Eclipse Sample Project

——从零开始创建基于 struts1.2 + Hibernate3.0 的 Web 工程

目　　录

1. 环境说明

安装 Eclipse 和安装 MyEclipse 组件的过程在这里不进行介绍。可以参考其它资料完成 Eclipse 环境的安装与资源的配置。

接下来的篇幅将给出一个在 MyEclipse 开发环境下的一个 Web 工程实例。说明在 MyEclipse 的集成开发环境下，如何利用 Struts 进行 Web 工程开发。

例子工程的环境：

Windows 2000 Professional 或者更高的版本。

J2SDK 5.0

Apache Tomcat 5.0

MySQL 4.0.14b

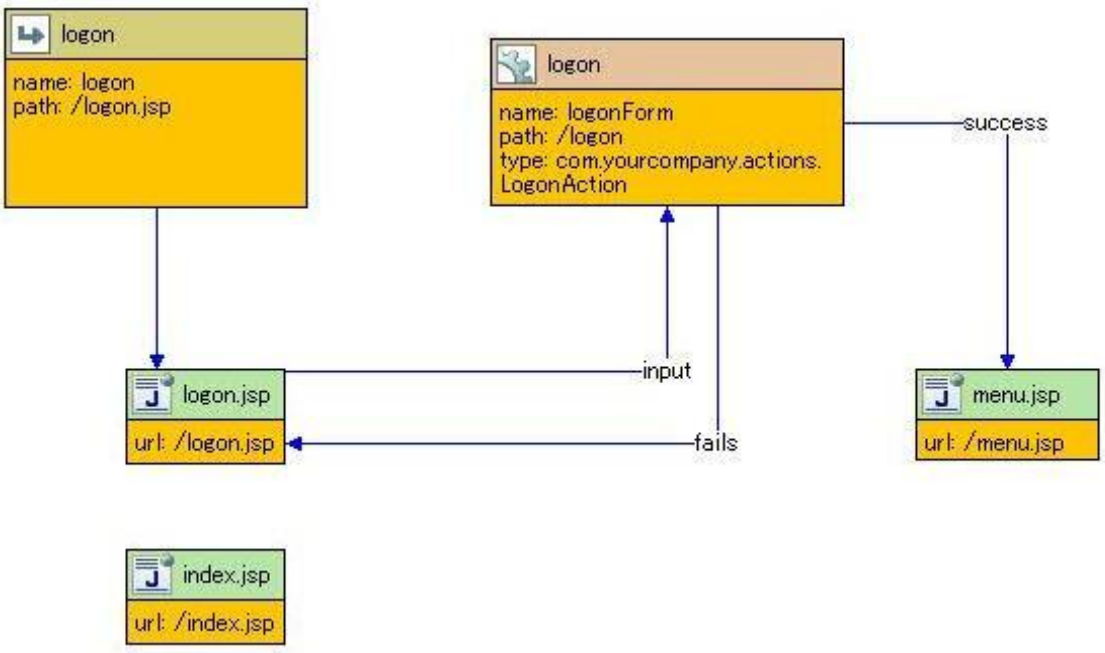
Eclipse-SDK-3.1

MyEclipse 4.0

SQLyog407　　（可选）

2. 例子工程说明

本例子是利用 struts 架构，实现用户 logon 的 Web 工程。在这个工程中，用户可以访问三个页面，index.jsp、logon.jsp、menu.jsp。他们的关系如下图所示：



其中，左上角的方框代表 struts 结构中/WEB-INF/struts-config.xml 中定义的，<global-forwards>，也就是在整个 Web 工程中，只要有 logon 的动作发生，都会使画面转移到

logon.jsp。那么处理 logon.jsp 画面的 formbeen 是上图上放右侧的模块表示的 formbean 来完成——logonForm。

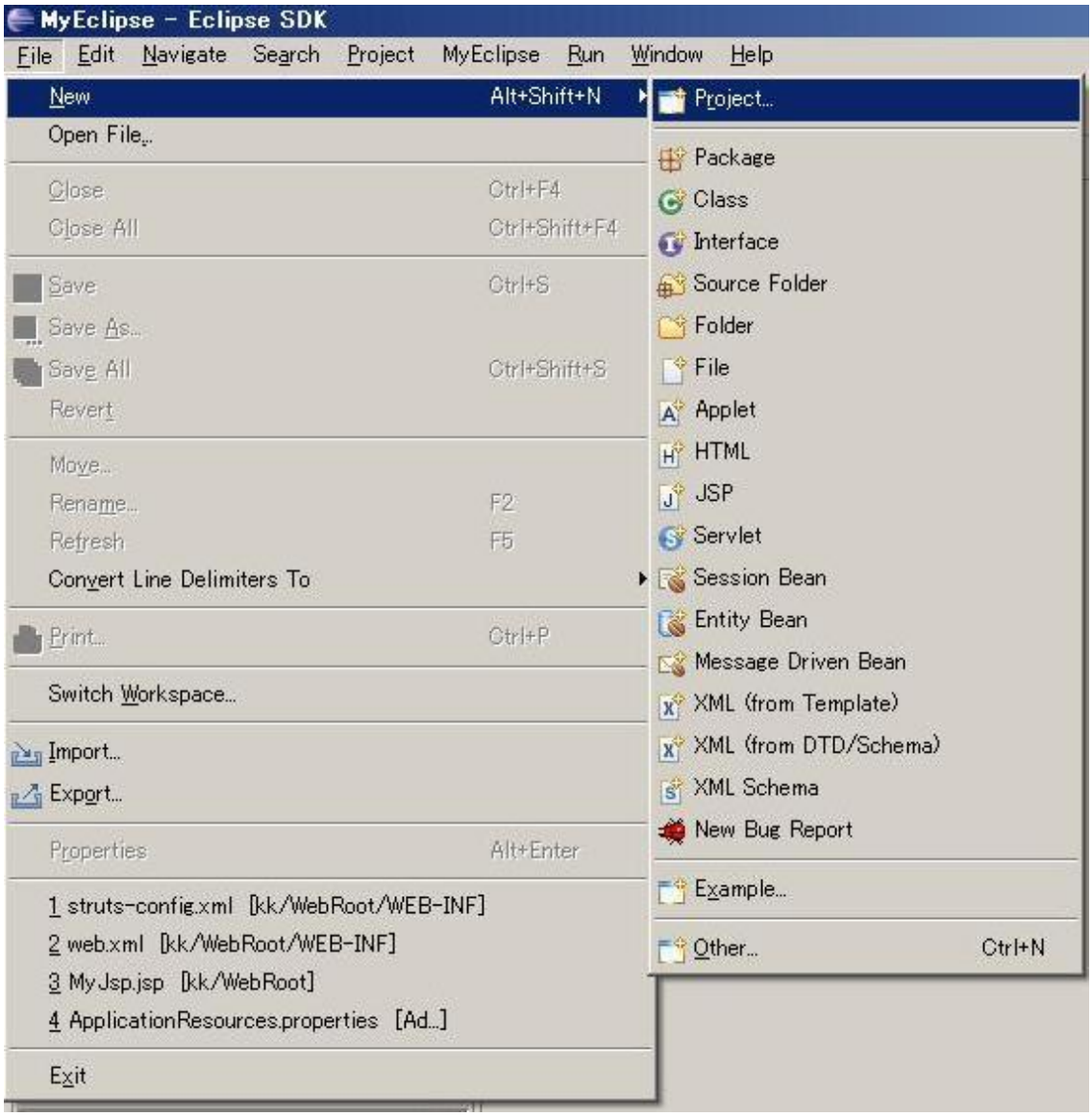
LogonFrom 处理后会有两个结果，“success”的时候，画面会转移到 menu.jsp，失败的话，回到 logon 画面。

在 logon 画面，用户需要输入用户名和密码，点击 submit，进行登录认证。输入内容的检测是在 logonForm 的 bean 中检查。判断操作在 logonAction 的 bean 中完成，在 logonAction 的 bean 中，需要访问 MySql 数据库的 Account 表格，进行用户和密码的查询。查询成功的话，证明用户认证通过。

