properties/\*.properties**</value>**
18. <!--要是有多个配置文件，只需在这里继续添加即可 -->
19. **</list>**
20. **</property>**
21. **</bean>**


25. <!-- 配置数据源 -->
26. **<bean** id="dataSource"
27. class="org.springframework.jdbc.datasource.DriverManagerDataSource"**>**
28. <!-- 不使用properties来配置 -->
29. <!-- **<property** name="driverClassName" value="com.mysql.jdbc.Driver" **/>**
30. **<property** name="url" value="jdbc:mysql://localhost:3306/learning" **/>**
31. **<property** name="username" value="root" **/>**
32. **<property** name="password" value="christmas258@" **/>** --**>**
33. <!-- 使用properties来配置 -->
34. **<property** name="driverClassName"**>**
35. **<value>**${jdbc\_driverClassName}**</value>**
36. **</property>**
37. **<property** name="url"**>**
38. **<value>**${jdbc\_url}**</value>**
39. **</property>**
40. **<property** name="username"**>**
41. **<value>**${jdbc\_username}**</value>**
42. **</property>**
43. **<property** name="password"**>**
44. **<value>**${jdbc\_password}**</value>**
45. **</property>**
46. **</bean>**
48. <!-- 自动扫描了所有的XxxxMapper.xml对应的mapper接口文件，这样就不用一个一个手动配置Mpper的映射了，只要Mapper接口类和Mapper映射文件对应起来就可以了。 -->
49. **<bean** class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>**
50. **<property** name="basePackage"
51. value="com.lin.dao" **/>**
52. **</bean>**
54. <!-- 配置Mybatis的文件 ，mapperLocations配置\*\*Mapper.xml文件位置，configLocation配置mybatis-config文件位置-->
55. **<bean** id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>**
56. **<property** name="dataSource" ref="dataSource" **/>**
57. **<property** name="mapperLocations" value="classpath\*:com/lin/mapper/\*\*/\*.xml"**/>**
58. **<property** name="configLocation" value="classpath:mybatis/mybatis-config.xml" **/>**
59. <!-- **<property** name="typeAliasesPackage" value="com.tiantian.ckeditor.model"
60. **/>** --**>**
61. **</bean>**
63. <!-- 自动扫描注解的bean -->
64. **<context:component-scan** base-package="com.lin.service" **/>**
66. **</beans>**

（4）日志打印log4j.properties

就放在src/main/resources

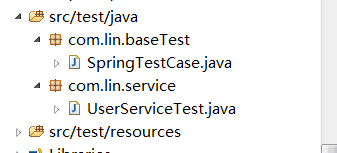
**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. log4j.rootLogger=DEBUG,Console,Stdout
3. #Console
4. log4j.appender.Console=org.apache.log4j.ConsoleAppender
5. log4j.appender.Console.layout=org.apache.log4j.PatternLayout
6. log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n
8. log4j.logger.java.sql.ResultSet=INFO
9. log4j.logger.org.apache=INFO
10. log4j.logger.java.sql.Connection=DEBUG
11. log4j.logger.java.sql.Statement=DEBUG
12. log4j.logger.java.sql.PreparedStatement=DEBUG
14. log4j.appender.Stdout = org.apache.log4j.DailyRollingFileAppender
15. log4j.appender.Stdout.File = E://logs/log.log
16. log4j.appender.Stdout.Append = true
17. log4j.appender.Stdout.Threshold = DEBUG
18. log4j.appender.Stdout.layout = org.apache.log4j.PatternLayout
19. log4j.appender.Stdout.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss}  [ %t:%r ] - [ %p ]  %m%n

# 四、单元测试

上面的配置完好，接下来就是测验成功

整个目录 如下：



（1）[**测试**](http://lib.csdn.net/base/softwaretest)基类

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **package** com.lin.baseTest;
3. **import** org.junit.runner.RunWith;
4. **import** org.slf4j.Logger;
5. **import** org.slf4j.LoggerFactory;
6. **import** org.springframework.test.context.ContextConfiguration;
7. **import** org.springframework.test.context.junit4.AbstractJUnit4SpringContextTests;
8. **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
10. /\*\*
11. \* 功能概要：
12. \*
13. \* @author linbingwen
14. \* @since  2015年9月28日
15. \*/
16. //指定bean注入的配置文件
17. @ContextConfiguration(locations = { "classpath:application.xml" })
18. //使用标准的JUnit @RunWith注释来告诉JUnit使用Spring TestRunner
19. @RunWith(SpringJUnit4ClassRunner.**class**)
20. **public** **abstract** **class** SpringTestCase **extends** AbstractJUnit4SpringContextTests{
21. **protected** Logger logger = LoggerFactory.getLogger(getClass());
22. }

