北京**软件科技有限公司

J2EE 开发技术手册

部　　门：技术部
作　　者：******

版 本 号：001
创作日期：2005-9-1

北京**软件科技有限公司

目录

1. 引言

1.1 目的

本手册的目的是为了方便 java 开发者对本平台、及平时开发中所用到的所有技术有个全面的把握。并在平时的开发中能够提供一定的帮助作用。

1.2 要求

本手册尽量通俗易懂，在平台搭建的介绍中，即使是对 java 知之甚少的人，只要按照手册的步骤，一步一步的配置，也会搭建出一个标准的 java 开发平台。在技术难点的讲解上，尽量做到清晰明了。

2. 平台软件

2.1 Eclipse 3.1【开发工具】

下载网址：http://www.eclipse.org/downloads/index.php

下载网址：http://www.jboss.com/products/jbosside/downloads(与 JBoss 集成)

2.2 MyEclipse4.0 【开发工具】

下载网址：http://www.myeclipseide.com

EnterpriseWorkbenchInstaller_4.0M2_E3.1

EnterpriseWorkbenchInstaller_4.0GA_E3.1（最新版：支持 JSF 可视化插件和 UML 插件）

2.3 Apache Tomcat 5.5.9【服务器】

普通版本： 下载网址：http://jakarta.apache.org/tomcat/

集成单点登录： 下载网址：http://www.josso.org

2.4 JBoss【服务器】

下载网址：http://www.jboss.com/products/jbosside/downloads

2.5 JDK1.5【JAVA 虚拟机】

下载网址：http://onesearch.sun.com

2.6 Hibernate 3.0【数据持久层】

下载网址：http://www.hibernate.org/

2.7 Java Faces Server 1.1【Web 层】

下载网址：

2.7 Subversion【源代码控制】

Svn【服务器端】	下载网址：http://subversion.tigris.org/
Smartsvn【客户端】	下载网址：http://www.smartsvn.com/
Rapidsvn【客户端】	下载网址：http://rapidsvn.tigris.org
TortoiseSVN【客户端】	下载网址：http://www.tortoisesvn.org
Cvs【服务器端】	下载网址：http://www.cvsnt.com

3. 平台搭建步骤

下载了上面所有的软件以后，我们就可以开始按照以下步骤来搭建 java 开发平台：

3.1 基本配置

3.1.1 建立目录

在系统的 D 盘新建一个文件夹 platform，作为我们 java 平台工作的根目录。然后再在 platform 里面分别新建六个文件夹：eclipse、myeclipse、jre、tomcat、study、svnclient、doc。这六个文件夹分别存放的文件如下：eclipse 存放 eclipse 安装文件、myeclipse 存放 myeclipse 安装文件、jre 存放 jdk 安装文件、tomcat 存放 tomcat 安装文件、study 存放自己建立的工程、Svnclient 存放版本控制客户端的安装文件、doc 存放平时开发中的一些帮助文档。

3.1.2 配置 Eclipse

把下载的 Eclipse 3.1 解压到 eclipse 中。

3.1.3 配置 myeclipse

下载并安装 MyEclipse4.0，注意在安装过程中首先要选择 eclipse 的目录，然后再选择 myeclipse 目录。安装完成后会在 eclipse 目录中多出 links 文件夹和 图标文件。两种 Myeclipse 版本的破解号分别如下：

Myeclipse4.0GA_E3.1 注册码

Subscriber：magsee **Subscription Code**：zLR8ZC-955-56-6067865966857138

Subscriber：magsee.com **Subscription Code**：zLR8ZC-955-56-6067865796537877

Myeclipse4.0M2_E3.1 注册码

Subscriber：etrelin **Subscription Code**：rAR7ZL-819-56-5467865348543780

Subscriber：dearmite **Subscription Code**：qAR7ZL-819-56-54678656594317125

Subscriber：abenmao **Subscription Code**：nAR7ZL-819-56-54678656017372584

Myeclipse4.03GA 注册码

Subscriber：gdmmking **Subscription Code**：tLR8ZC|956444|535353|634969005

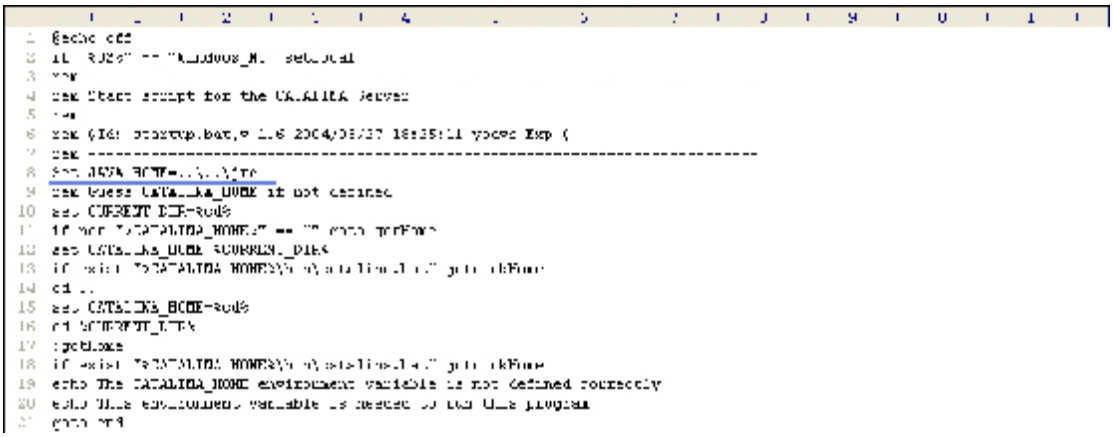
3.1.4 配置 jdk

下载并安装 JDK1.5，假如我的安装路径是 D:\jdk1.5 ，安装完毕后把 D:\jdk1.5 目录下的所有文件全部拷贝到 jre 文件夹下。如图所示：

名称	大小	类型	修改日期	属性
bin		文件夹	2005-8-16 16:34	
demo		文件夹	2005-8-16 16:33	
include		文件夹	2005-8-16 16:32	
jre		文件夹	2005-8-16 16:34	
lib		文件夹	2005-8-16 16:34	
sample		文件夹	2005-8-16 16:33	
src		文件夹	2005-10-4 13:54	
COPYRIGHT	3 KB	文件	2005-6-3 4:32	A
LICENSE	16 KB	文件	2005-6-3 3:35	R
LICENSE.rtf	21 KB	RTF 格式	2005-8-16 16:32	A
README.html	21 KB	HTML Document	2005-6-3 3:35	R
src.zip	17,631 KB	WinRAR ZIP 压缩...	2005-6-3 3:35	A
THIRDPARTYLICENSEREADME.txt	69 KB	文本文档	2005-6-3 3:35	R

3.1.5 配置 tomcat

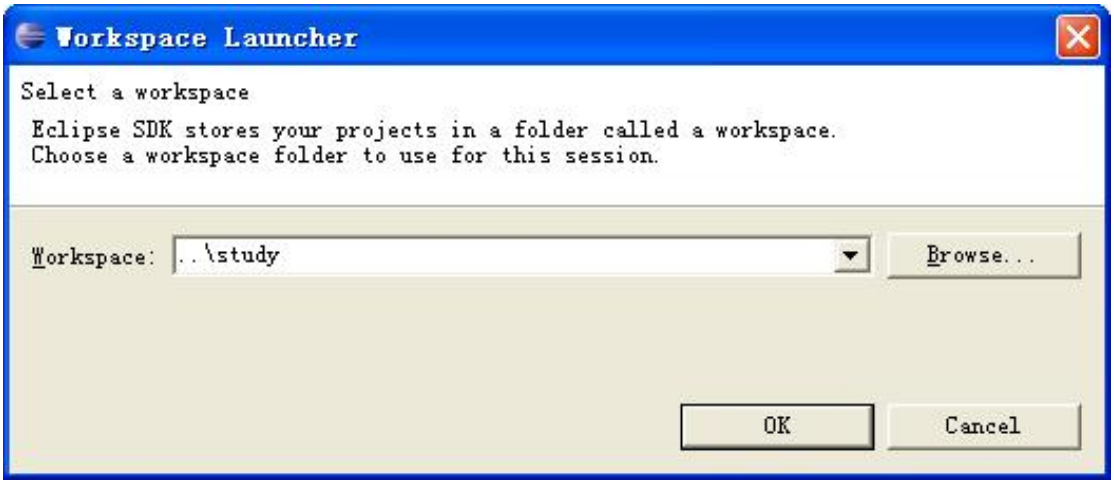
下载 Apache Tomcat 5.5.9 并解压到 tomcat 目录中，随后进入 D:\ platform\tomcat\bin 目录中，打开 startup 文件，在 rem----------下面一行加入如下一条语句：Set JAVA_HOME=..\..\jre（跳出 bin 目录、再跳出 tomcat 目录、最后才进入 jre 目录），如图所示：



```
@echo off
rem Start script for the CATALINA Server
rem
rem $Id: startup.bat,v 1.6 2004/03/27 18:35:11 yoavs Exp $
rem ---------------------------------------------------------------------------
set JAVA_HOME=..\..\jre
rem Guess CATALINA_HOME if not defined
set CURRENT_DIR=%cd%
if not "%CATALINA_HOME%" == "" goto gotHome
set CATALINA_HOME=%CURRENT_DIR%
if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
cd ..
set CATALINA_HOME=%cd%
cd %CURRENT_DIR%
:gotHome
if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
echo The CATALINA_HOME environment variable is not defined correctly
echo This environment variable is needed to run this program
goto end
```

