

# **基于 MyEclipse6.5 的 SSH 整合**

**hespermoon**

**2008-7-18**

## 1. 编写目的

最近在学习 Struts1.2+Spring2.5+Hibernate3.2 整合，这期间出现了很多问题，在网上找了若干的实例均有不同程度的问题，为了让想要学习这个整合的人少走弯路，特写下这篇文章，希望对大家有所帮助，如果有很么问题可以与我联系，Email: [zhaohuawei@live.cn](mailto:zhaohuawei@live.cn)。

## 2. 实验环境

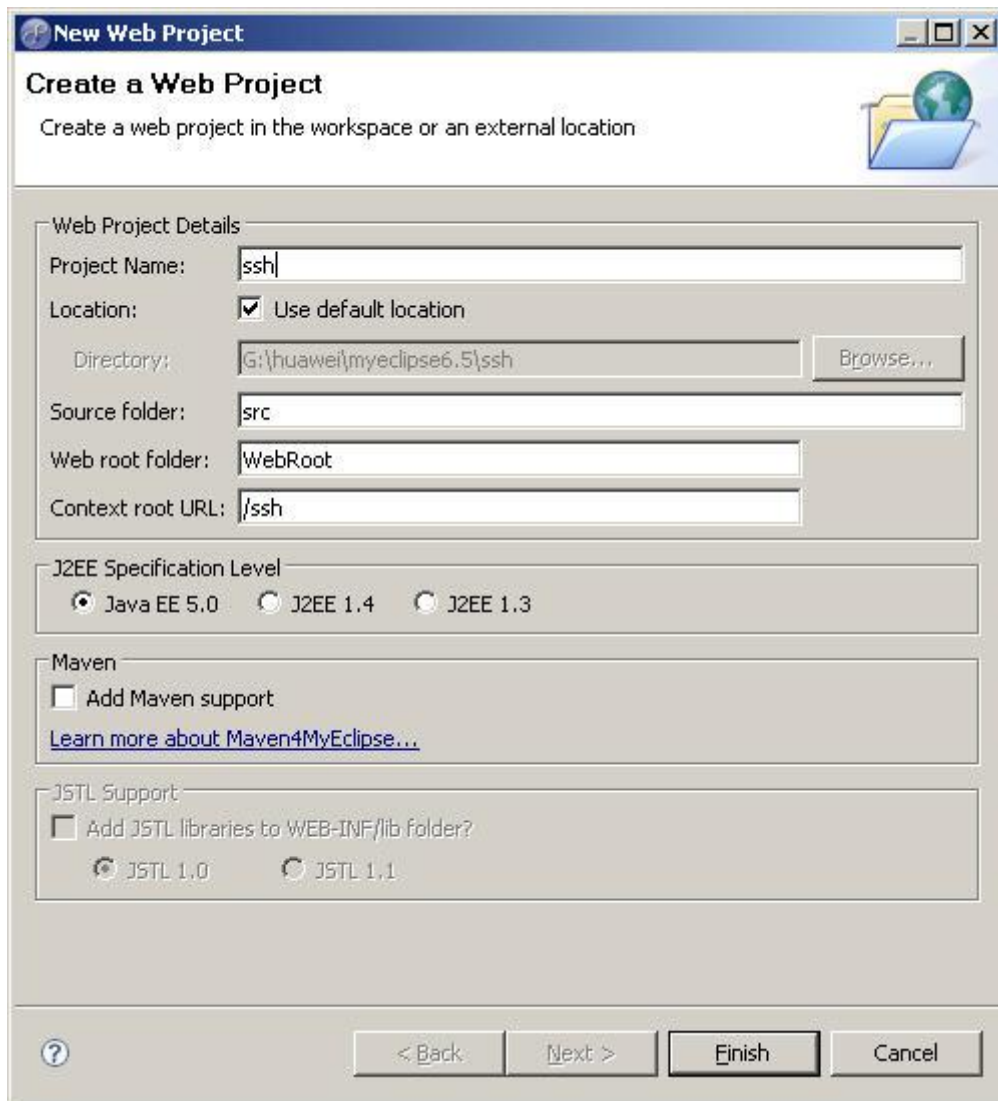
- MyEclipse6.5
- Tomcat5.5.26
- MySQL5.0
- 数据库脚本程序

```
CREATE TABLE user (  
    id int(11) NOT NULL auto_increment,  
    username varchar(50) default NULL,  
    password varchar(50) default NULL,  
    PRIMARY KEY (id)  
);  
  
INSERT INTO user VALUES ('1', 'admin', 'admin');
```

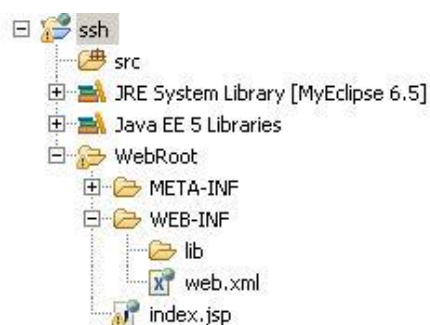
环境的搭建这里就不写了，估计大家应该很熟悉，不会的可以上网看一下，有很多这样的资料。

## 3. Go! 让我们开始创建工程

打开 MyEclipse，需要注意的是这里应该处于 MyEclipse Java Enterprise 视图；新建一个 Web Project，输入适当的工程名字，这里我们输入 ssh，Finish 即可。

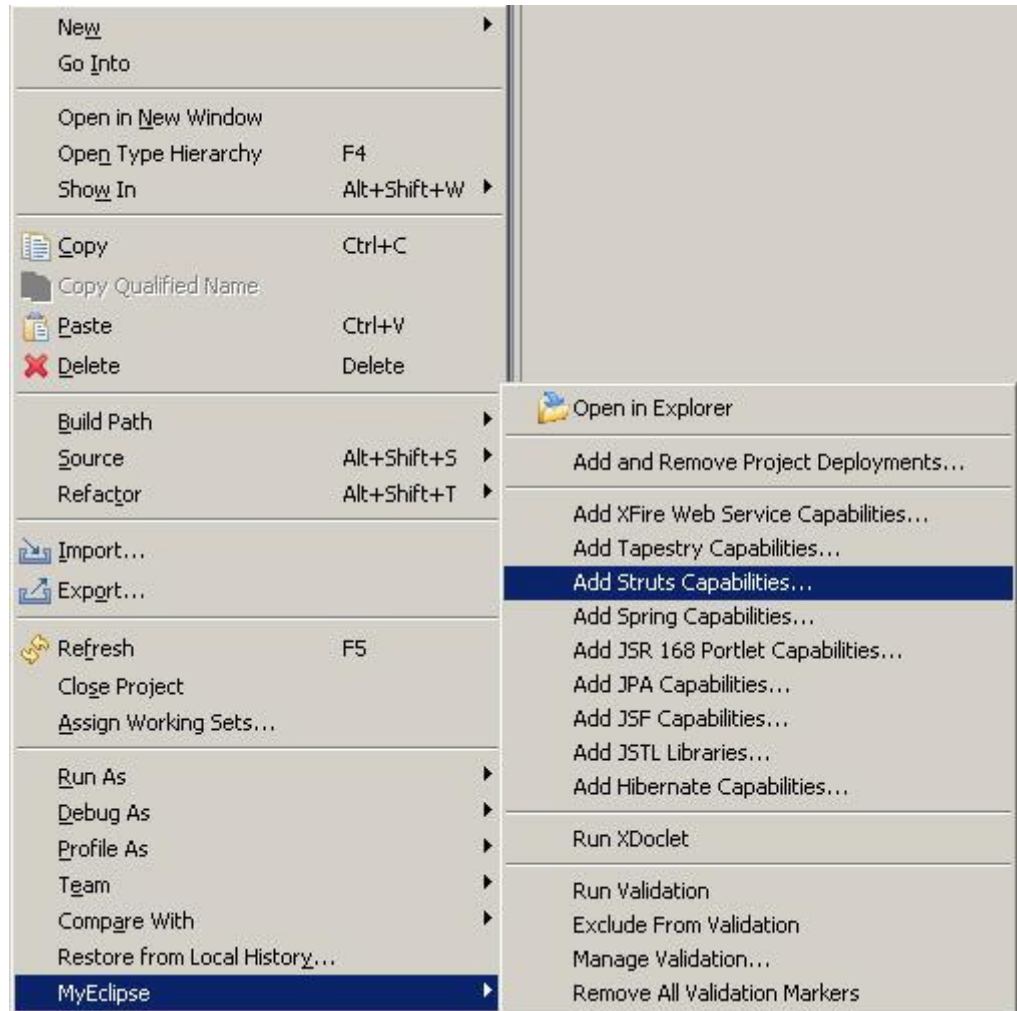


确定之后 MyEclipse 会生成名为 SSH 的项目，其中生成的目录结构如下所示：



## 4. 添加 Struts1.2 框架支持

在 ssh 工程上面右击，在弹出的菜单中选择 MyEclipse -> Add Struts Capabilities...，添加 Struts 的支持。



在弹出的对话框中选择 Struts 1.2，修改 Base package for new classes 成所需的包名，其余保持原状，Finish 即可

**Add Struts Capabilities**

**Struts Support for MyEclipse Web Project**  
Enable project for Struts development

Web project: ssh  
Web-root folder: /WebRoot  
Servlet specification: 2.5

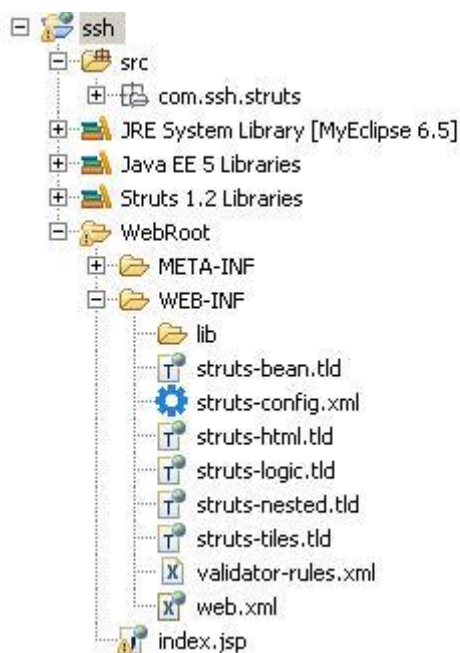
Struts config path: /WEB-INF/struts-config.xml Browse...  
Struts specification: ☐ Struts 1.1 ☒ Struts 1.2 ☐ Struts 1.3  
ActionServlet name: action  
URL pattern: ☒ \*.do ☐ /do/\*