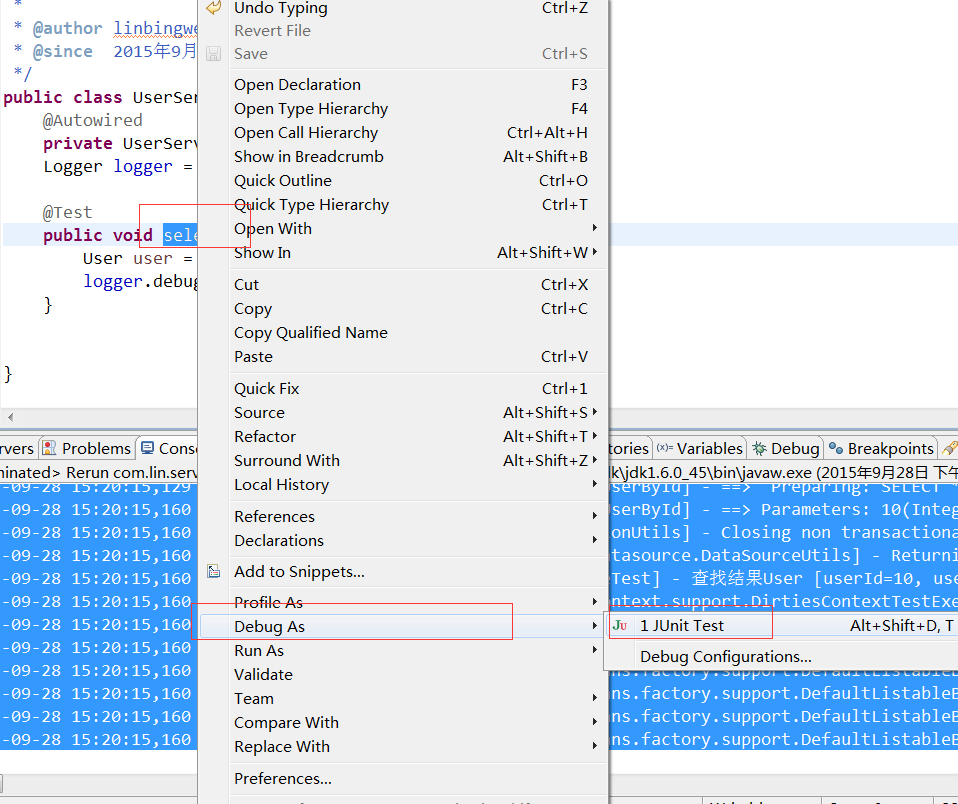
（2）测试类

**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

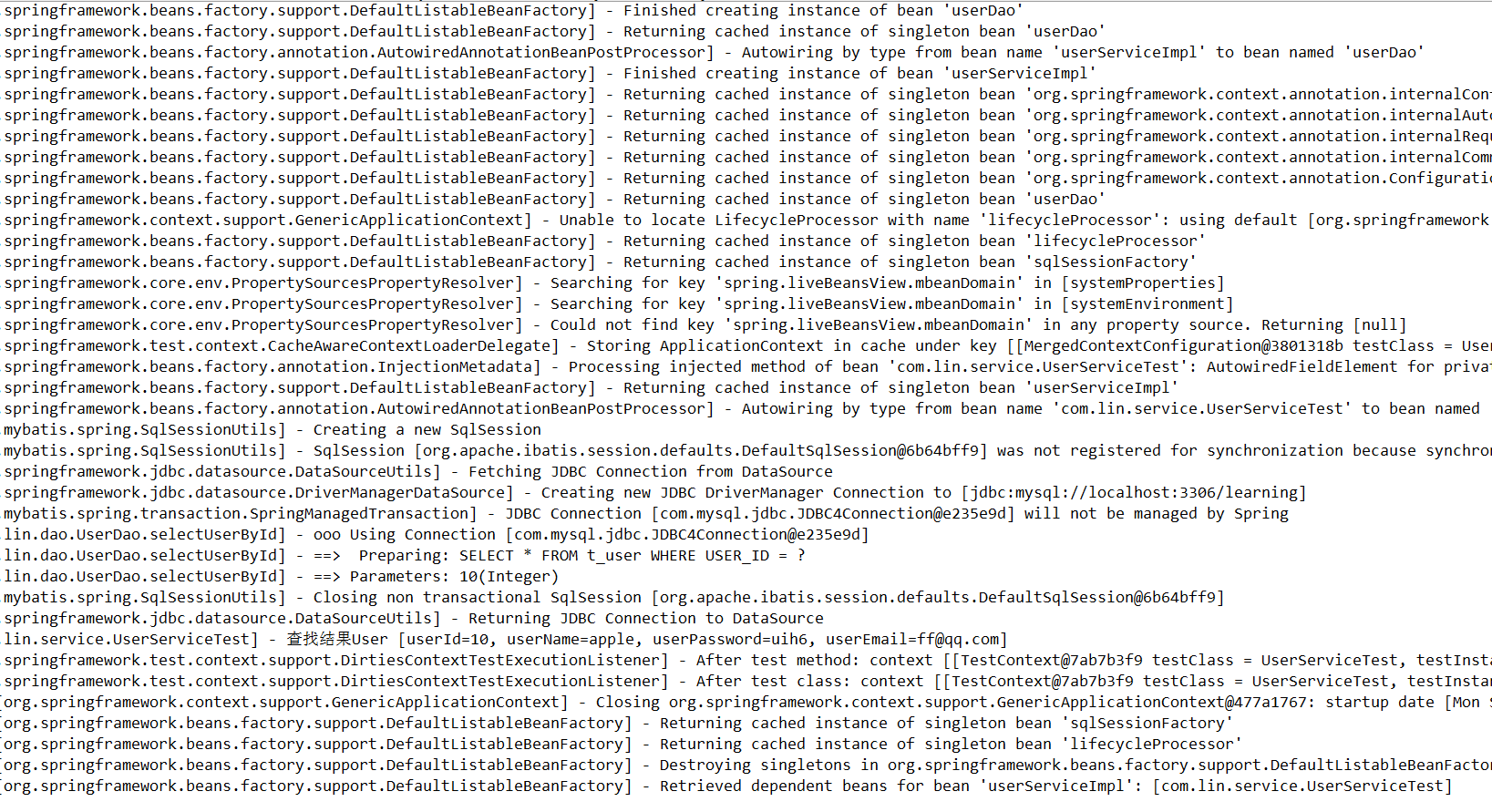
1. **package** com.lin.service;
3. **import** org.apache.log4j.Logger;
4. **import** org.junit.Test;
5. **import** org.springframework.beans.factory.annotation.Autowired;
7. **import** com.lin.baseTest.SpringTestCase;
8. **import** com.lin.domain.User;
10. /\*\*
11. \* 功能概要：UserService单元测试
12. \*
13. \* @author linbingwen
14. \* @since  2015年9月28日
15. \*/
16. **public** **class** UserServiceTest **extends** SpringTestCase {
17. @Autowired
18. **private** UserService userService;
19. Logger logger = Logger.getLogger(UserServiceTest.**class**);
21. @Test
22. **public** **void** selectUserByIdTest(){
23. User user = userService.selectUserById(10);
24. logger.debug("查找结果" + user);
25. }

