SSH框架整合

## 目的导向

主要是通过操作网页完成对数据库的操作

Spring 5.0

Hibernate 5.1

Struts 2.5

Mysql 5.0

JDK 8

IDE ECLIPSE

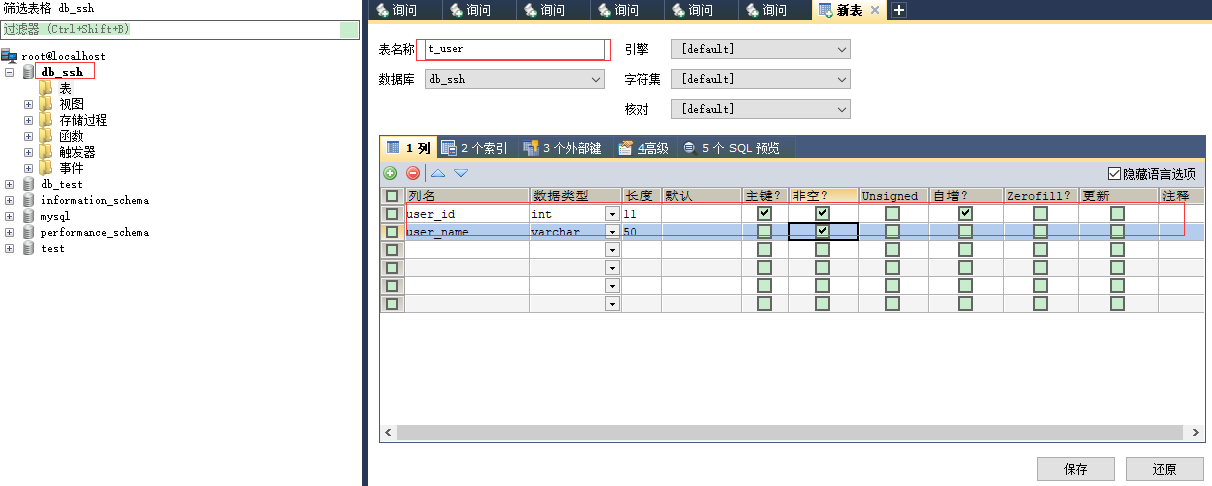
## 0准备工作

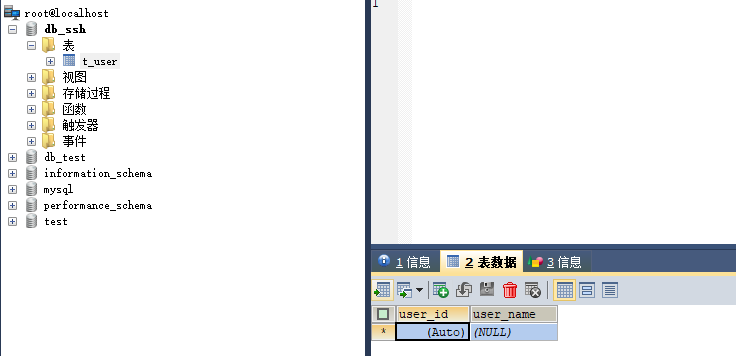
### 0.1Jar包下载

<https://download.csdn.net/download/qq_24691413/10391210>

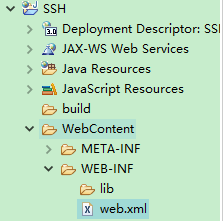
### 0.2数据库建表—mysql

数据库比较简单，不附原码



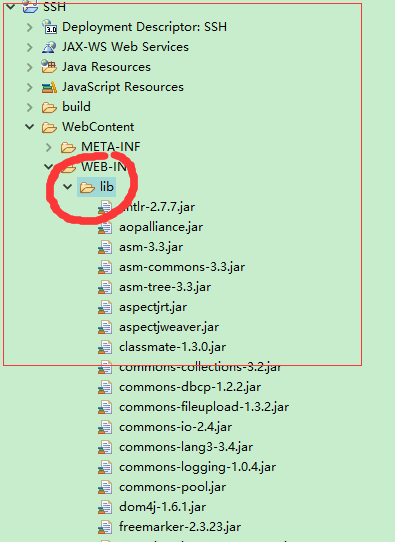


### 0.3创建web项目—基于eclipse



## 1hibernate

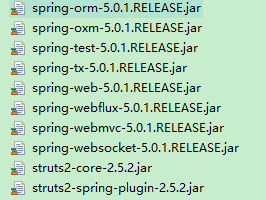
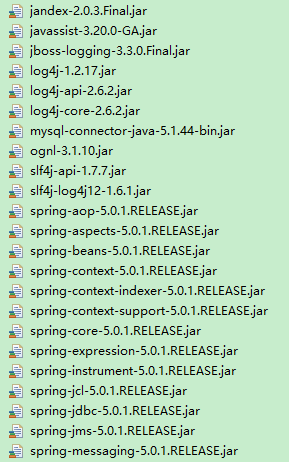
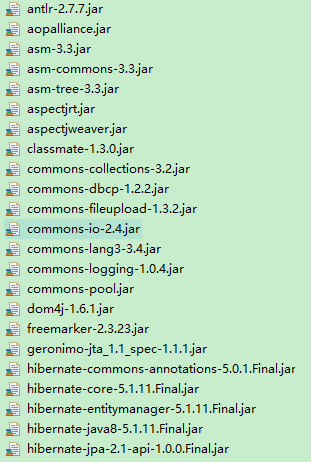
### 1．1Jar包



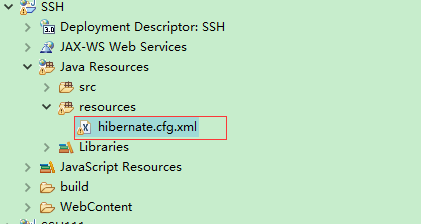
Jar包放在lib目录如图所示

Jar列表

下面是所有jar包列表，肯定是有好多没用的，但是水平太低



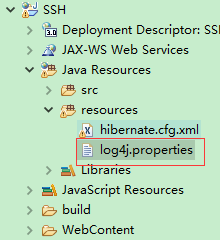
### 1.2创建resouese文件夹及 hibernate.cfg.xml



hibernate.cfg.xml 代码

|  |
| --- |
| <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory>    <!-- 数据库 连接 -->  <property name="connection.url">jdbc:mysql://localhost:3306/db\_ssh</property>  <property name="connection.driver\_class">com.mysql.jdbc.Driver</property>  <property name="connection.username">root</property>  <property name="connection.password">123456</property>  <!-- <property name="connection.url">jdbc:oracle:thin:@localhost:1521:orcl</property>  <property name="connection.driver\_class">oracle.jdbc.OracleDriver</property>  <property name="connection.username">scott</property> <property name="connection.password">123456</property> -->    <property name="show\_sql">true</property><!-- 是否显示sql语句 -->  <property name="format\_sql">true</property><!-- 格式化sql语句 -->    <!-- <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>设置方言 oracle -->  <property name="dialect">org.hibernate.dialect.MySQLDialect</property><!-- 设置方言 -->    <property name="current\_session\_context\_class">thread</property><!-- 线程绑定 -->  <!-- <property name="hibernate.hbm2ddl.auto">create(false,false)</property> --><!-- 若没有此表则创建此表 -->  <!-- 引入配置 -->  <!-- <mapping resource="org/hibernate/test/legacy/Simple.hbm.xml" /> -->  <mapping resource="cn/mayiwen/entity/User.hbm.xml" />  </session-factory>  </hibernate-configuration> |

