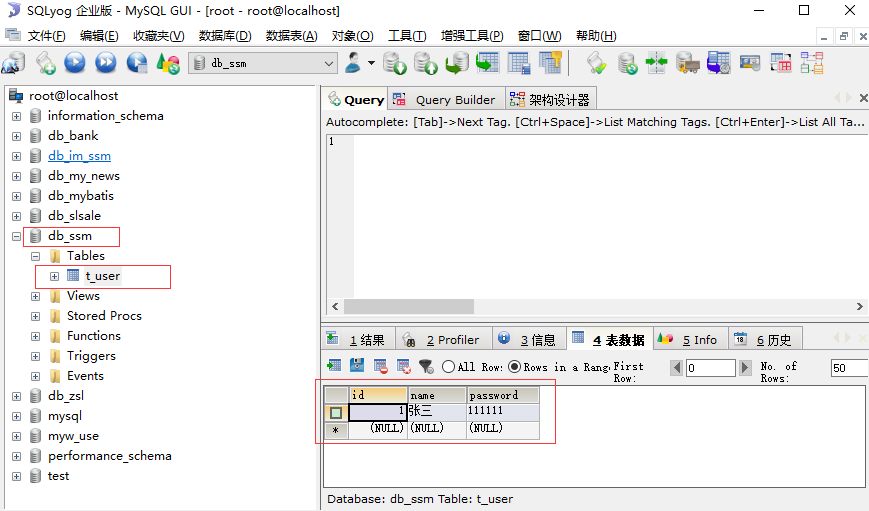
### [新手-图文]整合ssm框架，从mybatis到spring+mybatis再到ssm(sping+mybatis+spingmvc)，数据库mysql

### 0序 相关准备

#### Jar包

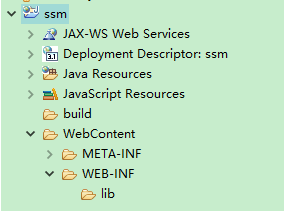


#### 数据库

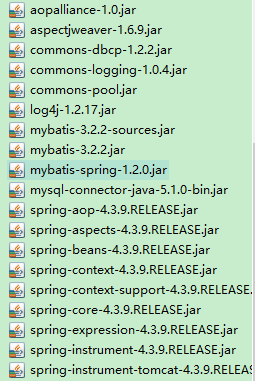
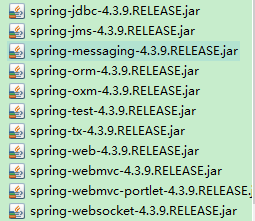


创建 db\_ssm数据库 t\_user表 照图中插入一条数据

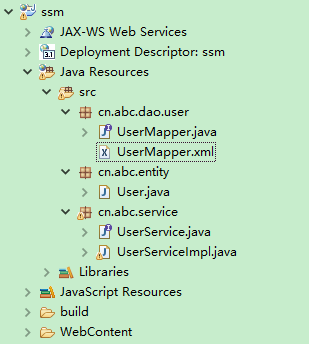
### 1，创建一个web项目



### 2，导入jar包

  有点多。。。

### 3，创建service dao entity实体



代码

Usermapper.java

|  |
| --- |
| **package** cn.abc.dao.user;  **import** cn.abc.entity.User;  **public** **interface** UserMapper {  **public** User getUserById(**int** id);  } |

Usermapper.xml

|  |
| --- |
| <?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=*"cn.abc.dao.user.UserMapper"*>  <select id=*"getUserById"* parameterType=*"Integer"* resultType=*"User"*>  select \* from t\_user where id = #{id};  </select>  </mapper> |

User.java

|  |
| --- |
| **package** cn.abc.entity;  **public** **class** User {  **private** **int** id;  **private** String name;  **private** String password;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  @Override  **public** String toString() {  **return** "User [id=" + id + ", name=" + name + ", password=" + password  + "]";  }  } |

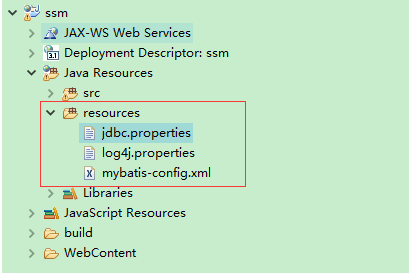
UserService.java

|  |
| --- |
| **package** cn.abc.service;  **import** cn.abc.entity.User;  **public** **interface** UserService {  **public** User getUserById(**int** id);  } |