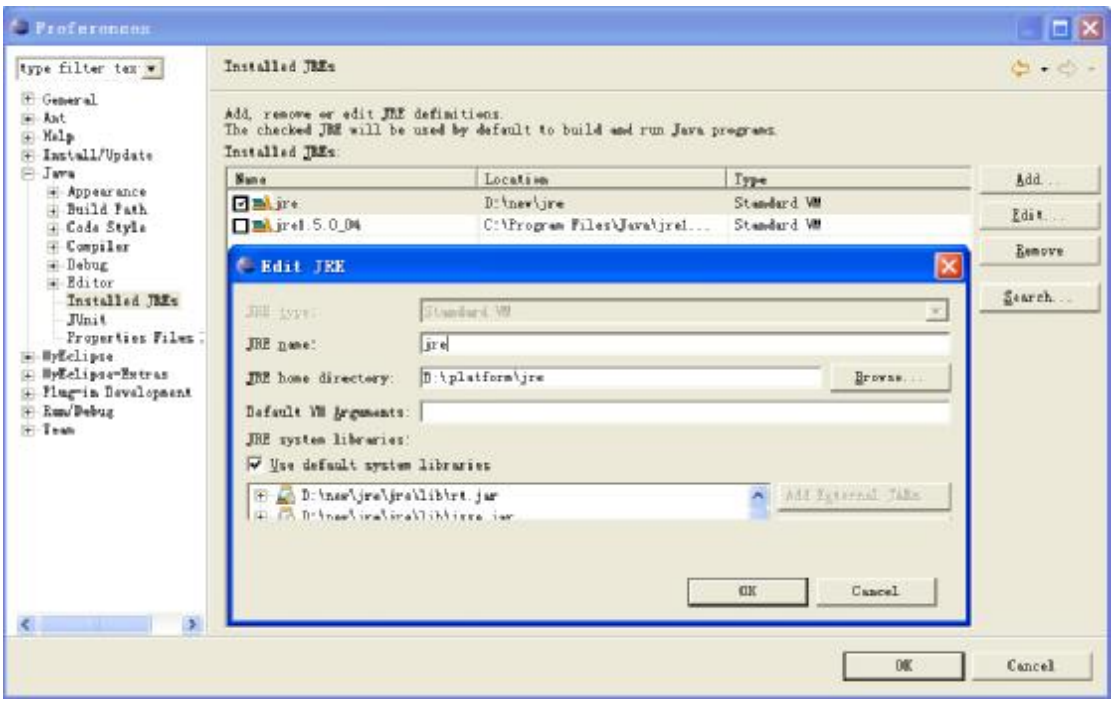
3.1.6 配置项目空间

打开 eclipse 文件夹中的 eclipse.exe 文件，点击 file->switch workspace 在 workspace 输入框中输入自己工作项目所存放的文件夹位置..\study（先跳出 eclipse 目录，再进入 study 目录）。如图所示：



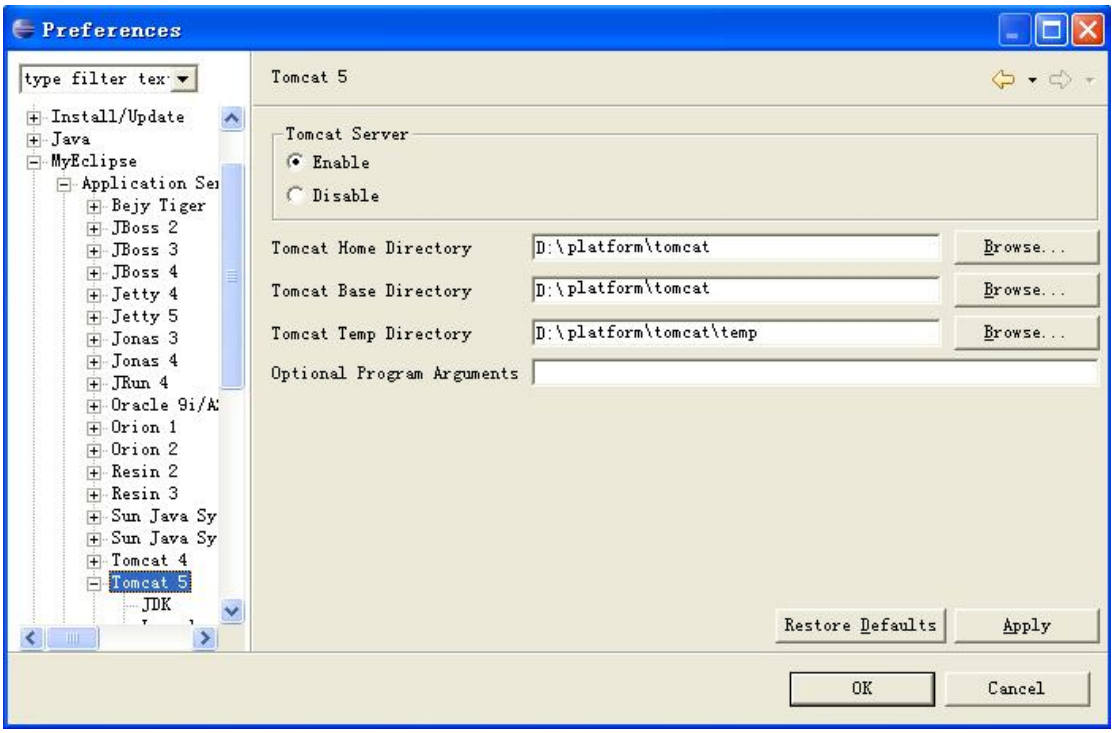
3.1.7 平台绑定 java 虚拟机

点击 window->preferences… 再点击 java->Editor->Installed JRES 导入自己 eclipse 目录下的 jre 即可。如图所示：



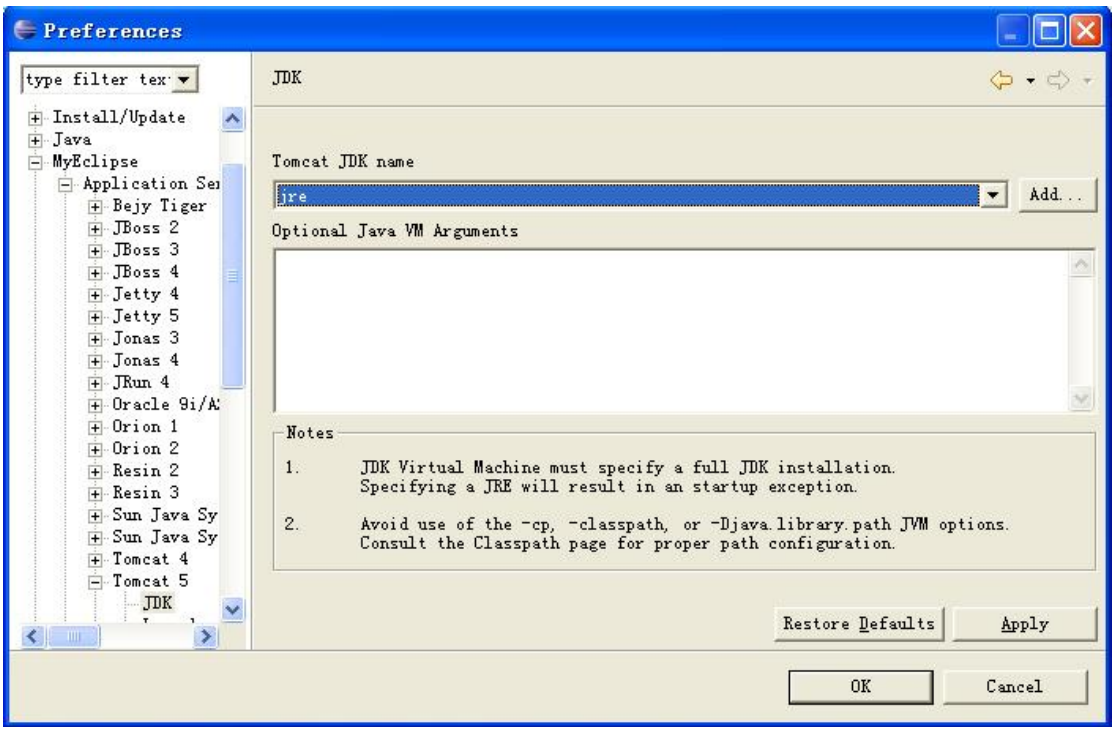
3.1.8 平台邦定 tomcat

点击 MyEclipse->Application Servers->Tomcat 5 然后选中自己 eclipse 目录下的 tomcat 目录即可。如图所示：



3.1.9 tomcat 绑定 jre

MyEclipse->Application Servers->Tomcat 5->JDK 选中 eclipse 目录下的 jre（即是在第八步骤中我们为 JRE 取的别名）即可。如图所示：



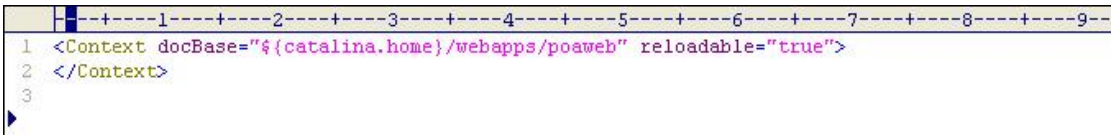
3.1.10 配置项目自动发布

如果想开发过程当中工程自动刷新的话，可以在 D:\latform\tomcat\conf\Catalina\localhost 目录下新建一个文件 poaweb（自己的项目名称）的 XML 文件 ，里面内容如下：

<Context docBase="${catalina.home}/webapps/poaweb" reloadable="true">

</Context>

如图所示：



注：${Catalina.home}会自动解析为 tomcat 的绝对路径，就是说上面的也可以写成如下形式：

<Context docBase="D:/ jplatform/tomcat/webapps/poaweb" reloadable="true">
</Context>

3.1.11 配置连接池

连接池的配置涉及到以下两个文件：hibernate.cfg.xml，另外一个文件需要手动新建,进入 tomcat\conf\Catalina\localhost 目录下新建一个 xml 文件，文件名为工程名 poaweb.xml hibernate.cfg.xml 文件内容如下：

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>
<session-factory>
    <property name="myeclipse.connection.profile">xuhuajian</property>
    <property name="connection.datasource">java:comp/env/haha</property>
    <!--
    <property name="connection.url">
        jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=study
    </property>
    <property name="connection.username">sa</property>
    <property name="connection.password">as</property>
    <property name="connection.driver_class">
        com.microsoft.jdbc.sqlserver.SQLServerDriver
    </property>
```

```
    -->
    <property name="dialect">
        org.hibernate.dialect.SQLServerDialect
    </property>
    <property name="show_sql">true</property>
    <mapping resource="pojo/Student.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

poaweb.xml 文件内容如下：

```
<!--
    Context configuration file for the Tomcat Manager Web App
    $Id: manager.xml,v 1.3 2004/08/26 17:03:34 remm Exp $
-->
<Context docBase="${catalina.home}/webapps/exercise" reloadable="true">
<WatchedResource>WEB-INF/web.xml</WatchedResource>
<!—mysql 版本配置-->
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver" maxActive="20" maxIdle="10"
maxWait="-1" name="haha" password="as" type="javax.sql.DataSource"
url="jdbc:mysql://localhost:3306/study" username="sa"/>

<!—oracle 版本配置-->
<Resource auth="Container" driverClassName="oracle.jdbc.driver.OracleDriver" maxActive="20"
maxIdle="10" maxWait="-1" name="haha" password="as" type="javax.sql.DataSource"
url="jdbc:oracle:thin:@localhost:1521:study " username="sa"/>

<!—sql 版本配置-->
<Resource SendStringParamAsUnicode="false" auth="Container"
driverClassName="com.microsoft.jdbc.sqlserver.SQLServerDriver" maxActive="20" maxIdle="10"
maxWait="-1" name="haha" password="as" type="javax.sql.DataSource"
url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=study" username="sa"/>
</Context>
```

注：tomcat 连接池的配置最好等到项目发布的时候去配置，如果不这样，那么在开发的过程中进行单独的类测试的时候会报错。另外，poaweb.xml 文件的内容也可以配置在 server.xml 文件中，效果是一样的。