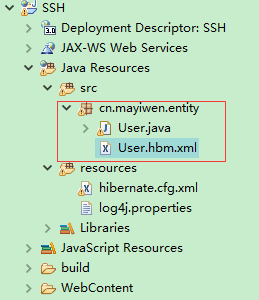
### 1.3创建log4j.properties在resources文件夹下



log4j.properties代码

|  |
| --- |
| ### direct log messages to stdout ###  log4j.appender.stdout=org.apache.log4j.ConsoleAppender  log4j.appender.stdout.Target=System.out  log4j.appender.stdout.layout=org.apache.log4j.PatternLayout  log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n    ### direct messages to file hibernate.log ###  #log4j.appender.file=org.apache.log4j.FileAppender  #log4j.appender.file.File=hibernate.log  #log4j.appender.file.layout=org.apache.log4j.PatternLayout  #log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n    ### set log levels - for more verbose logging change 'info' to 'debug' ###    log4j.rootLogger=warn, stdout    #log4j.logger.org.hibernate=info  log4j.logger.org.hibernate=debug    ### log HQL query parser activity  #log4j.logger.org.hibernate.hql.ast.AST=debug    ### log just the SQL  #log4j.logger.org.hibernate.SQL=debug    ### log JDBC bind parameters ###  log4j.logger.org.hibernate.type=info  #log4j.logger.org.hibernate.type=debug    ### log schema export/update ###  log4j.logger.org.hibernate.tool.hbm2ddl=debug    ### log HQL parse trees  #log4j.logger.org.hibernate.hql=debug    ### log cache activity ###  #log4j.logger.org.hibernate.cache=debug    ### log transaction activity  #log4j.logger.org.hibernate.transaction=debug    ### log JDBC resource acquisition  #log4j.logger.org.hibernate.jdbc=debug    ### enable the following line if you want to track down connection ###  ### leakages when using DriverManagerConnectionProvider ###  #log4j.logger.org.hibernate.connection.DriverManagerConnectionProvider=trace |

### 1.4创建User类与User.hbm.xml 映射文件



User.java

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.entity;  import java.io.Serializable;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class User implements Serializable {  private Integer id;  private String name;          /\* (non-Javadoc)  \* @see java.lang.Object#toString()  \*/  @Override  public String toString() {  return "User [id=" + id + ", name=" + name + "]";  }  /\*\*  \* @return the id  \*/  public Integer getId() {  return id;  }  /\*\*  \* @param id the id to set  \*/  public void setId(Integer id) {  this.id = id;  }  /\*\*  \* @return the name  \*/  public String getName() {  return name;  }  /\*\*  \* @param name the name to set  \*/  public void setName(String name) {  this.name = name;  }          } |

创建User.hbm.xml映射文件 User.hbm.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-mapping  PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd" >      <hibernate-mapping>  <!-- dynamic-update 动态更新 可以只更新那些发饰能变化的字段-->  <class name="cn.mayiwen.entity.User" table="`t\_user`" dynamic-update="true" ><!-- 类和表的对应关系 -->  <id name="id" column="`user\_id`" type="java.lang.Integer" >  <!-- 指定增长方式 -->  <!-- <generator class="sequence"><param name="sequence">SEQ\_DEPTNO</param></generator>序列 -->  <generator class="identity"></generator> <!--自增长 mysql sqlserver -->  <!-- <generator class="increment"></generator> 取出最大值+1 -->  <!--<generator class="native"></generator> 根据方言 来指定 -->  <!-- <generator class="assigned"></generator> 程序指定 -->  <!-- <generator class="uuid"></generator> 32位 -->  </id>  <property name="name" column="`user\_name`" type="java.lang.String" ><!-- 属性应该是 getset 的原则 -->  </property>  </class>        </hibernate-mapping> |

此时再回头看



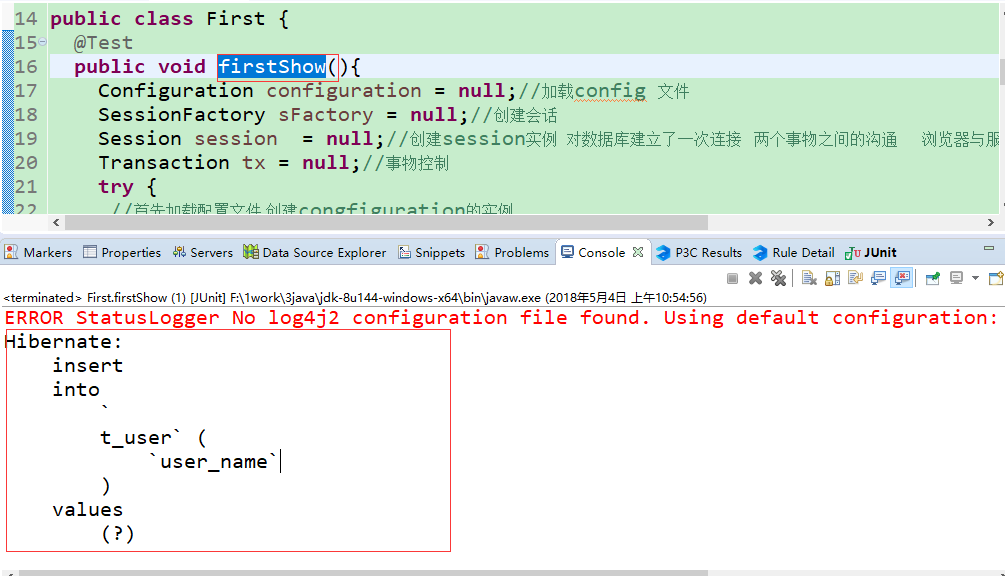
1.5编写first测试类，并根据提示引入junit



First.java

|  |
| --- |
| package cn.mayiwen.test;    import org.hibernate.HibernateException;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.Transaction;  import org.hibernate.cfg.Configuration;  import org.junit.Test;    import cn.mayiwen.entity.User;        public class First {  @Test  public void firstShow(){  Configuration configuration = null;//加载config 文件  SessionFactory sFactory = null;//创建会话  Session session = null;//创建session实例 对数据库建立了一次连接 两个事物之间的沟通 浏览器与服务器 程序与数据库  Transaction tx = null;//事物控制  try {  //首先加载配置文件 创建congfiguration的实例  configuration = new Configuration().configure();//因为是默认文件名 所以直接调用无参的配置信息 所以这个地方直接调用无参的构造方法就可以了  //构建工厂  sFactory = configuration.buildSessionFactory();  //通过工厂得到会话  session = sFactory.getCurrentSession();//opensession 也可以 建议使用此 这个会得到保护  //自动关闭 在事物结束的时候 会话关闭 增删改 查询都要开启事物  tx = session.beginTransaction();  User user = new User();  user.setName("马一文");  session.save(user);    tx.commit();  } catch (HibernateException e) {  e.printStackTrace();  if (tx != null) {  tx.rollback();//关闭  }  }                                }  } |

