JAVA 开发利器 My Eclips

全面详解

吾尝终日而思矣，不如须臾之所学也；吾尝跂而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫（船桨）者，非能水也，而绝江河。君子生非异也，善假于物也。

荀子《劝学》

关于版权

为什么学习 MyEclipse

Java 的开发中可以有多种开发方式，最基础的可以使用记事本，也可以使用智能集成开发工具。在理论上，不管使用什么工具，都是可以进行项目开发的，但是开发速度相差是非常大的。

MyEclipse 做为一款集成开发工具，大大节省了程序员的工作范围，能够很大程度的提高工作效率，从而使项目开发更快捷。

目前开发中可以选择的集成开发工具有很多，但是使用最多的还是 MyEclipse，在绝大部分的开发公司中都是使用它。所以是否能够熟练使用 MyEclipse 决定了是否能够很好的融入开发团队的关键。

为什么要写这本书

在开发团队中有这样两种人，一种是对技术了解的非常精通，但是开发速度很慢；另一种是技术一般，但是能够快速、准确的完成下达的任务；请问项目经理会更欣赏哪一个程序员？

MyEclipse 的功能是非常强大的，但是笔者在培训教学中，发现学员并没有使用到其中的精髓功能，还是使用记事本开发的思想。这就好像下面这个故事：

现在有木柱和细木棍两件东西，需要使用它们制作出火。有些人可能直接进行钻木取火，但是有些人可能会使用它们换一个打火机。

其中钻木取火就是 MyEclipse 的基本功能，而换打火机就是 MyEclipse 的精髓功能，在本书中就主要对这些精髓功能进行讲解。通过 MyEclipse 的精髓功能就能够大大提高程序员的工作效率，使用它们甚至可以自动生成 150 行的代码。如果手动编写 150 行的代码，可能需要花费 1 个小时的时间，但是如果使用其中的功能，就只需要 5 秒钟。

笔者写这本书的目的，就是尽可能的让 MyEclipse 完成它能够完成的工作，只将必要的工作交给程序员完成，从而加快开发。

本书有哪些特点

1、步骤清晰 表达准确

对于一个非初学者来说，本书的步骤可能有些繁琐。例如描述弹出菜单时，其中会这样讲解：在编辑区的程序中，将鼠标指定在 Student 类体中，单击鼠标右键，在弹出的菜单中选择 XXX 命令。在其中的讲解中，有几个关键点，分别是编辑区、类体中、单击右键，其

2、全面讲解 不留盲点

这里所说的全面讲解，并不是指所有功能，而是指一个功能。在很多书中，说的最多的一句话就是这些选项采用默认值就可以。在本书中，这句话使用的是比较少的，因为MyEclipse 中不会集成完成不需要改变的选项。

在本书中对操作界面进行讲解中，会将操作界面中的每一个选项都讲解到。在其中讲解该选项的作用，在什么地方用，实际开发中如何来用，尽量不让读者带着疑惑进行下面的学习。

3、作者心得 经验之谈

在本书的知识讲解中，穿插了大量的作者心得，其中包括注意点、小技巧和作者的经验之谈。当具有两种都可以用的选择时，作者通常给出自己的观点或者目前主流选择，可能和读者所在公司的选择不同，这种情况下以公司的选择为标准。

5、视频讲解 结合实践

本书编写后，笔者为本书录制了随书视频，在视频中涵盖书中知识点，并高于书中讲解。在对书中的知识点讲解后，还以 5 个综合案例讲解了如何使用 MyEclipse 进行开发。读者可以在光盘中中找到这些内容。

本书适合哪些读者

对 Java 基本了解，想进一步加快开发速度的程序员

各大中专院校、培训机构的在校学生和相关授课老师

想加快工作效率的开发团队

编程爱好者

本书作者:

2011 年 7 月

本书是一本 MyEclipse 的精装教程，目前 Java 语言的编程工具中最流行的就是 MyEclipse 集成开发工具。本书以开发技巧和案例为主线，对 MyEclipse 的重要开发功能进行讲解。全书分为 3 部分。第 1 部分是基础篇，讲解了 MyEclipse 开发环境的搭建和基本操作命令，通过使用基本操作命令就可以让 MyEclipse 自动生成程序员想要的代码。

第 2 部分是开发篇，讲解了目前最主流的 Java 项目开发所使用的框架，包括 Struts、Hibernate、Spring 和 Struts 2 框架，在其中主要讲解了 MyEclipse 中对这些框架集成的开发功能。

第 3 部分是应用篇，讲解了 Java 项目开发中将要使用到的某些技术，例如在开发之前要使用 UML 进行系统分析等工作；在开发工程中对程序进行调试和测试，开发完成后使用 CVS 或者 SVN 对项目和程序进行版本控制。通过本书的知识点，读者就可以全面了解使用 MyEclipse 工具进行 Java 项目开发中每一步骤的操作。

第1章　开发环境的搭建

从本章起，就进入我们的 MyEclipse 的开发之旅。MyEclipse 是一款强大的、智能的集成开发工具，在理论上只需要安装 MyEclipse 就可以进行日常的开发，因为在完整的 MyEclipse 中自带来 JRE、数据库、应用服务器等工具。但是在实际开发中，这些集成的工具并不完全适合的，所以 MyEclipse 还需要借助外部的一些工具和开发环境，在本章中就可以学习一下如何搭建优秀的 MyEclipse 开发环境。

1.1　下载和安装 JDK

JDK 是 Java 程序开发最核心的工具，通过 JDK 可以完成对 Java 程序的编译和运行，所以我们首先要安装 JDK。目前 JDK 的最新版本为 7.0，但是还处于测试阶段，如果真正用于开发，它不是稳定的，所以这里我们现在目前稳定的 JDK 6.0 版本。

1.1.1　下载 JDK

下载 JDK 可以从 SUN 公司的官方网站进行下载，官方首页地址为“http://java.sun.com”。进入首页后，在横向导航栏中，单击“Downloads”菜单，然后在其中选择“Java SE”选项，将跳转到选择 JDK 下载的页面，如图 1-1 所示。

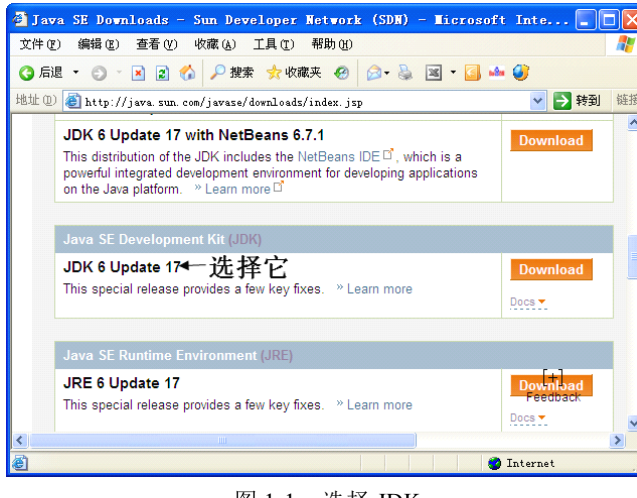


图 1-1　选择 JDK

在该页面中，有多个应用在不同环境中的 JDK 版本，这里我们选择仅仅用于 Java SE 开发的“JDK 6 Update 17”。单击后面的“Download”按钮，将进入选择操作系统版本的下载页面，如图 1-2 所示。

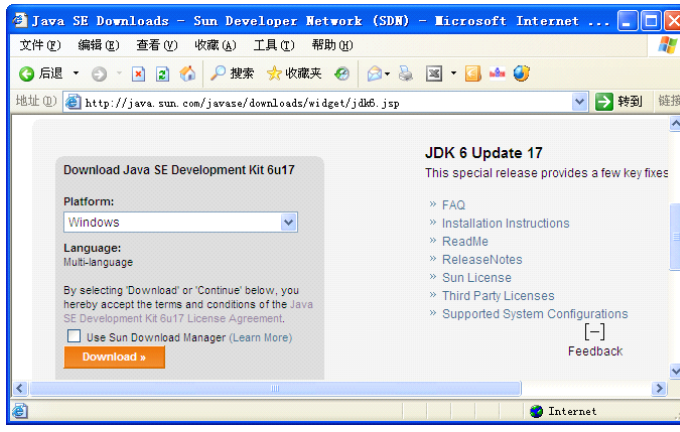


图 1-2　选择操作系统

在其中“Platform”下拉列表中选择“Windows”选项，它就表示下载安装在 Windows 操作系统下的 JDK，单击下面的“Download”按钮，将进入下载页面，如图 1-3 所示。

说明：Java 最大的特点就是能够应用在不同操作系统中，所以 JDK 肯定也是这样的，从“Platform”下拉列表中也可以看到这一点，其中包括 Windows、Linux 和 Solaris 等多种操作系统。

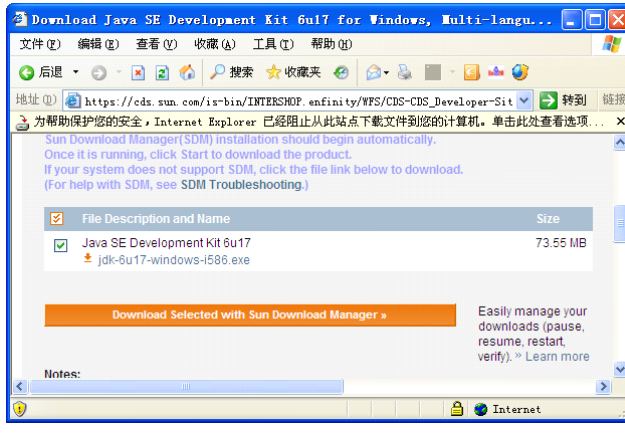


图 1-3　下载页面

在该页面中，单击“jdk-6u17-windows-i586.exe”超链接，将弹出下载对话框，单击“保存”按钮，就可以选择保存到本地硬盘中。在“jdk-6u17-windows-i586.exe”超链接上单击右键，也可以在弹出菜单中选择下载工具进行下载。

1.1.2　安装 JDK

下载完成后，就可以安装 JDK，Windows 操作系统中的 JDK 安装是非常简单的。双击下载的“jdk-6u17-windows-i586.exe”文件，将首先打开许可证协议确认界面，在其中单击“接受”按钮，将弹出自定义安装界面，如图 1-4 所示。

图 1-4　安装 JDK

其中列表中给出了可选安装的功能，默认是全部选中的，可以单击前面的按钮取消安装，这里保持默认值就可以。界面下方是给出的安装位置，单击“更改”可以安装到其他地方。

注意：本地安装时就是安装在默认位置下，读者修改位置后一定要记住具体位置，因为在后面的学习中将多次使用到它。并且 JDK 路径中不要有中文，不然后面的开发中可能会出现不能识别的问题。

单击“下一步”按钮，将开始安装 JDK。经过数分钟后 JDK 完成安装，将自动打开 Java 运行时环境自定义安装界面。Java 运行时环境配置和 JDK 配置完全相同。配置完成后单击“下一步”按钮进入 Java 运行时环境安装界面。出现安装完成界面后，单击“完成”按钮，将彻底完成 JDK 的安装。

说明：JDK 安装完成后，如果要使用控制台命令，进行开发，还需要进行必要的配置。但是在本书中是要借助 MyEclipse 开发，所以是不需要这一步的。

1.2　下载和安装 MySQL 数据库

目前开发中主流的数据库有很多，例如 Orcal、DB2、SQL Server 和 MySQL 等。其中 MySQL 数据库体积非常小，速度非常快，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。所以在本书中以 MySQL 数据库为例进行讲解，其他数据库的操作也是类似的。

1.2.1　下载 MySQL 数据库

MySQL 数据库的官方网站地址为“http://www.mysql.com/”，在其中可以单击导航栏中的“Downloads”选项，将进入版本选择页面。目前 MySQL 数据库的最新版本为 5.4 版本，但是还是测试版本，所以这里我们选择 5.1 版本。单击对应的超链接，将进入 MySQL 5.1 下载页面，然后在其中单击“MySQL Community Server”下的“Download”

按钮，将跳转到选择操作系统版本的页面，如图 1-5 所示。

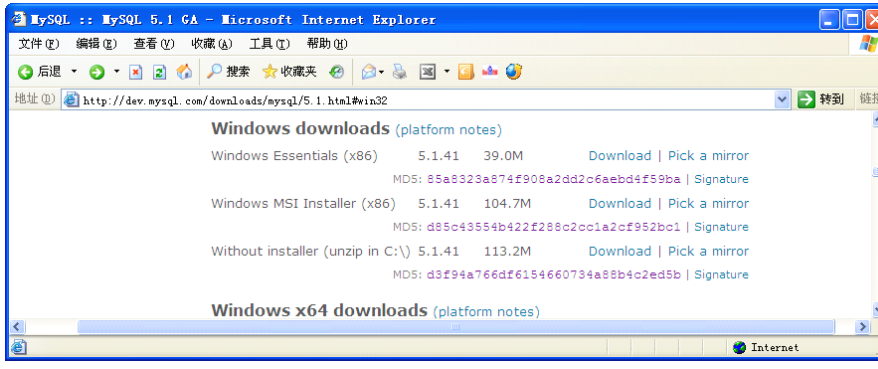


图 1-5　下载 MySQL 数据库

在其中选择“Windows”操作系统后，其中出现三种形式的版本，其中包括安装和直接解压就可用的。这里选择“Windows Essentials (x86)”安装版本，单击后面的“Download”按钮，将开始下载该版本的 MySQL 数据库。

如果单击“Pick a mirror”超链接将进入注册 MySQL 数据库用户的页面，在其中可以单击“No thanks, just take me to the downloads!”超链接跳过该步，然后选择某一镜像进行下载。

1.2.2　安装 MySQL 数据库

下载 MySQL 数据库后，就可以来安装它。双击下载的“mysql-essential-5.1.41-win32.msi”文件，将首先弹出选择安装类型界面，如图 1-6 所示。

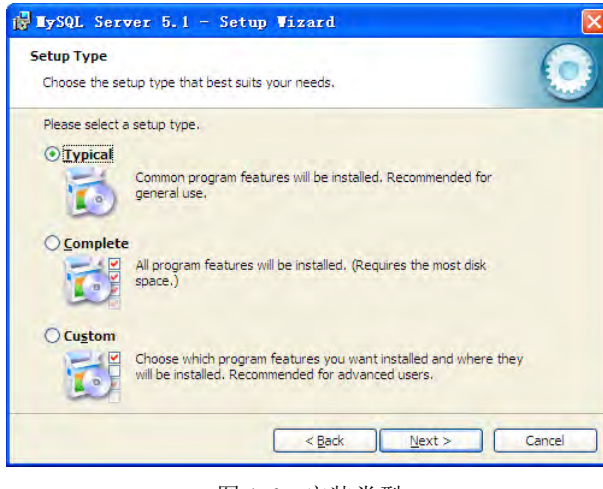
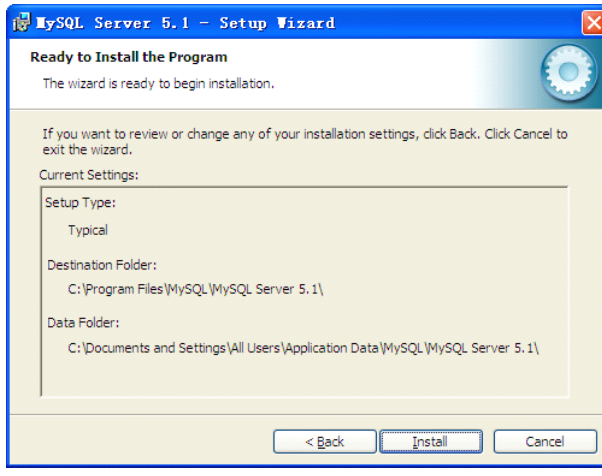


图 1-6　安装类型

其中“Typical”表示典型版本，通常就选择该选项，单击“Next”按钮，将进入安装界面，如图 1-7 所示。



题 1-7　安装界面

其中固定了 MySQL 的安装位置，是自动安装在 C 盘下，单击“Install”按钮，将开始安装 MySQL 数据库。

安装结束后，将出现是否配置服务器的界面，在其中选择“Configure the MySQL server now”（立即配置服务器）复选框。单击“Next”按钮后，可开始设置数据库服务器相关选项。

在“MySQL Server Instance Configuration”配置中选择“Standard Configuration”（标准设置）单选按钮。单击“Next”按钮后，将进入设置服务器选项的界面。在设置服务器选项界面中选择“Install As Windows Service”（安装为 Windows 服务）和“Include Bin Directory in Windows PATH”（添加 Bin 路径到 PATH 环境变量）两个复选框，单击“Next”按钮后，将弹出设置管理员密码的界面。

设置管理员密码是非常重要的，读者一定不要忘记自己设置的密码，在后面学习数据库操作和 Hibernate 时，将多次使用到该密码。单击“Next”按钮，系统将结束安装向导，安装 MySQL 数据库完成。

1.3　下载和安装 Tomcat 服务器

目前主流的应用服务器也是非常多的，例如 JBOSS、WebLogic、Tomcat 等。Tomcat 服务器是由 Apache 开源组织开发并维护的，能够支持 JSP 和 Servlet，以及各种开源框架的开发使用，而且 Tomcat 服务器是免费产品，并且提供了其源代码。所以在 Java Web 开发中，经常使用 Tomcat 服务器来进行操作。

下载 Tomcat 服务器可以通过 Apache 开源组织的官方网站进行下载，它的地址为“http://www.apache.org”。在该网站主页右侧导航栏中有很多该开源组织开发的项目，在其中单击“Tomcat”超链接，将进入 Tomcat 项目的首页。

在 Tomcat 项目首页的左侧 Download 列表中，单击“Tomcat 6.x”超链接，将进入该版本的下载页面，如图 1-8 所示。

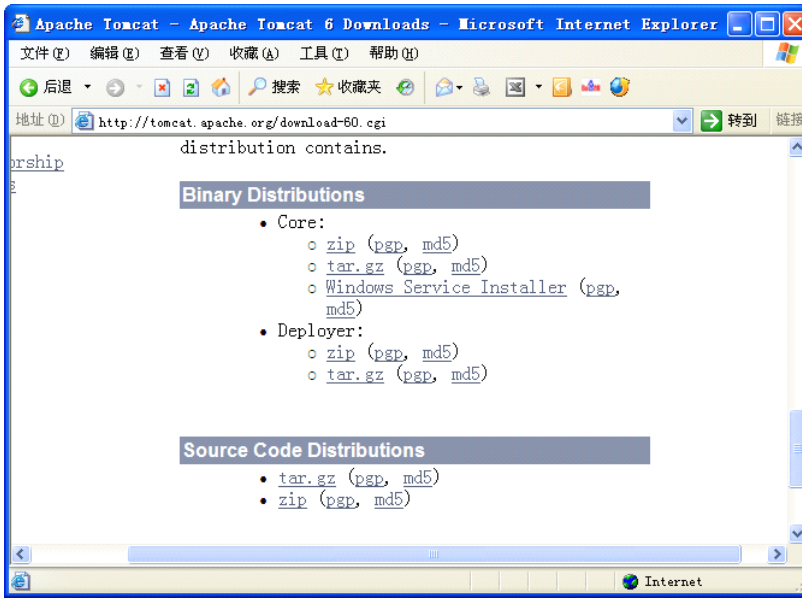


图 1-8　下载 Tomcat

其中“Binary Distributions”下的文件都是编译好的二进制文件，而“Source Code Distributions”下面的文件为 Tomcat 的源代码，这里选择下载编译好的二进制文件。“Binary Distributions”下又分为两类，一类是“Core”即 Tomcat 核心，一类是“Deployer”即 Tomcat 部署文件，这里选择下载“Core”下的文件。“Core”下的文件又分为三种，一种是 zip 格式的，即 Windows 下的压缩文件；一种是 tar.gz 格式，即 Linux 下的压缩文件；一种是 Windows Service Installer，即 Windows 安装文件。

技巧：在下载 Tomcat 时，最好选择 zip 格式，这样直接解压就可以用了，也不需要多余的配置。但是对于例如 MySQL 数据库这样的软件而说，虽然也有解压缩的安装文件，但是最好不要选择，因为后面还会有配置，显得更麻烦。

下载 zip 格式的 Tomcat 后，解压缩后就表示安装完成，可以将它安装在任意目录下，但是一定要记住该目录，在后面 MyEclipse 中还要通过该目录配置 Tomcat。

1.4　MyEclipse 的环境搭建

搭建完最基本的开发工具后，就可以来安装本书最重要的 MyEclipse。在学习 MyEclipse 之前，读者可能接触过 Eclipse。单纯的 Eclipse 只能进行 Java 桌面开发，如果要进行 Java Web 开发，还需要安装 Eclipse 插件。众多软件厂商和开源组织开发了相应的插件，其中以 MyEclipse 插件最为出名并常用。

注意：MyEclipse 的更新在近两年是非常快的，目前已经出现了 8.0 版本，但是发布稳定的最新版本是 7.5。如果读者本地已经安装了 6.0 以上版本的 MyEclipse，完全可以使用原版本的 MyEclipse 进行学习，它们的差别是不大的。

1.4.1 下载和安装 MyEclipse

MyEclipse 的官方网站为“http://www.myeclipseide.com”，因为 MyEclipse 是一款非开源的开发工具，所以只能在官方网站中下载试用版。

进入官方网站后，单击左边导航栏中的“Try/Download”链接，打开下载页面，然后在其中选择“MyEclipse 7.5GA”，如图 1-9 所示。

图 1-9 MyEclipse 7.5GA 下载页面

MyEclipse 7.5GA 的安装分为两种，一种是“All in ONE”版本，一种是“pulse”版本。其中“All in ONE”版本包含 MyEclipse 安装的全部文件，而“pulse”版本为插件安装版本，需要事先安装好了相应的 Eclipse。这里为了安装方便，选择下载其中的“All in ONE”版本。

技巧：MyEclipse 的官方网站经常打不开，可以通过代理服务器进行访问，也可以在网络中进行搜索，查找国内的下载资源进行下载。

下载完成 MyEclipse 后，就可以双击下载文件进行安装。MyEclipse 的安装是非常简单的，它的安装实际上就是将其中的文件解压缩到硬盘中。首先出现的是许可证协议界面，在其中选择“I accept the terms of the license agreement”，单击“Next”按钮，将弹出选择安装目录的界面，默认是安装在 C 盘，也可以安装在其他地方。

后面的操作，只需要单击“Next”按钮，就能够进行安装。MyEclipse 的安装时要花费一定时间的，当出现安装完成页面后，单击“Finish”按钮，将彻底完成 MyEclipse 的安装。

1.4.2 运行 MyEclipse

安装 MyEclipse 后，在桌面上就出现一个启动 MyEclipse 的快捷方式，它的图标为“  ”。双击该快捷方式，将开始运行 MyEclipse，首先出现的将是定义工作空间的界面，如图 1-10 所示。

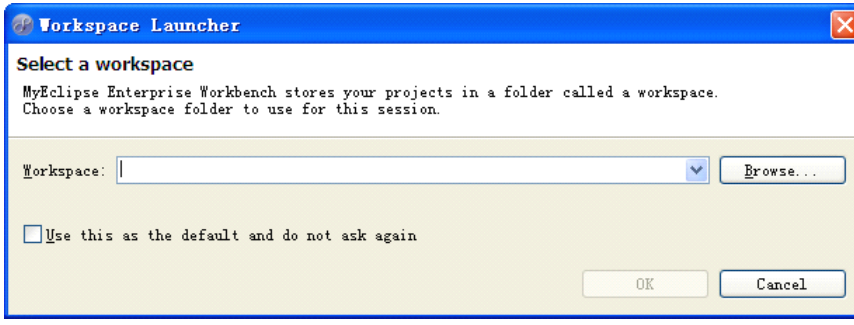


图 1-10　工作空间

工作空间就是用来放置 Java 项目文件的目录，可以任意指定。单击“Browse”按钮，可以指定当前计算机中的目录。下面的“Use this as the default and do not ask again”，表示是否将当前指定的工作空间做为默认空间，当选中该选项后，下次启动 MyEclipse 时就不会提醒该界面，从而以当前设置的工作空间启动。

选择某目录做为工作空间后，单击“OK”按钮，将开始启动 MyEclipse。启动 MyEclipse 是要花费一定时间的，尤其是当计算机内存比较小时。启动 MyEclipse 后，界面如图 1-11 所示。

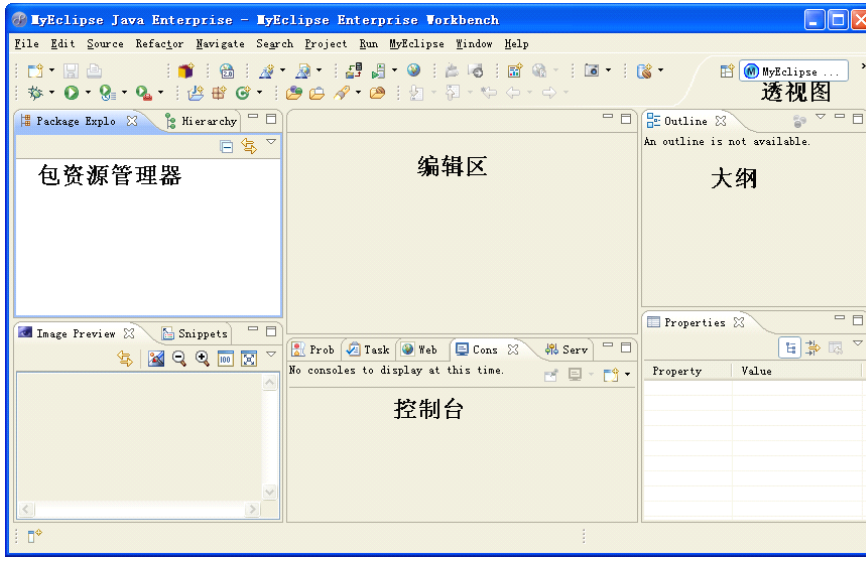
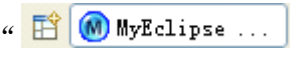


图 1-11　初始界面

在 MyEclipse 的初始启动界面中，将由一个初始布局，它通常被称为透视图。透视图是可以改变的，可以通过右上角的“ 

”按钮，也可以通过“Window”菜单。在后面的讲解中，如果用到其他透视图再详细讲解。

在一个透视图中，由多个视图界面组成，当前 MyEclipse 默认透视图下，就有以后经常用到的“Package Explo”包资源管理器视图界面和“Console”控制台视图界面。

在 MyEclipse 中，除了视图界面外，最上部还有菜单和工具栏，在后面的学习中，

主要是通过它们进行 MyEclipse 的操作。当讲解具体操作时，我们再对它们进行详细的功能讲解。

1.5 开发 HelloWorld 程序

在前面的讲解中，已经将开发环境搭建成功了，并且已经对 MyEclipse 的每一部分有所了解。在本节中，就来看一下如何使用 MyEclipse 进行项目开发。在这里将要开发一个非常简单的 HelloWorld 的 Java SE 项目。

1.5.1 创建 Java 项目

如果不使用 MyEclipse 开发工具，而是使用记事本开发 Java 程序，那就不需要创建项目，只需要使用 javac 和 java 命令就可以对 Java 程序进行编译和运行。但是在 MyEclipe 进行 Java 程序开发，就需要先来创建一个 Java 项目。

在 MyEclipse 的菜单中，选择“File”|“New”|“Java Project”命令，就会弹出创建 Java 项目的界面，如图 1-12 所示。

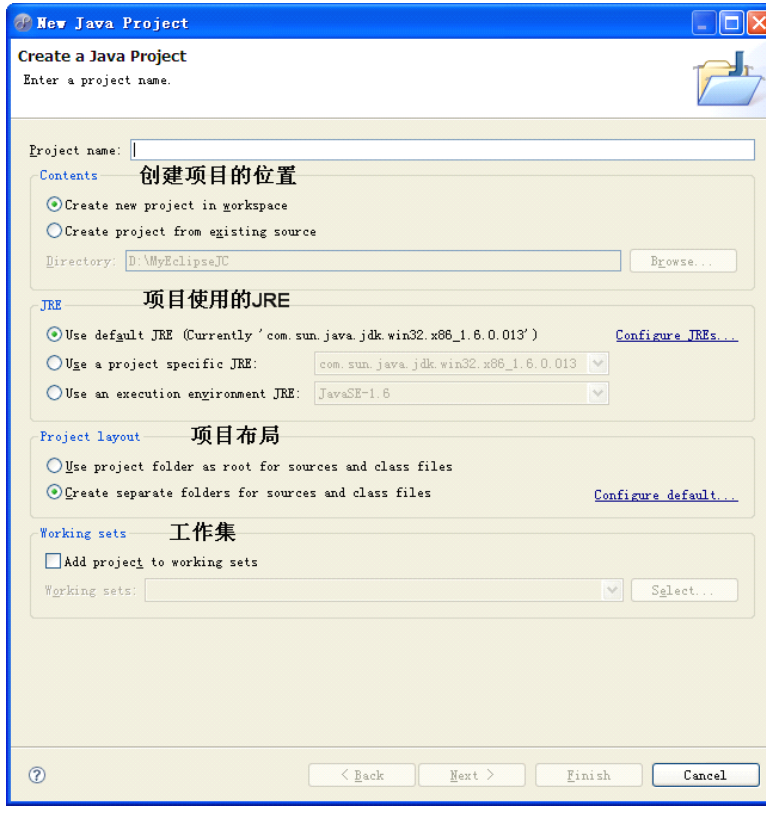


图 1-12　创建 Java 项目

在创建 Java 项目的界面中，“Project name”的文本框用来输入新创建 Java 项目的项目名称，这里我们输入“HelloWorld”。除此之外，还有对新建项目进行设置的选项，

依次是创建项目的位置、项目使用的 JRE、项目布局和工作集。下面我们对每一个选项进行简单讲解。

- Creat new project in workspace：在工作空间内创建项目，这是推荐使用的。
- Creat project from existing source：在现有位置中创建项目，如果选择该选项，则需要单击“Browse”按钮，然后选择项目放置的位置。
- Use default JRE(Currently ‘com.sun.java.jdk.win32.x86_1.6.0.013’)：使用 MyEclipse 默认的 JRE，在开发过程中使用该选项就可以。
- Use a project specific JRE：选择项目使用的 JRE，其中的选项就是目前 MyEclipse 中已经配置的 JRE。
- Use an execution environment JRE：使用项目运行时选择的 JRE。
- Use project folder as root for sources and class files：使用项目目录做为源文件和编译后的文件根目录。使用该选项造成的后果就是.java 文件和.class 文件在一个目录中。
- Create separate folders for sources and class files：分开存放源文件和编译后的文件，默认选择该选项，也是推荐使用的。
- Add project to working sets：添加项目到工作集中，如果选择该选项，则需要指定要添加到的工作集。通常情况下并不进行工作集操作，从而也就不选择它。

从上面的讲解中可以看到只需要填写项目名称，其他采用 MyEclipse 默认的选择就可以。在创建 Java 项目中可以直接单击“Finisth”按钮完成创建 Java 项目的操作；也可以单击“Next”按钮，继续对 Java 项目进行设置，如图 1-13 所示。

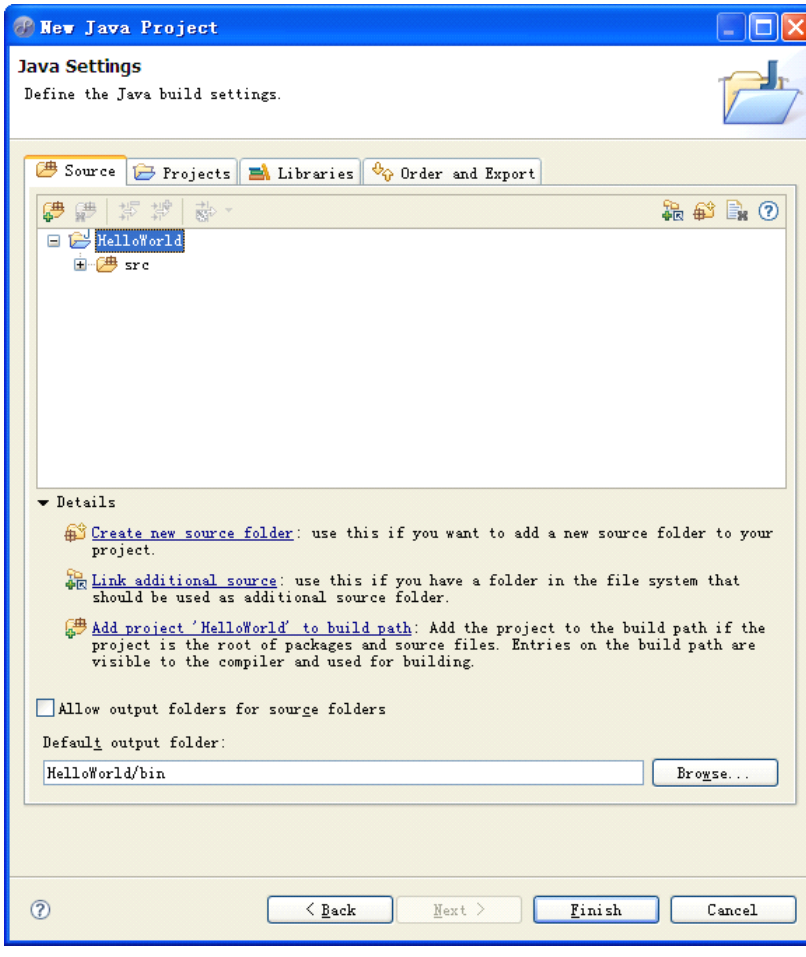


图 1-13　创建项目设置

在创建项目设置界面中，可以在选项卡中对源代码、项目、库和排序导出进行设置，这些在后面的讲解中会学习到。在界面的最下面是设置防止编译后文件的文件夹，通常我们就是用 bin 这个名称。单击“Finish”按钮，就会完成创建 Java 项目的操作，但是会填出一个选择视图的界面，如图 1-14 所示。

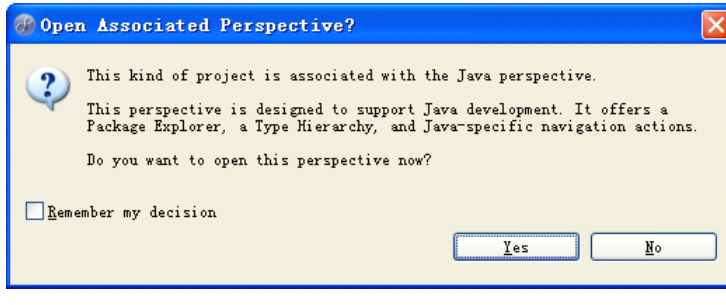


图 1-14　项目的视图选择

该选择视图界面是让用户选择是否采用 Java 视图进行开发，如果采用，则单击“Yes”按钮。通常情况下，我们仍然采用 MyEclipse 视图，这是每一个程序员的习惯有关。这

里单击“No”按钮，也就是不采用 Java 视图，而是采用 MyEclipse 视图。

经过上面的操作，就在 MyEclipse 的工作空间中创建了一个名称为“HelloWorld”的 Java 项目。该项目会出现在包资源管理器中，单击项目前的加号，可以讲项目中的内容展开，如图 1-15 所示。

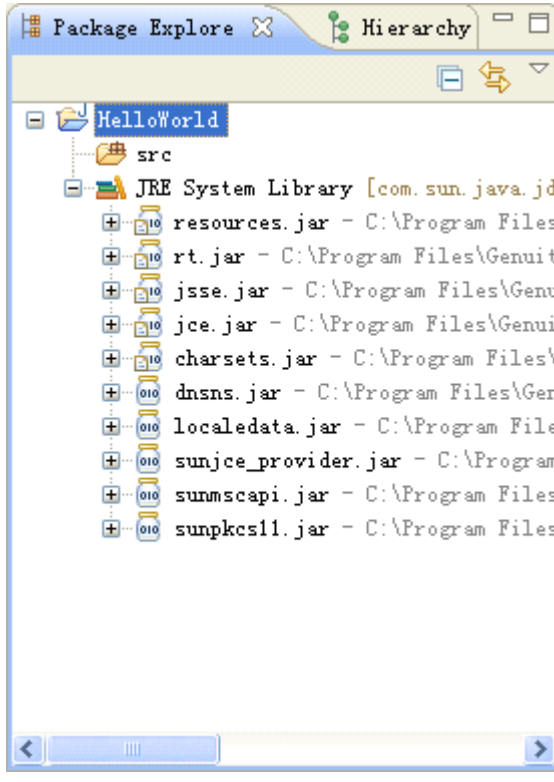


图 1-15　项目文件结构

在生成的项目文件结构中，src 目录就是用户放置 Java 源文件的目录，下面的.jar 文件就是该项目采用的 JRE 文件，这些都是 MyEclipse 为我们自动生成的。

说明：在 MyEclipse 中，创建项目或者程序都有多种方式。例如本小节学习的创建 Java 项目就有另外一种方式，在包资源管理器中的空白处，单击鼠标右键，在弹出菜单中，选择“New”|“Java Project”命令，同样能够进入创建 Java 项目的界面中。

1.5.2　创建 Java 类程序

创建 Java 项目后，就可以再 Java 项目中创建 Java 类程序。在 MyEclipse 中，选择“File”|“New”|“Class”命令，就会出现创建 Java 类程序界面，如图 1-16 所示。

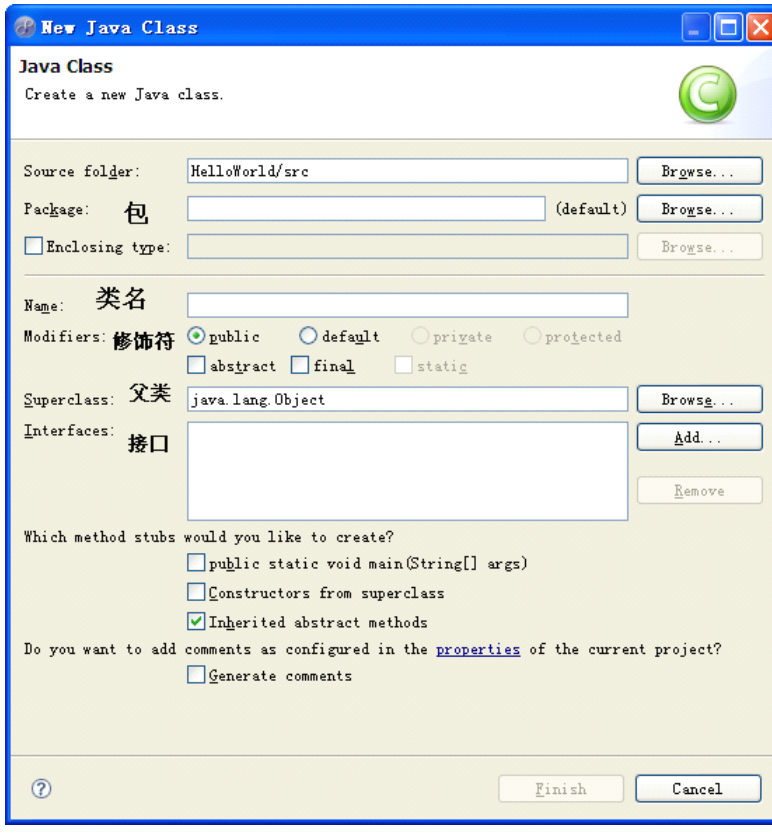


图 1-16　创建 Java 程序

在创建 Java 程序界面中，“Source folder”表示源代码保存的位置。“Package”表示 Java 程序所在的包，这里我们先采用默认的无包情况。“Name”表示类名，这里我们输入“Hello”，这样就创建了一个名称为“Hello”的类。“Modifiers”表示类的修饰符，在访问修饰符中可以选择“public”或者“default”，分别是创建公共类和默认访问修饰类；在非访问修饰符中可以选择“abstract”和“final”，虽然这里将它们设置为复选框，但是这两个修饰符是不能同时存在的。

“Superclass”表示新创建的类所继承的父类，因为 Java 中的所有的类都继承“java.lang.Object”类，所以默认新建类也是继承它的，这里也可以输入其他的父类，或者单击“Browse”按钮选择要继承的父类。“Interfaces”表示新创建类所实现的接口，单击“Add”按钮可以添加要实现的接口。因为一个类可以继承多个接口，所以该操作可以进行任意次。如果接口添加错误，也可以单击“Remove”按钮，将选中的接口删除。

接下来的选项是让程序员选择在程序中创建哪些方法或者程序段。第 1 个选项就是选择新建 Java 类中是否创建 main 方法，该方法是 Java 程序的入口方法，这里我们选中该选项。第 2 个选项表示是否生成和父类中相同的构造函数。第 3 个选项表示是否继承抽象方法，该抽象方式可能来自抽象父类，也可能来自实现的类，如果定义的是非抽象类，则会自动将抽象方法实现。

经过上面的填写和选择，单击“Finish”按钮，就会创建一个类名为“Hello”，具有 main 方法的 Java 程序。创建 Java 类程序后，包资源管理器和编辑区都会发生改变，

如图 1-17 是包资源管理器的变化。

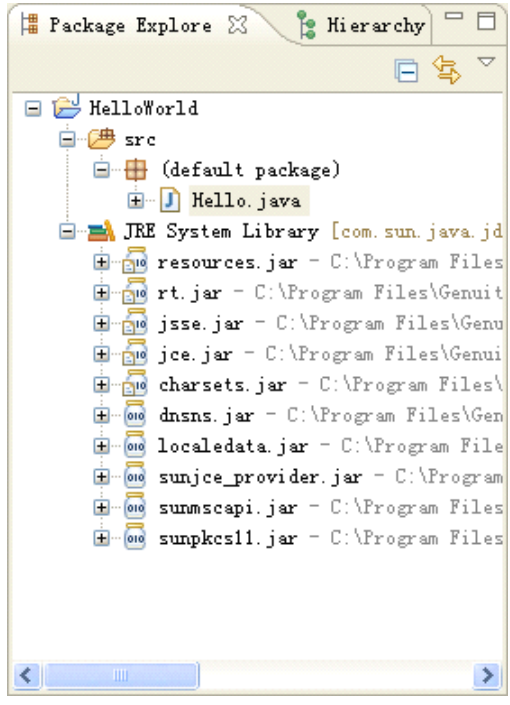


图 1-17　包资源管理器的变化

从图中可以看到在 src 下创建了一个“Hello.java”文件，其中“default package”并不是包名，而是只默认包，也就是没有使用包。

创建 Java 类程序后，同时会在编辑区中将创建的 Java 类程序的代码显示出来，代码内容如下：

```
01  public class Hello {
02      /**
03       * @param args
04       */
05      public static void main(String[] args) {
06          // TODO Auto-generated method stub
07      }
08  }
```

其中唯一改变的就是将其中的空行去掉。在第 1 行中定义了一个公共的类，类名为“Hello”，从第 2 行到第 4 行和第 6 行都是自动生成的注释。第 5 行是我们在创建 Java 类程序界面中选择 main 方法后自动生成的，如果不选择该选项，是不会出现该 main 方法的。将其中的第 6 行注释去掉，换成下面的代码：

```
System.out.println("Hello World!!!");
```

这样 Java 类程序就开发完成。

技巧：除了这种创建 Java 程序的方式外，还有另外一种更好的方式，那就是在 Java 程序要放置的程序或者包目录上，单击鼠标右键，在弹出菜单中选择“New”|“Class”命令。这种方式的好处是直接指定程序的位置，如果采用菜单的方式创建，经常还需

要指定程序位置。

1.5.3 运行 Java 类程序

如果不使用 MyEclipse 等集成开发工具，运行 Java 类程序需要使用 javac 和 java 命令。如果使用 MyEclipse 开发工具，它已经将 javac 和 java 命令集成。在 MyEclipse 开发工具中，编译时自动完成的，也就是说自动生成了 .class 文件。这里主要是关心如果运行 Java 类程序，这是需要程序员操作完成的。

运行 Java 类程序有多种方式，分别是菜单、工具栏和弹出菜单三种方式。在 MyEclipse 的菜单中，选择“Run”|“Run”命令，如果在编写 Java 类程序时没有保存，则会弹出如图 1-18 的界面。

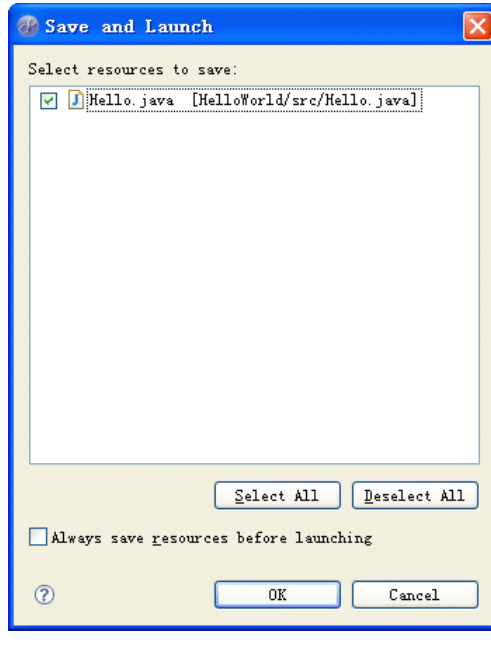


图 1-18　选择运行界面

因为目前项目中只有“Hello.java”这一个类程序处于编写状态，如果有多个程序，则都会列在该界面中，从而选择运行的程序。选择后，单击“OK”按钮，从而运行选中的程序。

如果编写的 Java 程序已经保存，则选择“Run”|“Run”命令后，直接运行当前保存的程序。

记忆：运行程序的快捷键是 Ctrl+F11。在 MyEclipse 中，大部分操作都有对应的快捷键，通过这些快捷键能够大大加快程序开发速度。

运行 Java 类程序后，会将运行结果显示到位于 MyEclipse 最下面的控制台中，如图 1-19 所示。

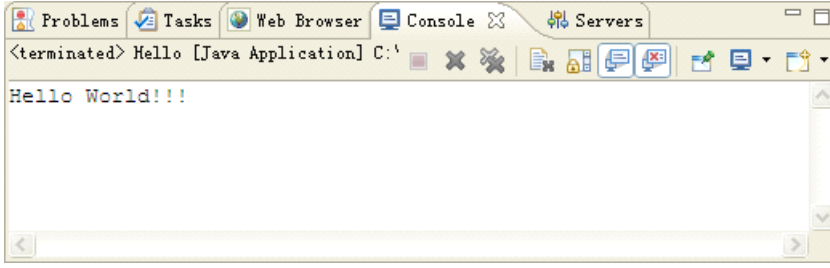


图 1-19　控制台

其中“Hello World！！！”就是上一小节中创建的 Java 类程序的运行结果。

说明：除了通过菜单运行 Java 类程序外，还可以通过工具栏中的运行按钮执行。直接单击该按钮，是对上次运行的程序再次运行，这通过用在开发过程中，如果想运行其他程序，需要通过旁边的下三角进行选择。在编辑区中，可以单击鼠标右键，在弹出菜单中，选择“Run As”|“Java Application”命令对当前程序进行运行。

1.6　开发接口程序

在上一节中，我们讲解了如何进行 Java 类程序开发。在 Java 程序中，除了类之外，还有接口，它也是非常重要的。在本节中，就来学习一下如何创建接口，以及相关的包和实现接口的类，这些在 Java 项目的开发中都是非常重要的。

1.6.1　创建包

在前面学习创建 Java 类程序时，并没有使用到包，而是直接放在 src 目录下。在本小节中就来看一下在 MyEclipse 如何创建包。在 MyEclipse 菜单中，选择“File”|“New”|“Package”命令，就会进入创建包界面，如图 1-20 所示。

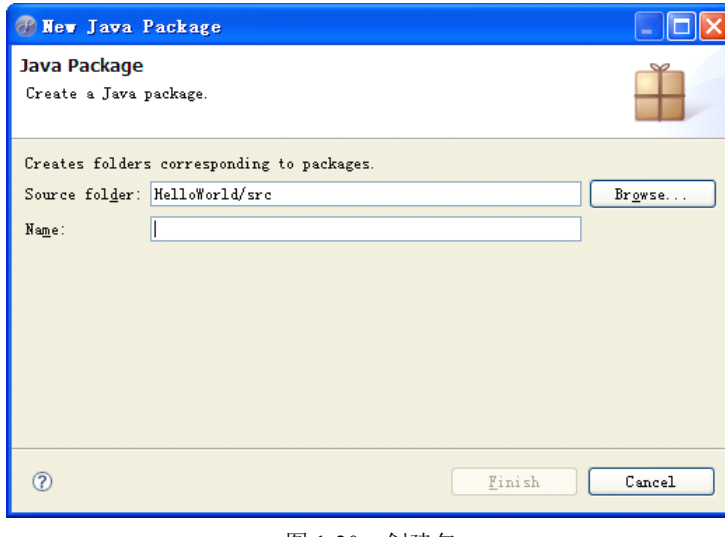


图 1-20　创建包

其中“Source folder”表示包定义的所在位置，“Name”表示定义的包名。这里的包名可以是一个单词，例如“service”，就是创建一个名称为“service”的包，使用该包通常保存 Web 项目中的业务逻辑层接口。包名也可以是多个单词，中间使用“.”连接，这样表示级联包，也就是在包下再创建子包。例如“service.impl”，表示在“service”包下再创建一个名称为“impl”的子包，使用该包通常保存 Web 项目中的业务逻辑层实现类。

开发完上面两个包后，包资源管理器如图 1-21 所示。

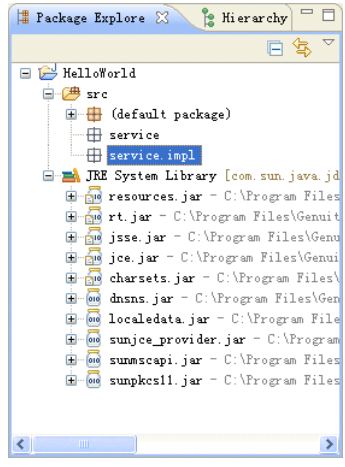


图 1-21　创建包后

从图中可以看到“service”和“service.impl”两个包都已经创建成功了。在包资源管理器中还有另外一种显示方式，单击右上角的倒三角，选择“Package Presentation”命令，在该命令下有两个选项，分别是“Flat”和“Hierarchical”，它们表示两种显示包的方式。默认的是“Flat”方式，也就是包平面显示方式，将所有的包并列进行显示，例如上面的图。“Hierarchical”方式是指分层方式，也就是将子包放在包下。我们将包显示方式选择为“Hierarchical”方式，则包资源管理器变化如图 1-22 所示。

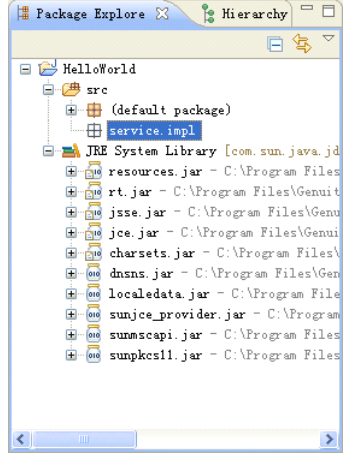


图 1-22　分层方式显示包

从结果中可以看到将两个包重叠在一起，这是因为目前包中并没有任何 Java 程序造成的结果。在后面创建 Java 程序后，读者可以再来进行切换包显示方式操作。

1.6.2 创建接口

接口也是 Java 程序中必不可少的一部分。在上一小节中，已经创建了“service”和“service.impl”两个包，在本小节中就在“service”包下创建一个接口。

在 MyEclipse 中创建接口也有多种方式，最常用的就是菜单和右键弹出菜单两种方式，因为我们这里要将接口创建在指定的包下，所以建议采用右键弹出菜单的方式。在“service”包目录上，单击鼠标右键，在弹出的菜单中，选择“New”|“Interface”命令，就会弹出创建接口界面，如图 1-23 所示。

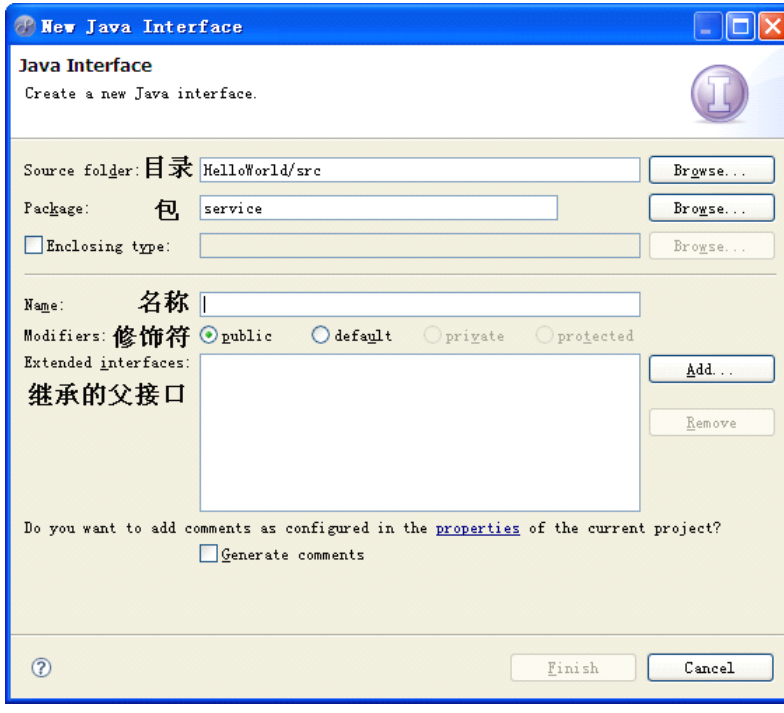


图 1-23　创建接口界面

在创建接口界面中，“Source folder”表示创建的接口程序所在目录，“Package”表示创建的接口所在的包，如果采用右键菜单的方式创建，会自动出现选择的包，例如这里的“Service”。

“Name”表示创建的接口名，这里我们填写“UserService”，也就是用户业务逻辑接口。“Modifiers”表示接口的修饰符，这里可以在 public 公共修饰符和默认修饰符之间选择。“Extended interfaces”表示新建接口继承的父接口，父接口可以由多个，单击“Addd”按钮完成添加。

填写上述信息后，单击“Finish”按钮，就会完成接口的创建，从而创建一个接口名为“UserService”的接口，并在编辑区中将自动生成的接口代码显示出来，如下：

```
01    package service;
```

```
02    public interface UserService {
03
04    }
```

从代码中可以看到，在第 1 行中指定接口所在的包。在其中第 3 行加入用于判断用户是否能够登录的方法，如下：

```
public boolean login(String username,String password);
```

这样我们的一个接口就创建完成了。

1.6.3 创建实现接口的类

在本小节中就来学习一下如何创建实现接口的类，这和前面创建普通的类是有所不同的，我们也就该类创建在包下。在“service.impl”包上，或者在“impl”子包上，单击鼠标右键，在弹出的右键菜单中，选择“New”|“Class”命令，将弹出创建 Java 类界面，如图 1-24 所示。

图 1-24　创建实现接口的类

因为我们是通过在包上的右键菜单进行创建，所以会将包自动添加上。在“Name”中填写“UserServiceImpl”，表示用户业务逻辑实现类。该类要实现上一小节创建的用户业务逻辑接口，所以要在“Interfaces”中，单击“Add”按钮，弹出如图 1-25 的选择接口界面。

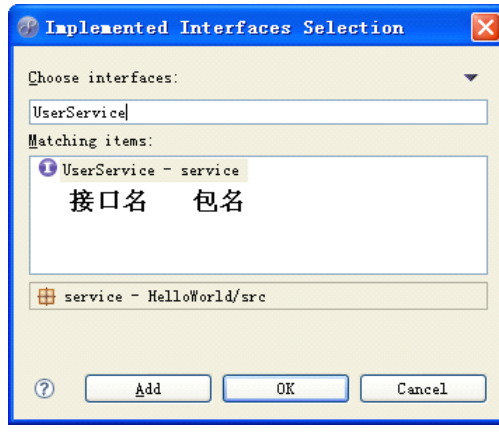


图 1-25 选择要实现的接口

在界面中输入要实现的接口，则在下面会将可以选择的接口列出来，如果在不同的包中具有同样名称的接口，则都会列在下面，并且在接口名的后面还会列出包名。选择要实现的接口，单击“OK”按钮，就会完成接口的添加，并且将接口显示到创建类界面中。

注意：一定要保持创建 Java 类程序界面中“Inherited abstract methods”选项的选中状态。因为接口中有抽象方法，在实现该接口的类中要实现该方法，通过该选项可以自动实现抽象方法，不然需要手动编写。

单击创建 Java 类程序界面中的“Finish”按钮，创建实现接口的类的操作就完成，并且将它的代码显示到编辑区中，代码如下：

```
01  package service.impl;
02  import service.UserService;
03  public class UserServiceImpl implements UserService {
04      public boolean login(String username, String password) {
05          // TODO Auto-generated method stub
06          return false;
07      }
08  }
```

上述代码都是自动生成的代码，其中第 1 行是包定义代码，指定新创建的类在“service.impl”包下。第 2 行和第 3 行自动让新创建的类实现“UserService”接口，如果该接口和实现类不在同一包下，还会自动导入接口。第 4 行是自动实现接口中的抽象方法，程序员可以在其中添加自己的业务操作。

Java中使用包管理类与介面，用以实现命名空间的功能。

在Java语言中，应用程序接口（API）化身成类，并且分组成为包。每个包中包含有相关的接口和类。

第2章　常用开发操作和技巧

在上一章中对 MyEclipse 有了基本了解一下，在本章中就来学习一些非常重要的常用开发操作和技巧。这些都是后面学习具体开发技术的基础。根据操作的内容不同，可以讲这些操作分为几类，例如配置方面的操作、源代码方面的操作，以及包围方式。

2.1　常用配置操作

MyEclipse 是一种视图话的开发工具，所以我们进行设置操作时主要是通过配置的手段来完成。这方面的知识点是比较零散的，而且后面的操作要基于其中的某些配置，所以我们把这些配置操作先进行讲解。

2.1.1　显示行号

因为后面讲解代码时，经常要说代码的行号的，所以先来看一下如何为 MyEclipse 中的代码加行号。在 MyEclipse 的菜单中，选择“Window”|“Preferences”命令，会弹出进行设置的界面，如图 2-1 所示。

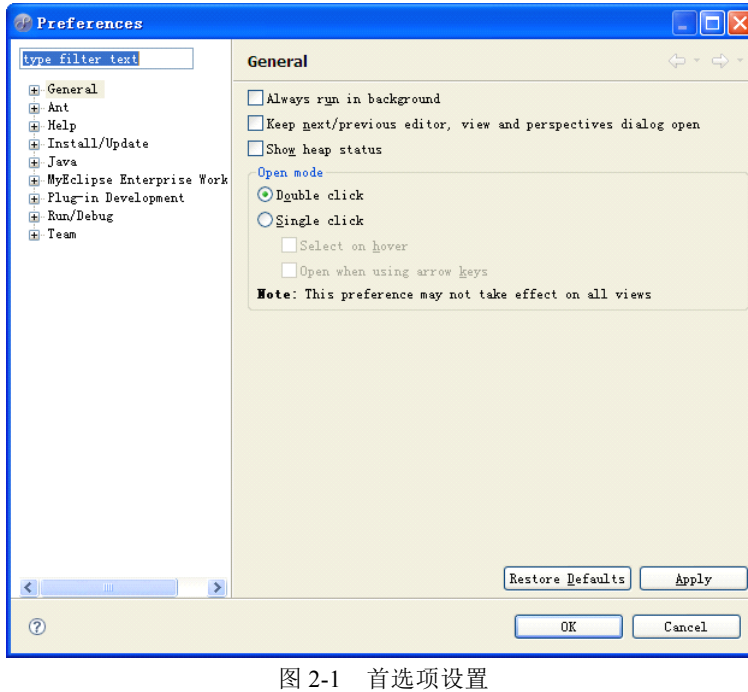


图 2-1　首选项设置

在 MyEclipse 中，大部分的设置都是通过首选项设置界面完成的。在其中的左侧菜单中，选择“General”|“Editors”|“Text Editors”命令，则首选项设置界面如图 2-2