3.2 附加插件

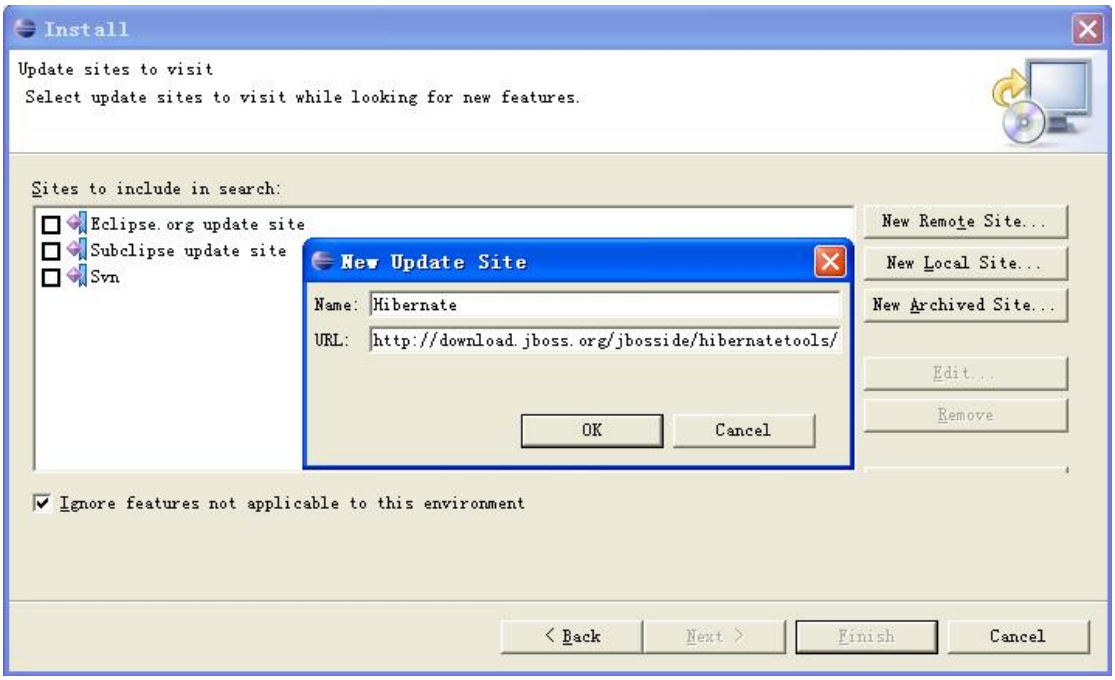
这样，我们的开发平台基本上搭建完毕，但为了开发方便，我们还要安装几个插件：

3.2.1 hibernate 插件

虽然 myelipse 中自带了 hibernate 的插件，但为了更高效的开发，我们应用另外一个插件，更新网址是：http://download.jboss.org/jbosside/hibernatetools/updates/development

操作的步骤是：Help->Software Updates->Find and Install...->Search for new features to install

->next->New Remote Site... 如下图所示：

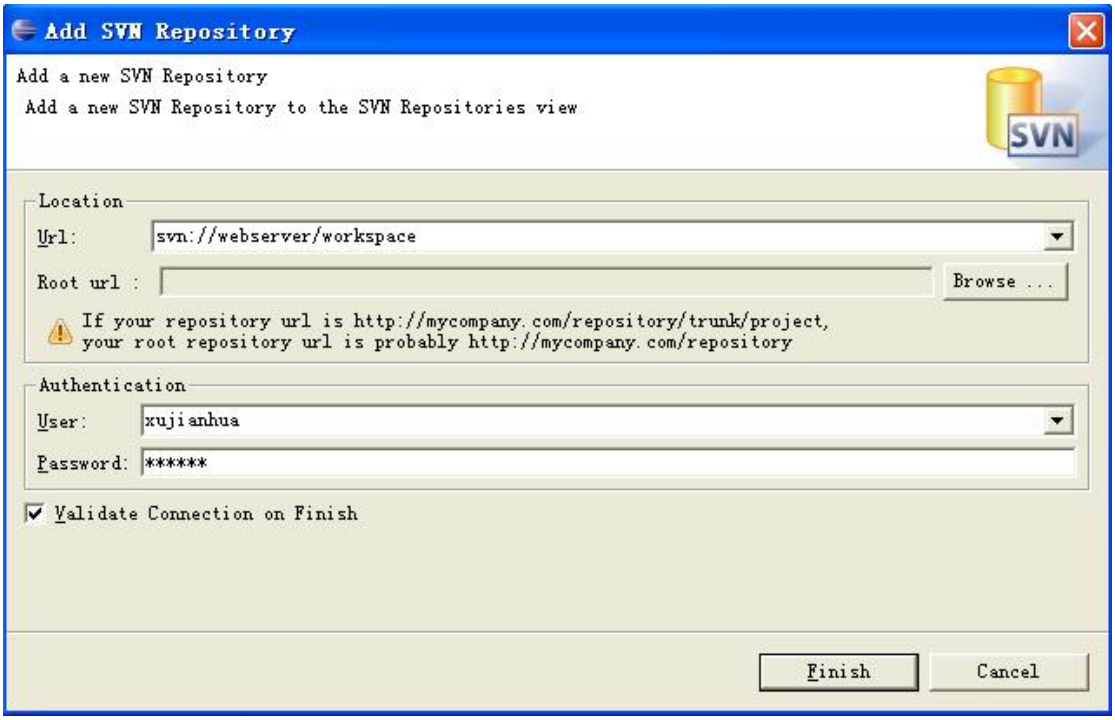


另外，还有个 Hibernate 插件的更新站点是：http://www.binamics.com/hibernatesync

当然，我们还可以用手动复制的方式来安装 Hibernate 插件，打开 http://www.hibernate.org 下载 Hibernate Tools 包，下载以后可以通过直接复制和 link 两种方式来安装 Hibernate 插件。

3.2.2 Svn 插件

这里介紹 Eclipse 的 SVN Plugin，叫做 Subclipse，官方网站 http://subclipse.tigris.org/ 他也有提供 update site 站点： http://subclipse.tigris.org/update。在 Eclipse 选项中选 Help->Software Updates->Find and Install-> Search for new features to install ->New Remote Site。URL 就输入刚刚的 http://subclipse.tigris.org/update。安装起来就好了。安装后会有許多新的 View 以及多一个 SVN Repository Exploring 的 Perspective。在 SVN Repository 中按右鍵 New -> Repository Location。以刚刚的例子来说，url 为 svn://webserver/workspace，账号为 xujianhua，密码为 123456。如下图：



使用 SVN 可以在 Project 按右键 Team -> Share Project -> SVN ->刚刚建立的 Repository 其余的使用方法和 CVS 都很类似。

3.2.3 UML 插件

虽然此 UML 插件功能强大，但不建议使用，因为最新版本的 Myeclipse（EnterpriseWorkbenchInstaller_4.0GA_E3.1）中已经自带 UML 插件，用起来简单而方便。此 UML 插件的下载地址是：http://www.omondo.com/download/index.html 或者直接下载安装包文件 http://eclipseuml.free.fr/eclipseUML_E310_freeEdition_2.1.0.beta.20050816.jar 下载后直接双击该 jar 文件即可安装，安装过程中必须选择 eclipse 的目录。

3.2.4 JBoss 插件

JBoss 插件的更新网址：http://jboss.sourceforge.net/jbosside/updates/

3.2.5 插件安装分类

第一种：建立 link 文件方式

打开 D:\platform\eclipse，我们会发现一个 links 文件夹，然后在你的 D:\platform\eclipse\links\目录下创建一个 link 文件，比如我要安装一个 vss 插件，我就在我的 links 目录下创建了：VSS_1.6.1.link 文件。VSS_1.6.1.link 文件的内容如下：

path=D:\\platform\\myplugins（等于 path=D:/platform/ myplugins），然后将自己的插件放在插件目录（PLUGIN_HOME，比如在我的机器上插件目录为 D:\platform\myplugins），而这个 VSS_1.6.1.link 文件则是指向的我的插件目录。

这种安装插件方式时需要注意，如果你是一个单独的 jar 文件，则最好在 link 文件指定的目录下创建这样级别的目录：D:\platform\myplugins\eclipse\plugins\xxx.xxx（插件名称和版本号），然后将 jar 文件放在这个目录下即可。比如 vss 插件在我的机器上的目录是

D:\ platform\myplugins\eclipse\plugins\org.vssplugin_1.6.1，下有一个文件：vssplugin.jar，两种方法的插件安装之后需要重新启动 eclipse 才能看到插件。如果某一个插件已经安装了想要重新安装，只要将文件替换或者修改 link 文件的路径即可。如果发现认不到插件，可能是你的插件不支持你的当前 eclipse 版本，请检查。也有可能是系统配置引起的，我出现过一次，我的解决方法是将 ECLIPSE_HOME 下的 configuration 目录下的所有文件删除，剩下 config.ini 文件。

第二种：直接拷贝

这种安装方式非常简单，比如我们现在的 eclipse 的根目录是 D:\platform\eclipse,只要将插件拷贝到 D:\ platform\eclipse\plugins 目录下就可以了，例如我有一个 weblogic 的插件（bea 站点上可以下载），解压缩之后得到得到一个目录：

【com.bea.eclipse.weblogic_1.1.1】,我就将这个目录直接放到 D:\ platform\eclipse\plugins 目录下，重新启动 eclipse 之后就看到 Run->“Start Weblogic”的菜单，安装之后，weblogic 插件的全路径为：D:\ platform\eclipse\plugins\com.bea.eclipse.weblogic_1.1.1]

第三种：在 Eclipse 中选择远程更新

这种安装方式上面已经提到，这也是最普遍的一种插件安装方式。比如上面的 Svn 插件、Hibernate 插件都是属于这种方式。

第四种：直接安装方式（这种安装方式有两种情况）

1、可执行文件（exe）：直接点击该 exe 文件，在安装的过程中选择 eclipse 目录即可比如 Myeclipse 即是这种方式。

2、可执行压缩文件（jar）：该文件类型是 Executable Jar File，也只要直接点击该文件，在安装的过程中选择 eclipse 目录即可。比如：eclipseUML_E310_freeEdition_2.1.0.beta.20050816.jar 即是这种方式。

4. 版本控制服务器

4.1 Svn 服务器

一、Subversion 的下载与安装

Subversion 的官方网站：http://subversion.tigris.org/ 。可以在官方网站上下载 Subversion 的最新版本。本说明使用的是 1.2.1 的 Win32 版。下载下来 Subversion 安装文件后，直接安装就可以了。下载下来 Subversion 安装文件后，直接安装就可以了。假设安装路径是 C:\svn. 另外、我们还必须下载一个有用的 SVNService.exe,从 http://dark.clansoft.dk/~mbn/svnservice/ 下载，是专为 SubVersion 开发的一个用来作为 Win32 服务挂接的入口程序。把 SVNService.zip 压缩文件解压以后，把里面的 SVNService.exe 拷贝到 C:\svn\Subversion\bin 目录中。