运行该类中firstShow（）方法

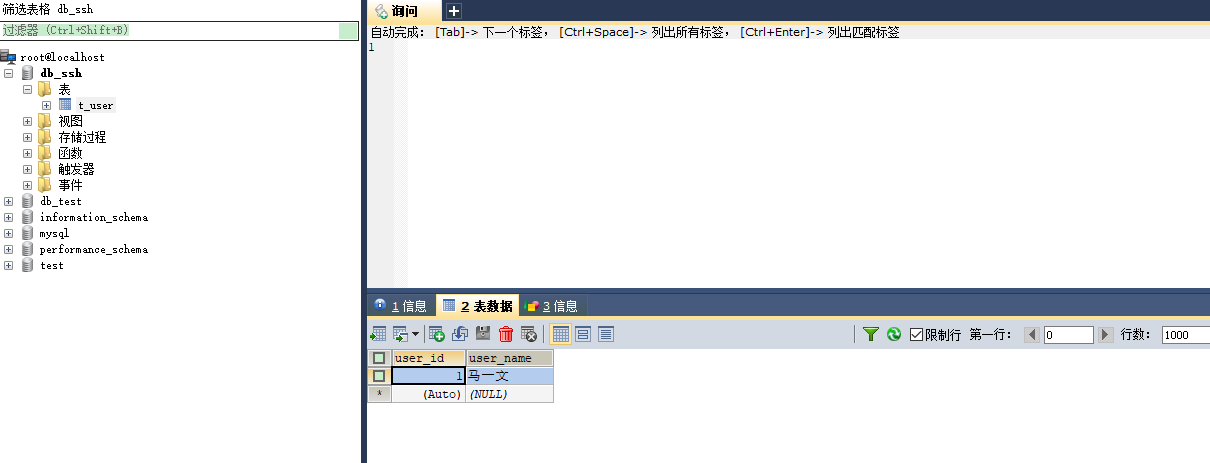


控制面板打印

### 1.5hibernate结束

至此 hibernate的配置结束了

查看数据库



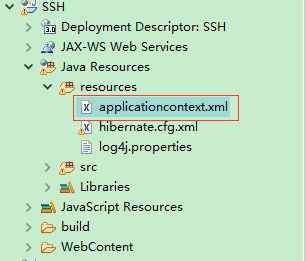
Ok已经有数据了

## 2Spring

### 2.0 spring 例子比较简单，这是之前的例子，照做



### 2.1在resources创建 applicationcontext.xml



applicationcontext.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:aop="http://www.springframework.org/schema/aop"  xmlns:cache="http://www.springframework.org/schema/cache"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:jdbc="http://www.springframework.org/schema/jdbc" xmlns:jee="http://www.springframework.org/schema/jee"  xmlns:jms="http://www.springframework.org/schema/jms" xmlns:lang="http://www.springframework.org/schema/lang"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:oxm="http://www.springframework.org/schema/oxm"  xmlns:p="http://www.springframework.org/schema/p" xmlns:task="http://www.springframework.org/schema/task"  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:util="http://www.springframework.org/schema/util"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.1.xsd  http://www.springframework.org/schema/cache http://www.springframework.org/schema/cache/spring-cache-4.1.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd  http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.1.xsd  http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.1.xsd  http://www.springframework.org/schema/jms http://www.springframework.org/schema/jms/spring-jms-4.1.xsd  http://www.springframework.org/schema/lang http://www.springframework.org/schema/lang/spring-lang-4.1.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd  http://www.springframework.org/schema/oxm http://www.springframework.org/schema/oxm/spring-oxm-4.1.xsd  http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-4.1.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.1.xsd  http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.1.xsd">      <bean id="helloSpring" class="cn.mayiwen.hello.HelloSpring">  </bean>        </beans> |

### 2.2创建HelloSpring类



HelloSpring.java

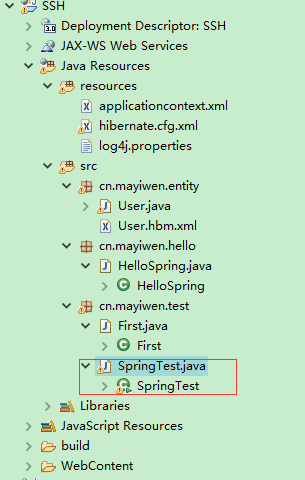
|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.hello;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class HelloSpring {  public void print() {  System.out.println("hello spring!!!");    }  } |

此时回头看



这样sping就完成了

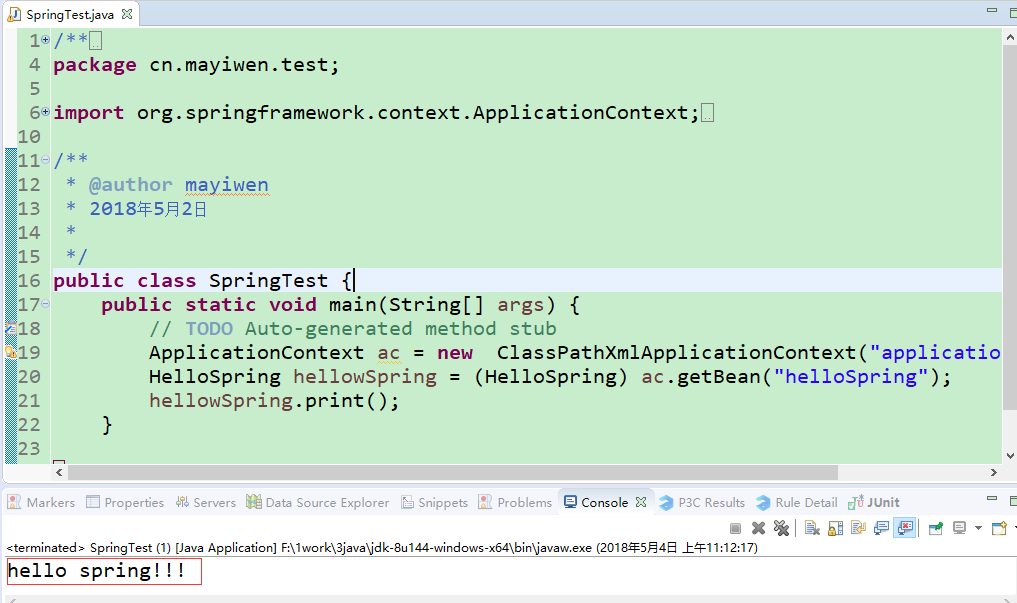
### 2.3测试spring



SpringTest

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.test;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  import cn.mayiwen.hello.HelloSpring;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class SpringTest {  public static void main(String[] args) {  // TODO Auto-generated method stub  ApplicationContext ac = new ClassPathXmlApplicationContext("applicationcontext.xml");  HelloSpring hellowSpring = (HelloSpring) ac.getBean("helloSpring");  hellowSpring.print();  }  } |

### 2.4spring结束 执行main（）方法



现在项目有hibernate与sping 单个的demo，下一步将此两个框架进行整合

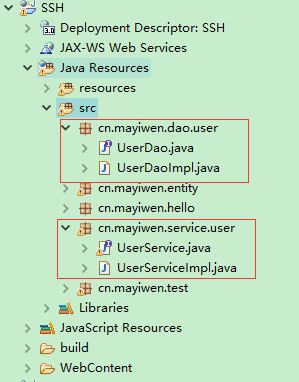
## 3整合spring与hibernate

### 3.0具体思路

通过sping管理类，调用与hibernate的方法，完成对数据库的操作

先构建三层架构的service层与dao层，spring调用service层方法，完成对dao层的操作，从而完成save操作

### 3.1创建service层dao层



#### UserDao

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.test;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  import cn.mayiwen.hello.HelloSpring;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class SpringTest {  public static void main(String[] args) {  // TODO Auto-generated method stub  ApplicationContext ac = new ClassPathXmlApplicationContext("applicationcontext.xml");  HelloSpring hellowSpring = (HelloSpring) ac.getBean("helloSpring");  hellowSpring.print();  }  } |