UserServiceImpl

|  |
| --- |
| **package** cn.abc.service;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.beans.factory.annotation.Qualifier;  **import** org.springframework.stereotype.Service;  **import** cn.abc.dao.user.UserMapper;  **import** cn.abc.entity.User;  @Service("userService")  **public** **class** UserServiceImpl **implements** UserService {  @Autowired  **private** UserMapper userMapper;    **public** UserMapper getUserMapper() {  **return** userMapper;  }  **public** **void** setUserMapper(UserMapper userMapper) {  **this**.userMapper = userMapper;  }  **public** User getUserById(**int** id) {  // **TODO** Auto-generated method stub  **return** userMapper.getUserById(id);  }  } |

### 4，创建resources文件夹



Jdbc.properties

|  |
| --- |
| driver=com.mysql.jdbc.Driver  #在和mysql传递数据的过程中，使用unicode编码格式，并且字符集设置为utf-8  url=jdbc:mysql://127.0.0.1:3306/db\_ssm?useUnicode=true&characterEncoding=utf-8  user=root  password=myw123456  #定义初始连接数  initialSize=5  #定义最大连接数  maxActive=20  #定义最大空闲  maxIdle=20  #定义最小空闲  minIdle=1  #定义最长等待时间  maxWait=60000  #暂时未知  removeAbandonedTimeout=180  #无用链接回收机制  removeAbandoned=true |

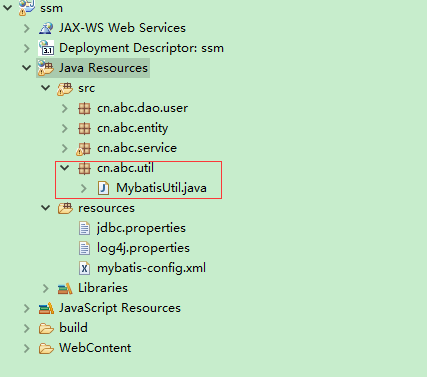
log4j.properties

|  |
| --- |
| log4j.rootLogger=DEBUG,CONSOLE,file  #log4j.rootLogger=ERROR,ROLLING\_FILE  log4j.logger.cn.smbms.dao=debug  log4j.logger.com.ibatis=debug  log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=debug  log4j.logger.com.ibatis.common.jdbc.ScriptRunner=debug  log4j.logger.com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate=debug  log4j.logger.java.sql.Connection=debug  log4j.logger.java.sql.Statement=debug  log4j.logger.java.sql.PreparedStatement=debug  log4j.logger.java.sql.ResultSet=debug  log4j.logger.org.tuckey.web.filters.urlrewrite.UrlRewriteFilter=debug  ######################################################################################  # Console Appender \u65e5\u5fd7\u5728\u63a7\u5236\u8f93\u51fa\u914d\u7f6e  ######################################################################################  log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender  log4j.appender.Threshold=error  log4j.appender.CONSOLE.Target=System.out  log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout  log4j.appender.CONSOLE.layout.ConversionPattern= [%p] %d %c - %m%n  ######################################################################################  # DailyRolling File \u6bcf\u5929\u4ea7\u751f\u4e00\u4e2a\u65e5\u5fd7\u6587\u4ef6\uff0c\u6587\u4ef6\u540d\u683c\u5f0f:log2009-09-11  ######################################################################################  log4j.appender.file=org.apache.log4j.DailyRollingFileAppender  log4j.appender.file.DatePattern=yyyy-MM-dd  log4j.appender.file.File=log.log  log4j.appender.file.Append=true  log4j.appender.file.Threshold=error  log4j.appender.file.layout=org.apache.log4j.PatternLayout  log4j.appender.file.layout.ConversionPattern=%d{yyyy-M-d HH:mm:ss}%x[%5p](%F:%L) %m%n  log4j.logger.com.opensymphony.xwork2=error |