所示。

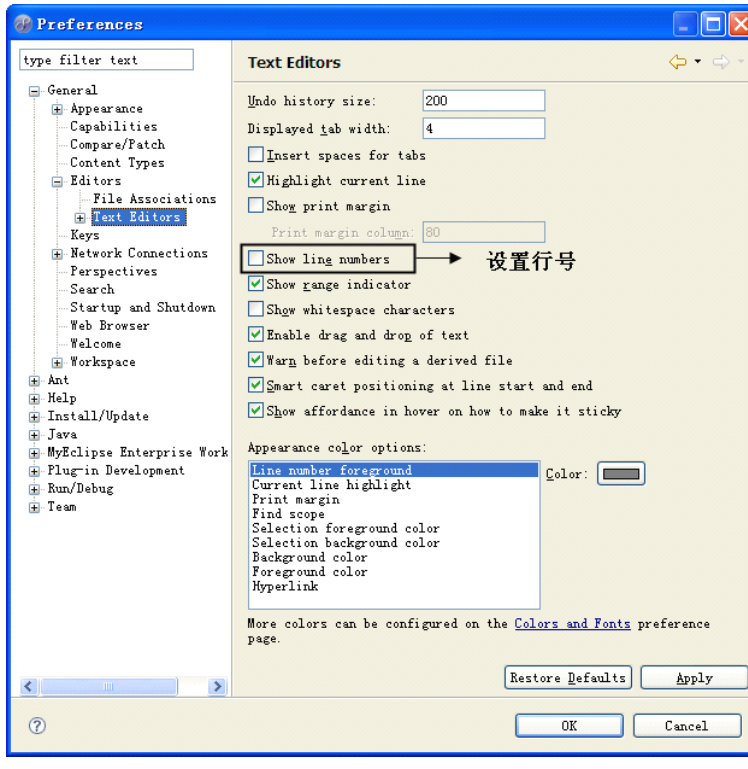


图 2-2　设置行号

其中“Show line numbers”选项就是用来设置是否显示行号，选中该选项，单击“OK”按钮，就会在 MyEclipse 的编辑区中，显示行号。

说明：显示代码的行号，还有另外一种简单的设置方法，那就是在编辑区的左边界上，单击右键，在其中选中“Show line numbers”选项，从而进行设置。具体操作可以观看本书随书光盘。

2.2.2　改变编码类型

在计算机的信息传输中，都是通过字节流的形式传递的，在其中就要用到一个非常重要的概念，那就是编码。通过编码可以讲文字转换为字节，也可以讲字节再转换为文字。

目前计算机中有多种编码形式，两种不兼容的编码是不能够结合使用的。例如 A 编码中，123 代表“书”这个字，我们要传递“书”这个字时，接收的就是 123。但是在反转换时使用的是 B 编码，但是在 B 编码中，123 对应的是“纸”这个字，那它得到的信息就是“纸”这个字了。这样就造成传输信息的失败。

在计算机的操作中，经常会出现乱码的情况，这就和使用的编码有关。在 MyEclipse 中，我们写代码时也会使用到某一种编码，在团队开发中一定要使用同一种编码，在本节中就来学习一下如何改变编码类型。

在首选项设置界面中，选择“General”|“Editors”|“Text Editors”|“Spelling”命令，就会出现如图 2-3 所示的界面。

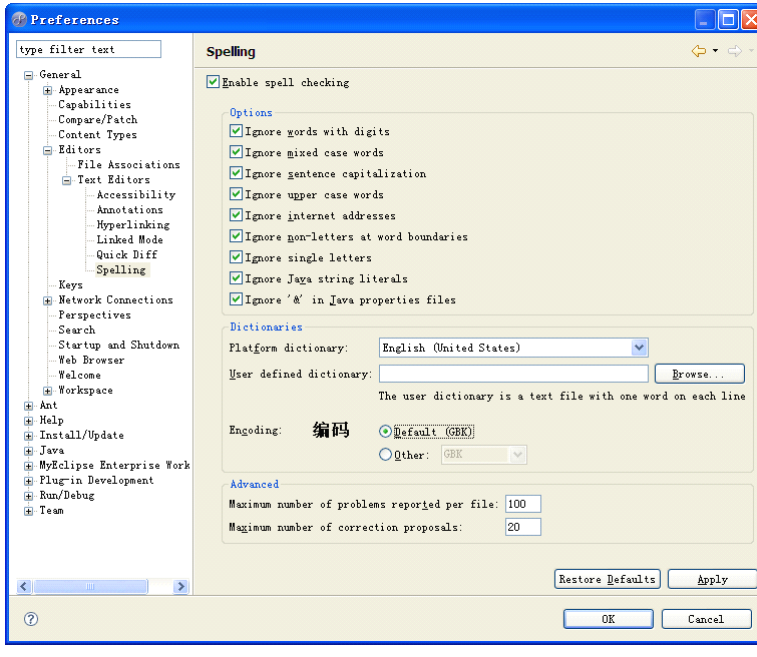


图 2-3　设置编码规则

其中“Encoding”选项就是用来设置代码的编码规则的，当前默认编码是“GBK”。如果想换为其他编号，可以选择“Other”选项，然后在后面的下拉列表中选择相应的编码。这样就完成改变编码类型的操作。

说明：在 MyEclipse 中，还可以为每一种程序进行相应的编码设置，例如 JSP、HTML、CSS 等。在后面学习具体每一种技术时再进行讲解。

2.2.3　导入项目

在项目开发中，经常使用到不同工作空间的两个项目，或者别人发过来的项目不在工作空间中，这时候就要用到导入项目的操作。通过这种操作可以将项目导入到当前工作空间中，并显示到 MyEclipse 中。

在 MyEclipse 的菜单中，选择“File”|“Import”命令，就会进入导入界面，如图 2-4 所示。

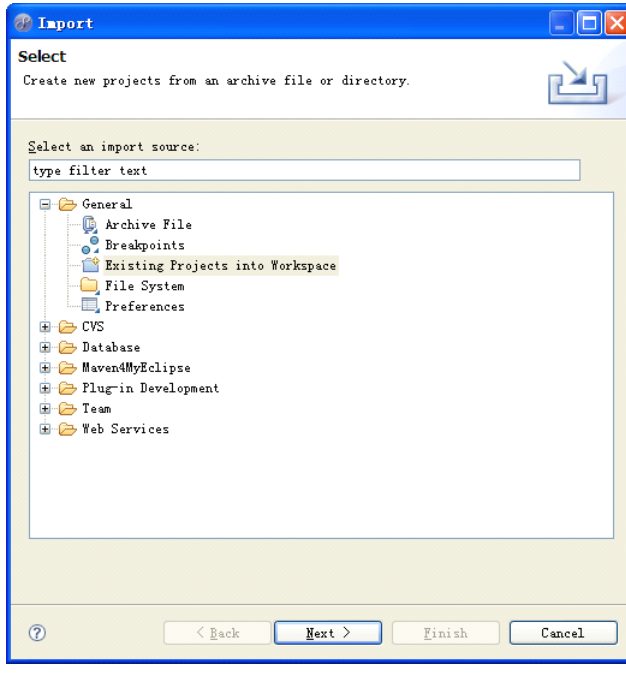


图 2-4　导入界面

在其中选择“General”|“Existing Projects into Workspace”命令，单击“Next”按钮，就会进入导入项目的界面，如图 2-5 所示。

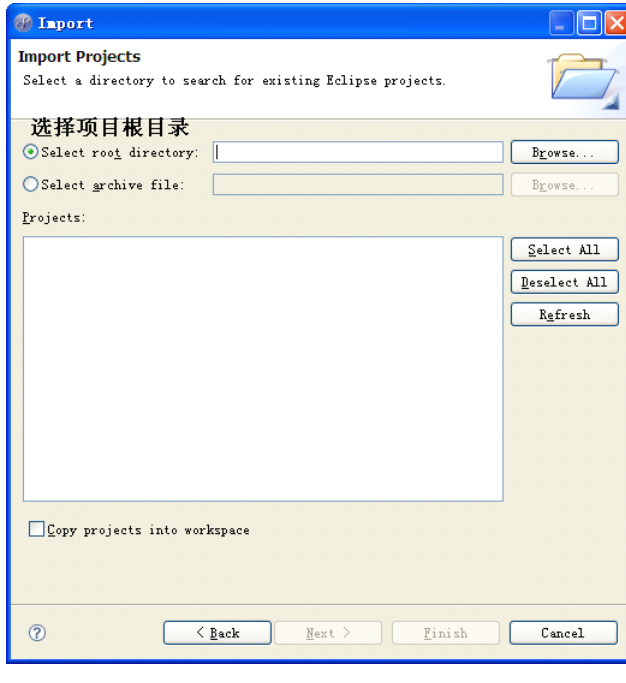


图 2-5　导入项目界面

其中“Select root directory”表示选择项目的根目录，“Select archive file”表示选

择归纳文件。在“Select root directory”选项后，单击“Browse”按钮，在界面中可以选择项目的根目录，也可以选择项目所在的目录。如果选择的是项目的根目录，表示将该项目导入；如果选择的是项目所在目录，就会将该目录下所有的项目导入。导入到下面的文本域中后，还可以通过“Select All”、“Deselect All”和“Refresh”按钮进行相应操作。最后单击“Finish”按钮，从而完成导入项目的操作，在 MyEclipse 中可以看到刚才选择的项目已经出现在其中。

2.2.4 导出项目

在 MyEclipse 中，可以将项目导入进来，也可以将当前项目导入到指定目录下。在 MyEclipse 菜单中，选择“File”|“Export”命令，就会进入导入界面，如图 2-6 所示。

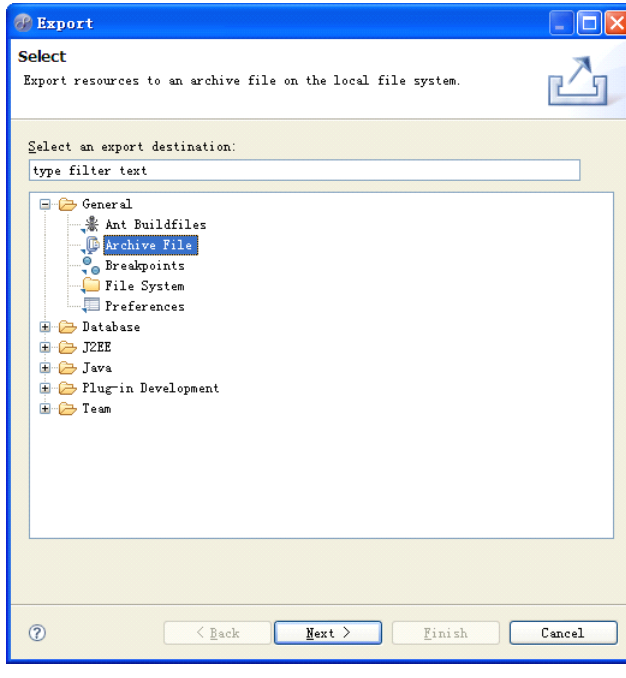


图 2-6 导出界面

在其中选择“General”|“Archive File”选项，单击“Next”按钮，就会进入导出项目界面，如图 2-7 所示。

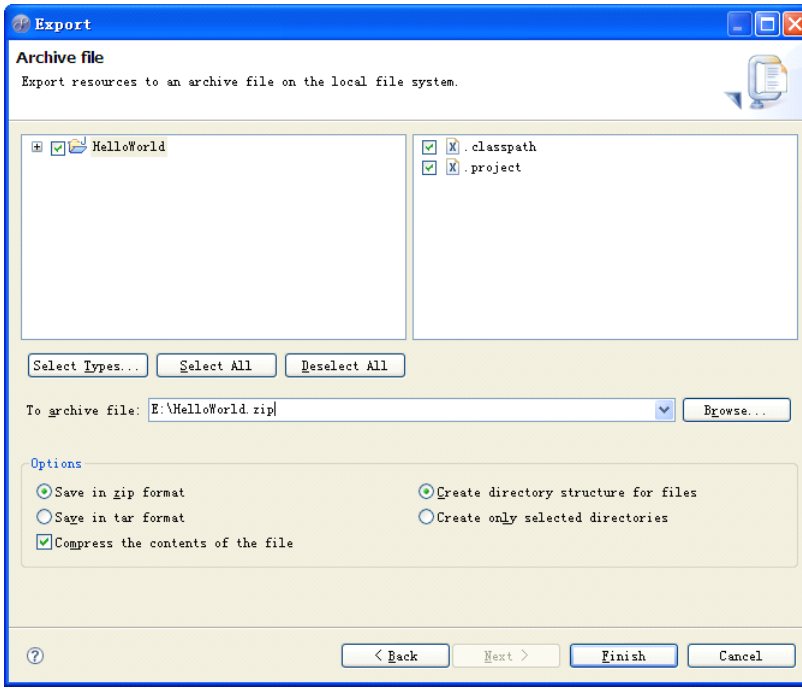


图 2-7　导出项目界面

在导出项目界面中，在左上角选择你要导出的项目，这里我们选择上一章创建的“HelloWorld”项目。在“To archive file”选项中，选择要到处的位置。这里通常到处为以项目名为文件名的 ZIP 文件。最后单击“Finish”按钮，从而在指定的位置出现一个文件名为“HelloWorld.zip”的文件，从而完成导出项目的操作。

2.2.5　修正代码错误

MyEclipse 是一种功能非常强大的集成开发工具，它会自动提醒程序开发过程中出现的错误，在出现错误的代码行上以红叉的形式标识，如图 2-8 所示。

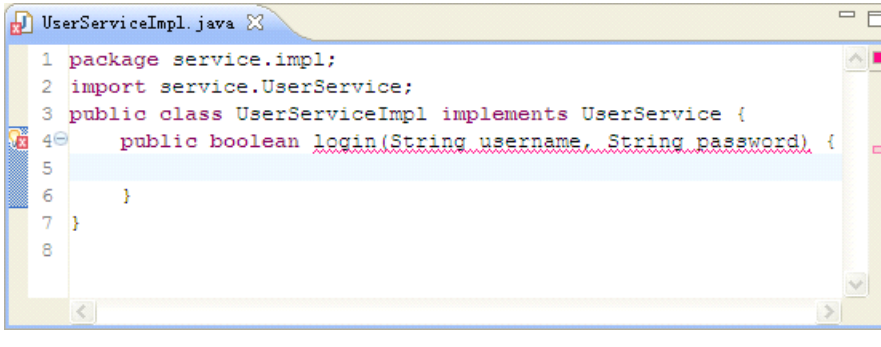


图 2-8　错误提示

在该 UserServiceImpl 类程序中，并没有为 login 方法定义返回值，所以出现了错误。在 MyEclipse 中，具有对这种错误快速处理的方法。

单击代码左面的红叉，如图 2-9 所示。

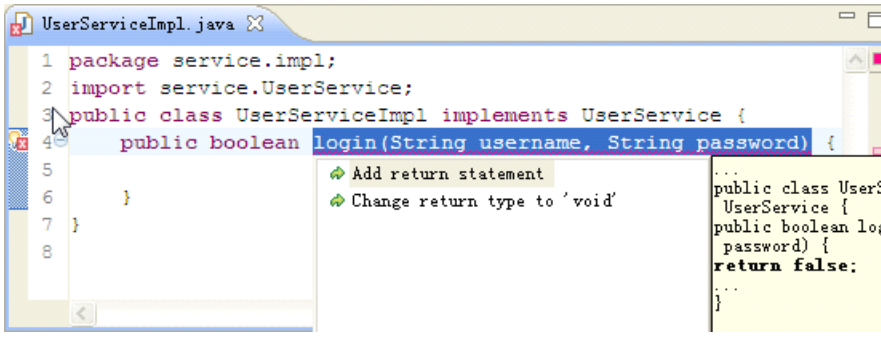


图 2-9　修改代码错误

从图中可以看到，当单击红叉时，会弹出一个窗口，给出两种解决该错误的方法。第一种是增加一个返回语句；第二种是改变方法的返回类型为 void。并在后面的窗口中给出修改后的部分代码。具体选择哪一种修正方式，由程序员根据实际需要进行选择，并不是所有的修正方式都是合适的。

记忆：修改代码错误的快捷键是 Ctrl+1，但是一定要首先将鼠标放在出现错误的一行才能够使用快捷键。使用快捷键的效果和单击红叉的效果相同的。

说明：在 MyEclipse 中，除了会显示程序中的错误外，还会对程序的一些地方提出警告，在这代码的最左端以点亮的灯泡的形式标识。警告并不是错误，只是给出一种建议，例如使用集合时，如果没有使用泛型，就会给出警告。

2.2.6　JRE 相关操作

JRE 是运行 Java 程序所必须的环境集合，如果只需要运行 Java 程序，只使用 JRE 就可以。如果还需要开发 Java 程序，就要使用到前面安装的 JDK。

安装 MyEclipse 后，在 MyEclipse 中就已经存在一个默认的 JRE。但是如果想使用最新版本的 JRE，或者想得到更多的源码信息，就要自己添加 JRE。

在首选项设置界面中，选择“Java”|“Installed JREs”命令，就会出现设置 JRE 界面，如图 2-10 所示。

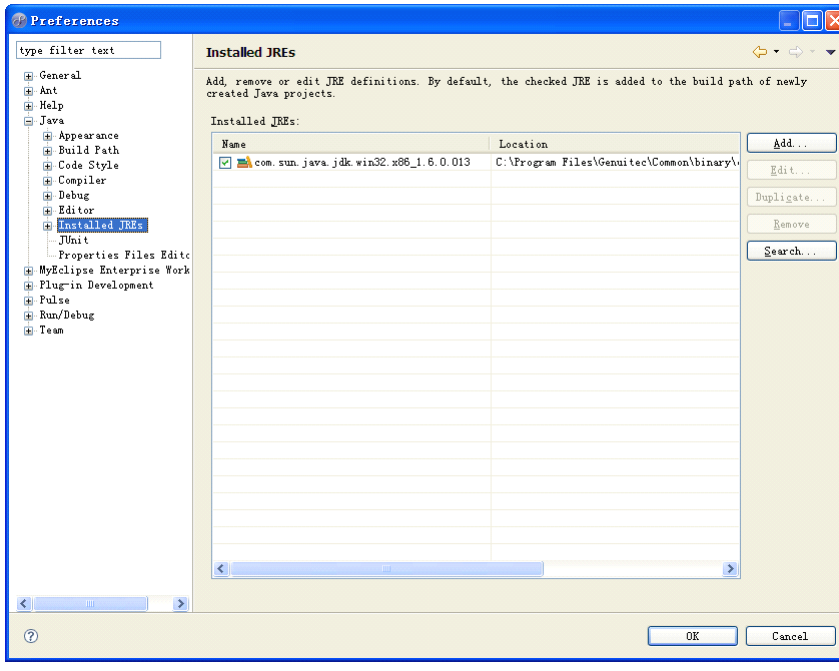


图 2-10　设置 JRE

在其中可以看到默认选中的是名称为“com.sun.java.jdk.win32.x86_1.6.0.013”的 JRE，这就是 MyEclipse 默认的 JRE。单击“Add”按钮，就可以添加外部的 JRE，如图 2-11 所示。

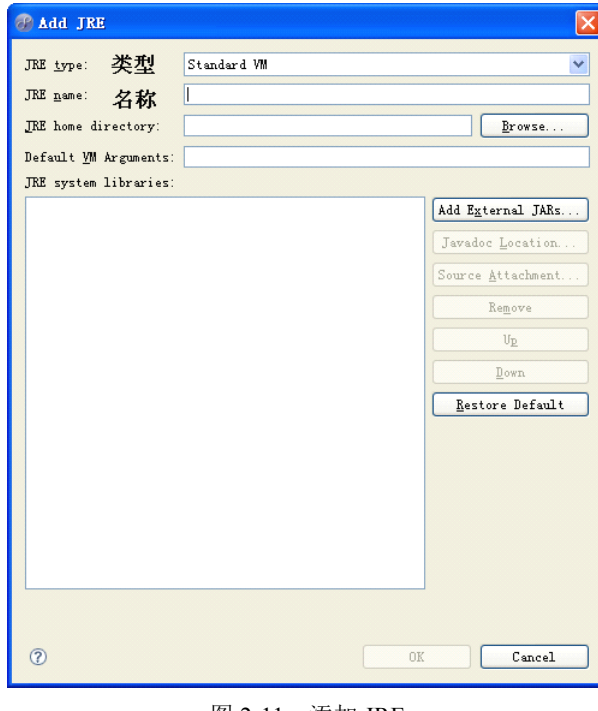


图 2-11　添加 JRE

在添加 JRE 界面中，单击“JRE home directory”选项后的“Browse”按钮，在弹出的界面中选择安装 JDK 后的 JRE 根目录，如果是默认安装，则根目录为“C:\Program Files\Java\jre6”，单击“确定”按钮，从而完成 JRE 的选择。并在添加 JRE 界面中自动填入“JRE name”、“ JRE home directory”和“JRE system libraries”选项，单击“OK”按钮，从而完成 JRE 的添加操作，从而也在设置 JRE 界面中添加一个新的 JRE 选项。

添加外部 JRE 后，就可以在默认和新添加的 JRE 之间进行选择。选中外部添加的 JRE，还可以通过右边的按钮，对它进行编辑、删除等操作，读者可以自己操作一下。

2.2.7 导入 JAR 包

在开发 Java 项目时，经常要用到已经完成的项目或者程序，它们通常以 JAR 包的形式出现的。例如在进行 JDBC 操作时，要导入数据库驱动包才能够正常操作。在开发中，如果想使用这些 JAR 包就要进行导入 JAR 包操作。

导入 JAR 包操作有多种方式，这里我们选择一种最常用、最不易出错的导入方式。复制要导入的 JAR 包，在 MyEclipse 中选择项目根目录，然后执行粘贴操作，这里可以在菜单中选择“Edit”|“Paste”命令，也可以直接使用“Ctrl+V”快捷键，从而将 JAR 复制到项目中。包资源管理器如图 2-12 所示。

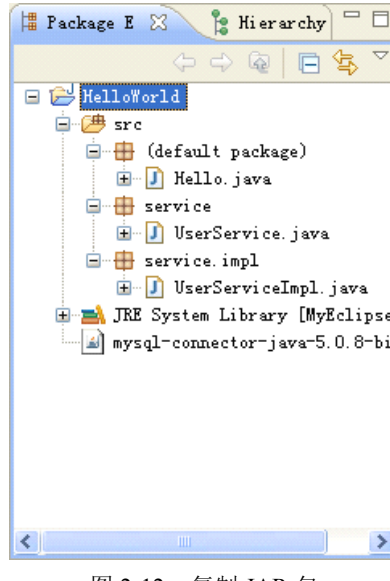


图 2-12　复制 JAR 包

选中导入的 JAR 包，单击鼠标右键，在弹出的菜单中，选择“Build Path”|“Add to Build Path”命令，从而将该 JAR 包导入到 Build Path 中。包资源管理器如图 2-13 所示。

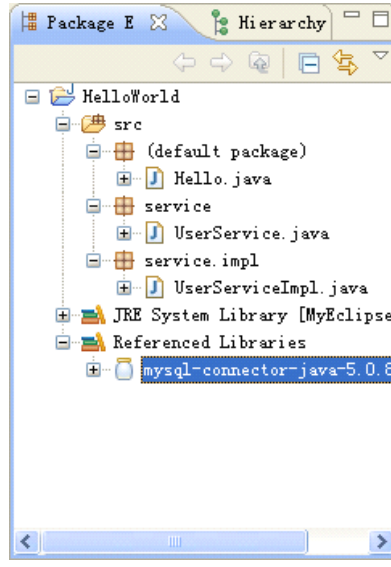


图 2-13　导入包

注意：导入 JAR 包时，还有其他方式，可以将外部 JAR 包不进行复制而是直接导入到项目中，这样做的产生的后果就是 JAR 包和项目的位置都不能变化，不然就需要重新导入。

如果已经不再用到导入的 JAR 包，也可以将它删除。选中要删除的 JAR 包，单击鼠标右键，在弹出的菜单中，选择“Build Path”|“Remove from Build Path”命令，就可以将 JAR 移除项目。

注意：本小节讲解的导入 JAR 包都是对于 Java SE 项目而言的，如果是 Java Web 项目，导入 JAR 包时非常容易的，只需要将 JAR 包复制到 WebRoot/WEB-INF/lib 目录下。

说明：在项目进行文件的复制、添加、删除等操作时，有时候并不能实现同步操作，这是和文件类型、计算机硬件有关的。如果不能看到操作结果，可以在菜单中选择“File”|“Refresh”命令，或者使用“F5”快捷键对项目进行刷新，从而看到操作结果。

2.2　源代码操作

在 MyEclipse 的菜单中，有一个“Source”菜单，在编辑区的右键弹出菜单中，也有“Source”选项，它们完成的操作几乎是一样的。通过它们可以完成源代码相关的操作，在后面的讲解中，只以其中的右键弹出菜单方式进行讲解。

记忆：在右键菜单中选择“Source”命令也是有快捷键的，快捷键为“Alt+Shift+S”。在编辑区中，使用该快捷键，就只会出现“Source”命令下的所有选项。

2.2.1 添加和删除注释

在 Java 程序中，注释是必不可少的一部分。通过注释能够使其他程序员很容易的看懂你的程序。在 Java 程序中有两种常用的注释方法，那就是单行注释和多行注释。

1.添加和删除单行注释

选中要单行注释的行，在编辑区的右键弹出菜单中，选择“Source”|“Toggle Comment”命令，就会将该行前面加上“//”，从而将当前行注释掉。在该行中，再次选择“Source”|“Toggle Comment”命令，就会将该行的注释去掉。

记忆：进行单行注释的快捷键是“Ctrl+7”，不管是加注释，还是去掉注释，都是该快捷键。

2.添加和删除多行注释

选中要进行多行注释的所有行，在编辑区的右键弹出菜单中，选择“Source”|“Add Block Comment”命令，就会为选中的行加上多行注释。

再次选择被注释的多行，选择选择“Source”|“Remove Block Comment”命令，就会将多行注释去掉。

记忆：添加多行注释的快捷键为“Ctrl+Shift+/”，删除多行注释的快捷键为“Ctrl+Shift+\”。

除了单行注释和多行注释外，还有文档注释，它是使用比较少的，通过选择“Source”|“Generate Element Comment”命令完成，快捷方式为“Alt+Shift+J”。

2.2.2 调整代码格式

在调整代码中，有两种操作，一种是调整行缩进，另外一种就是调整格式。我们通过下面打乱后的 HelloWorld 程序讲解它们之间的不同。

```
01  public class Hello
02  {
03          public static void main(String[] args)
04  {
05                  System.out.println("Hello World!!!");
06                  }
07  }
```

先来进行缩进操作。全部选中上述代码，在编辑区的右键弹出菜单中，选择“Source”|“Correct Indentation”命令，则 HelloWorld 程序的代码调整为：

```
01  public class Hello
02  {
03          public static void main(String[] args)
04          {
05                  System.out.println("Hello World!!!");
06          }
07  }
```

上述代码中，第 4 行和第 6 行的花括号的缩进已经符合规范。从代码的变化可以看

到，“Correct Indentation”命令的作用就是调整所有选中行中的缩进。

记忆：调整行缩进的快捷键为“Ctrl+I”。

在编辑区中，使用“Ctrl+Z”快捷键，让代码回到打乱状态，选择“Source”|“Format”命令，则 HelloWorld 程序的代码调整为：

```
01  public class Hello {
02      public static void main(String[] args) {
03          System.out.println("Hello World!!!");
04      }
05  }
```

从代码的变化可以看到，不但缩进符合的了规范，而且换行也符合了规范。“Format”命令的作用就是调整所有选中行的格式。

记忆：调整格式的快捷键为“Ctrl+Shift+F”，调整格式是包含调整缩进的功能的，所以在开发中完全可以只使用“Format”命令。

2.2.3 添加和组织导入接口和类

当程序中使用的接口或者类和当前程序类不在同一包下，并且接口或者类并不是在“java.lang.*”包下，就需要将接口或者类导入到当前程序中。

导入接口或者类有两种方式，一种是添加导入，另一种是组织导入。添加导入是导入当前选中的接口和类，在编辑区的右键弹出菜单中，选择“Source”|“Add Import”命令可以完成该功能。组织导入是将当前程序中需要导入的接口和类一起导入，而且如果当前导入的接口或者类没有使用，也会被删除掉。在编辑区中，选择“Source”|“Organize Import”命令可以完成组织导入的功能。

记忆：添加导入的快捷键为“Ctrl+Shift+M”，组织导入的快捷键为“Ctrl+Shift+O”。从两种导入的功能上可以看到组织导入覆盖了添加导入的功能，所以在开发中可以只使用组织导入。

在大部分情况下，直接进行上述操作都是可以完成相应功能的。但是当要导入的接口或者类在多个包中出现时，就要让程序员进行选择，到底要导入哪一个接口或者类。例如在 HelloWorld 程序的 main 方法中，输入如下代码：

```
Date date=new Date();
```

再进行添加导入或者组织导入操作时，就会弹出一个选择的界面，如图 2-14 所示。

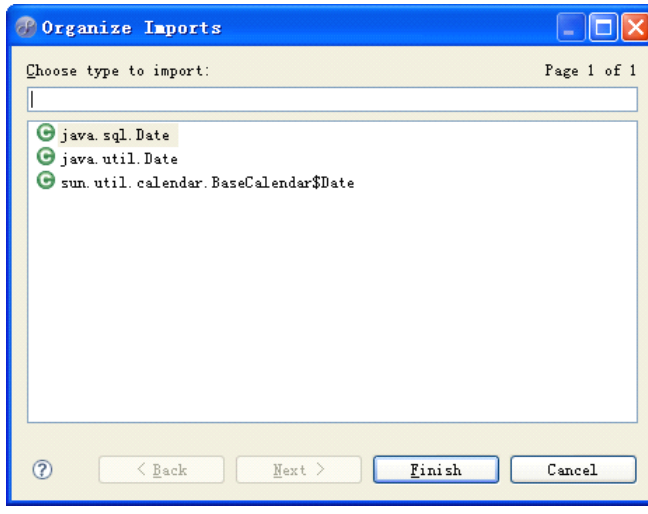


图 2-14　选择要导入的类

因为 Date 类存在多个包下，所以在下面列出所有的 Date 类，在其中选择要导入的 Date 类，例如选择“java.util.Date”，单击“Finish”按钮，从而完成导入操作。这是在 HelloWorld 程序的最上端就会出现如下代码。

```
import java.util.Date;
```

2.2.4　重写和实现方法

在开发 Java 类程序时，经常要用到继承父类和实现接口。如果父类和接口中有抽象方法，那在当前类中就要实现这些抽象方法。如果父类中具有非抽象方法，在当前类中也可以重写这些方法。

在 MyEclipse 中，重写和实现方法是通过一个命令来完成的。这里我们使用 HelloWorld 项目中的“service.impl”包下的“UserServiceImpl”程序举例，首先将它其中的 login 方法去掉，则代码变为：

```
01  package service.impl;
02  import service.UserService;
03  public class UserServiceImpl implements UserService {
04
05  }
```

在编辑区的右键弹出菜单中，选择“Source”|“Override/Implement Methods”命令，将弹出重写和实现方法界面如图 2-15 所示。

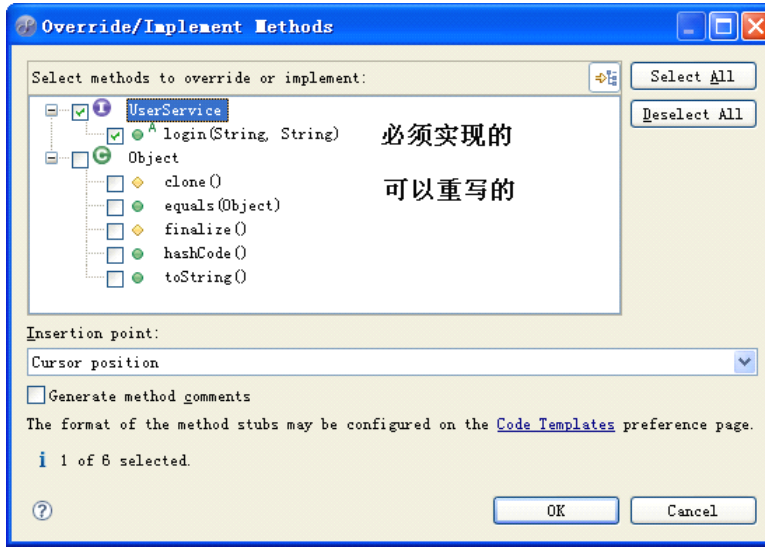


图 2-15　重写和实现方法

因为“UserServiceImpl”类实现了“UserService”接口，所以在类中必须实现 login 方法。在重写和实现方法界面中，首先将继承的父类和实现的接口列出来，然后在父类或者接口下列出可以重写或者实现的方法。必须实现的方法默认情况下是选中的状态，而可以重写的方法默认情况下是未选中状态。在其中选中 login 和 toString 方法，单击“OK”按钮，从而完成重写和实现方法的操作，“UserServiceImpl”类程序变为：

```
01  package service.impl;
02  import service.UserService;
03  public class UserServiceImpl implements UserService {
04      public boolean login(String username, String password) {
05          // TODO Auto-generated method stub
06          return false;
07      }
08      @Override
09      public String toString() {
10          // TODO Auto-generated method stub
11          return super.toString();
12      }
13  }
```

在其中第 8 行使用“@Override”注解标明下面的 toString 方法是重写的方法。

2.2.5　生成 Getter 和 Setter 方法

在进行 Java Web 项目开发中，JavaBean 是不可缺少的一部分。在标准的 JavaBean 中，应该具有属性，以及这些属性的 Getter 和 Setter 方法。在本小节中，就来学习一下如何自动生成 Getter 和 Setter 方法。

在 HelloWorld 项目中，创建名称为“po”的包，然后在该包下创建 Student 学生类，并为该类定义了姓名、年龄和性别三个属性，它的程序代码为：

```
01    package po;
02    public class Student {
03        private String name;
04        private int age;
05        private boolean sex;
06    }
```

因为设置和获取 JavaBean 中的属性值都是通过 Setter 和 Getter 方法完成的，所以通常将属性设置为 private 私有访问修饰符。

在编辑区的右键弹出菜单中，选择“Source”|“Generate Getters and Setters”命令，就会弹出生成 Getter 和 Setter 方法的界面，如图 2-16 所示。

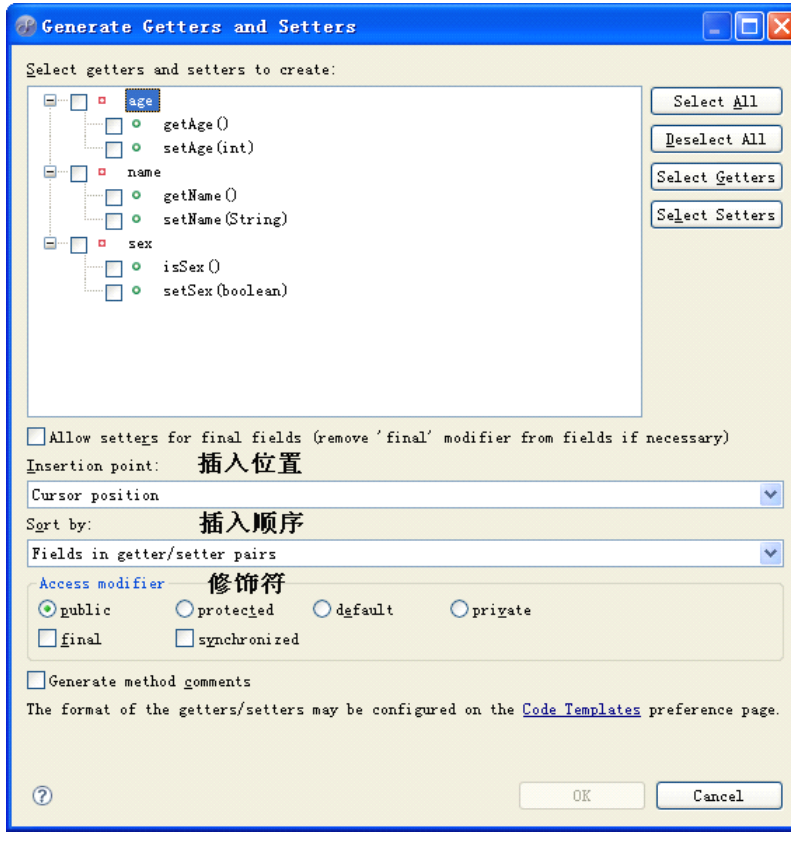


图 2-16　生成 Getter 和 Setter 方法

注意：在进行“Generate Getters and Setters”命令之前，要指定 Getter 和 Setter 方法插入的位置，必须在类中。如果鼠标放在类后，或者其他不能插入方法的位置，是不能使用“Generate Getters and Setters”命令进行操作的。

在生成 Getter 和 Setter 方法界面中的最上面的文本域中，可以选择要自动生成的方法。如果选择属性前面的复选框，则表示该属性的 Getter 和 Setter 方法都生成。在文本域的右键，也有对 Getter 和 Setter 方法进行操作的按钮。“Select All”按钮表示选中所有属性，从而将所有属性的 Getter 和 Setter 方法生成，这也是最常见的操作。“Deselect

All”按钮表示全部不选。“Select Getters”按钮表示选择 Getter 方法，也就是将所有属性的 Getter 方法生产。“Select Setter”按钮表示将所有属性的 Setter 方法生成。

在生产 Getter 和 Setter 方法界面的下半部分中，“Insertion point”表示生成的 Getter 和 Setter 方法插入的位置，在其中有三个选项，默认的是“Cursor position”，表示在鼠标光标位置插入，也就是在执行“Generate Getters and Setters”命令之前，鼠标的位置。“First mothod”选项表示在所有方法前插入，“Last mothod”选项表示在所有方法之后插入。

“Sort by”表示 Getter 和 Setter 方法的插入顺序。其中“Field in getter/setter pairs”是默认选项，表示依次生成每一个属性的 Getter 和 Setter 方法。“First getters,then setters”选项表示先将所有属性的 Getter 方法生产，再将所有属性的 Setter 方法生产。

“Access modifier”表示生成的 Getter 和 Setter 方法的访问修饰符。在理论上讲它们可以使四种访问修饰符中的任意一种，并且还可以选择 final 和 synchronized 两种非访问修饰符。但是为了让更多的程序使用该程序中的属性，通常就采用默认的 public 公共访问修饰符。

在这里我们选中所有的属性，其他选项采用默认选择，单击“OK”按钮，从而完成生成 Getter 和 Setter 方法的操作。Student 学生类程序变为：

```
01  package po;
02  public class Student {
03      private String name;
04      private int age;
05      private boolean sex;
06      public String getName() {
07          return name;
08      }
09      public void setName(String name) {
10          this.name = name;
11      }
12      public int getAge() {
13          return age;
14      }
15      public void setAge(int age) {
16          this.age = age;
17      }
18      public boolean isSex() {
19          return sex;
20      }
21      public void setSex(boolean sex) {
22          this.sex = sex;
23      }
24  }
```

从代码中也可以看到，所有属性的 Getter 和 Setter 方法自动被生成。

2.2.6 生成代理方法

代理模式是设置模式中非常重要的一种，本小节讲解的内容就要基于代理模式。在特定情况下，客户程序不想或者不能直接引用对象，这时候就要用到代理，通过生成一个代理对象做为客户程序和原对象的中介。

在代理模式中，应该具有三个角色程序。首先是抽象角色，它是真实对象和代理对象所在类所实现的接口或者是继承的父类。真实角色是指原对象，也就是客户程序要实际引用的对象。代理角色是生成的代理类，通过它为客户程序和真实角色之间建立桥梁。

在代理角色程序中，要给出真实角色类的引用，并且给出代理方法，该方法的名称应该和真实角色程序的方法相同，并在其中调用真实角色中的方法。在本节中就来学习如何使用 MyEclipse 自动生成这样的代理方法。

在这里我们单独开发一个项目，名称为“ch2”，再在其中创建“proxy”包。在“proxy”包下首先开发抽象角色程序，这里我们使用接口来实现。

```
01  package proxy;
02  public interface AbstractSubject {
03      public void sayHello();
04  }
```

再来开发实现该接口的真实角色，代码为：

```
01  package proxy;
02  public class RealSubject implements AbstractSubject{
03      public void sayHello() {
04          System.out.println("大家好，我是真实角色");
05      }
06  }
```

接下来就是本小节的重点了，开发代理角色，它同样要实现 AbstractSubject 接口，并在代码中具有真实角色类的引用，先只开发这些代码。

```
01  package proxy;
02  public class ProxySubject implements AbstractSubject {
03      private RealSubject realSubject;    //真实角色引用
04
05  }
```

将鼠标放在第 4 行中，在编辑区右键弹出菜单中，选择“Source”|“Generate Delegate Methods”命令，将会弹出选择代理方法的界面，如图 2-17 所示。

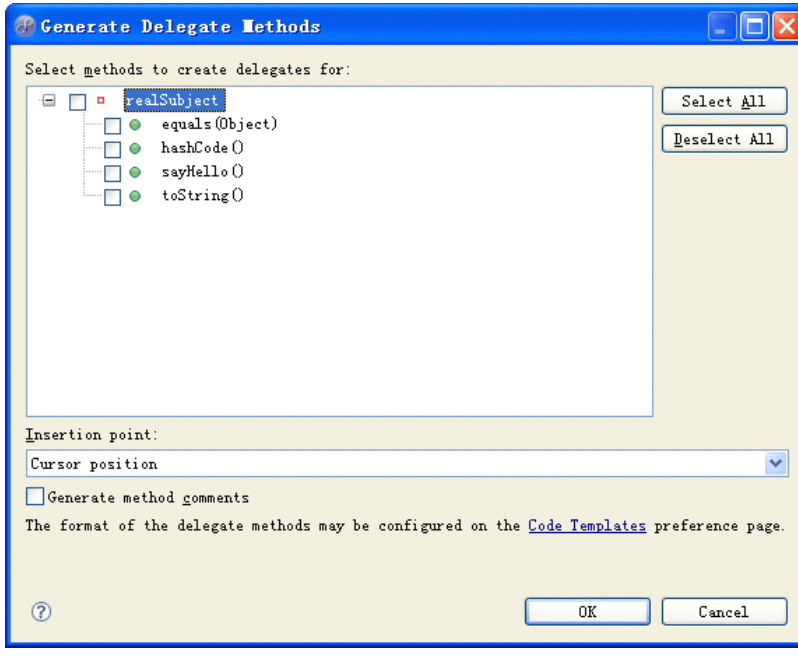


图 2-17　生成代理方法

在生成代理方法界面中，会按照给出的引用，查找该引用类下有哪些方法可以生成代理方法。这里我们选择“sayHello”方法。“Insertion point”表示生成代理方法的插入位置，这里可以选择所有方法前面，所有方法后面和鼠标光盘位置三种，这和生产 Getter 和 Setter 方法相同的。最后单击“OK”按钮，就会自动生成代理方法，则 ProxySubject 代理角色类的代码变为：

```
01    package proxy;
02    public class ProxySubject implements AbstractSubject {
03          private RealSubject realSubject;    //真实角色引用
04           public void sayHello() {
05                 realSubject.sayHello();
06           }
07    }
```

从代码中的变化可以看到，自动生成了一个名称为“sayHello”的方法，并在其中使用真实角色引用调用“sayHello”方法。到目前代理方法并没有开发完，我们要判断真实角色引用是否已经有对象，也就是是否已经注入对象，具体如何注入，这里就不再详细讲解。这里我们通过直接判断的方式，将 ProxySubject 代理角色类的代码变为：

```
01    package proxy;
02    public class ProxySubject implements AbstractSubject {
03          private RealSubject realSubject;    //真实角色引用
04           public void sayHello() {
05                 if(realSubject==null){
06                       realSubject=new RealSubject();
07                 }
08                 realSubject.sayHello();
```

```
09          }
10      }
```

从而已经将代理角色程序开发完成。最后就是通过编写一个客户端程序，来验证代理角色程序是否起到代理的功能。

```
01  package proxy;
02  public class TestProxy {
03          public static void main(String[] args) {
04                  AbstractSubject abstractSubject = new ProxySubject();//代理对象
05                  abstractSubject.sayHello();
06          }
07  }
```

运行上述代码，运行结果为：

```
大家好，我是真实角色
```

从程序和运行结果中可以看到，在客户端程序中，虽然并没有使用真实角色程序引用或者对象，但是仍然运行了真实角色程序中的方法，这就是代理角色中的代理方法起到的作用。

说明：在 Spring 的框架中，使用代理模式是非常多的。在其中的代理角色中并不需要创建真实角色对象，而是通过 IoC 技术。在代理方法中，还可以添加其他方法，从而完善原方法，使用它最多的就是 Spring 中的 AOP 技术。这里我们在后面的学习中都会学习到。

2.2.7 生成 hashCode 和 equals 方法

hashCode 和 equals 方法是 Object 类中的两个方法，所以在所有类中都可以重写这两个方法。在程序中，调用 hashCode 方法将返回对象的哈希码值。在 Java 应用程序执行期间，在同一对象上多次调用 hashCode 方法时，必须一致地返回相同的整数。

equals 方法用于判断对象是否和参数对象“相等”，这里的相等可以人为指定，例如规定姓名和年龄都相等的学生是相等时。在重写 equals 方法时，通常有必要重写 hashCode 方法，以维护 hashCode 方法的常规协定，该协定声明相等对象必须具有相等的哈希码。

在 MyEclipse 中，定义了相应命令，可以自动按照程序员的规定来自动生成 hashCode 和 equals 方法。例如开发一个 Student 学生类程序，并定义了姓名和年龄属性，代码为：

```
01  package hashcodeequals;
02  public class Student {
03          String name;
04          int age;
05
06  }
```

在第 5 行中，在编辑区右键弹出菜单中，选择“Source”|“Generate hashCode() and equals()”命令，将弹出生成 hashCode 和 equals 方法的界面，如图 2-18 所示。

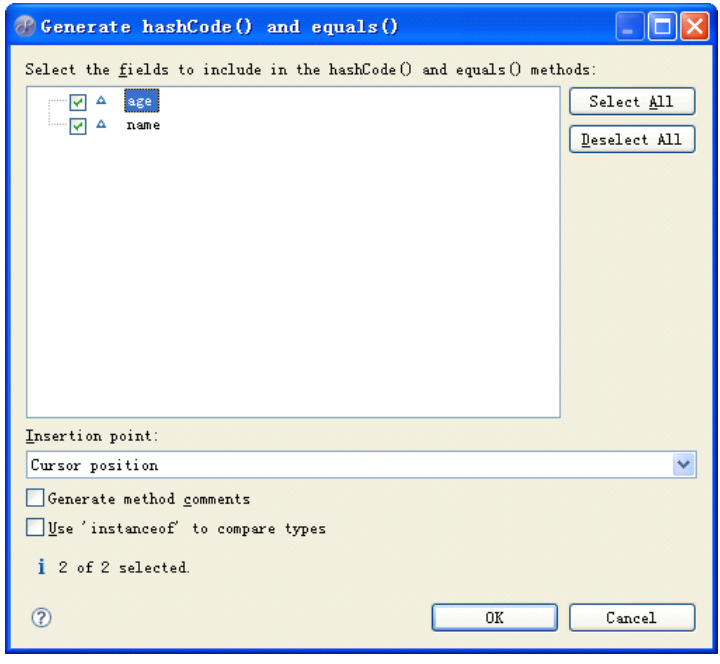


图 2-18　生成 hashCode 和 equals 方法

在生成 hashCode 和 equals 方法界面的文本域中，选择对哪些属性操作，也就是说当哪些属性相同时，认为两个对象是相等的。在“Insertion point”选项是选择生成的方法插入的位置，和前面两小节中讲解是相同的。最后单击“OK”按钮，则 Student 学生类程序代码变为：

```
01  package hashcodeequals;
02  public class Student {
03      String name;
04      int age;
05      @Override
06      public int hashCode() {              //保证相等的对象哈希码值相等
07          final int prime = 31;
08          int result = 1;
09          result = prime * result + age;
10          result = prime * result + ((name == null) ? 0 : name.hashCode());
11          return result;
12      }
13      @Override
14      public boolean equals(Object obj) {
15          if (this == obj)            //如果两个对象相同，则相等
16              return true;
17          if (obj == null)            //如果对象为空，则不相等
18              return false;
19          if (getClass() != obj.getClass())    //如果 Class 不同，则不相等
20              return false;
21          final Student other = (Student) obj;
```

```
22                  if (age != other.age)                //如果年龄不相等，则对象不相等
23                      return false;
24                  if (name == null) {                  //如果姓名为 null
25                      if (other.name != null)          //而参数对象姓名不为 null 则不相等
26                          return false;
27                  } else if (!name.equals(other.name))     //如果姓名不相等，则不相等
28                      return false;
29                  return true;                 //全符合条件，两个对象相等
30              }
31      }
```

其中的注释是后加的，为了让读者更容易的看懂代码。从代码的变化上可以看到自动生成了 hashCode 和 equals 方法，它们要比程序员自己写的代码规范的。

注意：在重写 equals 方法时，一定要进行相应的重写 hashCode 方法。这在使用 MyEclipse 开发工具时是不会出错的，因为将两个方法绑定在一起。如果手动重写时，不重写 hashCode 方法，在特定情况下就会出现错误，例如学习 HashSet、HashMap 等集合时。

2.2.8 生成构造函数

构造函数是 Java 程序中非常重要的组成部分，如果没有人为的定义构造函数，系统会自动给出一个无参构造函数。在 MyEclipse 中，生成构造函数有两种方式，一种是使用属性生成，另一种是从父类中生成。

1.使用属性生成

使用属性生成构造函数，是通过使用已有属性做为构造函数参数来生成构造函数。在本章项目中，在创建一个“constructor”包，在其中创建 Student 学生类程序，并在其中定义姓名和年龄两个属性，代码为：

```
01  package hashcodeequals;
02  public class Student {
03          String name;
04          int age;
05
06  }
```

将鼠标放在第 5 行，在右键弹出菜单中，选择“Source”|“Generate Constructor using Fields”命令，从而弹出使用属性生成构造函数界面，如图 2-19 所示。

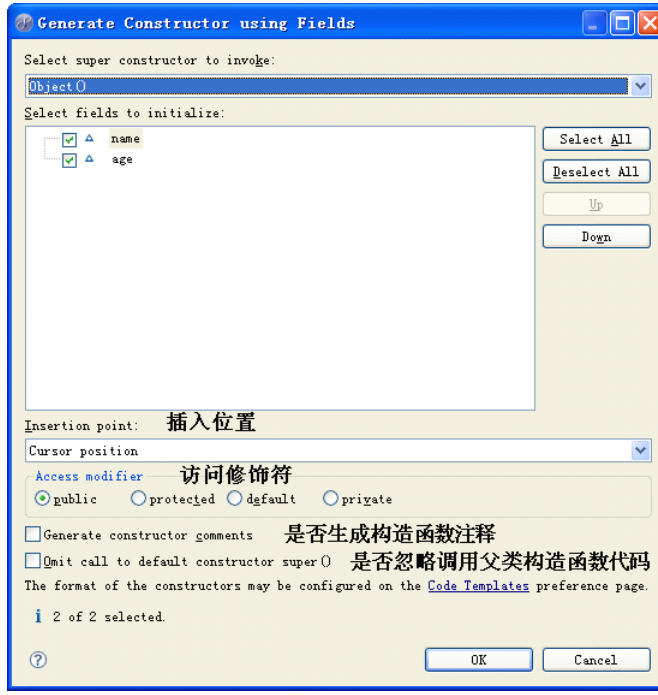


图 2-19　使用属性生成

在使用属性生成构造函数的界面中，在上半部分的文本域中，选择要使用的属性，默认是全部选择，从而生成具有所有属性的构造函数。也可以选择其中一部分属性，甚至不选择任何属性，这样生成的是无参构造函数。

在文本域的右边有四个按钮。“Select All”表示选择全部属性，“Deselect All”表示全部不选择。“Up”表示向上移动属性，“Down”表示向下移动属性。之所以要移动属性，是因为在构造函数中参数也是有顺序的。当前情况下，生成的两参构造函数，第 1 个参数是 name，第 2 个参数是 age。如果选中“name”属性，然后单击“Down”按钮，则“name”属性就会出现在“age”属性的下面，则生成的构造函数第 1 个参数是 age，第 2 个参数是 name。这两种生成的构造函数是不相同的。

“Insertion point”选项表示构造函数插入的位置，可以是当前鼠标光标位置，也可以是所有方法前或者所有方法后。“Access modifier”选项表示生成构造函数的访问修饰符，可以在所有四种访问修饰符中选择。

“Generate constructor comments”选项表示是否生成构造函数注释，默认是不生成。“Omit call to default constructor super()”选项表示是否忽略调用父类无参构造函数的代码，也就是“super()”。默认是不忽略，也就是生成“super()”代码。这里我们采用默认选择，单击“OK”按钮，从而完成使用属性生成构造函数的操作，则 Student 学生类程序变为：

```
01    package constructor;
02    public class Student {
03        String name;
04        int age;
05        public Student(String name, int age) {
```

```
06                  super();
07                  this.name = name;
08                  this.age = age;
09          }
10  }
```

注意：不管是否生成“super()”代码，是都会调用父类无参构造函数的，除非指定调用有参构造函数。从这里也可以看到，在生成有参构造函数后，最好再给出无参构造函数。因为有参构造函数会将系统自动生成的无参构造函数覆盖，但是在子类中会自动调用父类无参构造函数，如果没有无参构造函数，程序就会出错。

2.从超类中生成

从超类中生成构造函数是指在子类中选择生成和父类中相同的构造函数，例如在父类中具有 3 种构造函数，那么在子类中就可以选择生成其中的某几种。通过这种方式生成的构造函数是和父类中的构造函数具有相同参数个数、类型和顺序的。

我们先来开发一个 Person 类，让它做为父类，在其中具有姓名和年龄两个属性，并具有三种构造函数，代码为：

```
01  package constructor;
02  public class Person {
03          String name;
04          int age;
05          public Person() {
06                  super();
07          }
08          public Person(String name) {
09                  super();
10                  this.name = name;
11          }
12          public Person(String name, int age) {
13                  super();
14                  this.name = name;
15                  this.age = age;
16          }
17  }
```

然后开发一个 Teacher 老师类，让它继承 Person 类，并也为 Teacher 老师类定义了姓名和年龄两个属性，当前代码为：

```
01  package constructor;
02  public class Teacher extends Person {
03          String name;
04          int age;
05
06  }
```

将鼠标放在第 5 行中，在右键弹出菜单中，选择“Source”|“Generate Constructor from Superclass”命令，会弹出从超类中生成构造函数的界面，如图 2-20 所示。

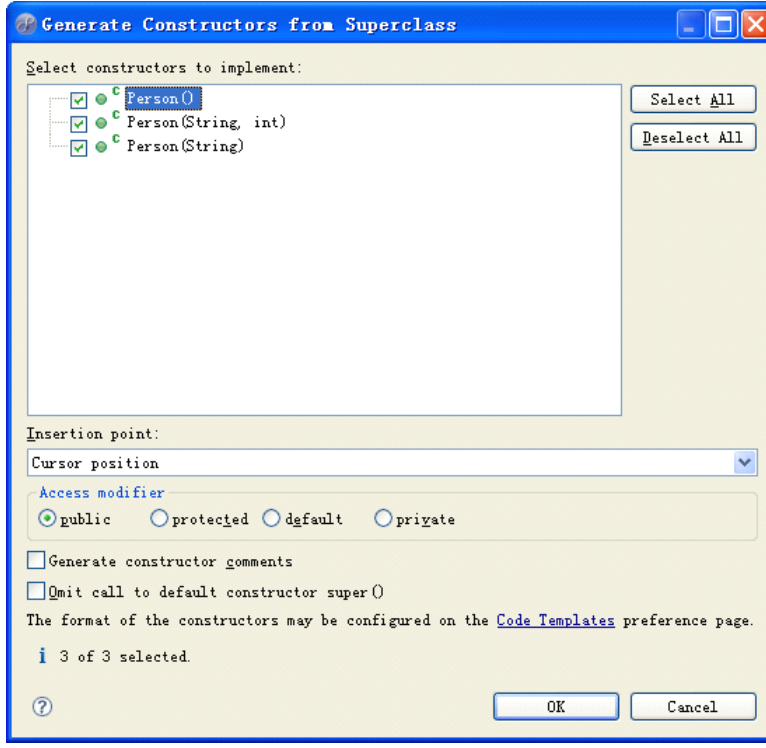


图 2-20　从超类中生成

该界面和通过属性生成构造函数界面相比，只有上半部分不同。在文本域中将父类中三种类型的构造函数列出来，通过选择，可以生成 Teacher 类中的构造函数。例如我们选择全部三种构造函数，单击“OK”按钮，则 Teacher 老师类的代码变为：

```
01  package constructor;
02  public class Teacher extends Person {
03      String name;
04      int age;
05      public Teacher() {
06          super();
07          // TODO Auto-generated constructor stub
08      }
09      public Teacher(String name, int age) {
10          super(name, age);
11          // TODO Auto-generated constructor stub
12      }
13      public Teacher(String name) {
14          super(name);
15          // TODO Auto-generated constructor stub
16      }
17  }
```

自动生成的三种构造函数的顺序变化是没有关系的。通过这种方式生成的构造函数和父类的具有相同的参数个数、类型和顺序。例如父类中的两参构造函数中，是先 name 后 age，则在子类中同样也是这样的顺序。

注意：我们完全可以在 Teacher 老师类中不定义姓名和年龄属性，也能生成这三种构造函数。这是因为这两个属性会从父类中继承下来，这里我们给出了这两个属性，是因为可以让构造函数看的更明白。

2.3 包围方式

在 Java 语言的流程控制语句中，大部分语句都是具有固定格式的。在控制语句中，包括条件语句、循环语句和异常处理语句等。在 MyEclipse 中，通过命令操作可以自动生成这些语句，从而减轻程序员的操作。在菜单中，选择“Source”|“Surround With”命令，可以看到包围方式选项。在编辑区的右键弹出菜单中，选择“Surround With”命令也可以得到包围方式选项。

记忆：在进行包围操作时，一定首先选择被包围的代码。包围方式的快捷键为“Alt+Shift+Z”。

2.3.1 条件语句包围

在 Java 的条件语句中包括两种，分别是 if 条件语句和 Switch 多条件分支语句。由于 Switch 语句是比较复杂的，所以在 MyEclipse 中并没有集成自动生成该语句的命令。所以这里说的条件语句包围就是 if 条件语句包围。

这里通过下面的代码讲解条件语句包围命令。

```
01  package surround;
02  public class IfTest {
03      public static void main(String[] args) {
04          System.out.println("自动生成 if 条件语句");
05      }
06  }
```

选中第 4 行的代码，在编辑区的右键弹出菜单中，选择“Surround With”|“3 if (if statement)”命令，则代码自动变为：

```
01  package surround;
02  public class IfTest {
03      public static void main(String[] args) {
04          if (condition) {
05              System.out.println("自动生成 if 条件语句");
06          }
07      }
08  }
```

然后将第 4 行的“condition”换为实际需要的条件表达式，就完成 if 条件语句的包围。

2.3.2 循环语句包围

在 Java 的循环控制语句中，包括 for 循环语句、while 循环语句和 do-while 循环语句，它们分别使用不同的命令。

1.for 循环语句

首先开发一个类名为“ForTest”的程序，代码如下：

```
01  package surround;
02  public class ForTest {
03      public static void main(String[] args) {
04          System.out.println("自动生成 for 循环语句");
05      }
06  }
```

选中第 4 行的代码，在编辑区的右键弹出菜单中，选择“Surround With”|“2 for(iterate over array)”命令，则代码自动变为：

```
01  package surround;
02  public class ForTest {
03      public static void main(String[] args) {
04          for (int i = 0; i < args.length; i++) {
05              System.out.println("自动生成 for 循环语句");
06          }
07      }
08  }
```

从自动生成的代码可以看到，是从 0 开始，然后到 args 参数长度结束。程序员可以根据自己实际需要进行改动。

2.while 循环语句

首先开发一个类名为“WhileTest”的程序，代码如下：

```
01  package surround;
02  public class WhileTest {
03      public static void main(String[] args) {
04          System.out.println("自动生成 while 循环语句");
05      }
06  }
```

选中第 4 行代码，在编辑区中的右键菜单中，选择“Surround With”|“7 while(while loop with condition)”命令，则代码自动变为：

```
01  package surround;
02  public class WhileTest {
03      public static void main(String[] args) {
04          while (condition) {
05              System.out.println("自动生成 while 循环语句");
06          }
07      }
08  }
```

将其中第 4 行的“condition”换为和实际需求相对应的条件表达式，则自动生成 while

循环语句就完成了。

3.do-while 循环语句

首先开发一个类名为“DoWhileTest”的程序，代码如下：

```
01  package surround;
02  public class DoWhileTest {
03      public static void main(String[] args) {
04          System.out.println("自动生成 while 循环语句");
05      }
06  }
```

选中其中的第 4 行，在编辑区的右键弹出菜单中，选择“Surround With”|“1 do(do while statement)”命令，则代码自动变为：

```
01  package surround;
02  public class DoWhileTest {
03      public static void main(String[] args) {
04          do {
05              System.out.println("自动生成 while 循环语句");
06          } while (condition);
07      }
08  }
```

然后将第 6 行的“condition”换为和实际需求相对应的条件表达式，则自动生成 do-while 循环语句完成。

2.3.3 异常处理语句包围

异常处理语句也是流程控制语句的一种，在很多书上之所以把它放在后面讲解是因为异常的概念要放在后面讲解。在 MyEclipse 中，有两种异常处理语句包围方式，它们之间有所区别。

1.简单异常处理语句包围

异常处理语句由 try 语句和 catch 语句组成，在 catch 语句中给出异常类型，如果使用简单异常处理语句，则直接给出 Exception。先来开发一个类名为“TryCatchTest”的程序，代码为：

```
01  package surround;
02  import java.net.ServerSocket;
03  public class TryCatchTest {
04      public static void main(String[] args) {
05          ServerSocket socket=new ServerSocket(8888);
06          System.out.println("自动生成异常处理语句");
07      }
08  }
```