#### UserDaoImpl

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.dao.user;  import org.springframework.orm.hibernate5.support.HibernateDaoSupport;  import cn.mayiwen.entity.User;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class UserDaoImpl extends HibernateDaoSupport implements UserDao {  /\* (non-Javadoc)  \* @see cn.mayiwen.dao.user.UserDao#save(cn.mayiwen.entity.User)  \*/  @Override  public void save(User user) {  // TODO Auto-generated method stub  this.getHibernateTemplate().save(user);      }  } |

#### UserService

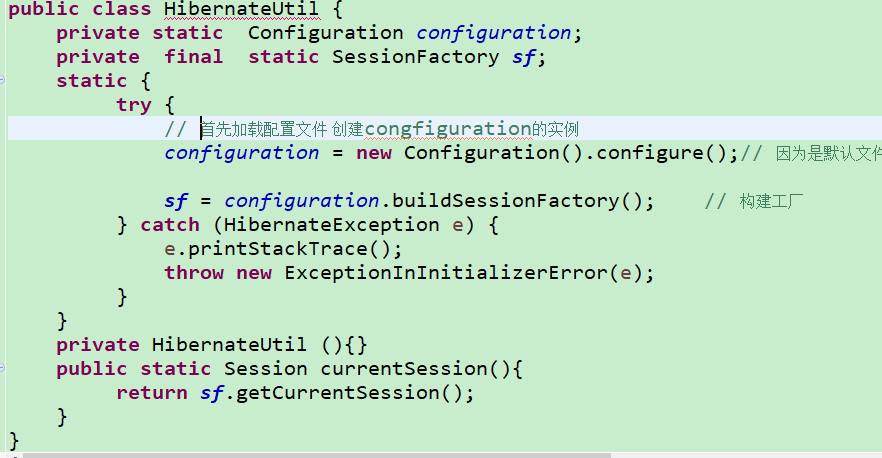
|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.service.user;  import org.springframework.transaction.annotation.Transactional;  import cn.mayiwen.entity.User;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public interface UserService {  /\*@Transactional(readOnly = false)\*/  public void save(User user);  } |

#### UserServiceImpl

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.service.user;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import cn.mayiwen.dao.user.UserDao;  import cn.mayiwen.entity.User;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  @Service("userService")  public class UserServiceImpl implements UserService {  @Autowired  private UserDao userdao;    /\* (non-Javadoc)  \* @see cn.mayiwen.service.user.UserService#save(cn.mayiwen.entity.User)  \*/  @Override  public void save(User user) {  // TODO Auto-generated method stub  userdao.save(user);  }  } |

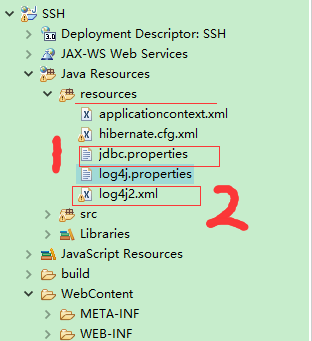
#### HibernateDaoSupport



这个类可以这么理解 

这货你熟悉不，没错，我就把它当这货。对错无关，这样好理解

### 3.2hibernate与spring的配置文件



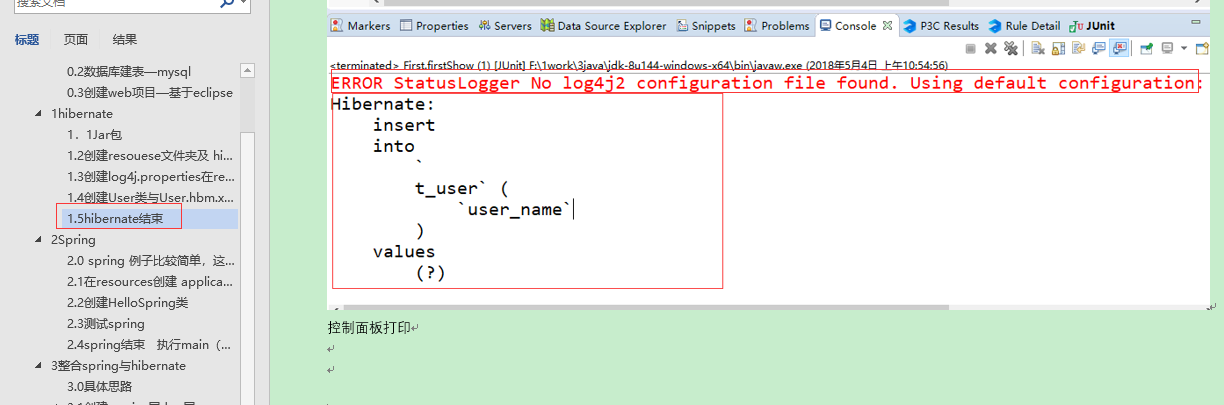
#### jdbc.properties

|  |
| --- |
| 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\_ssh?useUnicode=true&characterEncoding=utf-8  user=root  password=123456  #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 |

#### log4j2.xml

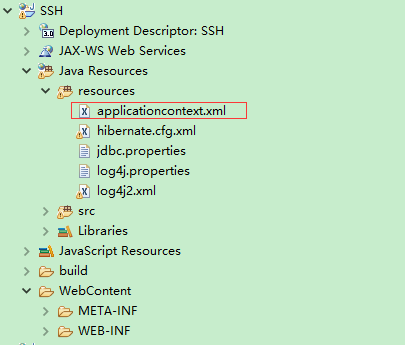
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Configuration status="warn">  <Appenders>  <Console name="Console" target="SYSTEM\_OUT">  <PatternLayout pattern="[%-5p] %d %c - %m%n" />  </Console>  <File name="File" fileName="dist/my.log">  <PatternLayout pattern="%m%n" />  </File>  </Appenders>  <Loggers>  <Logger name="mh.sample2.Log4jTest2" level="INFO">  <AppenderRef ref="File" />  </Logger>  <Root level="INFO">  <AppenderRef ref="Console" />  </Root>  </Loggers>  </Configuration> |