mybatis-config.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">  <!-- 通过这个配置文件完成mybatis与数据库的连接 -->  <configuration>  <!-- 引入 database.properties 文件-->  <properties resource=*"jdbc.properties"*/>    <settings>  <!-- 配置mybatis的log实现为LOG4J -->  <setting name=*"logImpl"* value=*"LOG4J"* />  <!-- 设置resultMap的自动映射级别为NONE(禁止自动匹配) -->  <!-- <setting name="autoMappingBehavior" value="NONE" /> -->  </settings>    <!--类型别名-->  <typeAliases>  <!-- <typeAlias alias="User" type="cn.smbms.pojo.User"/> -->  <package name=*"cn.abc.entity"*/>  </typeAliases>    <environments default=*"development"*>  <environment id=*"development"*>    <transactionManager type=*"JDBC"*></transactionManager>  <dataSource type=*"POOLED"*>  <property name=*"driver"* value=*"${driver}"*/>  <property name=*"url"* value=*"${url}"*/>  <property name=*"username"* value=*"${user}"*/>  <property name=*"password"* value=*"${password}"*/>  </dataSource>  </environment>  </environments>      <mappers>  <mapper resource=*"cn/abc/dao/user/UserMapper.xml"*/>  </mappers>  </configuration> |

### 5，编写工具类



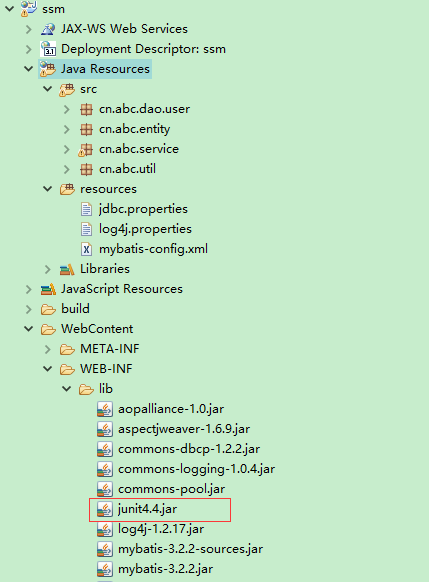
MyBatisUtil.java

|  |
| --- |
| **package** cn.abc.util;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** org.apache.ibatis.io.Resources;  **import** org.apache.ibatis.session.SqlSession;  **import** org.apache.ibatis.session.SqlSessionFactory;  **import** org.apache.ibatis.session.SqlSessionFactoryBuilder;  **public** **class** MybatisUtil {  **private** **static** SqlSessionFactory *sqlSessionFactory*;    **public** **static** SqlSessionFactory getSqlSessionFactory() {  **try** {  InputStream in = Resources.*getResourceAsStream*("mybatis-config.xml");  *sqlSessionFactory* = **new** SqlSessionFactoryBuilder().build(in);  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  System.***out***.println("加载文件出错");  e.printStackTrace();  }  **return** *sqlSessionFactory*;    }    **public** **static** SqlSession openSession() {  **return** *getSqlSessionFactory*().openSession();  }  } |