从代码中可以看到在第 5 行创建服务器端套接字，该语句会发生 IOException 异常。选中第 5 行代码，在编辑区的右键弹出菜单中，选择“Surround With”|“6 try(try catch block)”命令，则代码自动变为：

```
01  package surround;
```

```
02  import java.net.ServerSocket;
03  public class TryCatchTest {
04      public static void main(String[] args) {
05          try {
06              ServerSocket socket = new ServerSocket(8888);
07          } catch (Exception e) {
08              // TODO: handle exception
09          }
10          System.out.println("自动生成异常处理语句");
11      }
12  }
```

从运行结果中可以看到在自动生成的异常处理 catch 语句中，使用 Exception 异常进行判断，从而进行异常处理。

2.匹配异常处理语句包围

匹配异常处理语句包围是一种更智能的处理方式，使用它可以判断具体选中代码所发生的异常，然后使用对异常进行处理。仍然选中原“TryCatchTest”类程序中的第 5 行，在编辑区的右键弹出菜单中，选择“Surround With”|“Try/catch Block”命令，则代码自动变为：

```
01  package surround;
02  import java.io.IOException;
03  import java.net.ServerSocket;
04  public class TryCatchTest {
05      public static void main(String[] args) {
06          try {
07              ServerSocket socket=new ServerSocket(8888);
08          } catch (IOException e) {
09              // TODO Auto-generated catch block
10              e.printStackTrace();
11          }
12          System.out.println("自动生成异常处理语句");
13      }
14  }
```

在自动生成的第 8 行中可以看到，使用 IOException 异常进行处理。

说明：有些读者可能认为匹配异常处理语句要优于简单异常处理，这是不全面的答案。具体使用哪一种要看实际的情况。如果仅仅是让程序能够运行，而且还要在其中加入其它代码，这时候就应该采用简单异常处理语句包围，这样不用进行多 catch 语句处理。

2.3.4 多线程语句包围

在多线程的操作中，线程方法和同步块都是经常使用到的，线程方法就是 run 方法，同步块是使用 synchronized 关键字定义，这些都是有固定格式的。在 MyEclipse 中，也定义了相关命令生成这些内容。

1.同步块

先来看一下同步块的自动生成。首先定义一个类名为“SynchronizedTest”的程序，代码为：

```
01  package surround;
02  public class SynchronizedTest {
03      public static void main(String[] args) {
04          System.out.println("自动生成多线程");
05      }
06  }
```

选中其中的第 4 行，在编辑区的右键弹出菜单中，选择“Surround With”|“synchronized (synchronized block)”命令，在上述代码变为：

```
01  package surround;
02  public class SynchronizedTest {
03      public static void main(String[] args) {
04          synchronized (args) {
05              System.out.println("自动生成多线程");
06          }
07      }
08  }
```

然后将生成的第 4 行代码中的“args”换成实际需求的资源对象，则该同步块就开发完成。这里我们主要就是看同步块语句包围方式的使用，在实际开发中，是比较上的将同步块定义在 main 方法中的。

2.线程方法

在 Java 语言中，线程方法是一种固定方法名的方法，方法名为“run”。在 MyEclipse 中，也可以使用命令自动将某些功能代码放到多线程方法中。这里我们仍然以原“SynchronizedTest”类程序举例。

选中其中的第 4 行代码，在编辑区的右键弹出菜单中，选择“Surround With”|“runnable (runnable)”命令，则代码变为：

```
01  package surround;
02  public class SynchronizedTest {
03      public static void main(String[] args) {
04          new Runnable() {
05              public void run() {
06                  System.out.println("自动生成多线程");
07              }
08          }
09      }
10  }
```

在其中会自动生成一个多线程内部类，在内部类中定义 run 线程运行方法，从而将选中的功能代码放置在其中。这样就创建了一个实现 Runnable 接口的线程对象，我们可以将该对象封装到 Thread 线程对象中再调用 start 方法，从而启动线程。

注意：在实际开发中，多线程语句包围使用的是比较少的。不是因为功能不强大，

是因为这些语句在开发中使用时非常灵活的，自动生成后，需要程序员进行比较多的改动。程序员通常选择自己编写所有相关代码。

第3章　重构

重构是对软件程序的一种调整，使用重构的目的就是在不改变软件可察行为的前提下，调整软件结构，从而降低修改成本。重构是软件工程中非常重要的一个概念，只讲重构的知识点，500 页的书都不能全部包括，在本书中主要来学习 MyEclipse 中支持的对重构的操作。

由于重构的内容比较多，我们按照重构的对象不同，将重构分为物理结构重构、类内部重构、类层次上重构和引入重构。

记忆：在 MyEclipse 的菜单中，“Refactor”菜单中包含了所有重构操作。在编辑区右键弹出菜单中，重构的快捷键为“Alt+Shift+T”。这两种方式都可以进行重构操作，但是本书中的重构主要通过常用的包资源管理器节点的方式进行操作。

3.1　物理结构重构

物理结构重构是最简单的、最容易被理解的重构方式，包括重命名、移动、改变方法修饰符等操作。

3.1.1　重命名

如果读者使用的是 Windows 操作系统，那么对于重命名操作是不陌生的。通过进行重命名操作，可以改变文件的名称，但是当该文件被其他程序使用时，是不能进行修改的。

在 MyEclipse 中可以对类名、接口名，甚至是方法名、变量名进行重命名操作，而且在使用或者调用这些名称的地方都会随着改变。这里我们通过类名来讲解重命名的操作。

说明：重命名操作的对象不同，所经过的操作步骤是不同的，但是操作流程是非常类似的。读者可以学习完对类名的重命名后，对其他的也进行操作。

例如创建两个 Java 类程序，分别是 Student 学生类和 Test 测试类，先来看 Student 学生类代码：

```
01  package refactor;
02  public class Student {
03      String name;
04      int age;
05      public Student() {
06      }
07      public Student(String name, int age) {
```

```
08              this.name = name;
09              this.age = age;
10          }
11          public void say(){
12              System.out.println("我的名称为"+name+"；年龄为："+age);
13          }
14      }
```

再来看 Test 测试类的代码：

```
01  package refactor;
02  public class Test {
03          public static void main(String[] args) {
04              Student stu=new Student("Tom",21);
05              stu.say();
06          }
07  }
```

接下来我们就对 Student 学生类的类名进行重命名操作。首先我们来看一下当前项目的包资源管理器，如图 3-1 所示。

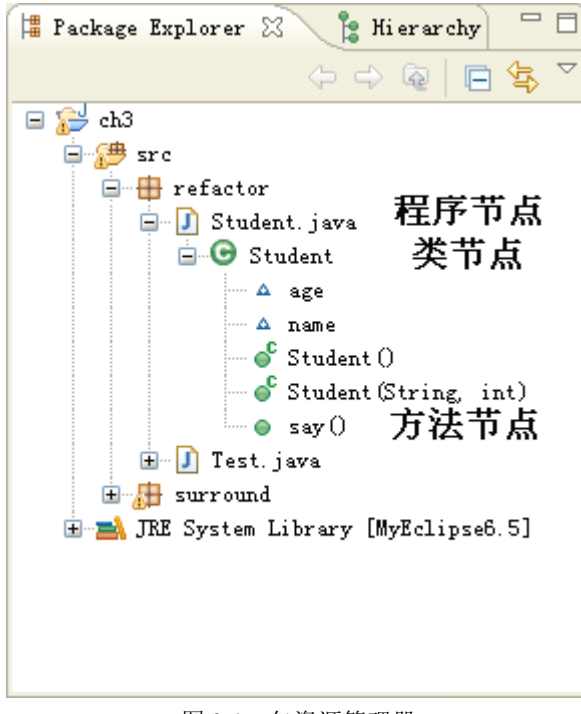


图 3-1　包资源管理器

该包资源管理器的显示，可能和读者的有所不同。在这里只需要单击节点前面的加号，就可以展开。例如在程序节点下可以有多个类节点，在类节点下可以有多个变量节点、构造函数节点和方法节点。

在进行重命名操作时，就在这些节点上操作。选中 Student 类节点，单击鼠标右键，在弹出菜单中，选择“Refactor”|“Rename”命令，将弹出重命名界面，如图 3-2 所示。

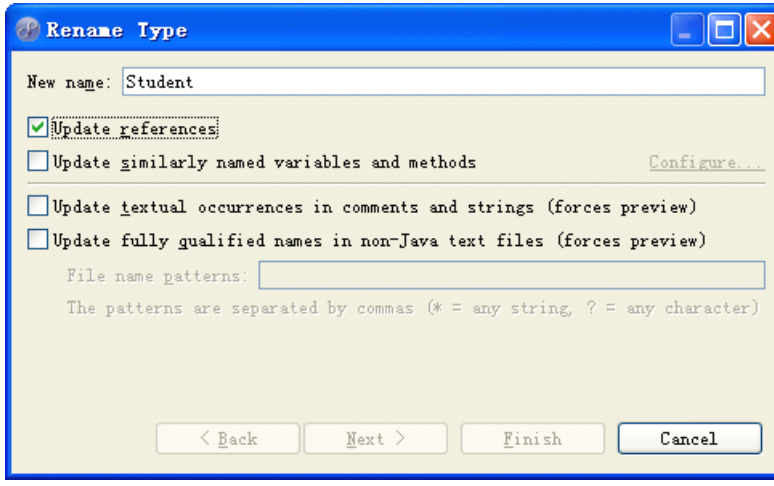


图 3-2　重命名类名

在重命名类名中，“New name”选项是用来填写重命名后的新类名，这里填写“Teacher”。“Update references”选项表示是否更新引用，默认是选中的状态，表示将引用该类的地方同时重命名。“Update similarly named variables and methods”选项表示是否更新以类似方式指定的变量和方法，默认是不选中的状态。下面两个选项是用来判断是否更新注释和非 Java 程序，通常是不选择这两项的。

按照上述操作后，可以单击“Finish”按钮完成操作，也可以单击“Next”按钮进入预览界面，如图 3-3 所示。

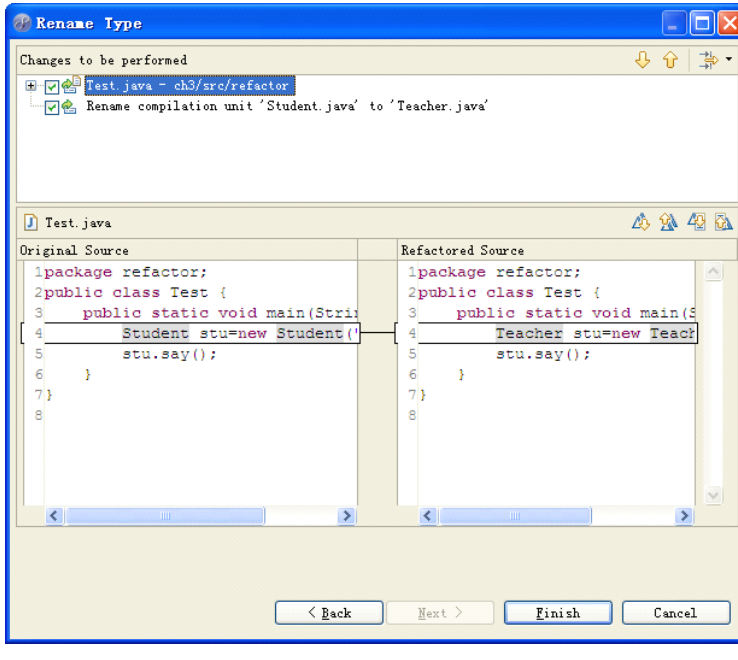


图 3-3　重命名预览

在重命名预览界面中，“Changes to be performed”栏中会出现有哪些程序发生了改变。在其中进行选择，会将改变的细节显示在界面的下半部分，包括原代码内容。查看

重命名的预览没有问题后，单击“Finish”按钮，从而完成类名的重命名。操作后的 Student 学生类代码变为：

```
01  package refactor;
02  public class Teacher {
03      String name;
04      int age;
05      public Teacher() {
06      }
07      public Teacher(String name, int age) {
08          this.name = name;
09          this.age = age;
10      }
11      public void say(){
12          System.out.println("我的名称为"+name+"；年龄为："+age);
13      }
14  }
```

从代码中可以看到，不但类名由“Student”变为了“Teacher”，而且其中的构造函数也随着变化。再来看使用 Student 学生类的 Test 类程序代码的变化：

```
01  package refactor;
02  public class Test {
03      public static void main(String[] args) {
04          Teacher stu=new Teacher("Tom",21);
05          stu.say();
06      }
07  }
```

从代码中可以看到，使用 Student 类的地方也由“Student”变为了“Teacher”，这里体现出了重命名操作的强大之处。但是类引用时不会改变的，例如“stu”是不会变化的，如果程序员觉得该引用名不合适，需要手动修改。

技巧：上述是通过包资源管理器中的节点进行重命名操作的，该操作也可以在编辑区中完成。选中要进行操作的类名和方法名，在右键弹出菜单中，选择“Refactor”|“Rename”命令，就进入重命名操作界面，从而进行相应操作。

记忆：重命名操作的快捷方式为“Alt+Shift+R”，使用该快捷键可以直接进入重命名操作页面。

3.1.2 移动

在 Windows 操作系统中，可以对文件进行剪切操作。在文字编辑工具中，也可以对文字等信息进行剪切操作。在 MyEclipse 的重构操作中，也有类似的操作，通常将它称为移动。

说明：和移动类似的操作，那就是复制粘贴。例如在 A 包下复制一个 Java 程序，然后选中 B 包后粘贴，这样就把这个 Java 程序复制了一份放到 B 包中。这是比较简单的，这里就不再进行讲解。

移动操作的对象和重命名操作一样，可以对类、接口、方法等进行操作，同样也会进行相应的改变。通常类和接口都会放在包下，当移动到其他包中后，则包语句代码会自动变化，这是比较简单的。在本小节中，主要来学习一下方法的移动。

由于在上一小节中已经将 Student 学生类重命名为 Teacher 老师类，所以这里我们可以再创建一个 Student 学生类程序，其代码为：

```
01  package refactor;
02  public class Student {
03      public static void sayHello(){
04          System.out.println("大家好，我是一个被移动的方法");
05      }
06      public static void main(String args[]){
07          Student.sayHello();
08      }
09  }
```

在包资源管理器中，选中“Student”类节点下的“sayHello”方法节点，单击鼠标右键，在弹出的右键菜单中，选择“Refactor”|“Move”命令，将弹出移动操作界面，如图 3-4 所示。

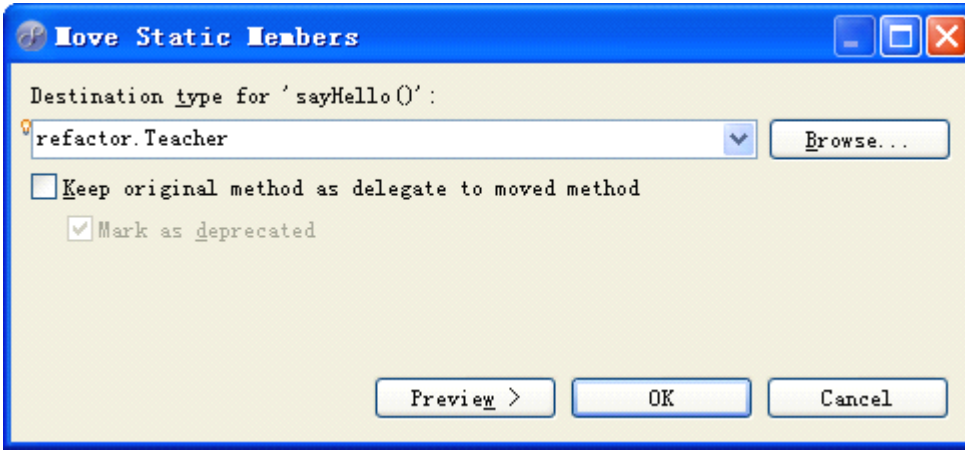


图 3-4　移动操作

单击其中的“Browse”按钮，将弹出填写和选择移动目标的界面，如图 3-5 所示。

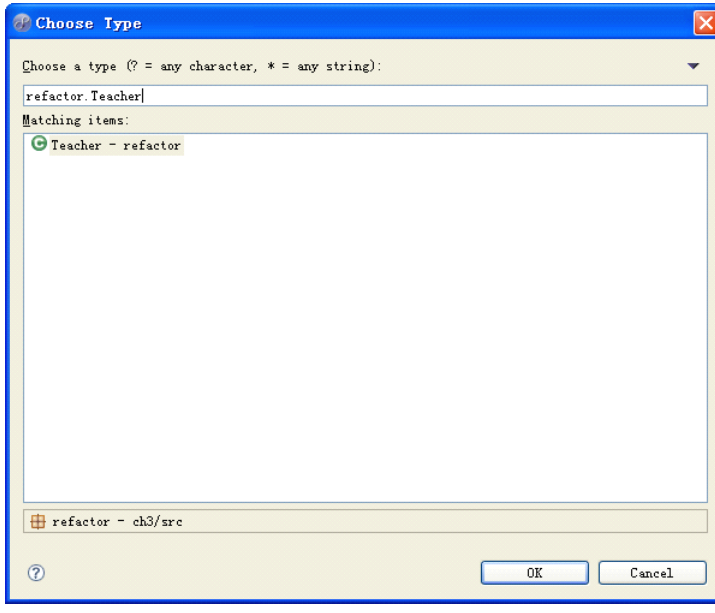


图 3-5　填写移动目标

在其中输入要移动的类，如果该类出现在多个包下，还要在下面的文本域中选择某一个包下的类。例如这里我们选择和当前 Student 学生类在同一包（refactor 包）下的 Teacher 类。单击“OK”按钮，从而完成移动目标的选择。

经过上面的操作，将回到移动操作界面中，在其中还可以单击“Preview”按钮，看一下移动操作的预览，如图 3-6 所示。

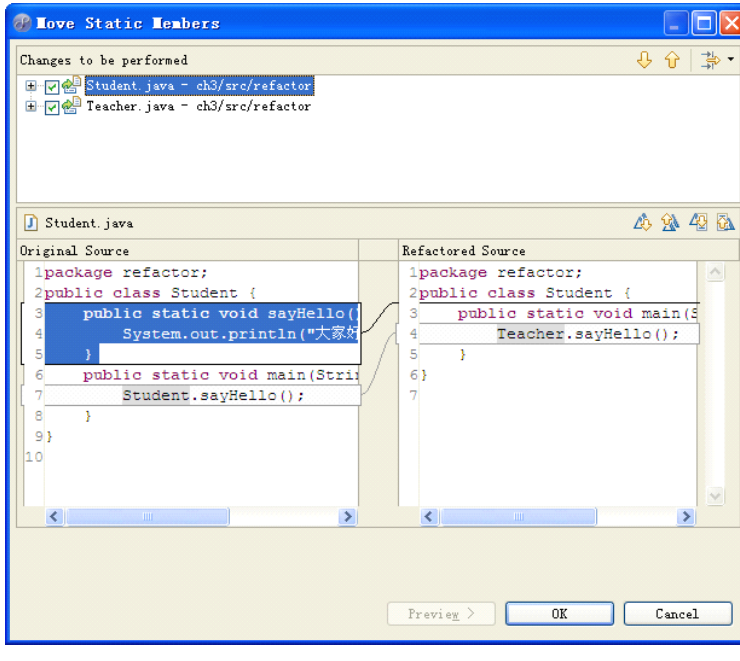


图 3-6　移动操作预览

在其中可以看到经过移动操作变化的程序，已经代码的具体变化，单击“OK”按

钮，完成预览的观看，并完成移动操作。这时候，Teacher 老师类中的代码变为：

```
01  package refactor;
02  public class Teacher {
03      String name;
04      int age;
05      public Teacher() {
06      }
07      public Teacher(String name, int age) {
08          this.name = name;
09          this.age = age;
10      }
11      public void say(){
12          System.out.println("我的名称为"+name+"；年龄为："+age);
13      }
14      public static void sayHello(){
15          System.out.println("大家好，我是一个被移动的方法");
16      }
17  }
```

可以看到在 Teacher 老师类中，多出了一个“sayHello”方法，这是因为我们将“sayHello”方法的移动目标定为 Teacher 老师类。再来看一下 Student 学生类程序代码的变化。

```
01  package refactor;
02  public class Student {
03      public static void main(String args[]){
04          Teacher.sayHello();
05      }
06  }
```

从代码中可以看到，Student 学生类中的“sayHello”方法已经被移动。在 main 方法中，原来由 Student 类调用的“sayHello”方法，已经变为 Teacher 老师类进行调用了，该方法的调用不在 Student 类中，也是不会受到影响的，从而不影响功能的实现。

注意：进行移动操作的方法只能是 static 修饰符修饰的静态方法，这样才能够做到移动后能够通过类名进行调用，从而不影响原有功能。

注意：在进行大部分重构操作后，都可以进行撤销操作，快捷键为“Ctrl+Z”，它不是仅仅撤销当前 Java 程序，而是将重构操作中所有改变的操作都撤销。

记忆：移动操作的快捷键为“Alt+Shift+V”，使用该快捷键可以直接进入移动操作界面中。

3.1.3 更改方法签名

在 Java 程序中，方法中的元素是比较多的，例如修饰符、返回类型、方法名、参数和抛出异常等。在 MyEclipse 中，可以通过界面化的操作，对这些元素进行更改。

在包资源管理器中，选中要更改的方法节点，例如这里我们更改移动后的“sayHello”方法。单击鼠标右键，在右键弹出菜单中，选择“Refactor”|“Change Method Signature”命令，就会进入更改方法签名界面，如图 3-7 所示。

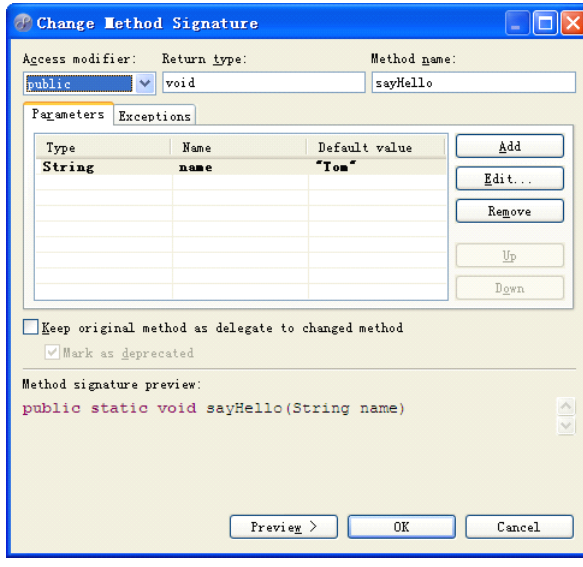


图 3-7　更改方法签名

其中“Access modifier”选项表示访问修饰符，它对应的是一个下拉列表，可以在四种访问修饰符中选择。“Return type”选项表示方法的返回类型，“Method name”选项表示方法名称。

接下来有两个选项卡，分别是“Parameters”和“Exceptions”。“Parameters”选项表示方法的参数，这里可以填写多个，也可以调整参数顺序。单击“Add”按钮添加一个参数后，可以调整该参数的类型、名称和默认值，分别由“Type”、“Name”和“Default value”对应。“Exceptions”选项表示方法的抛出异常类型，也是可以选择多个的。

在更改方法签名界面中，“Methon signature preview”选项表示方法的简单预览。如果想查看详细的预览，可以单击“Preview”按钮，预览界面如图 3-8 所示。

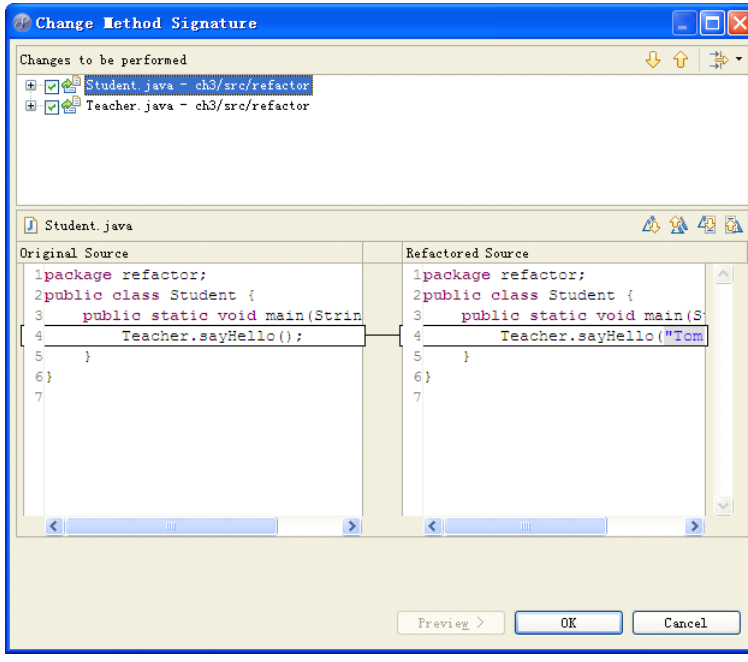


图 3-8　更改方法签名预览

该预览界面是和前面的重构预览界面非常类似的，变化的程序中，不但 Teacher 老师类，还有 Student 学生类，因为在 Student 学生类中调用了更改的方法。单击其中的“OK”按钮，则完成更改方法签名的操作。看下面 Teacher 老师类的代码变化：

```
01    package refactor;
02    public class Teacher {
03          String name;
04          int age;
05          public Teacher() {
06          }
07          public Teacher(String name, int age) {
08                this.name = name;
09                this.age = age;
10          }
11          public void say(){
12                System.out.println("我的名称为"+name+"；年龄为："+age);
13          }
14          public static void sayHello(String name){
15                System.out.println("大家好，我是一个被移动的方法");
16          }
17    }
```

可以看到，Teacher 老师类中的 sayHello 方法已经发生改变，其中多出了一个字符串类型的名称为 name 的参数。我们再来看 Student 学生类程序的变化：

```
01    package refactor;
02    public class Student {
03          public static void main(String args[]){
```

```
04                Teacher.sayHello("Tom");
05            }
06    }
```

可以看到调用该更改方法的代码也发生的改变，给出了一个字符串类型的参数。有些读者可能有这样的疑问，“Tom”参数值是什么地方来的呢？它是在更改方法签名界面中设置参数时，其中的第 3 项“Default value”，就是用来设置调用该方法时，使用的默认值。

注意：更改方法签名时，可能会造成程序的错误。例如我们将 sayHello 方法的访问修饰符该为 private 私有级别，这样在其他程序中调用该方法将发生错误。

记忆：更改方法签名的快捷键为“Alt+Shift+C”，使用该快捷键将直接进入更改方法签名界面。

3.2 类内部重构

类内部重构是指在当前操作类类体中进行重构操作，其中包括抽取、内联和职能转换等操作。在抽取操作中，又可以分为抽取方法、抽取局部变量和抽取常量等具体操作。在职能转换操作中，包括两种内部类的转换和变量间的转换。

3.2.1 抽取方法

在 Java 程序中，通常将完成特定的功能的代码放在一个方法中，这样增加了程序的重用性，也使程序更易读。在项目中，如果在某一方法中有多个功能代码，或者一段代码在多处使用，就可以进行抽取方法操作，将指定的代码放在一个方法中。

先来开发进行操作的“ExtractMethodTest”程序，其代码为：

```
01    package refactor;
02    public class ExtractMethodTest {
03        public void method1(){
04            System.out.println("这是第一个方法");
05            for(int i=0;i<10;i++){
06                System.out.println(i);
07            }
08        }
09        public void method2(){
10            System.out.println("这是第二个方法");
11            for(int i=0;i<10;i++){
12                System.out.println(i);
13            }
14        }
15    }
```

选中从第 5 行到第 7 行的 for 循环代码，在编辑区的右键弹出菜单中，选择“Refactor”|“Extract Method”命令，将弹出抽取方法界面，如图 3-9 所示。

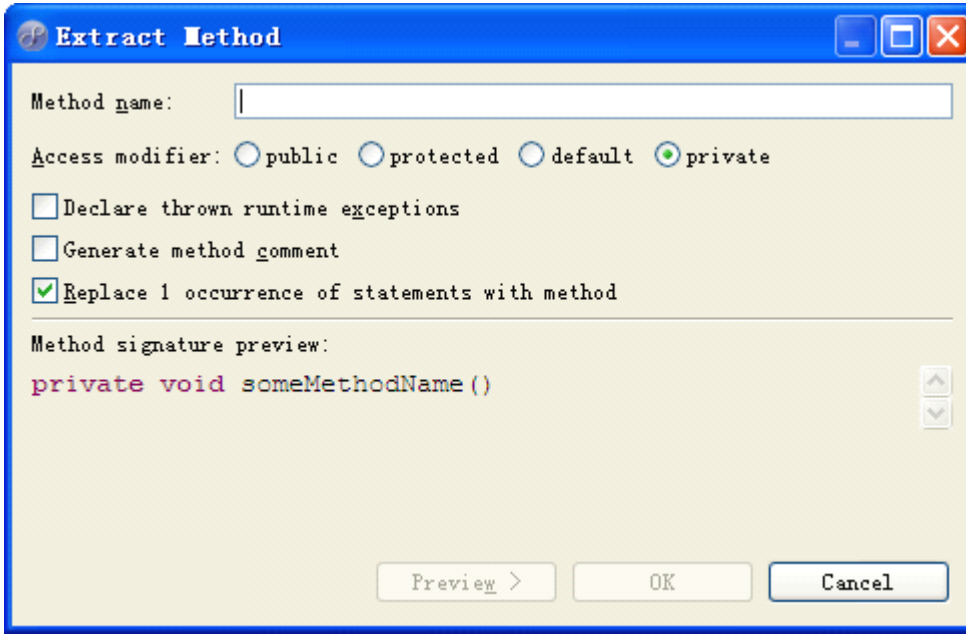


图 3-9　抽取方法界面

其中“Method name”选项用于填写抽取方法后的方法名，“Access modifier”选项用于选择抽取方法的访问修饰符。“Declare thrown runtime exceptions”选项表示是否声明抛出的运行时异常。“Generate method comment”选项表示是否生成注释。“Replace 1 occurrence of statements with method”选项表示将所有的该语句替换成方法，并给出了有几处需要替换。

“Method signature preview”选项表示简单预览，如果想看更详细的代码预览，可以单击“Preview”按钮，读者可以自己查看，这里就不再给出界面。单击“OK”按钮，从而完成抽取方法的操作。“ExtractMethodTest”程序的代码将变为：

```
01  package refactor;
02  public class ExtractMethodTest {
03      public void method1(){
04          System.out.println("这是第一个方法");
05          forMethod();
06      }
07      private void forMethod() {
08          for(int i=0;i<10;i++){
09              System.out.println(i);
10          }
11      }
12      public void method2(){
13          System.out.println("这是第二个方法");
14          forMethod();
15      }
16  }
```

从代码中可以看到，从第 7 行到第 11 行定义了一个名称为“forMethod”的方法，将 for 循环放在该方法中。而在原来的方法中 for 循环代码换为了调用“forMethod”方法。

注意：如果被抽取的代码在静态方法中，则抽取后的方法就是静态方法。同理如果被抽取的代码在非静态方法中，则抽取后的方法就是非静态方法。这样就会产生这样一个问题，如果被抽取代码既出现在静态方法中，又出现在非静态方法中，抽取得到非静态方法后，程序就发生错误，需要手动修改。

记忆：抽取方法操作的快捷键为“Alt|Shift+M”，在编辑区中，选中被抽取的代码，使用该快捷键将自动进入抽取方法的界面。

3.2.2 抽取局部变量

在上一小节中，我们学习了如何将一段功能代码抽取成一个方法，在本小节中就继续来学习如何将一个表达式抽取成局部变量。如果在一个程序中，一个表达式被使用多次，就可以将该表达式抽取成一个局部变量，将使用该表达式的地方都替换为该局部变量。

首先定义了一个类名为“ExtractLocalVariableTest”的程序，它的代码为：

```
01  package refactor;
02  public class ExtractLocalVariableTest {
03      public static void main(String args[]){
04          int i=5;
05          boolean b1=(i==5)&true;
06          boolean b2=(i==5)|false;
07      }
08  }
```

选中其中第 5 行的“i==5”表达式，在右键弹出菜单中，选择“Refactor”|“Extract Local Variable”命令，将弹出抽取局部变量界面，如图 3-10 所示。

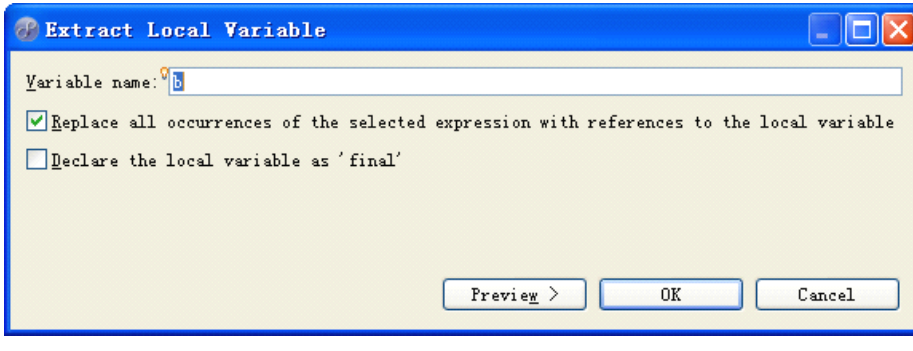


图 3-10 抽取局部变量

其中“Variable name”选项表示抽取后的局部变量名称。“Replace alll occurrences of the selected expression with references to the local variable”选项表示是否将所有出现该表达式的地方都替换成抽取后的局部变量引用，默认是选中的状态。“Declare the local

variable as ‘final’”选项是否使用 final 最终修饰符修饰抽取后的局部变量。

在界面中，可以单击“Preview”按钮查看改动预览，也可以单击“OK”按钮完成抽取局部变量操作，则操作后的“ExtractLocalVariableTest”程序代码为：

```
01  package refactor;
02  public class ExtractLocalVariableTest {
03      public static void main(String args[]){
04          int i=5;
05          boolean b = i==5;
06          boolean b1=(b)&true;
07          boolean b2=(b)|false;
08      }
09  }
```

从代码的变化可以看到，将“i==5”这个表达式抽取出来，赋值给 b 变量，将使用该表达式的地方都替换成了变量 b。

记忆：抽取局部变量操作的快捷键是“Alt+Shift+L”，使用该快捷键可以直接进入抽取局部变量的界面。

3.2.3 抽取常量

抽取常量是和抽取局部变量非常类似的，但是它的限制要求更多。因为常量的修饰符中有 static 静态修饰符和 final 最终修饰符，所以抽取出来的常量一定在方法外，也就是在类中，以成员的形式出现，这也就造成被抽取的表达式中使用的变量也要是成员变量。

先来定义一个类名为“ExtractConstantTest”的程序，其代码为：

```
01  package refactor;
02  public class ExtractConstantTest {
03      public static final int i=5;
04      public static void main(String args[]){
05          boolean b1=(i==5)&true;
06          boolean b2=(i==5)|false;
07      }
08  }
```

同样选中第 5 行的“i==5”表达式，然后在右键弹出菜单中，选择“Refactor”|“Extract Constant”命令，将弹出抽取常量的界面，如图 3-11 所示。

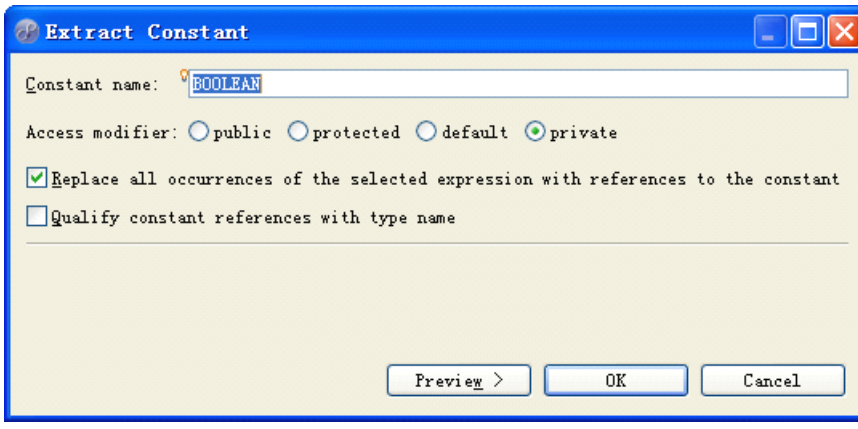


图 3-11　抽取常量

其中“Constant name”选项表示抽取后常量的名称，在 Java 的命名规范中通常使常量名大写。“Access modifier”选项表示抽取后常量的访问修饰符，可以在四种访问修饰符中任意选。“Replace alll occurrences of the selected expression with references to the constant”选项表示是否将所有出现该表达式的地方都替换成抽取后的常量引用，默认是选中的状态。“Qualify constant references with type name”选项表示是否使用类型名限定常量引用。

在抽取常量界面中，同样可以单击“Preview”按钮常量修改后预览。也可以单击“OK”按钮完成抽取常量操作，则操作后的“ExtractConstantTest”类程序代码变为：

```
01  package refactor;
02  public class ExtractConstantTest {
03      public static final int i=5;
04      private static final boolean BOOLEAN = i==5;
05      public static void main(String args[]){
06          boolean b1=(BOOLEAN)&true;
07          boolean b2=(BOOLEAN)|false;
08      }
09  }
```

从代码变化上可以看到将第 6 行和第 7 行中的“i==5”表达式替换成了常量，并把表达式赋值给该常量。

3.2.4　内联

内联是上述三种抽取操作的反向操作，它的操作对象是方法、局部变量和常量。通过内联操作，可以使用代码代替调用方法或者使用局部变量和常量的地方。这里我们通过方法的形式来讲解内联操作。

我们先来创建一个类名为“InlineTest”的程序，它的代码为：

```
01  package refactor;
02  public class InlineTest {
03      public void method1(){
04          System.out.println("这是第一个方法");
```

```
05              forMethod();
06          }
07          private void forMethod() {
08              for(int i=0;i<10;i++){
09                  System.out.println(i);
10              }
11          }
12          public void method2(){
13              System.out.println("这是第二个方法");
14              forMethod();
15          }
16      }
```

读者对该代码应该是不陌生的，这就是我们抽取方法后的代码。选中第 5 行的代码，在 MyEclipse 菜单中，选择“Refactor”|“Inline”命令，将弹出内联操作界面，如图 3-12 所示。

图 3-12　内联

其中“All invocations”选项表示是否内联所有调用的地方，这里选择该选项。然后会在该选项下出现一个“Delete method declaration”选项，它表示是否删除方法声明，也就是调用的方法，通常是选中的状态。“Only the selected invocation”选项表示仅仅内联选择的调用。

同样在该界面中，可以单击“Preview”按钮，查看代码预览，也可以单击“OK”按钮完成内联操作，则“InlineTest”类程序变为：

```
01  package refactor;
02  public class InlineTest {
03      public void method1(){
04          System.out.println("这是第一个方法");
05          for(int i=0;i<10;i++){
06              System.out.println(i);
07          }
08      }
09      public void method2(){
10          System.out.println("这是第二个方法");
```

```
11                  for(int i=0;i<10;i++){
12                      System.out.println(i);
13                  }
14          }
15  }
```

从代码的变化可以看到，将所有调用选中方法的地方都换为了方法中的代码，而且定义的方法被删除，这就是内联操作的作用。

记忆：内联操作的快捷键为“Alt+Shift+I”，使用该快捷键可以直接进入内联操作界面，但是要在前面先进行选择对什么对象进行操作。

3.2.5 将匿名内部类转换为嵌套内部类

学习本小节的操作前，读者一定要先了解 Java 程序中的内部类的知识。内部类从名称上就可以看出，它是在一个类的内部再定义另一个类。内部类在 Java 程序中扮演两种角色，在它所在的外部类外，它可以被看做是外部类的成员。在所在的外部类内，它又可以被看做是一个普通的类。

内部类根据在外部类中的成员角色不同又可以分为多种，本节中提到的匿名内部类和嵌套内部类就是其中两个。匿名内部类是指没有自己名称的内部类，通过使用父类或者实现的接口创建。嵌套内部类也成为静态内部类，在外部类中使用 static 静态修饰符修饰，类似于静态方法。在本小节中，就来看如何使用 MyEclipse 中的命令自动让匿名内部类转换成嵌套内部类。

先来开发一个具有匿名内部类的程序，其代码为：

```
01  package refactor;
02  public class CovertNestedTest {
03      public static void main(String[] args) {
04          new Runnable(){
05              public void run(){
06                  System.out.println("将匿名内部类转换为嵌套内部类");
07              }
08          };
09      }
10  }
```

该程序时不陌生的，在学习多线程语句包围时看到过该代码，其中将功能代码包围在线程方法中。选中其中的匿名内部类代码，也从第 4 行到第 8 行，但是要注意不要选择第 8 行的分号，不然不能进行操作。在 MyEclipse 的菜单中，选择“Refactor”|“Conver Anonymous Class to Nested”命令，将弹出进行内部类转换的界面，如图 3-13 所示。

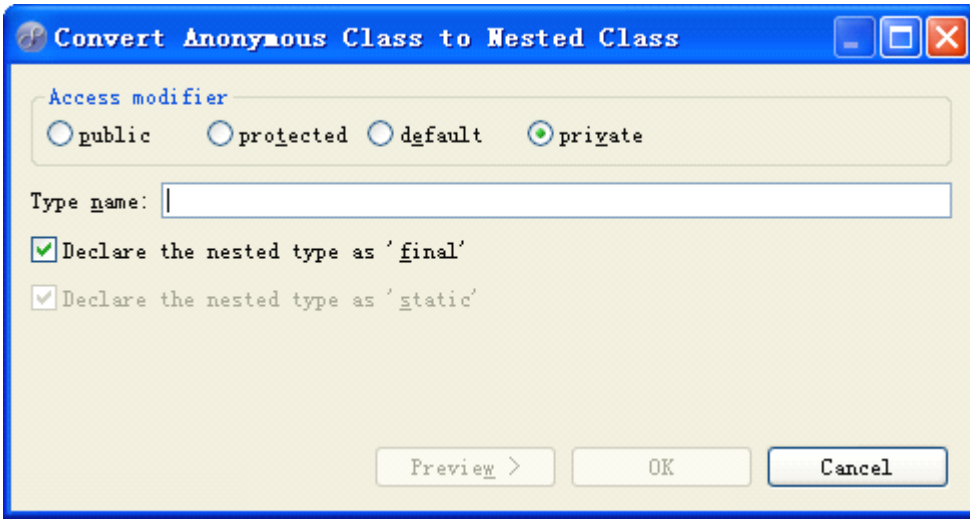


图 3-13　内部类转换界面

其中“Access modifier”选项表示生成嵌套内部类的访问修饰符，可以在四种级别的访问修饰符中选择。“Type name”选项表示生成嵌套内部类的类名，这里填写“MyThread”。“Declare the nested type as ‘final’”选项表示是否使用 final 修饰生成的嵌套内部类。单击“Preview”按钮可以看到转换后的预览，单击“OK”按钮将完成转换操作，则“CovertNestedTest”类程序代码变为：

```
01  package refactor;
02  public class CovertNestedTest {
03      private static final class MyThread implements Runnable {
04          public void run(){
05              System.out.println("将匿名内部类转换为嵌套内部类");
06          }
07      }
08      public static void main(String[] args) {
09          new MyThread();
10      }
11  }
```

在转换后的代码中，从第 3 行到第 7 行创建一个类名为“MyThread”的嵌套内部类，它实现“Runnable”接口，其中的功能代码是不会发生改变的。在第 9 行中创建了该嵌套内部类对象。

注意：在进行从匿名内部类向嵌套内部类转换时，一定要仅仅选择匿名内部类的代码，不要多选了任何内容，包括最后的分号，甚至是前后空格。

3.2.6　将成员类型内部类转换为顶级类

在上一小节中，我们学习了如何将匿名内部类转换成嵌套内部类，在本小节中将继续来学习如何将嵌套内部类转换为普通类。对于内部类中的成员内部类来说该操作也是适用的，我们这里仅使用静态内部类举例。

这里我们就不再开发举例程序，而是在变化后的“CovertNestedTest”类程序上进行操作。选中第 3 行中的“MyThread”类名，在 MyEclipse 的菜单中，选择“Refactor”|“Conver Member Type to Top Level”命令，将弹出转换为顶级类的界面，如图 3-14 所示。

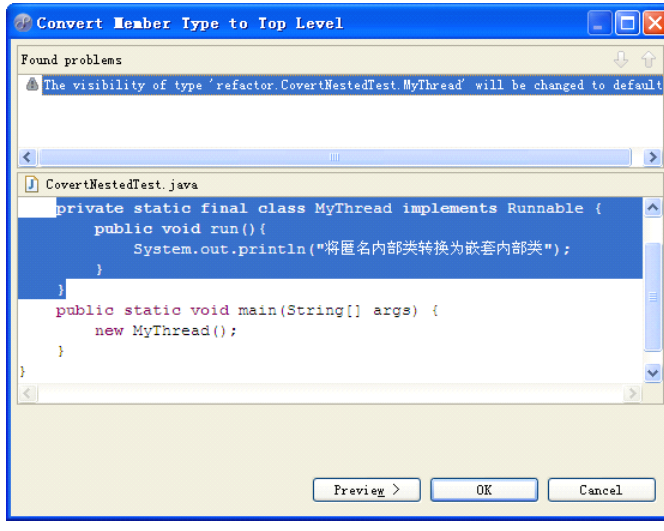


图 3-14　内部类向顶级类转换

在界面中将显示变化的代码，单击“Preview”按钮可以查看预览，单击“OK”按钮将完成转换操作。“CovertNestedTest”类程序变化后的代码为：

```
01  package refactor;
02  public class CovertNestedTest {
03      public static void main(String[] args) {
04          new MyThread();
05      }
06  }
```

可以看到其中的嵌套内部类已经没有了，而在“refactor”下多出了一个类程序，类名为“MyThread”，这就是原静态内部类类名，它的具体代码为：

```
01  package refactor;
02  final class MyThread implements Runnable {
03      public void run(){
04          System.out.println("将匿名内部类转换为嵌套内部类");
05      }
06  }
```

从这里可以看到生成后的新类和原嵌套内部类的功能相同的，只是修饰符不能再是原来的 private 和 static 修饰符。

注意：将成员内部类转换为顶级类和将嵌套内部类转换为顶级类是有一些不同的。在成员内部类可以调用外部类中的成员，所以为了不影响功能，将在转换后的顶级类中生成一个具有外部类类型参数的构造函数。

技巧：在 MyEclipse 中，并没有定义将匿名内部类转换为顶级类的命令，但是我

们可以将匿名内部类首先转换为嵌套内部类，再将潜逃内部类转换为顶级类。

3.2.7 将局部变量转换为成员变量

在实际开发中，定义完局部变量后，发现该变量将在多个语句块中使用，这时候就要将该局部变量提升为成员变量，这在数据库操作中经常出现。在本小节中，就来看一下在 MyEclipse 中如果自动将局部变量转换为成员变量。

首先来开发进行操作的“ConvertFieldTest”类程序，其代码为：

```
01  package refactor;
02  public class ConvertFieldTest {
03      public void getName(){
04          int age=21;
05          System.out.println("年龄为："+age);
06      }
07  }
```

选中第 4 行的“age”变量，在 MyEclipse 的菜单中，选择“Refactor”|“Convert Local Variable to Field”命令，将弹出转换成员变量界面，如图 3-15 所示。

图 3-15 局部变量转换成员变量

在其中“Field name”选项表示转换后的成员变量名称，通常和原局部变量名称相同。“Access modifier”选项表示转换后成员变量的访问修饰符，可以在四种级别的修饰符中选择。“Initialize in”选项表示成员变量的生成位置，其中“Field declaration”表示字段声明处，“Current method”表示当前方法前，“Class constructor”表示类构造函数处。

“Declare field as ‘static’”选项表示是否使用 static 修饰符修饰。“Declare field as ‘final”选项表示是否使用 final 修饰符修饰。单击“Preview”按钮可以看到转换后的预览。单击“OK”按钮将完成转换操作，则转换后的“ConvertFieldTest”类程序代码为：

```
01  package refactor;
02  public class ConvertFieldTest {
03      private int age;
04      public void getName(){
05          age = 21;
06          System.out.println("年龄为："+age);
07      }
08  }
```

从变化的代码可以看到，将第 5 行的局部变量变为第 3 行的成员变量，但是赋值操作仍然在方法中进行，从而不影响程序功能。

注意：如果局部变量在非静态方法中，则转换后就是非静态变量；如果局部变量在静态方法中，则转换后就是静态变量。如果是进行静态转换，则“Declare field as ‘static’”选项是默认选中的状态，而且不让修改。

3.3 类层次重构

类层次重构泛指父类与子类之间的重构，也可以是接口和实现类之间的重构。其中包括抽取超类、抽取接口等抽取操作，尽可能使用超类型的操作，以及对成员进行的下推和上拉操作。

3.3.1 抽取超类

抽取超类就是为当前操作类生成一个超类，生成的超类可以是非抽象类，也可以是抽象类。在抽取超类中，可以选择是否抽取子类中的方法，如果抽取，这些方法将移动到生成的超类中。

先来看进行操作的“ExtractSuperclassTest”类程序，其代码为：

```
01  package refactor;
02  public class ExtractSuperclassTest {
03      public void firstMethod(){
04          System.out.println("这是第一个方法");
05      }
06      public void secondMethod(){
07          System.out.println("这是第二个方法");
08      }
09  }
```

在包资源管理器中，选中“ExtractSuperclassTest”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Superclass”命令，将弹出抽取超类界面，如图 3-16 所示。

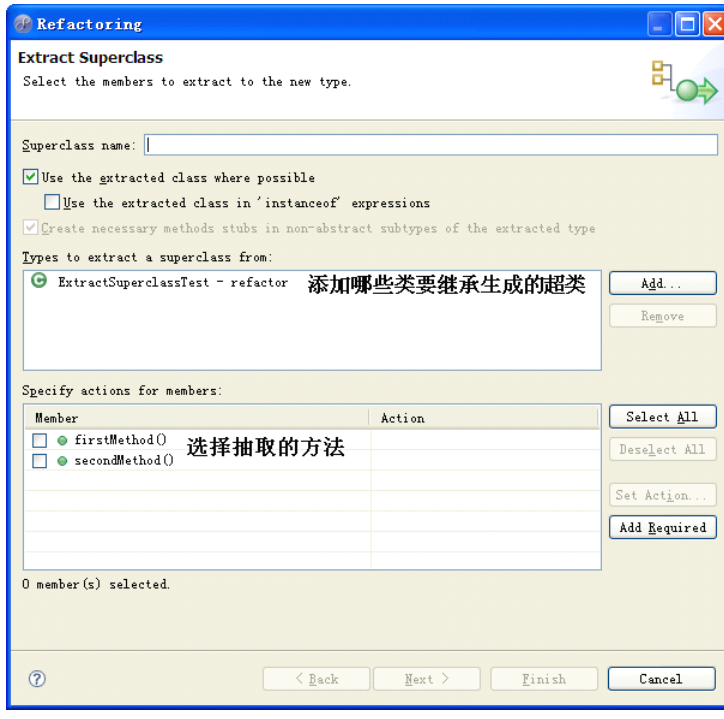


图 3-16　抽取超类

其中“Superclass name”选项表示抽取后超类的类名，这里填写“SuperClass”。“Use the extracted class where possible”选项表示是否尽可能的使用抽取的超类来引用子类对象。“Use the extracted calss in ‘instanceof’ expressions”选项表示是否在“instanceof”表达式中使用抽取的超类。“Create necessary methods stubs in non-abstract subtypes of the extracted type”表示当抽取抽象超类时，是否在子类中保留原方法实现抽象方法，只有当超类是抽象类时，该选项才可以选择。

“Type to extract a superclass from”栏表示哪些类将继承抽取后的超类，默认的是当前操作类，也可以单击“Add”按钮进行添加。“Specify actions for members”表示向父类中抽取哪些方法。在其中选中某一个成员，例如这里选中“firstMethod”方法，单击“Set Action”按钮，将弹出对成员操作界面，如图 3-17 所示。

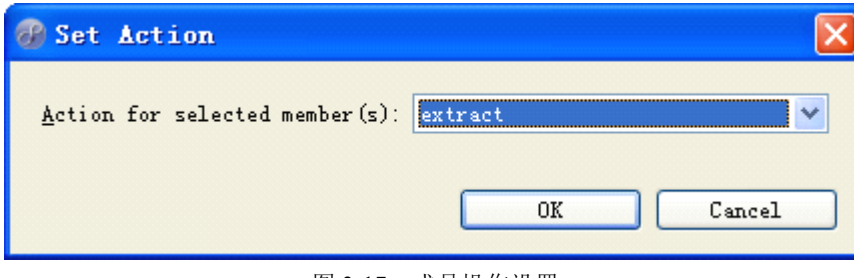


图 3-17　成员操作设置

其中默认的是“extract”选项，它表示抽取方法。除此之外，还包括“declare abstract in superclass”选项将抽取的方法声明为抽象方法，如果选择该操作，则抽取的超类也就

为抽象类，这里我们选择该选项。最后单击“OK”按钮完成抽取方法的操作。

回到抽取超类界面中，单击“Next”按钮，将进入抽取超类预览界面，如图 3-18 所示。

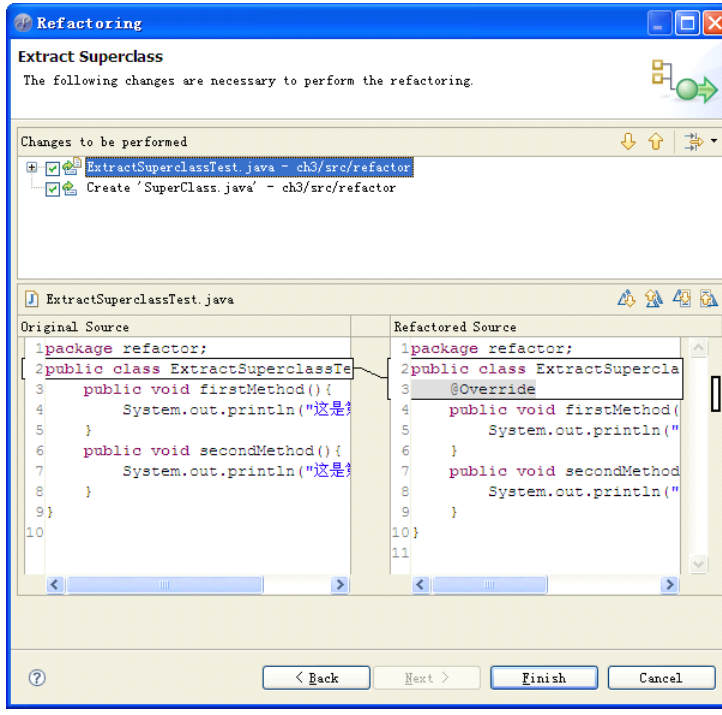


图 3-18　抽取超类预览

在预览界面中，单击“Finish”按钮，将完成抽取超类的操作。在当前“refactor”包下，将创建一个类名为“SuperClass”的程序，其代码为：

```
01  package refactor;
02  public abstract class SuperClass {
03      public SuperClass() {
04          super();
05      }
06      public abstract void firstMethod();
07  }
```

从代码中可以看到，生成一个类名为“SuperClass”的抽象父类，并为它自动生成了无参构造函数。在第 6 行中完成“firstMethod”的抽取，并把它抽取成抽象方法。

再来看抽取后的“ExtractSuperclassTest”类程序，变化后的代码为：

```
01  package refactor;
02  public class ExtractSuperclassTest extends SuperClass {
03      @Override
04      public void firstMethod(){
05          System.out.println("这是第一个方法");
06      }
07      public void secondMethod(){
```

```
08                System.out.println("这是第二个方法");
09            }
10    }
```

其中“firstMethod”方法是实现的超类中的抽象方法。如果在抽取超类界面中，将“secondMethod”方法也选中，并设置操作是抽取，则该方法将出现在“SuperClass”类程序中，在“ExtractSuperclassTest”类中也就不会再出现。

注意：如果一个类中有抽象方法，则这个类一定是抽象类。但是在抽象类中，不一定有抽象方法。在 MyEclipse 中，正是遵守该语法，当抽取方法是设置操作为抽象抽取，则超类将自动变为抽象类。

3.3.2 抽取接口

抽取接口是和抽取超类非常类似的，使用该操作可以从当前操作类抽取出该类所实现的接口，在其中也可以将方法抽取成抽象方法。

同样还是开发一个操作类，类名为“ExtractInterfaceTest”，它的程序代码为：

```
01    package refactor;
02    public class ExtractInterfaceTest {
03            public void firstMethod(){
04                    System.out.println("这是第一个方法");
05            }
06            public void secondMethod(){
07                    System.out.println("这是第二个方法");
08            }
09    }
```

在包资源管理器中，选中“ExtractInterfaceTest”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，如图 3-19 所示。

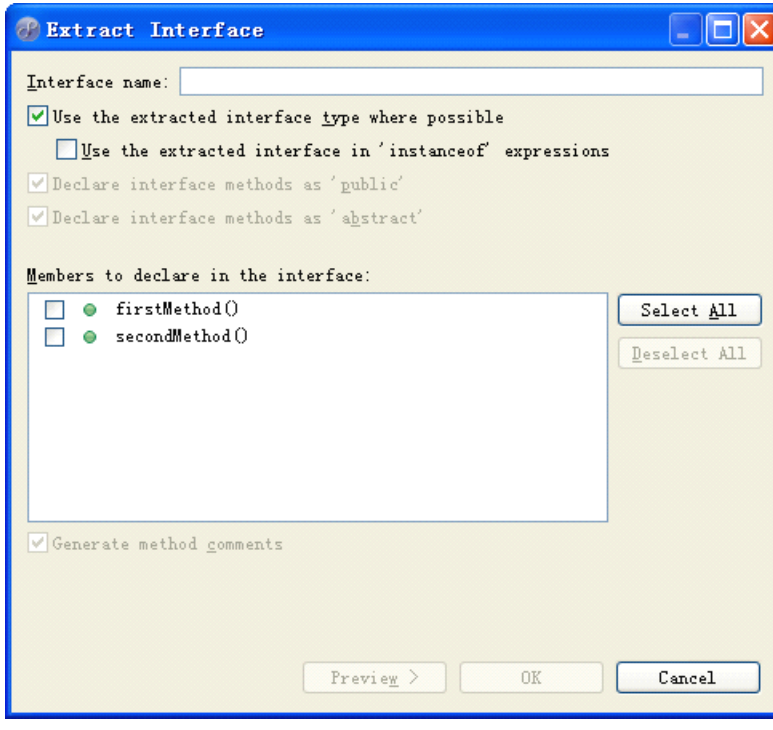


图 3-19　抽取接口

其中“Interface name”选项表示抽取后的接口名称，这里填写“MyInterface”。“Use the extracted interface type where possible”选项表示是否尽可能使用抽取的接口类型引用子类对象。“Use the extracted interface in ‘instanceof’ expressions”选项表示是否在“instanceof”表达式中使用抽取的接口。“Declare interface methods as ‘public’”选项表示是否抽取方法后使用 public 修饰符修饰，“Declare interface methods as ‘abstract’”选项表示是否抽取方法后使用 abstract’修饰符修饰，这两个选项仅仅是代码层面，因为在接口中，如果你没写这些修饰符，系统也会自动加上。

“Members to declare in the interface”选项栏中，用于选择哪些方法将抽取到接口中，这里我们将这两个方法全部选中。单击“Preview”按钮可以看到抽取接口后的预览，单击“OK”按钮，将完成抽取接口的操作。将在“refactor”包下创建一个接口为“MyInterface”的接口程序，其代码为：

```
01  package refactor;
02  public interface MyInterface {
03      public abstract void firstMethod();
04      public abstract void secondMethod();
05  }
```

从接口代码中可以看到将操作类中的两个方法抽取成抽象方法，放置在该接口中。再来看“ExtractInterfaceTest”类程序变化后的代码：

```
01  package refactor;
02  public class ExtractInterfaceTest implements MyInterface {
03      /* (non-Javadoc)
```

```
04          * @see refactor.MyInterface#firstMethod()
05          */
06         public void firstMethod(){
07             System.out.println("这是第一个方法");
08         }
09         /* (non-Javadoc)
10          * @see refactor.MyInterface#secondMethod()
11          */
12         public void secondMethod(){
13             System.out.println("这是第二个方法");
14         }
15     }
```

首先在第 2 行中让原“ExtractInterfaceTest”类程序实现抽取后的“MyInterface”接口。在其中为每一个方法生成了注释，表示该方法是实现了接口中的方法。

说明：抽取接口在 Java Web 开发中是经常要到的。目前 Web 开发中都采用分层思想，在数据访问层和业务逻辑层中，都是采用接口和实现类的组合。在开发完实现类后，就可以使用抽取接口操作，将相应的接口抽取出来。

3.3.3 尽可能使用超类型

超类型是指当前操作类的超类类型或者接口类型。在 Java 语法中，可以使用父类类型引用或者接口类型应用指向子类对象，这样使用的优点就是能够实现类层次上的多态。在本小节中，就来学习一下如何使用 MyEclipse 的命令，来进行尽可能使用超类型的操作。