Base package for new classes: com.ssh.struts Browse...  
Default application resources: com.ssh.struts.ApplicationResources

☒ Install Struts TLDs [View libraries...](#)

? Finish Cancel

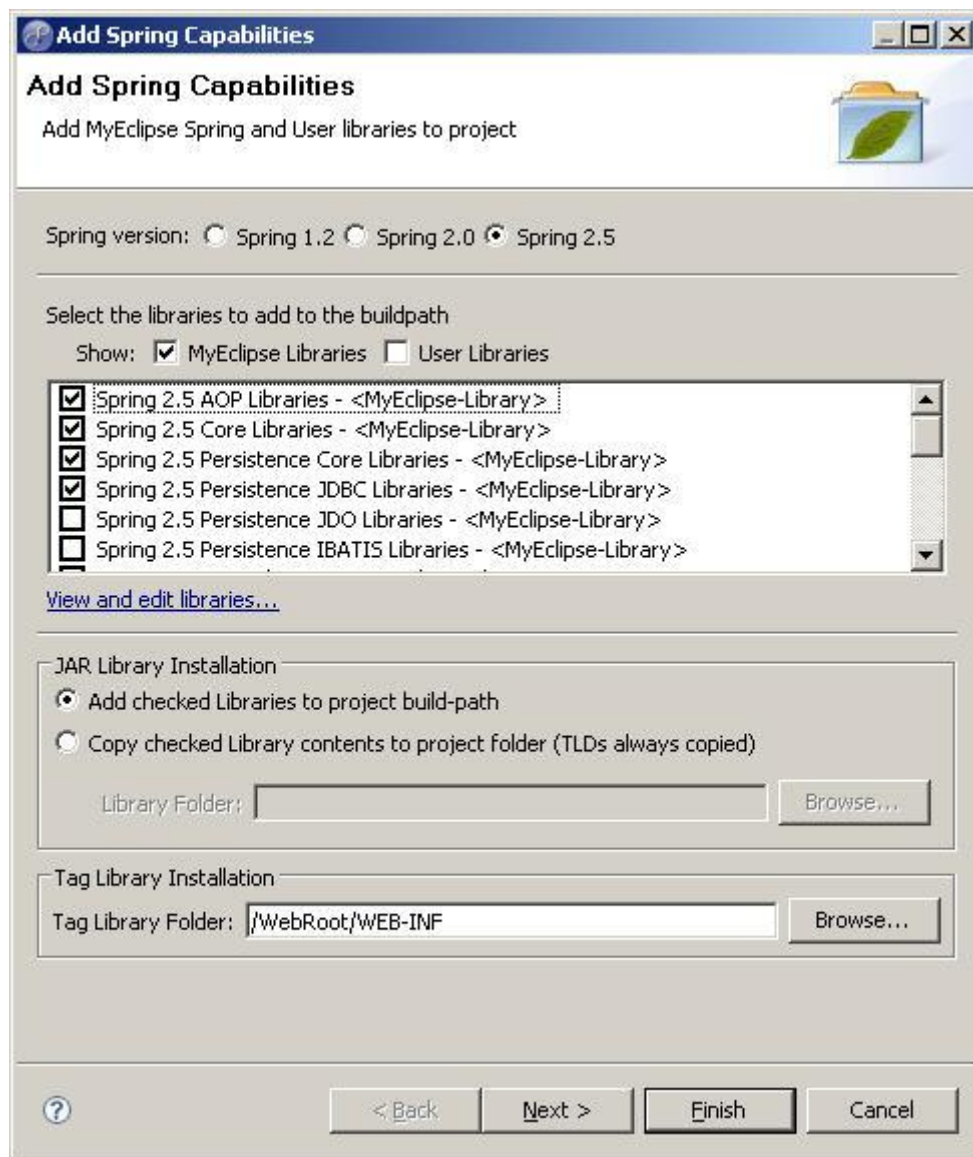
完成后的结构为：



## 5. 添加 Spring2.0 框架支持

在 ssh 工程上面右击，在弹出的菜单中选择 MyEclipse -> Add Spring Capabilities...，添加 Spring 框架支持

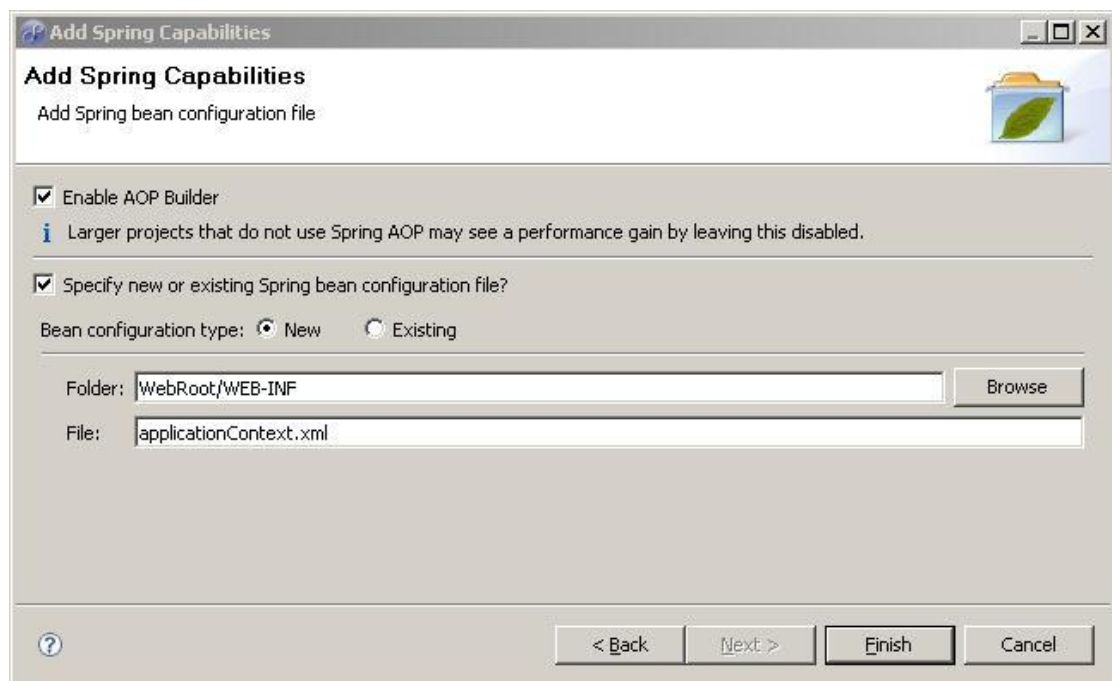




由于 Spring 采取最小化库发布的方式,使得 Spring 的库文件彼此都是分离的,因而我们需要自己选择需要的库, 需要引用的如下图:

- ☒ Spring 2.5 AOP Libraries - <MyEclipse-Library>
- ☒ Spring 2.5 Core Libraries - <MyEclipse-Library>
- ☒ Spring 2.5 Persistence Core Libraries - <MyEclipse-Library>
- ☒ Spring 2.5 Persistence JDBC Libraries - <MyEclipse-Library>
- ☐ Spring 2.5 Persistence JDO Libraries - <MyEclipse-Library>
- ☐ Spring 2.5 Persistence IBATIS Libraries - <MyEclipse-Library>
- ☐ Spring 2.5 J2EE Libraries - <MyEclipse-Library>
- ☐ Spring 2.5 Remoting Libraries - <MyEclipse-Library>
- ☐ Spring 2.5 Misc Libraries - <MyEclipse-Library>
- ☒ Spring 2.5 Testing Support Libraries - <MyEclipse-Library>
- ☒ Spring 2.5 Web Libraries - <MyEclipse-Library>
- ☐ Spring Webflow 1.0 Core - <MyEclipse-Library>
- ☐ Spring JavaConfig 1.0 Core - <MyEclipse-Library>
- ☐ Toplink Essentials - <MyEclipse-Library>
- ☐ OpenJPA - <MyEclipse-Library>
- ☐ Hibernate 3.2 Core Libraries - <MyEclipse-Library>
- ☐ Hibernate 3.2 Annotations & Entity Manager - <MyEclipse-Library>
- ☐ Hibernate 3.2 Advanced Support Libraries - <MyEclipse-Library>