二、Subversion 的建立与使用仓库使用

安装好 SubVersion，然后进入 DOS 命令，使用 svnadmin create 创建工作区域，比如 C:\Documents and Settings\Administrator>cd\svn\Subversion\bin 回车后显示如下：

C:\svn\Subversion\bin

C:\svn\Subversion\bin>svnadmin create d:\ mysvn 回车

尝试使用文件管理器去查看 mysvn 这个目录，里面密密麻麻配好了版本控制需要的数据库结构,我们要修改 D:\zhongguo\conf 文件夹下的两个文件：passwd,svnserve.conf

Passwd 文件修改后的效果如下：

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.
[users]                                 /*去掉前面的#号*/
harry = harryssecret                    /*去掉前面的#号*/
sally = sallyssecret                    /*去掉前面的#号*/
liu_xiaohua = 123456                    /*登录时的用户名和密码*/
```

svnserve.conf 文件修改后的效果如下：

```
### This file controls the configuration of the svnserve daemon, if you
### use it to allow access to this repository.  (If you only allow
```

```
### access through http: and/or file: URLs, then this file is
### irrelevant.)
### Visit http://subversion.tigris.org/ for more information.
[general]                    /*去掉前面的#号*/
### These options control access to the repository for unauthenticated
### and authenticated users.  Valid values are "write", "read",
### and "none".  The sample settings below are the defaults.
anon-access = read           /*去掉前面的#号*/
auth-access = write          /*去掉前面的#号*/
### The password-db option controls the location of the password
### database file.  Unless you specify a path starting with a /,
### the file's location is relative to the conf directory.
### Uncomment the line below to use the default password file.
password-db = passwd
### This option specifies the authentication realm of the repository.
### If two repositories have the same authentication realm, they should
### have the same password database, and vice versa.  The default realm
### is repository's uuid.
realm = My First Repository        /*去掉前面的#号*/
```

三、Subversion 服务器的搭建

等上面配置完以后，再执行以下命令：

C:\svn\Subversion\bin>svnservice -install -d -r d:\mysvn 结果如下表示成功

[该文件运行时可带参数，常用的参数有两个一个是“-d”，该参数表明服务器作为一个精灵进程一直运行，直到手动结束该程序。另一个参数就是“-r”，该参数指定服务器进程寻找 Repository 的根路径]

```
SVNService installed.
Commandline set: "-d" "-r" "d:\mysvn"
```

这样我们就把 svnservice 挂接成为 windows 的服务了，最后我们只要启动这个服务就行，C:\svn\Subversion\bin>net start svnservice 结果如下表示成功

```
SVNService 服务正在启动 .
SVNService 服务已经启动成功。
```

或者直接到 开始->所有程序->管理工具->服务中找到 SVNService，然后手工启动

注：svnservice 有时候回出现假服务现象，解决的方法如下：

首先停止服务：　　C:\svn\Subversion\bin>net stop　svnservice

然后彻底清除服务：C:\svn\Subversion\bin>svnservice.exe –remove

再安装服务：　　C:\svn\Subversion\bin>svnservice -install -d -r d:\svn

最后启动服务：　　C:\svn\Subversion\bin>net start svnservice

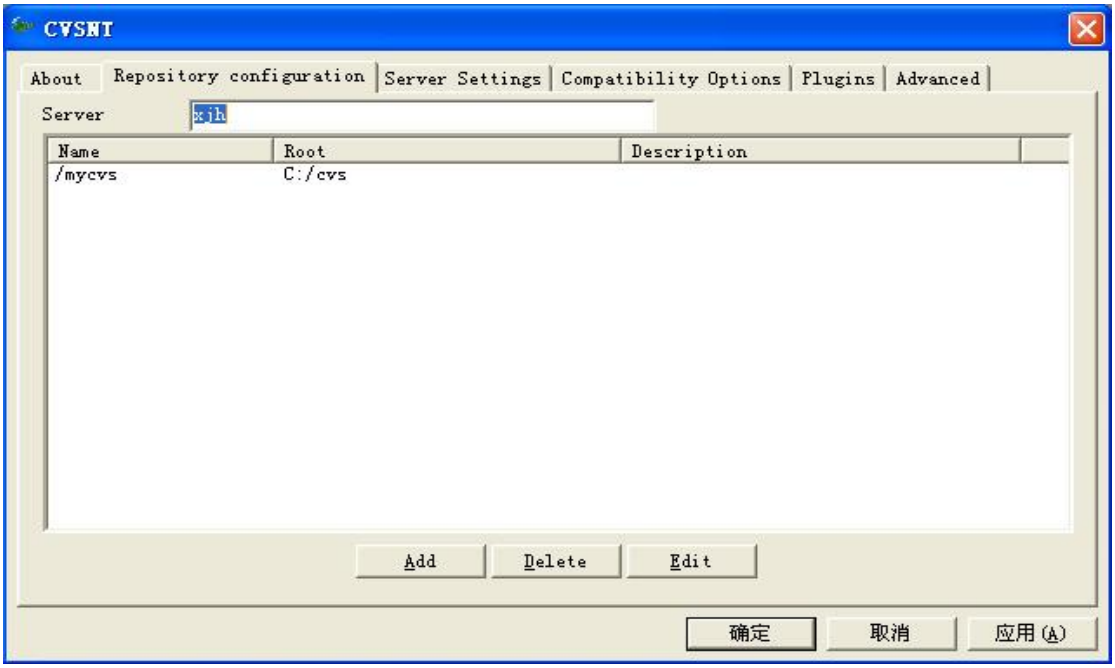
4.2　Cvs 服务器

一、安装 CVSNT

先到 http://www.cvsnt.com 下载服务器端软件，并安装，安装完后启动 CVSNT Control Panel：

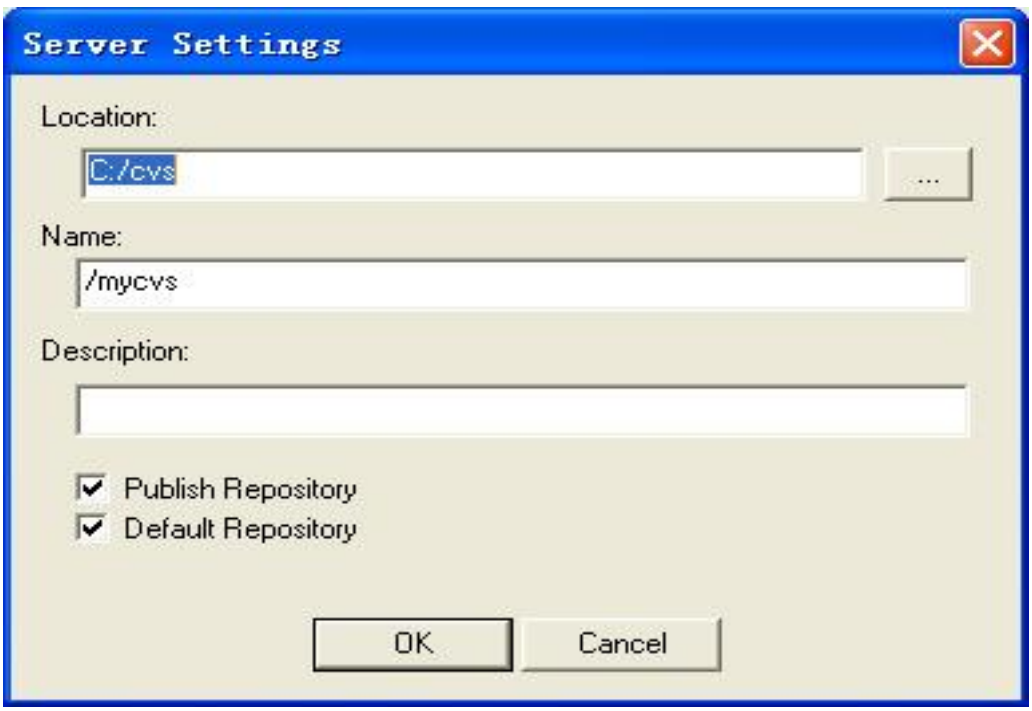
二、创建 CVS Repository

选择 Repository　configuration 页，点击 Add 按钮创建新的 CVS Repository。



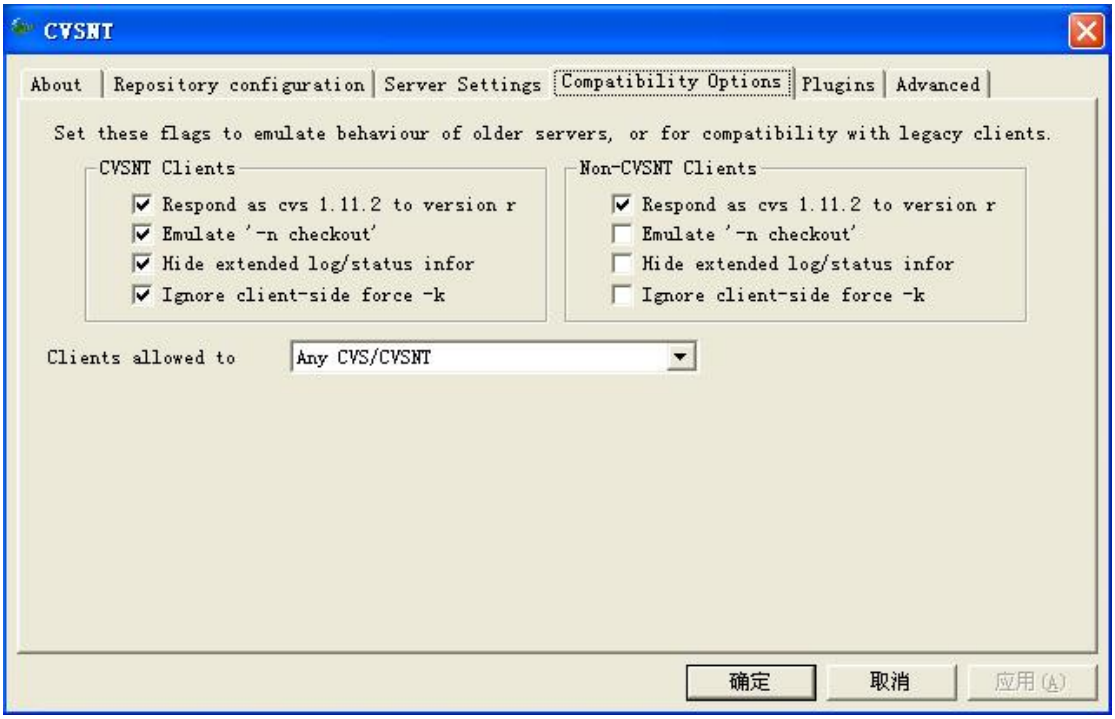
注意：

- Location 部分指定保存 Repository 的目录（可以新建）
- Name 部分指定 Repository 名（以/开始） 、本例中我用 mycvs
- 确保 Publish Repository 选中
- 在确认初始化 Repository 就创建好了 Repository

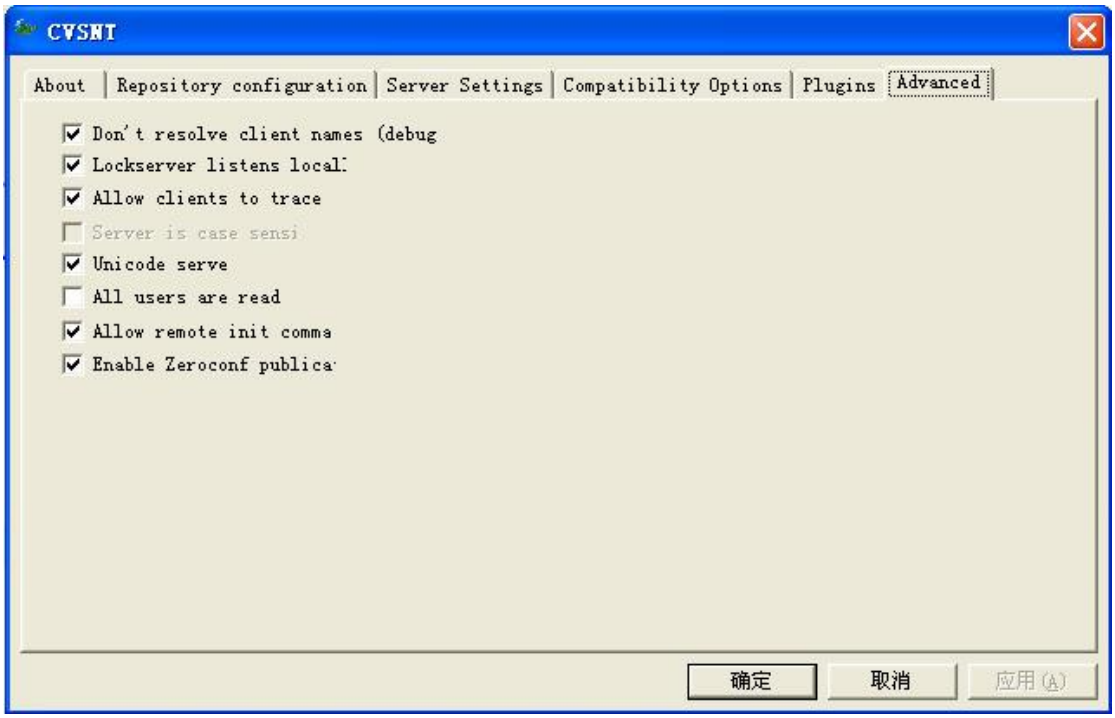


三、配置选项

- 为了使 Eclipse 能够正常和 CVSNT 通信，还要将 Compatibility 页中的 Respond as cvs 1.11.2 to version request 项选中：



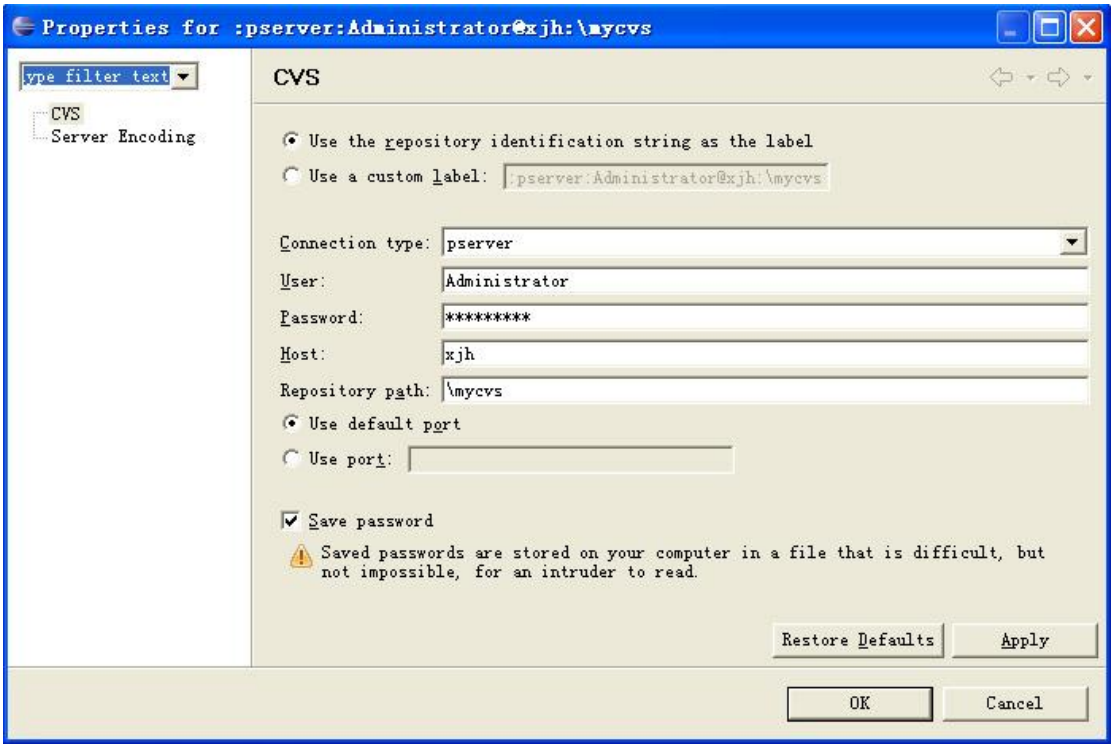
- 确保 Advanced 页中的 All users are read 不被选中，不然的话，项目不能提交进仓库



四、用户管理

- CVSNT 最简单的用户管理方式就是和 NT 验证结合，这样，创建一个 NT 用户也就创建了一个 CVS 用户，鼠标右击我的电脑->选种管理->本地用户和组。添加一个新用户即可。
- 通过系统管理员设置用户的访问权限来控制用户对 CVS 的访问，例如，为了使用户能够向 CVS 提交更新的文件，就必须给该用户变更的权限

五、共享工程



用户名和密码就是我本机登录是的用户名和密码。

5. 命名规范

5.1 包名

唯一的包名称前缀总是由全小写的 ASCII 字母组成，而且应该是一个顶级域名，诸如.com、.edu 或其他正式的顶级域名包名称的其余部分可以根据组织自己内部的命名传统而定，使用点将各部分分开例如:com.sun.eng 等。现对公司所有项目的包名规范如下。所有包名都以 mj 开始：

持久化包：mj.pojo　　实用工具包：mj.util　　消息包：mj.message

国际化包：mj.resource　　前台开发包：mj.dev.项目名　　后台开发包：mj.dev.admin

剩下的就是模块包：(比如我们 OA 办公自动化系统)

	前台	后台
公告模块	mj.dev.baiyyy.bulletin	mj.dev.admin.bulletin
邮件模块	mj.dev.baiyyy.email	mj.dev.admin.email
论坛模块	mj.dev.baiyyy.bbs	mj.dev.admin.bbs
工作流模块	mj.dev.baiyyy.workflow	mj.dev.admin.workflow

5.2 类名

类（和接口）名称应该是一个名词，描述类的目的。类名是大小写混合的，第一个字母大写，里面每个单词的首字母大写。使用完整的单词，避免使用缩写。例如：

Point, Shape, MovieEditor, ClientList

5.3 页面文件夹命名规范

前台：css、images、swf、js、admin、模块名（bulletin、eamil、bbs、workflow）

后台（包含在 admin 中）：模块名（bulletin、eamil、bbs、workflow）

5.4 页面名及页面的 Form 名

页面名都以小写字母开头，后面的单词首字母大写，比如：页面名为 addMan.jsp、相应的 Form 的名称应该是 addManForm 如下所示：