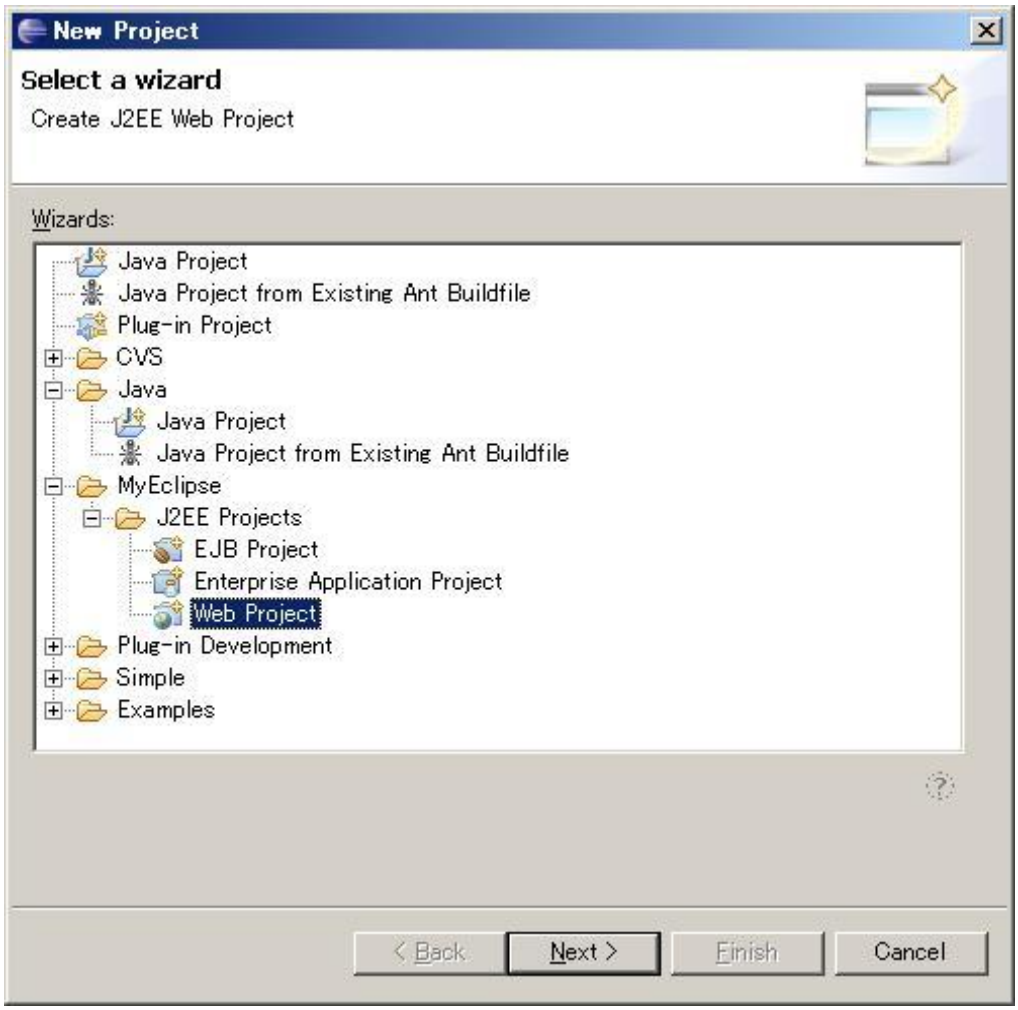
3. 建立工程

3.1. 创建新的 Web Project

启动 MyEclipse，通过 Wizard，建立一个 Web 工程。如下图：



点击project后,MyEclipse弹出工程的对话框,我们在所罗列的工程中选中MyEclipse下面的J2EE Projects的Web Project。如下图所示：

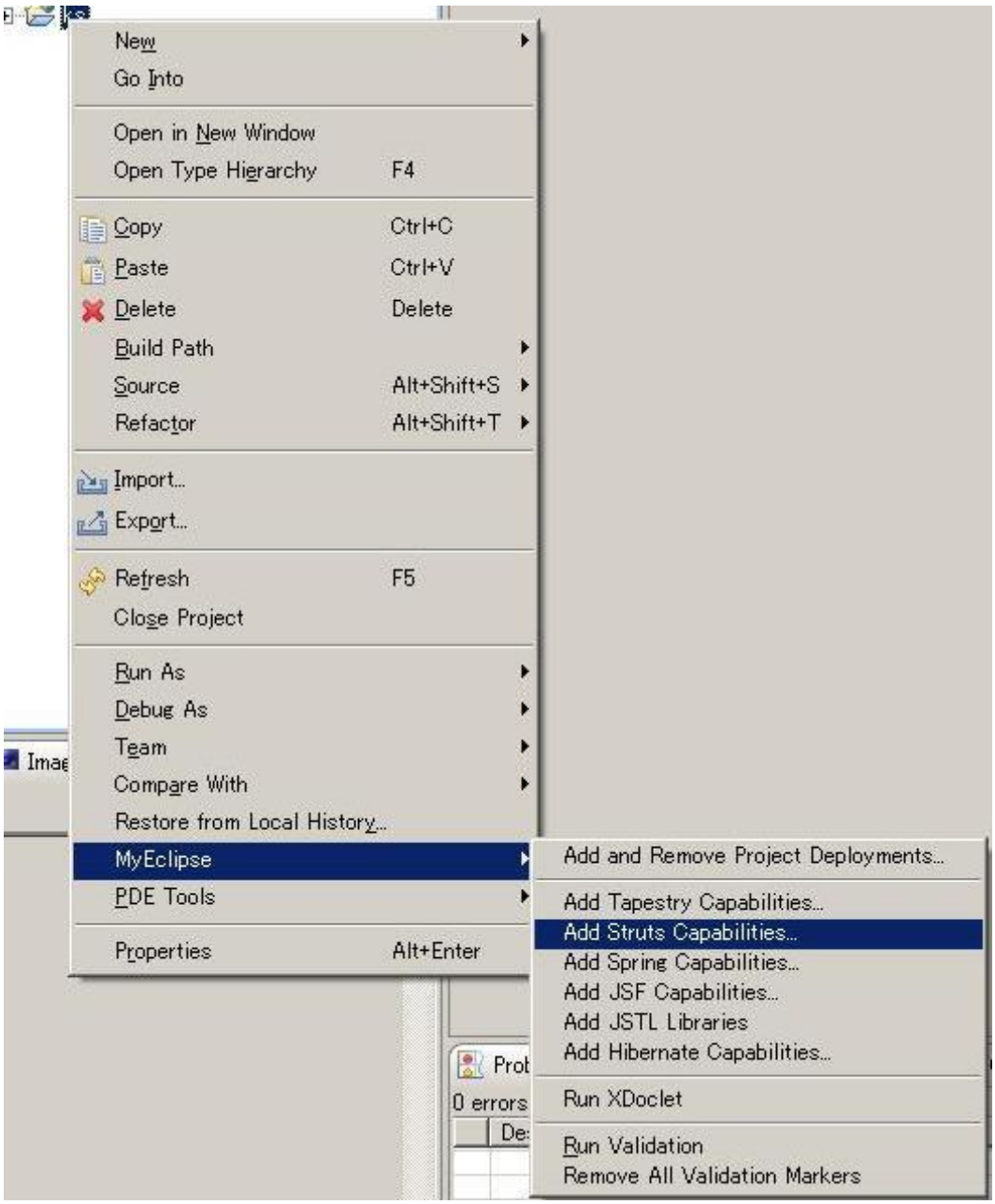


在Wizard的下一页，必要的输入信息是Project Name、Context root URL。最后点击finish，一个空的WebProject就已经建立起来了

3.2. 加载Struts

在新创建的Web Project中加载struts。

在MyEclipse集成开发环境下，已经集成了struts开发的环境，用户可以根据自己的实际需要，加载struts的环境。加载过程如下图：



右键点击刚刚创建好的工程，点击 MyEclipse 属性，在扩展的右键菜单里面，选择 Add Struts Capabilities。

MyEclipse 集成环境就会自动将 Struts 所需要的目录结构进行创建，并将需要的环境资源自动引用到相应的目录中去。

我们还要对自动生成的 Struts 进行修正，首先，修改 WEB-INF/web.xml 将<web-app>标签中的属性删除。<web-app XXXXXXXXXXXXXXX> -> <web-app>

其次要添加本工程的标签定义库，添加<taglib></taglib>

```
<taglib>
  <taglib-uri>/WEB-INF/app.tld</taglib-uri>
  <taglib-location>/WEB-INF/app.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
  <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
  <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
  <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
</taglib>
```

4. 完善工程

4.1. 添加 index 画面

在加载完 Struts 结构之后，就可以在这个工程中添加内容了。我们可以添加一个画面 JSP。添加 JSP 很简单，点击工程的右键，添加 JSP。Wizard 会弹出一个添加的对话框。如下图：

在 File Name 栏中改变文件名称，改成 index.jsp 把这个 jsp 作为 Web 工程的第一个画面。
Template To Use 栏可以选择第五项（选择选择其他项也可以）。
对于新生成的 jsp 文件进行必要的修改。
Index.jsp 文件：

```
<%@ page language="java" pageEncoding="UTF-8"%>

<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'index.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>
  <h3><bean:message key="index.heading"/></h3>
        <html:link page="/logon.jsp"><bean:message key="index.logon"/></html:link>
</body>
</html>
```

其中蓝色和橙色部分是修改的内容，蓝色部分是今后自动添加 jsp 文件必须修改的地方，让这个页面用到的 tag 都能在这个工程中定义的 tag 集合中找到。
橙色部分是画面的主要显示内容。<h3></h3>的内容是一段文本，内容被 struts 结构的静态文本集合中的 index.heading 给替换掉了。这个文本的内容可以在工程的 src\com\yourcompany\struts\ApplicationResources.properties 文件中找到。Struts 结构推荐用户将页面上的静态文本用

ApplicationResources 的形式替换。这样可以在大量的维护页面文字时候，感到便捷很多，同时，也大大减轻了多国语言版本网页的维护。

<html:link page="/logon.jsp">相当于 html 语言中的<a herf=xxxxx>。这个 tag 是在 /WEB-INF/struts-html.tld 中可以找到并在显示页面的时候，被转义成<a herf=xxxxx>