选择好后 Next, 在窗口中选择 Browse, 选择 ssh 工程下面的 WEB-INF 文件夹, 然后 Finish。



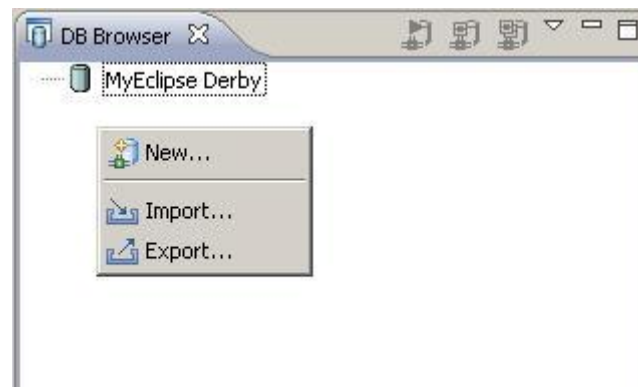


## 6. 配置数据源

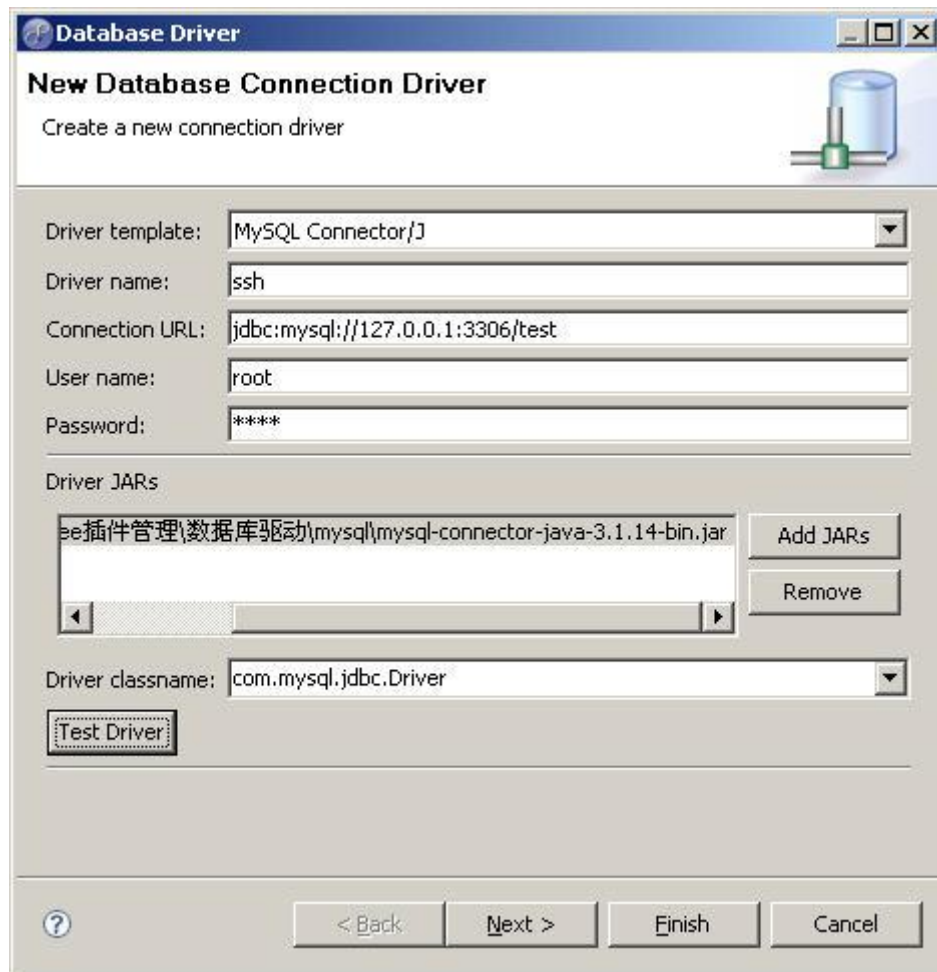
在 MyEclipse 右上角选择 MyEclipse Database Explorer, 打开数据库管理视图。



在左侧的 DB Browser 点击右键, 选择 New..., 打开 Database Driver 对话框。

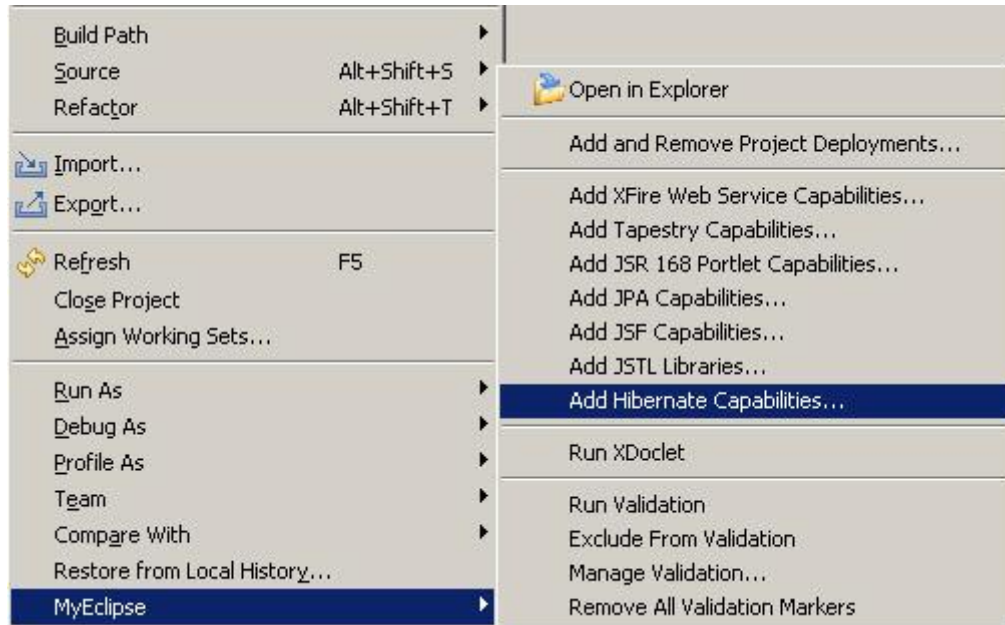


在 Database Driver 对话框中按照如下配置选择数据库驱动。首先选择 Driver Template, 在这里我们选择 MySQL。Driver Name 是以后配置时使用的驱动名字, 用以区分驱动, 这里使用 MySQL 即可。然后根据实际情况填写 URL, User name 和 Password。点击 Add JARs 添加数据库驱动文件。为方便配置, 可以选择 Save password 保存密码。配置完毕后 Finish 即可。