28. }

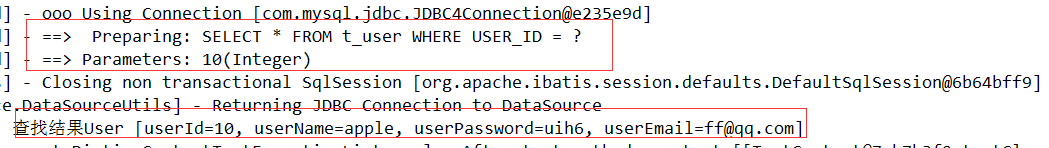
选中selectUserByIdTest，然后右键如下运行



输出结果：



重要打印的结果

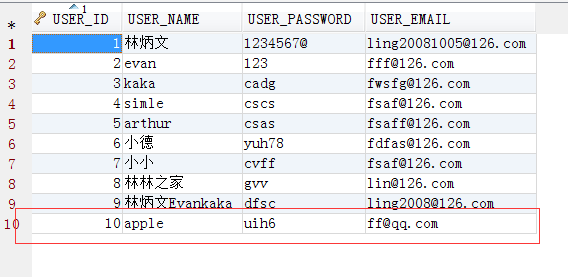


这里

**[plain]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. 2015-09-28 15:20:15,129 [main] DEBUG [com.lin.dao.UserDao.selectUserById] - ==>  Preparing: SELECT \* FROM t\_user WHERE USER\_ID = ?
2. 2015-09-28 15:20:15,160 [main] DEBUG [com.lin.dao.UserDao.selectUserById] - ==> Parameters: 10(Integer)
3. 2015-09-28 15:20:15,160 [main] DEBUG [org.mybatis.spring.SqlSessionUtils] - Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@6b64bff9]
4. 2015-09-28 15:20:15,160 [main] DEBUG [org.springframework.jdbc.datasource.DataSourceUtils] - Returning JDBC Connection to DataSource
5. 2015-09-28 15:20:15,160 [main] DEBUG [com.lin.service.UserServiceTest] - 查找结果User [userId=10, userName=apple, userPassword=uih6, userEmail=ff@qq.com]

数据库：



程序成功运行，并且结果正确！

到这里配置好spring+mybatis+mysql!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

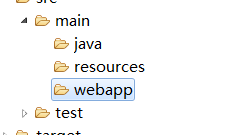
**=====================================================**