4.2. 添加 logon 画面

添加方法跟添加 index 画面相同。修改内容稍微有些差别。

Logon.jsp 的内容：

```
<%@ page language="java" pageEncoding="UTF-8"%>

<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html locale="true">
  <head>
    <html:base />

    <title>logon.jsp</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">
  </head>

  <body>
    <html:form action="/logon.do" method="post"  focus="userName">
      <table border="0">
        <tr>
          <td><bean:message key="prompt.username"/></td>
          <td><html:text property="userName" /></td>
        </tr>
        <tr>
```

```
        <td>Password:</td>
        <td><html:password property="password" /></td>
      </tr>
      <tr>
        <td colspan="2" align="center"><html:submit /></td>
      </tr>
    </table>
  </html:form>
 </body>
</html:html>
```

其中，蓝色部分是引用本工程的 tag 标示库，红色的部分是表单属性名称的修改，和指定 action 动作的名称。

4.3. 修改 WEB-INF/struts-config.xml

画面上出现了 form，那么根据 struts 的结构要求，就必须在 WEB-INF/struts-config.xml 中明确这个 form 的 formbean（表单内容校验的 java class）是什么。执行这个 form 的 action（表单执行的内部逻辑）是什么，以及 action 的结果会产生怎样的画面迁移。这些都是在 WEB-INF/struts-config.xml 中定义的。也就是标准的 MVC 架构所要求的。

```
<struts-config>
  <form-beans >
    <form-bean name="logonForm" type="com.yourcompany.forms.LogonForm" />
  </form-beans>

  <global-forwards >
    <forward name="logon" path="/logon.jsp" />
  </global-forwards>

  <action-mappings >
    <action
      path="/logon"
      type="com.yourcompany.actions.LogonAction"
      name="logonForm"
      scope="request"
      input="/logon.jsp">
```

```
      <forward name="success" path="/menu.jsp" />
      <forward name="fails" path="/logon.jsp" />
    </action>
  </action-mappings>

  <message-resources parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

其中<action>部分是说明 action 的属性。
Path 指定 Action 处理的 URL
Type 指定 Action 的类名
Name 指定 Action 主力的 ActionForm 名，与<form-beans >元素的 name 属性匹配。
Scope 指定 ActionForm 存在的范围
Input 指定包含客户提交表单的网页，如果 ActionForm 的 Validate 方法返回错误，则因该把用户请求转发到这个网页。
Validate 如果取值为 true，则表示 ActionServlet 应该调用 ActionForm 的 validate 方法
Forward 就是 Action 的 execute 方法执行完毕后，把客户请求在转发给相应的页面。

4.4. 添加 formbean 和 actionbean
添加方法跟 JSP 相同，但是在选择 superclass 的时候，formbean 要选择 ActionForm 作为类的父类。Actionbean 的父类是 Action
下面是各自的代码

LogonForm.java 文件内容:

```
package com.yourcompany.forms;
import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class LogonForm extends ActionForm {

        private static final long serialVersionUID = 7322786881443789688L;
        //     ----------------------------------------------------     Instance
Variables
```

```
    private String username = null;
    private String password = null;

        // ---------------------------------------------------------- Methods
    public String getUserName() {
                return (this.username);
    }

    public void setUserName(String username) {
        this.username = username;
    }

    public String getPassword() {
                return (this.password);
    }

    public void setPassword(String password) {
        this.password = password;
    }

        public void reset(ActionMapping mapping, HttpServletRequest request) {
                this.password = null;
                this.username = null;
        }

}
```

LogonAction.java 文件：

```
package com.yourcompany.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
```

```
import org.apache.struts.action.ActionMapping;

import com.yourcompany.forms.LogonForm;

public class LogonAction extends Action {

        public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest request, HttpServletResponse response) {

                String userName = null;
                String password = null;

                if (form != null) {
                        userName = ((LogonForm) form).getUserName();
                        password = ((LogonForm) form).getPassword();
                }
                if(userName.equals(“test1”) && password. equals(“test1”)){
                        return (mapping.findForward("success"));
                }
                else{
                        return (mapping.findForward("fails"));
                }
        }

}
```

4.5. 添加 menu 画面

与添加 index 画面的方法相同。下面是 menu.jsp 文件的内容

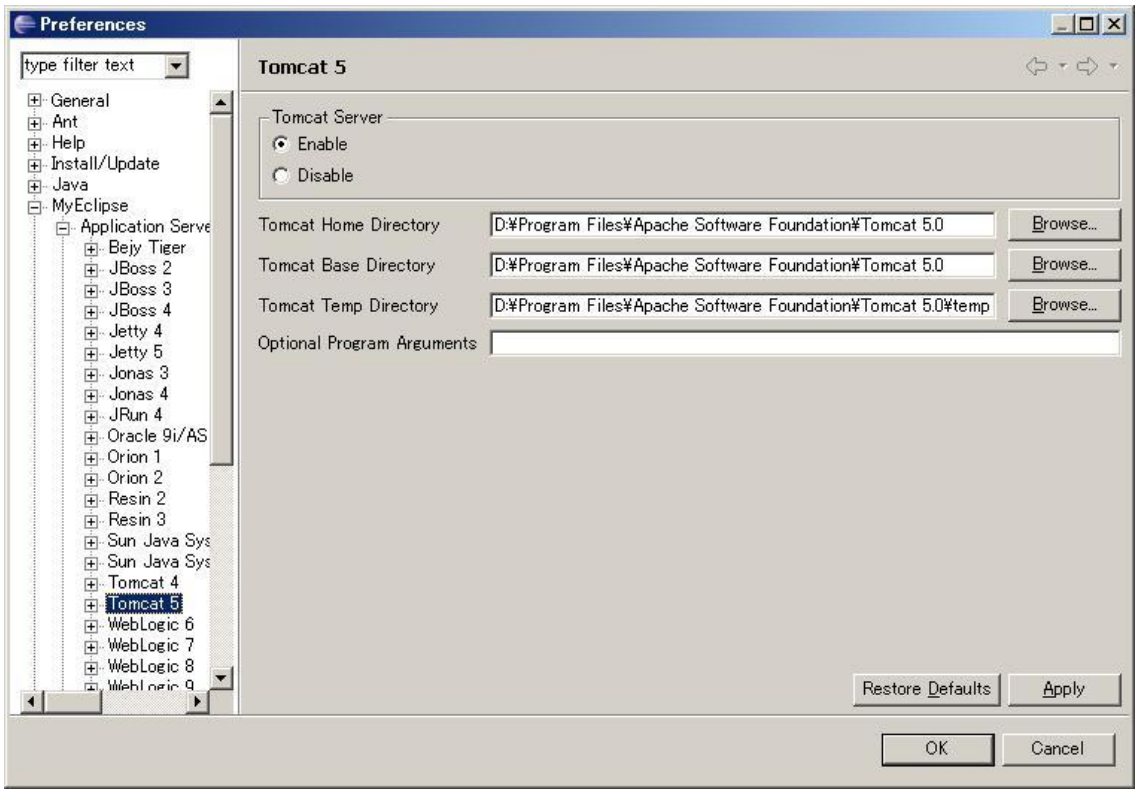
```
<%@ page language="java" pageEncoding="UTF-8"%>