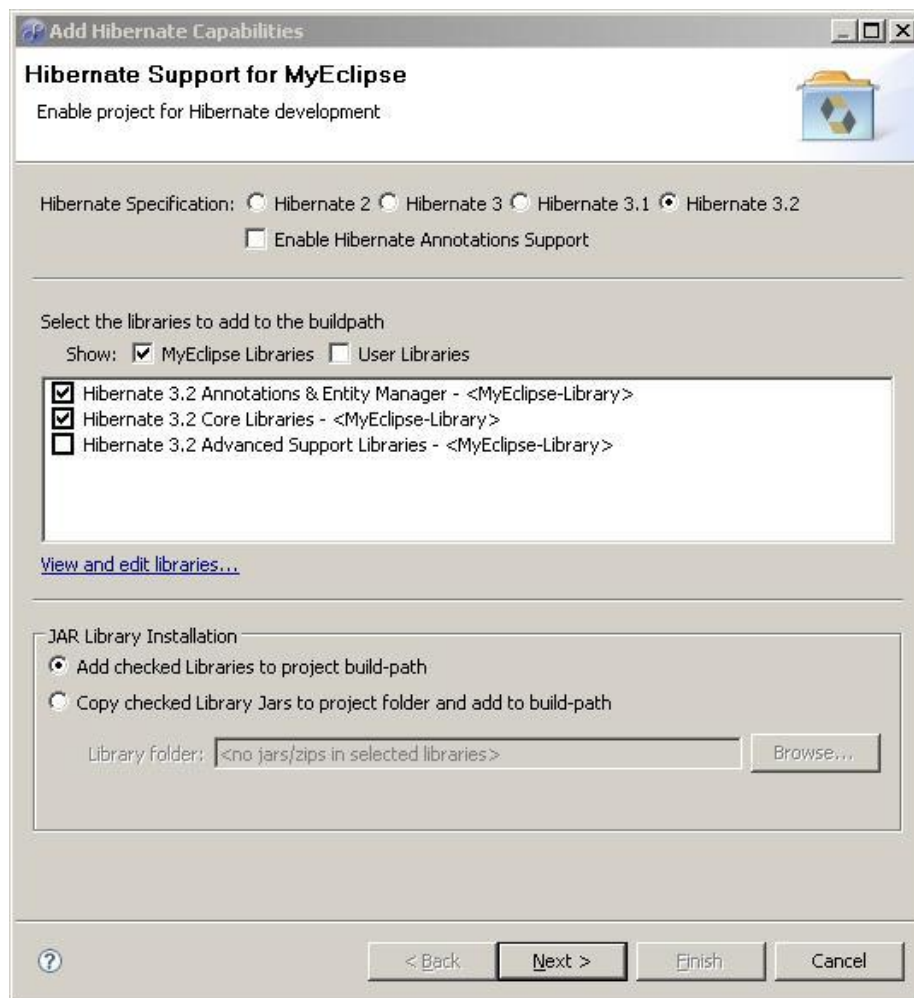


## 7. Spring 与 Hibernate 的整合

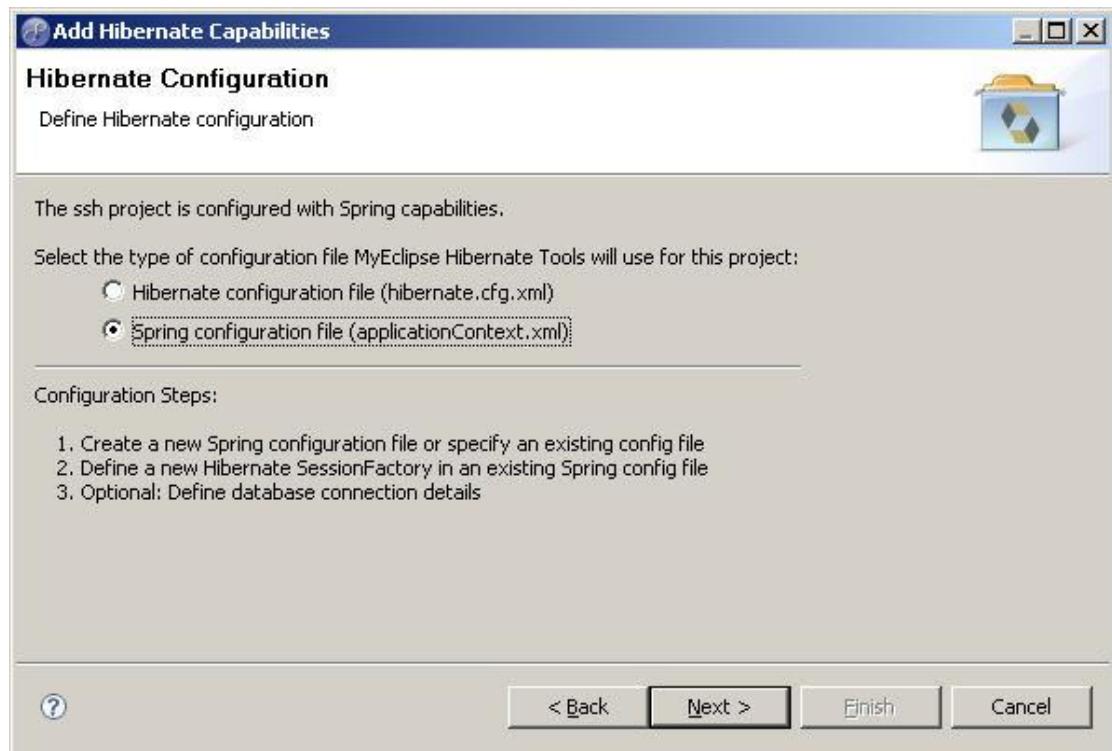
当配置完数据源后，就可以添加 Hibernate 支持了。切换到 MyEclipse Java Enterprise 视图，在 ssh 工程上面右击，在弹出的菜单中选择 MyEclipse -> Add Hibernate Capabilities...，添加 Hibernate 的支持。



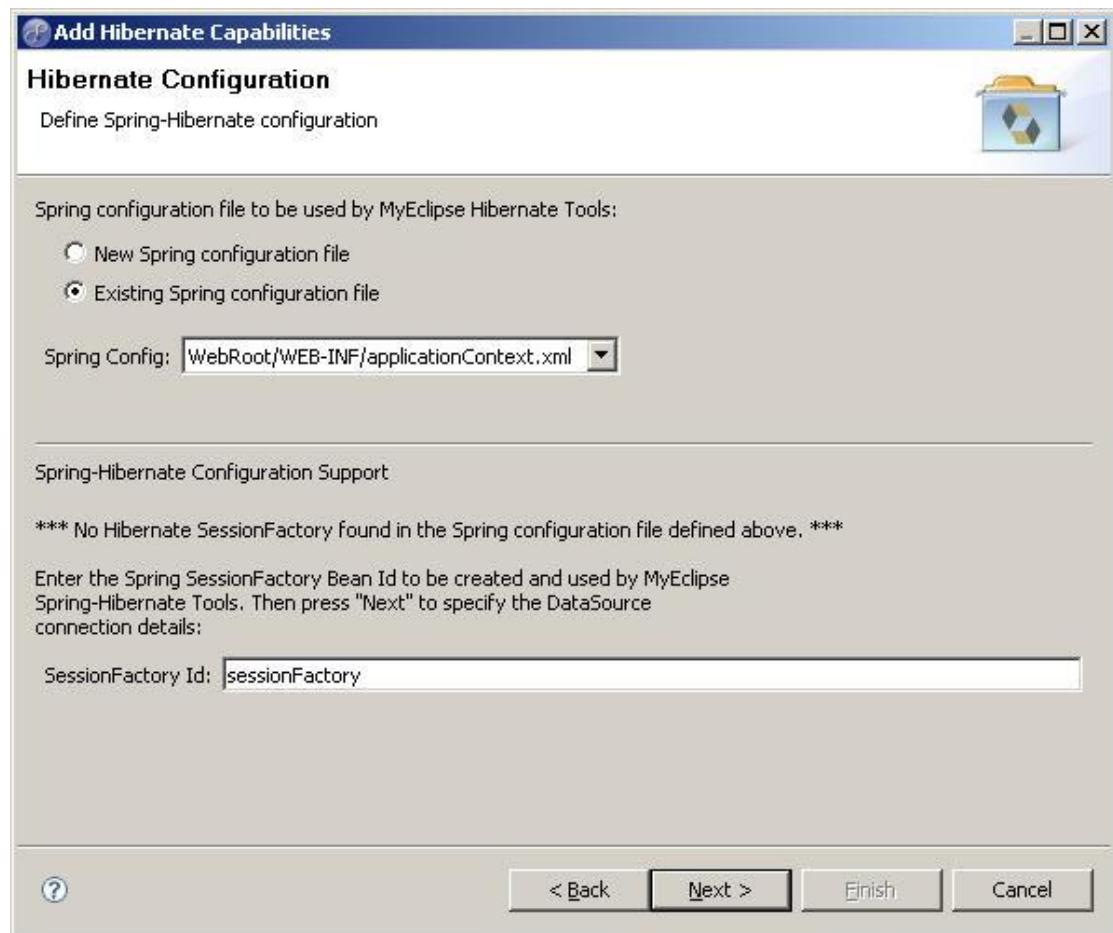
我们选择 Hibernate3.2，全部按照默认设置即可，然后点击 Next;



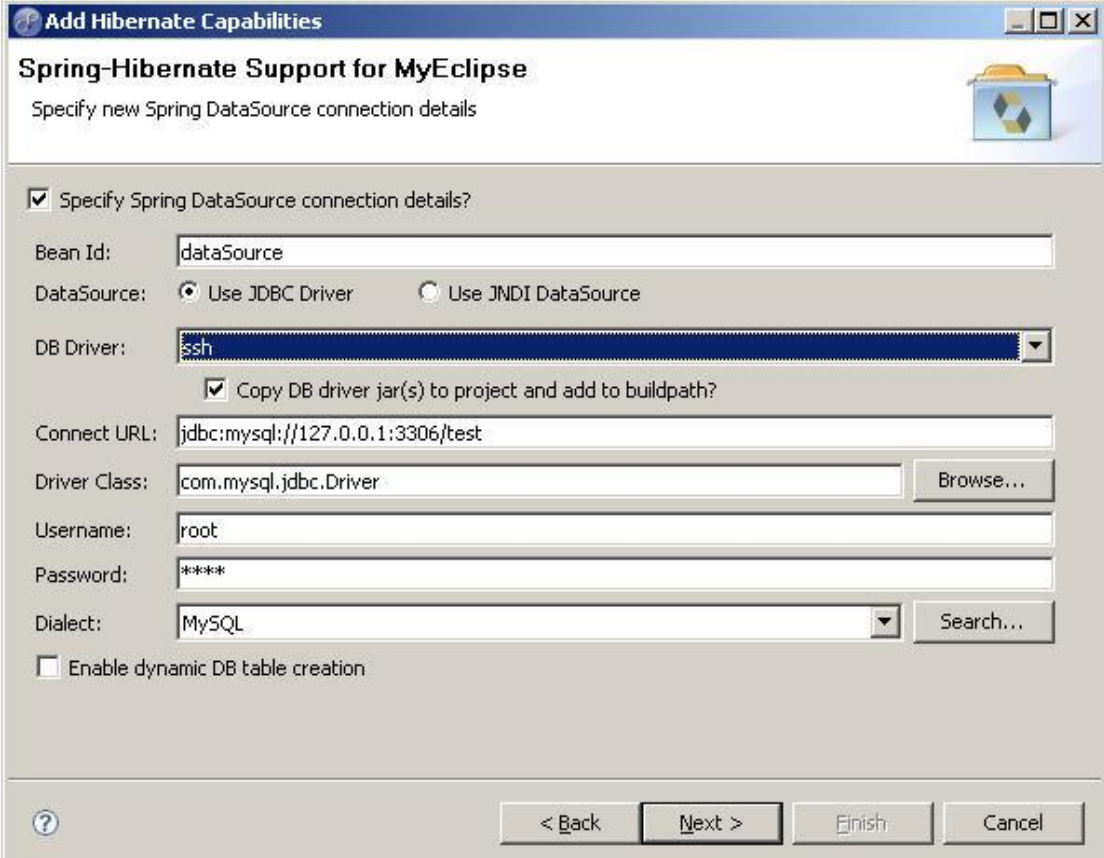
在对话框中选择 Spring configuration file，表示我们希望将 Hibernate 托管给 Spring 进行管理，这是将 Hibernate 与 Spring 进行整合的基础。然后点击 Next；