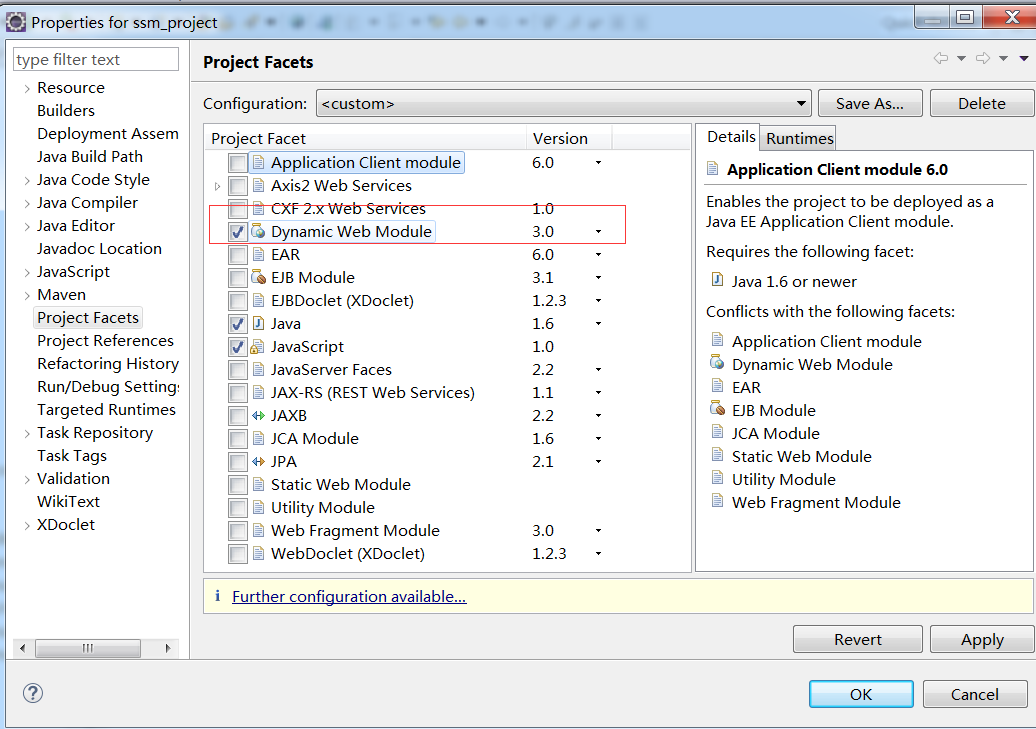
这里开始添加spirngMVC，并转换maven工程为web项目

# 五、转换web工程

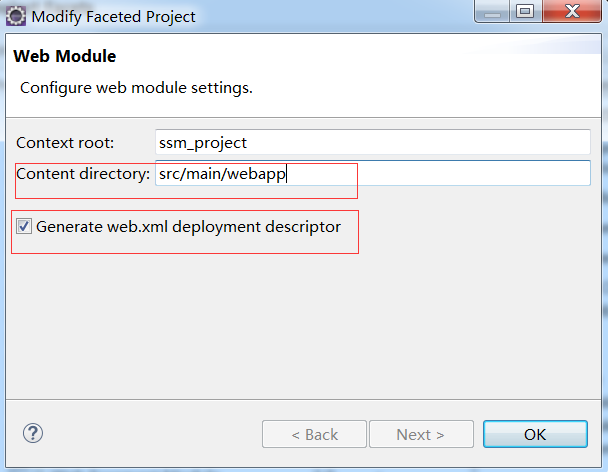
接上面的工程接着说，此时要加spingMVC之前，工程得再稍微转换下

此时webapp下的结果还没有显示出来，因为此时我们还没有配置此的项目为web项目

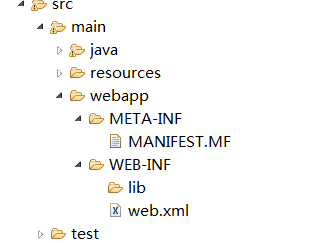


再次进去Properties配置，如下图所示：如果没有显示如下，那就先把dynamic web module的√给去掉，然后重新再打开 这个

进入，然后输入如下内容



确定OK之后,webapp下面就会自动生成一些文件，如下



# 六、配置SpringMVC

（1）修改POM文件，增加了一此内容，新的POM文件如下：

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.li</groupId>

<artifactId>ssm\_project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<project.build.sourceEncoding>

UTF-8

</project.build.sourceEncoding>

<!-- spring版本号 -->

<spring.version>4.0.2.RELEASE</spring.version>

<!-- log4j日志文件管理包版本 -->

<slf4j.version>1.7.5</slf4j.version>

<log4j.version>1.2.17</log4j.version>

<!-- mybatis版本号 -->

<mybatis.version>3.2.6</mybatis.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<!-- 表示开发的时候引入，发布的时候不会加载此包 -->

<scope>test</scope>

</dependency>

<!-- spring核心包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- mybatis核心包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<!-- mybatis/spring包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<!-- 导入Mysql数据库链接jar包 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.35</version>

</dependency>

<!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.2.2</version>

</dependency>

<!-- JSTL标签类 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- 日志文件管理包 -->

<!-- log start -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<!-- 格式化对象，方便输出日志 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.1.41</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- log end -->

<!--spring单元测试依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

<scope>test</scope>

</dependency>

<!-- 映入JSON -->

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<!-- javaee-api包 注意和项目使用的JDK版本对应 -->

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

<scope>provided</scope>

</dependency>

<!-- javaee-web-api包 注意和项目使用的JDK版本对应 -->

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-web-api</artifactId>

<version>7.0</version>

<scope>provided</scope>

</dependency>

<!-- 上传组件包 -->

<!-- <dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.9</version>

</dependency> -->

</dependencies>

</project>其实也就增加了下面两个

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. <!-- javaee-api包 注意和项目使用的JDK版本对应 -->
2. **<dependency>**
3. **<groupId>**javax**</groupId>**
4. **<artifactId>**javaee-api**</artifactId>**
5. **<version>**6.0**</version>**
6. **<scope>**provided**</scope>**
7. **</dependency>**
9. <!-- javaee-web-api包 注意和项目使用的JDK版本对应 -->
10. **<dependency>**
11. **<groupId>**javax**</groupId>**
12. **<artifactId>**javaee-web-api**</artifactId>**
13. **<version>**6.0**</version>**
14. **<scope>**provided**</scope>**
15. **</dependency>**

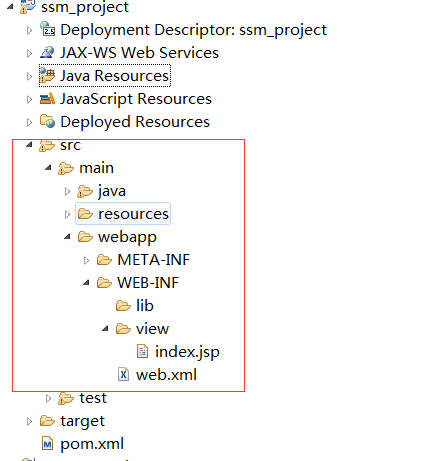
(2) 在src/main/resource中添加springmvc文件夹，然后添加文件spring-mvc.xml，内容如下：

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:p="http://www.springframework.org/schema/p"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xmlns:context="http://www.springframework.org/schema/context"
6. xmlns:mvc="http://www.springframework.org/schema/mvc"
7. xsi:schemaLocation="
8. http://www.springframework.org/schema/beans
9. http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
10. http://www.springframework.org/schema/context
11. http://www.springframework.org/schema/context/spring-context-3.2.xsd
12. http://www.springframework.org/schema/mvc
13. http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd"**>**
15. <!-- 扫描controller（controller层注入） -->
16. **<context:component-scan** base-package="com.lin.controller"**/>**
18. <!-- 对模型视图添加前后缀 -->
19. **<bean** id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"
20. p:prefix="/WEB-INF/view/" p:suffix=".jsp"**/>**
21. **</beans>**

（3）配置web,xml

目录如下 ：



这里是关键，web.xml是用来起动sping、spingMVC

放在src/main/webapp/WEB-INF下。内容如下：

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"
5. id="WebApp\_ID" version="2.5"**>**
6. **<display-name>**Archetype Created Web Application**</display-name>**
7. <!-- 起始欢迎界面 -->
8. **<welcome-file-list>**
9. **<welcome-file>**index.jsp**</welcome-file>**
10. **</welcome-file-list>**
12. <!-- 读取spring配置文件 -->
13. **<context-param>**
14. **<param-name>**contextConfigLocation**</param-name>**
15. **<param-value>**classpath:application.xml**</param-value>**
16. **</context-param>**
17. <!-- 设计路径变量值 -->
18. **<context-param>**
19. **<param-name>**webAppRootKey**</param-name>**
20. **<param-value>**springmvc.root**</param-value>**
21. **</context-param>**