```
<html>
<head><title>增加联系人</title></head>
<body>
<f:view>
    <h:form  id="addManForm">
        ……
    </h:form>
</f:view>
</body>
</html>
```

5.5 Faces-config.xml 配置文件

配置文件可以分模块：

如：faces-config-mail.xml

Bean的格式为： AddManBean

类的实例名称为： AddManBean

Bean注册如下所示：

```
<managed-bean>
<managed-bean-name>AddManBean</managed-bean-name>
  <managed-bean-class>pageBean.email.AddManBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

组件(组件中包括更多的 bean)注册如下：

```
    <component>
        <component-type>Scroller</component-type>
```

```
        <component-class>
            components.components.ScrollerComponent
        </component-class>
        <component-extension>
            <component-family>Scroller</component-family>
        </component-extension>
    </component>
```

注：如果组件是 jar 文件，并且该 jar 文件中有 faces-config.xml 文件的话，那么这里就不需要配置；如果没有 faces-config.xml 的话，那这里还是要配置的哦

5.6 代码规范

1、 方法名小写字母开头、并禁用 get、set 保留字

2、 方法及类都需要加注释，至少包含参数、返回值、具体功能描述

3、 对数据库的操作尽量用事务，以防止数据库死锁

4、 一个页面对应一个 Bean、便于以后的维护

5、 建议使用 log4j

6. 单点登录

6.1 Tomcat 下载

为了更好的把握单点登陆的思想，我们只要按照如下步骤配置就行：

首先，假设我们在 D 盘新建一个文件夹为 Test，再在 Test 文件夹中新建两个文件夹 jre 和 tomcat，分别用来存放 jdk1.5 和 tomcat5.5.9 。tomcat5.5.9 的下载网址如下：http://www.josso.org ，由于此 tomcat 已经集成了单点登录技术，因此我们只需要进行简单的一些配置便可以直接使用。此外，我们还要下载 tomcat 的 admin 模块，下载网址如下：