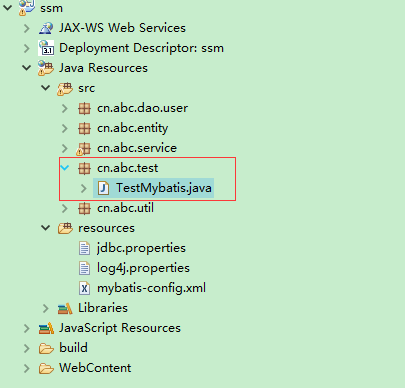
### 6,测试mybatis

添加junit jar包



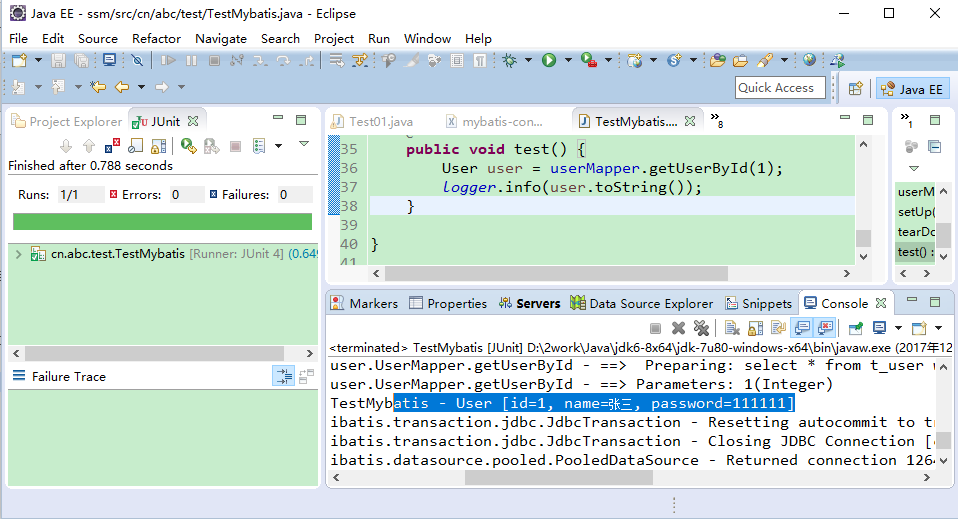


编写测试类



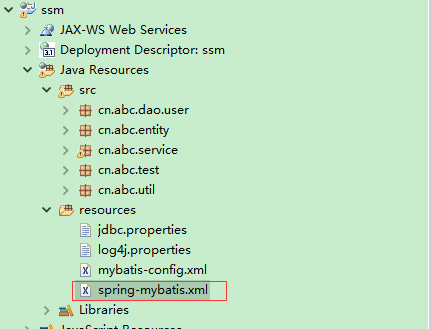
|  |
| --- |
| **package** cn.abc.test;  **import** org.apache.ibatis.session.SqlSession;  **import** org.apache.log4j.Logger;  **import** org.junit.After;  **import** org.junit.Before;  **import** org.junit.Test;  **import** cn.abc.dao.user.UserMapper;  **import** cn.abc.entity.User;  **import** cn.abc.util.MybatisUtil;  **public** **class** TestMybatis {  **private** **static** Logger *logger* = Logger.*getLogger*(TestMybatis.**class**);  **private** SqlSession sqlSession = **null**;  **private** UserMapper userMapper = **null**;  @Before  **public** **void** setUp() **throws** Exception {  sqlSession = MybatisUtil.*openSession*();  userMapper = sqlSession.getMapper(UserMapper.**class**);  }  @After  **public** **void** tearDown() **throws** Exception {  sqlSession.close();  }  @Test  **public** **void** test() {  User user = userMapper.getUserById(1);  *logger*.info(user.toString());  }  } |