在出现的对话框中选择 Existing Spring configuration file。因为我们已经添加了 Spring 的配置文件，所以这里选择的是已存在的配置文件。MyEclipse 会自动找到存在的那个文件。然后在 SessionFactory ID 中输入 Hibernate 的 SessionFactory 在 Spring 配置文件中的 Bean ID 的名字，这里我们输入 sessionFactory 即可。然后点击 Next；



在出现的对话框中的 **Bean Id** 里面输入数据源在 **Spring** 中的 **Bean ID** 的名字，这里我们输入 **dataSource**。然后在 **DB Driver** 里面选择我们刚刚配置好的 **ssh**，**MyEclipse** 会将其余的信息自动填写到表格里面。然后点击 **Next**；



**Add Hibernate Capabilities**


**Spring-Hibernate Support for MyEclipse**

Specify new Spring DataSource connection details

☒ Specify Spring DataSource connection details?

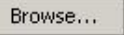
Bean Id:

DataSource: ☒ Use JDBC Driver ☐ Use JNDI DataSource

DB Driver:  


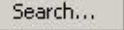
☒ Copy DB driver jar(s) to project and add to buildpath?

Connect URL:






Driver Class:  

Username:

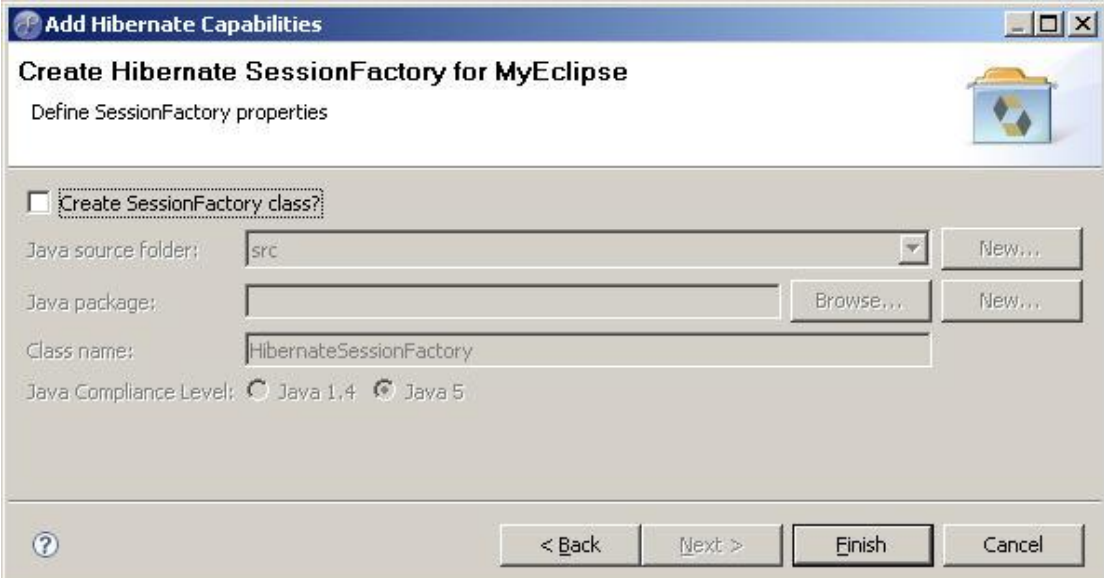
Password:

Dialect:   

☐ Enable dynamic DB table creation

在出现的对话框中取消 Create SessionFactory class。点击 Finish 即可。







**Add Hibernate Capabilities**

**Create Hibernate SessionFactory for MyEclipse**

Define SessionFactory properties




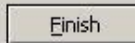
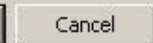
☐ Create SessionFactory class?

Java source folder:   

Java package:   

Class name:

Java Compliance Level: ☐ Java 1.4 ☒ Java 5

此时 MyEclipse 会自动打开 Spring 的配置文件，文件内容为：

```

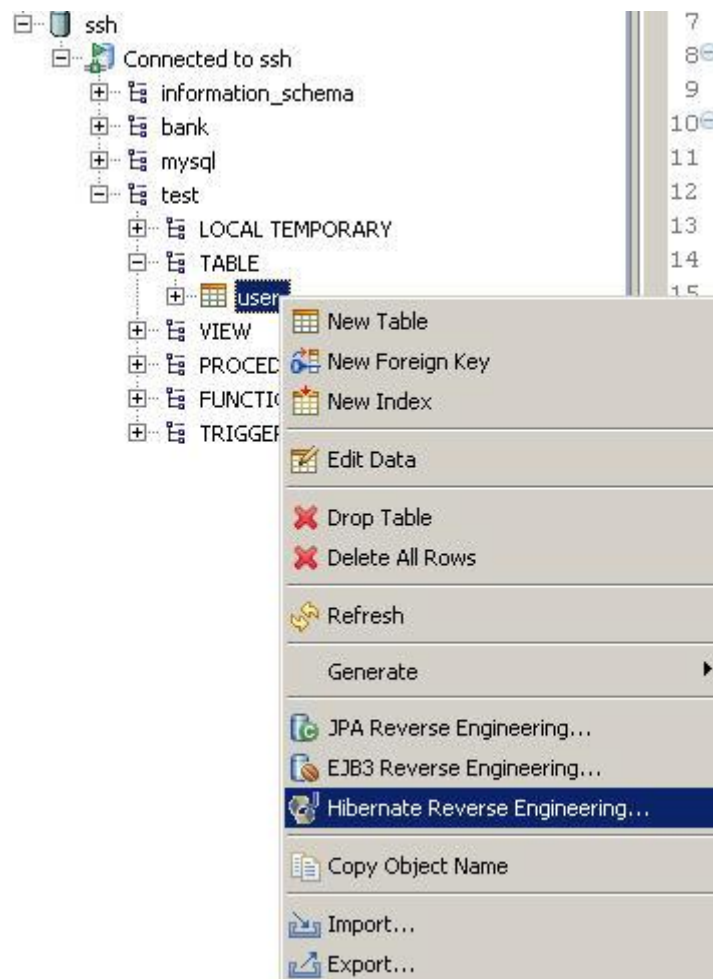
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf

    <bean id="dataSource"
        class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName"
            value="com.mysql.jdbc.Driver">
        </property>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/test"></property>
        <property name="username" value="root"></property>
        <property name="password" value="root"></property>
    </bean>
    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource">
            <ref bean="dataSource" />
        </property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">
                    org.hibernate.dialect.MySQLDialect
                </prop>
            </props>
        </property>
    </bean></beans>

```

此时，切换到 MyEclipse DataBase Explorer 视图，右键打开数据库连接，在需要使用的数据表格上面点击右键，选择 Hibernate Reverse Engineering…。这里我们使用刚刚建立在 test 数据库中添加的 user 表。





在打开的对话框中修改 Java src folder 为我们建立的/ssh/src，这里需要选择到 src 文件夹，并且需要填写 Java package，这是 MyEclipse 生成的类所在的包，我们将其取名为 user。然后选择 Java Data Object，建立 POJO 类。然后选择 Java Data Access Object。其中，POJO 类是数据库表格所对应的 Java 类，JDO 类是 MyEclipse 自动生成的对数据库的一些操作。这里会封装一些常用的操作，简化我们的编写。填写完成后点击 Next。



**Hibernate Reverse Engineering**

### Hibernate Mapping and Application Generation

Generate Hibernate mapping and Java classes from database explorer tables

Java src folder:

Java package:

☒ Create POJO <> DB Table mapping information

- ☒ Create a Hibernate mapping file (\*.hbm.xml) for each database table
- ☐ Add Hibernate mapping annotations to POJO (Hibernate 3.2 only)
- ☒ Update Hibernate configuration with mapping resource location

☒ Java Data Object (POJO <> DB Table)

☐ Create abstract class

Base persistent class:

☒ Java Data Access Object (DAO) (Hibernate 3 only)