<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html locale="true">
```

```
  <head>
    <html:base />
        <title>menu.jsp</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">
  </head>
    <body>
          <h3><bean:message key="menu.message"/></h3>
  </body>
</html:html>
```

5. 调试工程

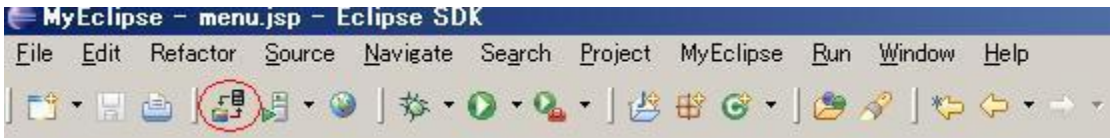
如果本地机器已经安装了 Tomcat5，那么可以在 MyEclilpse 的环境下调试工程了。

指定 Tomcat5 的 web application service 如下图：



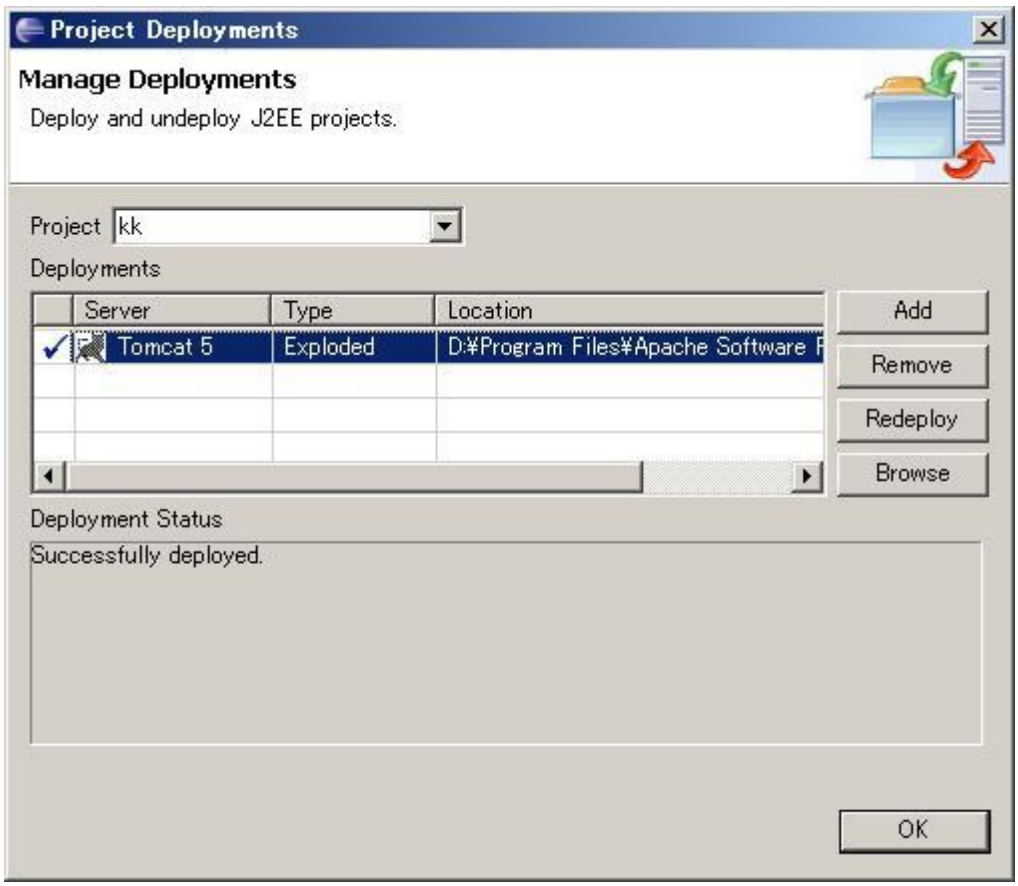
将 Enable 选项打开，并且指定 Tomcat 的安装目录。

配置目前的工程到 Tomcat 中去。

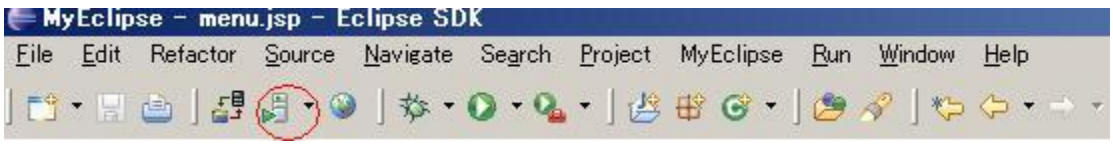


点选工具栏上的被红线圈出的按钮

在弹出来的对话框中选择，我们创建的工程，和添加 Tomcat5 的 web application service



然后就可以按下工具栏上的启动服务按钮，调试这个 Web 工程了。



每一次改动工程中的文件后，要想看看改动后的效果，那么就在配置工程中，重新发布这个工程到 tomecat5 去。

6. Tomcat 下如何配置 mysql 的数据库连接池

6.1. 配置 server.xml

配置 Tomcat 的 server.xml 文件,路径:【TOMCAT_HOME】\common\lib 下的 server.xml 文件在</host>之前加入以下内容以添加 JNDI 数据源:

```
<Context path="/DBTest" docBase="DBTest"
        debug="5" reloadable="true" crossContext="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="localhost_DBTest_log." suffix=".txt"
            timestamp="true"/>
  <Resource name="jdbc/TestDB"
              auth="Container"
              type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/TestDB">
    <parameter>
      <name>factory</name>
    <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
    <!-- Maximum number of dB connections in pool. Make sure you
         configure your mysqld max_connections large enough to handle
         all of your db connections. Set to 0 for no limit.
         -->
    <parameter>
      <name>maxActive</name>
      <value>100</value>
    </parameter>
    <!-- Maximum number of idle dB connections to retain in pool.
         Set to 0 for no limit.
         -->
    <parameter>
      <name>maxIdle</name>
      <value>30</value>
    </parameter>
    <!-- Maximum time to wait for a dB connection to become available
         in ms, in this example 10 seconds. An Exception is thrown if
         this timeout is exceeded.  Set to -1 to wait indefinitely.
         -->
```

```
    <parameter>
      <name>maxWait</name>
      <value>10000</value>
    </parameter>
    <!-- MySQL dB username and password for dB connections  -->
    <parameter>
     <name>username</name>
     <value>root</value>
    </parameter>
    <parameter>
     <name>password</name>
     <value> </value>
    </parameter>
    <!-- Class name for mm.mysql JDBC driver -->
    <parameter>
       <name>driverClassName</name>
       <value>org.gjt.mm.mysql.Driver</value>
    </parameter>
    <!-- The JDBC connection url for connecting to your MySQL dB.
         The autoReconnect=true argument to the url makes sure that the
         mm.mysql JDBC Driver will automatically reconnect if mysqld closed the
         connection.  mysqld by default closes idle connections after 8 hours.
         -->
    <parameter>
      <name>url</name> <value>jdbc:mysql://192.168.0.208:3306/db_test_account?autoRec
onnect=true</value>
    </parameter>
  </ResourceParams>
</Context>
```

注意：

*其中蓝色字体表示你这个应用的路径和别名，也就是你访问自己配置的这个 web 站点的名字，注意区分大小写，必须一致，否则系统无法正常运行（例：http://localhost:8080/DBTest）

*其中红色字体表示数据源的名字，注意将会被 web.xml 和你访问数据库的程序调用

6.2. 配置 web.xml

配置 Web 用程序的 web.xml 文件

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
    <!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <description>MySQL Test App</description>
  <resource-ref>
      <description>DB Connection</description>
      <res-ref-name>jdbc/TestDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

6.3. 访问数据库的程序片段

```
package swt.db.DBUtility;

import javax.naming.*;
import javax.sql.*;
import java.sql.*;

public class DBTest {

  String foo = "Not Connected";
  int bar = -1;

  public void init() {
    try{
      Context ctx = new InitialContext();
      if(ctx == null )
          throw new Exception("Boom - No Context");
      DataSource ds =
            (DataSource)ctx.lookup(
               "java:comp/env/jdbc/TestDB");
      if (ds != null) {
        Connection conn = ds.getConnection();
        if(conn != null)  {
            foo = "Got Connection "+conn.toString();
```

```
            Statement stmt = conn.createStatement();
            ResultSet rst =
                stmt.executeQuery(
                  " select UserName from t_account ");
            if(rst.next()) {
               foo=rst.getString(1);
               bar=208;
            }
            conn.close();
          }
        }
    }catch(Exception e) {
      e.printStackTrace();
    }
 }
 public String getFoo() { return foo; }
 public int getBar() { return bar;}
}
```

6.4. Jsp 页面(index.jsp)

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ page import ="swt.db.DBUtility.*" %>
<html>
  <head>
    <title>DB Test</title>
  </head>
  <body>
  <%
    DBTest tst = new DBTest();
    tst.init();
  %>
  <h2>Results</h2>
    Foo <%= tst.getFoo() %><br>
    Bar <%= tst.getBar() %>
  </body>
</html>
```

启动 Tomcat 在浏览器上输入 http://localhost:8080/DBTest

备注：

连接池配置(Database Connection Pool (DBCP) Configurations)

DBCP 使用的是 Jakarta-Commons Database Connection Pool 要使用连接池需要如下的组件即 jar 文件

Jakarta-Commons DBCP 1.1 对应 commons-dbcp-1.1.jar。

Jakarta-Commons Collections 2.0 对应 commons-collections.jar。

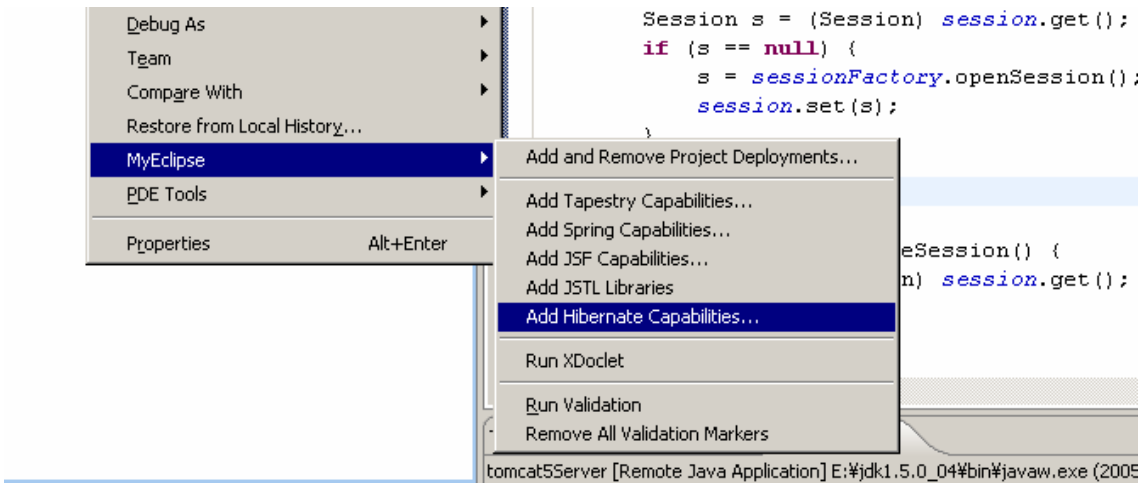
Jakarta-Commons Pool 1.1 对应 commons-pool-1.1.jar。