还记得这个吗？



这就是没有log4j2.xml的原因

#### 【修改】applicationcontext.xml

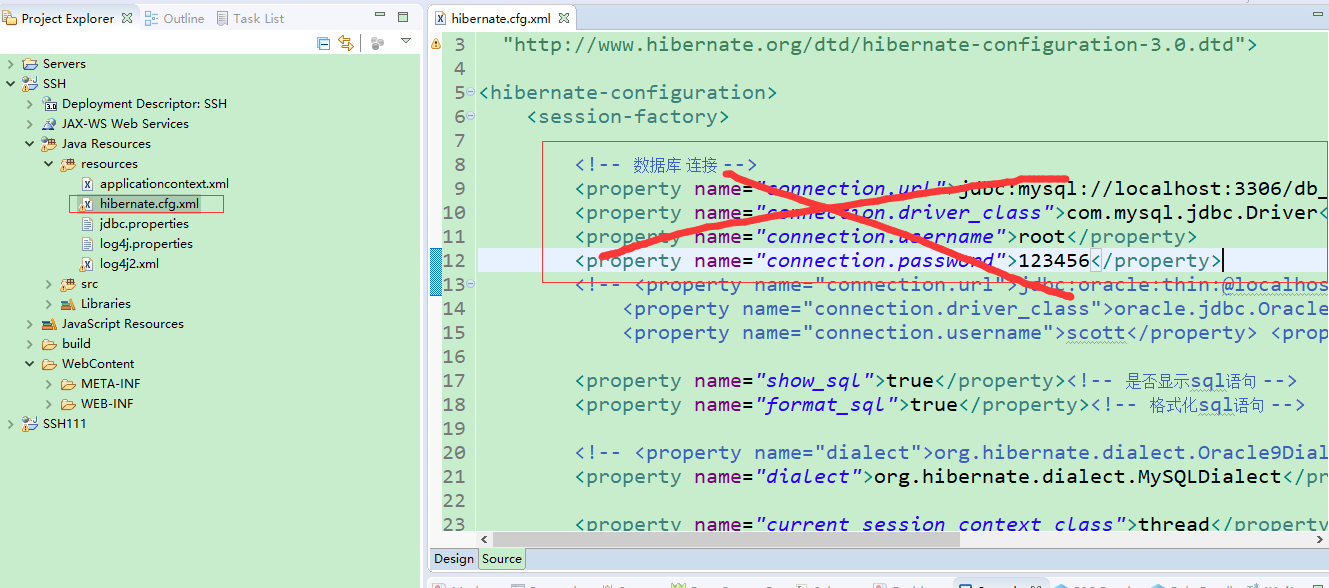


applicationcontext.xml

修改后

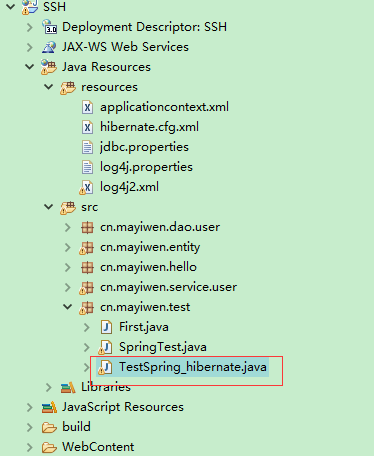
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:cache="http://www.springframework.org/schema/cache"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:jdbc="http://www.springframework.org/schema/jdbc" xmlns:jee="http://www.springframework.org/schema/jee"  xmlns:jms="http://www.springframework.org/schema/jms" xmlns:lang="http://www.springframework.org/schema/lang"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:oxm="http://www.springframework.org/schema/oxm"  xmlns:p="http://www.springframework.org/schema/p" xmlns:task="http://www.springframework.org/schema/task"  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:util="http://www.springframework.org/schema/util"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.1.xsd  http://www.springframework.org/schema/cache http://www.springframework.org/schema/cache/spring-cache-4.1.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd  http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.1.xsd  http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.1.xsd  http://www.springframework.org/schema/jms http://www.springframework.org/schema/jms/spring-jms-4.1.xsd  http://www.springframework.org/schema/lang http://www.springframework.org/schema/lang/spring-lang-4.1.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd  http://www.springframework.org/schema/oxm http://www.springframework.org/schema/oxm/spring-oxm-4.1.xsd  http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-4.1.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.1.xsd  http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.1.xsd">  <!-- 引入数据源 -->  <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sessionFactoryBean -->  <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">  <property name="dataSource" ref="dataSource"></property>  <property name="configLocation" value="classpath:hibernate.cfg.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>    <!-- 扫描注解定义 -->  <context:component-scan base-package="cn.mayiwen"></context:component-scan>    <!-- 定义事物管理器 -->  <bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">  <!-- 创建事务管理器, 管理sessionFactory(因为所有的session都是从sessionFactory获取的) -->  <property name="sessionFactory" ref="sessionFactory" />  </bean>    <!-- 使用注解 @Transactional 使用事物  <tx:annotation-driven transaction-manager="transactionManager"/> -->    <tx:advice id="txAdvice" transaction-manager="transactionManager">  <tx:attributes>  <tx:method name="find\*" read-only="true"/>  <tx:method name="search\*" read-only="true"/>  <tx:method name="query\*" read-only="true"/>  <tx:method name="add\*" propagation="REQUIRED" />  <tx:method name="save\*" propagation="REQUIRED" />  <tx:method name="register\*" propagation="REQUIRED" />  <tx:method name="del\*" propagation="REQUIRED" />  <tx:method name="update\*" propagation="REQUIRED" />  <tx:method name="do\*" propagation="REQUIRED" />  <tx:method name="\*" propagation="REQUIRED" read-only="true"/>    </tx:attributes>    </tx:advice>  <aop:config>  <aop:pointcut expression="execution(\* cn.mayiwen.service.\*.\*.\*(..))" id="serviceMethod"/>  <aop:advisor advice-ref="txAdvice" pointcut-ref="serviceMethod"/>  </aop:config>      <bean id="userDao" class="cn.mayiwen.dao.user.UserDaoImpl">  <property name="sessionFactory" ref="sessionFactory"></property>  </bean>      <bean id="helloSpring" class="cn.mayiwen.hello.HelloSpring">  </bean>        </beans> |

#### 【修改】hibernate.cfg.xml 连接数据库的任务就给spring了



|  |
| --- |
| <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory>        <property name="show\_sql">true</property><!-- 是否显示sql语句 -->  <property name="format\_sql">true</property><!-- 格式化sql语句 -->    <!-- <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>设置方言 oracle -->  <property name="dialect">org.hibernate.dialect.MySQLDialect</property><!-- 设置方言 -->    <property name="current\_session\_context\_class">thread</property><!-- 线程绑定 -->  <!-- <property name="hibernate.hbm2ddl.auto">create(false,false)</property> --><!-- 若没有此表则创建此表 -->  <!-- 引入配置 -->  <!-- <mapping resource="org/hibernate/test/legacy/Simple.hbm.xml" /> -->  <mapping resource="cn/mayiwen/entity/User.hbm.xml" />  </session-factory>  </hibernate-configuration> |