☒ Generate precise findBy methods

DAO type: ☐ Basic DAO ☒ Spring DAO ☐ JNDI DAO

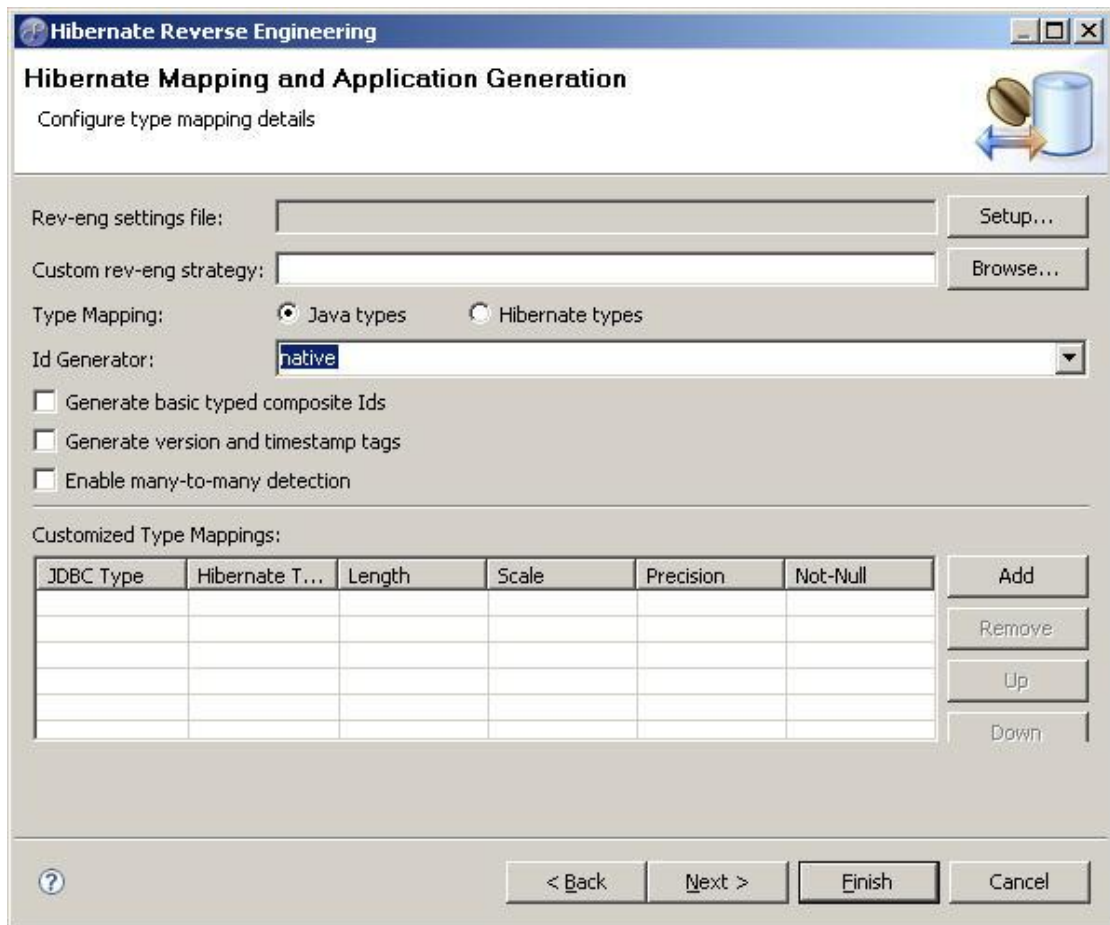
Spring config file:

SessionFactory Id:

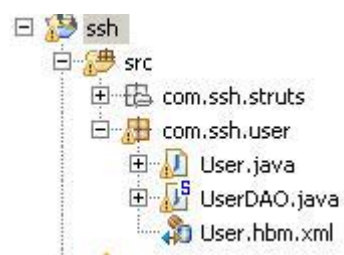
☐ Use custom templates

Template directory:

在出现的对话框中选择 ID Generator 为 native，然后点击 Finish 即可。



此时 ssh 项目的目录结构如下所示。其中的 User.java 是 MyEclipse 生成的使用面向对象的 Java 语言对数据库表格进行的抽象，User.hbm.xml 是将数据库表格中的字段和 POJO 类的属性进行映射的定义，UserDAO.java 封装了一些 MyEclipse 自动生成的对数据库的操作。



这时我们修改一下文件目录，使之更好的组织。我们建立一个 **dao** 包，将 DAO 类与 POJO 类分开。然后我们在 **struts** 包下面建立 **action** 和 **form** 包，用来管理 Struts 的 Action 和 Form。为了将实现与接口进行解耦，我们建议在 **dao** 包下面添加接口，然后建立 **dao.impl** 包，将实际的 DAO 类放在这里。DAO 类是直接 与数据库打交道的类，为了对业务逻辑进行封装，我们将业务全部写在 **service** 类里面，和 **dao** 一样，我们先建立 **service** 包，里面添加业务接口，具体的实现放在 **service.impl** 里面。

将 **UserDAO.java** 移动到 **dao** 的 **impl** 包下面，并在 **dao** 包下建立接口 **IUserDAO**，内容为：

```
package com.ssh.dao;
import java.util.List;
import com.ssh.user.User;

public interface IUserDAO {
    public User findById(Integer id);
    public List findByUsername(Object username);
    public void save(User user);
}
```

在 **service** 中建立接口 **IUserService.java**，在 **service** 中 **impl** 中建立 **UserService.java**

### **IUserService.java**

```
package com.ssh.service;
import com.ssh.user.User;

public interface IUserService {
    public User getUserById(Integer id);
    public User getUserByUsername(String username);
    public void addUser(User user);
}
```

## UserService.java

```
package com.ssh.service.impl;

import java.util.List;
import com.ssh.dao.IUserDAO;
import com.ssh.service.IUserService;
import com.ssh.user.User;

public class UserService implements IUserService {
    private IUserDAO userDAO;

    public void addUser(User user) {
        userDAO.save(user);
    }

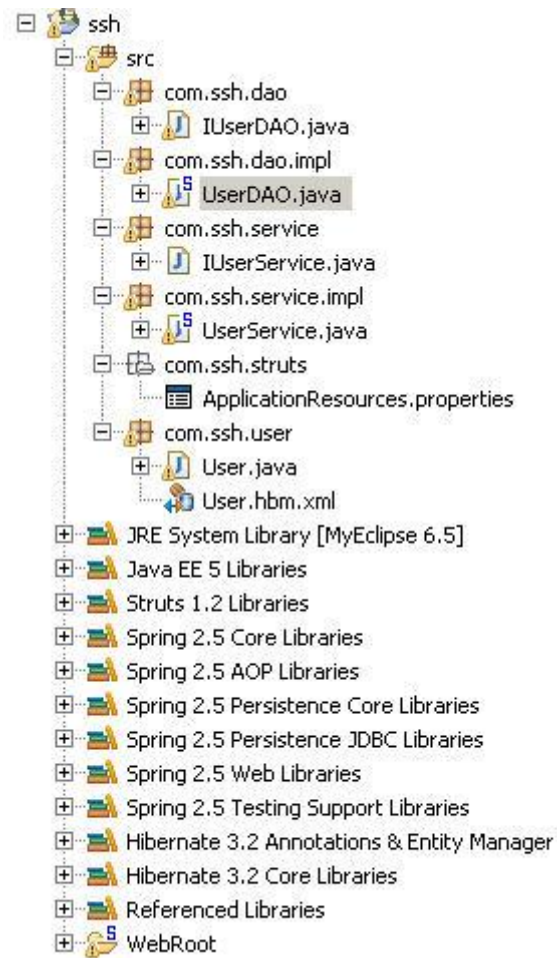
    public User getUserById(Integer id) {
        return userDAO.findById(id);
    }

    public User getUserByUsername(String username) {
        List list = userDAO.findByUsername(username);
        if (list.size() == 0) {
            return null;
        } else {
            return (User) list.get(0);
        }
    }

    public IUserDAO getUserDAO() {
        return userDAO;
    }

    public void setUserDAO(IUserDAO userDAO) {
        this.userDAO = userDAO;
    }
}
```

此时的整体工程结构为：



此时的 applicationContext 内容需要配置 bean 内容为：

```
<bean id="UserDAO" class="com.ssh.dao.impl.UserDAO">
    <property name="sessionFactory">
        <ref bean="sessionFactory" />
    </property>
</bean>
```

## 8. 整合 Struts 和 Spring

Struts 和 Spring 整合方式其核心是让 Struts 能够访问到交给 Spring 进行托管的类, 这个我在网上看到了一个方式, 自我认为很好, 因此只要我们可以让 Struts 从 Spring 获得需要的类就可以了。为了达到这个目标, 我们创建一个类 BaseAction:

```
package com.ssh.struts.action;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;
import org.springframework.web.struts.ActionSupport;

public class BaseAction extends ActionSupport {
    protected Object getBean(String id) {
        WebApplicationContext ctx = WebApplicationContextUtils
            .getWebApplicationContext(this.servlet.getServletContext());
        return ctx.getBean(id);
    }
}
```