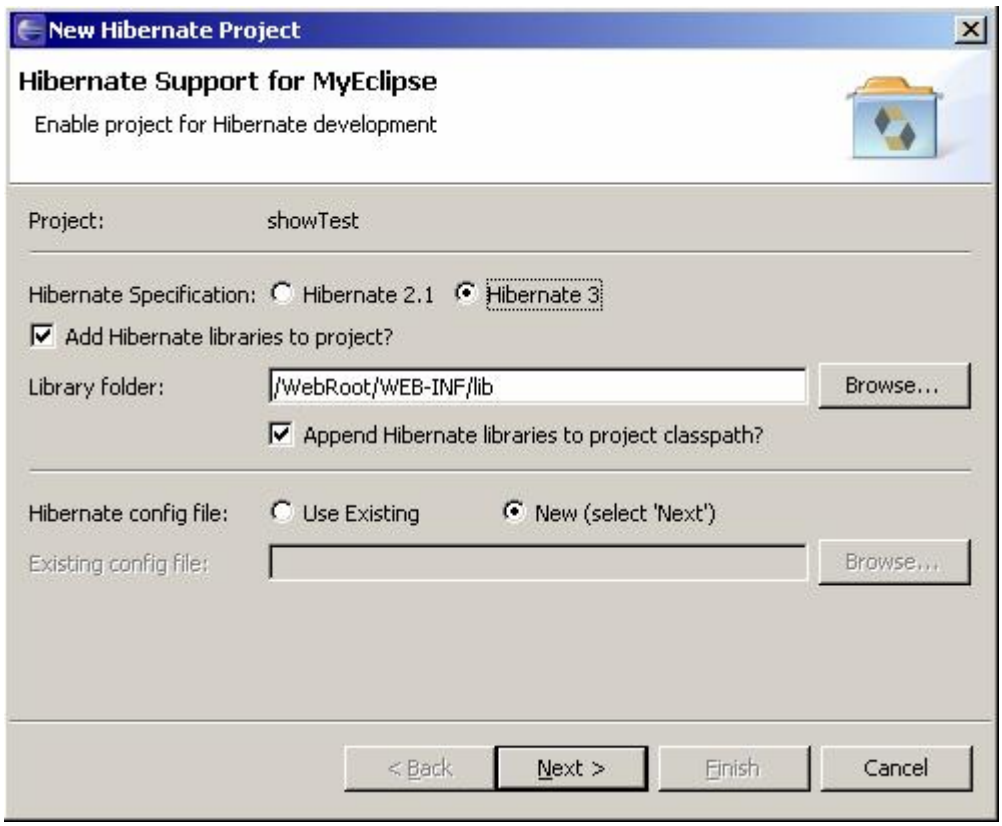
这三个 jar 文件要与你的 JDBC 驱动程序一起放到【TOMCAT_HOME】\common\lib 目录下以便让 tomcat 和你的 web 应用都能够找到。