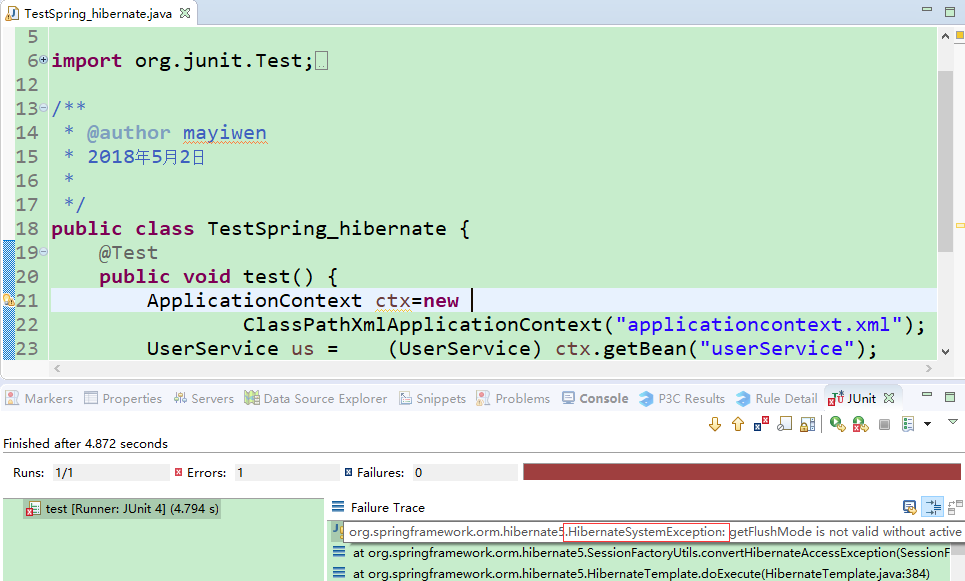
### 3．3编写测试



TestSpring\_hibernate

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.test;  import org.junit.Test;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  import cn.mayiwen.entity.User;  import cn.mayiwen.service.user.UserService;  /\*\*  \* @author mayiwen  \* 2018年5月2日  \*  \*/  public class TestSpring\_hibernate {  @Test  public void test() {  ApplicationContext ctx=new  ClassPathXmlApplicationContext("applicationcontext.xml");  UserService us = (UserService) ctx.getBean("userService");  User user = new User();  user.setName("myw122");  us.save(user);  }  } |

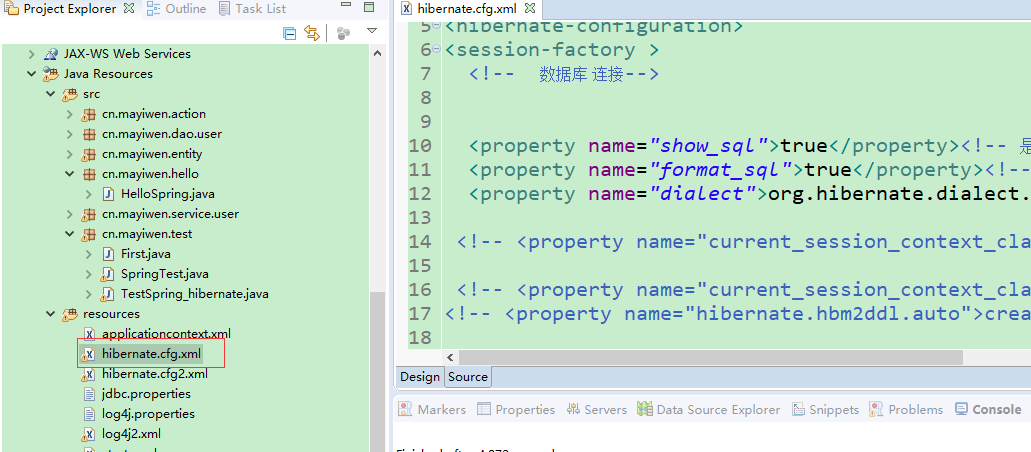
### 3.4，测试错误与解决错误



Hibernate出现错误



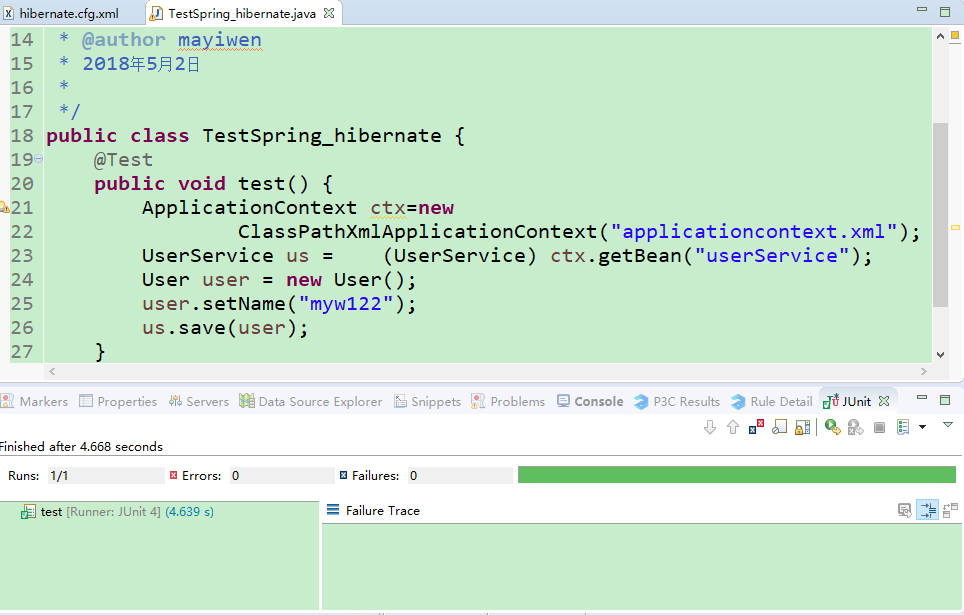
好注释掉



注释后的代码

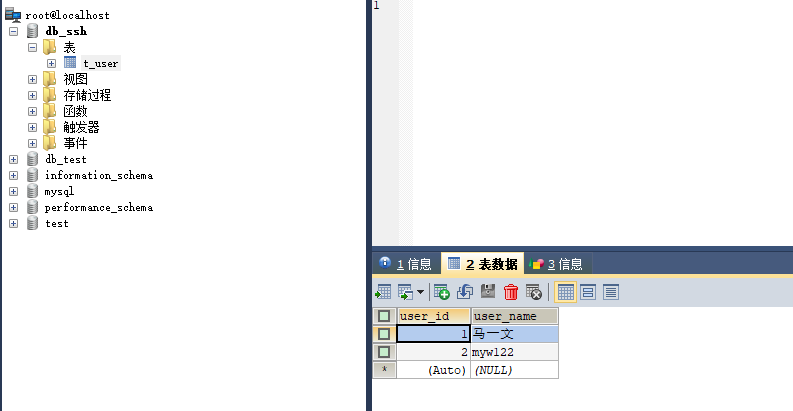
|  |
| --- |
| <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory>        <property name="show\_sql">true</property><!-- 是否显示sql语句 -->  <property name="format\_sql">true</property><!-- 格式化sql语句 -->    <!-- <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>设置方言 oracle -->  <property name="dialect">org.hibernate.dialect.MySQLDialect</property><!-- 设置方言 -->    <!--<property name="current\_session\_context\_class">thread</property> 线程绑定 -->  <!-- <property name="hibernate.hbm2ddl.auto">create(false,false)</property> --><!-- 若没有此表则创建此表 -->  <!-- 引入配置 -->  <!-- <mapping resource="org/hibernate/test/legacy/Simple.hbm.xml" /> -->  <mapping resource="cn/mayiwen/entity/User.hbm.xml" />  </session-factory>  </hibernate-configuration> |