24. <!-- Spring字符集过滤器 -->
25. **<filter>**
26. **<filter-name>**SpringEncodingFilter**</filter-name>**
27. **<filter-class>**org.springframework.web.filter.CharacterEncodingFilter**</filter-class>**
28. **<init-param>**
29. **<param-name>**encoding**</param-name>**
30. **<param-value>**UTF-8**</param-value>**
31. **</init-param>**
32. **<init-param>**
33. **<param-name>**forceEncoding**</param-name>**
34. **<param-value>**true**</param-value>**
35. **</init-param>**
36. **</filter>**
37. **<filter-mapping>**
38. **<filter-name>**SpringEncodingFilter**</filter-name>**
39. **<url-pattern>**/\***</url-pattern>**
40. **</filter-mapping>**
42. <!-- 日志记录 -->
43. **<context-param>**
44. <!-- 日志配置文件路径 -->
45. **<param-name>**log4jConfigLocation**</param-name>**
46. **<param-value>**classpath:log4j.properties**</param-value>**
47. **</context-param>**
48. **<context-param>**
49. <!-- 日志页面的刷新间隔 -->
50. **<param-name>**log4jRefreshInterval**</param-name>**
51. **<param-value>**6000**</param-value>**
52. **</context-param>**
53. **<listener>**
54. **<listener-class>**org.springframework.web.util.Log4jConfigListener**</listener-class>**
55. **</listener>**
57. **<listener>**
58. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
59. **</listener>**
61. <!-- springMVC核心配置 -->
62. **<servlet>**
63. **<servlet-name>**dispatcherServlet**</servlet-name>**
64. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
65. **<init-param>**
66. **<param-name>**contextConfigLocation**</param-name>**
67. <!--spingMVC的配置路径  -->
68. **<param-value>**classpath:springmvc/spring-mvc.xml**</param-value>**
69. **</init-param>**
70. **<load-on-startup>**1**</load-on-startup>**
71. **</servlet>**
72. <!-- 拦截设置 -->
73. **<servlet-mapping>**
74. **<servlet-name>**dispatcherServlet**</servlet-name>**
75. **<url-pattern>**/**</url-pattern>**
76. **</servlet-mapping>**
78. <!-- 错误跳转页面 -->
79. **<error-page>**
80. <!-- 路径不正确 -->
81. **<error-code>**404**</error-code>**
82. **<location>**/WEB-INF/errorpage/404.jsp**</location>**
83. **</error-page>**
84. **<error-page>**
85. <!-- 没有访问权限，访问被禁止 -->
86. **<error-code>**405**</error-code>**
87. **<location>**/WEB-INF/errorpage/405.jsp**</location>**
88. **</error-page>**
89. **<error-page>**
90. <!-- 内部错误 -->
91. **<error-code>**500**</error-code>**
92. **<location>**/WEB-INF/errorpage/500.jsp**</location>**
93. **</error-page>**
94. **</web-app>**

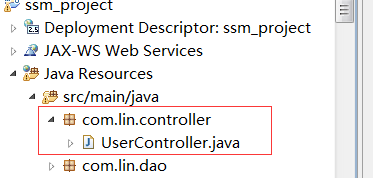
（4）添加index.jsp

在src/main/webapp/WEB-INF下新建一个文件夹view,添加一个index.jsp，内容如下：

**[html]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

1. **<**%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%**>**
2. **<html>**
3. **<body>**
4. **<h2>**Hello World!**</h2>**
5. ${user.userId}**<br>**
6. ${user.userName}**<br>**
7. ${user.userPassword}**<br>**
8. ${user.userEmail}**<br>**
9. **</body>**
10. **</html>**