我们在上一小节抽取接口后的“ExtractInterfaceTest”类程序基础上，添加 main 入口方法，其代码为：

```
01  package refactor;
02  public class ExtractInterfaceTest implements MyInterface {
03      /* (non-Javadoc)
04       * @see refactor.MyInterface#firstMethod()
05       */
06      public void firstMethod(){
07          System.out.println("这是第一个方法");
08      }
09      /* (non-Javadoc)
10       * @see refactor.MyInterface#secondMethod()
11       */
12      public void secondMethod(){
13          System.out.println("这是第二个方法");
14      }
15      public static void main(String[] args) {
16          ExtractInterfaceTest ei=new ExtractInterfaceTest();
17          ei.firstMethod();
```

```
18              ei.secondMethod();
19          }
20      }
```

在包资源管理器中，选中“ExtractInterfaceTest”类节点，单击鼠标右键，在弹出的菜单中选择“Refactor”|“Use Supertype Where Possible”命令，将弹出选择超类型的界面，如图 3-20 所示。

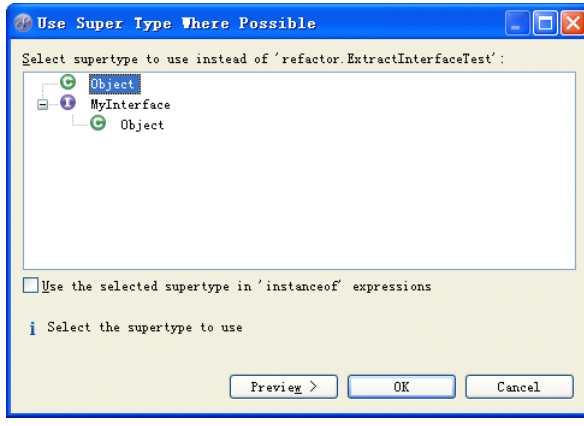


图 3-20　选择使用的超类型

因为所有的类都是 Object 的子类，所以在其中肯定有 Object 这一选项。如果操作的类继承了超类，或者实现了接口，则在该界面中将会把超类和接口的类型显示出来。例如“ExtractInterfaceTest”类实现了“MyInterface”接口，则在其中显示“MyInterface”，这里我们就选择“MyInterface”选项。单击“Preview”按钮，将看到代码的预览。单击“OK”按钮将完成操作，则“ExtractInterfaceTest”类程序变化后的代码为：

```
01  package refactor;
02  public class ExtractInterfaceTest implements MyInterface {
03          /* (non-Javadoc)
04           * @see refactor.MyInterface#firstMethod()
05           */
06          public void firstMethod(){
07                  System.out.println("这是第一个方法");
08          }
09          /* (non-Javadoc)
10           * @see refactor.MyInterface#secondMethod()
11           */
12          public void secondMethod(){
13                  System.out.println("这是第二个方法");
14          }
15          public static void main(String[] args) {
16                  MyInterface ei=new ExtractInterfaceTest();
17                  ei.firstMethod();
18                  ei.secondMethod();
19          }
```

```
20    }
```

上述代码中第 16 行的“ExtractInterfaceTest”类引用已经变为了“MyInterface”接口引用，从而就变为使用接口引用指向实现类对象。如果代码中有多种同样情况都会使用同种操作更改。

说明：在 JavaWeb 的分层开发中，本小节的操作也经常被用到。例如在 Spring 框架的开发中，使用最多的就是使用接口引用指向子类对象，但是这里的对象是通过系统创建后注入的，该知识点要到后面的讲解中才会学到。

3.3.4 下推

下推和上拉是重构中非常重要的两个概念，它们的操作对象就是父类和子类，在本小节中先来学习一下下推。使用下推操作，可以将父类中的成员（包括方法和变量）移动到它的直接子类中。如果操作的是方法，在父类中可以不保留或者保留该方法的抽象方法。

同样还是先来开发进行操作的程序，这里我们通过 Person 类和 Student 类表示父类和子类。先来看 Person 类的代码：

```
01    package ref;
02    public class Person {
03        int time=8;
04        public void sleep(){
05            System.out.println("睡觉,时间为"+time+"小时");
06        }
07        public void eat(){
08            System.out.println("吃饭");
09        }
10    }
```

然后再来看继承 Person 类的 Student 类，其代码为：

```
01    package ref;
02    public class Student extends Person {
03        public void study(){
04            System.out.println("学习");
05        }
06    }
```

因为下推是对父类中的成员进行操作，所以在包资源管理器中，选中“Person”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Push Down”命令，将弹出下推操作界面，如图 3-21 所示。

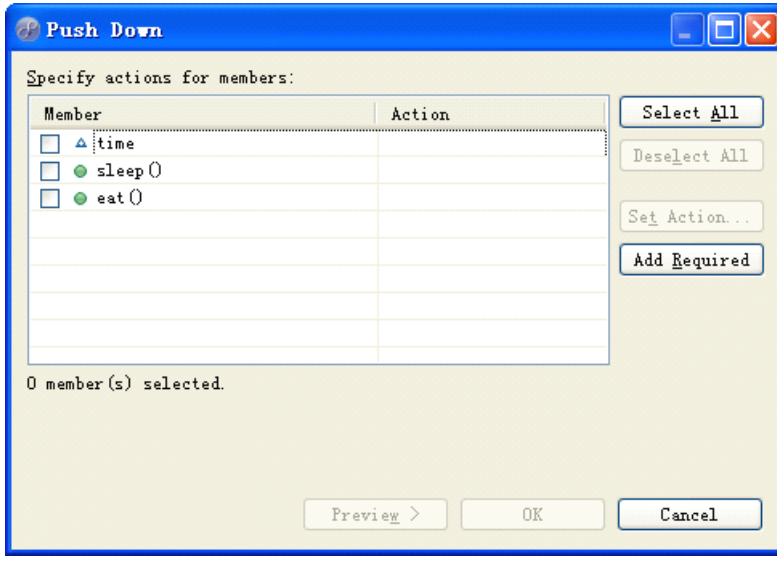


图 3-21　下推

其中“Specify actions for members”选项表示对父类中的哪些成员进行下推操作。在右边有四个按钮，其中“Select All”按钮表示全部选择，“Deselect All”按钮表示全部不选择。“Set Action”按钮表示对成员的操作设置，例如选择并选中“sleep”方法，然后单击该按钮，将弹出选择操作界面，如图 3-22 所示。

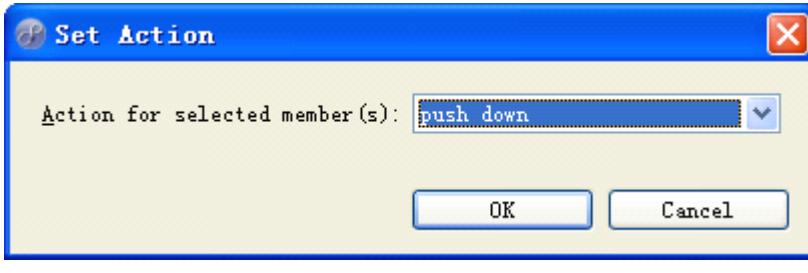


图 3-22　成员下推操作

其中有两个选项，“push down”选项表示进行下推操作，将选中的方法移动到子类中，它也是默认选项。除此之外，“leave abstract declaration”选项表示在父类中保留抽象方法，如果选择该选项，则父类自动变为抽象类。

在下推操作界面中，“Add Required”按钮表示将已选择成员所必须的成员也选择上。例如在“sleeep”方法中调用了“time”变量，选择“sleeep”方法后，单击该按钮，将自动将“time”变量选择上。按照上述选择，单击“Preview”按钮，将查看操作预览。单击“OK”按钮将完成下推操作，则“Person”类程序将变为：

```
01  package ref;
02  public class Person {
03      public void eat(){
04          System.out.println("吃饭");
05      }
06  }
```

可以看到其中的 sleep 方法和 time 变量已经消失，这里就是将它们下推到 Student 类中。继续来看 Student 类的代码：

```
01  package ref;
02  public class Student extends Person {
03      int time = 8;
04      public void study(){
05          System.out.println("学习");
06      }
07      public void sleep() {
08          System.out.println("睡觉,时间为"+time+"小时");
09      }
10  }
```

可以看到原来属于 Person 类的 sleep 方法和 time 变量移动到它的直接子类，也就是 Student 类中。

3.3.5 上拉

上拉和下推正好相反，它是将子类中的成员移动到它的父类中。在操作中也可以选择是整体移动，还是在父类中保留抽象方法。这里我们开发一个继承 Person 类的 Teacher 类，其代码为：

```
01  package ref;
02  public class Teacher extends Person {
03      public void teach(){
04          System.out.println("讲课");
05      }
06      public void refresher(){
07          System.out.println("复习");
08      }
09  }
```

在包资源管理器中，选中“Teacher”类节点，单击鼠标右键，在弹出的右键菜单中，选择“Refactor”|“Pull Up”命令，将弹出上拉操作界面，如图 3-23 所示。

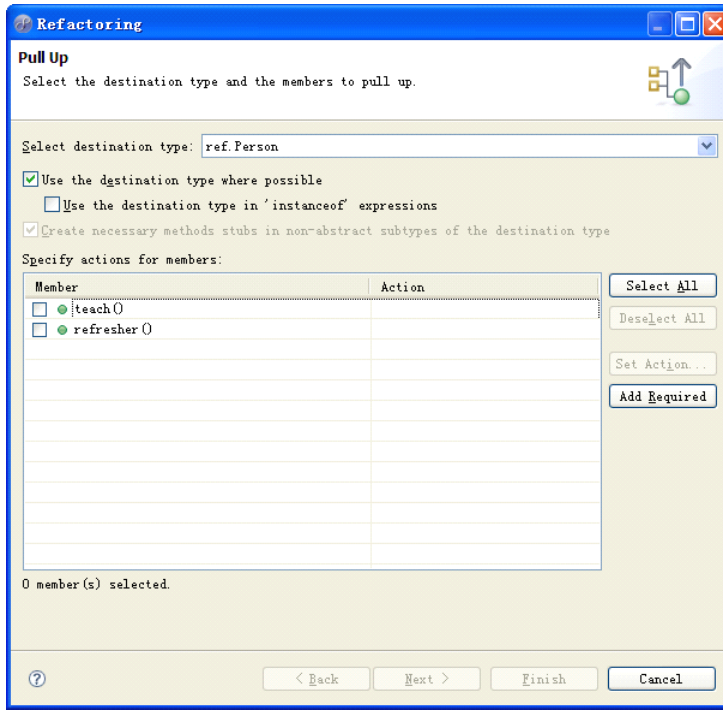


图 3-23　上拉

其中“Select destination type”选项表示选择上拉的目标类型，也就是将成员上拉到哪一个父类中。

注意：上拉目标类型可以选择，并不表示 Java 类能够继承多个父类，这里指的是级联父类。例如人类又继承了生物类，则这里还会出现生物类的选项。

下面的三个选项在前面已经多次讲解道，这里就不再重复讲解。“Specify actions for members”选项表示对操作类中的哪些成员进行上拉操作，这里和下推操作是非常相似的。选择并选中 teach 方法，单击“Set Action”按钮，将弹出选择上拉操作界面，如图 3-24 所示。

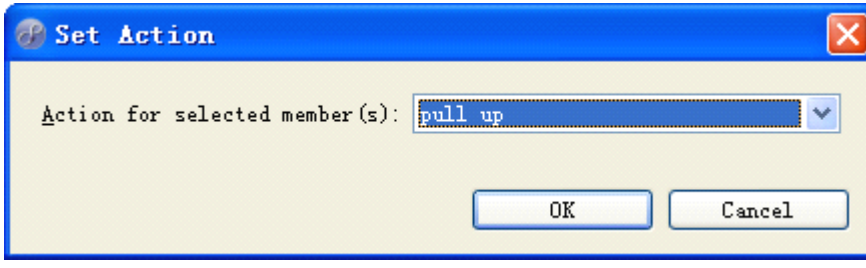


图 3-24　成员上拉操作

其中有两个选项，“pull up”选项表示上拉移动选中的成员，它也是默认的选项。除此之外，“declare abstract in destination”选项表示将上拉的方法转换为抽象方法，放置在父类中，如果选择该选项，则父类中所有子类内都会实现该上拉的方法。读者可以自己进行该选项的操作，这里我们采用默认的选项。

回到原界面，单击“Next”按钮将查看到代码的变化和预览。单击“Finish”按钮将完成上拉操作，则 Teacher 类的代码将变为：

```
01  package ref;
02  public class Teacher extends Person{
03      public void refresher(){
04          System.out.println("复习");
05      }
06  }
```

可以看到 Teacher 类中的“teach”方法已经消失，它被上拉到 Person 类中。再来看一下 Person 类的代码：

```
01  package ref;
02  public class Person {
03      public void eat(){
04          System.out.println("吃饭");
05      }
06      public void teach() {
07          System.out.println("讲课");
08      }
09  }
```

可以看到在 Teacher 类中消失的“teach”方法已经移动到 Person 类中，从而完成上拉操作。

3.4 引入重构

引入重构是指在原事物的基础上，引入新事物，从而进行封装操作。其中包括引入间接方法、引入工厂、引入参数对象、引入参数、引入方法封装字段等操作。在实际开发中灵活使用这些引入操作，能够大大加快开发速度。

3.4.1 引入间接方法

在 Java 程序中，每一个方法完成一个功能，项目的实现就是通过调用这些方法完成的。在项目开发中期，有可能会发现一些方法的功能并不完善，但是通常不改变已经测试通过的方法，而是对这些方法进行重构，产生间接方法。在间接方法中调用原方法，然后进行功能补充，让调用原方法的地方调用这个间接方法。在本小节中就来学习一下在 MyEclipse 中，如何使用命令完成该引入间接方法的重构操作。

这里我们通过电梯来模拟该操作，电梯最基本的功能就是上下运动，除此之外还应该有开门和关门的功能。来看模拟电梯的代码：

```
01  package ref;
02  public class Elevator {
03      public void work(){
04          System.out.println("电梯上下运动");
05      }
```

```
06          public void open(){
07                  System.out.println("开门");
08          }
09          public void close(){
10                  System.out.println("关门");
11          }
12          public static void main(String[] args) {
13                  Elevator elevator = new Elevator();
14                  elevator.work();
15          }
16  }
```

上述代码的 main 方法中，仅调用了 word 方法，是不能完成电梯的功能的。由于调用的地方可能有很多，如果将开门和关门的方法再调用，是非常麻烦的。这时候就可以使用引入间接方法的重构。

在包资源管理器中，选中“work”方法节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Introduce Indirection”命令，将弹出进行引入间接方法操作的界面，如图 3-25 所示。

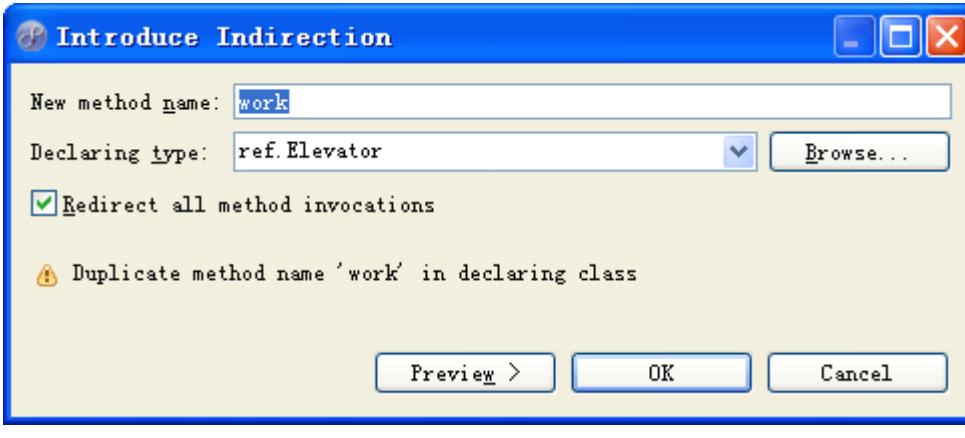


图 3-25　引入间接方法

其中“New method name”选项表示引入间接方法的方法名，这里填写“workAll”。“Delaring type”选项表示将引入的间接方法放置在哪一个类中，这里我们就选择和原方法在同一个类中。“Redirect all method invocations”选项表示是否重定向所有方法的调用，也就是是否所有的调用都重新调用间接方法，默认是选中状态。

单击“Preview”按钮将看到操作的预览效果。单击“OK”按钮就完成引入间接方法的操作，则“Elevator”类代码变为：

```
01  package ref;
02  public class Elevator {
03          public void work(){
04                  System.out.println("电梯上下运动");
05          }
06          public void open(){
```

```
07              System.out.println("开门");
08          }
09          public void close(){
10              System.out.println("关门");
11          }
12          public static void main(String[] args) {
13              Elevator elevator = new Elevator();
14              Elevator.workAll(elevator);
15          }
16          public static void workAll(Elevator elevator) {
17              elevator.work();
18          }
19      }
```

上述代码的第 16 行生成了一个方法名为“workAll”的方法，它就是自动生成的间接方法，在其中第 17 行中调用“work”方法，从而完成原方法功能，这里只从而给出当前类类型的参数，是因为它方法可以定义到其他类中。

最后就是程序员手动完成重构后的功能，在“workAll”方法中，添加其他功能，具体代码如下：

```
01  package ref;
02  public class Elevator {
03      public void work(){
04          System.out.println("电梯上下运动");
05      }
06      public void open(){
07          System.out.println("开门");
08      }
09      public void close(){
10          System.out.println("关门");
11      }
12      public static void main(String[] args) {
13          Elevator elevator = new Elevator();
14          Elevator.workAll(elevator);
15      }
16      public static void workAll(Elevator elevator) {
17          elevator.open();
18          elevator.work();
19          elevator.close();
20      }
21  }
```

在第 16 行的间接方法中，在执行原方法前调用 open 方法，在原方法后调用 close

方法，从而完善原方法的功能。

说明：有些读者可能会感觉到北小节学习的内容和前面学习的代理方法有所类似的。如果仅仅是完善某一个方法的功能，建议使用引入间接方法的操作。如果是对某一类功能的完善，则就应该使用代理。

3.4.2 引入工厂

工厂模式也是设计模式中非常重要的一种，其中工厂类是必不可少的。在 Java Web 项目开发中，在使用 Spring 框架之前，通常使用工厂类来创建对象。使用工厂类创建实现类对象的优势是可以很好的进行分层，例如在业务逻辑层中只需要知道数据访问接口就行，可不用知道具体是如何实现的。

在本小节中，就来学习一下如何在工厂类中自动生成创建实现类对象的方法。这里我们首先要创建一个数据访问层实现类程序，其代码为：

```
01  package ref;
02  public class UserDAOImpl {
03      public UserDAOImpl(){
04      }
05  }
```

在进行本小节的操作前，还要创建一个工厂类，其代码为：

```
01  package ref;
02  public class UserDAOFactory {
03  }
```

注意：在实现类中只所以要定义无参构造函数，是因为在工厂类中创建对象时要知道调用哪一个构造函数，本小节操作也是基于该无参构造函数的。在 MyEclipse 中，也不会自动创建工厂类，所以在执行操作前一定要首先把工厂类创建出来。

在包资源管理器中，选中“UserDAOImpl”的无参构造函数节点，单击鼠标右键，在弹出的右键菜单中，选择“Refactor”|“Introduce Factory”命令，将弹出引入工厂操作界面，如图 3-26 所示。

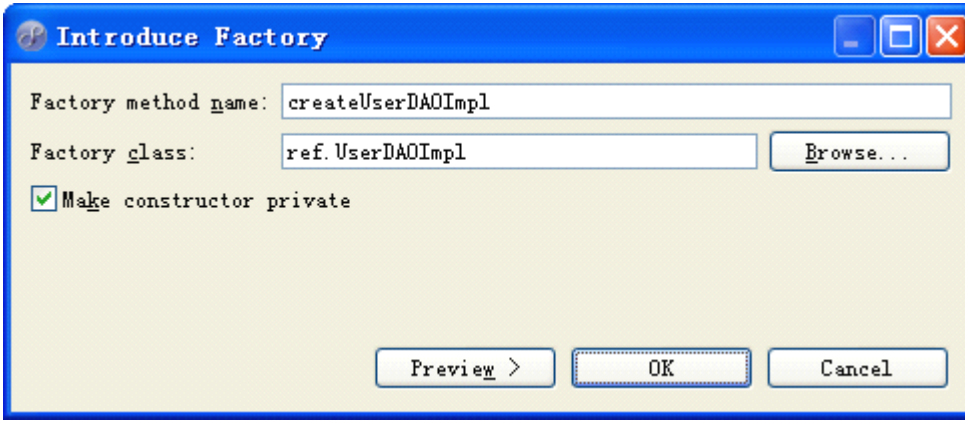


图 3-26　引入工厂

其中“Factory method name”选项表示在工厂类中定义的创建实现类对象的方法名，这里就是用默认的“createUserDAOImpl”。“Factory class”选项表示工厂类的类名，这里选择填写“UserDAOFactory”。“Make constructor private”选项表示是否改变构造函数的访问级别，这里采用默认选中状态。

单击“Preview”按钮将看到引入工厂操作的预览。单击“OK”按钮将完成引入工厂操作，则“UserDAOImpl”类程序代码变为：

```
01  package ref;
02  public class UserDAOImpl {
03      UserDAOImpl(){
04      }
05  }
```

可以看到构造函数的访问级别已经发生改变。再来看工厂类的代码：

```
01  package ref;
02  public class UserDAOFactory {
03      public static UserDAOImpl createUserDAOImpl() {
04          return new UserDAOImpl();
05      }
06  }
```

可以看到在其中第 3 行创建了方法名为“createUserDAOImpl”的方法，在其中返回“UserDAOImpl”类对象。在实际开发中，经常定义方法的返回类型为接口类型，所以可以对“UserDAOImpl”类执行抽取接口操作，具体操作见抽取接口一节的讲解。

3.4.3 引入参数对象

在实际开发中，经常会遇到这样的现象，就是一组参数总是同时传递。例如在网站中，用户注册和登录的操作会对应着注册和登录方法，在这两个方法中都需要传递用户名和密码这两个参数。在这种情况下就可以定义一个用户类，向注册和登录方法中传递用户类类型的参数，在该参数中包含了用户名和用户密码。

这里我们通过一个士兵的程序来更形象的说明该重构操作，同样也看一下在 MyEclipse 中，是如何通过命令执行引入参数对象操作的，士兵类程序代码为：

```
01  package ref;
02  public class Soldier {
03      public void sleep(String grade,String name){
04          System.out.println(grade+" "+name+"请示睡觉");
05      }
06      public void eat(String grade,String name){
07          System.out.println(grade+" "+name+"请示吃饭");
08      }
09  }
```

士兵在执行某件事时都要进行请示，例如“一等兵张三请示睡觉”、“二等兵李四请示吃饭”等。在 sleep 和 eat 方法中都需要士兵等级和士兵姓名两个参数，这时候就可以使用引入参数对象的重构操作。

在 MyEclipse 编辑区中，选中“sleep”方法名，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Introduce Parameter Object”命令，将弹出引入参数对象界面，如图 3-27 所示。

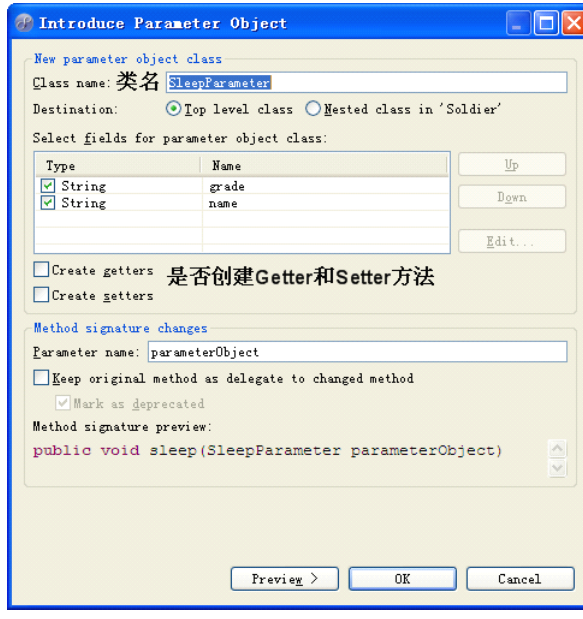


图 3-27　引入参数对象

其中“Class name”选项表示创建参数对象的类型，这里填写“Parameter”。“Destination”表示创建的类的位置，“Top level class”选项表示创建顶级类，“Nested class in ‘Soldier’”选项表示在士兵类中创建嵌套内部类，这里选择默认的顶级类。

“Select fields for parameter object class”表示将哪些参数设置到引入的参数类中，这里将两个参数都选中。“Create getters”和“Create setters”选项表示是否为参数设置 Getter 和 Setter 方法，它们的选择关系到参数的调用方式，这里将这两个选项选中。“Parameter name”表示参数名称，这里保留默认的“parameterObject”。

在界面的最下方会看到简单的预览，也可以单击“Preview”按钮查看详细预览。单击“OK”按钮将完成引入参数对象的操作，则士兵类代码将变为：

```
01  package ref;
02  public class Soldier {
03      public void sleep(Parameter parameterObject){
04          System.out.println(parameterObject.getGrade()+" "
05                                  +parameterObject.getName()+"请示睡觉");
06      }
07      public void eat(String grade,String name){
08          System.out.println(grade+" "+name+"请示吃饭");
09      }
10  }
```

从代码中可以看到，将 sleep 方法中的参数换为“Parameter”类型的参数，因为在操作中设置了 Getter 和 Setter 方法，所以获取参数值时通过它们设置。

除了士兵类的改变为，在和士兵类同包下，还会创建“Parameter”类程序，其代码为：

```
01  package ref;
02  public class Parameter {
03      private String grade;
04      private String name;
05      public Parameter(String grade, String name) {
06          this.grade = grade;
07          this.name = name;
08      }
09      public String getGrade() {
10          return grade;
11      }
12      public void setGrade(String grade) {
13          this.grade = grade;
14      }
15      public String getName() {
16          return name;
17      }
18      public void setName(String name) {
19          this.name = name;
20      }
21  }
```

该程序都是自动生成的，其中生成了 grade 和 name 两个属性，它们就是操作方法中的原参数。第 5 行中生成了具有上面两个参数的构造函数，当创建对象时将设置两个属性值。从第 9 行到第 20 行生成了两个参数的 Getter 和 Setter 方法，这是在引入参数对象界面中选择的。

3.4.4 引入参数

引入参数和引入参数对象是两种完全不一样的操作，它们只是名称相似。在开发阶段的方法中，一些变量通常给出值或者表达式来进行功能的测试，在测试没有问题后，通常使用方法参数来代替值或者表达式。通过使用引入参数的重构操作就可以自动完成这一点。

我们在原来士兵类的基础上定义一个新方法，它的代码变为：

```
01  package ref;
02  public class Soldier {
03      //省略其他方法
04      public void sports(){
05          String myGrade="一等兵";
06          String myGame="张三";
07          System.out.println(myGrade+myGame+"请示活动");
08      }
09  }
```

在第 4 行的 sports 方法中，将通过使用参数给出的士兵等级和士兵姓名以具体局部变量的形式给出。选中“""一等兵""”内容，在编辑区的右键弹出菜单中，选择“Refactor”|“Introduce Parameter”命令，将弹出引入参数的界面，如图 3-28 所示。

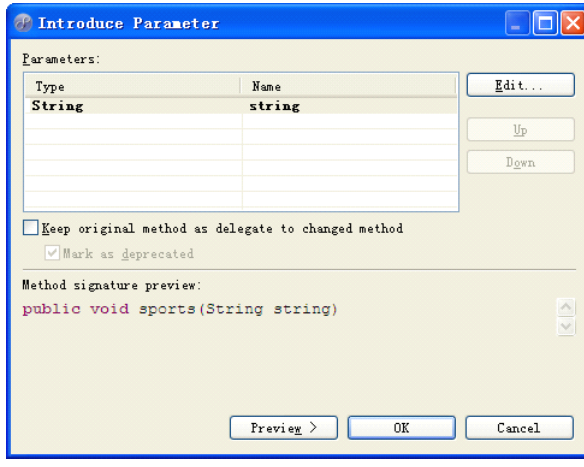


图 3-28　引入参数

其中“Type”选项表示引入参数的类型，它和参数的内容有关，通常是正确的。“Name”选项表示引入参数的名称，这里将它改为“grade”。“Keep original method as delegate to changed method”选项表示是否保留原方法作为更改方法的代理，默认是不选中的状态。

在界面的下面可以看到方法的简单预览，单击“Preview”按钮可以看到完整预览。单击“OK”按钮将完成引入参数的操作，则士兵类的代码将变为：

```
01  package ref;
02  public class Soldier {
03      //省略其他方法
04      public void sports(String grade){
05          String myGrade=grade;
06          String myGame="张三";
07          System.out.println(myGrade+myGame+"请示活动");
08      }
09  }
```

从代码中可以看到将原局部变量值以参数代替，在方法中，给出了一个“grade”字符串参数。继续对“""张三""”进行参数，可以继续为方法添加一个“name”字符串参数，这样就和前面两个方法相似了。

3.4.5　引入方法封装字段

封装字段的重构操作和前面学习过的生成 Getter、Setter 方法非常类似的，它们都完成为字段生成 Getter 和 Setter 方法的功能。在本小节中就来学习一下在 MyEclipse 中，如何进行封装字段的重构操作。

先来开发一个进行操作的 User 用户类，其代码为：

```
01  package ref;
02  public class User {
03      int id;
04      String username;
05  }
```

在包资源管理中，选中“id”变量节点，单击鼠标右键，在弹出的菜单中，选择选择“Refactor”|“Encapsulate Field”命令，将弹出封装字段操作的界面，如图 3-29 所示。

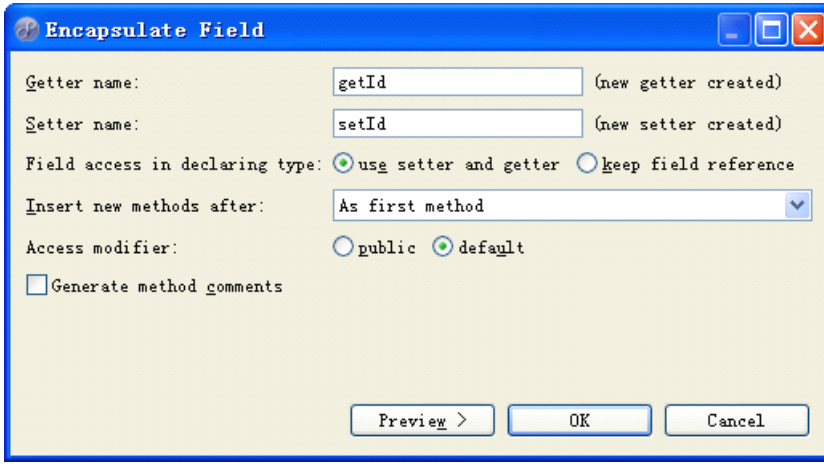


图 3-29　封装字段

其中“Getter name”和“Setter name”选项用于填写生成的 Getter 和 Setter 方法的名称，默认给出最标准的方法名。“Field access in declaring type”表示声明类型中的字段访问，“use setter and getter”选项表示使用 Setter 和 Getter 方法，“keep field reference”选项表示保留字段引用。“Insert new methods agter”选项表示生成的方法放在哪个方法之后，如果类中有多个方法，都会做为选项出现，默认是“As first method”，表示做为第一个方法。“Access modifier”选项表示生成方法的访问修饰符，通常选择 public。

单击“Preview”按钮将看到封装字段后的预览。单击“OK”按钮将完成封装字段的操作，操作后的 User 用户类的代码将变为：

```
01  package ref;
02  public class User {
03      private int id;
04      String username;
05      public void setId(int id) {
06          this.id = id;
07      }
08      public int getId() {
09          return id;
10      }
11  }
```

因为进行操作的是“id”变量，从而从第 5 行到第 10 行生成了它的 Setter 和 Getter 方法，并且“id”变量自动变为 private 私有访问级别。

说明：如果仅仅是想为字段生成 Getter 和 Setter 方法，就应该使用 MyEclipse 中的“Generate Getters and Setters”命令。因为该命令要比重构中的封装字段更简单。

第4章　管理数据库

数据库在项目开发中是必不可少的，在 MyEclipse 中集成了管理数据库的功能，通过 MyEclipse 可以对数据库进行操作。使用 MyEclipse 对数据库进行管理，还是后面学习 Hibernate 等数据访问层框架的基础。

在 MyEclipse 中，通过 MyEclipse Database Explorer 进行管理的，它的中文名称被叫做 MyEclipse 数据库浏览器或者 MyEclipse 数据库管理器，并为它定义了单独的视图，在本章中就主要在“MyEclipse Database Explorer”视图下讲解如何管理数据库。

4.1　认识 MyEclipse 中自带的数据库

在 MyEclipse 的默认安装中，会自带一款开源数据库，叫做 Derby。Derby 数据库是一款纯 Java 语言开发的开源免费数据库，在 Java JDK 6.0 版本中也包含该数据库，它已经是 Java 的一部分，所以也把它称为 Java DB。

4.1.1　切换到数据库视图

视图在第 1 章中，已经讲解过了。不同的视图对应着不同的程序开发，例如进行 Java 程序开发时，就可以选择 Java 视图。默认的视图 MyEclipse Java 开发视图，在该视图中进行 Java 程序开发已经非常方便的，所以在前面的开发中并没有切换视图。但是在管理数据库中，如果不切换到数据库视图，操作其中是非常不方便的。

在 MyEclipse 的菜单中，选择“Window”|“Open Perspective”|“MyEclipse Database Explorer”命令。如果在“Open Perspective”下并没有“MyEclipse Database Explorer”命令，可以选择“Other”命令，在弹出的界面中，选择“MyEclipse Database Explorer”命令。这时 MyEclipse 中将发生大的改变，其中界面左半部分是数据库浏览器，右上部分是编辑区，左下部分是控制台区，如图 4-1 所示。

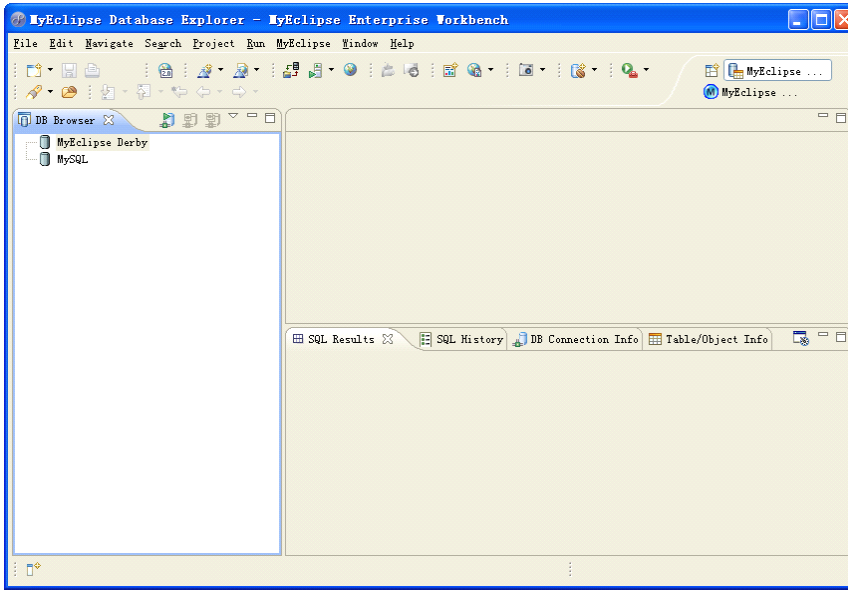


图 4-1　数据库视图界面

在数据库浏览器中，会将目前 MyEclipse 中能够管理的数据库显示出来，如图 4-2 所示。

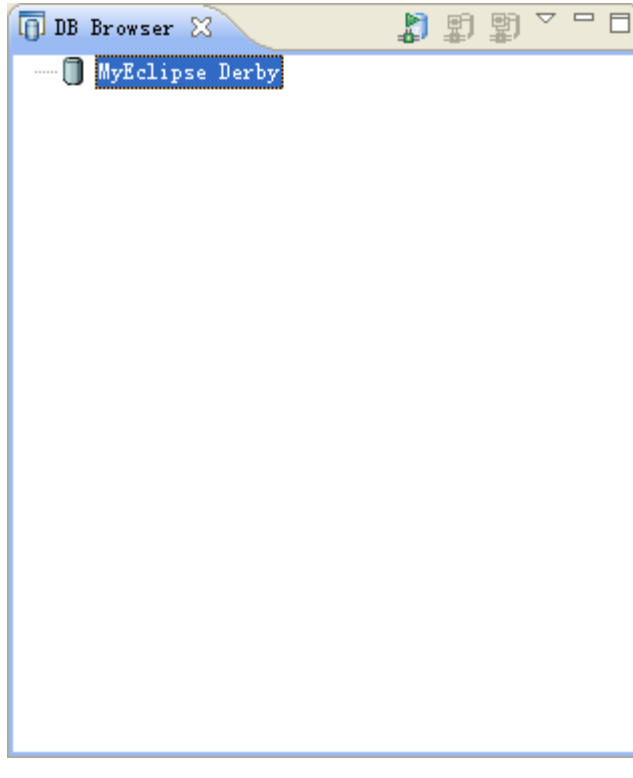


图 4-2　数据库浏览器

因为在 MyEclipse 自带了 Derby 数据库，所以在这里默认会将这个数据库显示出来。

当加入新的外部数据库后，同样也会显示在这里。

4.1.2 启动服务器服务

在管理数据库操作中，启动服务器服务并不是必须的，因为在打开数据库连接时，会自动执行该操作。但是在 Java 程序开发中，如果要进行数据库操作，就需要启动服务器的服务。

启动服务器服务并不需要在数据库视图下操作的，在原视图下就可以完成。在控制台区，有一个“Servers”选项卡，选择该选项卡就可以看到目前有哪些服务可以启动，如图 4-3 所示。

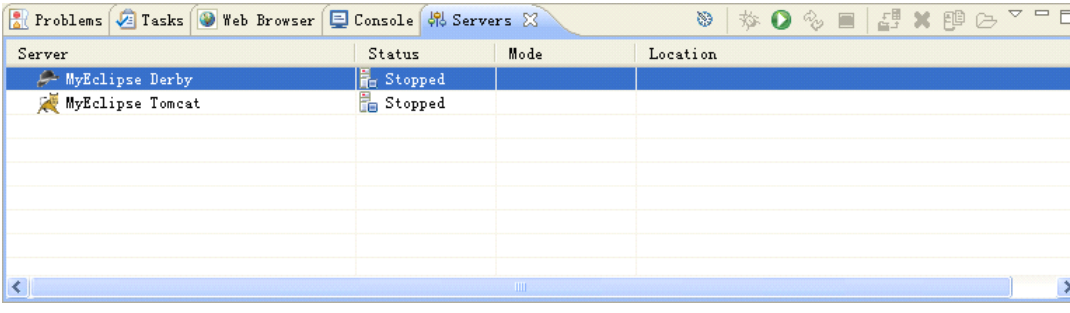


图 4-3 启动服务

其中“MyEclipse Derby”就是要启动的 Derby 数据库服务器服务，选中该服务，单击鼠标右键，在弹出的菜单中，选择“Run Server”命令将启动服务，并进入到控制台中。控制台显示内容为：

Apache Derby Network Server － 10.2.2.0 - (485682) 已启动并且已准备好 2009-11-04 06:59:47.359 GMT 时在端口 1527 上接受连接

显示该内容后，表示 Derby 数据库服务已经启动，这时候就可以在 Java 程序中进行 Derby 数据库的操作。

4.1.3 打开数据库连接

回到数据库视图中，我们继续来学习如何打开数据库连接。在数据库浏览器中，选择要打开的数据库。当前情况下只有 Derby 数据库，所以我们选择该数据库。单击鼠标右键，在弹出菜单中，选择“Open connection”命令将打开当前选择的数据库。

说明：打开数据库连接，也可以使用数据库浏览器最上面的按钮进行操作，其中第一个按钮就是打开数据库连接按钮。

打开数据库连接后，将把当前数据库中已经创建的具体数据库，以及每一个数据库下已经创建的数据表列出来。单击“Connected to MyEclipse Derby”节点前面的加号，将把所有数据库信息显示出来。单击某一数据库节点前的加号，将把该数据库中的所有表、视图等显示出来。单击“TABLE”节点，将把当前数据库下的所有数据表显示出来。如图 4-4 所示。

图 4-4　打开数据库连接

从图中可以看到能够将数据库的信息细分到具体字段上，并人为地为它们起了相应的名称，在后面的学习中会经常提到这些名称。

打开数据库连接后，我们来看一下如何关闭数据库连接。在数据库浏览器的最上面，打开按钮的后面，有两个关闭按钮。第一个“  ”按钮表示关闭单个连接，选择某一连接对象后，可以使用该按钮。第二个“  ”按钮表示关闭所有数据库连接。

4.1.4　查看数据库详细信息

在上一小节中打开数据库连接后，可以看到该数据库中具有的库、表等信息。如果仅能看到这些信息，还不能体现 MyEclipse 的强大之处。在控制台区，具体两种查看数据库详细信息的具体视图，分别是“DB Connection Info”和“Table/Object Info”。选择“DB Connection Info”选项卡，将显示连接数据库的详细信息，如图 4-5 所示。

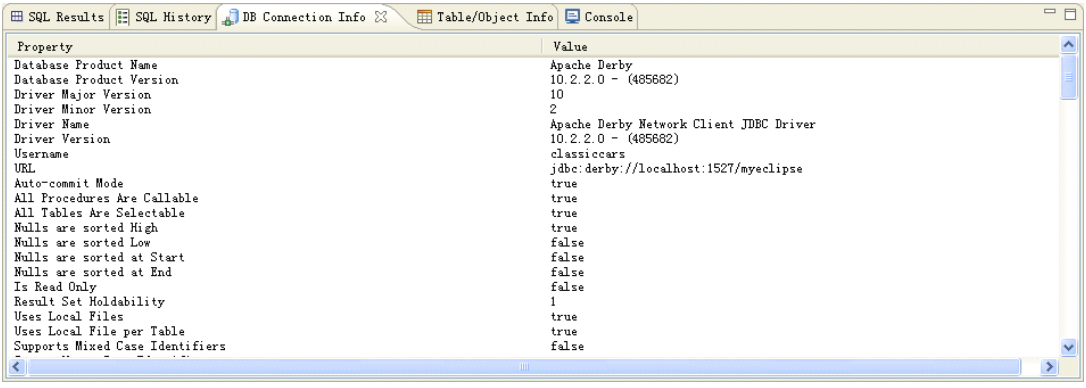


图 4-5　连接数据库信息

因为其中的内容是非常多的，在图中并不能完全体现，读者可以在自己的 MyEclipse 中查看到。这里也不能对这些信息完全讲解到，我们选择几条重要的信息进行说明。其中“URL”和“Username”就是在 JDBC 编程中经常用到的信息，连接 MyEclipse 中默认 Derby 数据库的 URL 为“jdbc:derby://localhost:1527/myeclipse”，username 为“classiccars”，连接密码这里并没有给出，密码也是“classiccars”。

选择“Table/Object Info”选项卡，可以查看到具体表格信息，如图 4-6 所示。

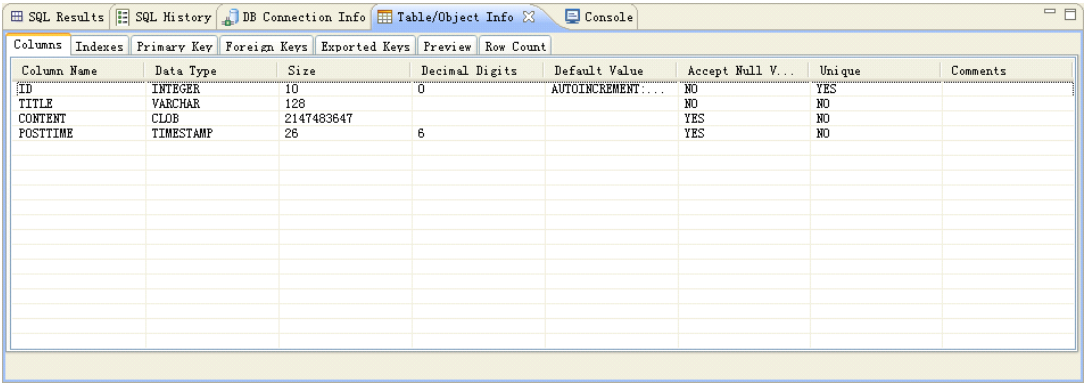


图 4-6　具体表格信息

在其中又具有显示更详细信息的选项卡。“Columns”显示表的列信息，包括列名、类型、大小等。“Indexes”显示表的索引信息，“Primary Key”显示表的主键信息，“Foreign Keys”显示表的外键信息，“Exported Keys”显示表的导出主键信息。“Priview”显示表的具体数据，在实际开发中经常用来显示表格中已有哪些数据。“Row Count”表示当前表中具有多少条数据。

4.2　对 MySQL 数据库的管理

在上一节中，主要讲解了 MyEclipse 中自带的 Derby 数据库，在目前实际开发中，Derby 数据库使用并不多的，所以从本节开始主要讲解如何使用 MyEclipse 管理外部数据库。这里我们通过 MySQL 数据库为例进行讲解，在学习本节之前，一定要首先安装

MySQL 数据库。

4.2.1 建立 MySQL 连接

我们首先来学习一下如何建立 MyEclipse 外部的 MySQL 数据库连接。在 MyEclipse 的数据库浏览器中，单击鼠标右键，在弹出的菜单中，选择“New”命令，将弹出建立数据库连接的界面，如图 4-7 所示。

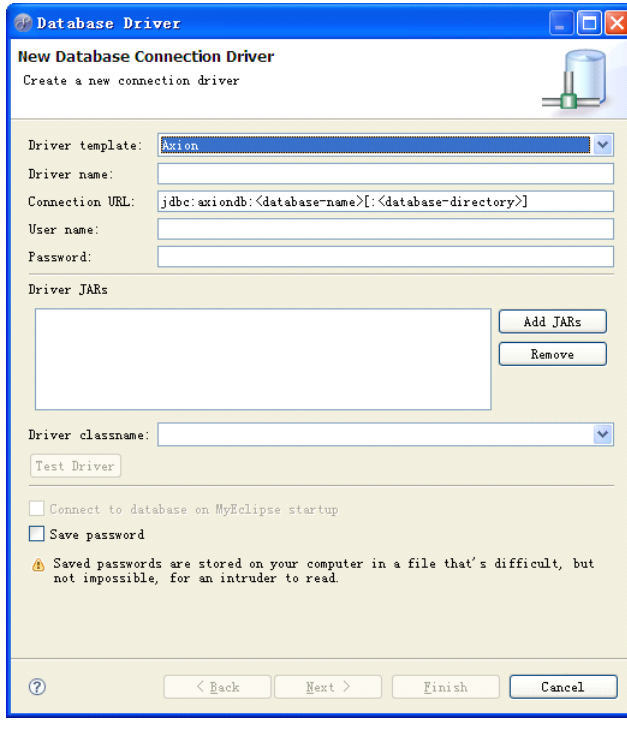


图 4-7 建立外部数据库连接

其中“Driver template”选项表示建立连接的数据库类型，从该选项中也可以看出在 MyEclipse 中支持非常多的数据库，这里选择“MySQL Connector/J”选项，它就表示 MySQL 数据库。“Driver name”选项表示建立连接后的名称，也就是显示在数据库浏览器中的名称，这里填写“MySQL“。

“Connection URL”选项表示建立连接的 URL，这里填写和 MySQL 建立连接的 URL，也就是“jdbc:mysql://localhost:3306/myeclipse”。“User name”和“Password”选项分别是进行连接用到的用户名和密码。

注意：这里没有给出具体的用户名和密码的内容，是因为每个人安装 MySQL 时设置的密码有所不同的，要根据本地的设置来填写。

“Driver JARS”栏用于导入 MySQL 的驱动包，单击“Add JARS”按钮，在弹出的界面中找到本地驱动包。导入驱动包后，在“Driver calssname”选项中将自动填写用于连接的驱动串。

单击“Test Driver”按钮，可以对当前连接进行测试，单击后将弹出输入密码界面，

如图 4-8 所示。

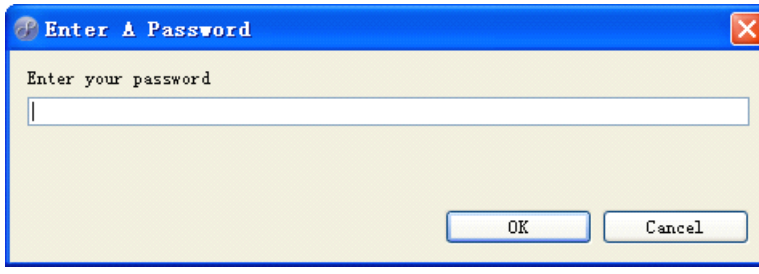


图 4-8　输入连接密码

输入安装 MySQL 数据库时设置的连接数据库的密码后，单击“OK”按钮，如图弹出如图 4-9 所示的界面，则表示连接数据库的配置正确。

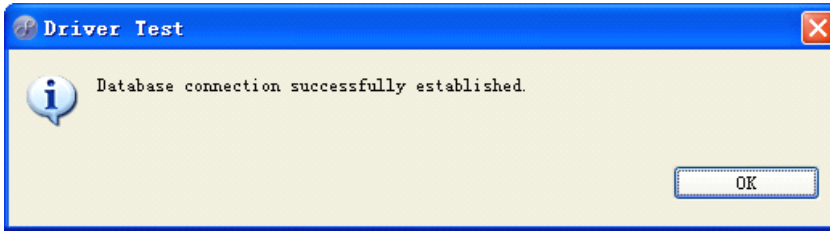


图 4-9　测试连接成功

如果在建立连接界面中，选中“Save Password”选项，再次对连接进行测试，将不用再输入密码。单击“Finish”按钮将完成建立 MySQL 连接操作。在数据库浏览器中，将多出现一个名称为“MySQL”的数据库连接，对于它也可以进行上一节学习的打开和查看数据库操作。

说明：测试在开发中是非常重要的组成部分，读者在开发中也要养成测试的习惯。通过测试能够保证自己所做的每一步都是正确的，所以方便后面的使用。

4.2.2　编辑和复制数据库连接

建立外部数据库连接后，还可以对该连接进行相应操作，例如编辑、复制和删除等。在数据库浏览器中，选中上一小节建立的“MySQL”连接，单击鼠标右键，在弹出的菜单中，选择“Edit”命令，将弹出对数据库连接进行编辑操作的界面，如图 4-10 所示。

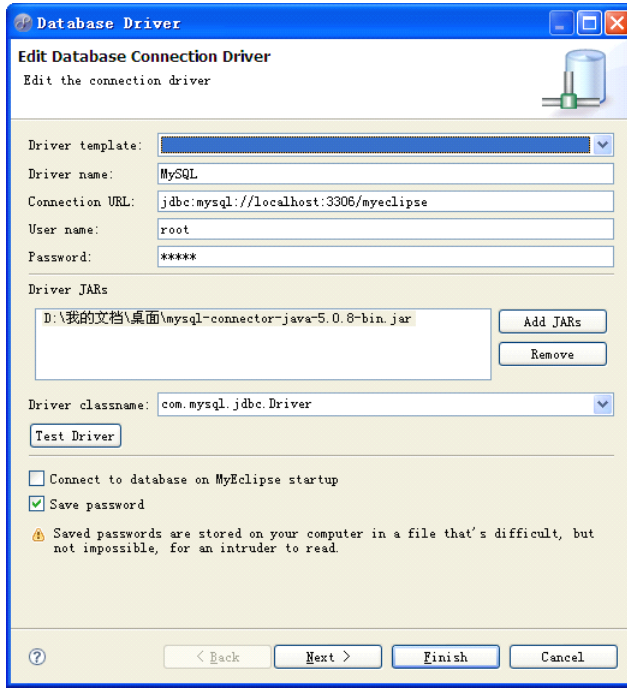


图 4-10　编辑数据库连接

在其中会首先将“MySQL”连接的信息显示出来，在该界面中可以对这些进行修改，修改后单击“Finish”按钮将完成对数据库连接的编辑。

在弹出的右键菜单中，还可以选择“Copy”命令，将弹出和编辑界面类似的界面，它的操作结果就是另外创建一个数据库连接。例如不再连接 MySQL 数据库中的 myeclipse 数据库，而是连接 java 数据库，这时候就可以改动其中的 URL，变为“jdbc:mysql://localhost:3306/java”，然后改变连接名称，这样就复制出来另外一个连接。

除了“Edit”命令和“Copy”命令外，还有“Delete”命令，通过它可以将建立的链接删除。

4.2.3　导出和导入数据库连接

在 MyEclipse 中，可以对 Java 项目进行导出和导入操作，同样对数据库连接也可以进行导出导入操作。在数据库浏览器中，选中要导入的数据库连接，单击鼠标右键，在弹出的菜单中，选择“Export”命令，将弹出导出操作的界面，如图 4-11 所示。

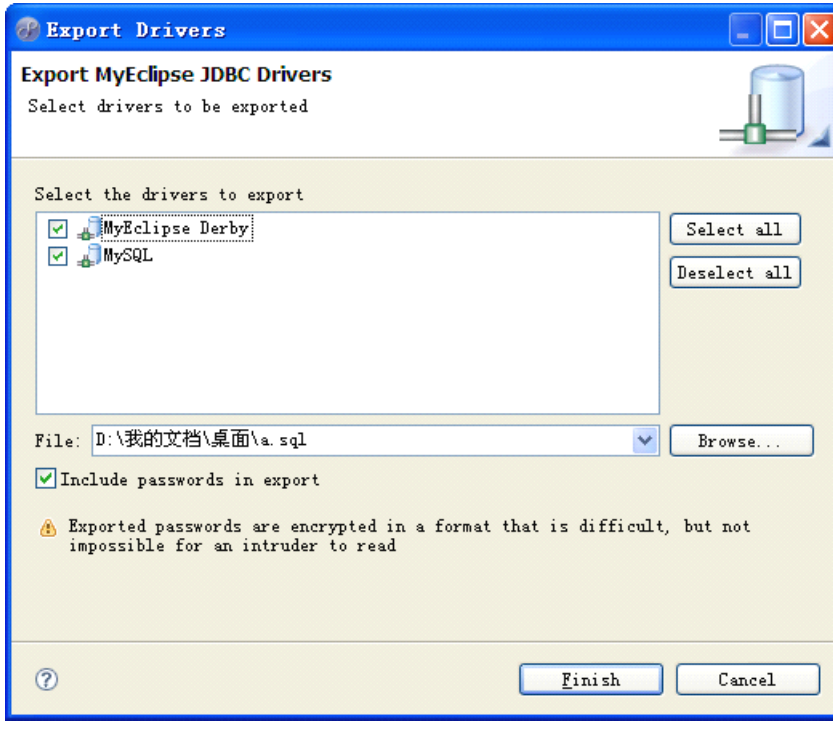


图 4-11　导出数据库连接

在其中“Select the drivers to export”选项中选择要导入的数据库，这里我们仅选择“MySQL”。“File”选项表示导出的目标文件，也就是将信息导出到哪一个文件中，这里可以自己填写，也可以单击“Browse”按钮选择导出的位置。单击“Finish”按钮将完成导出数据库连接的操作。

导出数据库连接后，我们再来看一下如何导入数据库连接。首先我们将数据库浏览器中的“MySQL”连接删除，然后在其中单击鼠标右键，在弹出的菜单中选择“Import”命令，将弹出导入连接的界面，如图 4-12 所示。

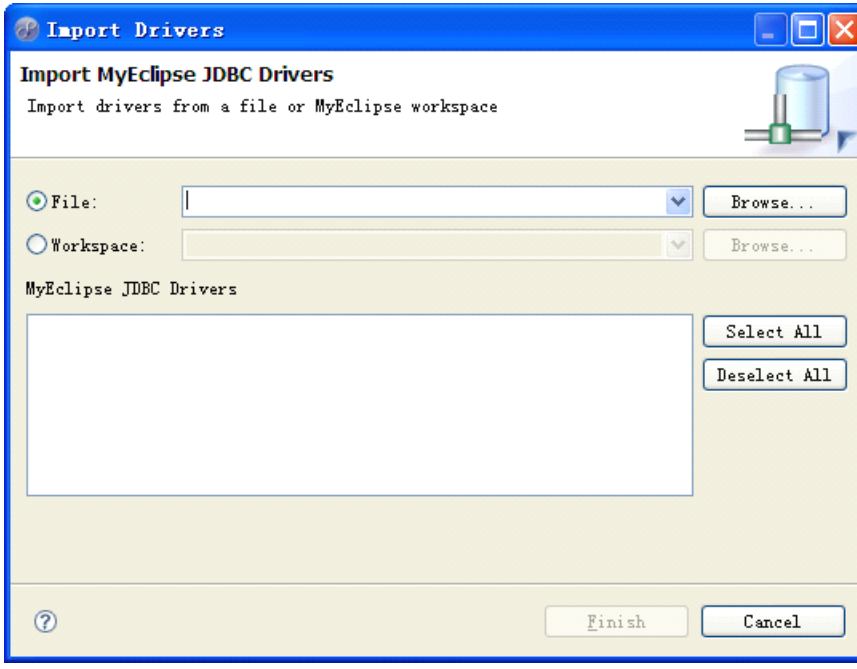


图 4-12　导入数据库连接

其中“File”选项表示要导入的文件，这里选择“a.sql”文件，它是上面导出操作生成的文件。选择该文件后，在“MyEclipse JDBC Drivers”栏中就会将能够导入的数据库连接显示出来。单击“Finish”按钮就会将该数据库连接导入，从而将“MySQL”连接继续显示在数据库浏览器中，如图 4-13 所示。

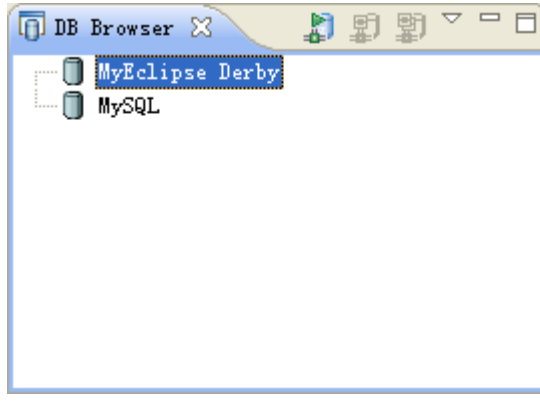


图 4-13　导入后

4.3　进行数据库操作

在前面的讲解中，我们主要对数据库连接进行操作。在本节中将主要来学习如何在 MyEclipse 中进行数据库的相关操作，包括创建和删除数据库、创建和删除数据表、插入数据等，以及 MyEclipse 中特有的集成功能。

4.3.1 创建数据库

创建数据库是数据库操作中最基本的操作之一，在 MyEclipse 中也可以通过 SQL 语句进行创建，我们这里主要来看如何界面化创建数据库。

在数据库浏览器中，选中“Connected to MySQL”节点，单击鼠标右键，在弹出的菜单中选择“Create Database”命令，将弹出创建数据库的界面，如图 4-14 所示。

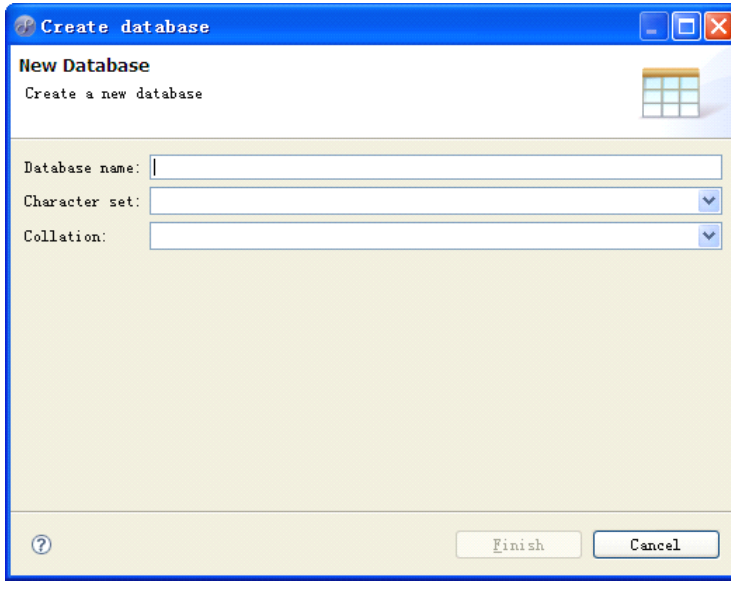


图 4-14　创建数据库

其中“Database name”选项用来输入新建数据库的名称，这里填写“mydb”。“Character set”选项表示创建数据库的字符集，这里选择表示简体中文的“gb2312”。“Collation”选项表示创建数据库的字符编码，这里选择和字符集对应的“gb2312_chinese_ci”。单击“Finish”按钮，则完成创建数据库的操作，从而在 MySQL 中创建了一个名称为“mydb”的数据库。