测试



测试成功

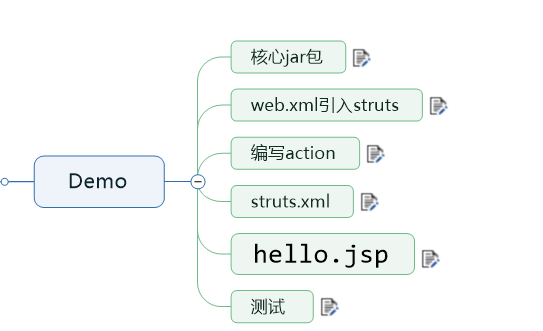
### 3.5.spring 与 hibernate 整合结束



好 ，下一步 ，先写个demo

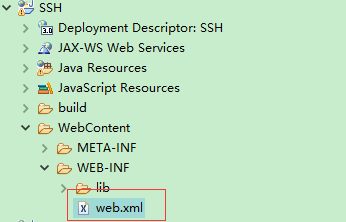
## 4.编写struts

### 4.0思路



很简单的几步，jar包都导进去了。剩下的很简单

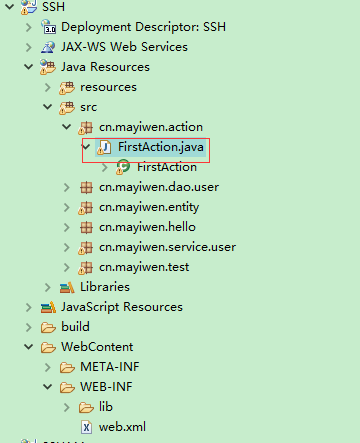
### 4.1web.xml引入struts



Web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID" version="3.0">  <display-name>SSH</display-name>  <filter>  <filter-name>Struts2</filter-name>  <filter-class>  org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter  </filter-class>  </filter>    <filter-mapping>  <filter-name>Struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>  </web-app> |

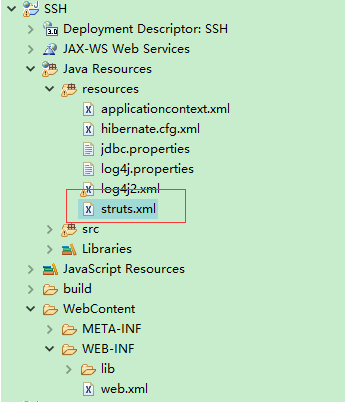
### 4.2编写action



FirstAction

|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.action;  import com.opensymphony.xwork2.ActionSupport;  /\*\*  \* @author mayiwen  \* 2018年5月3日  \*  \*/  public class FirstAction extends ActionSupport {  /\* (non-Javadoc)  \* @see com.opensymphony.xwork2.ActionSupport#execute()  \*/  @Override  public String execute() throws Exception {  // TODO Auto-generated method stub  return SUCCESS;  }  } |

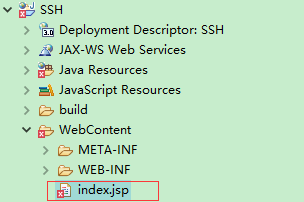
### 4.3struts.xml



struts.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"  "http://struts.apache.org/dtds/struts-2.5.dtd">    <struts><!-- 这是Struts 的标签 -->  <!-- 开发者模式 -->  <constant name="struts.devMode" value="true" />    <constant name="struts.enable.DynamicMethodInvocation" value="true" />  <!-- 包 name是包的名字 extends 继承自 struts-default 包-->  <package name="HELLO" extends="struts-default" namespace="/">    <action name="first"  class="cn.mayiwen.action.FirstAction">  <result name="success">/index.jsp</result>  </action>    </package>    </struts> |

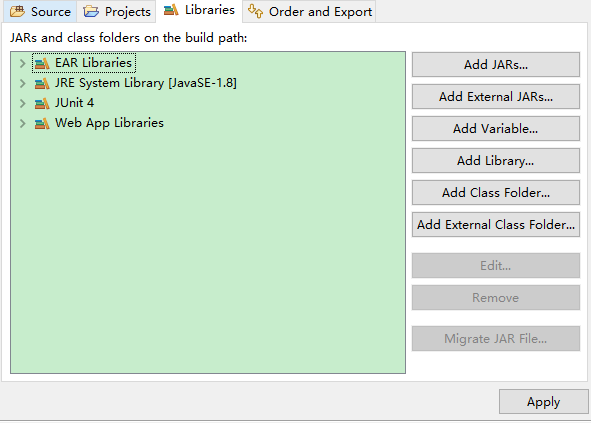
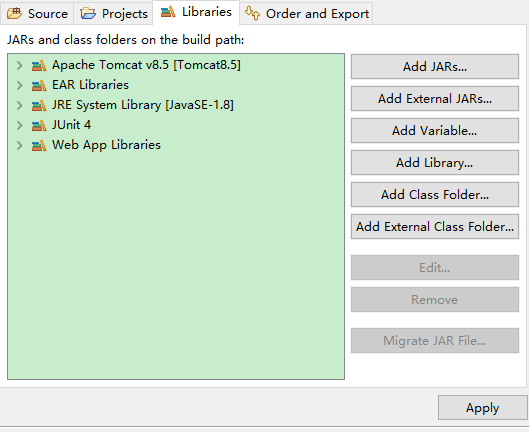
4.4Index.jsp



Index.jsp

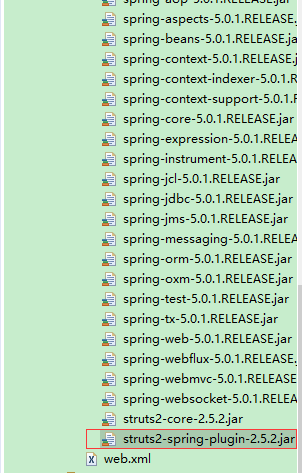
|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Insert title here</title>  </head>  <body>  <h1>你好</h1>  </body>  </html> |