（5）最后就是controller的书写了

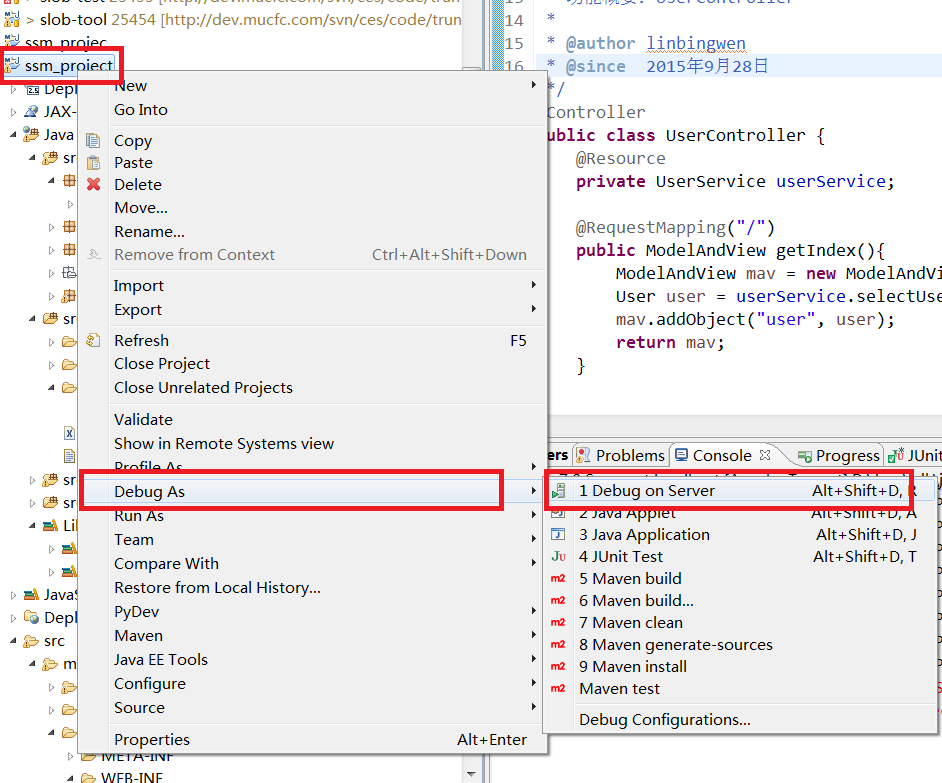


在src/main/java下新建一个包com.lin.controller.然后新建一个类UserController.java,其内容如下：

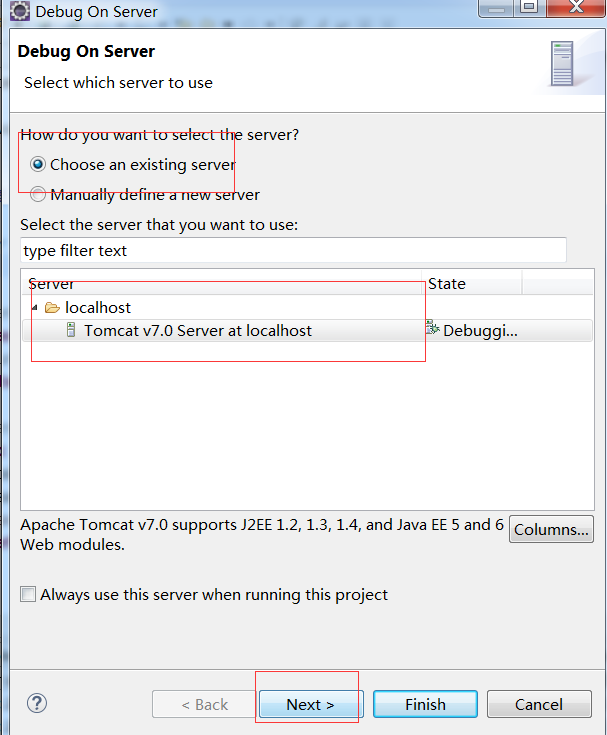
**[java]** [view plain](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ) [copy](http://blog.csdn.net/evankaka/article/details/48785513?spm=5176.100239.blogcont28591.10.3strbJ)

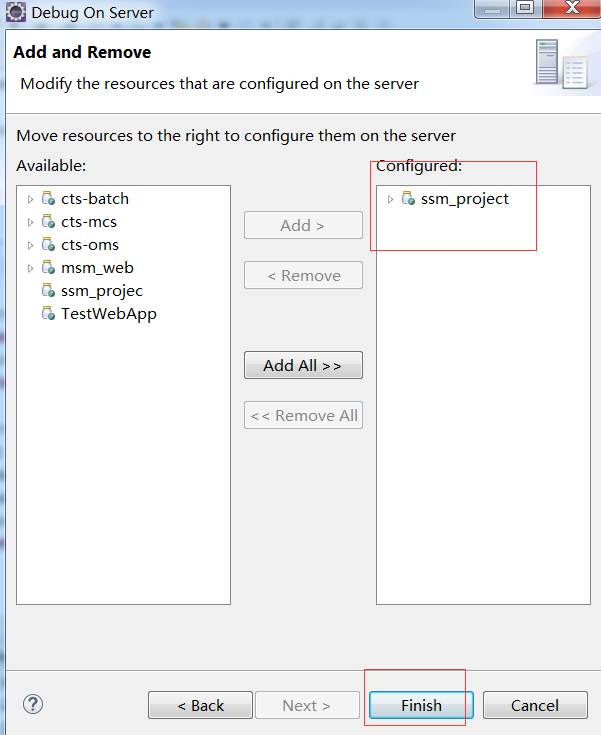
1. **package** com.lin.controller;
3. **import** javax.annotation.Resource;
5. **import** org.springframework.stereotype.Controller;
6. **import** org.springframework.web.bind.annotation.RequestMapping;
7. **import** org.springframework.web.servlet.ModelAndView;
9. **import** com.lin.domain.User;
10. **import** com.lin.service.UserService;
12. /\*\*
13. \* 功能概要：UserController
14. \*
15. \* @author linbingwen
16. \* @since  2015年9月28日
17. \*/
18. @Controller
19. **public** **class** UserController {
20. @Resource
21. **private** UserService userService;
23. @RequestMapping("/")
24. **public** ModelAndView getIndex(){
25. ModelAndView mav = **new** ModelAndView("index");
26. User user = userService.selectUserById(1);
27. mav.addObject("user", user);
28. **return** mav;
29. }
30. }