注意：创建数据库时设置字符集和编码是非常重要的，尤其是对于中文信息储存时。如果选择的字符集和编码不支持中文，则保存信息后显示的将是乱码。

4.3.2 创建数据表

创建数据库后，我们就可以继续在数据库下创建数据表。创建数据表也是可以通过 SQL 语句完成的，但是这里仍然先来学习如何通过界面进行创建。

选中“mydb”数据库节点，单击鼠标右键，在弹出的菜单中，选择“new Table”命令，将弹出创建数据表的界面，如图 4-15 所示。

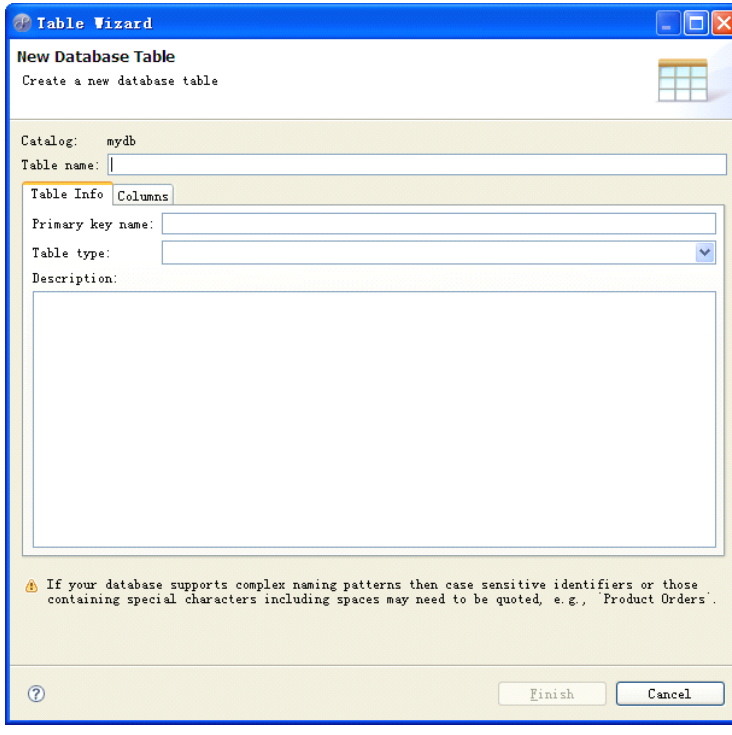


图 4-15　创建数据表

其中“Catalog”选项表示新创建的数据表所属哪个数据库，这是会自动填写好的。“Table name”选项表示新创建数据表的名称，这里填写“user”，表示创建一个用户表。下面有两个选项卡，在“Table Info”选项卡中，填写数据表的相关信息，“Primary key name”表示数据表的主键名称，“Table type”表示数据表的类型，“Description”表示数据表的描述，这些都是可以不填写的。

在“Columns”选项卡中完成向数据表中添加字段的功能，单击“Add”按钮将弹出创建数据表字段界面，如图 4-16 所示。

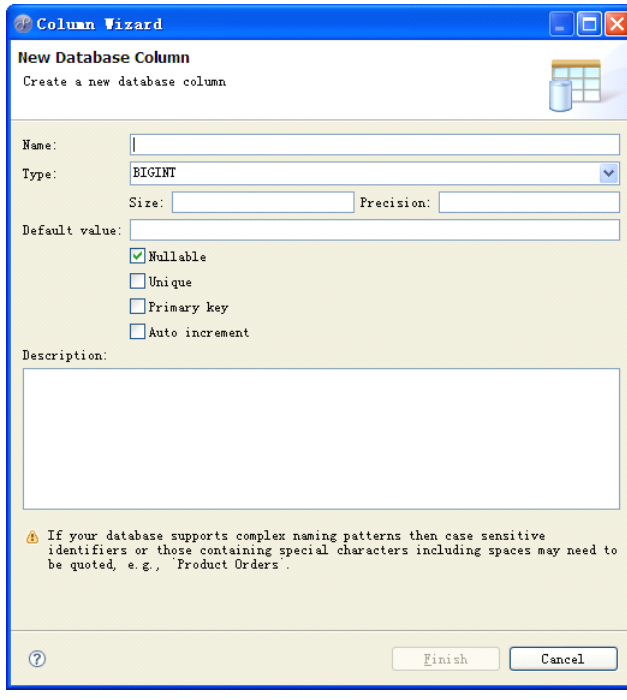


图 4-16　创建字段

其中“Name”选项表示字段的名称，这里填写“id”。“Type”选项表示字段的类型，在其中选择“INT”。“Size”表示字段大小，对于 id 而言，通常填写 11。“Precision”表示大小精度，对于 INT 类型来说，是不需要设置它的。“Default value”选项表示字段的默认值，这里可以不填写。

“Nullable”选项表示字段内容是否可以为空，选中表示可以为空，它也是默认选择。“Unique”选项表示字段内容是否是唯一的，也就是是否可以重复。“Primary key”选项表示字段是否为主键。“Auto increment”选项表示是否自动递增。在“Description”选项中可以给出对该字段的描述。单击“Finish”按钮，将完成 id 字段的创建。

使用同样的创建字段的方法，再创建一个“name”字段。回到创建数据表的界面中后，单击“Finish”按钮，从而完成数据表的创建。这样就在“mydb”数据库下创建了一个 user 用户表，其中具有 id 和 name 两个字段。

注意：不同数据库之间，字段的类型是有所不同的。在创建数据表时，要根据数据库的不同来选择相对应的类型。

4.3.3　删除数据表

删除数据表是和创建数据表相对应的，它是非常容易操作的。选中要删除的数据表节点，单击鼠标右键，在弹出的菜单中，选择“Drop Table”命令，将弹出选择是否删除数据表的对话框，如图 4-17 所示。

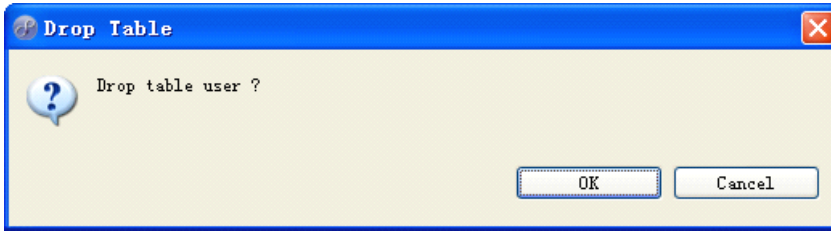


图 4-17　删除数据表

单击“OK”按钮，将完成删除 user 数据表的操作。

4.3.4　进行 SQL 语句编辑

在 MyEclipse 中也可以通过 SQL 语句来进行数据库操作，SQL 语句是数据库操作中最直接的操作方式。选中“Connected to MySQL”节点，单击鼠标右键，在弹出的菜单中，选择“New SQL Editor”命令，将在编辑区中出现用于输入 SQL 语句的编辑界面，如图 4-18 所示。

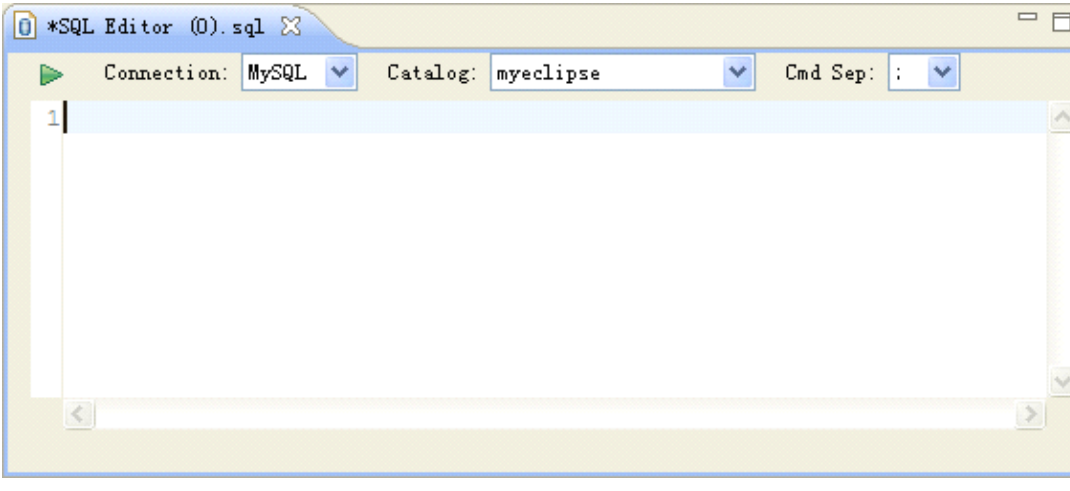


图 4-18　SQL 语句编辑器

其中最左端的绿三角是执行 SQL 语句按钮，“Connection”选项表示对哪一个连接进行操作，这里选择“MySQL”。“Catalog”选项表示对哪一个数据库进行操作，这里选择上一小节创建的“mydb”数据库。“Cmd Sep”选项表示使用什么符号做为分隔符。

例如我们在其中输入插入数据的 SQL 语句，内容如下：

```
insert into user values (1,'Tom');
insert into user values (2,'Jerry');
```

然后单击执行按钮，或者使用它的快捷键“Ctrl+F9”，将会将这两条 SQL 语句执行，执行后在编辑区的下面会显示已更多记录数量。如果 SQL 语句中，存在错误，也会进行报错。

打开控制台区的“Table/Object Info”选项卡下的“Preview”选项，将看到刚执行

SQL 语句插入的两条记录，如图 4-19 所示。

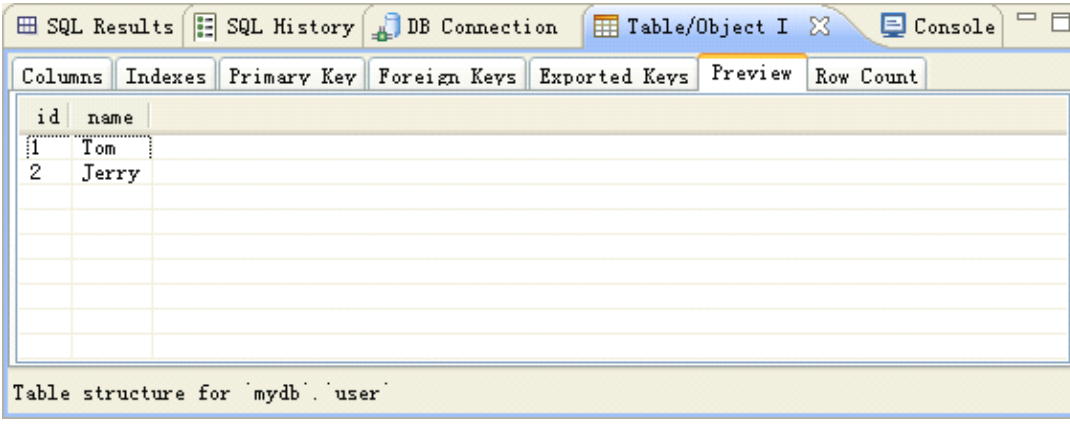


图 4-19　插入记录结果

在控制台区的“SQL History”选项卡中，会将所有已经执行过的 SQL 语句显示出来，如图 4-20 所示。

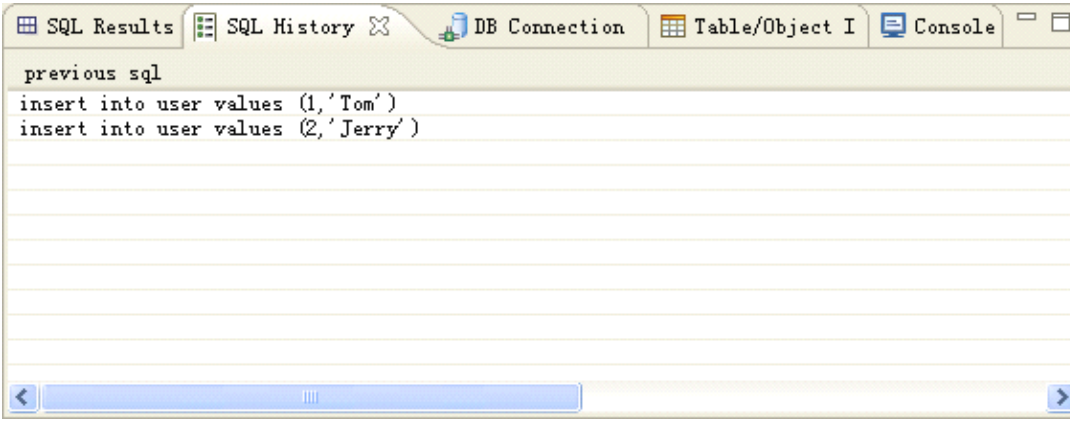


图 4-20　SQL 语句历史记录

在其中选中某条记录，单击鼠标右键，在弹出的菜单中有三个命令。使用“Open in editor”命令，将打开编辑区，并且将当前选中的 SQL 语句放在编辑区中。使用“Remove from history”命令将把该条 SQL 语句从历史记录中删除。使用“Copy to Clipboard”命令将对该条 SQL 语句进行复制，然后可以粘贴到编辑区，或者其他地方。

说明：在 SQL 语句编辑区中可以进行所有的数据库操作，例如创建数据库、创建数据表，以及增删改查操作，但是这些操作大部分都是可以通过界面化的操作完成的，所以在本小节中主要通过插入数据来讲解它的使用。

4.3.5　编辑数据表数据

向数据表中插入数据后，可能还会对已经插入的数据进行修改，这里我们主要来看一下在 MyEclipse 中如何界面化的进行该操作。

首先要选中要进行编辑操作的数据表节点，这里我们选中“user”数据表节点，单击鼠标右键，在弹出的菜单中选择“Edit Data”命令，将在控制台区中出现一个新的“Edit table”选项卡，并在其中将该数据表中已有数据显示出来，如图 4-21 所示。

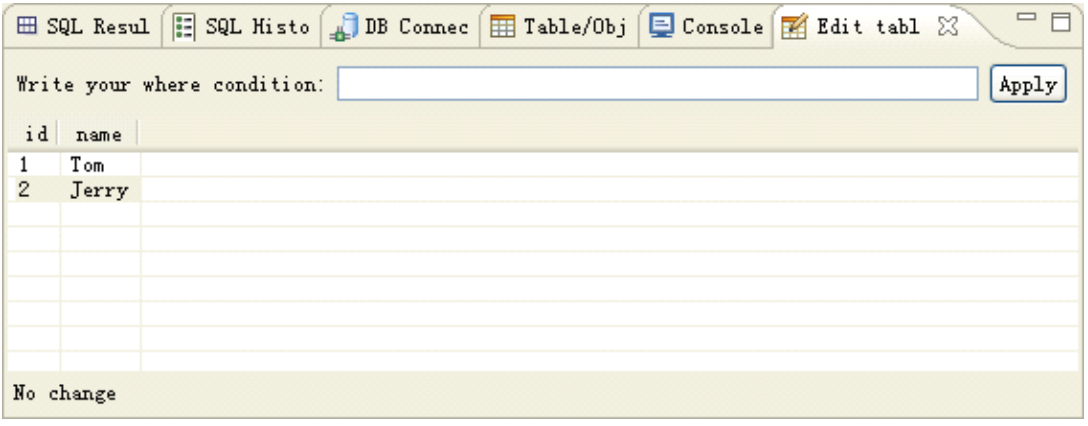


图 4-21　编辑数据

在其中就可以选择已有数据，对这些数据进行修改了。例如我们将“Jerry”改为“Lucy”，按下回车键，将在该界面下显示“Cell updated successfully”信息，表示数据更新成功。

4.3.6　删除数据表所有数据

删除数据表所有数据和删除数据表是不同的两种操作，删除数据表数据是将表下所有数据删除，但不删除表，从而使该表中不再拥有任何数据。

选中要进行删除数据操作的数据表节点，这里仍然以 user 表为例。单击鼠标右键，在弹出的菜单中选择“Delete All Rows”命令，将弹出删除对话框，如图 4-22 所示

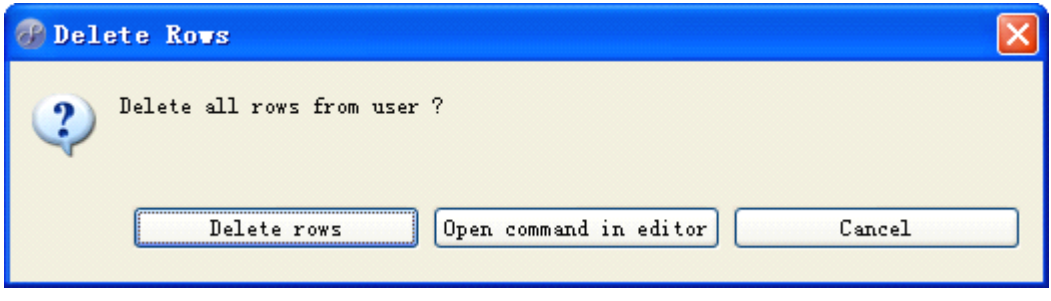


图 4-22　删除数据

其中“Delete rows”按钮表示执行删除数据表所有数据操作。“Open command in editor”按钮表示生成删除的 SQL 语句，并显示在 SQL 语句编辑区中，在编辑区中通过 SQL 语句执行删除操作。“Cancel”按钮表示取消。

4.3.7　创建和删除外键

外键是用于建立和加强表与表之间关系的数据库重要事物，例如购物网站中，用户

会对应一个订单，但是不可能把订单信息都保存到用户表中。通常将它们分别保存在不同的表中，然后在用户表中创建一个外键来连接订单的主键，从而使这两个表建立关系。

为了进行该操作，我们在“mydb”数据库下再创建一个表示订单的数据表，表名为“order”，其中有两个字段，分别是“id”和“price”。创建该表完成后，就可以进行创建外键的操作了。

选中“user”数据表节点，单击鼠标右键，在弹出的菜单中选择“New Foreign Key”命令，将弹出创建外键操作的界面，如图 4-23 所示。

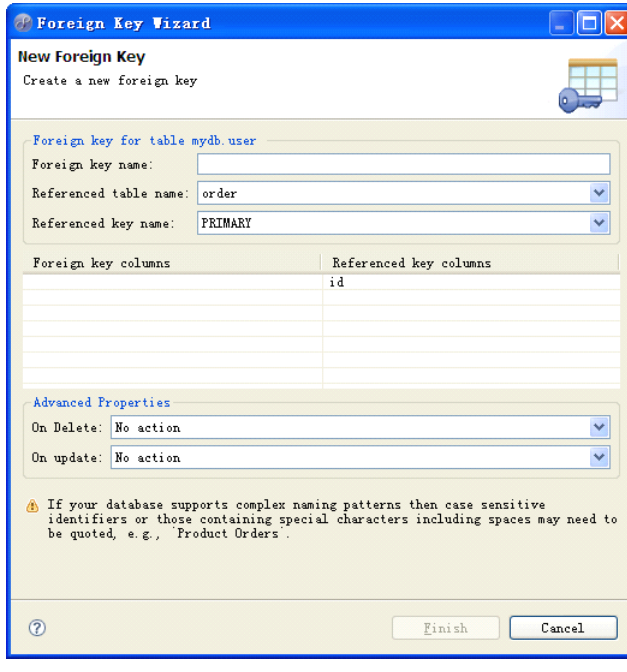


图 4-23　创建外键

其中“Foreign key name”选项表示创建外键的列名，该列名是不能和已有列名重复的，这里填写“orderid”。“Referenced table name”选项表示外键引入的表名，这里选择“order”。“Referenced key name”选项表示外键引入的键名，通常选择“PRIMARY”，表示主键。选择主键后，在下面会自动将主键列名显示出来，在其左面填写“id”。

说明：只所以这里有多行，是因为外键可以是组合外键，由多个列组成。读者可以自己查看数据库相关的资料进行学习。

在“Advanced Properties”选项表示对外键的高级设置，“On Delete”和“On update”表示进行删除和更新操作时，另一个表的操作，其中有四个选项。“No action”表示不执行任何动作，“Cascade”表示级联操作，“Set null”表示将字段设置为 null，“Restrict”表示对该操作进行限制。这里就选择默认的“No action”选项，单击“Finish”按钮将完成创建外键的操作。

在控制台区的“Table/Object Info”选项卡下的“Foreign Keys”选项中，将可以看到创建的外键，如图 4-24 所示。

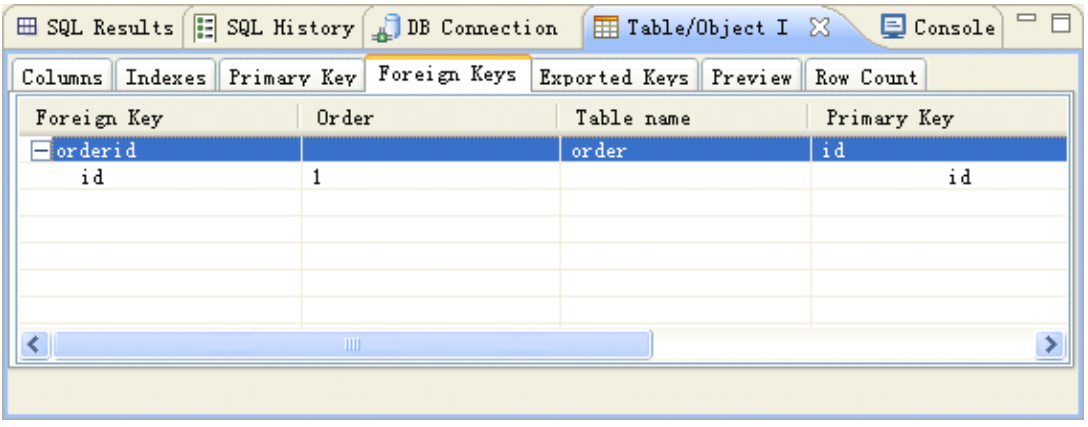


图 4-24　显示外键

可以看到外键已经创建成功。选中创建的外键，单击鼠标右键，在弹出的菜单中有四个命令。其中“Drop Foreign Key”命令表示删除外键，“New Foreign Key”命令表示创建外键，选择该命令后同样会弹出创建外键界面。“Generate DDL”命令表示生成执行该结果的 SQL 语句，例如我们选择该命令，则在 SQL 语句编辑区中将显示：

```
alter table `mydb`.`user`
    add index `orderid`(`id`),
    add constraint `orderid`
    foreign key (`id`)
    references `mydb`.`order`(`id`)
```

“Copy to Clipboard”命令表示将外键信息复制到剪切板中。

说明：有些读者可能会认为使用界面化创建外键，还没有直接写 SQL 语句方便。但是对于初学者和开发中更关注 Java 程序开发的人来学，这种界面化的操作是非常方便的，能够使他们从 SQL 语句中脱离出来。

4.3.8　创建和删除索引

在数据库中，使用索引可以使查询操作不用对整个表进行扫描，就可以在其中找到所需的数据。对于具有庞大数据的数据库而言，索引时必不可少的。同样这里我们对索引不进行过多的讲解，而是更关心如何使用 MyEclipse 进行索引的创建和删除。

选中要创建索引的数据表，例如这里仍然对 user 数据表进行操作。单击鼠标右键，在弹出的菜单中，选择“New Index”命令，将弹出创建索引操作界面，如图 4-25 所示。

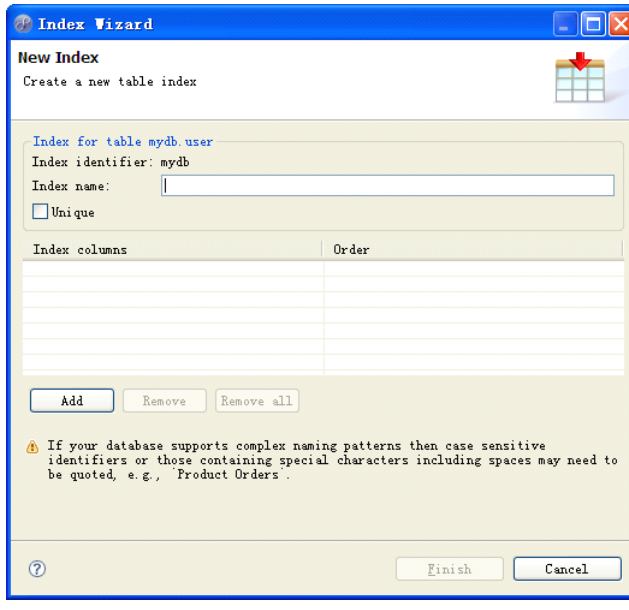


图 4-25　创建索引

其中“Index name”选项表示创建的索引名称，这里填写“user_index”。“Unique”选项表示创建的索引是否唯一的。接下来就是添加索引列，单击“Add”按钮，则在上面将列显示出来。其中“Index columns”表示索引列名，这里选择“name”。“Order”表示索引顺序，其中有两个选项，“Ascending”表示升序，“Descending”表示降序。最后单击“Finish”按钮，则索引创建完成。

在控制台区的“Table/Object Info”选项卡下的“Foreign Keys”选项中，将可以看到创建的索引，如图 4-26 所示。

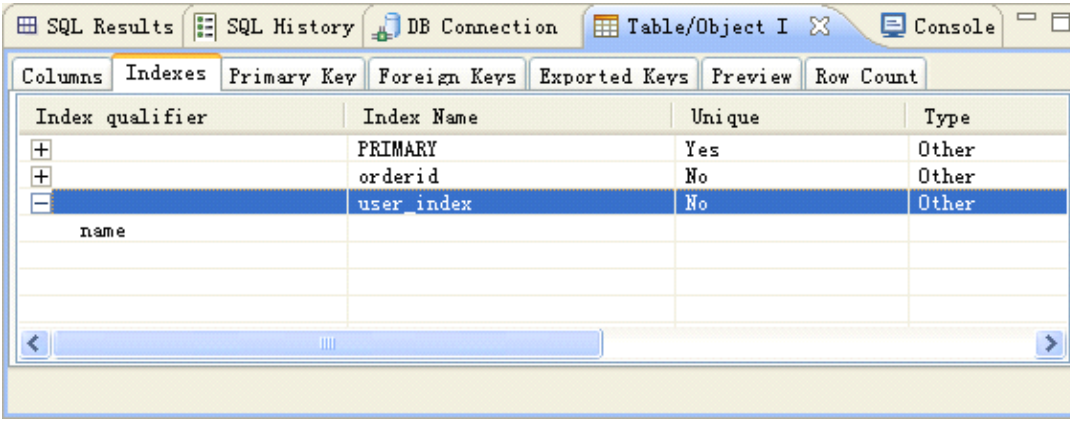


图 4-26　显示索引

和外键一样，选中创建的“user_index”索引，单击鼠标右键，在弹出的菜单中有四个命令。“Drop Index”命令表示删除索引，“New Index”表示创建新索引。“Generate DDL”表示生成创建索引的 SQL 语句，则出现在 SQL 语句编辑区中的内容为：

```
create index `user_index` on `mydb`.`user`(`name`)
```

“Copy to Clipboard”命令表示将索引信息复制到剪切板中。

4.3.9 查询数据表数据

查询也是数据库操作中非常重要的一种，在 MyEclipse 中也集成了该功能。在前面的操作中，可以看到可以查看到数据表中的所有数据。但是如果要查询某列数据，或者按照条件进行查询，就要借助 SQL 语句来完成。

选中要进行查询操作的数据表节点，这里仍然以 user 数据表举例。单击鼠标右键，在弹出的菜单中，选择“Generate”|“Select Statement”命令，将会在 SQL 语句编辑区中出现查询数据库的语句，内容为：

```
select
    `id`,
    `name`
from
    `mydb`.`user`
```

默认给出的 SQL 语句是进行全部查询的。如果想进行部分查询，可以更改查询的列，例如将 SQL 语句改为：

```
select
    `name`
from
    `mydb`.`user`
```

则只讲用户表的所有用户名查询出来。在查询 SQL 语句中也可以加入条件，例如查询姓名为“Tom”的用户，则它的 SQL 语句为：

```
select
    `id`,
    `name`
from
    `mydb`.`user`
where name='Tom'
```

说明：在实际开发中，通常将通过本小节操作得到的 SQL 语句放到程序中。所以在生成以后，一定要运行测试 SQL 语句的正确性。

4.3.10 生成创建数据表的 SQL 语句

在 MyEclipse 中，可以通过 SQL 语句完成数据库操作，也可以通过界面化完成后生产完成该操作的 SQL 语句。在前面学习创建外键和索引时，已经见到这种反向操作。

在前面学习创建数据表时，是通过界面化的方式创建的，在本小节中看一下要生成同样的数据表的 SQL 语句。选中 user 数据表节点，单击鼠标右键，在弹出的菜单中，选择“Generate”|“DDL”命令，将会将 SQL 语句显示在编辑区中。其 SQL 语句的内容为：

```
create table `mydb`.`user`(
    `id` INT not null auto_increment,
  `name` VARCHAR(20),
```

```
    primary key (`id`)
);
```

从生成的 SQL 语句中也可以看到，user 数据表中有两个字段，其中 id 字段是主键，并且它是不能为空、依次递增的。

4.4 使用 JDBC 开发网络商城的数据访问层

在前面的学习中已经对基本的数据库操作进行了讲解，在本节中我们通过一个实际的项目来学习如何使用 MyEclipse 进行快速开发，在这里主要来开发网络商城的数据访问层。

4.4.1 创建项目对应的数据库和数据表

在实际开发中，通常一个项目对应一个数据库，这里就创建一个新的数据库。在 MyEclipse 数据库视图中，打开连接 MySQL 数据库的连接，在连接节点上，单击鼠标右键，选择“Create Database”命令，在弹出的创建数据库界面中，输入如下信息，如图 4-27 所示。

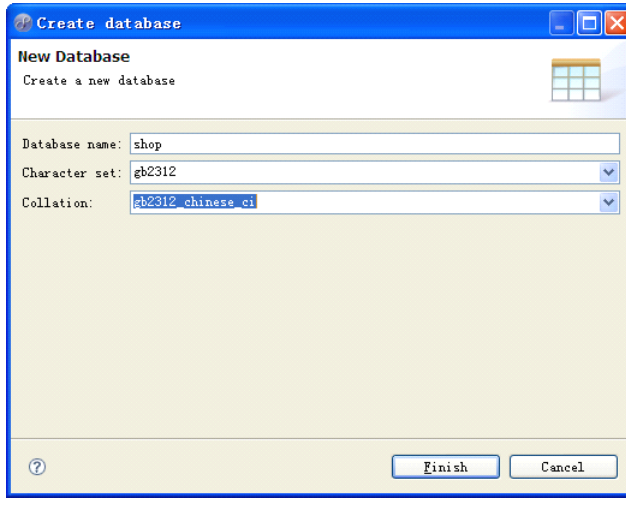


图 4-27　创建网络商城数据库

单击“Finish”按钮将完成数据库的创建。在一个项目中通常由多张数据表，它们都放在一个数据库中。例如网络商城中最少应该具有用户表和商品表，这里仅以其中的商品表举例讲解。

选中新创建的 shop 数据库节点，单击鼠标右键，在弹出的菜单中选择“new Table”命令，将弹出创建数据表界面，输入必要信息后，如图 4-28 所示。

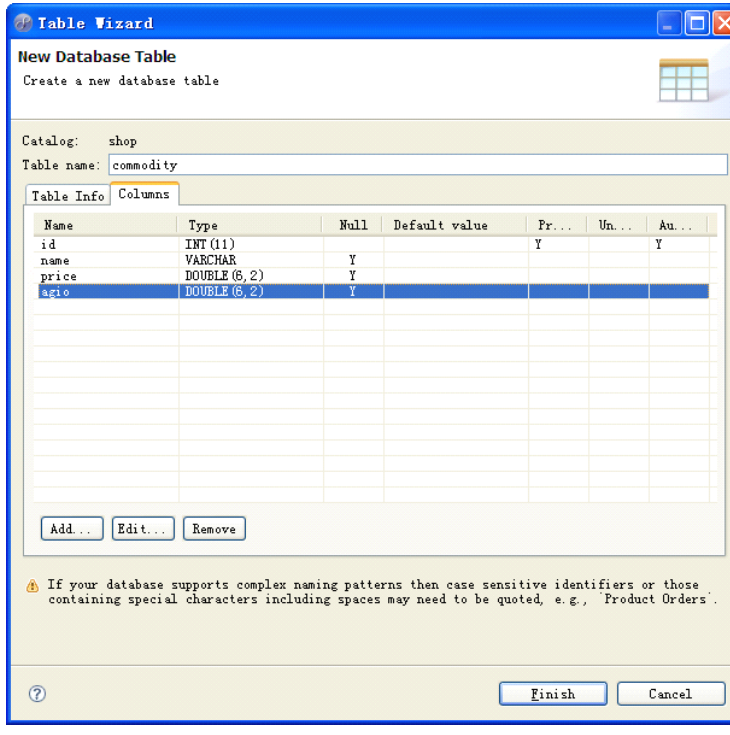


图 4-28　创建商品数据表

在商品数据表中，定义了商品 id、商品名称、商品价格、商品折扣价等四个字段，这些都可以根据实际需求进行增加，例如所属种类等信息。单击“Finish”按钮，完成商品数据表的创建。

4.4.2　创建 Java 项目并导入驱动包

数据库层的相关数据表创建完成后，在 MyEclipse 中，将视图切换为 Java 程序开发视图。因为我们是开发的项目中的数据访问层，所以这里创建 Java 项目就可以。

在 MyEclipse 菜单中，选择“File”|“New”|“Java Project”命令，将弹出创建 Java 项目的界面。为了让读者更容易在光盘中找到本项目，这里将项目名称定义为“ch4”，在实际开发中，该名称并不规范的。其他设置如图 4-29 所示。

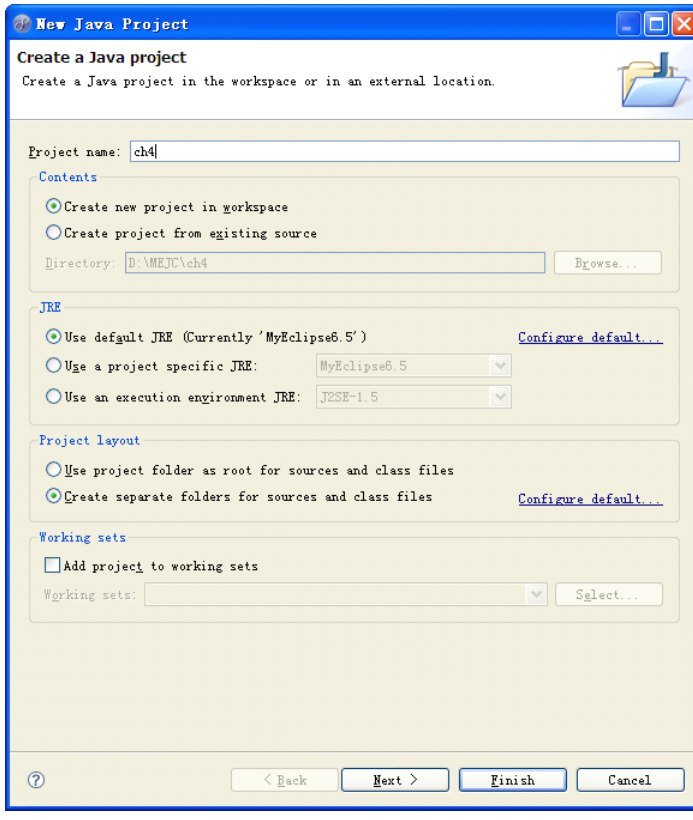


图 4-29　创建网络商城项目

单击“Finish”按钮将完成本项目的创建。因为本项目时要连接数据库的，所以接下来就是将 MySQL 驱动包导入到项目中。复制 MySQL 驱动包，选择“ch4”项目根目录，然后执行粘贴操作，从而将 JAR 复制到项目中。

选中项目中的 JAR 包，单击鼠标右键，在弹出的菜单中，选择“Build Path”|“Add to Build Path”命令，从而将该 JAR 包导入到 Build Path 中，从而完成驱动包的导入。

4.4.3　创建数据库封装类

在进行 JDBC 开发中，有很多重复执行的操作，例如加载驱动、建立数据库连接和关闭数据库资源等。在实际开发中，通常将这些操作封装到一个类中，每一个操作对应一个独立方法，当需要使用该操作时，调用相应的方法来完成。

在 MyEclipse 中并没有提供创建数据库封装类的快速操作，但是值得庆幸的是该类重用性非常强。在不同项目中使用该封装类时，几乎不用改动，或者改动非常少的代码。

在实际开发中，是不会将所有的类放在同一个包中的，而是按照类的功能不同，将它们放在不同包中。例如数据库封装类在项目中起到工具的作用，所以通常把它放在 util 包中。在包名中通常还会加入域名信息，例如“com.sanqing.util”包，在部署时，该项目的域名通常就是“www.sanqing.com”。

创建包和类是非常简单的，而且在前面的学习中已经多次提到，这里就不再给出它们的创建方法。接下来我们就来看一下数据库封装类的代码。

```
01  package com.sanqing.util;
02  import java.sql.Connection;
03  import java.sql.DriverManager;
04  import java.sql.PreparedStatement;
05  import java.sql.ResultSet;
06  import java.sql.SQLException;
07  public class DBConnection {
08      private static final String DBDRIVER = "com.mysql.jdbc.Driver" ;  //驱动串
09      private static final String DBURL = "jdbc:mysql://localhost:3306/shop";//连接 URL
10      private static final String DBUSER = "root" ;                    //用户名
11      private static final String DBPASSWORD = "admin";                //数据库密码
12      public static Connection getConnection(){
13          Connection conn = null;                          //声明一个连接对象
14          try {
15              Class.forName(DBDRIVER);                     //注册驱动
16          conn = DriverManager.getConnection(DBURL,DBUSER,DBPASSWORD);
17          } catch (ClassNotFoundException e) {
18              e.printStackTrace();
19          } catch (SQLException e) {
20              e.printStackTrace();
21          }
22          return conn;
23      }
24      public static void close(Connection conn) {
25          if(conn != null) {                               //如果 conn 连接对象不为空
26              try {
27                  conn.close();                            //关闭 conn 连接对象对象
28              } catch (SQLException e) {
29                  e.printStackTrace();
30              }
31          }
32      }
33      public static void close(PreparedStatement pstmt) {
34          if(pstmt != null) {                              //如果 pstmt 预处理对象不为空
35              try {
36                  pstmt.close();                           //关闭 pstmt 预处理对象
37              } catch (SQLException e) {
38                  e.printStackTrace();
39              }
40          }
41      }
42      public static void close(ResultSet rs) {
43          if(rs != null) {                          //如果 rs 结果集对象不为 null
44              try {
45                  rs.close();                              //关闭 rs 结果集对象
```

```
46                      } catch (SQLException e) {
47                          e.printStackTrace();
48                      }
49                  }
50              }
51      }
```

其中不同开发环境中需要改动的只有第 9 行到第 11 行，其中第 9 行是连接数据库的 URL，根据数据库不同需要进行改动。第 10 行和第 11 行是连接数据库用到的用户名和密码，根据本地数据库的设置进行对应修改。

4.4.4 创建商品实体类

通常数据库中的一张数据表就对应着一个实体类。例如现在数据库中有商品数据表，我们就应该创建一个商品实体类，通过这个类来完成对商品的增删改查等操作。

在项目中，通常将实体类放在“bean”包下，如果实体类比较多，又可以根据实体类的用途分为“vo”、“po”等包。

实体类中的属性通常是和数据表中的字段相对应的，则商品实体类的程序代码为：

```
01  package com.sanqing.bean;
02  public class Commodity {
03      private int id;             //商品 ID
04      private String name;    //商品名称
05      private double price;   //商品价格
06      private double agio;    //商品折扣
07  }
```

为了获取和设置属性值，还要为商品实体类中的属性定义 Getter 和 Setter 方法。在商品类体中，单击鼠标右键，在弹出菜单中，选择“Source”|“Generate Getters and Setters”命令，就会弹出生成 Getter 和 Setter 方法的界面，选中所有属性，如图 4-30 所示。

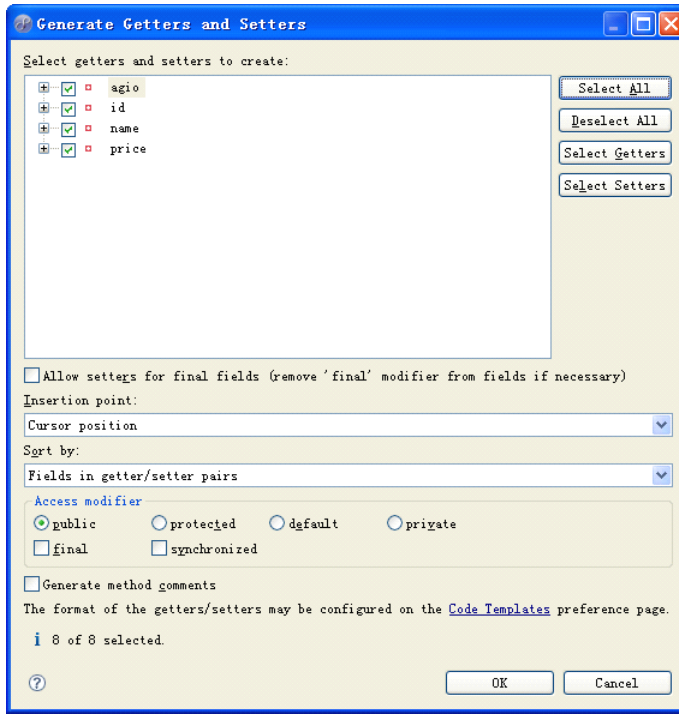


图 4-30　创建实体类

单击“OK”按钮将完成实体类的创建，商品实体类的代码变为：

```
01  package com.sanqing.bean;
02  public class Commodity {
03      private int id;                 //商品 ID
04      private String name;   //商品名称
05      private double price;   //商品价格
06      private double agio;    //商品折扣
07      public int getId() {
08          return id;
09      }
10      public void setId(int id) {
11          this.id = id;
12      }
13      public String getName() {
14          return name;
15      }
16      public void setName(String name) {
17          this.name = name;
18      }
19      public double getPrice() {
20          return price;
21      }
22      public void setPrice(double price) {
```

```
        this.price = price;
    }
    public double getAgio() {
        return agio;
    }
    public void setAgio(double agio) {
        this.agio = agio;
    }
}
```

可以看到，通过 MyEclipse 的命令自动生成很多行代码，并且生成一个标准的实体类程序。

4.4.5 创建数据访问层实现类

数据访问层实现类是最直接操作数据库的程序，在其中定义了操作数据库的方法，包括增加、删除、更新和查询操作方法。其中查询通常由两种方法，一种是查询全部，另一种是根据 id 主键查询。

数据访问层实现类的命名有通用规范，例如本节创建的“CommodityDAOImpl”类，其中“Commodity”是和实体类类名对应，表示对该商品进行数据库操作；“DAO”是数据访问的缩写；“Impl”是实现的缩写。接下来就来看一下“CommodityDAOImpl”类的程序代码。

```
01  package com.sanqing.dao;
02  //省略导入接口和类的代码
03  public class CommodityDAOImpl {
04      //实现添加商品的方法
05      public void addCommodity(Commodity commodity) {
06          Connection conn = DBConnection.getConnection();      //获得连接对象
07          String addSQL = "insert into
08                          commodity(name,price,agio) values(?,?,?)";
09          PreparedStatement pstmt = null;                      //声明预处理对象
10          try {
11              pstmt = conn.prepareStatement(addSQL);    //获得预处理对象并赋值
12              pstmt.setString(1, commodity.getName());  //设置第一个参数
13              pstmt.setDouble(2, commodity.getPrice()); //设置第二个参数
14              pstmt.setDouble(3, commodity.getAgio());  //设置第三个参数
15              pstmt.executeUpdate();                    //执行更新
16          } catch (SQLException e) {
17              e.printStackTrace();
18          } finally{
19              DBConnection.close(pstmt);                //关闭预处理对象
20              DBConnection.close(conn);                 //关闭连接对象
21          }
22      }
23      //省略其他操作方法
```

```
24    }
```

该程序在开发中，是需要手动完成的，根据实际需求来定义不同的方法。在本小节数据访问层实现类中定义了增加商品、删除商品、更新商品、查询所有商品和按照商品 ID 查询商品 5 种方法，这里仅给出增加商品方法，其他方法代码见光盘。

注意：定义数据库操作方法时，通过将参数定为对应的实体类类型，如果这里定义了是具体信息参数，例如商品名称、商品价格等，就可以使用引入参数对象的重构操作，进行替换。

4.4.6 创建数据访问层接口

数据访问层接口的命名也是遵循规范的，通常由实体类名加“DAO”来命名，例如“CommodityDAO”。数据访问层接口可以和实现类一起放在“dao”包中，也可以将实现类单独放在“dao”包的“impl”子包下。

数据访问层接口是可以通过 MyEclipse 的集成功能自动生成的。在包资源管理器中，选中“CommodityDAOImpl”数据访问层实现类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，将其中的方法全部选中，如图 4-31 所示。

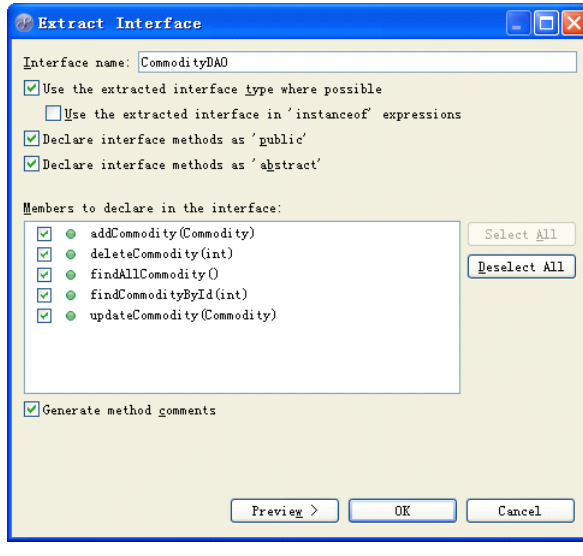


图 4-31　抽取数据访问层接口

单击“OK”按钮将完成数据访问层接口的抽取。在“dao”包下将创建名称为“CommodityDAO”的数据访问层接口程序，其代码为：

```
01    package com.sanqing.dao;
02    import java.util.List;
03    import com.sanqing.bean.Commodity;
04    public interface CommodityDAO {
05        public abstract void addCommodity(Commodity commodity);
06        public abstract void deleteCommodity(int commodityId);
```

```
07          public abstract List<Commodity> findAllCommodity();
08          public abstract void updateCommodity(Commodity commodity);
09          public abstract Commodity findCommodityById(int commodityId);
10      }
```

并且在数据访问层的实现类中实现该接口。

4.4.7 创建数据访问层工厂类

在业务访问层中通常借助工厂类来创建数据访问层实现类对象。在本小节中就来看一下如何快速创建数据访问层的工厂类。在操作前，要完成准备的工作，首先在数据访问层实现类中定义一个无参构造函数。然后定义工厂类，工厂类通常不和接口和实现类放在同一包下，而是自定义一个“factory”包，工厂类的代码为：

```
01  package com.sanqing.factory;
02  public class CommodityDAOFactory {
03
04  }
```

接下来就是定义工厂类中的创建实现类对象的方法。在包资源管理器中，选中“UserDAOImpl”的无参构造函数节点，单击鼠标右键，在弹出的右键菜单中，选择“Refactor”|“Introduce Factory”命令，将弹出引入工厂操作界面，填入如下信息后，如图 4-32 所示。

图 4-32　创建工厂方法

单击“OK”按钮，将完成工厂类的创建，则工厂类的代码将变为：

```
01  package com.sanqing.factory;
02  import com.sanqing.dao.CommodityDAOImpl;
03  public class CommodityDAOFactory {
04          public static CommodityDAOImpl createCommodityDAOImpl() {
05                  return new CommodityDAOImpl();
06          }
07  }
```

通常该操作生成的工厂方法并不是规范的，其中方法的返回值为实现类类型，这就失去了工厂类的作用。工厂类的作用就是让业务逻辑层中并不关心数据访问层实现类，

只需要关心数据访问层接口。

如果想变为规范的方法，还需要经过“尽可能使用超类型”的重构操作。在包资源管理器中，选中“CommodityDAOImpl”类节点，单击鼠标右键，在弹出的菜单中选择“Refactor”|“Use Supertype Where Possible”命令，将弹出选择超类型的界面，选择“CommodityDAO”接口节点，如图 4-33 所示。

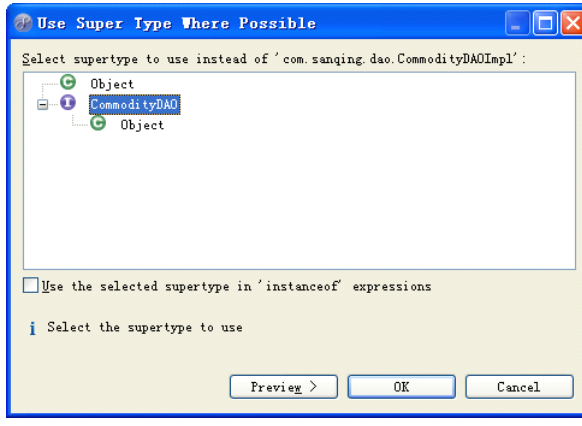


图 4-33　选择接口类型

单击“OK”按钮，将完成类型的转换。这是数据访问层工厂类的代码变为：

```
01  package com.sanqing.factory;
02  import com.sanqing.dao.CommodityDAO;
03  import com.sanqing.dao.CommodityDAOImpl;
04  public class CommodityDAOFactory {
05      public static CommodityDAO createCommodityDAOImpl() {
06          return new CommodityDAOImpl();
07      }
08  }
```

这就变为一个规范的工厂类。

4.4.8　开发数据访问层测试程序

到目前为止，网络商城的数据访问层应该算开发完成了。但是在实际开发中，通常开发一部分程序后，要对它进行测试，以验证该部分程序的正确性。

测试程序仅用于开发阶段，当进行项目部署时，通常就会将测试程序删除。本节讲解的网络商城数据访问层的测试程序代码如下：

```
01  package com.sanqing.test;
02  import com.sanqing.bean.Commodity;
03  import com.sanqing.dao.CommodityDAO;
04  import com.sanqing.factory.CommodityDAOFactory;
05  public class Test {
06      public static void main(String[] args) {
07          CommodityDAO commodityDAOImpl =
08                  CommodityDAOFactory.createCommodityDAOImpl();
```

```
09              Commodity commodity = new Commodity();
10              commodity.setId(2);
11              commodity.setName("food");
12              commodity.setPrice(58.50);
13              commodity.setAgio(36.00);
14              commodityDAOImpl.addCommodity(commodity);
15          }
16      }
```

该程序是需要手动编写的，当使用到其他包中的接口或者类时，可以使用“导入接口和类”命令将它们导入。

在代码的第 7 行中调用工厂类中的创建实现类对象方法，并使用接口引用指向它。在第 9 行中创建了商品对象，并依次将 id、商品名称、商品价格、商品折扣价等信息设置到商品对象中。在第 14 行中调用实现类中的 addCommodity 方法，从而完成商品的添加。

在 MyEclipse 中切换到数据库视图中，在控制台区的“Table/Object Info”选项卡下的“Preview”选项，将看到执行程序后插入的商品记录，如图 4-34 所示。

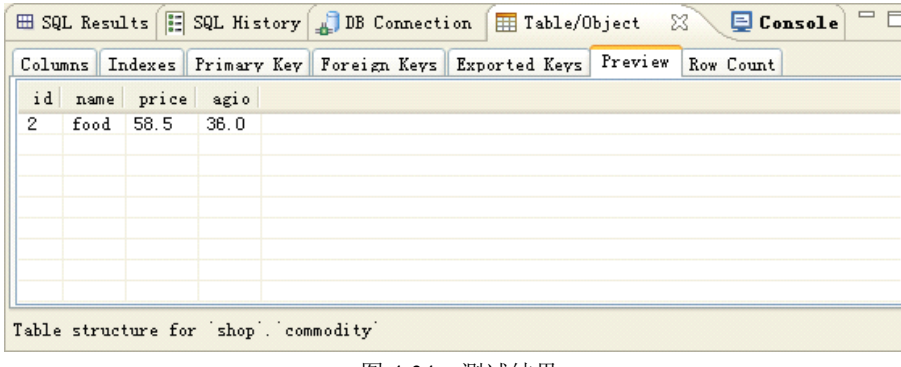


图 4-34　测试结果

从该结果可以看到，数据访问层开发正确，可以放到项目中做为项目的一部分。

第5章　管理服务器并进行Web开发

在上一章中已经学习了如何使用 MyEclipse 对数据库进行管理，在本章中将主要学习如何对服务器进行管理，这里所说的服务器是指用于 Java Web 项目部署的应用服务器。除了学习服务器的管理外，还会学习最简单的 Web 开发，包括 JSP、Servlet 等 Web 程序的开发。

5.1　认识 MyEclipse 中的服务器

在配置外部数据库时，我们已经看到在 MyEclipse 中能够管理很多种数据库。同样在 MyEclipse 中可以管理很多种应用服务器，例如常用的 Tomcat、Jboss、WebLogic 和 WebSphere 等。

注意：在 MyEclipse 中自带了一个 Tomcat 服务器，通过该服务器可以发布常见的 Web 项目，例如 JSP、SSH 的项目，但是不能够对 EJB 项目进行发布的。

在 MyEclipse 的界面的控制台区中，有一个“Servers”选项卡，在这里可以查看目前 MyEclipse 中能够使用哪些服务器。在控制台区中，默认是由该选项卡的，如果不存在，也可以在 MyEclipse 的菜单中，选择“Window”|“Show View”|“Servers”命令可以显示，如果在“Show View”选项下没有“Servers”命令，可以在“Other”命令的弹出界面中选择。“Servers”界面如图 5-1 所示。

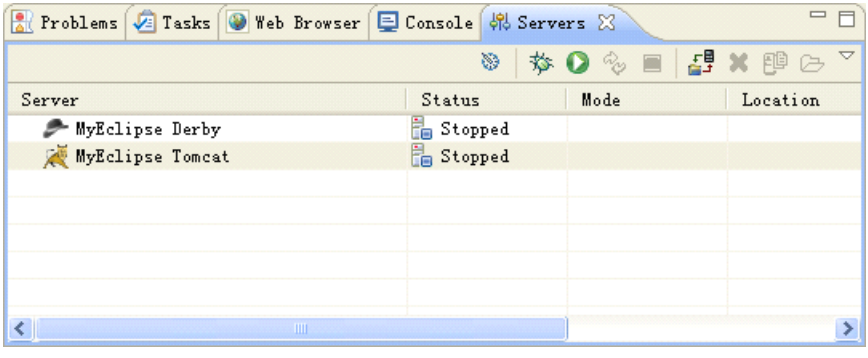


图 5-1　Servers 界面

其中“MyEclipse Derby”是 MyEclipse 中内置的数据库，而“MyEclipse Tomcat”是 MyEclipse 中内置的应用服务器。在它们的上面有几个操作按钮，其中使用比较多的就是其中的第 3 个按钮，表示以运行模式启动服务器。

对应用服务器进行操作，也可以单击鼠标右键，在弹出的菜单中，“Configure”命令表示配置服务器的选择器；“Debug Server”命令表示以调试模式启动服务器；“Run Server”命令表示以运行模式启动服务器；“Restart”命令表示重新启动服务器；“Stop”

命令表示停止服务器；“Add Deployment”命令表示向选中的服务器中发布项目。在后面的学习中，使用到这些命令后，还会进行详细的讲解。

5.2 管理外部 Tomcat 服务器

MyEclipse 中的内置服务器是不能完成所有 Java Web 项目的操作的，所以我们有必要学习一下如何在 MyEclipse 中对外部的应用服务器进行管理。这里通过外部的 Tomcat 服务器讲解该操作。

5.2.1 配置 Tomcat 路径

在 MyEclipse 的菜单中，选择“Window”|“Preferences”命令，将打开首选项设置界面，在常用操作一章已经多次使用到该界面。在首选项设置界面的导航栏中，选择“MyEclipse Enterprise Workbench”|“Servers”|“Tomcat”|“Tomcat 6.x”命令，则当前首选项界面如图 5-2 所示。

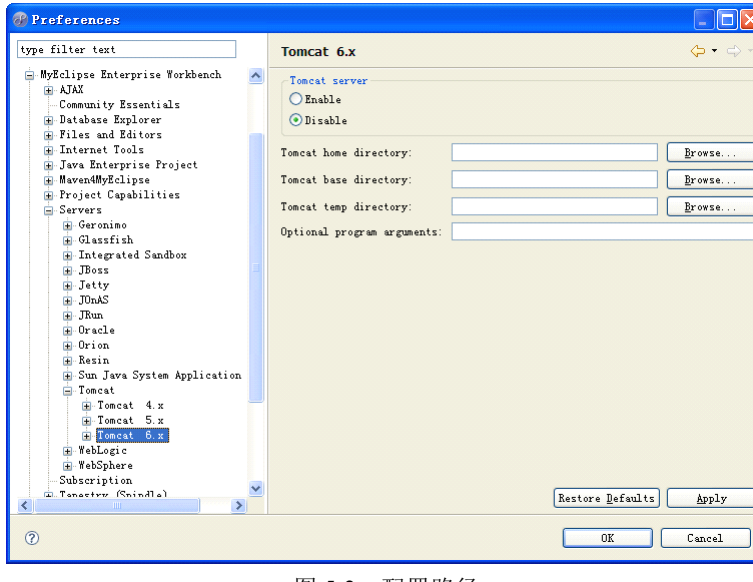


图 5-2 配置路径

其中“Tomcat home directory”表示 Tomcat 的主目录，“Tomcat base directory”表示 Tomcat 的基础目录，“Tomcat temp directory”表示 Tomcat 的临时参数目录。笔者的 Tomcat 服务器安装在 D 盘下，所以“Tomcat home directory”选项下选择“D:\apache-tomcat-6.0.10”，读者可以根据本地安装位置进行填写。“Optional program arguments”选项表示 Tomcat 的相关参数，通常这一项是不需要填写的，都是采用默认值。

说明：前三个选择 Tomcat 目录的选项是级联关系的，当选择其中一个后，其他的会自动填写。例如填写 Tomcat 的主目录为“D:\apache-tomcat-6.0.10”后，将自动填写临时参数目录为“D:\apache-tomcat-6.0.10\temp”。

5.2.2 启动

仅配置外部 Tomcat 的路径，单击“OK”按钮后，该服务器并不会出现在服务器界面中，这是因为还没有启动该配置。

在配置路径的上面有两个单选按钮选项，其中“Enable”选项表示启动外部 Tomcat 的配置，“Disable”选项表示不启动外部 Tomcat 的配置。默认是不启动的状态，所以要选择“Enable”选项。这是再单击“OK”按钮，就完成对外部 Tomcat 服务器的配置。

技巧：在实际开发中，不同的项目可能需要的服务器不同，这时候就需要将不需要的服务器停止，从而不影响 MyEclipse 的速度。这种操作主要就是通过“Enable”选项和“Disable”选项完成的，这样的好处就是下次使用时，不需要重新配置服务器。

5.2.3 添加 JDK 支持

对于 Tomcat 6.x 版本的应用服务器来说，经过上面两步的操作就可以在 MyEclipse 中使用了，因为它将使用其中默认的 JRE。但是对于一些低版本的服务器，或者一些特殊的服务器，还需要为该服务器添加 JDK 支持。例如 Tomcat 4 版本就需要 JDK 1.4 的支持，这时候就要为它指定 JDK。

在首选项设置界面中，单击“Tomcat 6.x”节点前的加号，选择“JDK”命令，将出现添加 JDK 支持的界面，如图 5-3 所示。

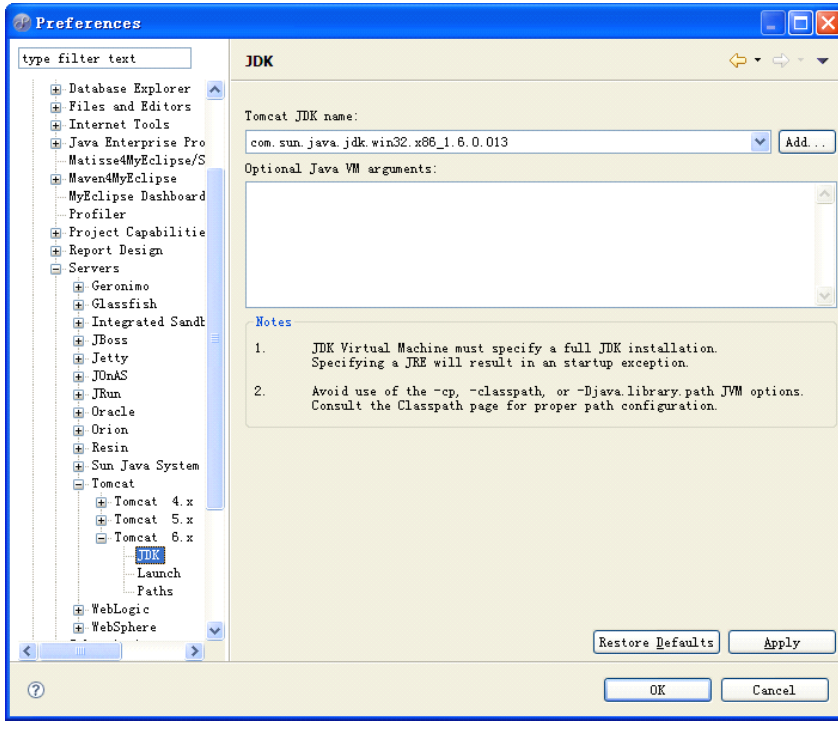


图 5-3 添加 JDK 支持

在其中“Tomcat JDK name”选项中可以选择需要使用的 JDK，默认是 MyEclipse 6.5。默认安装下，是没有其他 JDK 版本的，需要单击“Add”按钮完成其他 JDK 的添加，