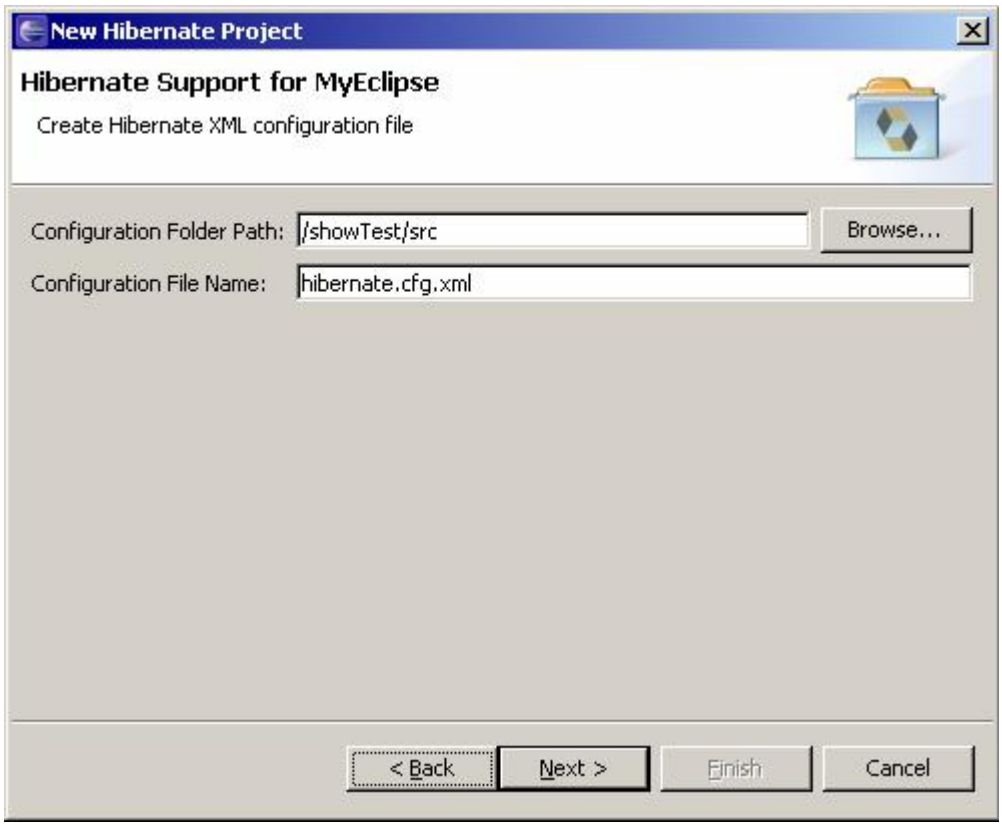
7. Tomcat5.0 下配置 Hibernate3.0 应用

7.1. 在 Tomcat 下建立数据库连接池，如 6 中所示

7.2. 在 Struts 应用中添加 Hibernate3.0 支持

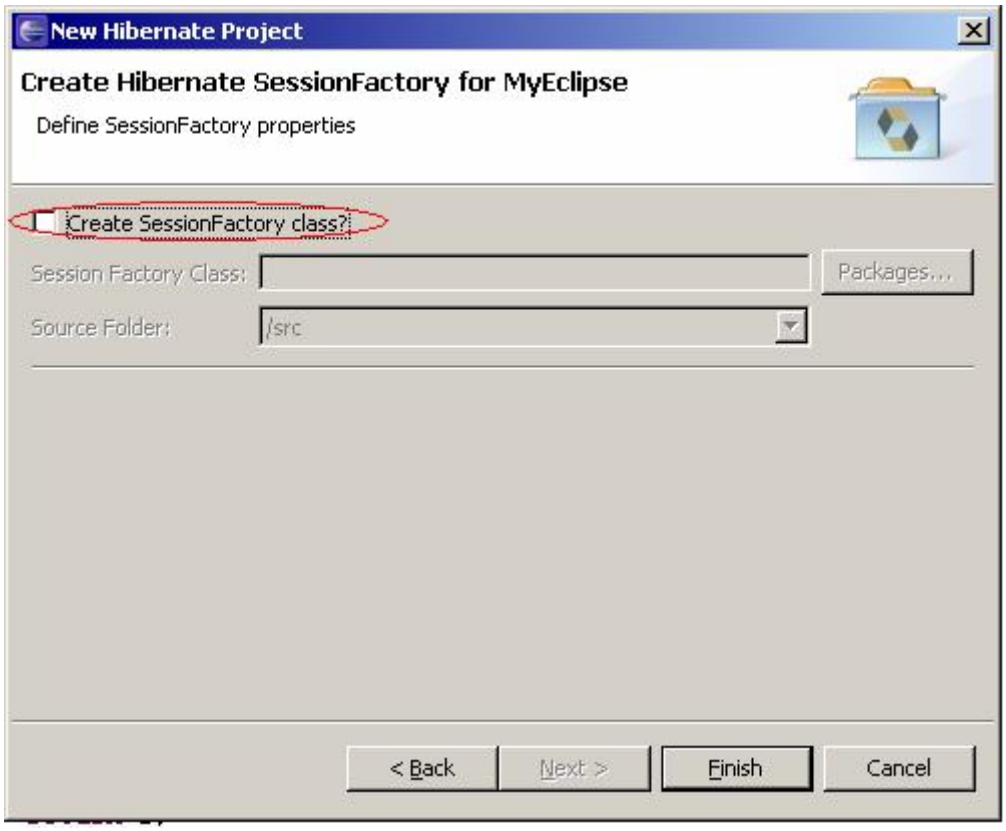




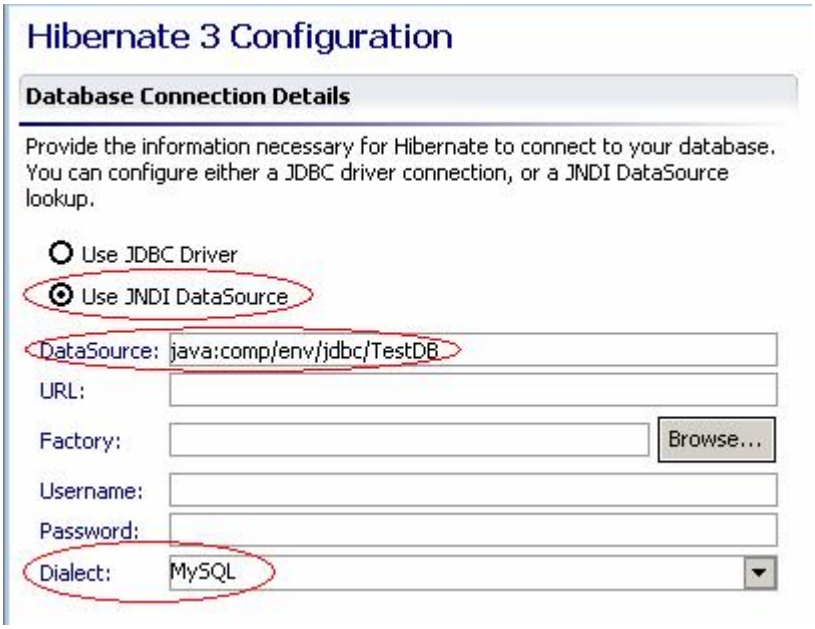


注意：hibernate.cfg.xml 文件一定要存放到跟目录下，默认的也就是/web 应用/src,这个部

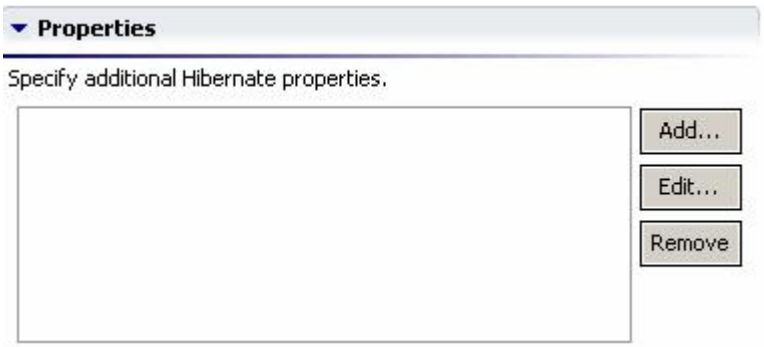
署这个应用的时候 hibernate.cfg.xml 才会出现在 classes 目录下，也就是 hibernate 存放配置文件的默认录入下.



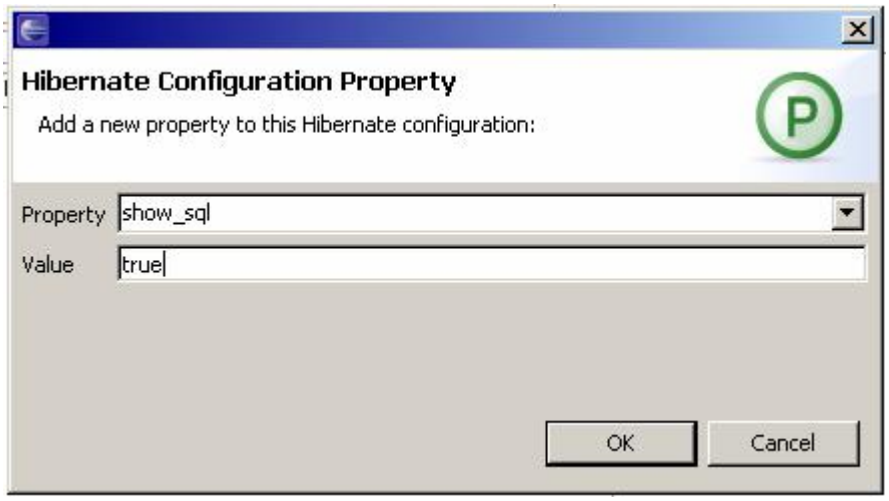
由于我们采用 Tomcat 提供的数据库连接池，所以这里我们将是用自己创建的 SessionFactory 类。点击完成 MyEclipse 会自动把 Hibernate 所需的的类库加入到当前应用中。接下来就是配置 Hibernate 连接数据库的所需的参数，以及性能参数（可选）。



既然我们选用应用服务器所提供的数据库连接池，那么在这里我们只须要指定数据源的名字：java:comp/env/jdbc/TestDB，其中 jdbc/TestDB 就是我们在 Tomcat 中配置的数据源，也就是我们上面提到的 jdbc/TestDB，资源名称一定要匹配。其他的参数由于已经在 Tomcat 中配置过了，所以在这里就不用配置了，Dialect 一定要指定跟我们数据库匹配的语言。



我们可以在这里配置 Hibernate 一些调整性能的参数(针对不同的数据库有些属性可能无效)。



在这里我们设置 show_sql 为 true，这样在开发调试过程成控制台可以打印真正在数据库端执行的 sql 语句便于查找问题。其他一些属性可以参阅 http://www.hibernate.org。

到这里 Hibernate 的配置已经基本完成，下面创建 SessionFactory 用来和数据库进行交互（Hibernate 官方文档提供）。

```
package com.db;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static Log log = LogFactory.getLog(HibernateUtil.class);

    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            log.error("Initial SessionFactory creation failed.", ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static final ThreadLocal session = new ThreadLocal();

    public static Session currentSession()  {
        Session s = (Session) session.get();
```

```
            if (s == null) {
                s = sessionFactory.openSession();
                session.set(s);
            }
            return s;
        }

        public static void closeSession() {
            Session s = (Session) session.get();
            if (s != null)
                s.close();
            session.set(null);
        }
}
```

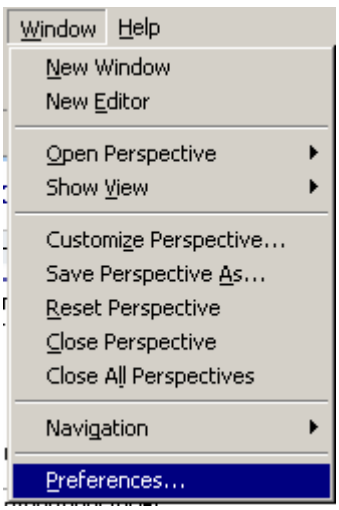
后面将会提到如何使用 HibernateUtil,下面建立我们所需的数据库
（略）我们可以使用 208 上的 db_test_account 这个数据库中的表 message 来测试我们的 Hibernate 配置是否成功。数据库样例：

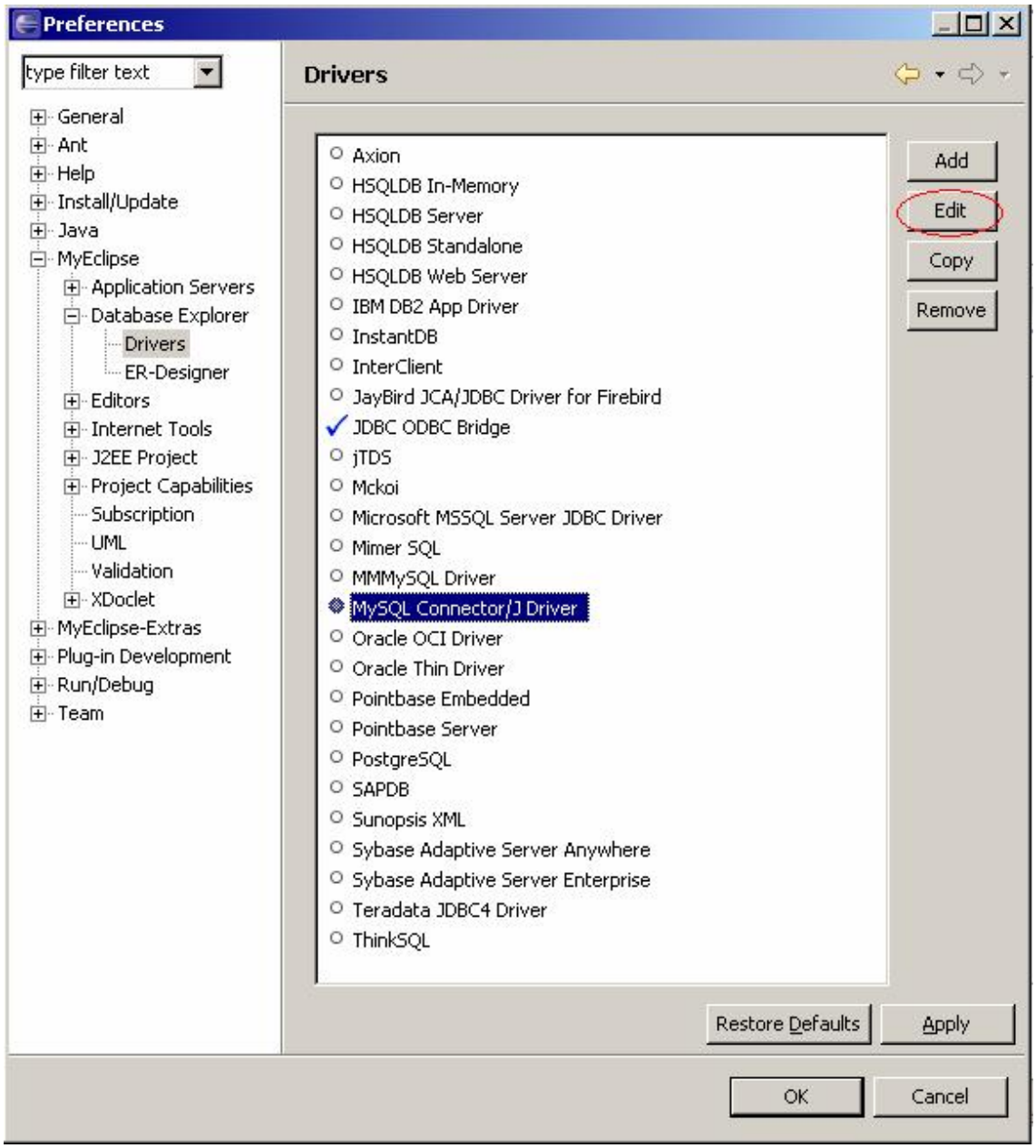
	Id	text	nextMessage
☐	3	a	(NULL)
☐	4	b	3