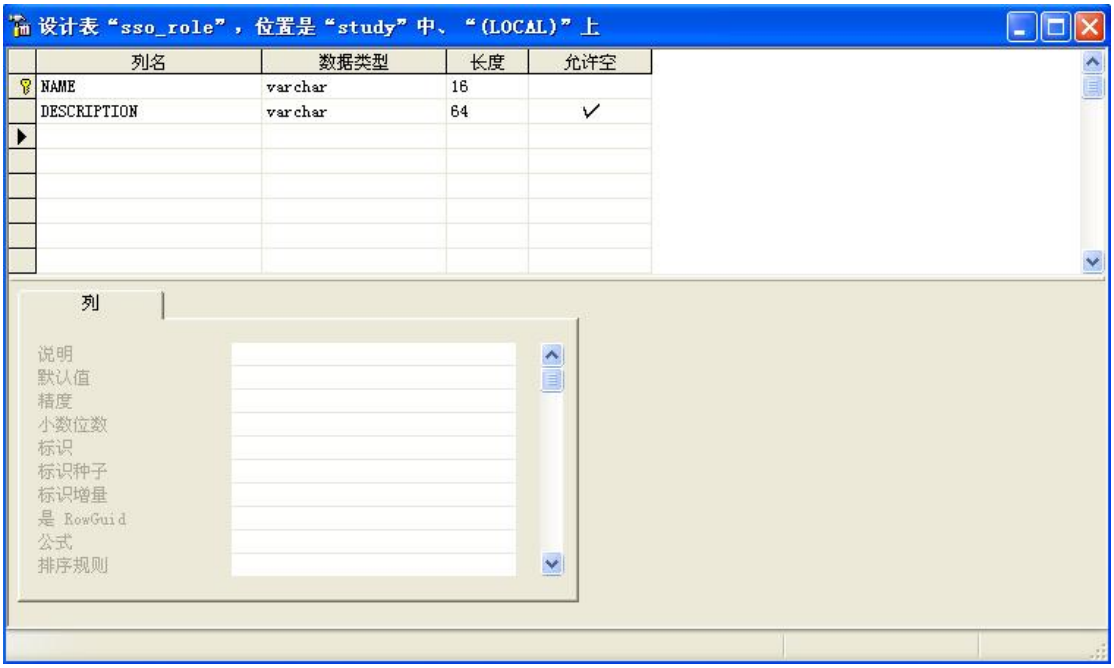
http://www.apache.org/dist/jakarta/tomcat-5/v5.5.9/bin/jakarta-tomcat-5.5.9-admin.zip

把 admin.zip 中的内容解压后复制到 tomcat5.5.9 文件夹下对原文件进行覆盖即可。

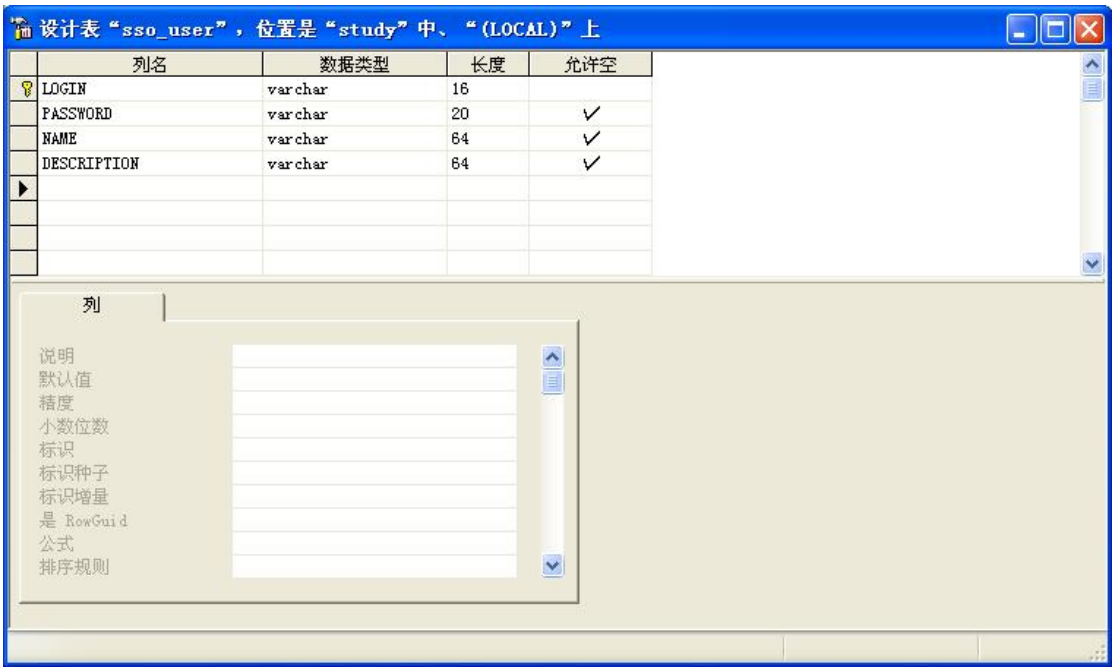
6.2 数据库配置

在这里我们使用 SQL Server，首先我们新建一个数据库 study，然后在数据库中新建三个表 ： sso_role、sso_user、sso_user_role

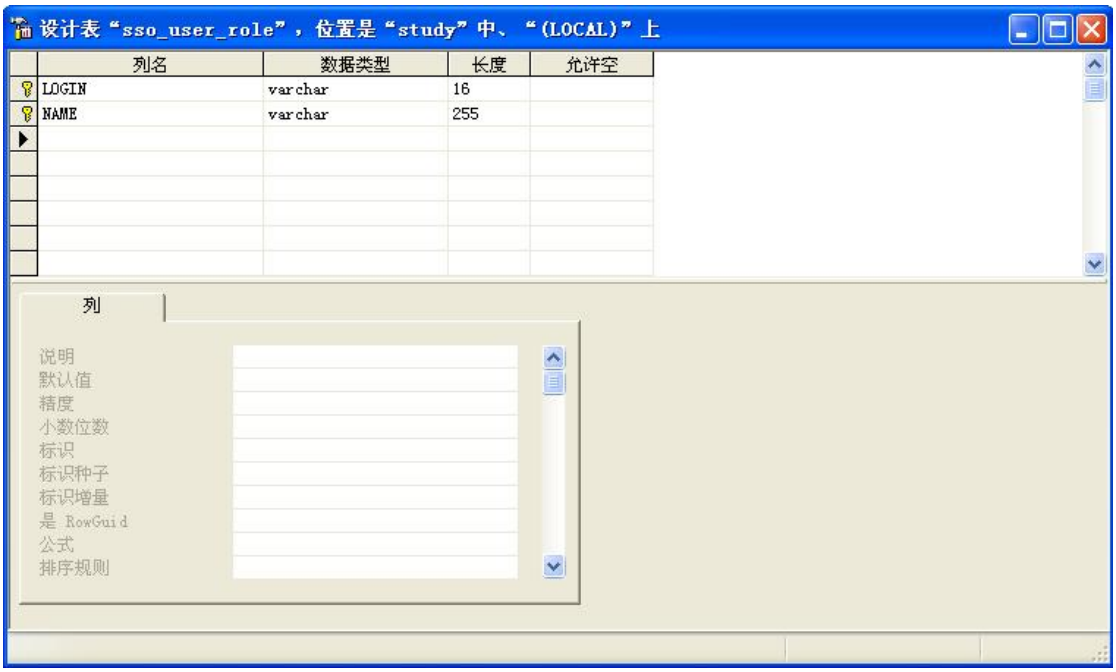
sso_role 表设计如下：



sso_user 表设计如下：

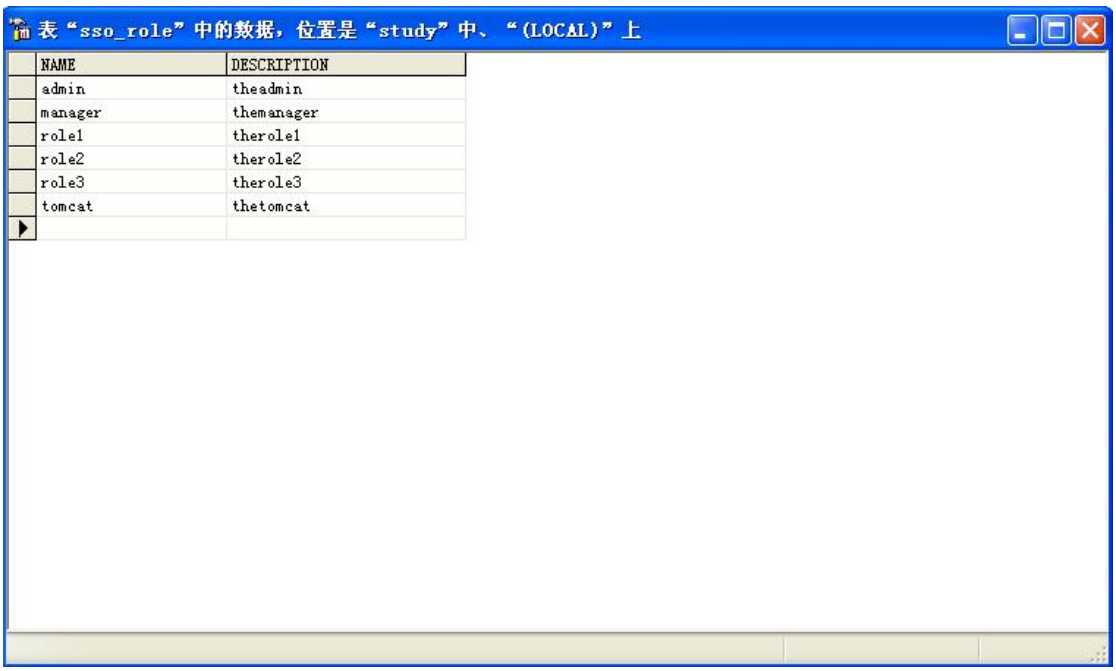


sso_user_role 表设计如下：



随后在这些表里填入一些数据就可以，如下所示：

sso_role 表数据如下：



sso_user 表数据如下：

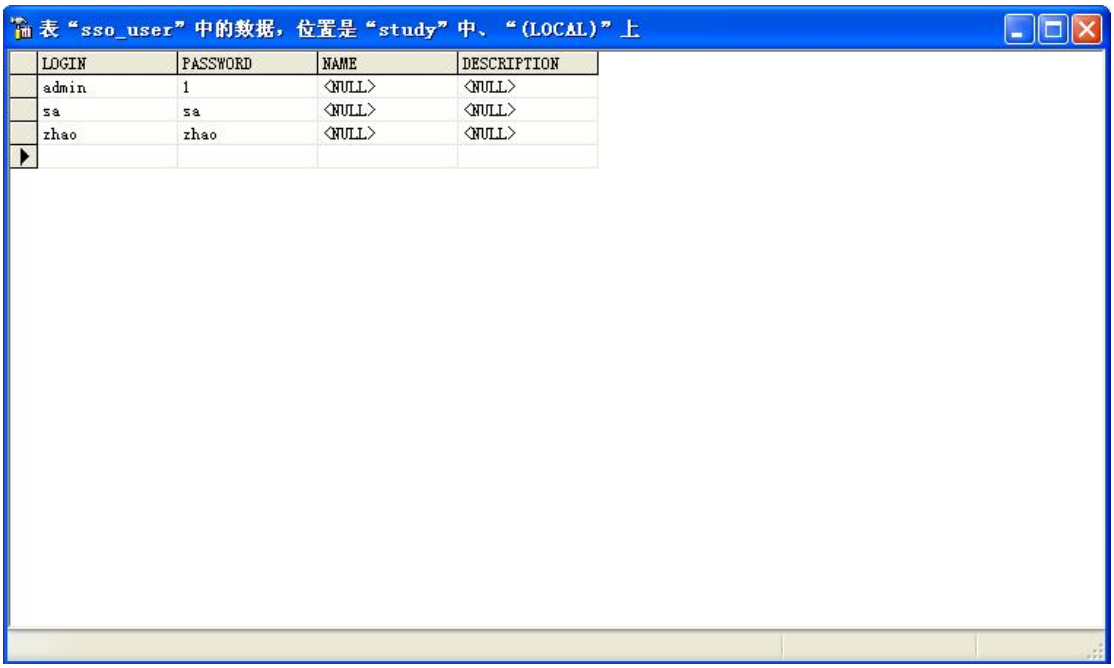


表 "sso_user" 中的数据，位置是 "study" 中、 "(LOCAL)" 上

LOGIN	PASSWORD	NAME	DESCRIPTION
admin	1	<NULL>	<NULL>
sa	sa	<NULL>	<NULL>
zhao	zhao	<NULL>	<NULL>