（6）最后的运行！  
最后，就是成果了，已经全部都配置好了。我们可以开始以web工程运行了！

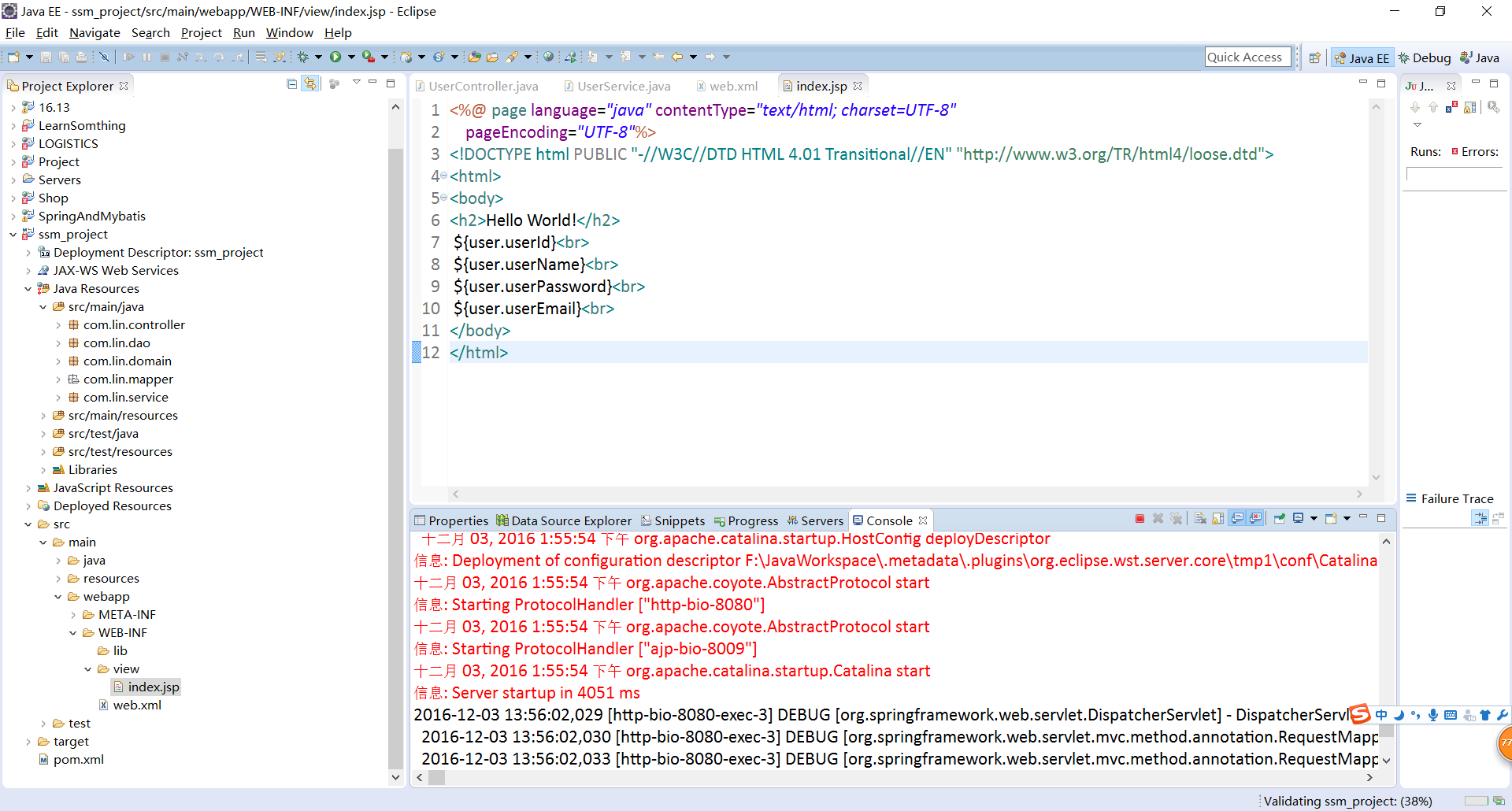


然后一路next

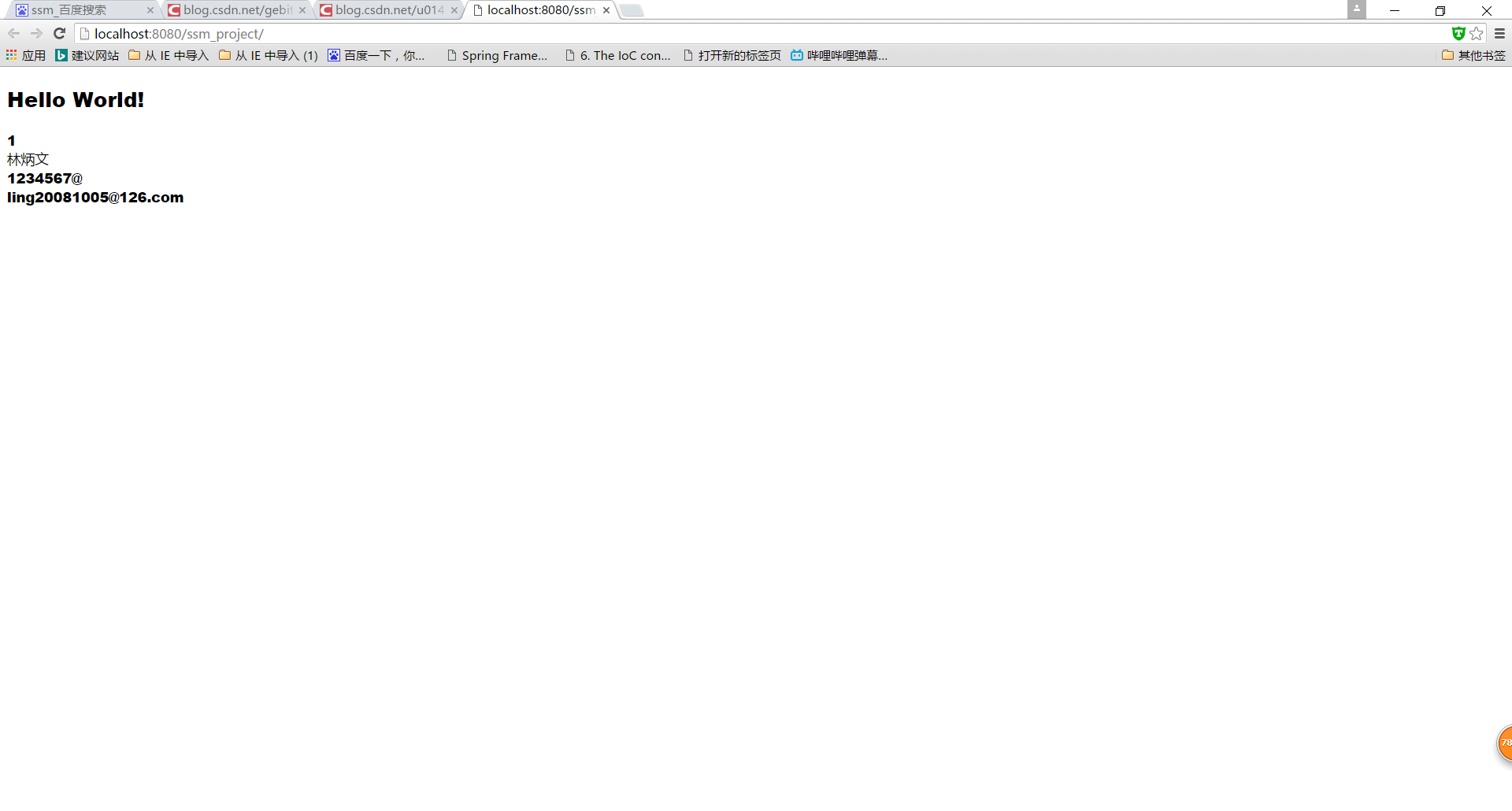




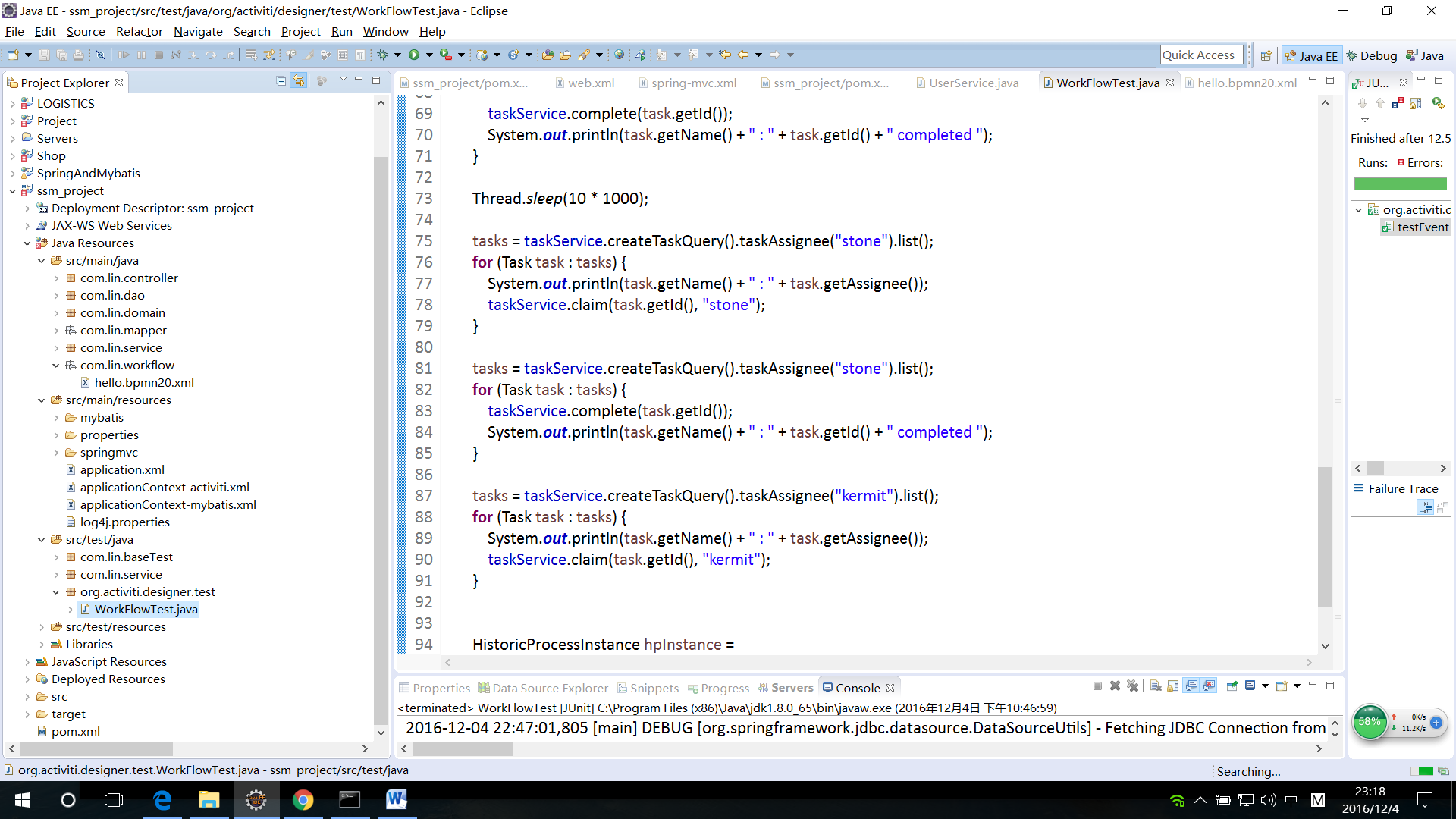
接下来控制台就会打印日志出来了！如下



打印浏览器，输入网址：<http://localhost:8080/ssm_project/>



加入activiti功能后的项目目录以及测试类



问题1：无法项目中引入maven仓库中的jar包到web中添加

1. **<classpathentry** kind="con" path="org.eclipse.m2e.MAVEN2\_CLASSPATH\_CONTAINER"**>**
2. **<attributes>**
3. **<attribute** name="maven.pomderived" value="true"**/>**
4. **<attribute** name="org.eclipse.jst.component.dependency" value="/WEB-INF/lib"**/>**
5. **</attributes>**
6. **</classpathentry>**

问题2，在maven下载依赖包时，因为一些原因下载失败，可以尝试，到本地仓库删除不完整的包，然后重新更新项目下载。

问题3，