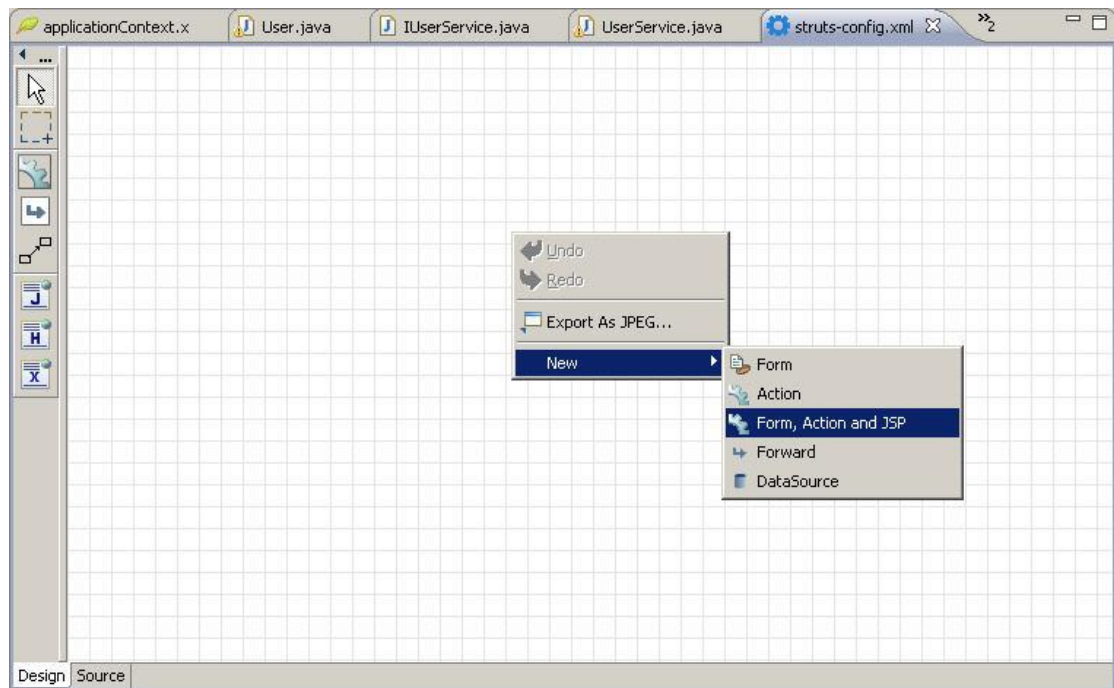
为了让 Web 容器能够初始化 Spring, 我们需要修改 web.xml 文件, 增加以下内容:

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
<servlet>
    <servlet-name>SpringContextServlet</servlet-name>
    <servlet-class>
        org.springframework.web.context.ContextLoaderServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

## 9. 啊，让看看我们工作是否有效

通过上面的配置，我们已经能够让这三个框架来协同作战了，让我们现在来测试一下。

首先创建基于 Struts 的 JSP 页面。打开 WEB-INF 下面的 struts-config.xml，单击右键，选择 New -> Form, Action and JSP。



在弹出的对话框中添加 User case，然后点击 Add 生成 Properties 代码。这会自动生成相应的 Form 代码。输入完成后选择 JSP 选项卡，选上 Create JSP form，修改路径，然后点击 Next。

Config/Module:

Use case:

---

Name:

---

Form Impl: ☒ New FormBean ☐ Existing FormBean ☐ Dynamic FormBean

Superclass:

Form type:

---

Optional Details

Properties:

username - [java.lang.String] <html:text/>

password - [java.lang.String] <html:password/>

Config/Module:

Use case:

---

Name:

---

Form Impl: ☒ New FormBean ☐ Existing FormBean ☐ Dynamic FormBean

Superclass:

Form type:

---

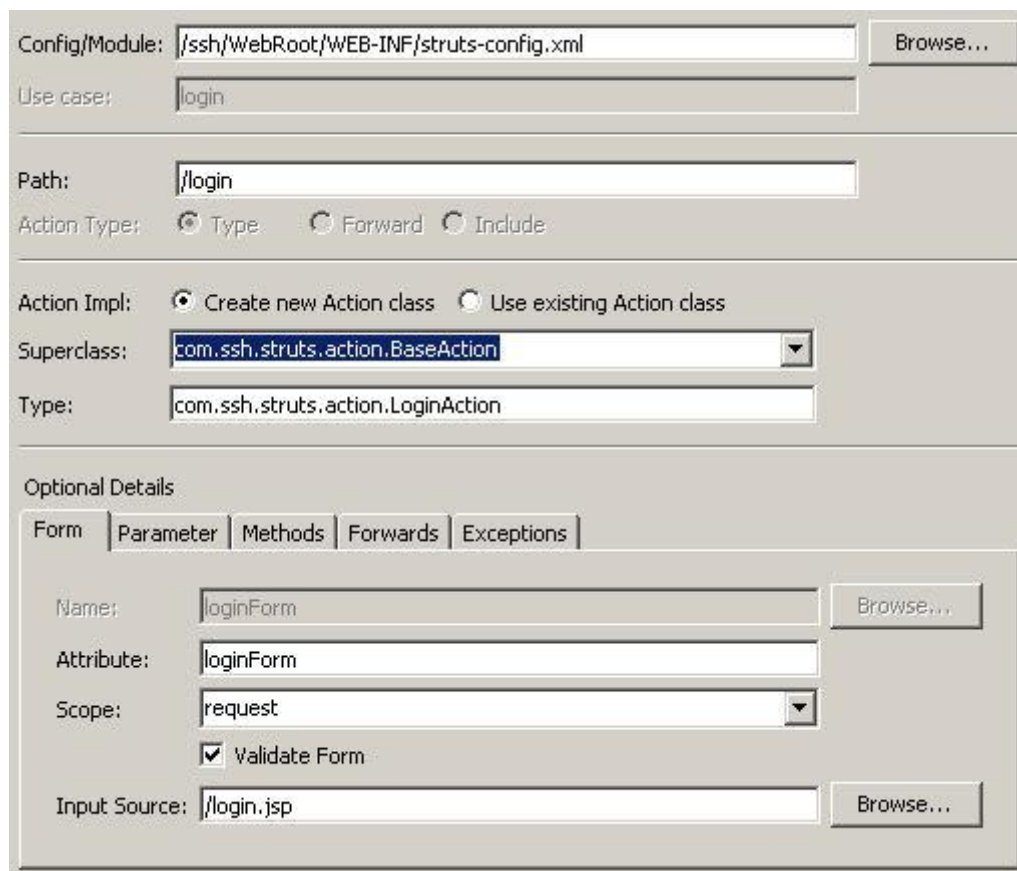
Optional Details

☒ Create JSP form?

New JSP Path:



在出现的对话框中修改 Path，将 Superclass 更改为前面定义的 BaseAction，然后将 Input Source 修改为自己需要的路径。完成后 Finish 即可。



The image shows a configuration dialog box for a Struts action. The 'Config/Module' field is set to '/ssh/WebRoot/WEB-INF/struts-config.xml'. The 'Use case' field is set to 'login'. The 'Path' field is set to '/login'. The 'Action Type' is set to 'Type'. The 'Action Impl' is set to 'Create new Action class'. The 'Superclass' is set to 'com.ssh.struts.action.BaseAction'. The 'Type' is set to 'com.ssh.struts.action.LoginAction'. The 'Optional Details' section is expanded, showing the 'Form' tab. The 'Name' is 'loginForm', the 'Attribute' is 'loginForm', the 'Scope' is 'request', and the 'Validate Form' checkbox is checked. The 'Input Source' is set to '/login.jsp'.

Config/Module: /ssh/WebRoot/WEB-INF/struts-config.xml Browse...

Use case: login

Path: /login

Action Type: ☒ Type ☐ Forward ☐ Include

Action Impl: ☒ Create new Action class ☐ Use existing Action class

Superclass: com.ssh.struts.action.BaseAction

Type: com.ssh.struts.action.LoginAction

Optional Details

Form Parameter Methods Forwards Exceptions

Name: loginForm Browse...