执行测试



好，信息打出来了。Mybatis成功 ，下面开始整合spring

### 7，整合spring 创建spring-mybatis.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:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-3.2.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-3.2.xsd "*>  <!-- 引入 -->  <context:property-placeholder location=*"classpath:jdbc.properties"*/>      <!-- 数据源 -->  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>  <property name=*"driverClassName"* value=*"${driver}"*></property>  <property name=*"url"* value=*"${url}"*></property>  <property name=*"username"* value=*"${user}"*></property>  <property name=*"password"* value=*"${password}"*></property>  <property name=*"initialSize"* value=*"${initialSize}"*></property>  <property name=*"maxActive"* value=*"${maxActive}"*></property>  <property name=*"maxIdle"* value=*"${maxIdle}"*></property>  <property name=*"minIdle"* value=*"${minIdle}"*></property>  <property name=*"maxWait"* value=*"${maxWait}"*></property>  <!-- 开启无用链接的回收机制 true 是开启 当前的空闲链接数《2 且当前活动数》最大活动数-3 出发无用连接的回收 -->  <property name=*"removeAbandoned"* value=*"${removeAbandoned}"*></property>  <!-- 无用链接回收机制 -->  <property name=*"removeAbandonedTimeout"* value=*"${removeAbandonedTimeout}"*></property>    <!-- 配置sql心跳 重启数据库 使用新的连接 保证连接池的连接是真实有效的连接 testWhileIdle testOnBorrow testOnreturn  下面的都是sql心跳包 主要是把无用的连接给回收 mysql如果8个小时没有动态，会自己断开连接 -->  <!-- 开启Evict的定时校验，循环校验 -->  <property name=*"testWhileIdle"* value=*"true"*></property>  <!-- 定义事件间隔 单位毫秒 大于0的时候才会开启Evict -->  <property name=*"timeBetweenEvictionRunsMillis"* value=*"60000"*></property>  <!-- 在进行borrowObject处理时，会对拿到的连接进行校验 默认是false -->  <property name=*"testOnBorrow"* value=*"false"*></property>  <!-- 在进行returnObject处理时，会对返回的连接进行校验 默认是false -->  <property name=*"testOnReturn"* value=*"false"*></property>  <!-- 校验使用的sql语句 validationQuery 复杂的校验sql会影响性能 告诉 数据库我还活着 -->  <property name=*"validationQuery"* value=*"select 1"*></property>  <!-- 每次校验所有的连接 -->  <property name=*"numTestsPerEvictionRun"* value=*"${maxActive}"*></property>  </bean>    <!-- 配置SqlSessionFactoryBean -->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"*></property>  <property name=*"configLocation"* value=*"classpath:mybatis-config.xml"*></property>  <!-- 引入mybatis的配置 此处应保留mybatis的配置文件，以适应拓展 -->  </bean>  <!-- 配置事务 -->  <!-- 需要 aop tx -->  <bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <property name=*"dataSource"* ref=*"dataSource"*></property>  </bean>  <!-- 定义事物管理器 用上面的定义来进行事务处理 这个有问题 不能打开，用到的时候在研究  <tx:advice id="txAdvice" transaction-manager="transactionManager">  <tx:attributes>  <tx:method name="hl\*" propagation="REQUIRED" rollback-for="Exception" />  </tx:attributes>  </tx:advice>  -->      <!-- 配置SqlSessionTemplate -->  <!-- <bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">  <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>  </bean> -->    <!-- Dao 将词包下的接口转换为实体类-->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"cn.abc.dao"*></property>  </bean>  <!-- Service -->  <!-- 扫描注解定义 -->  <context:component-scan base-package=*"cn.abc"*></context:component-scan>    <!-- redis 配置暂缺 -->  </beans> |

下面的这个东西是关键，它可以扫描dao接口，转换为bean。比如此例中就是把UserMapper接口转化为了userMapper bean

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

<property name=*"basePackage"* value=*"cn.abc.dao"*></property>

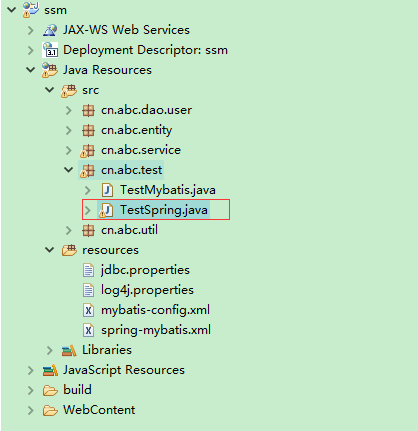
</bean>

### 8,修改mybatis-config.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">  <!-- 通过这个配置文件完成mybatis与数据库的连接 -->  <configuration>  <!-- 引入 database.properties 文件-->  <properties resource=*"jdbc.properties"*/>    <settings>  <!-- 配置mybatis的log实现为LOG4J -->  <setting name=*"logImpl"* value=*"LOG4J"* />  <!-- 设置resultMap的自动映射级别为NONE(禁止自动匹配) -->  <!-- <setting name="autoMappingBehavior" value="NONE" /> -->  </settings>    <!--类型别名-->  <typeAliases>  <!-- <typeAlias alias="User" type="cn.smbms.pojo.User"/> -->  <package name=*"cn.abc.entity"*/>  </typeAliases>  <!--  <environments default="development">  <environment id="development">  配置事务管理，采用JDBC的事务管理  <transactionManager type="JDBC"></transactionManager>  POOLED:mybatis自带的数据源，JNDI:基于tomcat的数据源  <dataSource type="POOLED">  <property name="driver" value="${driver}"/>  <property name="url" value="${url}"/>  <property name="username" value="${user}"/>  <property name="password" value="${password}"/>  </dataSource>  </environment>  </environments>    将mapper文件加入到配置文件中  <mappers>  <mapper resource="cn/smbms/dao/user/UserMapper.xml"/>  </mappers> -->  </configuration> |