这时候将弹出添加 JDK 的界面，如图 5-4 所示。

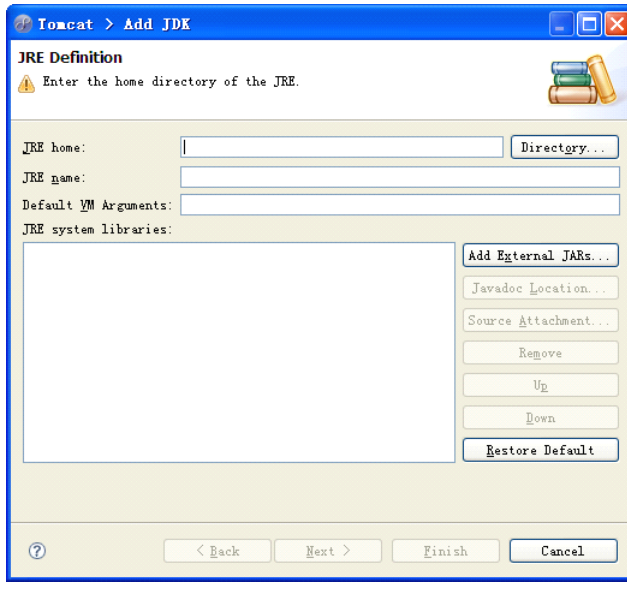


图 5-4　添加新的 JDK

该界面是和添加外部 JRE 非常类似的。单击其中的“Browse”按钮，选择 JDK 安装的根目录，如果是默认安装，则路径应该为“C:\Program Files\Java\jdk1.6.0_10”。选择 JDK 路径后，则其他选项将自动填写。单击“OK”按钮，将完成外部 JDK 的配置，从而在添加 JDK 支持界面中将多出一个 JDK 选项。

注意：在添加新的 JDK 界面中，设置的路径一定要选择 JDK 的根目录，而不是 JRE 的根目录。这和添加外部 JRE 是不一样的。

回答首选项设置界面中，单击“OK”按钮，将完成外部 Tomcat 应用服务器的添加，这时候再查看“Servers”界面，如图 5-5 所示。

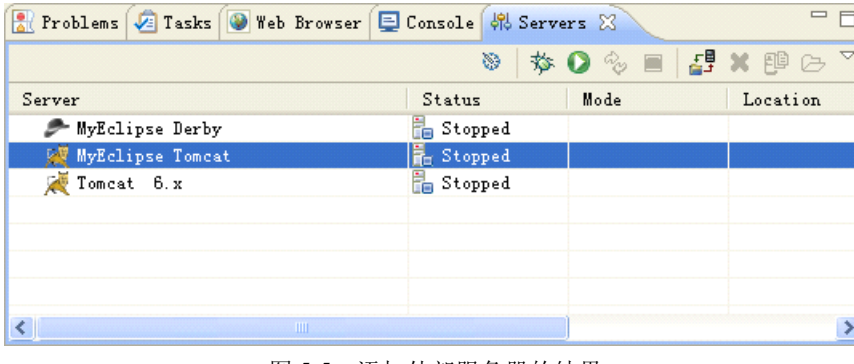


图 5-5　添加外部服务器的结果

可以看到在其中多出了一个服务器，正是前面添加操作中添加的外部 Tomcat 应用服务器。在后面的学习中都是基于该服务器操作的。

5.3 开发 Web 项目中的 HelloWorld

因为服务器的大部分操作都是对 Java Web 项目进行操作，所以有必要先来学习一下在 MyEclipse 中如何进行 Java Web 项目的开发。和开发 Java 项目一样，在 MyEclipse 中也集成了大量对 Java Web 项目支持的功能。

5.3.1 创建 Web 项目

我们首先来学习一下该如何创建 Web 项目。在 MyEclipse 的菜单中，选择“File”|“New”|“Web Project”命令，将弹出创建 Web 项目界面，如图 5-6 所示。

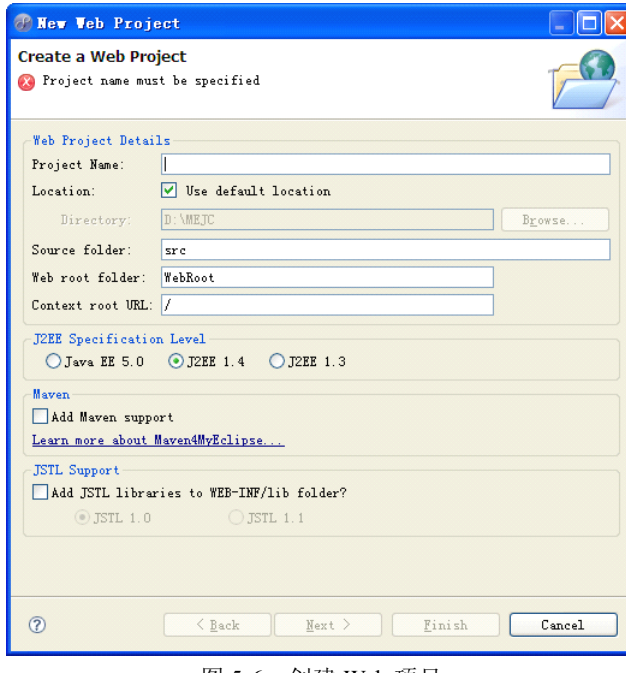


图 5-6　创建 Web 项目

其中“Project Name”选项表示项目的名称，这里填写“FirstWeb”。“Location”选项表示项目文件存放的位置，默认会选中“Use default location”，表示保存在当前工作空间中；也可以取消选中状态，然后在“Directory”中选择保存的目录位置。

“Source folder”选项表示项目中 Java 源代码文件的保存目录，其中主要包括 Java 包和“.java”的文件，通常采用默认的“src”目录。“Web root folder”选项表示 Web 相关程序的存放位置，这些文件包括 JSP 程序、HTML 程序等，以及固定的 WEB-INF 目录，通常使用默认的“WebRoot”目录。“Context root URL”选项表示发布项目后，使用的访问路径，默认是项目名称，这里会随着项目名称的改变而改变。

“J2EE Specification Level”选项表示 J2EE 规范版本的选择，这里的版本取决于发布应用服务器的类型和版本。“Add JSTL libraries to WEB-INF/lib folder”选项表示是否要将 JSTL 相关包加入到项目中。如果选择了“Java EE 5.0”选项，则该选项是不用选择的，会自动完成该操作。

说明：Tomcat 6.0 等目录主流应用服务器都执行目前最高版本“Java EE 5.0”，所以通常选择该版本。如果选择低版本，后面的学习中会多添加很多操作，所以在后面的学习中不特殊说明，都是创建的“Java EE 5.0”版本 Java Web 程序。

最后单击“Finish”按钮，将完成第一个 Java Web 项目的创建。并且将创建后的项目显示在包资源管理中，如图 5-7 所示。

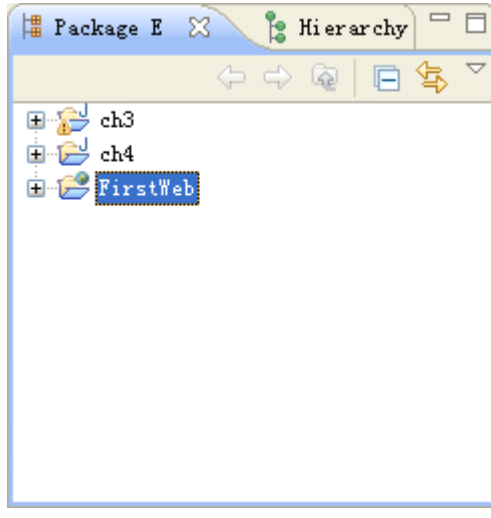


图 5-7　创建 Web 项目后

5.3.2　浏览生成的项目

经常上一小节的操作，将在包资源管理器中创建一个名称为“FirstWeb”的 Java Web 项目，单击每一层前面的加号，将会将其中的具体文件显示出来，如图 5-8 所示。

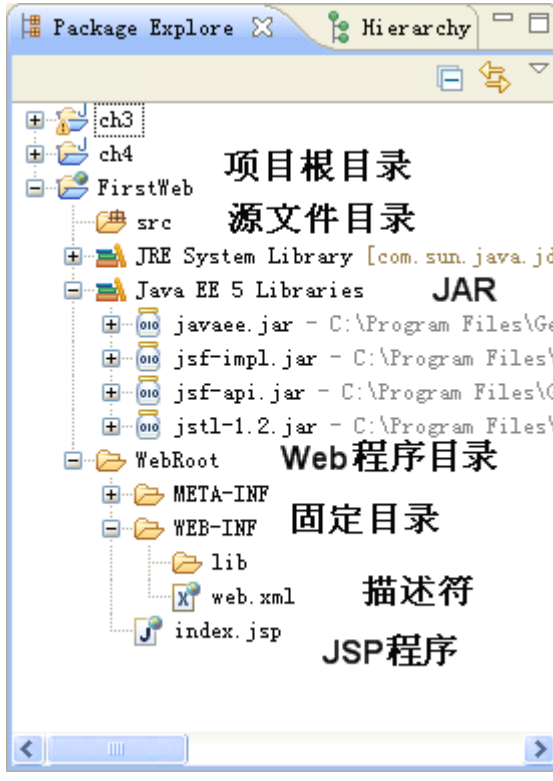


图 5-8　创建项目后目录

说明：这是将大部分选项以默认的形式创建的，所以会有 src 和 WebRoot 目录，如果创建时进行自定义设置，这里也会发生改变，如果没有特别说明，后面的项目也采用默认的命名。

其中“src”目录用于存放 Java 源文件，JavaBean 和 Servlet 等程序都会保存在该目录下。创建 Java Web 项目后，会自动将 Web 所需的 JAR 包导入到项目中，例如其中的“jstl-1.2.jar”文件就是开发 JSTL 标签用到的 JAR 包。

“WebRoot”目录是用于放置 Web 程序的目录，JSP、HTML 等程序都可以直接放在它的根目录下，在发布和部署 Java Web 项目时，就是发布的该目录。

注意：发布 Java Web 项目时，“src”目录是不会发布的，而是仅将编译后的.class 文件发布。编译后的.class 文件放在“WebRoot/WEB-INF/classes”目录下。讲解发布时，还会更详细的看到这一点。

“WEB-INF”目录是 Java Web 项目的固定目录，名称和位置都是固定的。在该目录下，“lib”目录用于保存项目所用到的 JAR 包，项目中的 JAR 包在发布时都会放在该目录下。“classes”目录是不会在 MyEclipse 中显示出来的，只有在发布后才会看到该目录，在其中编译后的.class 文件。“web.xml”文件是 Java Web 项目的描述符，在其中配置项目的相关内容，使用最多的就是对 Servlet 等程序进行必要配置。

“index.jsp”文件是一个 JSP 程序，它通常也是 Java Web 项目部署后的访问首页。JSP 等页面显示程序可以直接放在“WebRoot”目录下，也可以在该目录下创建子目录，

将 JSP 程序放在子目录中。

5.4 发布项目

在前面对 Tomcat 服务器和 Java Web 项目有所了解后，在本节中就来看一下如何将 Java Web 项目发布到 Tomcat 服务器中。在本节中，主要讲解发布方式、发布操作和如何运行发布后的项目等内容。

5.4.1 发布方式

在 MyEclipse 中，支持两种发布 Java Web 项目的方式，分别是散包发布和打包发布，这两种发布方式主要是根据项目的不同阶段来选择。

散包发布一般用于开发阶段，在 MyEclipse 中使用这种方式发布时，会将项目中的目录和文件按照 Java EE 规范发布到应用服务器的发布目录下。散包发布的优点就是能够实时同步，执行一次发布以后，如果在 MyEclipse 中对 JSP 程序进行了修改，会自动使用修改后的文件替换服务器中的原文件。这时候的项目不用重新发布，只需要刷新页面就能够看到修改后的效果。

注意：并不是所有的服务器都支持散包发布的，在 Java EE 规范中规定项目要以打包的形式进行发布。但是目前主流的服务器，例如 Tomcat、Jboss 等都执行散包发布，所以在开发阶段就可以使用这种方步方式，在本书中也主要采用这种方式。

打包发布是一种最规范的发布方式，适用于所有服务器，它主要用于最后项目部署时。使用打包方式时，会将项目中的文件打包成单个的 ZIP 文件，后缀是比较灵活的，可以使 WAR 或者 JAR，将该文件放到应用服务器的发布目录下，就完成发布操作。打包发布的缺点也就是散包发布的优点，那就是不能够实时同步。

5.4.2 指定项目发布

发布项目可以通过多种操作完成，在本书中主要讲解两种最常用的发布操作，一种是指向项目发布，另一种是指定服务器发布。在本小节中就先来讲解指定项目发布。

在包资源管理器中，选中要发布的项目，例如选中“FirstWeb”项目。单击 MyEclipse 工具栏中的  发布按钮，将弹出指定项目发布界面，如图 5-9 所示。

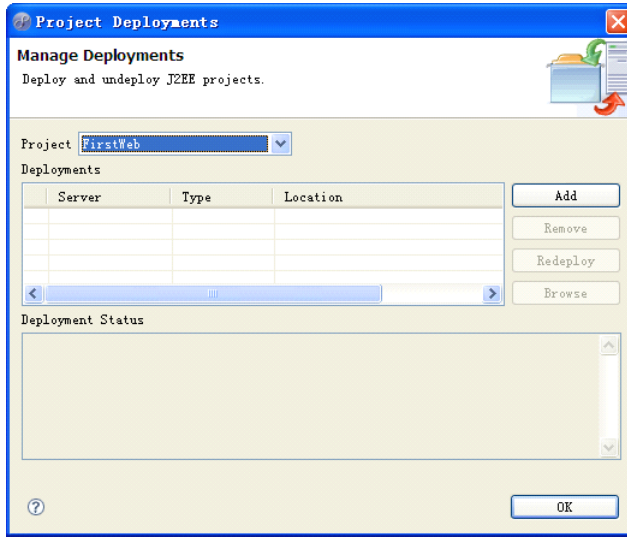


图 5-9　指定项目发布

其中“Project”选项表示要发布的项目，默认是前面操作中选中的项目，也可以在其中选择，这里会将当前工作空间的 Java Web 项目都列出来。“Delployments”表示发布到哪一个应用服务器下，单击“Add”按钮，将弹出选择服务器的界面，如图 5-10 所示。

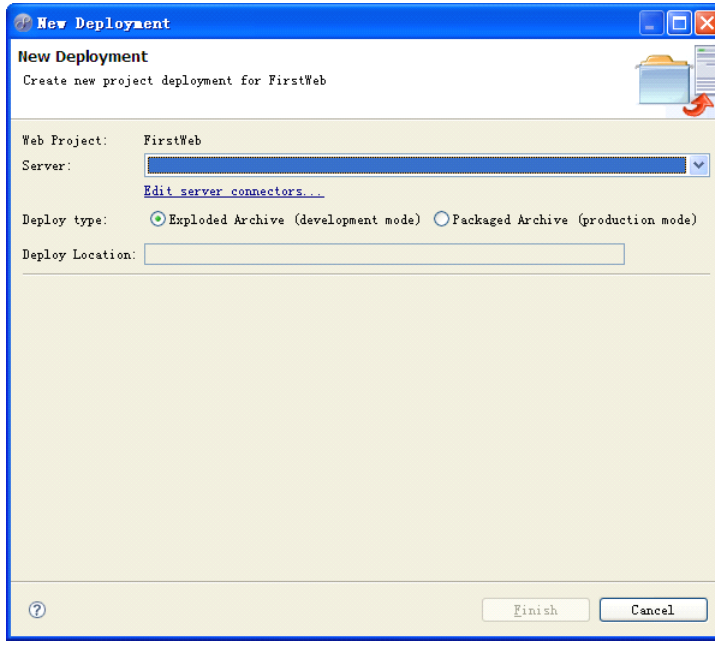


图 5-10　选择服务器

其中“Web Project”表示发布的项目，它是固定的。“Server”选项表示选择发布到哪一个应用服务器中，这里选择“Tomcat 6.x”，也就是配置的外部 Tomcat 服务器。“Deploy type”表示发布的方式，“Exploded Archive (development mode)”选项表示散

包发布，“Packaged Archive (production mode)”选项表示打包发布，这里选择散包发布。“Deploy Location”选项表示发布的具体目录，选择服务器后，该选项会自动填充。

单击“Finish”按钮将完成应用服务器的选择，回到指定项目发布界面，单击“OK”按钮将完成项目的发布。

注意：将多个项目部署到一个服务器中是完全没有问题的，只需要进行多次发布操作。但是在开发阶段，笔者并不建议这样操作，多余的项目会加重服务器的负担，从而使服务器的速度变慢。

5.4.3 指定服务器发布

在本小节中继续学习另一种发布操作，就是指定服务器发布。在前面配置外部 Tomcat 服务器后，将在 Servers 界面中多一种“Tomcat 6.x”的服务。选中该服务，单击鼠标右键，在弹出的菜单中，选择“Add Deployment”命令，将弹出指定服务器发布界面，如图 5-11 所示。

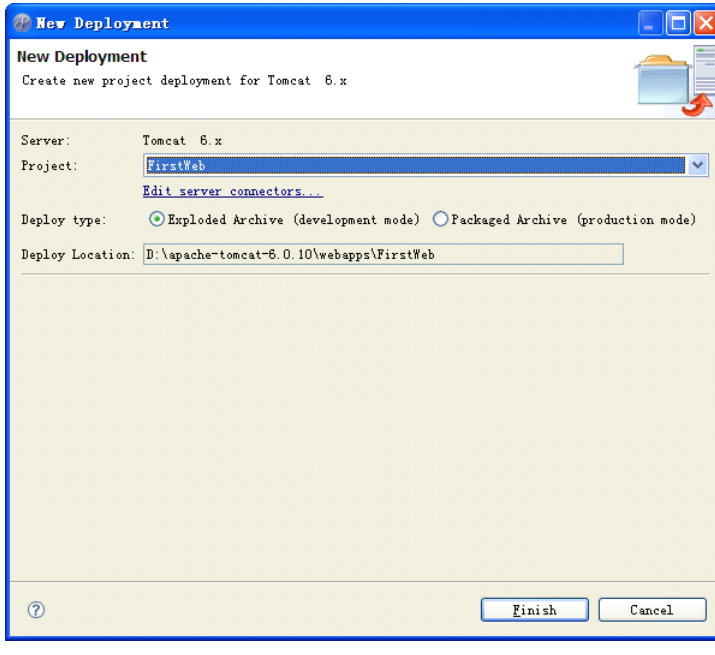


图 5-11 指定服务器发布

其中“Server”选项表示发布到的服务器，采用指定服务器的操作发布，该选项是固定的，直接显示前面操作的服务器。“Project”选项表示要发布的项目，会将当前工作空间的项目都列在其中，这里选择“FirstWeb”项目。“Deploy type”选项表示发布方式，“Deploy Location”选项表示发布的目录，这和前面的讲解时一样的。单击“Finish”按钮将完成指定服务器发布操作。

说明：从操作步骤来看，指定服务器发布要比指定项目发布的操作更简单。读者可以选择自己习惯的发布操作，在后面的学习和操作中，笔者主要采用指定服务器发布的操作。

5.4.4 启动服务器和运行项目

不管使用哪种方式和操作，发布项目成功后，就可以运行项目。在运行项目之前要首先启动服务器。在 Servers 界面中，选中要启动的服务器，因为我们的项目是发布到“Tomcat 6.x”服务器下，所以这里选中“Tomcat 6.x”服务器。单击鼠标右键，在弹出的菜单中，选择“Run Server”命令，将开始启动服务器。

这时候在控制台中将显示启动服务器的信息，当其中显示的信息出现如下信息时，表示启动完成。

信息: Server startup in 2312 ms

其中的数字就表示启动所使用的时间。

启动服务器后，打开任意一款浏览器，这里就以最常用的 IE 浏览器演示。打开 IE 浏览器，输入如下地址：

http://localhost:8080/

其中“localhost”表示本地服务器，这里也可以使用本地的 IP 地址代替，“8080”表示访问的端口号。按下回车键，或者单击浏览器中的“转到”按钮，显示结果如图 5-12 所示。

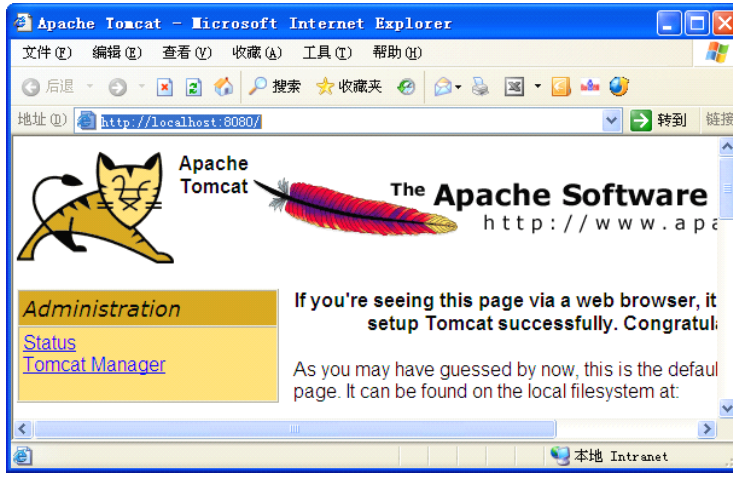


图 5-12　启动服务器页面

出现该界面后，表示在 MyEclipse 中启动外部的 Tomcat 服务器成功。如果出现的是不可访问页面，要查看在 MyEclipse 中是否启动了服务器，或者检查 Tomcat 是否安装正确。验证 Tomcat 服务器启动成功后，再次输入如下地址：

http://localhost:8080/FirstWeb/

其中“FirstWeb”就是访问项目的 URL，这是和创建项目时的“Context root URL”选项对应的。按下回车键，或者单击浏览器中的“转到”按钮，显示结果如图 5-13 所示。

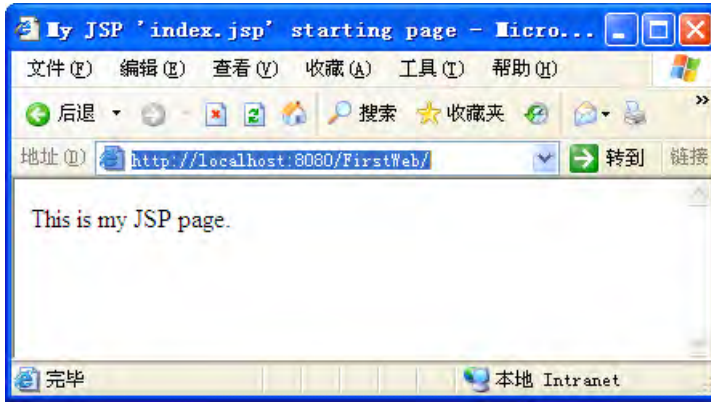


图 5-13　项目运行结果

如果读者也出现该结果，表示项目发布成功。因为项目的默认首页是“index.jsp”程序，其中显示“This is my JSP page.”信息就是“index.jsp”程序的功能。这里就不多对该程序进行讲解。

对于 Servers 界面中的服务器而言，除了具有启动命令外，还有“Restart”命令和“Stop”命令。“Restart”命令表示重新启动服务器，“Stop”命令表示停止服务器，它们在开发中也经常用使用到，操作是比较简单的，这里就不再详细讲解。

5.5　开发 HTML 页面程序

在 Java Web 项目中，由多种实现不同功能的程序组成，例如 HTML 程序、JSP 程序、Servlet 程序、过滤器等。在本节中就来学习一下如何使用 MyEclipse 工具开发这些程序。

HTML 页面程序是最直接的显示页面内容程序，学习过 Web 开发的读者对它肯定是不陌生的。在本小节中主要来看在 MyEclipse 中是如何开发 HTML 页面程序的。

说明：实际项目的最后的页面效果 HTML 程序通常是不用 MyEclipse 创建开发的，因为 MyEclipse 做出来的效果远没有 DreamWeaver 等专业开发 HTML 的工具做出来的好。但是 Java 程序员一定要做出来一个基本效果，然后才能提供给美工人员，让它们再加工。所以 Java 程序员希望在 MyEclipse 中也能开发 HTML 页面程序。

5.5.1　创建 HTML 页面程序

在 MyEclipse 中，创建 HTML 等 Web 程序的方式有很多种的，可以通过菜单或者工具栏完成，这里我们主要使用最常用的方式创建。在包资源管理器中，选中 Java Web 项目中的“WebRoot”目录，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择要创建程序界面，如图 5-14 所示。

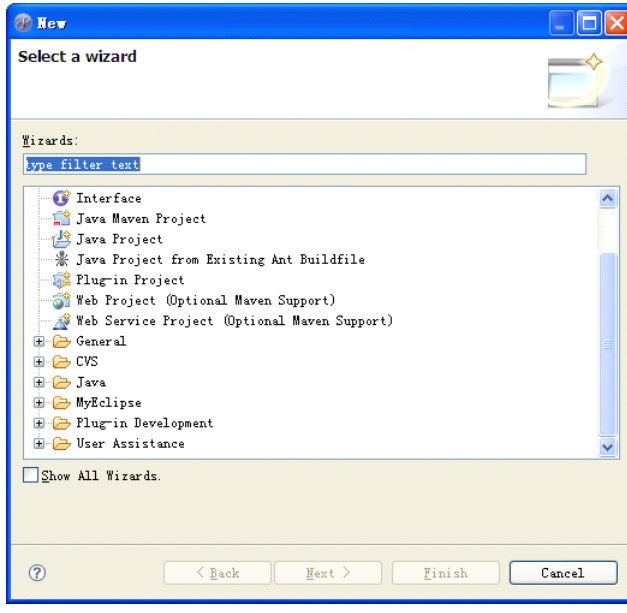


图 5-14　选择要创建的程序

在其中选择“MyEclipse”|“Web”节点，可以看到其中有两个创建 HTML 程序的节点，分别是“HTML（Advanced Templates）”选项和“HTML（Basic Templates）”选项，它们分别对应着高级模板和基础模板。

5.5.2　使用基础模板创建

在选择创建程序界面中，选择“HTML（Basic Templates）”选项，单击“Next”按钮，将弹出创建基础模板的 HTML 程序界面，如图 5-15 所示。

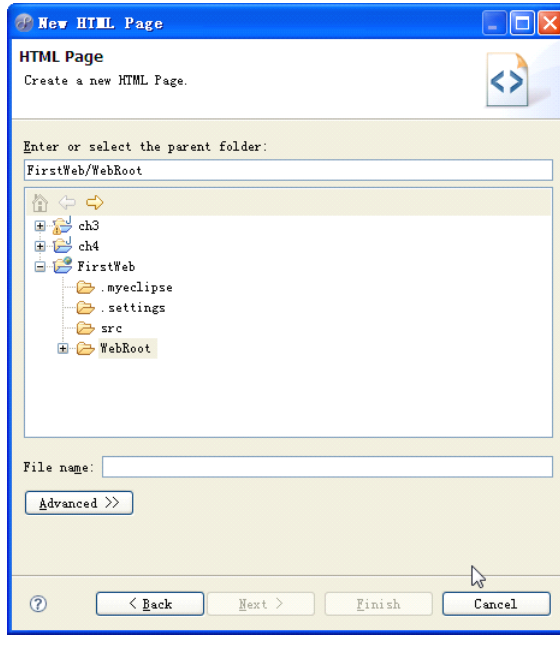


图 5-15　创建基础模板 HTML 程序

其中“Enter or select the parent folder”选项表示输入或者选择文件创建的位置，这里我们就让它创建在“WebRoot”目录下，也就是 Web 程序的根目录。“File name”选项表示新创建程序的文件名，这里填写“HTMLOfBasic.html”，HTML 程序的扩展名可以是“.html”，也可以是“.htm”。单击“Next”按钮可以看到创建 HTML 程序的预览，单击“Finish”按钮将完成基础模板的 HTML 程序的创建。

在包资源管理器的“FirstWeb”项目的“WebRoot”目录下将创建文件名为“HTMLOfBasic.html”的 HTML 程序，并且在编辑区中会将该程序显示出来，如图 5-16 所示。

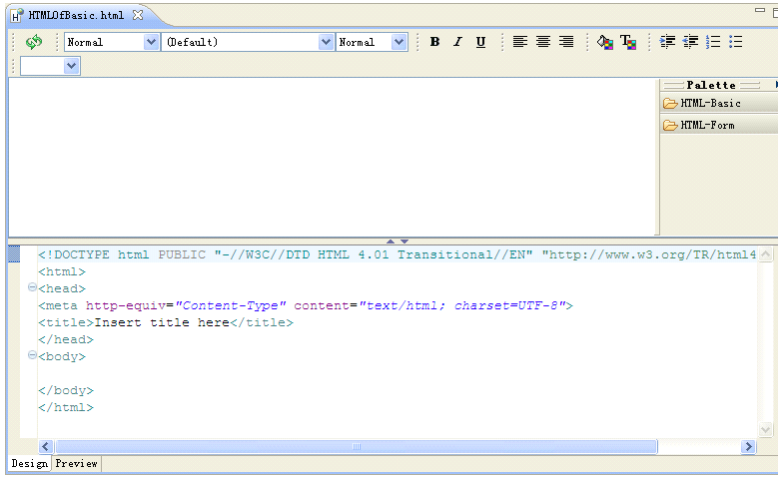
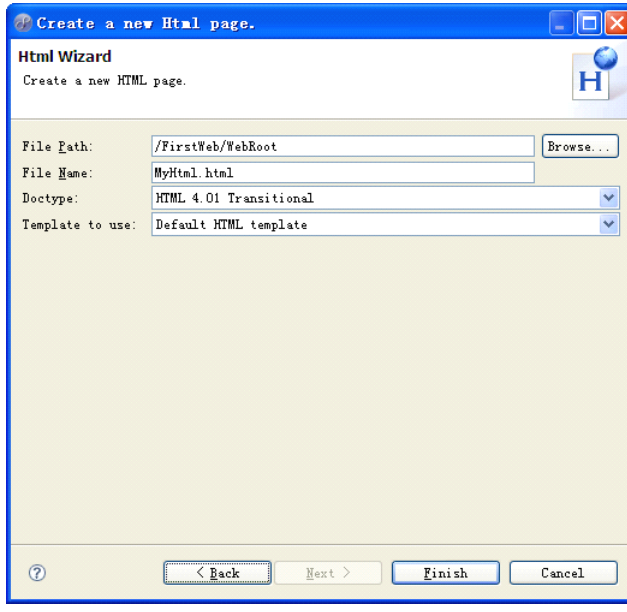


图 5-16　基础模板 HTML 程序代码

其中编辑区的下半部分可以看到基础模板 HTML 的程序代码，在其中给出了 HTML 程序最基本的头标记、体标记等基础标记，并且使用“<meta>”标记给出当前 HTML 程序的编码规则是“UTF-8”。该界面具体如何使用在后面的小节中进行详细讲解。

5.5.3　使用高级模板创建

使用高级模板创建 HTML 程序的创建操作有两种，第一种就是在图 5-14 的选择创建程序界面中，选择“HTML（Advanced Templates）”选项。第二种就是选中“WebRoot”目录，单击鼠标右键，在弹出的菜单中，选择“New”|“HTML（Advanced Templates）”命令。使用这两种操作都将弹出创建使用高级模板 HTML 程序的界面，如图 5-17 所示。



5-17　创建高级模板 HTML 程序

其中“File Path”表示新创建的 HTML 程序所在目录，如果使用前面的创建方式，该目录将会自动填写。“File Name”表示创建 HTML 程序的文件名，这里填写“HTMLOfAdvanced.html”。“Doctype”表示新创建 HTML 程序的文档类型，这里使用默认的选择就可以。“Template to use”表示模板的选择，其中有两个选项，默认的“Default HTML tempate”选项表示使用默认 HTML 模板，“HTML tempate with a form”选项表示创建一个拥有提交表单的 HTML 程序。

先选择“Default HTML tempate”选项，单击“Finish”按钮，将完成高级模板 HTML 程序的创建。在编辑区中同样会出现上下两部分，其中上半部分是和基础模板相同的。在下半部分中将显示自动生成的代码，内容为：

```
01  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
02  <html>
03    <head>
04      <title>HTMLOfAdvanced.html</title>
05      <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
06      <meta http-equiv="description" content="this is my page">
07      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
08      <!--<link rel="stylesheet" type="text/css" href="./styles.css">-->
09    </head>
10    <body>
11      This is my HTML page. <br>
12    </body>
13  </html>
```

其中从第 5 行到第 7 行使用<meta>标记给出 HTML 程序的相关信息，分别是页面关键字、页面描述和程序的编码。第 8 行是一行注释代码，其中给出了 CSS 程序，将页面需要 CSS 程序时，可以去掉该行代码注释。

如果在创建高级模板程序界面中，选择“HTML tempate with a form”选项，则创建的 HTML 程序代码为：

```
01  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
02  <html>
03    <head>
04      <title>HTMLOfAdvanced2.html</title>
05      <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
06      <meta http-equiv="description" content="this is my page">
07      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
08      <!--<link rel="stylesheet" type="text/css" href="./styles.css">-->
09    </head>
10    <body>
11      <form name="f1" id="f1" action="" method="post">
12        <table border="0">
13          <tr>
14            <td>Login:</td>
15            <td><input type="text" name="login" id="login"></td>
16          </tr>
17          <tr>
18            <td>Password:</td>
19            <td><input type="password" name="password" id="password"></td>
20          </tr>
21          <tr>
22            <td colspan="2" align="center"><input type="submit"></td>
23          </tr>
24        </table>
25      </form>
26    </body>
27  </html>
```

其中从第 11 行到第 25 行创建了一个表单，其中 action 属性指定的提交目标需要手动填写。第 15 行创建了用于填写用户名的文本框，第 19 行创建了用于填写密码的密码框，第 22 行创建了提交按钮。当页面中需要提交表单时，可以使用该模板创建 HTML 程序。

说明：在实际开发中，基础模板和高级模板的选择并没有很大区别的，因为高级模板中自动生成的内容很少被使用到。只所以使用高级模板比较多，是因为在“New”选项下直接就有创建高级模板 HTML 程序的命令。后面学习的 JSP 程序也是这样的。

5.5.4 编写 HTML 程序

当使用编辑区对 HTML 程序编写时，通常将编辑区分为上下两部分，其中下半部分是 HTML 程序代码，在其中可以手动编写代码；在上半部分中，可以通过界面的方式创建 HTML 程序，它是本书讲解的重点。在上下两部分中间，有两个三角，单击它们可以使整个编辑区只显示上半部分或者下半部分。在编辑区的左下角，有两个选项，

“Design”选项表示设计 HTML 程序；“Preview”选项表示预览设计的效果，其中给出了 IE 浏览器和 Mozilla 浏览器的预览。

在以界面方式创建 HTML 程序的部分中，最上面的工具栏是对 HTML 中的文件进行操作的。这里我们没有必要对它们依次讲解，它们都是非常简单的。例如我们在界面中输入“大家好，这是我的第一个 HTML 程序”，然后单击  加粗按钮，继续单击  居中按钮。这时界面中的内容也会发生改变，并且下半部分的程序也会同时改变，生成后的代码为：

```
01  <div align="center">
02  <strong>大家好，这是我的第一个 HTML 程序
03  </strong></div>
```

其中第 1 行指定的就是文字居中，标记指定的就是加粗。单击左下角的“Preview”选项，可以查看到 HTML 程序预览。HTML 程序是并不需要发布项目就可以查看的，所以直接使用外部的浏览器也可以查看到它的效果，如图 5-17 所示。

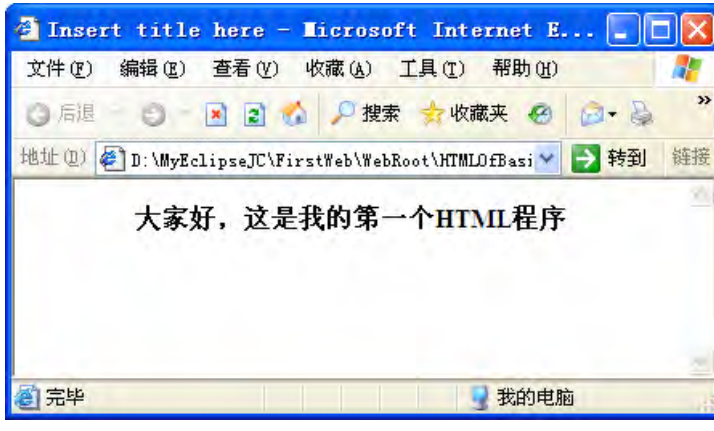


图 5-17　HTML 程序运行效果

在以界面方式创建 HTML 程序的部分中，右边是用于添加 HTML 元素的选项，如图 5-18 所示。

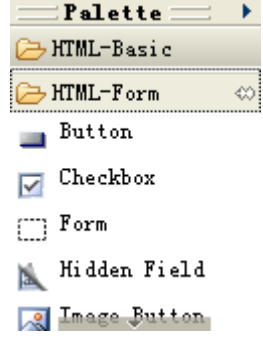


图 5-18　创建 HTML 元素

其中“HTML-Basic”栏中包含用于 HTML 基本元素的创建，例如超链接、图片、

表格等。在“HTML-Form”栏中包含用于表单创建的元素，例如表单、文本框、按钮等。这里我们以表单元素举例，读者可以自己了解一下其他元素的创建，都是比较简单的。

单击“HTML-Form”栏中的“Form”选项，将弹出创建表单的界面，如图 5-19 所示。

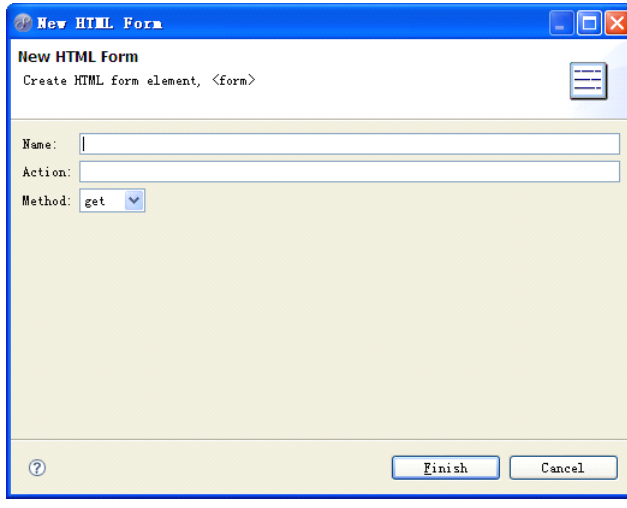


图 5-19　创建表单

其中“Name”表示创建表单的名称，这里填写“myForm”。“Action”表示创建表单的提交目标，这里可以填写“HTMLOfAdvanced.html”，也就是高级模板 HTML 程序。“Method”表示创建表单的提交方法，可以在 get 和 post 两种之间选择，对于本程序而言，它们是没有区别的。单击“Finish”按钮将完成表单的创建，在 HTML 程序将多出如下代码：

```
<form method="get" action="HTMLOfAdvanced.html" name="myForm">…</form>
```

在界面中，选择表单元素添加的位置，或者在程序代码中将焦点定义在<form>标记中。继续单击“HTML-Form”栏中的“Text Field”选项，将弹出创建文本框的界面，如图 5-20 所示。

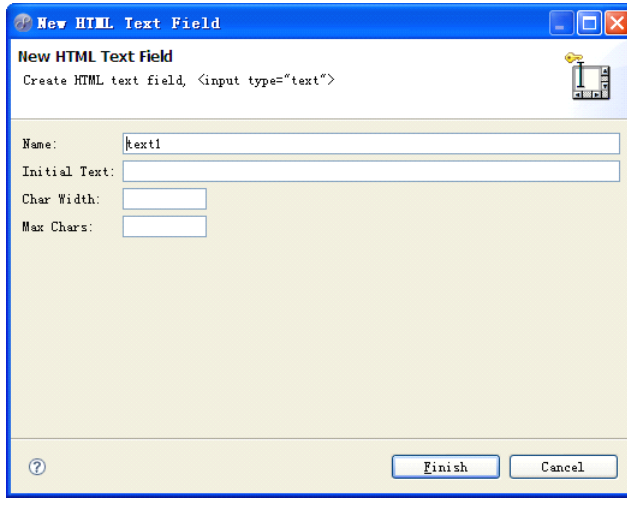


图 5-20　创建文本框

其中“Name”表示创建文本框的名称，这里填写“myText”。“Initial Text”表示文本框的初始值，填写“请在这里输入内容”。“Char Width”表示文本框的长度，“Max Chars”表示文本框中能够输入的最大字符数，当不填写这两项时会给出一个默认值。单击“Finish”按钮，将完成文本框的创建，程序代码中将多出如下代码：

```
<input type="text" value="请在这里输入内容" name="myText">
```

使用外部浏览器，打开目前修改后的程序，运行结果如图 5-21 所示。

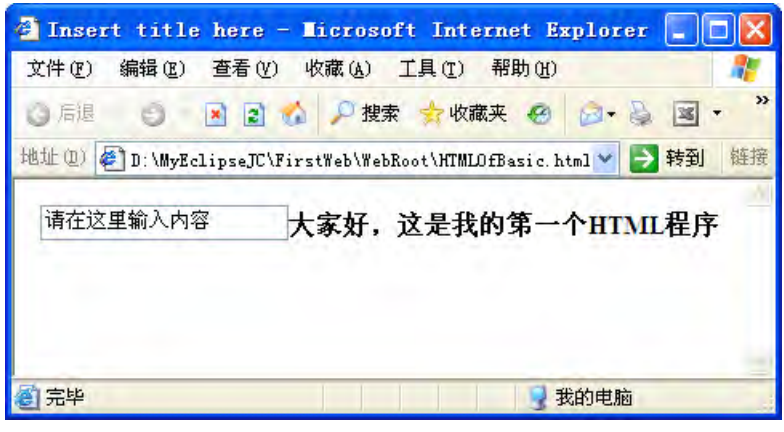


图 5-21　加入表单元素

技巧：使用界面创建的 HTML 程序代码的编写经常是不规范的。在编辑区中可以全部选中这些代码，单击鼠标右键，在弹出的菜单中选择“Source”|“Format”命令，将把全部代码调整为规范格式。

5.6　开发 JSP 程序

在本节中继续来学习使用 MyEclipse 开发 JSP 程序。JSP 程序是在 HTML 程序的基础上加入 Java 程序组成的，所以在创建 JSP 程序时，是和创建 HTML 程序非常类似的，也分为基础模板和高级模板。

5.6.1　使用基础模板创建 JSP 程序

在包资源管理其中，选中要创建 JSP 程序的目录，例如选中“WebRoot”目录。单击鼠标右键，在弹出的菜单中，选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web”节点，如图 5-22 所示。

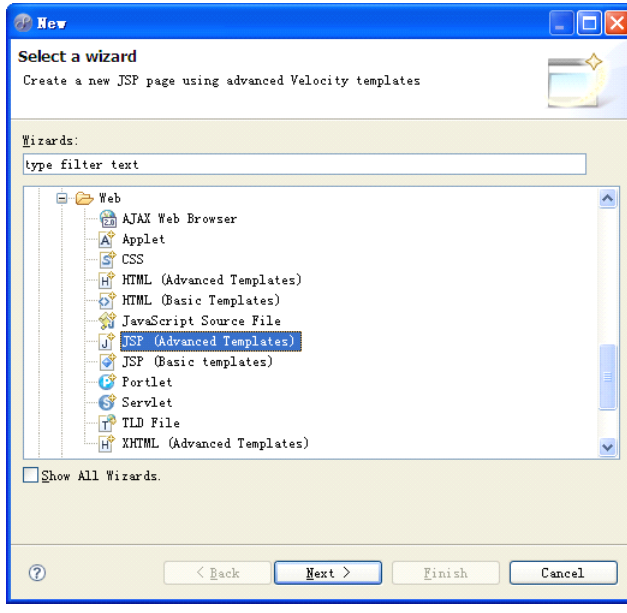


图 5-22　选择 JSP 程序

从界面中可以看到，创建 JSP 程序中，也存在高级模板和基础模板。在其中选择“JSP（Basic Templates）”选项，单击“Next”按钮，将进入创建基础模板的 JSP 程序界面，如图 5-23 所示。

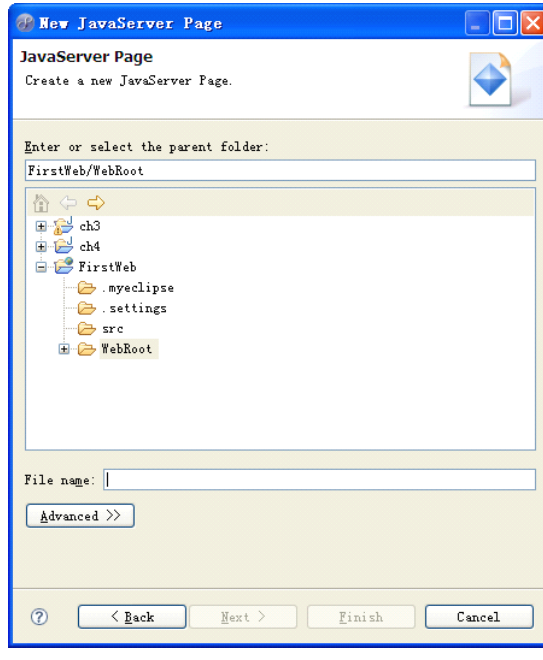


图 5-23　创建基础模板 JSP 程序

可以看到是和创建 HTML 程序非常相似的。“Enter or select the parent folder”选项表示 JSP 程序创建的位置。“File name”选项表示创建的 JSP 程序的文件名，这里填写“JSPOfBasic.jsp”。单击“Next”按钮，将看到 JSP 程序的预览。单击“Finish”按钮

后将完成基础模板 JSP 程序的创建，并在编辑区中打开 JSP 程序，如图 5-24 所示。

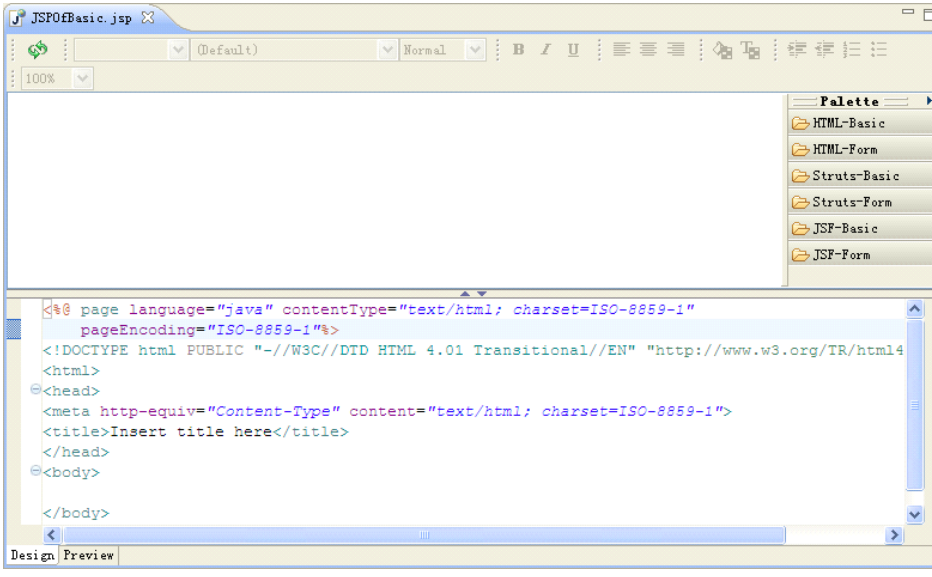


图 5-24　编辑区打开 JSP 程序

可以看到在编辑区中打开 JSP 程序，也是分为上下两部分的。在上半部分中的右边，可以看到不但可以添加 HTML 标记元素，还可以添加 Struts 和 JSF 的标记元素，这些内容在后面专门讲解对应技术时进行详细讲解。

在下半部分中可以看到具体的 JSP 程序代码，和 HTML 程序最大的区别就是在程序的最开头的部分多出了两行代码，代码内容为：

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

这两行代码是使用 JSP 技术中的 page 指令指定 JSP 程序的编码规则。

5.6.2　使用高级模板创建 JSP 程序

在选择 JSP 程序界面中，选择“JSP（Advanced Templates）”选项，或者选中要创建目录，单击鼠标右键，在弹出的菜单中选择“New”|“JSP（Advanced Templates）”命令，都将弹出使用高级模板创建 JSP 程序的界面，如图 5-25 所示。

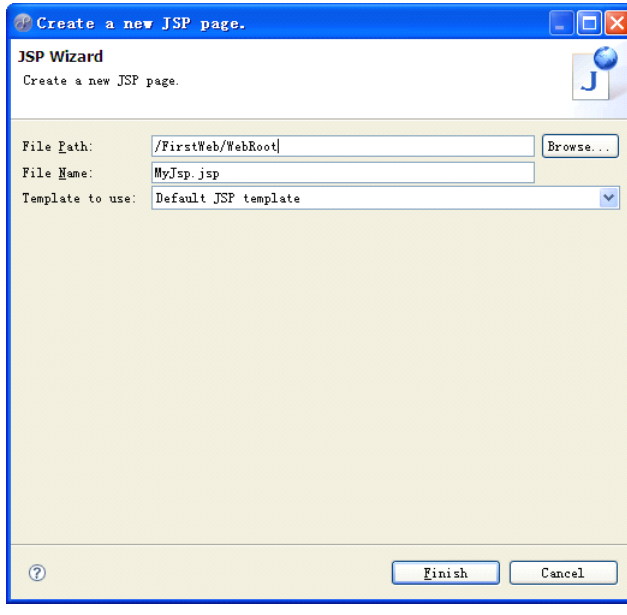


图 5-25　高级模板创建 JSP 程序

其中“File Path”表示新创建 JSP 程序所在的目录。“File Name”表示创建 JSP 程序的文件名，这里填写“JSPOfAdvanced.jsp”。“Template to use”表示使用的模板，通常选择“Default JSP template”选项，也就是默认 JSP 模板。

说明：在“Template to use”中有很多非常好用的模板，例如 Struts 模板、JSF 模板等，它们将在后面专门讲解对应技术时进行讲解。

单击“Finish”按钮，将完成高级模板 JSP 程序的创建，这时候会在编辑区中打开 JSP 程序代码，其具体代码为：

```
01  <%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
02  <%
03  String path = request.getContextPath();
04  String basePath = request.getScheme()+"://"+request.getServerName()+
05                                          ":"+request.getServerPort()+path+"/";
06  %>
07  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
08  <html>
09    <head>
10      <base href="<%=basePath%>">
11      <title>My JSP 'JSPOfAdvanced.jsp' starting page</title>
12       <meta http-equiv="pragma" content="no-cache">
13       <meta http-equiv="cache-control" content="no-cache">
14       <meta http-equiv="expires" content="0">
15       <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
16       <meta http-equiv="description" content="This is my page">
17       <!--
18       <link rel="stylesheet" type="text/css" href="styles.css">
```

```
19          -->
20      </head>
21      <body>
22          This is my JSP page. <br>
23      </body>
24  </html>
```

在其中第 1 行中使用 page 指令指定语言是 Java，导入 util 包和设置 JSP 程序的编码为“ISO-8859-1”。在第 3 行中使用 request 内置对象调用 getContextPath 方法，从而获得当前应用的根目录，它在实际开发中经常被用到，通过相对路径的方式，从而避免路径错误。第 4 行中使用到了三个 request 内置对象中的方法，分别是获取网络协议、主机名和端口号，然后加入当前应用的根目录，从而得到完整访问路径。后面其他的内容就和 HTML 程序中的非常相似，这里就不再进行讲解。

5.6.3 改变 JSP 程序的编码

在前面讲解 HTML 程序时，我们通过“大家好，这是我的第一个 HTML 程序”这句话讲解了 HTML 程序的界面化编写，但是如果将这句话复制到当前 JSP 程序中，会发现是不能够保存的。这是因为当前 JSP 程序的编码规则是“ISO-8859-1”，在该编码中是不支持中文的。如果想在 JSP 页面中显示中文，需要改变 JSP 程序的编码。

在 MyEclipse 的菜单中，选择“Window”|“Preferences”命令，将打开首选项设置界面，在其中左边选择“MyEclipse Enterprise Workbench”|“Files and Editors”|“JSP”节点，界面如图 5-26 所示。

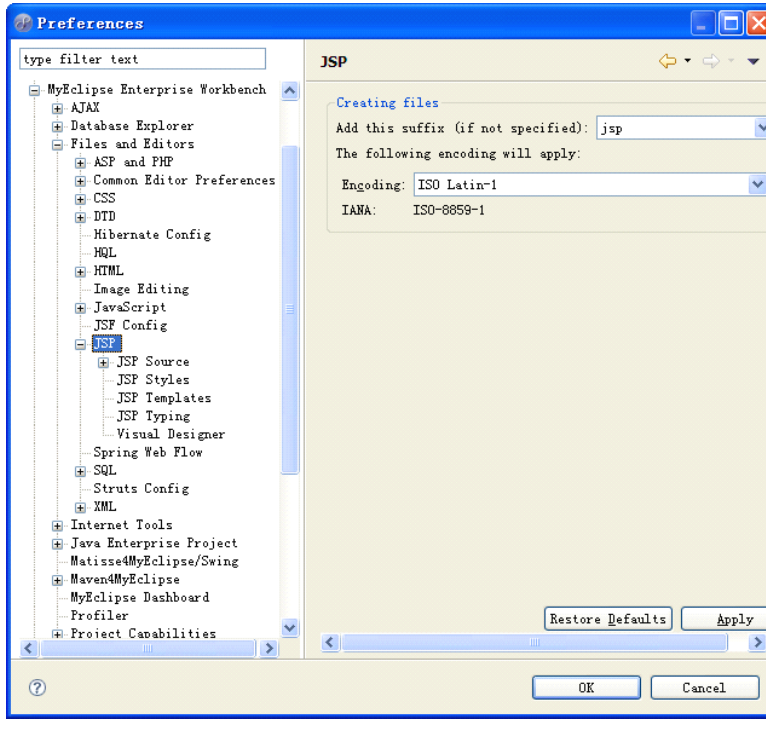


图 5-26 改变 JSP 程序的编码

其中“Encoding”就是用于编码的选择，默认的就是“ISO-8859-1”。在其中使用最多的就是选择“ISO 10646/Unicode(UTF-8)”或者“Chinese,National Standard”选项，它们分别对应着“UTF-8”和“GB18030”编码。“UTF-8”编码是 Unicode 的一种变长字符编码，从而它能够对中文进行编码。“GB18030”编码是我国定义的汉字编码规则，可以说它专业的中文编码，它和 GBK、gb2312 等中文编码类似的。这里我们选择“ISO 10646/Unicode(UTF-8)”选项，也就是定义成“UTF-8”编码。

单击“OK”按钮，再次创建高级模板的 JSP 程序，其代码中的 page 指令内容将变为：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
```

其中使用 pageEncoding 属性指定编码规则为“UTF-8”。这时候再在其中输入中文，会发现是能够正常保存的，不会出现错误的。

注意：编码是 Web 开发中非常重要的一个知识点，一定要对编码有所了解。如果处理不好，在页面中就会出现乱码。在后面的其他技术的讲解中，还会讲解到编码。

5.6.4 在 JSP 程序中使用 EL 表达式

EL 全名为 Expression Language，它原来是 JSTL1.0 为了方便存取数据所定义的语言。到了 JSP2.0 以后，EL 正式纳入成为标准规划之一。

说明：是否能够在 JSP 程序中使用 EL 表达式，这要取决于当前开发环境。要想在其中能够直接使用 EL 表达式，要在创建 Java Web 项目时选择 J2EE 5.0 版本。

EL 表达式使用最多的就是获取保存在某一范围内的属性值。在 EL 表达式中有四个与存储范围有关的内置对象，分别是 applicationScope、sessionScope、requestScope、pageScope。这些内置对象使用时和 JSP 中的 pageContext，request，session，application 对象是一样的。但是这四个内置对象只能用来取得某个范围的属性值，即只有 getAttribute()方法，不能取得其他的信息。

在 EL 表达式中，也可以不指定范围，如直接使用${name}。这时 EL 将默认从 page 范围开始查找 name 属性，如果找不到，再依次到 request、session、application 范围中找。如果仍然没有找到就返回 null，在网页中也就没有显示。

在 MyEclipse 的包资源管理器中，选中 FirstWeb 项目中的“WebRoot”目录，单击鼠标右键，在弹出的菜单中选择“New”|“JSP（Advanced Templates）”命令，在创建 JSP 程序界面中，填写文件名为“JSPOfEL.jsp”。创建完成后将在编辑区中打开该代码，在 body 标记中添加如下内容：

```
01      <%
02       request.setAttribute("name","Tom");
03       %>
04       欢迎${name}登录本 网站！
```

其中第 2 行使用 request 内置对象保存一个 name 信息，在第 4 行中使用 EL 表达式获取 name 信息，从而将这些信息显示到页面中。

注意：运行 JSP 程序是要首先发布项目，然后启动服务器的。

启动服务器后，打开浏览器，输入如下地址：

http://localhost:8080/FirstWeb/JSPOfEL.jsp

浏览器如图 5-27 所示。

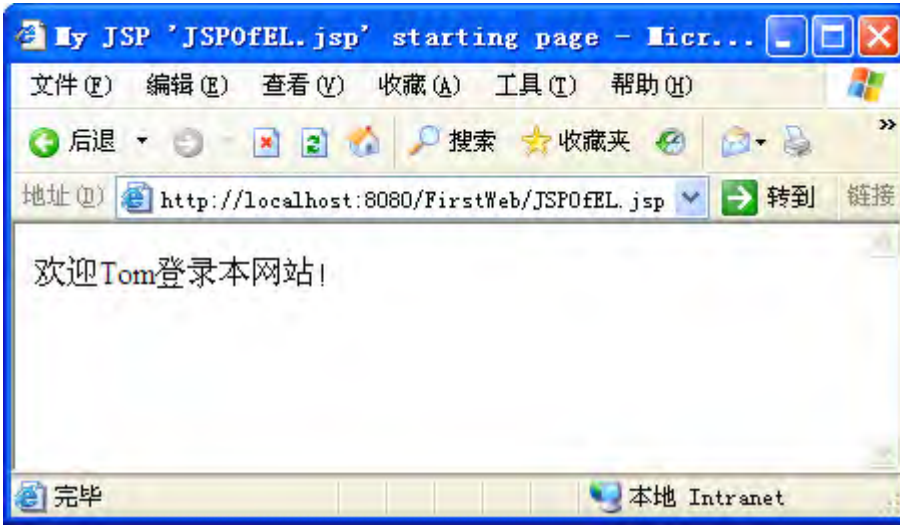


图 5-27　使用 EL 表达式

从运行结果中可以看到，将保存在 request 内置对象中的属性值获取到。如果不运行项目，而是直接使用浏览器打开 JSP 程序，将会直接将 EL 表达式显示在页面中，读者可以自己尝试一下。

5.6.5　在 JSP 程序中使用 JSTL 标签

JSTL 的全称为 JSP Standard Tag Library，其中文名为 JSP 标准标记库。是一个实现了 Web 应用程序中常见的通用功能的定制标记库集，这些功能包括迭代和条件判断、数据管理格式化、XML 操作以及数据库访问。

如果想在 JSP 程序中使用 JSTL 标签，要首先将 JSTL 标签的相关 JAR 包导入到项目中。如果创建 Java Web 项目时，选择的是 J2EE 5.0 版本，则会自动将 JSTL 标签 JAR 包导入到项目中，在“FirstWeb”项目的“Java EE 5 Libraries”节点下可以看到有一个“jstl-1.1.jar”文件，它就是 JSTL 标签相关 JAR 包。

如果创建 Java Web 项目时，选择的是 J2EE 1.4 版本，那么就需要手动的导入 JSTL 标签 JAR 包。JAR 包可以去官方网站进行下载，然后将下载后的 JAR 包复制到“WebRoot”|“WEB-INF”|“lib”目录下。

JSTL 标签库分为五类，分别是 JSTL 核心标签库、数据库标签库、I18N 格式化标签库、XML 标签库和 JSTL 函数标签库。这里我们以其中的核心标签库中的流程控制相关标签进行讲解。

在 JSP 页面中要使用到核心标签库，必须使用<%@taglib%>指令，指定核心标签库的前缀以及 uri，代码如下。