建表语句：

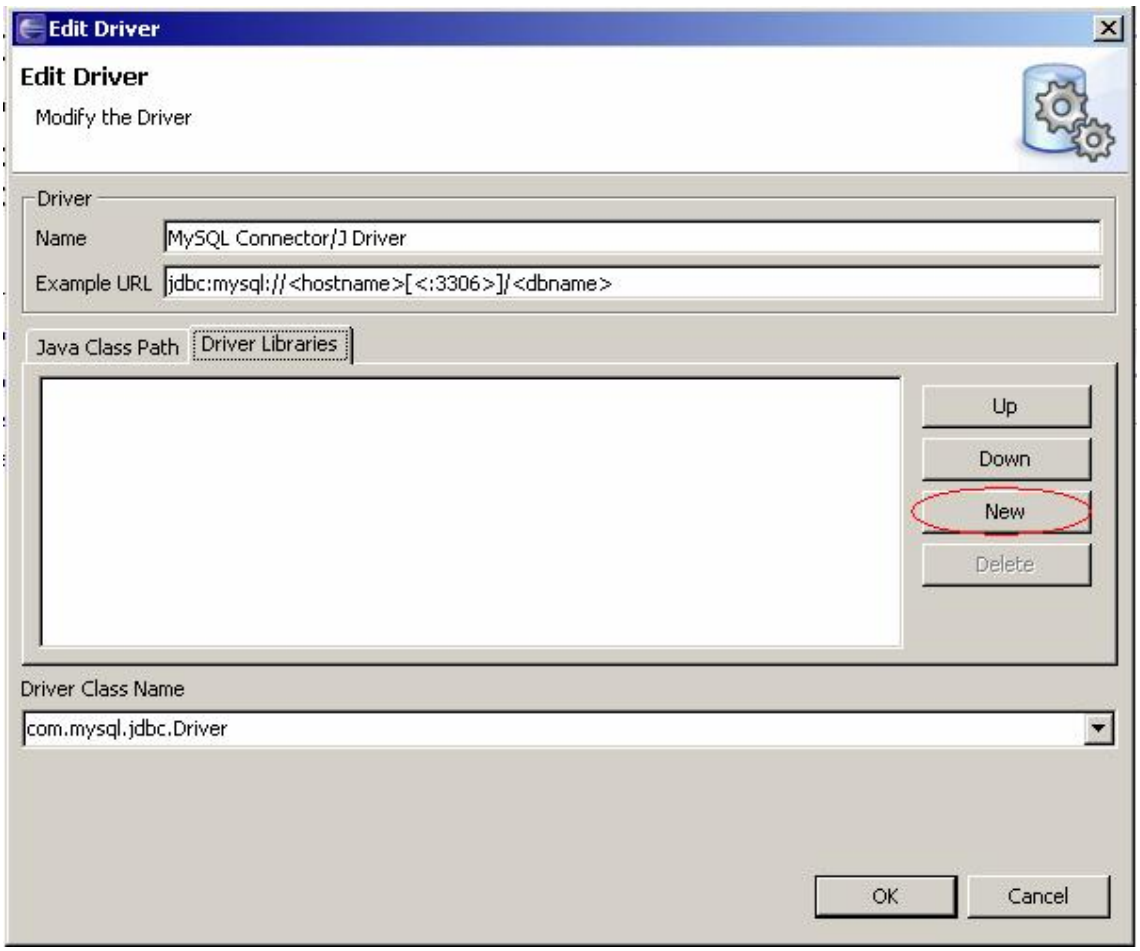
```
CREATE TABLE `message` (
              `Id` varchar(50) NOT NULL default '0',
              `text` varchar(50) default NULL,
              `nextMessage` int(4) default NULL,
              PRIMARY KEY  (`Id`)
            ) TYPE=MyISAM
```

创建 O/R Mapping:

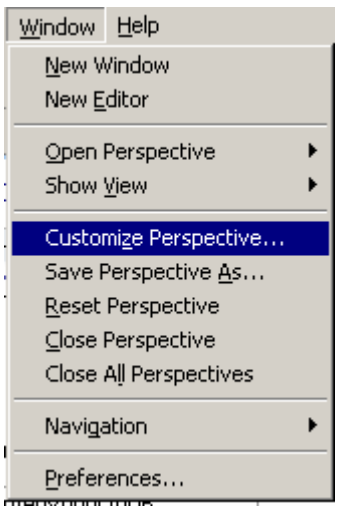


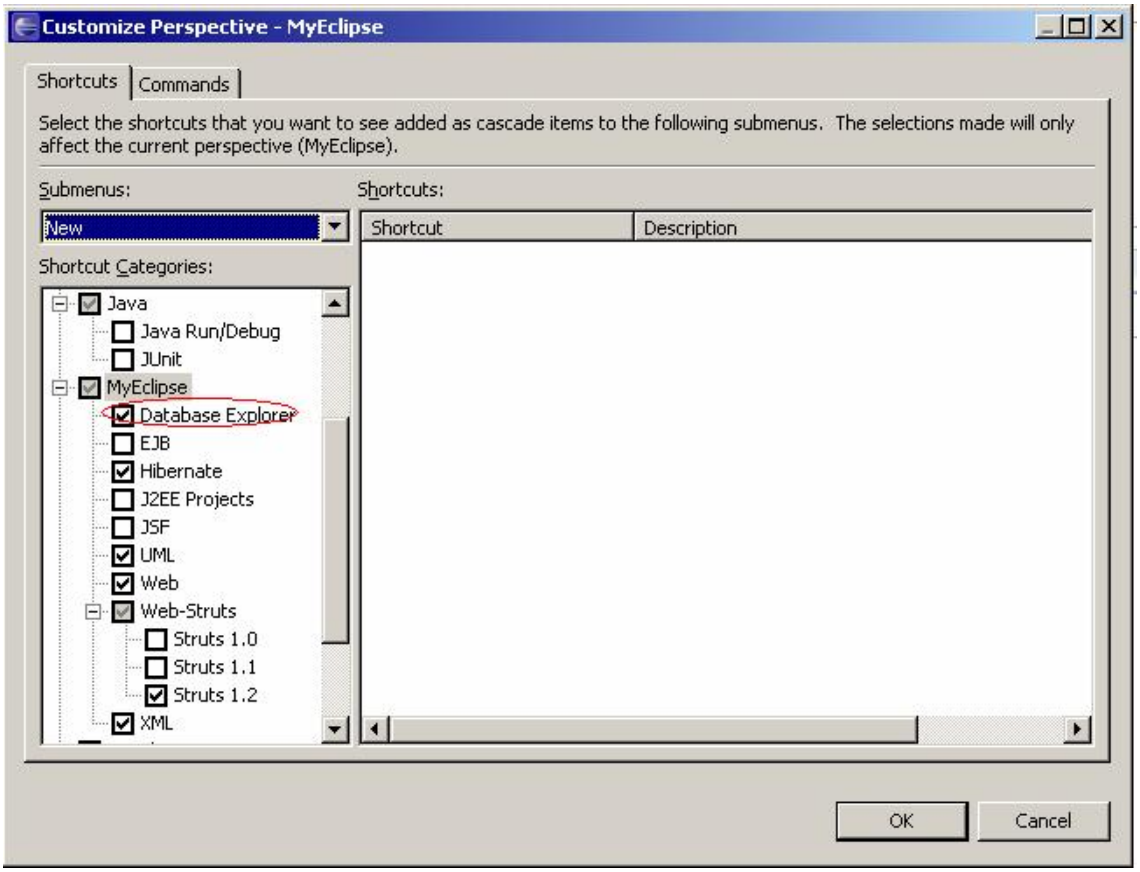

Preferences
type filter text
General
Ant
Help
Install/Update
Java
MyEclipse
Application Servers
Database Explorer
Drivers
ER-Designer
Editors
Internet Tools
J2EE Project
Project Capabilities
Subscription
UML
Validation
XDoclet
MyEclipse-Extras
Plug-in Development
Run/Debug
Team
Drivers
Axion
HSQLDB In-Memory
HSQLDB Server
HSQLDB Standalone
HSQLDB Web Server
IBM DB2 App Driver
InstantDB
InterClient
JayBird JCA/JDBC Driver for Firebird
JDBC ODBC Bridge
jTDS
Mckoi
Microsoft MSSQL Server JDBC Driver
Mimer SQL
MMMySQL Driver
MySQL Connector/J Driver
Oracle OCI Driver
Oracle Thin Driver
Pointbase Embedded
Pointbase Server
PostgreSQL
SAPDB
Sunopsis XML
Sybase Adaptive Server Anywhere
Sybase Adaptive Server Enterprise
Teradata JDBC4 Driver
ThinkSQL
Add
Edit
Copy
Remove
Restore Defaults
Apply
OK
Cancel