sso_user_role 表数据如下：

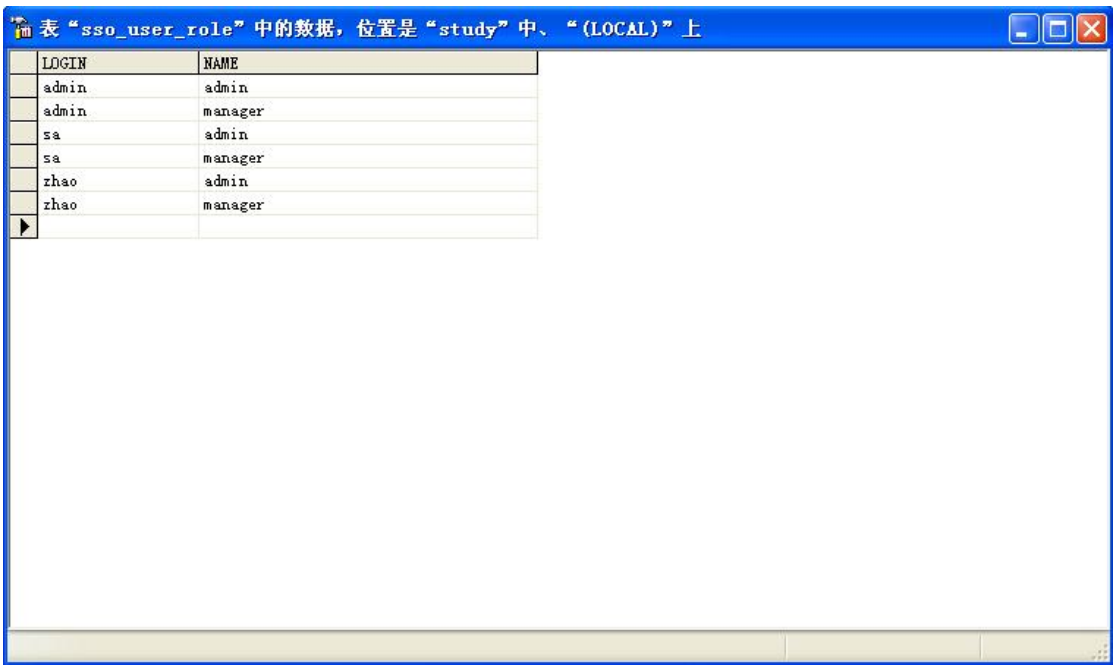


表 "sso_user_role" 中的数据，位置是 "study" 中、 "(LOCAL)" 上

LOGIN	NAME
admin	admin
admin	manager
sa	admin
sa	manager
zhao	admin
zhao	manager

这样我们要用的数据库就配置好了。

6.3 Server.xml 文件配置

进入 D:\Test\tomcat\conf 目录下，打开 server.xml 文件，配置如下：

```
<Realm  className="org.apache.catalina.realm.JDBCRealm"
  driverName="com.microsoft.jdbc.sqlserver.SQLServerDriver"
  connectionURL="jdbc:microsoft:sqlserver://localhost:1433;DataBaseName=study"
  connectionName="sa" connectionPassword="sa"
  userTable="sso_user" userNameCol="login" userCredCol="password"
```

```
userRoleTable="sso_user_role" roleNameCol="name" />
```

（注意：去掉这段话前后的注释）

如果是 Mysql 数据库的话配置如下：

```
<Realm  className="org.apache.catalina.realm.JDBCRealm"
        driverName="com.mysql.jdbc.Driver"
        connectionURL="jdbc:mysql://localhost:3306/brain"
        connectionName="root" connectionPassword="root"
        userTable="sso_user" userNameCol="login" userCredCol="password"
userRoleTable="sso_user_role" roleNameCol="name" />
```

6.4 Tomcat-users.xml 文件配置

集成单点登录 tomcat 的 tomcat-user.xml 文件内容如下：

```
<!--
  NOTE:  By default, no user is included in the "manager" role required
  to operate the "/manager" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary.
-->
<tomcat-users>
  <user name="tomcat" password="tomcat" roles="tomcat" />
  <user name="role1"  password="tomcat" roles="role1"  />
  <user name="both"   password="tomcat" roles="tomcat,role1" />
</tomcat-users>
```

而没有集成单点登录 tomcat 的 tomcat-user.xml 文件内容如下：

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <user username="xujianhua" password="xujianhua" roles="admin,manager"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
```

</tomcat-users>

好了，所有的配置几乎已经完成，下面来看看我们的结果，启动 tomcat，在浏览器中输入 http://localhost:8080 然后我们输入用户名和密码(我们刚才在数据库中建的两个测试帐号和密码：sa、zhao ）只要我们登录一次，再进入其他的模块就不用重新登录了。这就是单点登录技术给我们带来的方便之处。哦，差点忘了，如果我们用 SQL server 的话，我们要把如下三个驱动程序包（msbase.jar、mssqlserver.jar、msutil.jar）拷贝到 jre\jre\lib\ext 目录下，同样如果用 Mysql 的话，要把 Mysql 的驱动程序包拷贝到这个文件夹下。如果没有这些驱动程序包的话，tomcat 启动的时候会报错。

下面是网上摘录的一些关于单点登录资料：

一旦你设置了 realm 和验证的方法，你就需要进行实际的用户登录处理。一般说来，对用户而言登录系统是一件很麻烦的事情，你必须尽量减少用户登录验证的次数。作为缺省的情况，当用户第一次请求受保护的资源时，每一个 web 应用都会要求用户登录。如果你运行了多个 web 应用，并且每个应用都需要进行单独的用户验证，那这看起来就有点像你在与你的用户搏斗。用户们不知道怎样才能把多个分离的应用整合成一个单独的系统，所有他们也就不知道他们需要访问多少个不同的应用，只是很迷惑，为什么总要不停的登录。

Tomcat 4 的"single sign-on"特性允许用户在访问同一虚拟主机下所有 web 应用时，只需登录一次。为了使用这个功能，你只需要在 Host 上添加一个 SingleSignOn Valve 元素即可，如下所示：

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn"
debug="0"/>
```

在 Tomcat 初始安装后，server.xml 的注释里面包括 SingleSignOn Valve 配置的例子，你只需要去掉注释，即可使用。那么，任何用户只要登录过一个应用，则对于同一虚拟主机下的所有应用同样有效。

使用 single sign-on valve 有一些重要的限制：

1、 value 必须被配置和嵌套在相同的 Host 元素里，并且所有需要进行单点验证的 web 应用（必须通过 context 元素定义）都位于该 Host 下。

2、 包括共享用户信息的 realm 必须被设置在同一级 Host 中或者嵌套之外。

3、 不能被 context 中的 realm 覆盖。

4、 使用单点登录的 web 应用最好使用一个 Tomcat 的内置的验证方式（被定义在 web.xml 中的<auth-method>中），这比自定义的验证方式强，Tomcat 内置的的验证方式

包括 basic、digest、form 和 client-cert。

5、 如果你使用单点登录，还希望集成一个第三方的 web 应用到你的网站中来，并且这个新的 web 应用使用它自己的验证方式，而不使用容器管理安全，那你基本上就没招了。你的用户每次登录原来所有应用时需要登录一次，并且在请求新的第三方应用时还得再登录一次。当然，如果你拥有这个第三方 web 应用的源码，而你又是一个程序员，你可以修改它，但那恐怕也不容易做。

6、 单点登录需要使用 cookies。

7. 附件上传

8. 分页

8.1 第一种方案

该分页组件的下载网址是：

https://sdlcweb3d.sun.com/ECom/EComActionServlet;jsessionid=B848250011ABC35FBA032A23F87A5A39

下载后解开压缩包文件 jsf-1_1.zip。 进入该目录：jsf-1_1\samples\components\src 并复制该目录下 Components 文件夹到自己所建项目的 src 目录下，然后再在 src 目录下新建一个文件夹 util。把目录 jsf-1_1\samples\components\src\demo\model 下的 ResultSetBean.java 文件拷贝到 util 文件夹下。

其次、我们还要在 faces-config.xml 文件中进行一些配置，如下：

组件注册：（主要是为了在页面中使用它的标签）

```
<component>
    <component-type>Scroller</component-type>
    <component-class>
        components.components.ScrollerComponent
    </component-class>

    <component-extension>
        <component-family>Scroller</component-family>
    </component-extension>
</component>
```

Bean 注册：

```
<managed-bean>
    <managed-bean-name>ResultSetBean</managed-bean-name>
    <managed-bean-class>com.util.ResultSetBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

最后、我们把分页页面做成一个单独的页面，方便统一调用。例如我们在 WebRoot 目录下新建

一个文件夹 util，分页的文件名就叫做 resultSet.jsp。内容如下：

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsf/demo/components" prefix="d"%>
<d:scroller navFacetOrientation="SOUTH" for="table"
    actionListener="#{ResultSetBean.processScrollEvent}">
    <f:facet name="previous">
        <h:outputText value="上一页 "/>
    </f:facet>
    <f:facet name="next">
        <h:outputText value="下一页"/>
    </f:facet>
    <f:facet name="number">
    </f:facet>
    <f:facet name="current">
    </f:facet>
</d:scroller>
```

在我们的 datatable 中的格式如下：

```
<h:dataTable var="data" rows="3" id="table" value="#{studentinfo.info}"
binding="#{ResultSetBean.data}">
```

注：此处的 id="table"中的 table 要与 resultSet.jsp 文件中 for="table"中的 table 同名。

好了，这样我们的分页就完成了。

8.2 第二种方案

这种方案是直接使用 jsf-1_1\samples 目录下的 demo-components.jar 文件，把该压缩文件拷贝到 WebRoot\WEB-INF\lib 目录下，而不需要把 jsf-1_1\samples\components\src 目录下 Components 文件夹复制到自己所建项目的 src 目录，剩下和上面唯一的区别就是 faces-config.xml 文件中不需要对组件进行注册。因为该 jar 文件中已经进行了详细的配置。