新建jsp出现错误，我是没添加server 加上

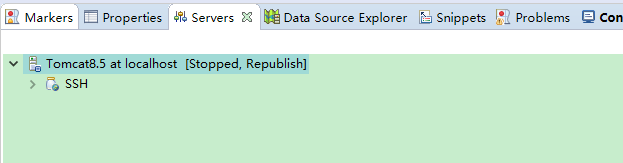
ok了

### 4.3删除jar包

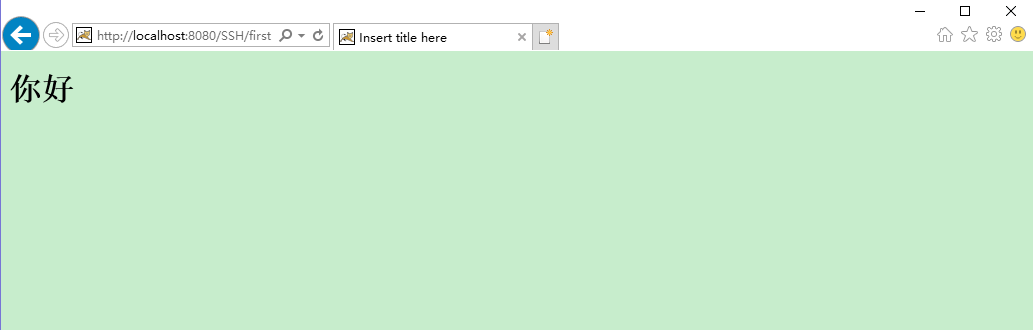


不然跑不起来

### 4.4做测试



跑起来 访问 http://localhost:8080/SSH/first



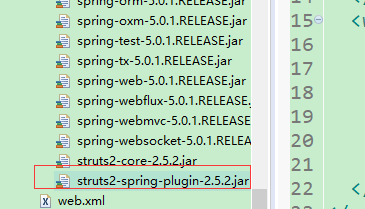
### 4.5 整理

此时项目里有 spring+hibernate整合好的，与struts一个

现在只要整合spring与struts就可以了。

## 5SSH整合

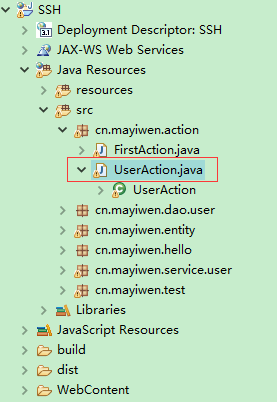
### 5.1jar包



### 5.2修改web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">  <display-name>SSH111</display-name>  <!-- context-param->listener -> filter -> servlet -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationcontext.xml</param-value>  <!-- <param-value>classpath:applicationContext\*.xml</param-value> -->  </context-param>  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class>  </listener>  <!--contextConfigLocation在 ContextLoaderListener类中的默认值是 /WEB-INF/applicationContext.xml-->    <!-- struts过滤器 -->  <filter>  <filter-name>Struts2</filter-name>  <filter-class>  org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter  </filter-class>  </filter>      <filter>  <filter-name>OpenSessionInViewFilter</filter-name>  <filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>OpenSessionInViewFilter</filter-name>  <url-pattern>\*.action</url-pattern>  </filter-mapping>  <filter-mapping>  <filter-name>Struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>  </web-app> |

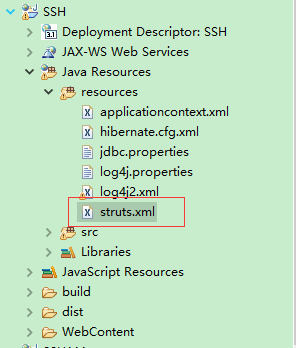
### 5.3创建UserAction



UserAction

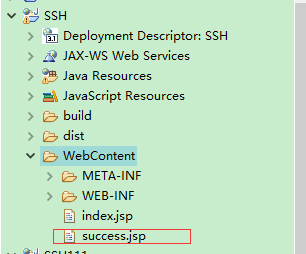
|  |
| --- |
| /\*\*  \*  \*/  package cn.mayiwen.action;  import org.springframework.beans.factory.annotation.Autowired;  import com.opensymphony.xwork2.ActionSupport;  import cn.mayiwen.entity.User;  import cn.mayiwen.service.user.UserService;  /\*\*  \* @author mayiwen  \* 2018年5月3日  \*  \*/  public class UserAction extends ActionSupport {  private String name;//接收姓名  @Autowired  private UserService userservice;              /\* (non-Javadoc)  \* @see com.opensymphony.xwork2.ActionSupport#execute()  \*/  @Override  public String execute() throws Exception {  // TODO Auto-generated method stub  User user = new User();  user.setName(this.getName());  userservice.save(user);  return SUCCESS;  }  /\*\*  \* @return the name  \*/  public String getName() {  return name;  }  /\*\*  \* @param name the name to set  \*/  public void setName(String name) {  this.name = name;  }  /\*\*  \* @return the userservice  \*/  public UserService getUserservice() {  return userservice;  }  /\*\*  \* @param userservice the userservice to set  \*/  public void setUserservice(UserService userservice) {  this.userservice = userservice;  }        } |

### 5.4Struts配置



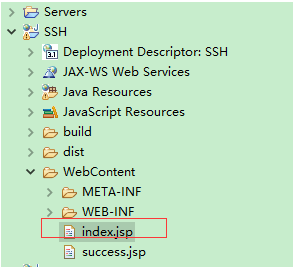
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"  "http://struts.apache.org/dtds/struts-2.5.dtd">    <struts><!-- 这是Struts 的标签 -->  <!-- 开发者模式 -->  <constant name="struts.devMode" value="true" />    <constant name="struts.enable.DynamicMethodInvocation" value="true" />  <!-- 包 name是包的名字 extends 继承自 struts-default 包-->  <package name="HELLO" extends="struts-default" namespace="/">    <action name="first"  class="cn.mayiwen.action.FirstAction">  <result name="success">/index.jsp</result>  </action>  <action name="user"  class="cn.mayiwen.action.UserAction">  <result name="success">/success.jsp</result>  </action>  </package>    </struts> |

### 5.5Success.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>Insert title here</title>  </head>  <body>  添加成功  </body>  </html> |

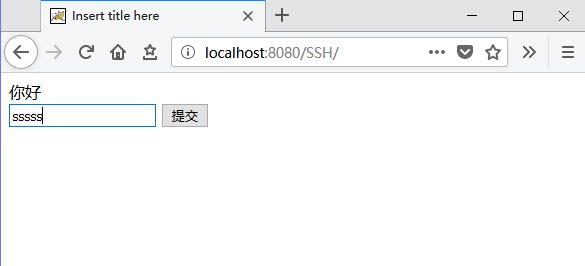
### 5.6修改index.jsp



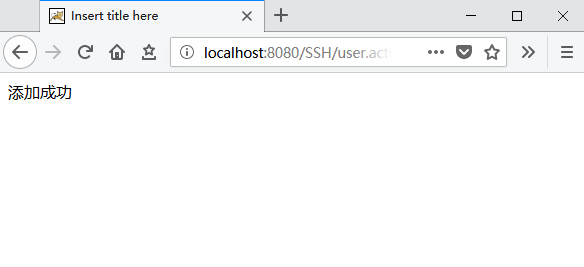
|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Insert title here</title>  </head>  <body>  你好  <form action="user.action" method="post">  <input type="text" name="name">  <input type="submit" value="提交">  </form>  </body>  </html> |

### 5.7测试

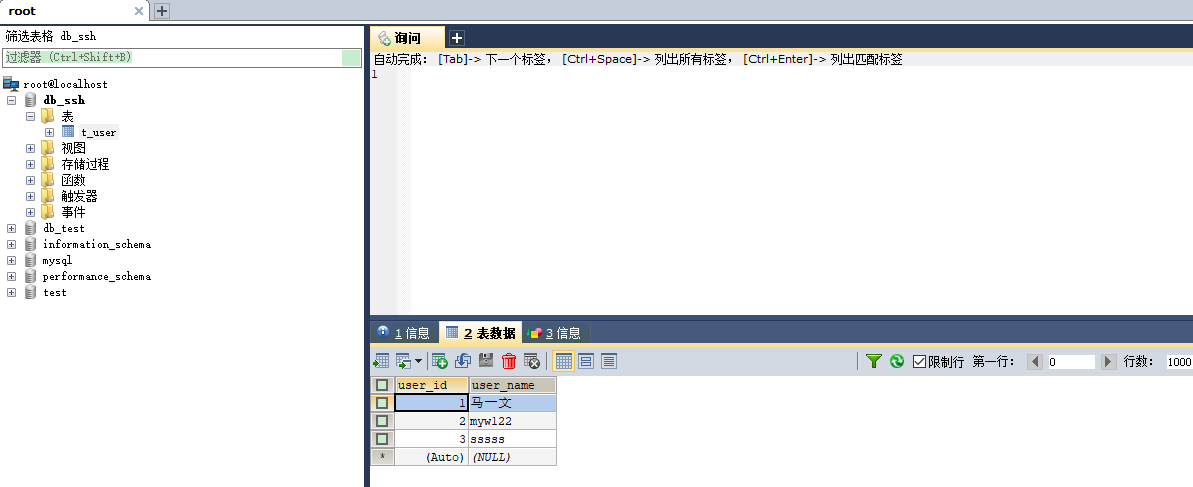
#### 输入<http://localhost:8080/SSH/>



### 提交



#### 数据库



整合完成