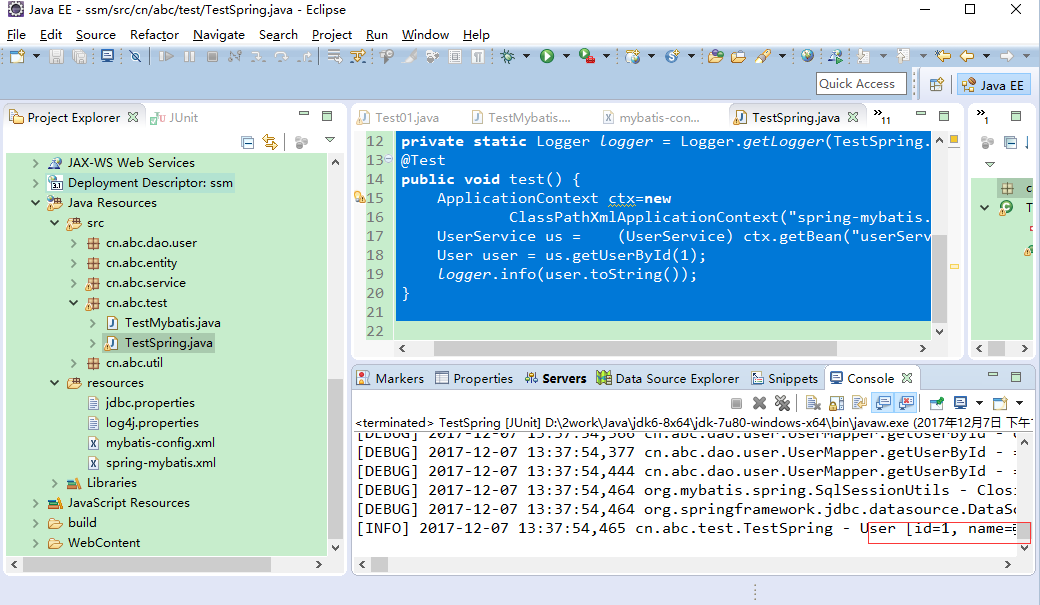
只保留了一个typeAliases 别的都被注释掉了

### 9创建测试类



|  |
| --- |
| **package** cn.abc.test;  **import** org.apache.log4j.Logger;  **import** org.junit.Test;  **import** org.springframework.context.ApplicationContext;  **import** org.springframework.context.support.ClassPathXmlApplicationContext;  **import** cn.abc.entity.User;  **import** cn.abc.service.UserService;  **public** **class** TestSpring {  **private** **static** Logger *logger* = Logger.*getLogger*(TestSpring.**class**);  @Test  **public** **void** test() {  ApplicationContext ctx=**new**  ClassPathXmlApplicationContext("spring-mybatis.xml");  UserService us = (UserService) ctx.getBean("userService");  User user = us.getUserById(1);  *logger*.info(user.toString());  }  } |

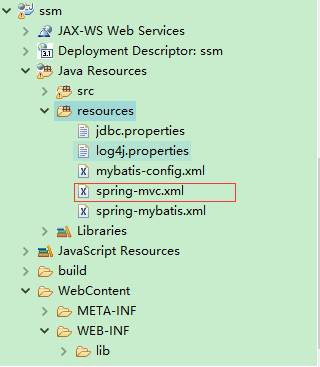
运行结果



出来了，至此**spring+mybatis**的整合已经结束了。此时由于mybatis-config文件的修改，TestMybatis已经不能用了

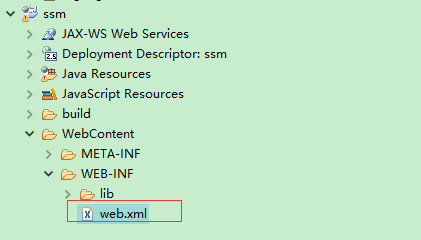
下面整合spring mvc

### 10，整合spring mvc 创建spring-mvc.xml文件



|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"*  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:p=*"http://www.springframework.org/schema/p"*  xmlns:util=*"http://www.springframework.org/schema/util"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*  xmlns:cache=*"http://www.springframework.org/schema/cache"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.3.xsd*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-4.3.xsd*  *http://www.springframework.org/schema/jdbc*  *http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd*  *http://www.springframework.org/schema/cache*  *http://www.springframework.org/schema/cache/spring-cache-4.3.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-4.3.xsd*  *http://www.springframework.org/schema/util*  *http://www.springframework.org/schema/util/spring-util-4.3.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd"*>  <!-- 以annotation的方式 装配controller -->  <mvc:annotation-driven></mvc:annotation-driven>  <!-- 使用注解的包， 这个可以扫描controller 包括子集 -->  <context:component-scan base-package=*"cn.abc.controller"*/>  <!-- 视图解析器 -->  <bean id=*"viewResolver"*  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"*></property>  </bean>          </beans> |