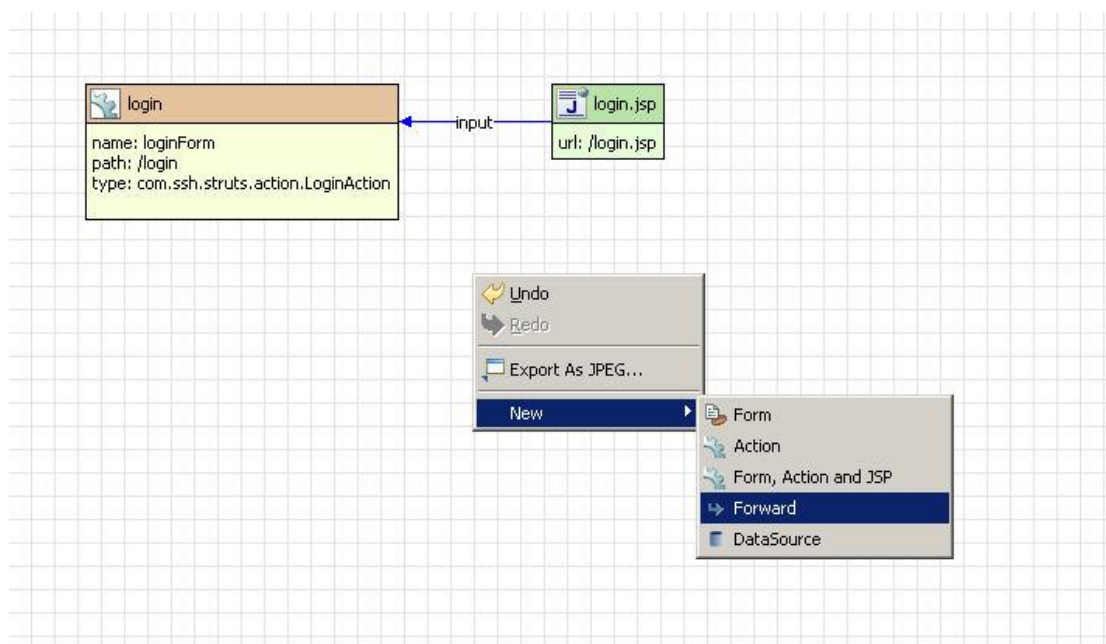
Attribute: loginForm

Scope: request

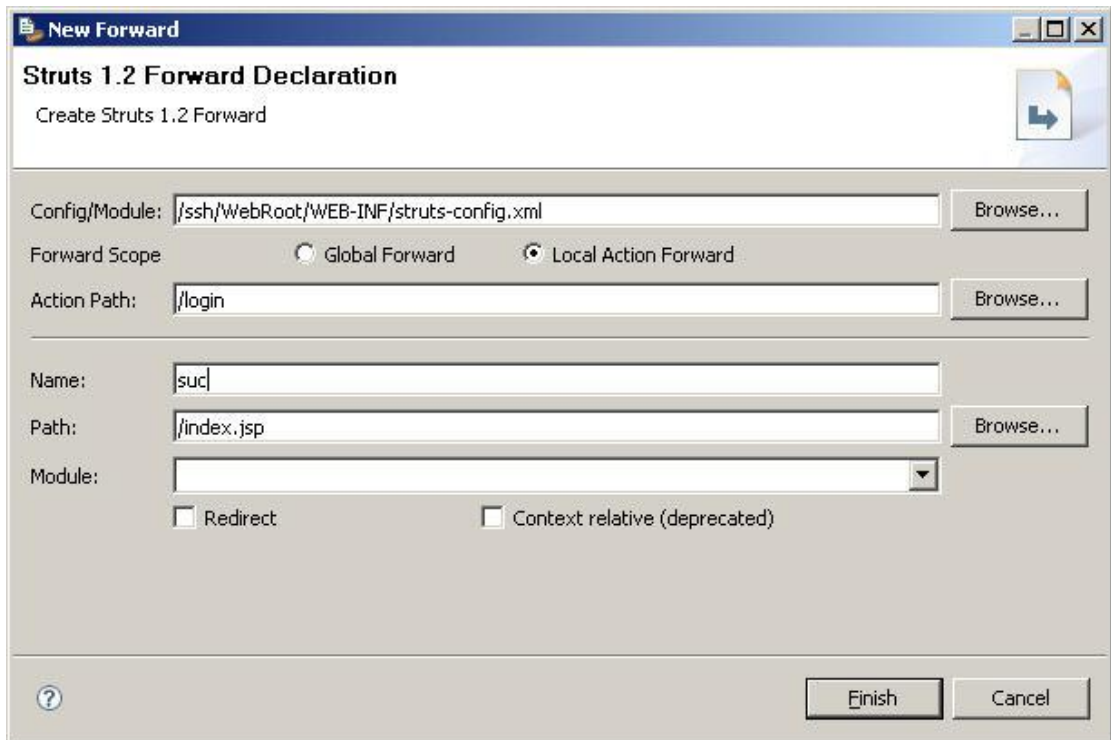
☒ Validate Form

Input Source: /login.jsp Browse...

然后在 struts-config.xml 点击右键，选择 New -> Forward，准备创建一个 ActionForward。



在弹出的对话框中,选择 Local Action Forward,通过 Browser 填写 Action Path,然后填写 Name 和 Path,完成后点击 Finish 即可。



The image shows a 'New Forward' dialog box titled 'Struts 1.2 Forward Declaration'. It contains the following fields and options:

- Config/Module:** /ssh/WebRoot/WEB-INF/struts-config.xml (with a 'Browse...' button)
- Forward Scope:** Radio buttons for 'Global Forward' and 'Local Action Forward' (selected).
- Action Path:** /login (with a 'Browse...' button)
- Name:** suc
- Path:** /index.jsp (with a 'Browse...' button)
- Module:** (empty dropdown menu)
- Options:** Checkboxes for 'Redirect' and 'Context relative (deprecated)' (both unchecked).
- Buttons:** '?', 'Finish', and 'Cancel' at the bottom.

在 LoginAction 中添加如下代码

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    LoginForm loginForm = (LoginForm) form;
    String username = loginForm.getUsername();
    String password = loginForm.getPassword();
    ActionForward forward = mapping.getInputForward();
    IUserService service = (IUserService) getBean("userService");
    User userFromDB = service.getUserByUsername(username);
    if (userFromDB.getPassword().equals(password)) {
        forward = mapping.findForward("suc");
    }
    return forward;
}
```

login.jsp 内容

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>

<html>
  <head>
    <title>JSP for LoginForm form</title>
  </head>
  <body>
    <html:form action="/login">
      password : <html:password property="password"/><html:errors
property="password"/><br/>
      username : <html:text property="username"/><html:errors
property="username"/><br/>
      <html:submit/><html:cancel/>
    </html:form>
  </body>
</html>
```

Ok, 这时我们可以启动 Tomcat 进行查看页面了

## 10. 黎明前的黑暗

上面的工作完成以后理论上应该可用, 但是当你运行的事后却发现会有一个非常诡异的异常出现

## HTTP Status 500 -

**type** Exception report

**message**

**description** The server encountered an internal error () that prevented it from fulfilling this request.

**exception**

```
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'session
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initialize
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBe
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory$1.run (Abst
    java.security.AccessController.doPrivileged (Native Method)
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean
    org.springframework.beans.factory.support.AbstractBeanFactory$1.getObject (AbstractBeanF
    org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton (Def
    org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean (AbstractBeanFac
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean (AbstractBeanFacto
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean (AbstractBeanFacto
    org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSing
    org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitial
    org.springframework.context.support.AbstractApplicationContext.refresh (AbstractApplicat
    org.springframework.web.context.ContextLoader.createWebApplicationContext (ContextLoader
    org.springframework.web.context.ContextLoader.initWebApplicationContext (ContextLoader.j
    org.springframework.web.context.ContextLoaderServlet.init (ContextLoaderServlet.java:81)
    javax.servlet.GenericServlet.init (GenericServlet.java:212)
    org.apache.catalina.startup.HostConfig.checkResources (HostConfig.java:1116)
    org.apache.catalina.startup.HostConfig.check (HostConfig.java:1214)
    org.apache.catalina.startup.HostConfig.lifecycleEvent (HostConfig.java:293)
    org.apache.catalina.util.LifecycleSupport.fireLifecycleEvent (LifecycleSupport.java:120)
    java.lang.Thread.run (Thread.java:619)
```

**note** The full stack trace of the root cause is available in the Apache Tomcat/5.5.26 logs.

这个问题本来面目是这样，由于 MyEclipse 给我们做的事情太全面了，全面到 spring 和 hibernate 的包都是重复引用的。

其解决办法就是干掉多余的包，在 Tomcat 5.5\webapps\ssh\WEB-INF\lib 下的 asm-2.2.3.jar 就可以了，在重启 Tomcat 我们看看我们干了这么长时间的成果吧，



username : admin

password : ●●●●●

Submit Cancel

真令人失望又失败了，呵呵有如下异常：

**type** Exception report

**message**

**description** The server encountered an internal error () that prevented it from fulfilling this request.

**exception**

```
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'userService' defined
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:553)
    org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:475)
    org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:261)
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:185)
    org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:164)
    org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:545)
    org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:838)
    org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:162)
    org.springframework.web.context.ContextLoader.createWebApplicationContext(ContextLoader.java:255)
    org.springframework.web.context.ContextLoader.initWebApplicationContext(ContextLoader.java:199)
    org.springframework.web.context.ContextLoaderServlet.init(ContextLoaderServlet.java:81)
    javax.servlet.GenericServlet.init(GenericServlet.java:212)
    org.apache.catalina.startup.HostConfig.deployDirectory(HostConfig.java:926)
    org.apache.catalina.startup.HostConfig.deployDirectories(HostConfig.java:889)
    org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:492)
    org.apache.catalina.startup.HostConfig.start(HostConfig.java:1149)
    org.apache.catalina.startup.HostConfig.lifecycleEvent(HostConfig.java:311)
    org.apache.catalina.util.LifecycleSupport.fireLifecycleEvent(LifecycleSupport.java:120)
    org.apache.catalina.startup.Catalina.start(Catalina.java:552)
    sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

哎，谁叫俺的脑子不好使能，忘了加入 **bean** 的初始化，在 `applicationContext.xml` 中加入如下内容就可以了

```
<bean name="userService" class="com.ssh.service.impl.UserService">
    <property name="userDAO">
        <ref bean="UserDAO" />
    </property>
</bean>
```

历经千难万阻，我们终于看到了一个想看的巨破巨简陋的界面，呵呵，就到这里吧，估计大家要疯了。

This is my JSP page.