```
<%@ taglib    prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

其中 prefix 为“c”，表示标签的前缀为“c”，这时就必须使用<c:XXX>这种格式。

核心标签通常设置前缀为“c”，也可以直接为其他内容，但是阅读性就会降低。uri 用来指定核心标签库规范文件（TLD 文件）地址。

技巧：uri 属性值是比较长的，有些程序员选择死记硬背，这是没有必要的。打开“jstl-1.1.jar”|“WEB-INF”目录下的“c.tld”文件，其中“<uri>http://java.sun.com/jsp/jstl/core</uri>”指定的就是 url，将其中的“http://java.sun.com/jsp/jstl/core”复制到<%@taglib%>指令的 uri 属性就可以。其他的标签库也是这样的。

使用前面讲解的创建 JSP 程序的方式，创建一个文件名为“JSPOfJSTL.jsp”的 JSP 程序。在编辑区中，在 page 指令下加入<%@taglib%>指令，引入核心标签库。然后在 body 标记中加入如下代码：

```
01  <%
02      ArrayList list = new ArrayList();  //创建列表用来存放旅游地信息
03      list.add("故宫");                  //添加故宫
04      list.add("长城");                  //添加长城
05      list.add("香山");                  //添加香山
06      list.add("北海");                  //添加北海
07      pageContext.setAttribute("list",list);//将 list 信息保存在 page 范围
08  %>
09  <table border="1">
10      <tr>
11          <td>序号</td>
12          <td>名称</td>
13      </tr>
14      <c:forEach var="lvyou" items="${pageScope.list}" varStatus="stus">
15          <tr>
16              <td><c:out value="${stus.index + 1}"></c:out></td><%-列表序号--%>
17              <c:choose>
18                  <c:when test="${(stus.index + 1) % 2 == 0}"><%--偶数 --%>
19                  <td bgcolor="#CCCCCC"><c:out value="${lvyou}"/></td>
20                  </c:when>
21                  <c:otherwise>
22                      <td><c:out value="${lvyou}"/></td>
23                  </c:otherwise>
24              </c:choose>
25          </tr>
26      </c:forEach>
27  </table>
```

其中从第 2 行到第 7 行表示创建一个列表集合，然后将集合信息保存到 page 范围内。在第 14 行中使用<c:forEach>标签对 page 范围内的列表进行遍历。在第 16 行中使用<c:out>标签进行内容的输入，这里结合了 EL 表达式。第 17 行的<c:choose>标签是判断标签的父标签，在该标签中，第 18 行使用<c:when>标签判断序号是否为偶数，它相当于 if 语句；第 21 行中使用<c:otherwise>标签，它相当于 else 语句。

5.7 开发 JavaBean 程序

开发 JavaBean 程序并不能算是一个新知识点，因为 JavaBean 其实就是一个简单的 Java 类。在 Java Web 项目中通常将 JavaBean 程序放在“src”目录下。在 Web 的程序中，通常所说的 JavaBean 是指实体类程序，也可以把它称为标准的 JavaBean 程序。标准的 JavaBean 应该具有如下几个特点：

1、JavaBean 必须是一个公开的类，也就是说 JavaBean 的类访问权限必须是 public 的。

2、JavaBean 必须具有一个无参数的构造方法。如果在 JavaBean 中定义了自定义的有参构造方法，就必须添加一个无参数构造方法，否则将无法设置属性；如果没有定义自定义的有参构造方法，则可以利用系统自动生成的无参构造方法。

3、JavaBean 一般将属性设置成私有的，通过使用 setXXX()方法和 getXXX()方法来进行属性的设置和获取。

其中构造函数可以通过生成构造函数的操作来自动生成。setXXX()和 getXXX()方法可以通过生成 Setter 和 Getter 方法的操作自动生成。这些都在第 2 章中详细讲解过，这里就不再详细讲解。

5.8 开发 Servlet 程序

在当前 Java Web 项目的分层思想中，前面两节讲解的 HTML 程序和 JSP 程序主要是做显示层，而本节将要讲解的 Servlet 主要就是做控制层。在 Servlet 中完成参数的接收，然后使用参数调用业务逻辑层，接下来根据调用结果的不同跳转到不同的页面。

5.8.1 创建 Servlet 程序

在 MyEclipse 中集成了创建 Servlet 程序的功能，会自动生成 Servlet 程序中的固定代码，并给出配置代码。Servlet 程序将创建在 src 下，并且需要为它定义单独的一个包，例如包名为“com.sanqing.servlet”。然后选中包目录，单击鼠标右键，在弹出的菜单中，选择“New”|“Servlet”命令，将弹出创建 Servlet 程序的界面，如图 5-28 所示。

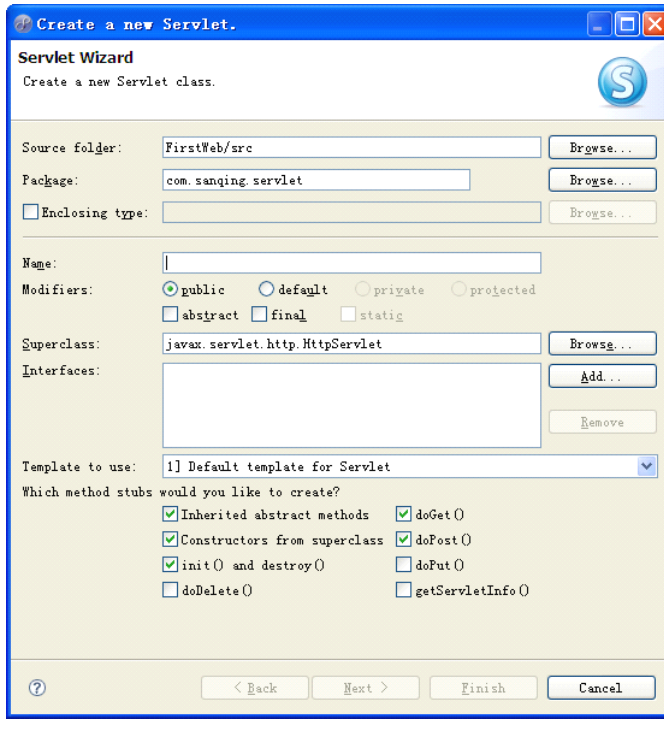


图 5-28　创建 Servlet 程序

其中“Source folder”表示创建 Servlet 程序的目录位置。“Package”表示创建 Servlet 的所在包。“Name”表示 Servlet 的名称，它的名称通常以“Servlet”结尾，例如“HelloServlet”。“Modifiers”表示 Servlet 程序的修饰符，通常使用 public 公共访问修饰符，它也是默认选项。“Superclass”表示 Servlet 程序继承的父类，当使用这种方式创建 Servlet 时将自动继承“javax.servlet.http.HttpServlet”类。“Interfaces”表示 Servlet 程序实现的接口，通常是不需要实现接口的。“Template to use”表示 Servlet 程序使用的模板，目前只有默认模板一个选项。

“Which method stubs would you like to create”表示在 Servlet 程序中将要创建哪些方法，其中最重要的就是生命周期方法。“init() and destroy()”选项表示初始化和销毁方法，当初始化和销毁 Servlet 时执行对应的方法。“doGet()”选项和“doPost()”选项都表示 Servlet 的执行方法，分别对应 GET 和 POST 提交方式。这里我们先只选择其中的“doGet()”选项和“doPost()”，单击“Next”按钮，将进入 Servlet 配置页面，如图 5-29 所示。

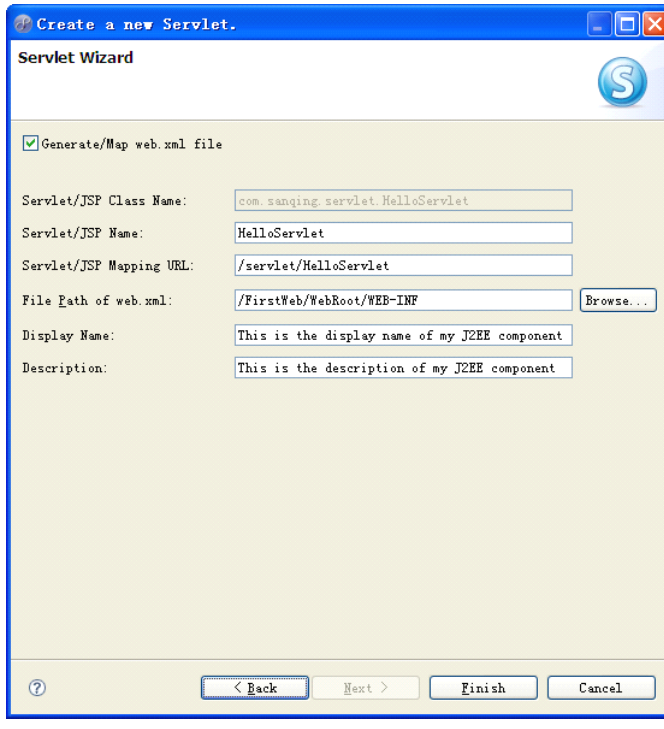


图 5-29　对 Servlet 进行配置

其中“Servlet/JSP Class Name”表示 Servlet 程序类的类名，这里一定要是具有包名的完整类名。“Servlet/JSP Name”表示 Servlet 程序的名称，理论上这里可以任意取，但是通常以 Servlet 的类名来命名。“Servlet/JSP Mapping URL”表示访问 Servlet 的 URL，该 URL 出现在提交表单或者浏览器访问地址中。“File Path of web.xml”表示在其中进行 Servlet 配置的“web.xml”文件所在目录。“Display Name”和“Description”都表示描述性的语言，它们是可以不写的。

单击“Finish”按钮，将完成 Servlet 程序的创建。在编辑区中将打开“HelloServlet.java”和“web.xml”两个文件，其中“HelloServlet.java”Servlet 程序部分代码如下：

```
01  package com.sanqing.servlet;
02  //省略导入接口和类的代码
03  public class HelloServlet extends HttpServlet {
04      public void doGet(HttpServletRequest request, HttpServletResponse response)
05              throws ServletException, IOException {
06          response.setContentType("text/html");
07          PrintWriter out = response.getWriter();
08          out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML
09                                                  4.01 Transitional//EN\">");
10          out.println("<HTML>");
11          out.println("    <HEAD><TITLE>A Servlet</TITLE></HEAD>");
12          out.println("    <BODY>");
13          out.print("       This is ");
14          out.print(this.getClass());
```

```
15                  out.println(", using the GET method");
16                  out.println("    </BODY>");
17                  out.println("</HTML>");
18                  out.flush();
19                  out.close();
20              }
21              //省略 doPost 方法代码
22      }
```

其中从第 8 行到第 17 行的代码都是输出 HTML 标记，以及其中的内容，从而组成一个页面。再来看在“web.xml”文件中对 Servlet 的配置文件，其配置代码为：

```
01      <servlet>
02          <servlet-name>HelloServlet</servlet-name>
03          <servlet-class>com.sanqing.servlet.HelloServlet</servlet-class>
04      </servlet>
05      <servlet-mapping>
06          <servlet-name>HelloServlet</servlet-name>
07          <url-pattern>/servlet/HelloServlet</url-pattern>
08      </servlet-mapping>
```

其中从第 1 行到第 4 行使用<servlet>标记对 Servlet 程序进行配置，第 3 行中使用<servlet-class>标记配置了 Servlet 的全称类名，并为它使用<servlet-name>标记定义了一个名称。

从第 5 行到第 8 行使用<servlet-mapping>标记对 Servlet 程序的映射进行配置。其中的 Servlet 名称一定要有前面配置类时的名称对应上，然后使用<url-pattern>标记配置访问 URL 路径。这些内容都是通过配置 Servlet 界面自动生成的。

启动或者重启服务器，启动后，打开浏览器，在其中输入如下地址：

http://localhost:8080/FirstWeb/servlet/HelloServlet

其中“/servlet/HelloServlet”就是在“web.xml”文件中对 Servlet 进行的 URL 配置，访问后的页面如图 5-30 所示。

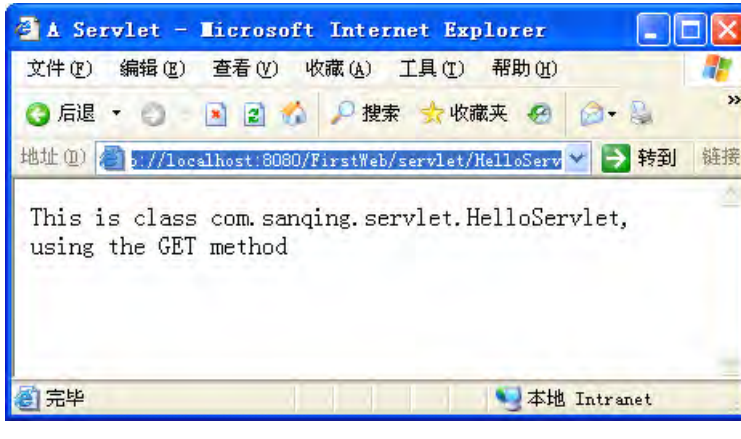


图 5-30　Servlet 程序的运行结果

其中的信息就是运行 Servlet 时输出的内容，“con.sanqing.servlet.HelloServlet”就

是“this.getClass()”的输出结果。

说明：在本小节中主要讲解的是 Servlet 的创建，在实际开发中 Servlet 的功能并不是输出页面，该功能通常由 JSP 程序来完成。Servlet 的主要功能就是作为控制层，接收参数、调用后台、执行跳转。

5.8.2 创建 Filter 过滤器程序

Filter 技术是 Servlet 2.3 新增加的功能，其中文名称为过滤器。Filter 和 Servlet 是非常类似的，也是一个 Java 类，所以其创建方式和普通的 Java 类完全相同。自定义的 Filter 类必须实现 Filter 接口，还必须实现 Filter 接口中定义的 init()方法、doFilter()方法和 destroy()方法。其中 init()方法和 destroy()方法分别用来初始化和销毁时调用，通常并不在这两个方法中写任何代码。

doFilter()方法用来实现过滤，所有的处理代码都放置在该方法中。doFilter()方法接收三个参数，分别是 request、response 和 chain。其中 request 和 response 参数用来传递给下一个 Filter（如果有多个 Filter）或者 JSP 和 Servlet。chain 参数则通过调用其 doFilter 方法来调用下一个 Filter，获得调用原始的 JSP 或者 Servlet 等其他内容。

在 MyEclipse 中，并没有集成创建 Filter 过滤器的功能，但是也是不难操作的。因为 Filter 本身就是一个 Java 类，通过创建 Java 类的步骤，在创建 Java 类的界面中，选择实现 Fileter 接口，如图 5-31 所示。

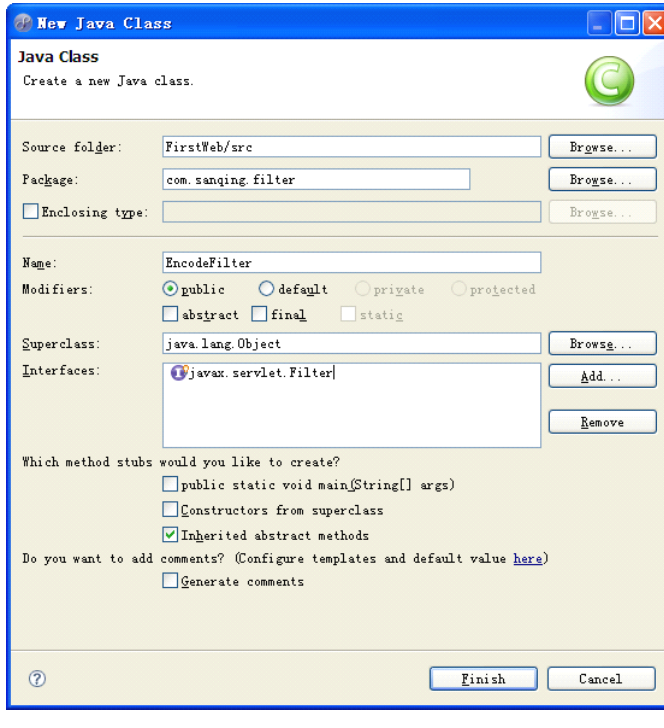


图 5-31　创建过滤器

单击“Finish”按钮，将完成过滤器的创建。在编辑区中打开程序后，在 doFilter

方法中加入过滤的功能代码，例如这里加入对编码进行过滤的代码，则过滤器的完整代码为：

```
01  package com.sanqing.filter;
02  //省略导入接口和类的代码
03  public class EncodeFilter implements Filter {
04      public void destroy() {
05      }
06      public void doFilter(ServletRequest req, ServletResponse res,
07              FilterChain chain) throws IOException, ServletException {
08          HttpServletRequest request=(HttpServletRequest)req;
09          request.setCharacterEncoding("UTF-8");
10          chain.doFilter(req, res);
11      }
12      public void init(FilterConfig arg0) throws ServletException {
13      }
14  }
```

开发完过滤器类后，和 Servlet 一样，还需要在“web.xml”文件中进行配置，可以直接通过代码进行配置，也可以通过界面化得操作进行配置。在编辑区中打开“web.xml”文件，在左下角中选择“Design”选项，将进入视图化配置的页面，在其中左边再选择“Filters”节点，界面如图 5-32 所示。

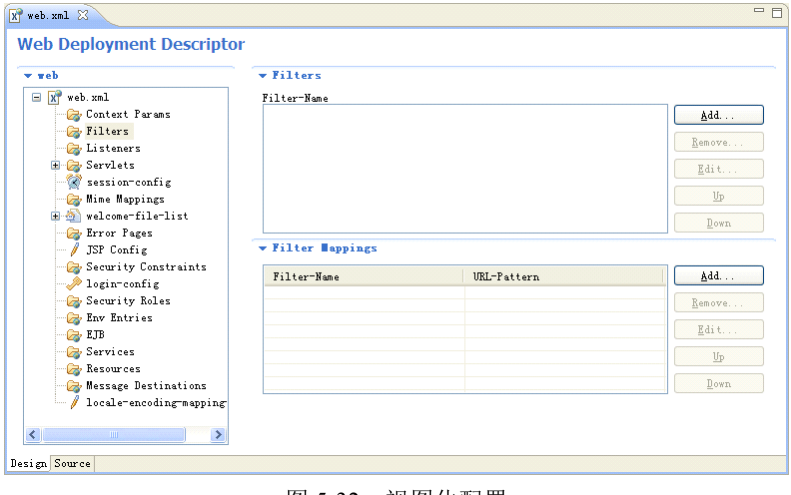


图 5-32　视图化配置

其中右上角部分是对过滤器类进行配置，单击“Add”按钮，将弹出过滤器类配置界面，如图 5-33 所示。

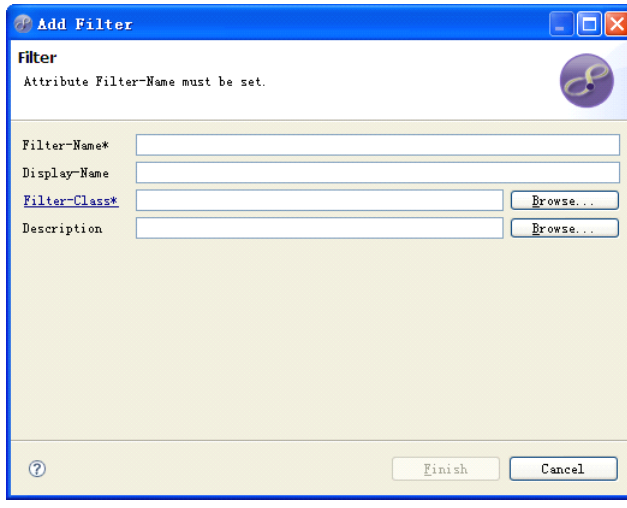


图 5-33　配置过滤器类

其中“Filter-Name”表示配置过滤器的名称，这里填写“EncodeFilter”。“Filter-Class”表示配置过滤器的类名，这里要是包括包名的全称类名，可以输入，也可以单击“Browse”按钮选择。其他两项可以选择不填，单击“Finish”按钮将完成过滤器类的配置，并且自动进入更详细的配置界面，在其中可以对初始化参数等进行配置，这里就不再讲解。

再次选择“Filters”节点，在视图化配置界面的右下部分中，单击“Add”按钮，将弹出过滤器映射的配置界面，如图 5-34 所示。

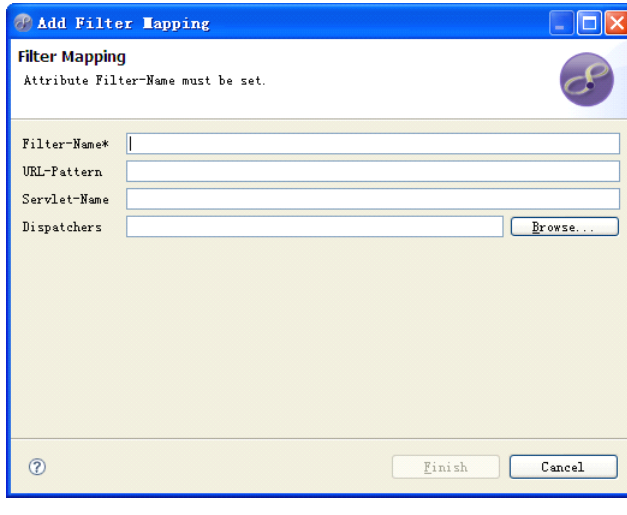


图 5-34　对过滤器映射进行配置

其中“Filter-Name”表示配置过滤器的名称，这里填写“EncodeFilter”，从而和前面的过滤器类对应。“URL-Pattern”表示过滤的 URL 地址，也就是对哪些 URL 访问进行过滤，这里填写“/*”，它表示对所有请求进行过滤。单击“Finish”按钮，将完成过滤器映射的配置。

选择编辑区左下角的“Source”选项，将看到“web.xml”文件的具体代码，可以发

现其中多出了如下代码：

```
<filter>
    <filter-name>EncodeFilter</filter-name>
    <filter-class>com.sanqing.filter.EncodeFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>EncodeFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

这就是通过前面界面化操作自动生成的代码。

说明：在“web.xml”文件的视图化配置中，不但能够对过滤器进行界面化配置，还能够对 Servlet、欢迎页等其他内容进行界面化配置。在后面的学习中，使用到时进行详细讲解。

5.9 开发网络商城的显示层和控制层

在管理数据库一章中，已经开发了网络商城的数据访问层，在本章学习完 JSP 和 Servlet 等 Web 程序后，我们就来开发网络商城的显示层和控制层。首先要创建一个名称为“store”的 Java Web 项目，然后将管理数据库一章开发的数据访问层相关包都复制到“store”项目中。因为要连接数据库，所以需要将数据库驱动包导入到 Web 项目中，这里只需要复制到“WebRoot”|“WEB-INF”|“lib”目录下。

接下来将本章中开发的编码过滤器程序复制到本项目中，不要忘记对它的配置，从而能够完成对中文信息的处理。

说明：传输中的信息的编码是和 JSP 程序或者 HTML 程序的编码无关的，它和 HTTP 协议有关，所有通过该协议传输的编码都是“ISO-8859-1”。“ISO-8859-1”编码是不支持中文的，所以要进行比较的编码处理。

5.9.1 开发录入商品的 HTML 程序

网络商城的主要功能就是让用户来购买商品，要将商品显示在页面中，这些商品需要在系统后台进行录入。录入商品的页面只需要有一个提交表单，所以这里开发一个 HTML 程序就可以。

这里我们选择高级模板来创建 HTML 程序。在创建 HTML 程序的界面中，选择默认模板，文件名填写“AddCommodity.html”，创建完成后将在编辑区中将该程序打开。删除原 body 标记对中的内容，然后在界面化操作部分中创建一个 Form 表单，创建界面如图 5-35 所示。

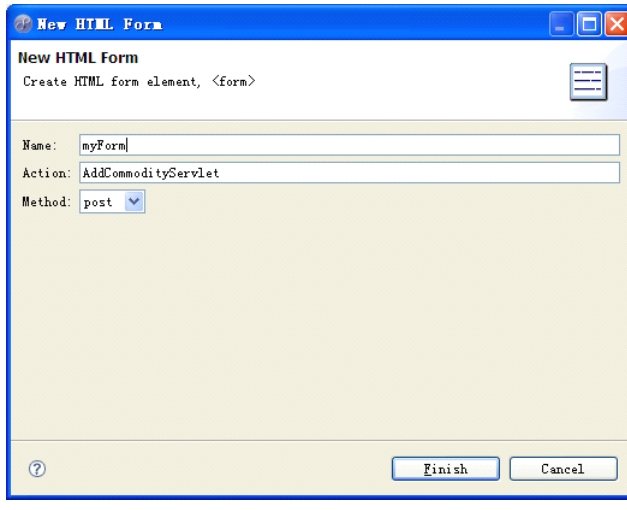


图 5-35　创建录入商品表单

创建表单完成后，在表单中输入“商品名称”，然后单击“Text Field”选项，弹出创建用于输入商品的文本框，填写必要信息后，界面如图 5-36 所示。

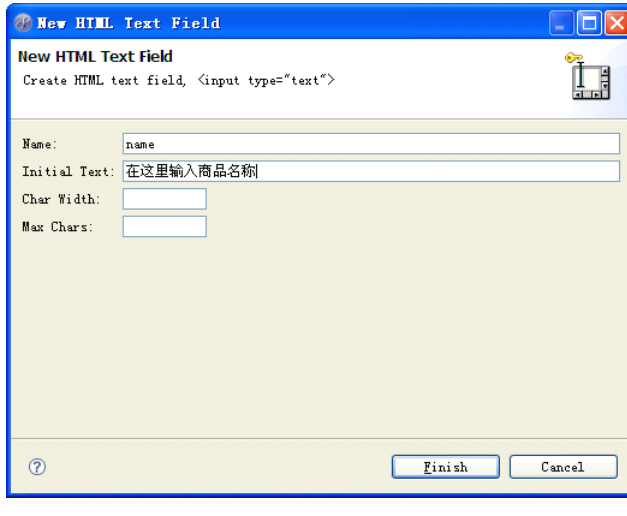


图 5-36　创建输入商品名称文本框

然后使用同样的办法，创建用于输入商品价格和折扣价格的文本框。最后定义提交按钮，单击“Button”选项，将弹出创建按钮的界面，填写必要信息后，界面如图 5-37 所示。

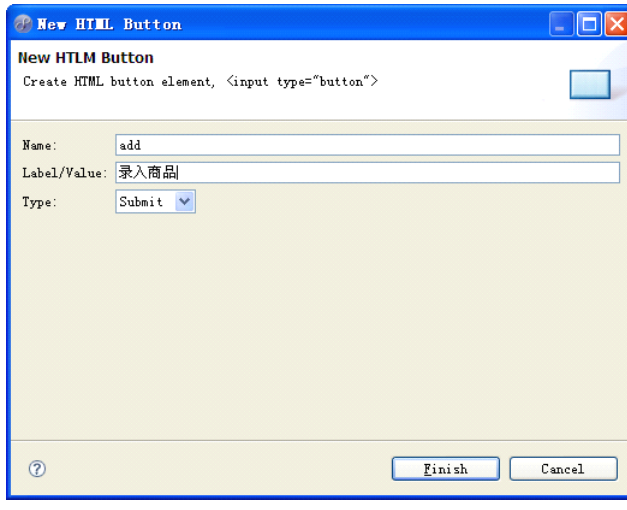


图 5-37　创建提交按钮

单击“Finish”按钮，从而完成提交按钮的创建，从而也完成录入商品表单的创建。选择编辑区左下角的“Preview”选项，则可以看到 IE 浏览器的预览，如图 5-38 所示。

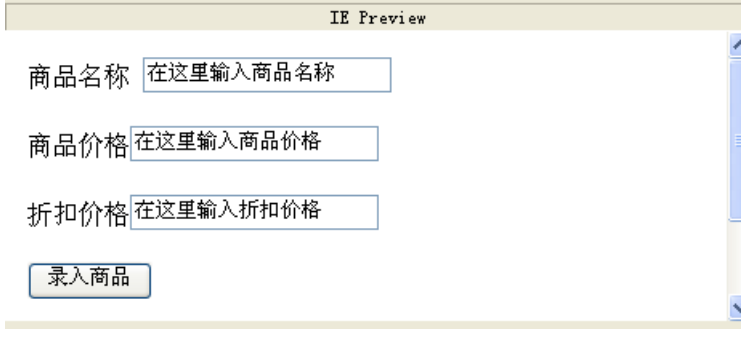


图 5-39　IE 浏览器预览

查看自动生成的 HTML 程序代码，可以看到代码的编写时不符合规范的。选中其中的表单代码，在右键弹出菜单中，选择“Source”|“Format”命令，调整后的代码为：

```
<form method="post" action="AddCommodityServlet" name="myForm">
    <p>
         商品名称
        <input type="text" value="在这里输入商品名称" name="name">
    </p>
    <p>
         商品价格
        <input type="text" value="在这里输入商品价格" name="price">
    </p>
    <p>
         折扣价格
        <input type="text" value="在这里输入折扣价格" name="agio">
    </p>
```

```
14          <p>
15               
16              <input type="submit" value="录入商品" name="add">
17          </p>
18      </form>
```

注意：使用这种方式创建的 HTML 程序效果并不十分美观的，这里主要是演示如何快速开发。在实际开发中，具体页面效果由专门的美工人员完成。

5.9.2 开发录入商品的 Servlet 程序

上一小节开发的录入商品 HTML 程序是项目中的显示层，单击“录入商品”按钮后会将表单中的信息提交到 Servlet 程序中。在本节中就来开发录入商品的 Servlet 程序，它主要来做控制层。

选中项目中的“com.sanqing.servlet”包，单击鼠标右键，在弹出的菜单中选择“New”|“Servlet”命令，在弹出的创建 Servlet 界面中，输入相关信息后，如图 5-38 所示。

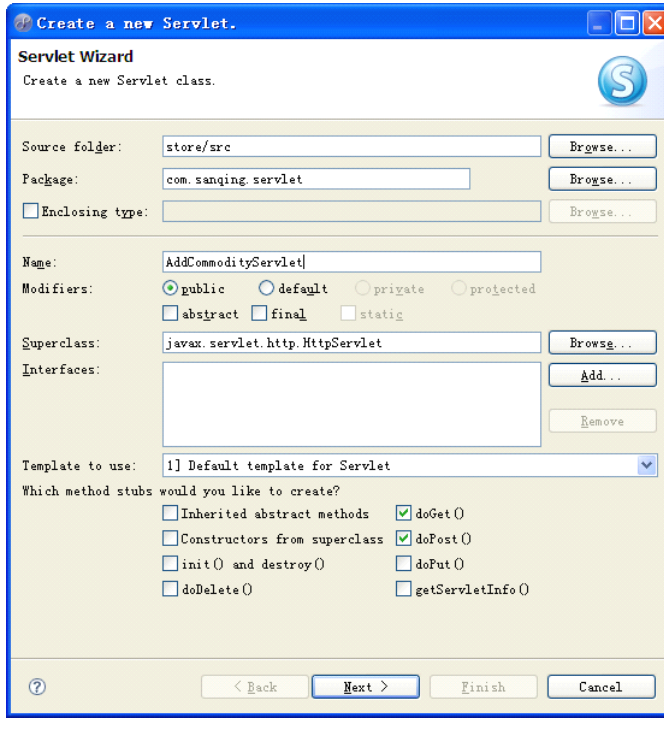


图 5-38　创建录入商品 Servlet

单击“Next”按钮，将进入配置 Servlet 的页面，修改其中的信息后，如图 5-39 所示。

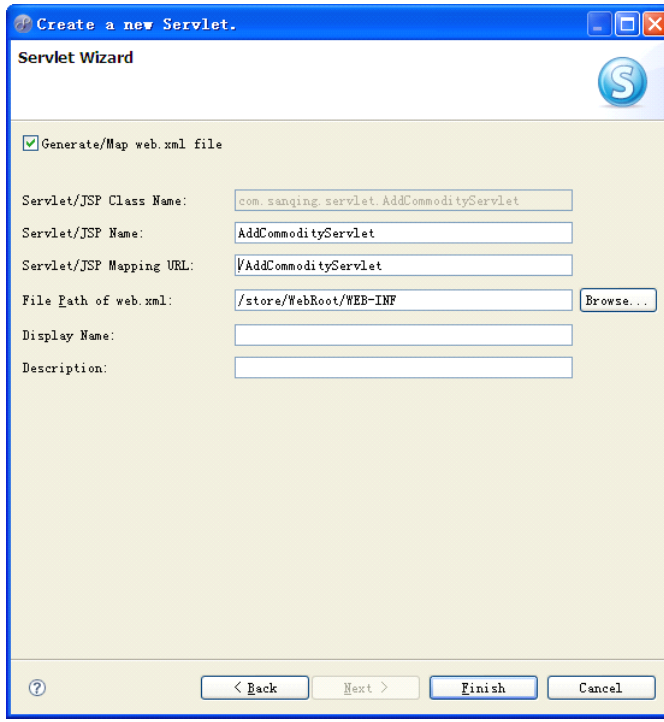


图 5-39　配置录入商品 Servlet

技巧：其中主要是对 Servlet 的 URL 进行修改，去掉其中的“/servlet”，这样能够保证 Servlet 的访问也在项目根目录下。

在编辑区中打开 Servlet 程序后，将代码改为如下内容：

```
01  package com.sanqing.servlet;
02  //省略导入接口和类的代码
03  public class AddCommodityServlet extends HttpServlet {
04      public void doGet(HttpServletRequest request, HttpServletResponse response)
05              throws ServletException, IOException {
06          doPost(request,response);
07      }
08      public void doPost(HttpServletRequest request, HttpServletResponse response)
09              throws ServletException, IOException {
10          String name=request.getParameter("name");  //接收表单参数
11          String price=request.getParameter("price");
12          String agio=request.getParameter("agio");
13          Commodity commodity=new Commodity();  //创建商品对象并设置信息
14          commodity.setName(name);
15          commodity.setPrice(Double.parseDouble(price));
16          commodity.setAgio(Double.parseDouble(agio));
17          CommodityDAO commodityDAO=
18                  CommodityDAOFactory.createCommodityDAOImpl();
19          commodityDAO.addCommodity(commodity);      //调用录入商品方法
20          request.getRequestDispatcher("AddSuccess.html").forward(request, response);
```

```
21          }
22      }
```

其中只在 doPost 方法中编写了代码，在 doGet 方法中调用 doPost 方法，在实际开发中通常也是这样操作的。

从第 10 行到第 12 行是接收提交表单中的信息。在第 13 行中创建了商品对象，然后依次将获取到的商品相关信息设置到对象中。在第 17 行通过商品 DAO 工厂类获取到商品 DAO 实现类后，在第 19 行调用 addCommodity 方法，从而将商品的信息保存到数据库中。在第 20 行执行服务器跳转，跳转到录入成功页面。

因为我们是通过 MyEclipse 中的集成功能创建的 Servlet，所以会自动对它进行配置，读者可以自己看一下它的配置信息。

5.9.3 开发录入商品成功页面

在录入商品 Servlet 中，录入商品成功后，将进行服务器跳转，跳转到“AddSuccess.html”HTML 程序。在本小节中就来开发该程序。

这里仍然采用高级模板来创建 HTML 程序，首先将原 body 标记中的内容换为“录入商品成功”信息，选中这行文字，在视图操作的工具栏中，依次选择 

选项、单击  加粗按钮、  居中按钮和  设置字体颜色按钮，在其中选择自己喜欢的颜色，这样成功页面的显示信息就开发完成。

然后开发一个能够进入显示所有商品页面的超链接。先在 body 标记中输入“显示所有商品”信息，选中后再在“HTML-Basic”选项中单击“Hyperlink”节点，在弹出的创建超链接界面，输入相关信息，如图 5-40 所示。

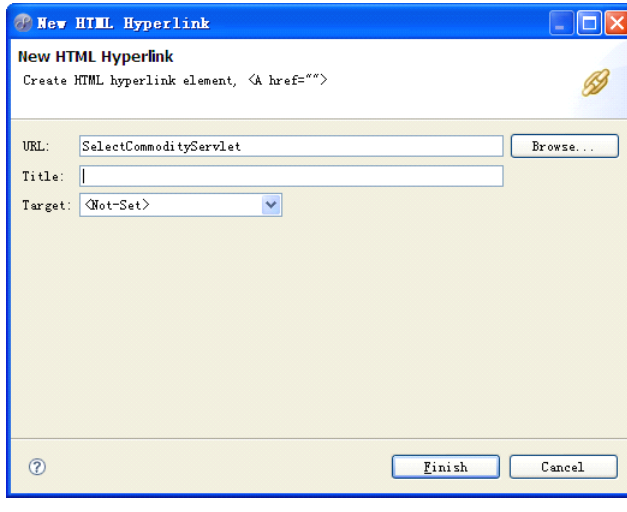


图 5-40　创建超链接

单击“Finish”按钮，将完成超链接的创建。对于超链接而言，也可以像普通文字一样进行操作，例如单击居中按钮等。

单击编辑区左下角的“Preview”选项，将可以查看到 IE 浏览器的预览，如图 5-41 所示。

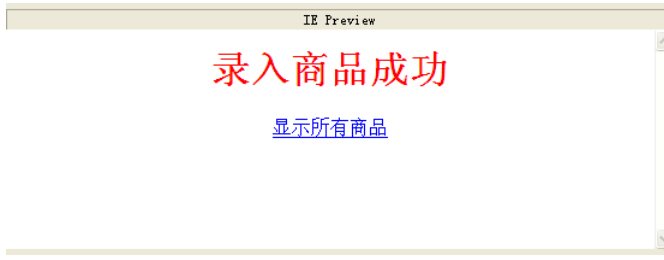


图 5-41　录入成功页面预览

调整自动生成的 HTML 程序代码格式，其自动生成的代码内容为：

```
<div align="center">
    <h1>
        <font color="#ff0000"><strong>录入商品成功</strong>
        </font>
    </h1>
</div>
<div align="center">
    <a href="SelectCommodityServlet">显示所有商品</a>
</div>
```

其中“显示所有商品”超链接指向查询所有商品 Servlet 的 URL。

5.9.4　开发查询所有商品 Servlet 程序

单击录入成功页面中的“显示所有商品”超链接，将会访问显示所有商品 Servlet，在本小节中就来开发该 Servlet 程序。

选中项目中的“com.sanqing.servlet”包，单击鼠标右键，在弹出的菜单中选择“New”|“Servlet”命令，在弹出的创建 Servlet 界面中，输入相关信息后，如图 5-42 所示。

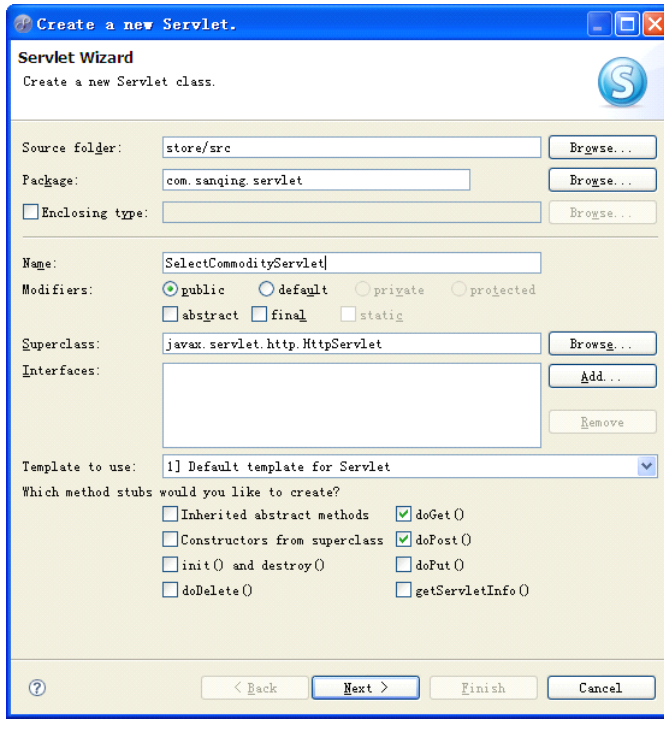


图 5-42　创建查询商品 Servlet

单击“Next”按钮，将进入配置查询所有商品 Servlet 的页面，修改其中的信息后，如图 5-43 所示。

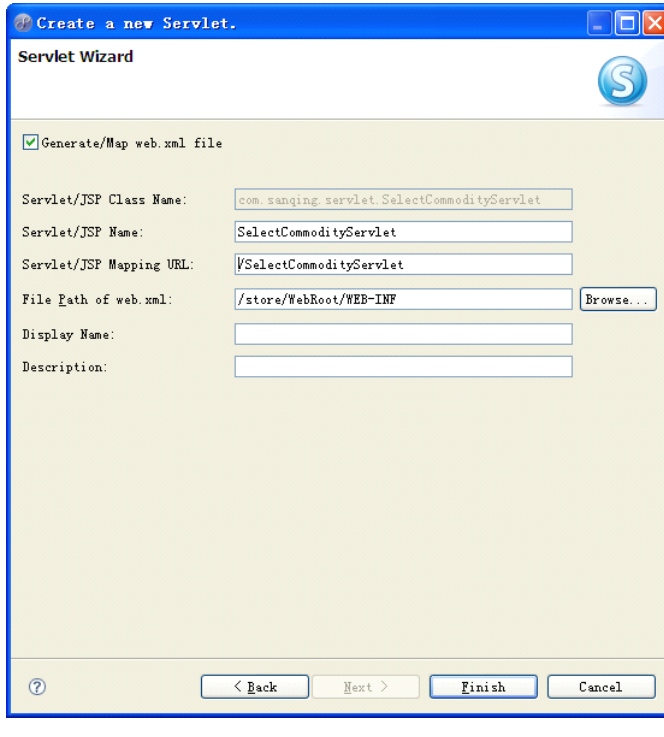


图 5-43　配置查询商品 Servlet

在编辑区中打开 Servlet 程序代码后，修改其中的代码，修改后的代码为：

```
01  package com.sanqing.servlet;
02  //省略导入接口和类的代码
03  public class SelectCommodityServlet extends HttpServlet {
04      public void doGet(HttpServletRequest request, HttpServletResponse response)
05              throws ServletException, IOException {
06          doPost(request,response);
07      }
08      public void doPost(HttpServletRequest request, HttpServletResponse response)
09              throws ServletException, IOException {
10          CommodityDAO commodityDAO=
11                  CommodityDAOFactory.createCommodityDAOImpl();
12          List<Commodity> commodityList=commodityDAO.findAllCommodity();
13          request.setAttribute("commodityList", commodityList);
14          request.getRequestDispatcher("AllCommodity.jsp")
15                                          .forward(request, response);
16      }
17  }
```

其中第 10 行通过 DAO 工厂类得到商品 DAO 实现接口，然后在第 12 行调用 findAllCommodity 方法，从而得到目录所有商品组成的集合。在第 13 行中将商品集合保存到 request 范围中。在第 14 行中进行服务器跳转，跳转到显示所有商品页面。

5.9.5 开发显示所有商品的 JSP 程序

在查询所有商品 Servlet 中将跳转到显示所有商品页面中，在本小节中就来开发该页面，在这里开发使用 JSTL 标签的 JSP 程序。

选中“WebRoot”目录，单击鼠标右键，在弹出的菜单中选择“New”|“JSP（Advanced Templates）”命令，将弹出使用高级模板创建 JSP 程序的界面，在其中填入文件名，如图 5-44 所示。

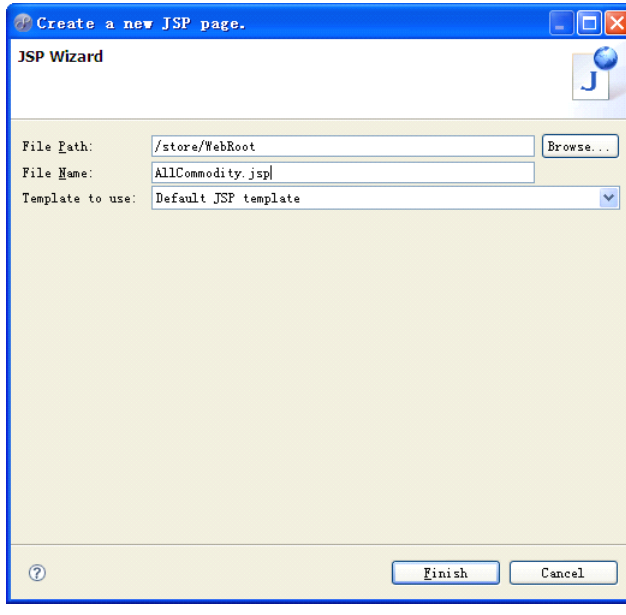


图 5-44　创建显示商品 JSP 程序

单击“Finish”按钮，将完成 JSP 程序的创建。因为在其中要使用到 JSTL 标签，所以在 page 指令下给出如下 taglic 指令。

```
<%@ taglib    prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

然后在 body 标记中，加入如下代码：

```
01    <table border="0" align="center">
02          <tr>
03                <td>商品名称</td>
04                <td>商品原价</td>
05                <td>折扣价格</td>
06          </tr>
07           <c:forEach var="commodity" items="${requestScope.commodityList}"
08                                                                  varStatus="stus">
09                <tr>
10                      <td><c:out value="${commodity.name}"></c:out></td>
11                      <td><DEL><c:out value="${commodity.price}"></c:out></DEL></td>
12                      <td><FONT color="red">
13                              <c:out value="${commodity.agio}"></c:out></FONT></td>
14                </tr>
15           </c:forEach>
16    </table>
```

技巧：使用 JSTL 标签后，对标签是不能够进行界面化设置的。例如第 11 行的删除线标记不能够直接添加，但是我们可以在其中先测试文字，然后进行界面化操作。加入相应标记后，再使用 JSTL 标签替换测试文字。

5.9.6　修改项目的首页

创建 Java Web 项目后，默认的项目首页是“WebRoot”根目录下的“index.jsp”JSP 程序，但是在该网络商城的项目中并没有使用该文件，所以要修改项目的首页。

打开“WebRoot”|“WEB-INF”目录下的“web.xml”文件，选择编辑区左下角的“Design”选项，将进入视图化配置的界面。在其中左面，选择“welcome-file-list”节点，在右面可以看到原“index.jsp”文件，选中它，单击“Remove”按钮，将它删除。再单击“Add”按钮，将弹出选择首页界面，如图 5-45 所示。

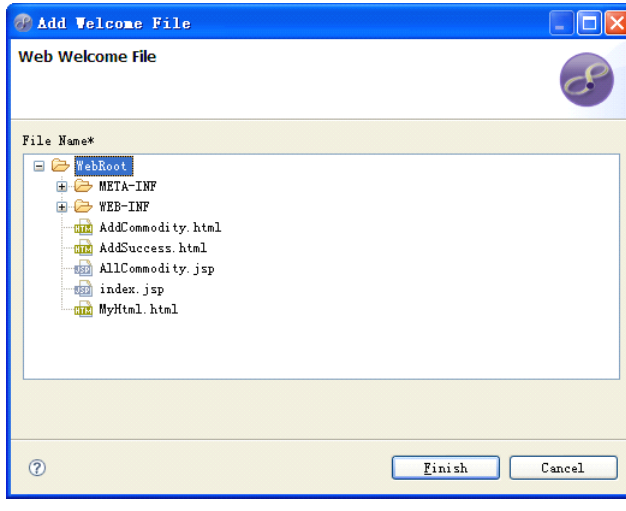


图 5-45　首页设置

在其中选择“AddCommodity.html”节点，也就是录入商品页面，单击“Finish”按钮，将完成首页的修改。

5.9.7　运行网络商城项目

到目前为止，我们的网络商城项目已经开发完成，它由 HTML 程序、JSP 程序、Servlet 程序和 Filter 过滤器程序组成，完成录入商品和显示所有商品两个功能。在本节中就来看一下如何发布和运行该项目。

在“Servers”界面中选择“Tomcat 6.x”服务器，单击鼠标右键，选择“Add Deployment”命令，将弹出发布项目界面，填入“store”项目后，如图 5-46 所示。

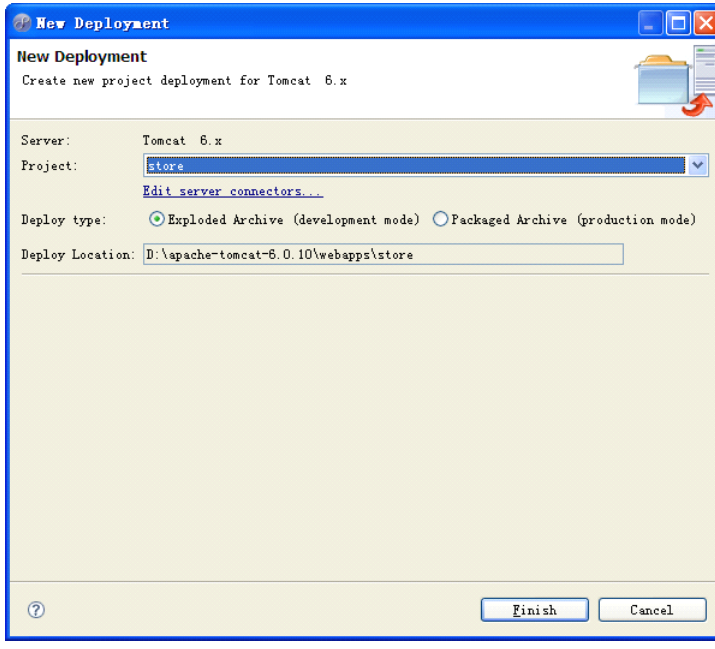


图 5-46　发布网络商城项目

单击“Finish”按钮将完成项目的发布。再次选中“Tomcat 6.x”服务器，在右键弹出菜单中，选择“Run Server”命令，将开始运行服务器。启动服务器完成后，打开浏览器，输入如下地址：

http://localhost:8080/store/

将进入网络商城项目首页，也就是录入商品页面，在其中填入一件商品，运行结果如图 5-47 所示。

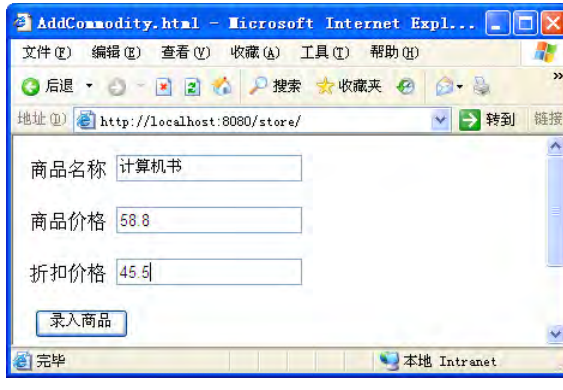


图 5-47　录入商品页面

单击“录入商品”按钮，将执行录入商品的 Servlet 程序，然后自动跳转到录入商品成功页面，如图 5-48 所示。

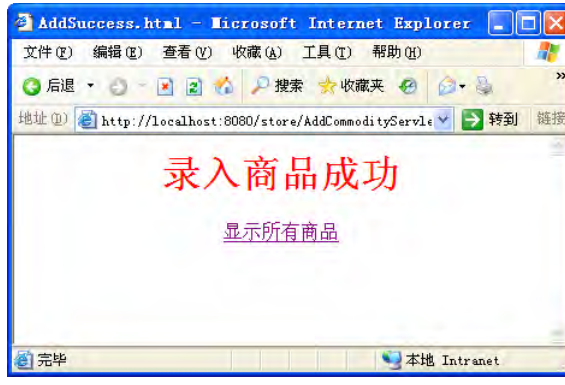


图 5-48　录入成功页面

单击“显示所有商品”超链接，将执行查询所有商品的 Servlet 程序，然后跳转到显示所有商品的页面，如图 5-49 所示。

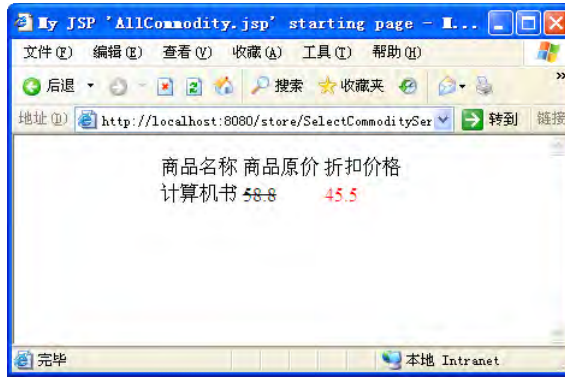


图 5-49　显示所有商品

因为目前只录入了一件商品，所以页面中只有“计算机书”这一件商品信息。

第6章　进行Struts开发

Struts 是流行最早和最广的实现 Web 层 MVC 架构的开源框架。Struts 是在 JSP 和 Servlet 技术的基础上开发出来的框架技术，通过它能够很好的将 Web 程序进行分层操作。

目前 Struts 存在两个版本，分别是 Struts 1.x 和 Struts 2，这两个版本之间有很大的区别。在本书中将使用两章来分别讲解这两个版本，本章先来讲解 Struts 1.x 版本。

6.1　开发 Struts 项目

由于 Struts 1.x 版本非常流行，所以当 MyEclipse 安装完成后就在其中集成了开发 Struts 项目的功能，而且是非常强大的。在本节中就先来看一下在 MyEclipse 中是如何创建 Struts 项目的。

6.1.1　创建 Web 项目

Struts 是一种 Web 项目的开发框架，所以仅能够在 Java Web 项目中使用。在 MyEclipse 开发 Struts 项目时，是要基于基本的 Java Web 项目的，所以我们要首先创建一个基本的 Web 项目。

在 MyEclipse 的菜单中，选择“File”|“New”|“Web Project”命令，将弹出创建 Java Web 项目的界面，在其中填写项目名等信息后，如图 6-1 所示。

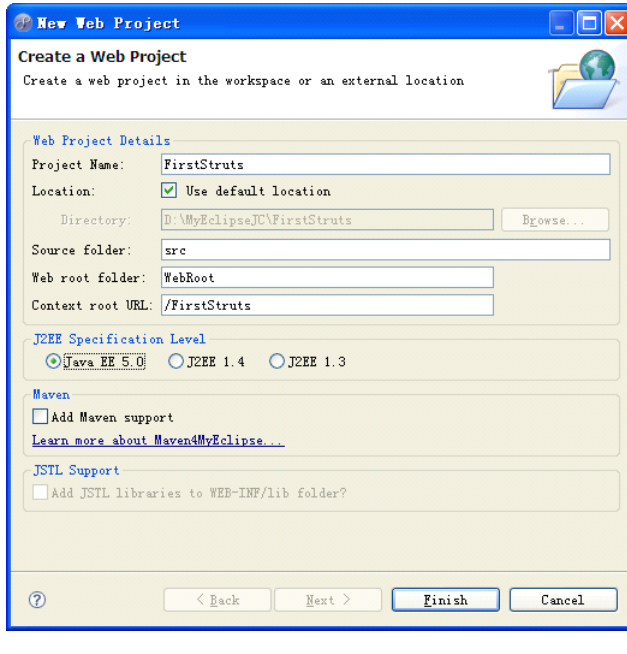


图 6-1　创建普通 Web 项目

这里将本节将要创建的 Struts 项目命名为“FirstStruts”。单击“Finish”按钮，将完成基本 Web 项目的创建。

6.1.2　加入 Struts 框架支持

创建基本的 Java Web 项目后，还不能进行 Struts 开发，还需要在其中加入 Struts 框架支持。

在包资源管理器中，选中要加入 Struts 框架支持的项目，这里选中“FirstStruts”项目。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Struts Capabilities”命令，将弹出加入支持操作界面，如图 6-2 所示。

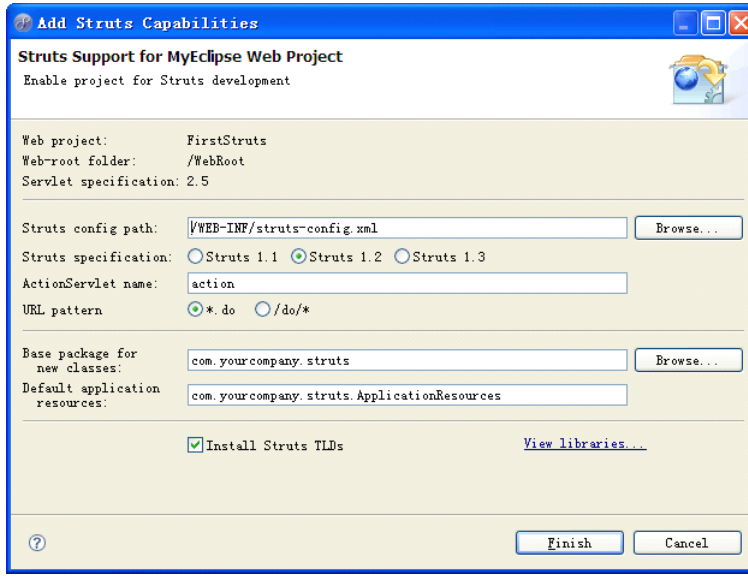


图 6-2　加入 Struts 框架支持

其中“Struts config path”表示 Struts 配置文件的名称和所在目录，通常采用默认值。“Struts specification”表示要加入 Struts 的版本，有三个版本可以选择，，这里选择最常用的 Struts 1.2 版本。“ActionServlet name”表示 Struts 核心控制器配置的名称，通常就采用默认的“action”。“URL pattern”表示使用中心控制器对哪些请求进行处理。

“Base package for new classes”表示新建的 Action 和 ActionForm 的根包名，这里填写“com.sanqing.struts”。“Default application resources”表示用来输入 Struts 1 的默认的资源文件名，它的位置默认就在 struts 包下。将单击“Finish”按钮将完成 Struts 框架支持。

说明：向普通项目中加入 Struts 支持，主要做的工作就是向项目加入 JAR 包，以及给出 Struts 的配置文件、在 web.xml 中对核心控制器进行配置。这些操作也是可以手动完成的，但是要相对复杂的多。

6.1.3 认识 Struts 项目结构

创建 Web 项目并加入 Struts 框架支持后，将在包资源管理器中多出一个新项目，将该项目展开，如图 6-3 所示。

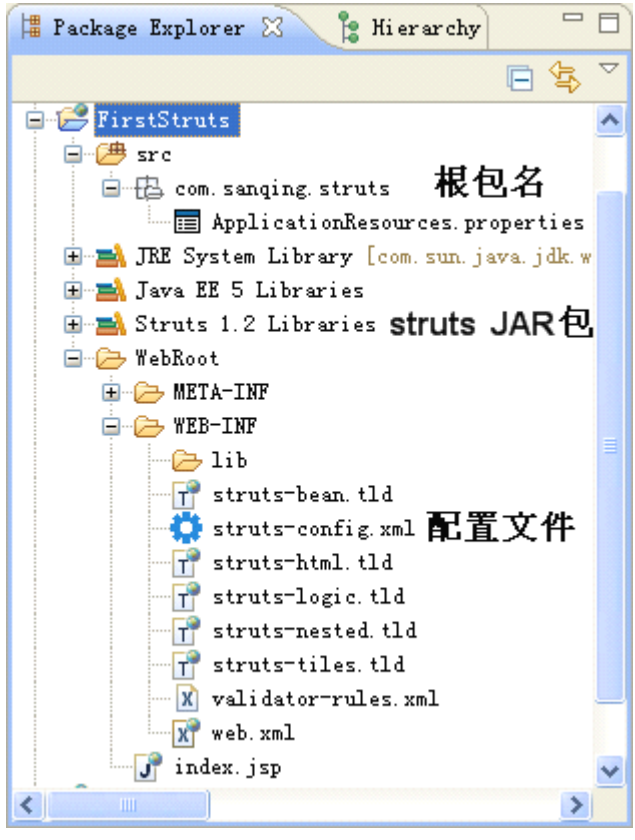


图 6-3　Struts 项目结构

其中“com.sanqing.struts”就是 Struts 程序的根包，Action 程序通常放在“action”子包下，ActionForm 程序通常放在“form”子包下。在“Struts 1.2 Libraries”节点下有 Struts 框架相关的 JAR 包，这是在 MyEclipse 中，在实际项目目录中，这些 JAR 包都保存在 lib 目录下。

在“WebRoot”|“WEB-INF”目录下，有多个文件，其中“web.xml”是 Web 项目中的配置文件，在前面的普通项目中就见过。“Struts-config.xml”是 Struts 框架的配置文件，在其中可以对 Action 和 ActionForm 等程序进行配置。

首先打开“web.xml”文件，其中部分代码如下：

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
```

```
08      <init-param>
09          <param-name>debug</param-name>
10          <param-value>3</param-value>
11      </init-param>
12      <init-param>
13          <param-name>detail</param-name>
14          <param-value>3</param-value>
15      </init-param>
16      <load-on-startup>0</load-on-startup>
17  </servlet>
18  <servlet-mapping>
19      <servlet-name>action</servlet-name>
20      <url-pattern>*.do</url-pattern>
21  </servlet-mapping>
```

其中第 3 行中的“org.apache.struts.action.ActionServlet”就是 Struts 中的核心控制器类，从名称和配置上可以看出，它是一个 Servlet。在其中配置核心控制器的名称为“action”，在第 20 行配置它的 URL 为“*.do”，表示所有以“.do”结尾的请求都有中心控制器处理，这些信息都是在加入 Struts 框架支持页面中配置的。

注意：从这里也可以看出，不管是什么框架，都是离不开基本的 JSP 和 Servlet 程序的。在框架中仅仅是对操作进行了封装，使开发更简单。

从第 4 行到第 16 行是对核心控制器中的初始参数进行配置。“config”参数表示以相对路径的方式给出 Struts 配置文件的名称和目录位置，它的默认值就是“/WEB-INF/struts-config.xml”，所以这里也可以不配置。

“debug”表示日志记录的详细程度级别，默认是 0，表示最少记录。“detail”表示设置 Digester 的记录级别，默认也是 0。“<load-on-startup>”标记配置的是核心控制器在服务器中的加载次序，数值越小，越先加载。

然后打开“Struts-config.xml”，它是 Struts 的配置文件，部分代码为：