点击 Edit

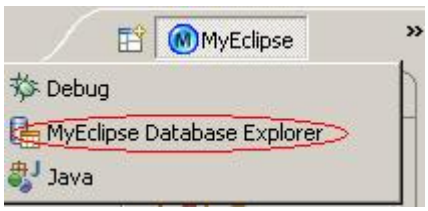


点击 New 添加 MySQL 的驱动，保存。



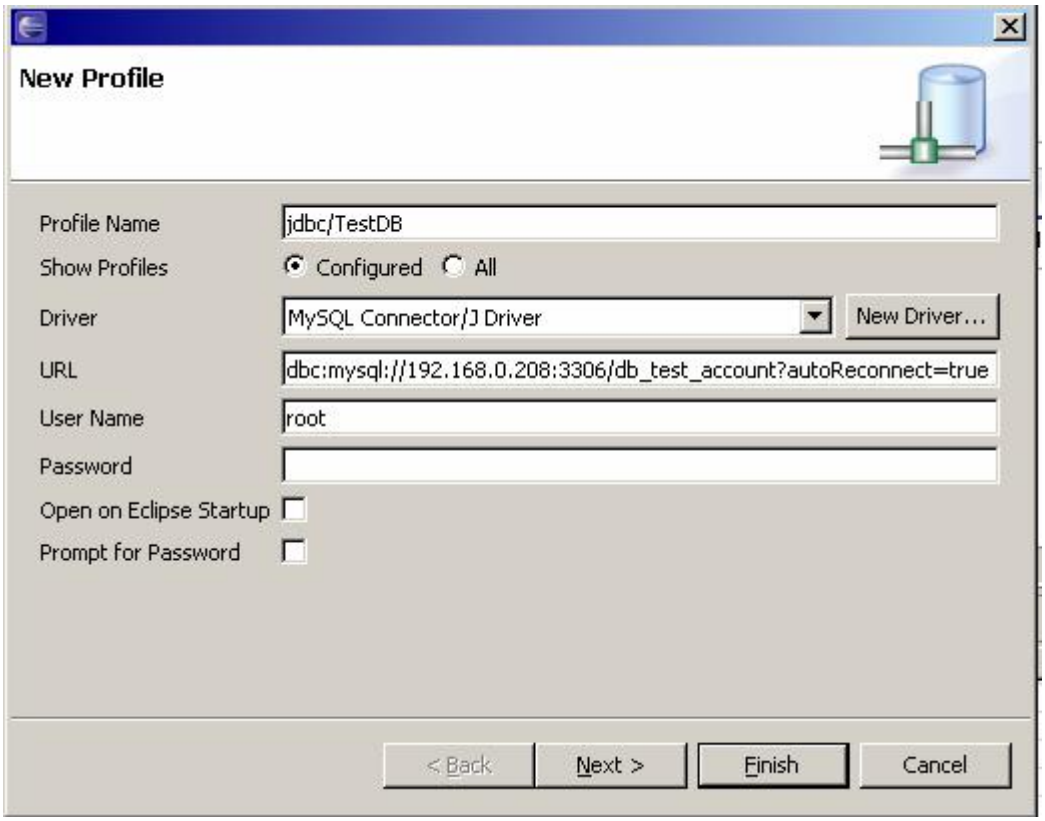


激活 Database Explorer，保存。

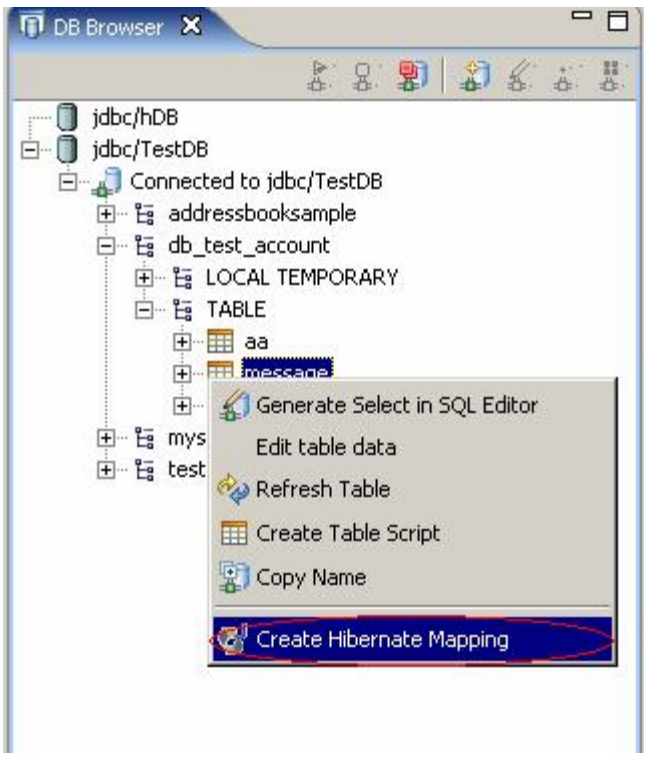


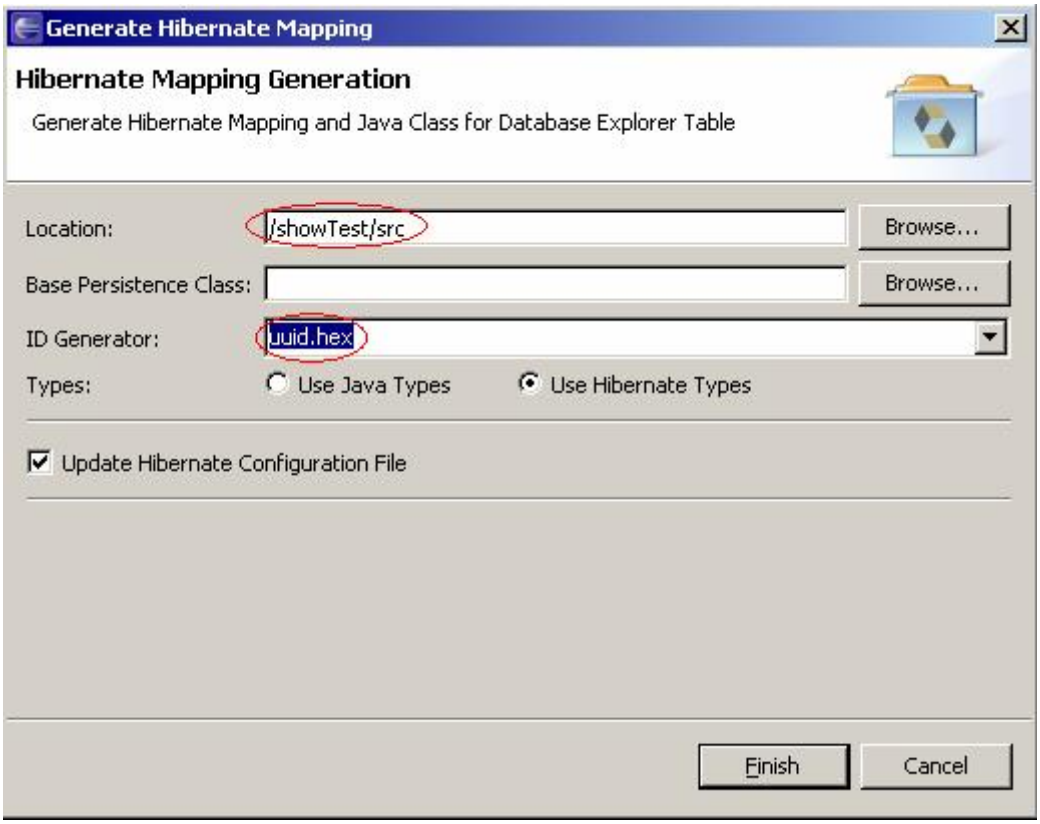
选择 Database Explorer，创建数据库链接：

点击创建新的数据库,添加所需必要的参数：



保存，然后右键点击该数据库选择**open database**，测试是否配置成功，如果不能连接查看该连接的配置参数。





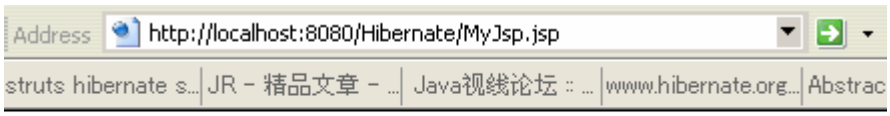
点击完成后MyEclipse会自动生成POJO和Map文件，并更新Hibernate的配置文件（主要是加载Map文件）。ID Generator选项可以根据你的需要进行选择，具体含义请参阅hibernate官方文档。下面建立一个测试用的jsp页面来看看Hibernate是否好用。

//MyJsp.jsp

```
<%@ page language="java" import="com.db.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServe
rPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'MyJsp.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>
    <%
                org.hibernate.Session s = HibernateUtil.currentSession();
                String hql = " from Message where text='b'";
                try {
                        org.hibernate.Query query = s.createQuery(hql);
                        java.util.List msgList = query.list();
                        hello.Message msg = (hello.Message) msgList.get(0);
                        out.println(msg.getId());
                        out.println(msg.getText());
                        out.println(msg.getNextmessage());
                } catch (org.hibernate.HibernateException e) {
                        e.printStackTrace();
                }
                HibernateUtil.closeSession();
    %>
  </body>
</html>
```

正常的结果如下：



4 b 3