9. Javascript 非空验证

比如首页登录页面的验证：有两个字段用户名（userName）和密码(password)，他们对应的 JSF 代码分别如下：

```
<f:view>
<h:form onsubmit="return check(this.form)" id="LoginForm">
<h:inputText id="userName" value="#{LoginBean.userName}"  maxlength="20" size="24"/>
<h:inputSecret id="password" value="#{LoginBean.password}" maxlength="20" size="24"/>
</h:form>
</f:view>
```

验证的 javascript 代码如下所示：

```
<script type="text/javascript">
<!--
    function check(){
        var name=LoginForm["LoginForm:userName"].value;
        var password=LoginForm["LoginForm:password"].value;
        if(name==""){
            alert('用户名不允许为空');
            LoginForm["LoginForm:userName"].focus();
            return false;
        }else if(password==""){
            alert('密码不允许为空');
            LoginForm["LoginForm:password"].focus();
            return false;
        }else if(LoginForm["LoginForm:code"].value==""){
            alert('验证码不允许为空');
            LoginForm["LoginForm:code"].focus();
            return false;
        }else{
            sending.style.visibility="visible";
            return true;
        }
    }
//-->
</script>
```

10. 弹出菜单

10.1 页面

```
<h:outputLink
onclick="window.open('modify.jsf?userid=#{data.id}','','menubar=0,too
lbar=0,directories=0,resizable=0,location=0,status=0,scrollbars=0,wid
th=400,height=250');return false">
<h:outputText value="修改密码"></h:outputText>
</h:outputLink>
```

注：#{data.id}是向弹出页面传递的参数

10.2 对话框

在 bean 中定义一个 String 变量：private String script="";

然后是它的 get、set 方法，如下：

```
public String getScript() {
```

```
            return script;
}
public void setScript(String script) {
            this.script = script;
}
```

页面中调用该 bean 中的方法，script 在该方法中表示如下：

```
public void save(){
        try{
            Session session=HibernateSessionFactory.currentSession();
            Transaction ta=session.beginTransaction();
            Student mm=new Student();
            mm.setName(this.getUsername());
            mm.setPassword(this.getUserpassword());
            mm.setSex(this.getUsersex());
            mm.setCity(this.getUsercity());
            mm.setMemo(this.getUsermemo());
            session.save(mm);
            ta.commit();
            HibernateSessionFactory.closeSession();
            this.setScript("<script>alert('增加成功')</script>");
            this.setUsername("");
            this.setUserpassword("");
            this.setUsersex("");
            this.setUsercity("");
            this.setUsermemo("");
            }catch(Exception e ){
            this.setScript("<script>alert('增加失败')</script>");
            e.printStackTrace();
            }
}
```

JSF 页面中如下所示：

```
<f:view>
   <h:form id="studentForm">
   --------------------
        <h:outputText escape="false" value="#{student.script}"/>
        <h:commandButton value="确定"  action="#{student.save}"/>
   --------------------
    </h:form>
</f:view>
```

11. 日期时间控件

12. 拦截模式

13. 联动菜单

14. 验证码

15. MD5 加密

16. 在线编辑

17. 邮件外发

18. 项目发布

18.1 一个 tomcat 单个工程

我们的目的是：在浏览器中输入 http://localhost:8080 定位到自己想要到达的页面：

进入 tomcat 的 conf 目录，打开 server.xml 文件，我们在该文件中会看到如下一些语句：

```
<Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true"
        xmlValidation="false" xmlNamespaceAware="false">
```

我们只要在该语句下面加入如下一段话，

```
<Context  path=""  docBase="${catalina.home}/webapps/oaweb"  debug="0"  reloadable="true"></Context>
```

另外，打开 conf 目录下的 web.xml 文件，我们在该文件中会看到如下一些语句：

```
<welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
```

我们只要加入如下一条语句即可：<welcome-file>portal\test.jsp</welcome-file>,修改后效果如下

```
<welcome-file-list>
        <welcome-file> portal\test.jsp </welcome-file>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
```

这样修改以后，当我们发布我们工程的时候，客户只要在浏览器中输入 http://localhost:8080 就能立即定位到 portal 文件夹下的 test.jsp 文件。当然，我们还可以配置很多的虚拟主机：

```
<Host name="localhost" appBase="webapps"
     unpackWARs="true" autoDeploy="true"
     xmlValidation="false" xmlNamespaceAware="false">
     <Context  path=""  docBase="${catalina.home}/webapps/oaweb"  debug="0"
reloadable="true"></Context>
</Host>
```

注：上面的 server.xml 文件中的内容也可以配置在 conf/下的 oaweb.xml 文件中，最后效果是一样的。

特别注意:如果 server.xml 文件主机的配置中没有

```
<Context  path=""  docBase="${catalina.home}/webapps/oaweb"  debug="0"  reloadable="true"></Context>
```

的话，当输入 http://localhost:8080 时，系统会自动查找 webapps\ROOT 目录下的 index.jsp 文件，相反，如果主机中进行了上面的配置，当输入 http://localhost:8080/oaweb(或者 http://localhost:8080)时，系统会在 oaweb 目录下进行如下顺序的查找，index.html、index.htm、index.jsp。这些都是在 conf/web.xml 文件中设置的。以此类推，当 server.xml 文件主机的配置如下：

```
<Context  path=""  docBase="${catalina.home}/webapps/oaweb/portal"  debug="0"  reloadable="true"></Context>
```

当输入 http://localhost:8080/oaweb/portal(或者 http://localhost:8080)时,系统会在 portal 目录下进行如下顺序的查找，index.html、index.htm、index.jsp.如果有的话，就显示查找到的页面，如果没有查找到的话，就把文件夹下的文件目录显示出来。

最后,绝对路径:docBase="${catalina.home}/webapps/oaweb" 也可以配置成相对目录:docBase="/oaweb"

或 docBase="oaweb"。两者都可以。

18.2 一个 tomcat 多个工程

注意：在开发阶段最好不要进行此项配置，以免影响开发效率。最多就是在工程的一级目录下面新建一个 index.html 文件，也就是下面所说的第一条。其他任何东西都不需要配置。

首先、index.html 文件的配置

在每个工程的一级目录下新建一个 index.html 文件.形式是：oa(工程名)/index.html、mis（工程名）/index.html.文件内容如下：

```
<html>
  <head>
     <meta http-equiv="Refresh" content= "0; URL=index.jsf"/>
  </head>
</html>
```

其次、tomcat\conf 目录下 server.xml 文件的配置：如下图

像这样就可以在一个 tomcat 中任意配置多个项目，并且使用不同的端口号：下图中配置了两个项目，分别使用不同的端口：8070 和 8060

解释如下：

一、当我们在 IE 浏览器中输入 Http://localhost:8070 的时候，tomcat 服务器会根据 Server.xml 文件自动查找 oaweb 项目下的 index.html 文件。当程序定位到 index.html 时，会自动跳转到此工程中你所要指定的文件：index.faces。

二、当我们在 IE 浏览器中输入 Http://localhost:8060 的时候，tomcat 服务器会根据 Server.xml 文件自动查找 health 项目下的 index.html 文件。当程序定位到 index.html 时，会自动跳转到此工程中你所要指定的文件：index.jsf。

以次类推：我们可以在一个 tomcat 中配置更多的工程。并分别 不同的端口号

试一试：把上面的 index.html 文件内容修改后，在对比下效果

```
<html>
  <head>
     <meta http-equiv="Refresh" content= "0; URL=../index.jsf"/>
  </head>
</html>
```

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----
<Server port="8005" shutdown="SHUTDOWN">
    <Listener
        className="org.apache.catalina.mbeans.ServerLifecycleListener" />
    <Listener
        className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
    <Listener
        className="org.apache.catalina.storeconfig.StoreConfigLifecycleListener" />
    <GlobalNamingResources>
        <Environment name="simpleValue" type="java.lang.Integer"
            value="30" />
        <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"
            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
    </GlobalNamingResources>

<!-- 华盛迈杰公司OA系统-->
    <Service name="Hsmj">
        <Connector port="8070" maxHttpHeaderSize="8192" maxThreads="150"
            minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
            redirectPort="8443" acceptCount="100" connectionTimeout="20000"
            disableUploadTimeout="true" />
        <Connector port="8009" enableLookups="false" redirectPort="8443"
            protocol="AJP/1.3" />
        <Engine name="Hsmj" defaultHost="localhost">
            <Realm
                className="org.apache.catalina.realm.UserDatabaseRealm"
                resourceName="UserDatabase" />
            <Host name="localhost" appBase="webapps" unpackWARs="true"
                autoDeploy="true" xmlValidation="false" xmlNamespaceAware="false">
                <Context path=""
                    docBase="${catalina.home}/webapps/oaweb" debug="0"
                    reloadable="true">
                </Context>

            </Host>

        </Engine>

    </Service>
<!-- 中国骨质疏松网系统-->

    <Service name="Baiyang">
        <Connector port="8060" maxHttpHeaderSize="8192" maxThreads="150"
            minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
            redirectPort="8443" acceptCount="100" connectionTimeout="20000"
            disableUploadTimeout="true" />
        <Connector port="8009" enableLookups="false" redirectPort="8443"
            protocol="AJP/1.3" />
        <Engine name="Baiyang" defaultHost="localhost">
            <Realm
                className="org.apache.catalina.realm.UserDatabaseRealm"
                resourceName="UserDatabase" />
            <Host name="localhost" appBase="webapps" unpackWARs="true"
                autoDeploy="true" xmlValidation="false" xmlNamespaceAware="false">
                <Context path=""
                    docBase="${catalina.home}/webapps/health" debug="0"
                    reloadable="true">
                </Context>

            </Host>

        </Engine>

    </Service>
<!-- 其它系统-->

</Server>

```

19. 模式应用

单件模式，如下：

```
public class Singleton {

    private Singleton(){}

    //注意这是 private 只供内部调用

    private static Singleton instance = new Singleton();

    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
        return instance;
     }
}
```

20. Hibernate 连接多个数据库

比如：一个工程同时连接到三个数据库，那么我们在 src 目录下新建三个文件夹 A、B、C .

然后在每个目录下新建 SessionFactory.java 和 hibernate.cfg.xml 两个文件，如下：

A 文件夹：A_SessionFactory.java 和 A_hibernate.cfg.xml

B 文件夹：B_SessionFactory.java 和 B_hibernate.cfg.xml

C 文件夹：C_SessionFactory.java 和 C_hibernate.cfg.xml

注意：关键是 SessionFactory.java 里的配置，分别如下：

```
private static String CONFIG_FILE_LOCATION = "/A/A_hibernate.cfg.xml";

private static String CONFIG_FILE_LOCATION = "/B/B_hibernate.cfg.xml";

private static String CONFIG_FILE_LOCATION = "/C/C_hibernate.cfg.xml";
```

总结：其实 SessionFactory.java 和 hibernate.cfg.xml 两个文件可以随便放在那个文件夹里，关键是要注意 SessionFactory.java 类的包名和 hibernate.cfg.xml 文件的路径。