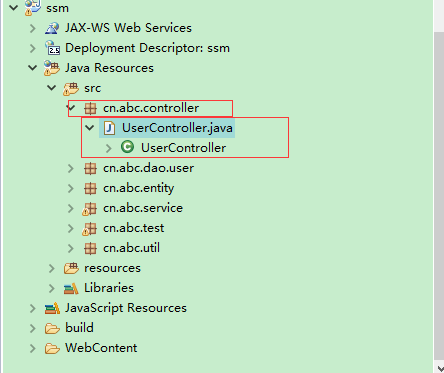
### 11，配置web.xml 照图中的路径找，没有文件要自己创建一个web.xml



|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app version=*"2.5"*  xmlns=*"http://java.sun.com/xml/ns/javaee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*  *http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>  <display-name></display-name>  <!-- 加载-->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mybatis.xml</param-value>  </context-param>  <!-- Spring监听器 -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mvc.xml</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  </web-app> |

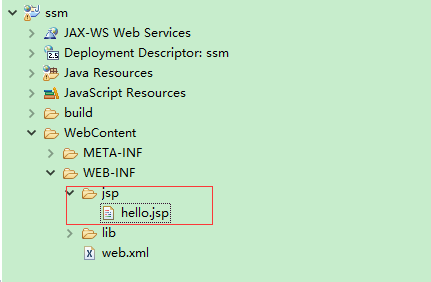
这个地方有两个关键点context-param 与ContextLoaderListener 这两个东西就把sping的东西整合到web项目了，仔细看看

### 12，编写controller



|  |
| --- |
| **package** cn.abc.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** cn.abc.entity.User;  **import** cn.abc.service.UserService;  @Controller  **public** **class** UserController {  @Autowired  **private** UserService userService;  @RequestMapping("/haha")  **public** String helloWorld(Model model){  User user = userService.getUserById(1);  model.addAttribute("user", user);  **return** "hello";  }      } |

### 13，在下图位置创建文件夹及jsp文件

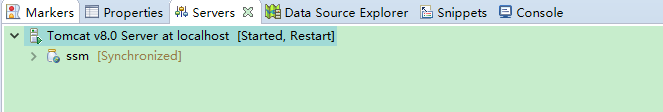


|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href=*"*<%=basePath%>*"*>    <title>My JSP 'hello.jsp' starting page</title>    <meta http-equiv=*"pragma"* content=*"no-cache"*>  <meta http-equiv=*"cache-control"* content=*"no-cache"*>  <meta http-equiv=*"expires"* content=*"0"*>  <meta http-equiv=*"keywords"* content=*"keyword1,keyword2,keyword3"*>  <meta http-equiv=*"description"* content=*"This is my page"*>  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>    <body>  ${user.name}  </body>  </html> |

里面只有一个el表达式

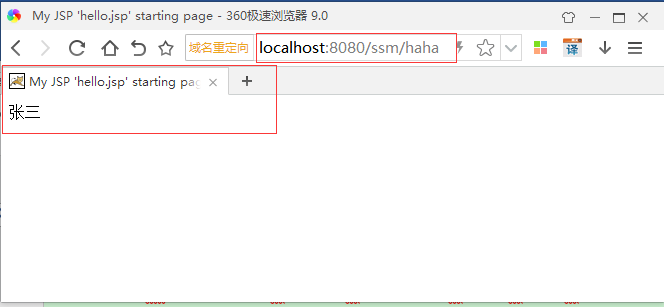
如果有jsp报错 右键项目》properties》java build path》libraries》add lib》service runtime》选择tomcat

### 14，测试



在浏览器中输入<http://localhost:8080/ssm/haha>

就可以看到结果了



这个张三就是从数据库取出来的数据

源码