```
01  <struts-config>
02      <data-sources />
03      <form-beans />
04      <global-exceptions />
05      <global-forwards />
06      <action-mappings />
07      <message-resources parameter="com.sanqing.struts.ApplicationResources" />
08  </struts-config>
```

当开发 Struts 相关的程序后，需要在该配置文件中进行配置，具体配置方法在后面的讲解中将会学习到。其中第 7 行中是对 Struts 的资源文件进行配置，它是加入 Struts 框架支持时自动加入的。

6.2 开发 Struts 项目中的相关程序

因为 Struts 是一种 MVC 框架，所以每一层都有相应的程序构成。在一个 Struts 项目中通常由 JSP 页面、Action、ActionForm 和 ActionForward 等程序组成。在 MyEclipse 中有多种创建这些程序的方法，我们主要讲解其中使用最多的方式，本节先来讲解通过新建向导的形式创建。

6.2.1 创建 Struts 项目中的 JSP 页面

在 Struts 项目中，应用于显示层的程序也是 JSP 程序，而且在其中也可以使用普通的 JSP 程序和 JSTL 标签，但是通常不这样做，而是在 JSP 程序中使用 Struts 自定义的标签。

在包资源管理器的“FirstStruts”项目中，选中要创建 JSP 页面的目录，这里选择“WebRoot”目录。单击鼠标右键，在弹出的菜单中，选择“New”|“JSP（Advanced Templates）”命令，将弹出创建 JSP 程序页面，在其中输入 JSP 程序名称和选择 Struts 模板后，如图 6-4 所示。

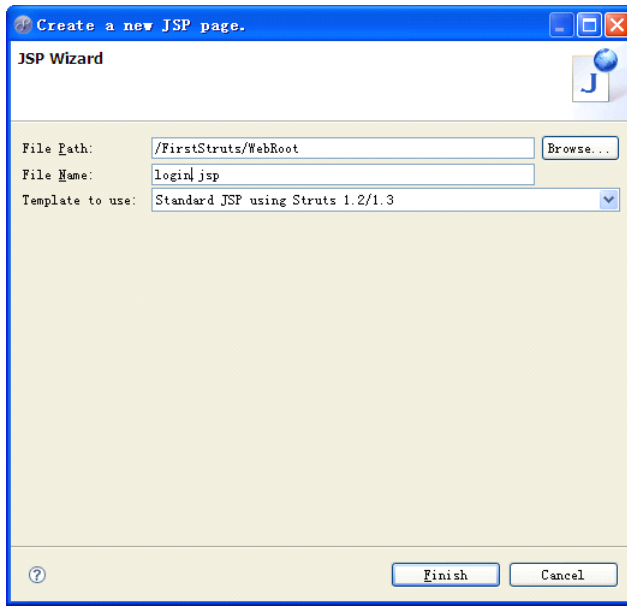


图 6-4　创建 Struts 项目中的 JSP 程序

在“Templates to use”中选择“Standard JSP using Struts 1.2/1.3”选项，这就是 Struts 项目中 JSP 程序所使用的模板。单击“Finish”按钮，将完成 JSP 程序的创建，并且在编辑区中打开该程序。和前面创建普通 JSP 程序所不同的就是，在 page 指令加入了 4 行定义标签的 taglib，具体代码如下所示。

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
```

这些就是 Struts 中的标签库。其中“bean”标签主要用来创建 bean 和访问 bean 及其属性，并且也能够根据 cookies、headers 和 parameters 的值创建相关 bean。“html”标签主要用来创建页面中的表单及其表单中的元素。“login”标签中主要包括流程控制相关的标签，这和前面讲解的 JSTL 中的流程控制标签非常相似的。“tiles”标签主要用来在页面中创建 tiles 样式。其中使用最多的就是“html”标签和“login”标签。

在编辑区的上半部分中，也就是界面化创建页面的部分中，其中具有“Struts-Basic”和“Struts-Form”两个选项。其中在“Struts-Form”中包含着创建表单及其元素的节点，单击“Form”节点，将弹出创建表单的界面，输入 URL 后，如图 6-5 所示。

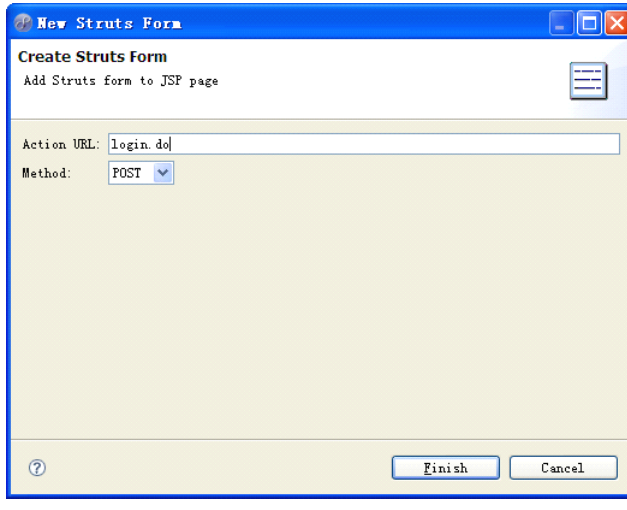


图 6-5　创建 Struts 表单

注意：创建表单时填写的提交 URL 为“login.do”，这和访问 Servlet 不同的。但是在一些程序中提交 URL 并没有以“.do”结尾，是因为在其中使用的是 Struts 标签，在框架中会自动给加上。如果使用的普通 HTML 标记，那就必须加上“.do”。

单击“Finish”按钮，将完成表单的创建。将鼠标放在表单中，单击“Struts-Form”选项中的“Text Field”节点，将弹出创建文本框的界面，如图 6-6 所示。

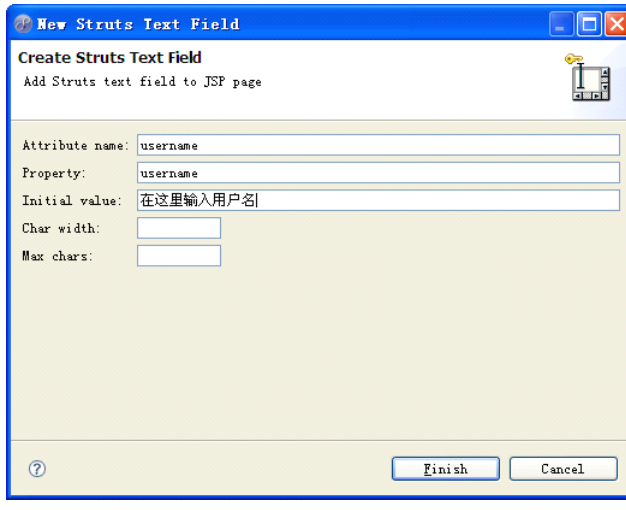


图 6-6　创建用户名文本框

其中“Attribute name”是文本框的名称，它对应着文本框标签的 name 属性。而“Property”也可以说是文本框的名称，只是它和 ActionForm 中的属性相对应，例如这里我们填写“username”，则在 ActionForm 应该也有一个名称为“username”的属性。接下来的三个选项分别表示默认值、文本框长度和最大输入字符数。

单击“Finish”按钮，则用于输入用户名的文本框已经创建完成。然后在“Struts-Form”选项中再单击“Password”节点，将弹出创建密码框的界面，它是和创建文本框非常类似的，这里就不再给出界面，通过它创建一个用于输入密码的密码框。

最后“Struts-Form”选项中单击“Button”节点，将弹出创建按钮的界面，在其中输入相关信息后，如图 6-7 所示。

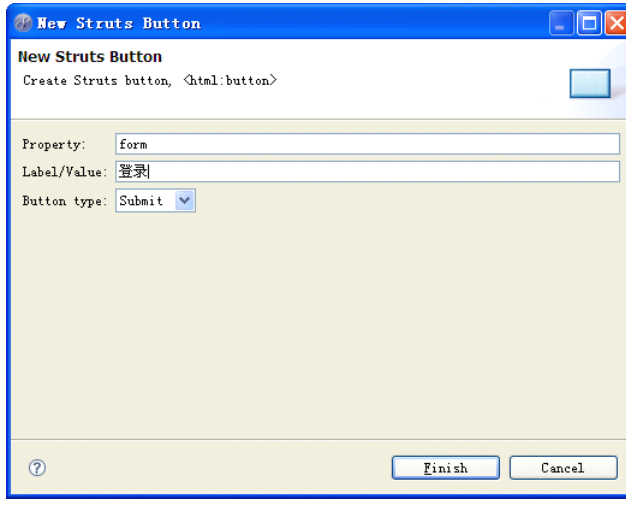


图 6-7　创建提交按钮

其中“Label/Value”表示提交按钮上的内容。“Button type”表示创建的按钮类型，如果在其中选择“Button”选择，则创建的仅是普通按钮。如果想创建提交按钮，需要

选择“Submit”选项。单击“Finish”按钮，将完成提交按钮的创建。

通过视图化的形式创建表单后，查看 JSP 程序的代码，可以看到多出了如下代码：

```
01  <html:form method="POST" action="login.do">
02      用户名<html:text value="在这里输入用户名" property="username"
03                                                  name="username"></html:text><br>
04      密码<html:password redisplay="true" value="在这里输入密码"
05              property="password" name="password"></html:password><br>
06      <html:submit value="登录" property="form"></html:submit>
07  </html:form>
```

注意：本代码是手动做的代码格式调整，因为在 MyEclipse 的编辑区的右键弹菜单中，选择“Source”|“Format”命令进行的调整并不是规范的。

6.2.2 开发 ActionForm 程序

ActionForm 是 Struts 项目中重要的组成程序之一。在 AcitonForm 中可以完成对属性的接收，并且还可以对属性进行验证。在本小节中就来看一下在 MyEclipse 中如何创建 ActionForm 程序。

在 Struts 项目中，通常将 actionForm 单独放在一个包中。在创建 Struts 项目时，已经创建了“com.sanqing.struts”根包名，现在在该包下创建“form”子包，这就是用于存放 ActionForm 的包。选中“form”子包，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“Struts 1.2”节点，界面如图 6-8 所示。

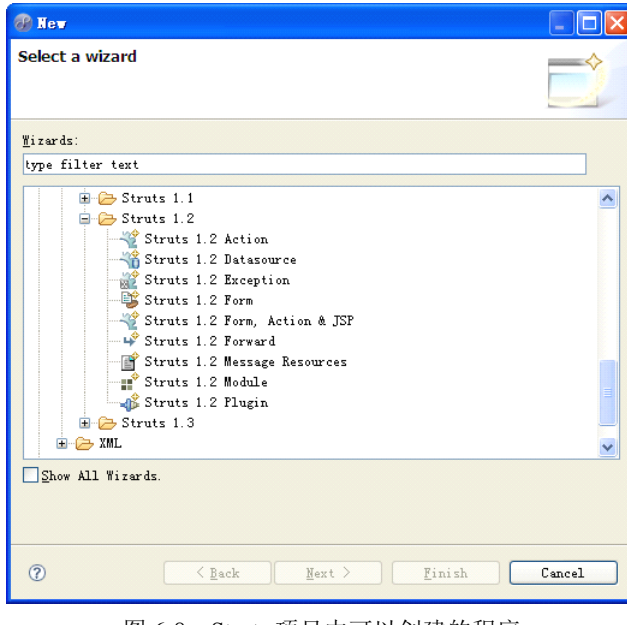


图 6-8　Struts 项目中可以创建的程序

在其中选择“Struts 1.2 Form”选项，单击“Next”按钮，将弹出创建 ActionForm 程序的界面，填写相应信息后，如图 6-9 所示。

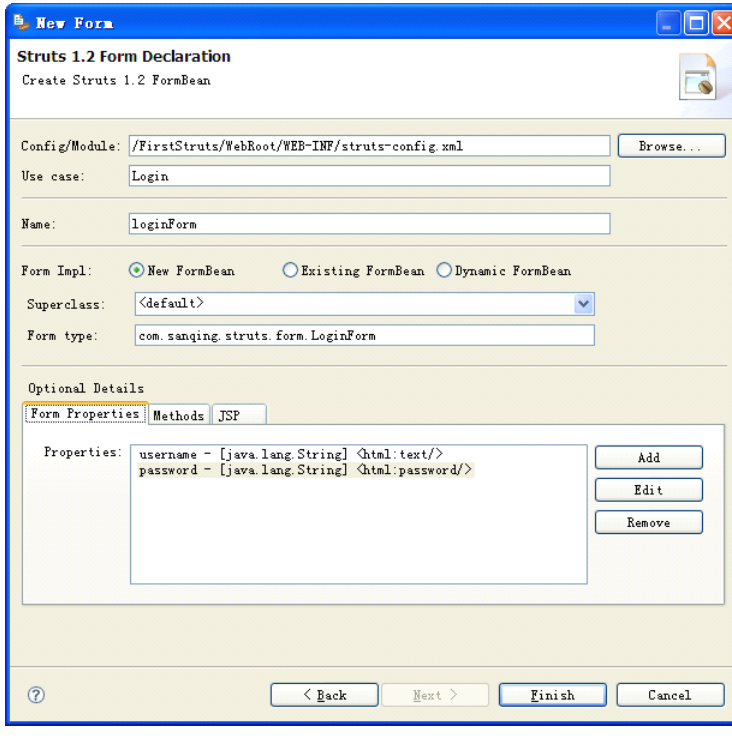


图 6-9　创建 ActionForm 程序

其中“Config/Module”表示在哪一个文件中对 ActionForm 进行配置。“Use case”表示 ActionForm 的基础用例，该选项可以不填，如果填写，则下面“Name”和“Form type”选项将自动填写。“Name”表示 ActionForm 的名称，对 ActionForm 进行配置时将用到该名称。

“Form Impl”表示 ActionForm 的类型，其中“New FormBean”选项表示新建该 ActionForm；“Existing FormBean”选项表示使用已经存在的 ActionForm；“Dynamic FormBean”选项表示使用动态 ActionForm。这里选择“New FormBean”选项来创建一个新的 ActionForm。“Superclass”表示 ActionForm 继承的父类，这里可以采用默认的父类，也可以在其中选择。“Form type”表示 ActionForm 的包含包名的完整类名。

“Form Properties”选项卡表示新创建的 ActionForm 中有哪些属性，单击“Add”按钮，将弹出创建属性的界面，如图 6-10 所示。

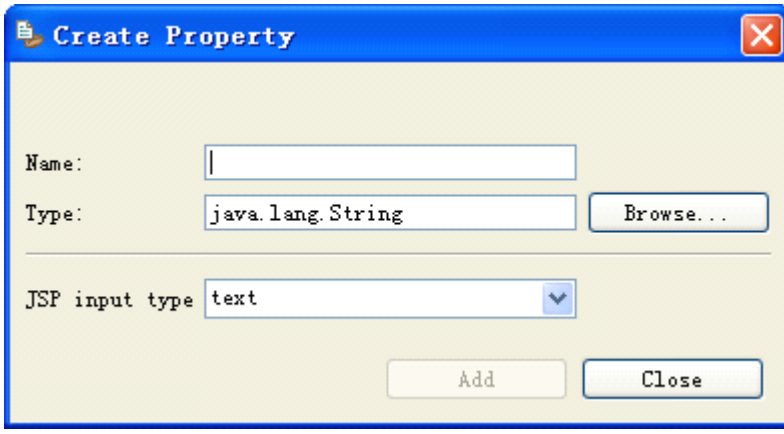


图 6-10　创建属性

其中“Name”表示属性的名称，这里要和表单中元素的 property 属性相对应上。“Type”表示属性的类型，默认是 String 字符串类型。“JSP input type”表示 JSP 页面中表单元素的类型。通过该界面添加“username”和“password”两个属性。

创建 ActionForm 程序界面中的“Methods”选项卡表示在 ActionForm 中创建哪些方法，这些方法都是重写父类中的方法。默认选中“validate”验证方法和“reset”重置方法，其中“validate”验证方法用于对表单元素值的验证，它的使用是比较多的。

注意：使用“validate”方法进行验证属于服务器验证，该验证可以由页面中的 JavaScript 完成，但是 JavaScript 容易被屏蔽掉，所以服务器验证是非常必要的。

“JSP”选项卡表示是否创建和该 ActionForm 相对应的 JSP 页面，如果选中其中的选项，将自动在填写的位置中创建 JSP 程序。因为我们已经创建了 JSP 程序，所以这里保持默认不选择的状态。

对这些信息设置后，单击创建 ActionForm 程序界面中的“Finish”按钮，将完成 ActionForm 的创建。打开位于“com.sanqing.struts.form”包下的“LoginForm”程序，可以看到自动生成的 ActionForm，它的重要代码如下所示：

```
01  package com.sanqing.struts.form;
02  import javax.servlet.http.HttpServletRequest;
03  import org.apache.struts.action.ActionErrors;
04  import org.apache.struts.action.ActionForm;
05  import org.apache.struts.action.ActionMapping;
06  public class LoginForm extends ActionForm {
07      private String username;
08      private String password;
09      public ActionErrors validate(ActionMapping mapping,
10              HttpServletRequest request) {
11          return null;
12      }
13      public void reset(ActionMapping mapping, HttpServletRequest request) {
14      }
15      public String getUsername() {
```

```
16              return username;
17          }
18          public void setUsername(String username) {
19              this.username = username;
20          }
21          public String getPassword() {
22              return password;
23          }
24          public void setPassword(String password) {
25              this.password = password;
26          }
27      }
```

其中第 7 行和第 8 行定义了表示用户名和密码的属性，从第 15 行开始依次为这两个属性定义了 Getter 和 Setter 方法。第 9 行定义了 validate 验证方法，在方法中可以输入用于验证的相关代码。第 13 行定义了 reset 重置方法，可以对属性进行重置操作。

自动生成 ActionForm 的同时，还会在“struts-config.xml”文件中对 ActionForm 进行配置，配置代码为：

```
01      <form-beans >
02        <form-bean name="loginForm" type="com.sanqing.struts.form.LoginForm" />
03      </form-beans>
```

其中第 2 行的 type 属性指定的就是 ActionForm 的完整类名，name 属性就是为 ActionForm 起的名称，它是在创建界面中“Name”选项指定的。当在 Action 中使用 ActionForm 时，将使用到该名称。

6.2.3 开发 Action 程序

Action 是 Struts 项目中最核心的程序。在 Action 程序中调用 ActionForm，从而获取到通过 ActionForm 获取到的属性值。然后使用属性值调用业务逻辑层，最后根据业务逻辑层的返回值进行跳转。从功能上可以看到，Action 程序是和 Servlet 程序非常相似的，它们都做控制层。

在 Struts 项目中，通常将 Action 程序单独放在一个包下。在“com.sanqing.struts”根包下，创建一个名称为“action”的子包。选中“action”子包，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Action”选项，单击“Next”按钮，将弹出创建 Action 程序的界面，如图 6-11 所示。

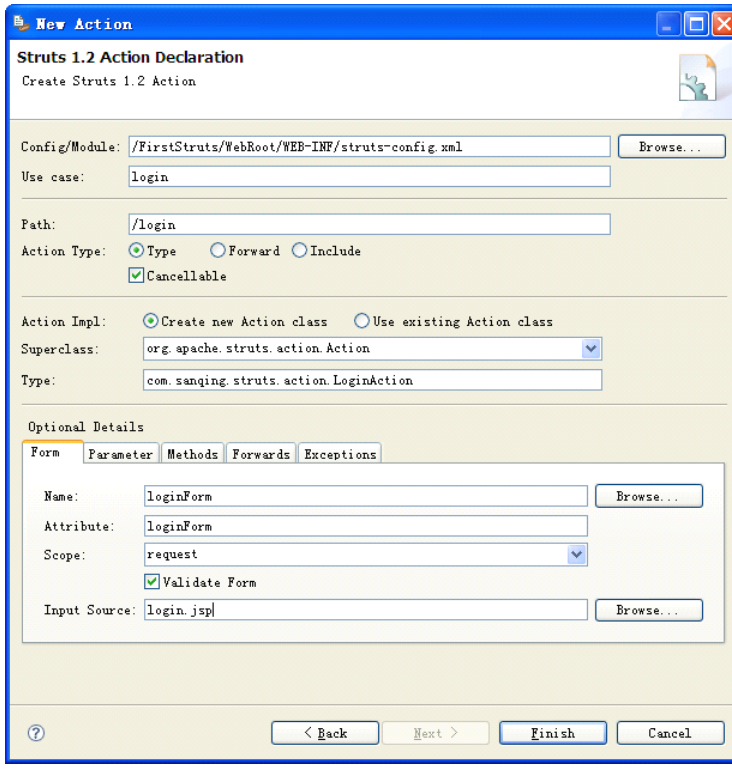


图 6-11　创建 Action 程序

其中“Config/Module”和“Use case”两个选项是和创建 ActionForm 界面中的选项相同的，作用也相同。“Path”表示访问该 Action 使用的访问路径，当前 Action 的完整访问路径为“http://localhost:8080/ FirstStruts/login.do”，这里需要注意不要忘记“.do”。“Action Type”表示新创建 Action 的类型，通常采用默认的“Type”选项。

在“Action Impl”中有两个选项，“Create new Action class”选项表示创建新 Action，“Use existing Action class”选项表示使用已经存在的 Action。“Superclass”表示创建 Action 继承的父类。“Type”表示新创建 Action 的全称类名。

技巧：根据 Action 继承父类不同，则 Action 的种类不同，最简单的 Action 继承的是“org.apache.struts.action.Action”类。除了最基本的 Action 外，还有动态 Action 等。使用动态 Action 可以避免 Action 的数量膨胀，从而使多种操作定义在一个 Action 中。

在界面的下半部分中，有 5 个选项卡。“Form”选项卡是对 Action 中使用的 AtionForm 进行设置，这些都对应配置文件中的属性，其中“Attribute”指定 ActionForm 的名称，“Scope”指定作用范围，“Validate Form”表示是否使用 ActionForm 中的验证方法，“Input Source”表示验证失败后返回的页面。

在“Parameter”选项卡中用来设置 Action 的参数，当使用 DispatchAction 等特殊 Action 时才会使用到，对于普通的 Action 来说是没有作用的。在“Methods”选项卡中，设置了在 Action 中定义哪些方法，这里采用默认的 execute 方法就可以，它也是 Action

的主体方法，功能代码都定义在该方法中。

在“Forwards”选项卡中用来设置 Action 中使用的 ActionForward，也就是执行的页面跳转。单击“Add”按钮，将弹出添加 ActionForward 的界面，如图 6-11 所示。

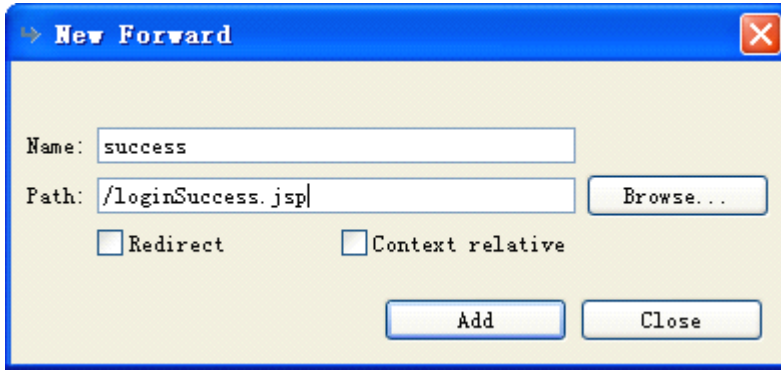


图 6-11　添加跳转

通过这个界面首先添加登录成功的跳转，在其中“Name”中输入“success”，在“Path”中输入“/loginSuccess.jsp”，它表示当返回 success 字符串时，跳转到登录成功页面。使用同样的方式，添加登录失败跳转，失败后跳转回登录页面。

在“Exceptions”选项卡中，是对 Action 中的异常进行添加。单击其中的“Add”按钮，将弹出添加异常的界面，如图 6-12 所示。

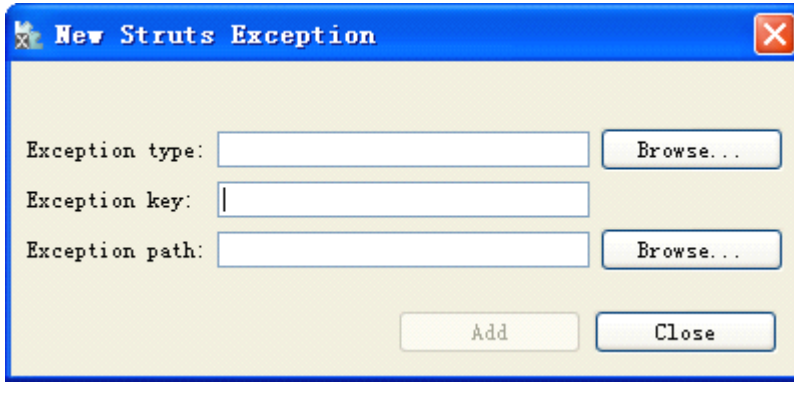


图 6-12　添加异常

其中“Exception type”表示异常的类型，这里可以是 API 中已定义的异常，也可以是自定义异常。“Exception key”表示异常指定的 key 键值。“Exception path”指定发生该异常后跳转到的页面。在 Action 中，异常是可以不添加的，具体异常的创建在后面还会讲解到。

回到创建 Action 程序界面，单击“Finish”按钮将完成创建。在编辑区中打开自动创建 Action 的程序，其代码为：

```
01  package com.sanqing.struts.action;
02  import javax.servlet.http.HttpServletRequest;
03  import javax.servlet.http.HttpServletResponse;
```

```
04  import org.apache.struts.action.Action;
05  import org.apache.struts.action.ActionForm;
06  import org.apache.struts.action.ActionForward;
07  import org.apache.struts.action.ActionMapping;
08  import com.sanqing.struts.form.LoginForm;
09  public class LoginAction extends Action {
10      public ActionForward execute(ActionMapping mapping, ActionForm form,
11              HttpServletRequest request, HttpServletResponse response) {
12          LoginForm loginForm = (LoginForm) form;
13          return null;
14      }
15  }
```

其中第 10 行就是 Action 的主方法，所有的功能代码都将放置在该方法中。在第 12 行中使用到了 LoginForm，它就是我们在上一小节中创建的 ActionForm。在 LoginForm 中保存了提交表单中的用户名和密码属性值，获取这些值后就可以调用相应的方法，然后根据不同的调用结果，跳转到不同的页面。

注意：这里我们只是将自动生成的代码给出，并没有加入任何操作的代码。如果想完成上述功能需要加入相应的代码。

同时生成 Action 的同时，还会同时在 Struts 配置文件中对该 Action 进行配置，配置代码如下：

```
01  <action-mappings >
02      <action
03        attribute="loginForm"
04        input="login.jsp"
05        name="loginForm"
06        path="/login"
07        scope="request"
08        type="com.sanqing.struts.action.LoginAction">
09        <set-property property="cancellable" value="true" />
10        <forward name="errer" path="/login.jsp" />
11        <forward name="success" path="/loginSuccess.jsp" />
12      </action>
13  </action-mappings>
```

其中使用的属性都是和前面界面操作相对应的。例如第 3 行指定 Action 中使用的 ActionForm，第 6 行的 path 属性指定该 Action 的访问地址。第 10 行和第 11 行分别定义了登陆成功跳转和登录失败跳转。

6.2.4 开发全局 ActionForward 跳转

学习上面的三种 Struts 框架中的程序，对于大部分项目来说已经足够了，但是它们并不是 Struts 框架的全部程序。在本小节中将学习 Struts 框架中的 ActionForward，在前面学习 Action 时，已经看到它的主方法的返回值就是 ActionForward，它通常使用 ActionMapping 调用 findForward 方法获取 ActionForward，这里完全也可以通过 new 的

方法创建 ActionForward 对象然后返回。但是这并不是本小节讲解的重点，本小节主要讲解如何创建全局 ActionForward。

注意：ActionForward 和前面讲解的 ActionForm 和 Acton 有很大不同，它并不是一个程序，而仅仅是 Struts 配置文件中的一个配置。

全局 ActionForward 并不是必须的，但是在某些功能中，使用全局 ActionForward 可以节省很多代码。例如在聊天室网站中，是要求用户必须登录才能浏览的，这时候就应该使用全局 ActionForward 跳转，不管用户浏览网站中的哪一个页面，判断没有登录，就会跳转到登录页面。

在选择创建 Struts 程序的界面中，选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 ForWard”选项，单击“Next”按钮，将弹出创建 ActionForward 的界面，如图 6-13 所示。

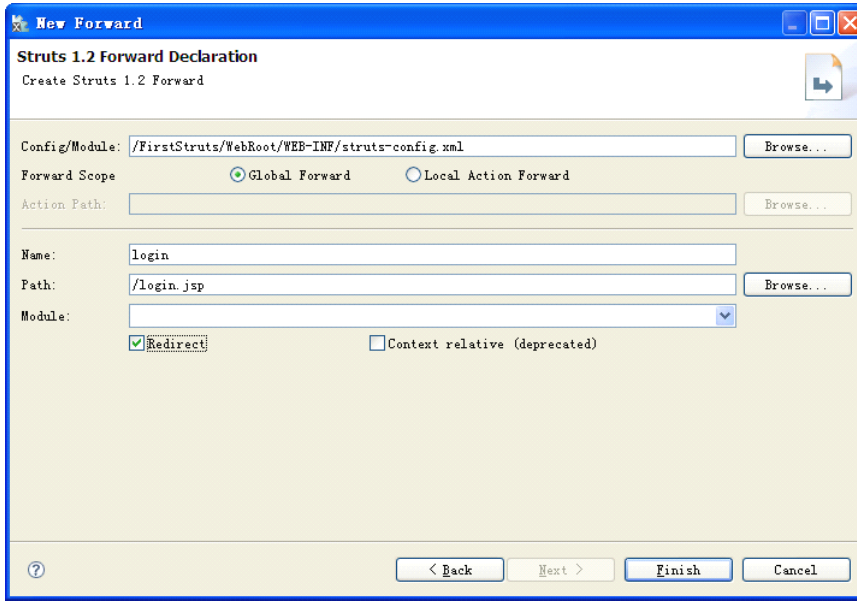


图 6-13　创建 ActionForward

其中“Config/Module”表示在哪一个文件中进行配置。“Forward Scope”表示 ActionForward 的范围，其中“Global Forward”表示全局的，也就是对所有 Action 起作用；“Local Avction Forward”表示局部的，如果选择该项时，需要指定作用在哪一个 Action 中。

“Name”表示 ActionForward 的名称，它和 Action 中返回的字符相对应。“Path”表示全局跳转的 URL 地址。“Redirect”表示是否重定向操作，对于全局跳转而言，该选项通常是选中的。

单击“Finish”按钮，将完成全局 ActionForward 的创建。在 Struts 的配置文件中，将多出一条全局跳转的配置，其代码如下：

```
01      <global-forwards >
02        <forward
03          name="login"
```

```
04              path="/login.jsp"
05              redirect="true" />
06        </global-forwards>
```

如果有多条全局跳转，都将配置在<global-forwards >标记对中。但在 Action 中返回“login”时，虽然没有定义局部跳转，但是会使用该全局跳转，从而跳转到登录页面。

6.2.5 开发 Struts 中的异常

在 Struts 项目中，有两种异常处理的方法，分别是编程式异常和声明式异常。在编程式异常处理中是通过手动的编写 try-catch 语句，根据获取的不同异常执行不同的跳转，它不是 Struts 特有的。声明式异常是 Struts 框架特有的，在代码中使用 throw 语句抛出异常，该异常将抛出 Struts 框架，然后在 Struts 框架中进行处理。声明式异常主要是在 Struts 配置文件中进行配置声明。

在进行声明式异常操作前，要进行一些准备工作。如果项目中需要的异常在 API 中并不存在，这时候要进行自定义异常。例如这里定义一个密码错误异常，其代码为：

```
01    package com.sanqing.struts.exception;
02    public class PasswordErrorException extends Exception {
03          public PasswordErrorException() {
04                super();
05          }
06          public PasswordErrorException(String message) {
07                super(message);
08          }
09    }
```

在该自定义异常中除了无参构造函数外，还定义了拥有字符串参数的构造函数，从而能够传递异常信息。

声明式异常中的信息并不是手动给出的，而是定义在一个资源文件中，通过键值对的方式获取。该资源文件就是定义在根包下的“ApplicationResources.properties”文件，打开该文件，在编辑区中有两种视图方式，在左下角选择“Properties”选项，将可以看到界面视图。单击“Add”按钮，将弹出添加资源信息的界面，添加后，界面如图 6-14 所示。

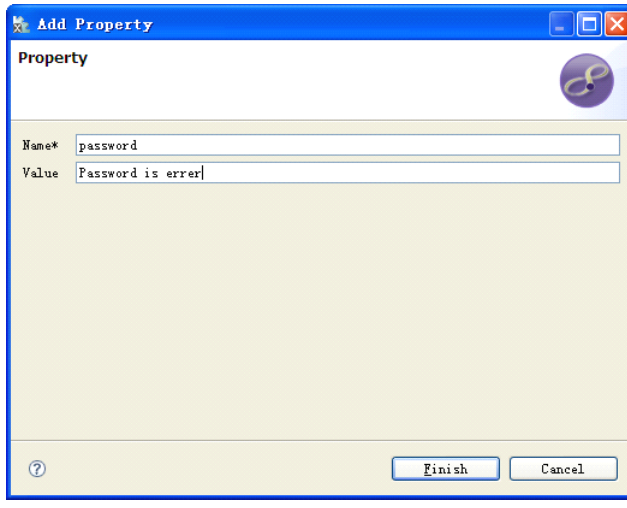


图 6-14　添加资源信息

单击“Finish”按钮，将完成资源信息的添加。保存后，选择编辑区左下角的“Source”选项，将可以看到多出了如下一行信息。

password=Password is errer

经过上面两步的操作，我们就可以继续来进行声明式异常的开发了。在选择创建 Struts 程序的界面中，选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Exception”选项，单击“Next”按钮，将弹出创建声明式异常的界面，如图 6-15 所示。

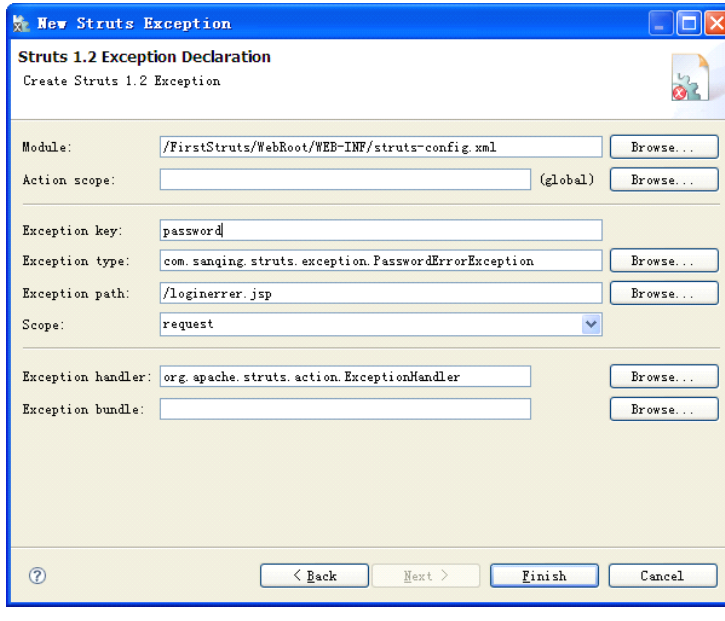


图 6-15　声明式异常

其中“Action scope”表示异常的处理范围，单击“Browse”按钮可以选择作用在哪一个 Action 中，如果不填，则定义一个全局异常，这里我们选择全局。“Exception key”表示异常对应的资源键，也就是资源文件中等号前面的内容。“Exception type”表示异

常类型，这里选择自定义的密码错误异常。“Exception path”表示当发生异常后跳转的页面。“Scope”表示异常的作用范围，可以选择“request”或者“session”，默认是“request”。“Exception handler”表示 Struts 框架中进行异常处理的类，使用界面创建时会自动给出默认值。

单击“Finish”按钮，将完成全局声明异常的创建。在 Struts 的配置文件中，将多出声明式异常的代码，具体代码如下：

```
<global-exceptions >
  <exception
    key="password"
    path="/loginerrer.jsp"
    type="com.sanqing.struts.exception.PasswordErrorException" />
</global-exceptions>
```

进行如上配置后，当 Struts 项目抛出 PasswordErrorException 异常后，将自动跳转到“loginerrer.jsp”，并且在该页面中可以将资源文件中“password”对应的内容输出。

说明：在实际开发中，编程式异常和声明式异常都是会被用到的。编程式异常更容易理解，因为完成时 Java 基本语法中的内容。而声明式异常在代码量上更少，更简单。

6.2.6 同时创建 JSP+ActionForm+Action 程序

在 Struts 项目中，最重要的三种程序就是 JSP 页面程序、ActionForm 程序和 Action 程序，它们是紧密结合在一起的。所以在 MyEclipse 中定义了同时创建这三种程序的选项，在开发中通常使用该选项进行创建。只所以把该内容放在最后来讲解，是因为它要借助前面很多的内容。

在选择创建 Struts 程序的界面中，选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Form,Action & JSP”选项，单击“Next”按钮，将首先弹出创建 ActionForm 的界面，该界面是和图 6-9 相同的，这里再创建一个用于注册的 ActionForm,，如图 6-16 所示。

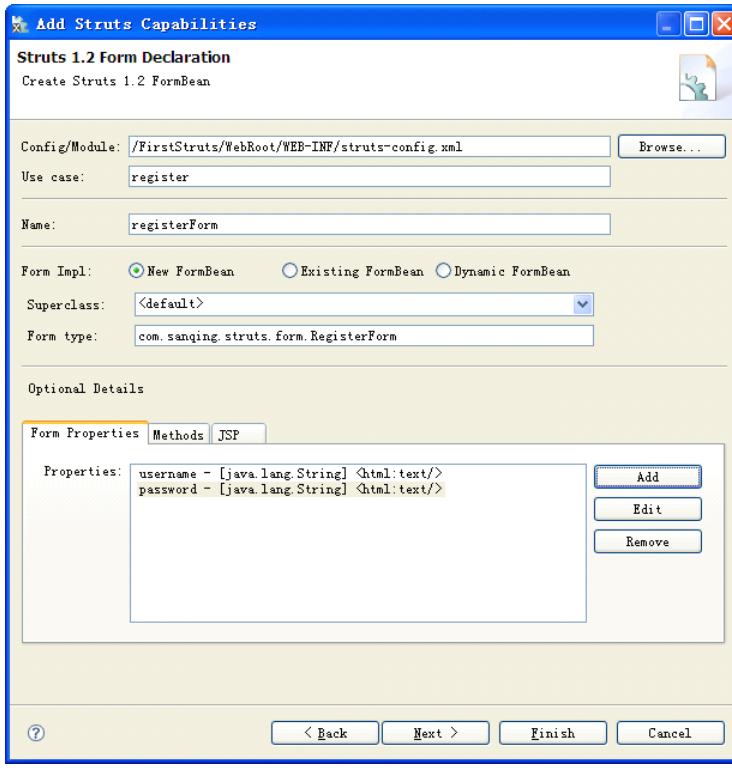


图 6-16　创建注册 ActionForm

选择其中的“JSP”选项卡，选中“Create JSP form”选项，然后输入 JSP 程序地址，如图 6-17 所示。

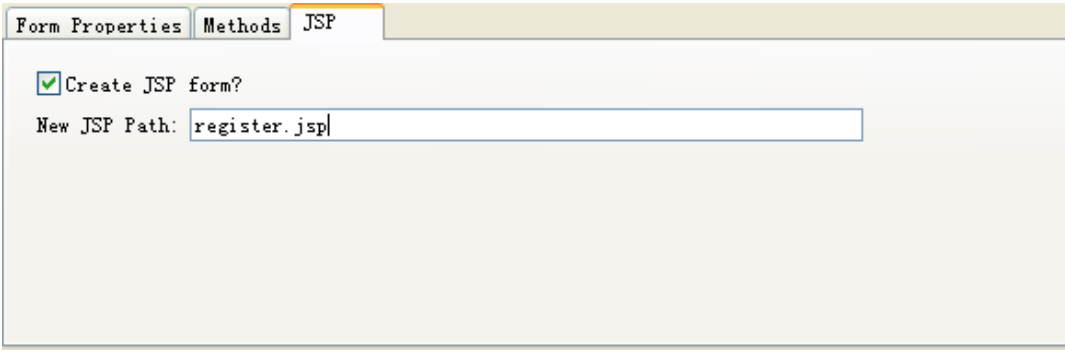


图 6-17　同时创建 JSP 程序

在创建 ActionForm 界面中单击“Next”按钮将进入到创建 Action 的界面，如图 6-18 所示。

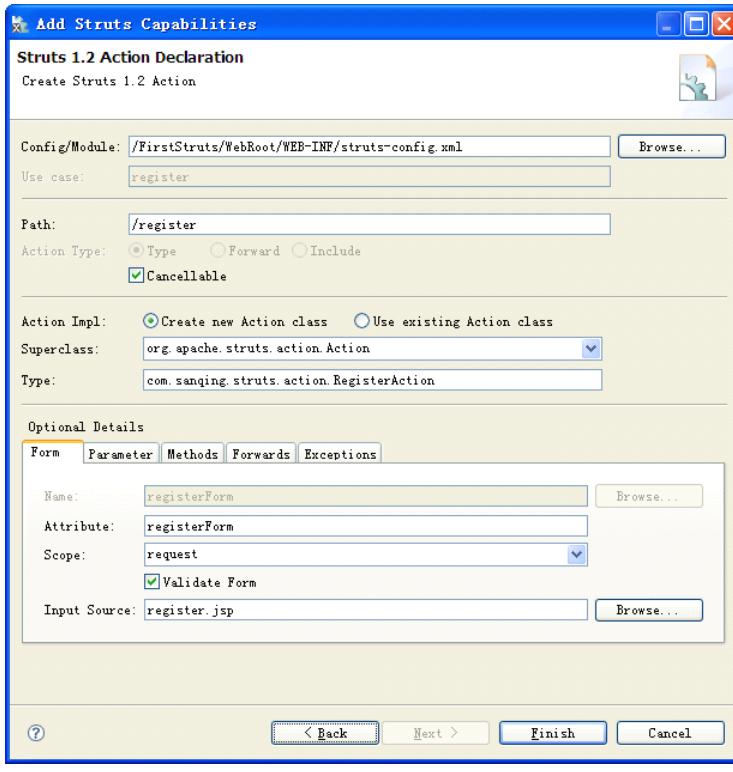


图 6-18　创建注册 Action

该界面中的信息都是自动生成的，其中会根据 ActionForm 的名称生成对应的 Action 的名称。在“Form”选项卡中会自动对 ActionForm 进行配置。在“Forwards”选项卡中需要添加跳转，如图 6-19 所示。

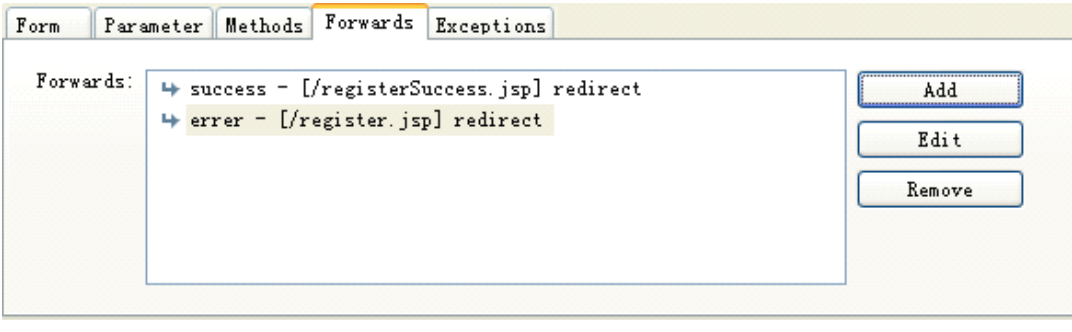


图 6-19　添加跳转

在创建注册 Aciton 界面中，单击“Finish”按钮将完成三种程序的创建。从生成的程序可以看到，使用这种方式创建的 ActionForm 和 Action 与单独创建它们是没有差别的。它们之间最大的区别就是使用这种方式自动生成了 JSP 页面，其中重要代码为：

```
01    <html:form action="/register">
02        username : <html:text property="username"/>
03                                <html:errors property="username"/><br/>
04        password : <html:text property="password"/>
```

```
05                              <html:errors property="password"/><br/>
06          <html:submit/><html:cancel/>
07    </html:form>
```

可以看到自动生成的代码是非常规范的，并且给出了错误处理标签。

6.3 通过视图创建 Struts 程序

在上一节中，已经学习了通过界面化的创建方式开发了 Struts 项目最重要的几种程序。在 MyEclipse 中，不但集成了该功能，而且还集成了一种视图化的创建方式，这种方式给人的感觉更直观，更能看出程序之间的关系。

6.3.1 认识视图操作界面

打开 Struts 的配置文件，默认是“WebRoot”|“WEB-INF”目录下的“struts-config.xml”文件。在编辑区中如果打开的是代码形式，选择编辑区左下角的 “Design”选项，将以视图的形式打开该程序，如图 6-20 所示。

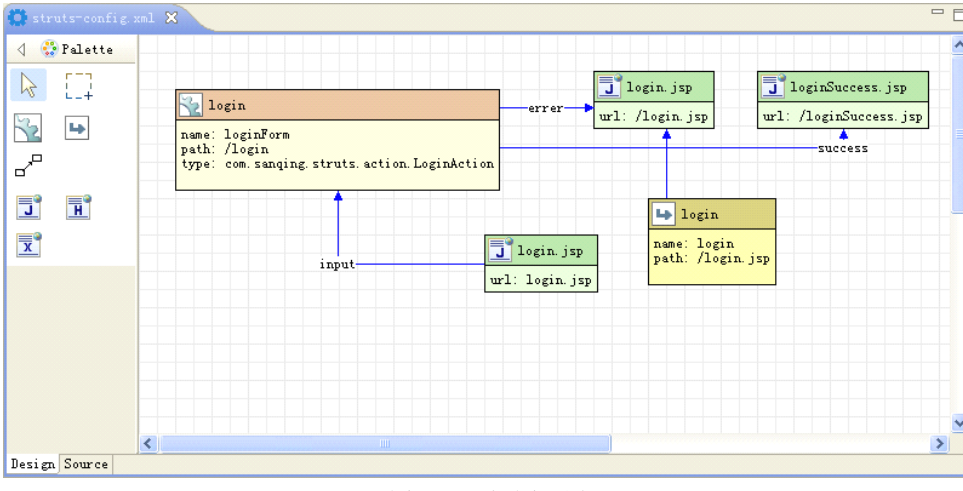


图 6-20 视图配置

其中左半部分是用于创建程序的按钮，它们分别对应的创建程序如图 6-21 所示。

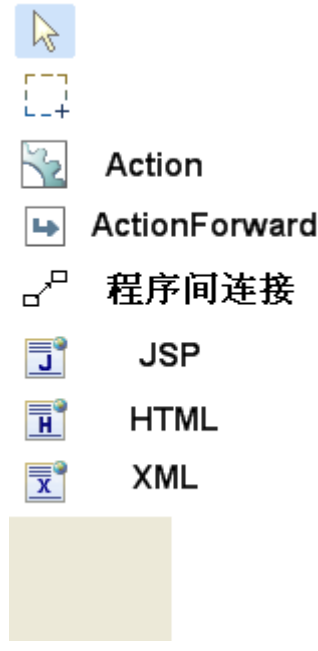


图 6-21　创建按钮

在编辑区的右边，是用于显示已有程序的“画布”，在其中不但显示出有什么程序，而且还会显示它们之间的关系。在“画布”中，单击鼠标右键，在弹出菜单中选择“New”命令，将出现多个选项，通过它们可以进入对应创建界面，从而创建相对应的程序。

6.3.2　通过视图创建程序

这里我们通过 Action 和 JSP 程序讲解如何使用本节中所讲到的视图。首先单击创建 Action 的按钮，然后进入“画布”，选择创建的位置，再次单击鼠标，将弹出创建 Action 的界面，在其中输入相应信息后，单击“Finish”按钮，将完成 Action 的操作。在“画布”中将出现该 Action，如图 6-22 所示。

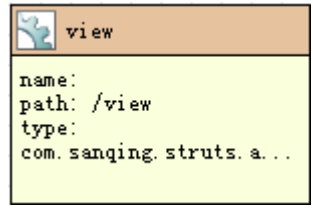


图 6-22　视图创建 Action

在该 Action 视图中，单击鼠标右键，在弹出的菜单中选择“Properties”命令，将出现弹出创建 Action 界面，在其中可以对原有信息进行修改。双击 Action 视图可以打开该 Action 的代码。

创建完 Action 后，再来创建一个 JSP 程序。单击创建 JSP 程序的按钮，同样进入“画布”后单击鼠标，将弹出创建 JSP 程序的界面，在其中输入文件名为“view.jsp”，单击“Finish”按钮将完成 JSP 程序的创建，如图 6-23 所示。

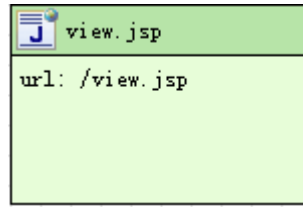


图 6-23　视图创建 JSP

对该视图同样能够进行 Action 视图的操作，可以对它进行修改和打开代码。

6.3.3　建立程序间连接

创建 Action 和 JSP 程序后，就可以在这两个程序之间建立连接。单击建立连接按钮，在画布中，首先单击 Action 视图，然后单击 JSP 视图，将弹出建立 ActionForward 的界面，如图 6-24 所示。

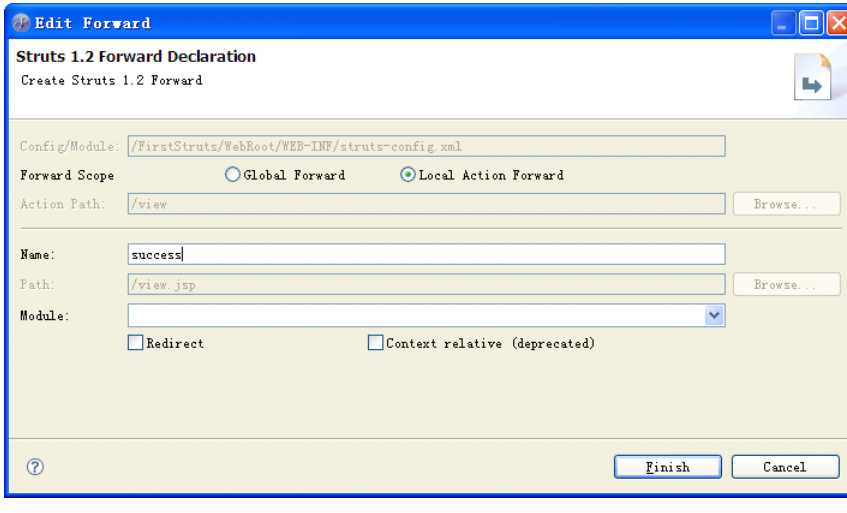


图 6-24　建立程序间连接

其中只需要填写“Name”选项，单击“Finish”按钮完成程序间的连接。在“画布”中将把两个程序建立连接，如图 6-25 所示。

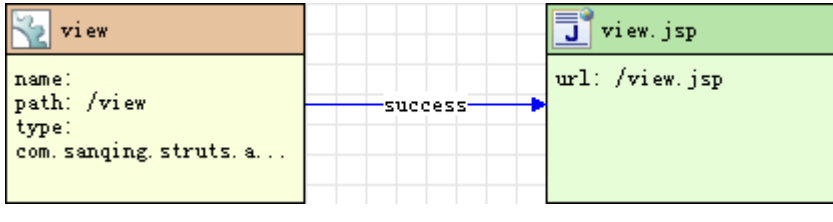


图 6-25　建立连接后

其中“success”就是界面中配置的，表示当 Action 中返回“seccess”时，跳转到“view.jsp”页面。进行该操作后，会在 Struts 配置文件中进行相应的配置。

6.4 快速搭建论坛项目后台

在实际开发中，每一个系统项目都要有一个后台，管理员在后台中进行管理。在本节中将开发一个论坛系统，管理员将对论坛中的主题进行管理，包括发表主题，查看已有主题、删除主题等。

说明：开发具体程序之前，要首先创建一个 Java Web 项目，我们给它取名为“BBS”，它就是我们的论坛项目，然后为该项目加入 Struts 框架支持。

6.4.1 开发论坛数据访问层

数据访问层在 Web 项目中的主要作用就是操作数据库，执行增删改查等数据库操作。在前面的学习中我们已经学习了使用 JDBC 技术开发数据访问层，这里我们仍然采用它来开发，从而也就不详细的讲解。

注意：只要是开发具有数据访问层的程序，就需要将数据库的驱动包导入到 Java Web 项目中，不然不能够正常的操作数据库。

在前面的学习中已经知道，使用 JDBC 技术开发的数据访问层主要由数据库封装类，DAO 实现类、DAO 接口、DAO 工厂类等程序组成，除此之外，还要创建相应操作实体类，例如这里应该创建主题实体类。这些代码在我们的项目中都可以查看到，这里我们只给出 DAO 实现类代码，如下所示。

```
01  package com.sanqing.dao;
02  //省略导入接口和类的代码
03  public class TopicDAOImpl implements TopicDAO {
04      public TopicDAOImpl(){
05      }
06      public void addTopic(Topic topic) {
07          Connection conn = DBConnection.getConnection();      //获得连接对象
08      String addSQL = "insert into topic(title,content,username,date) values(?,?,?,?)";
09          PreparedStatement pstmt = null;                      //声明预处理对象
10          try {
11              pstmt = conn.prepareStatement(addSQL);     //获得预处理对象并赋值
12              pstmt.setString(1, topic.getTitle());          //设置第一个参数
13              pstmt.setString(2, topic.getContent());        //设置第二个参数
14              pstmt.setString(3, topic.getUsername());       //设置第三个参数
15              pstmt.setDate(4, new java.sql.Date(topic.getDate().getTime()));
16              pstmt.executeUpdate();                                    //执行更新
17          } catch (SQLException e) {
18              e.printStackTrace();
19          } finally{
20              DBConnection.close(pstmt);                     //关闭预处理对象
21              DBConnection.close(conn);                      //关闭连接对象
22          }
```

```
23          }
24          public void deleteTopic(int id) {
25              Connection conn = DBConnection.getConnection();      //获得连接对象
26              String updateSQL = "delete from topic where id=?";
27              PreparedStatement pstmt = null;                      //声明预处理对象
28              try {
29                  pstmt = conn.prepareStatement(updateSQL);//获得预处理对象并赋值
30                  pstmt.setInt(1, id);                  //设置第一个个参数
31                  pstmt.executeUpdate();                    //执行更新
32              } catch (SQLException e) {
33                  e.printStackTrace();
34              } finally{
35                  DBConnection.close(pstmt);                       //关闭预处理对象
36                  DBConnection.close(conn);                        //关闭连接对象
37              }
38          }
39          public List<Topic> findAllTopic() {
40              Connection conn = DBConnection.getConnection();      //获得连接对象
41              String updateSQL = "select * from topic";
42              PreparedStatement pstmt = null;                      //声明预处理对象
43              List<Topic> topicList = new    ArrayList<Topic>();
44              try {
45                  pstmt = conn.prepareStatement(updateSQL);//获得预处理对象并赋值
46                  ResultSet rs = pstmt.executeQuery();          //执行查询
47                  while(rs.next()) {
48                      Topic topic = new Topic();          //实例化
49                      topic.setId(rs.getInt(1));          //获得主题 ID
50                      topic.setTitle(rs.getString(2));
51                      topic.setContent(rs.getString(3));
52                      topic.setUsername(rs.getString(4));
53                      topic.setDate(rs.getDate(5));
54                      topicList.add(topic);
55                  }
56              } catch (SQLException e) {
57                  e.printStackTrace();
58              } finally{
59                  DBConnection.close(pstmt);                  //关闭预处理对象
60                  DBConnection.close(conn);                   //关闭连接对象
61              }
62              return topicList;                               //返回查询到的所有商品
63          }
64      }
```

其中第 6 行定义了增加主题的方法，当发表新主题时，将执行该方法，从而将主题中的信息保存到数据库中。在第 24 行中定义了删除指定主题的方法，该方法接收主题 id 参数，从而将该主题删除。第 39 行定义了查询所有主题的方法，通过该方法可以将

目前数据库中的所有主题记录查询出来。

6.4.2 开发发表主题功能

在论坛中，最主要的功能应该算是发表主题。发表主题的用户可以使普通用户，也可以是后台管理员。这里我们主要是通过 MyEclipse 集成的功能，快速开发完成发表主题功能所需要的 JSP、ActionForm 和 Action 程序。

在执行创建之前，要首先在“com.sanqing.struts”包下创建“action”和“form”子包。然后选中“form”子包，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“Struts 1.2”节点，将进入选择创建程序的界面，在其中选中“struts 1.2 Form,Action & JSP”选项，如图 6-26 所示。

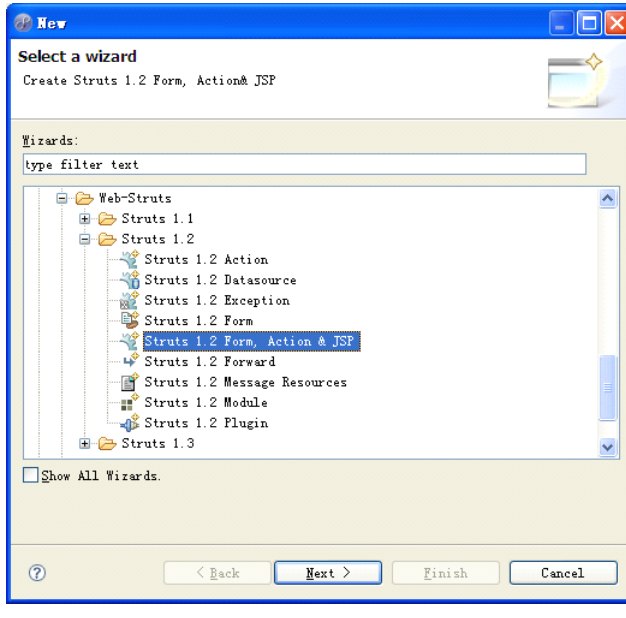


图 6-26　选择同时创建三个程序

单击“Next”按钮，将弹出创建 ActionForm 的界面，因为这里是表示论坛中的主题，所以为它起名为“TopicForm”，其他信息如图 6-27 所示。

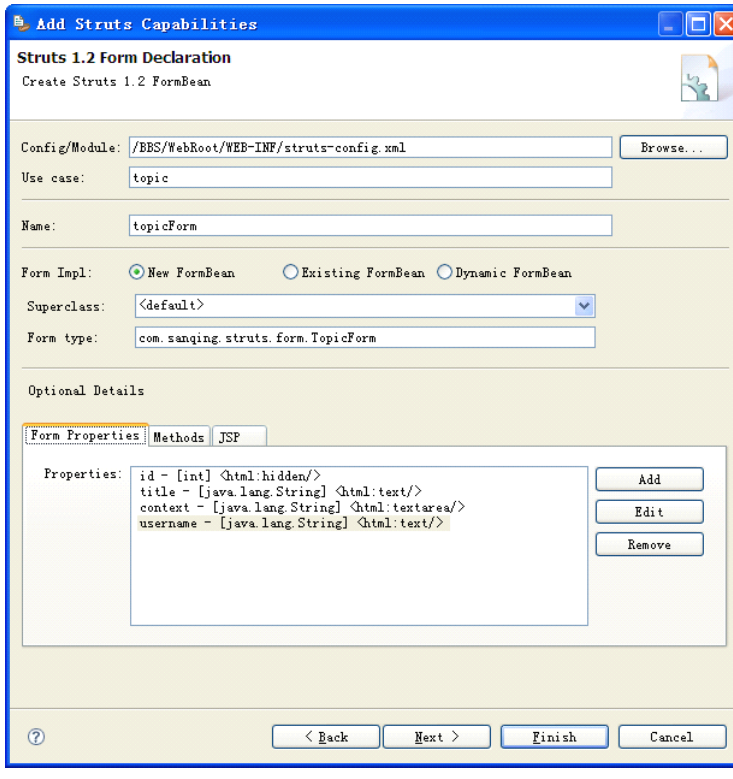


图 6-27　创建主题 ActionForm

其中“JSP”选项卡中的选择和填写信息如图 6-28 所示。

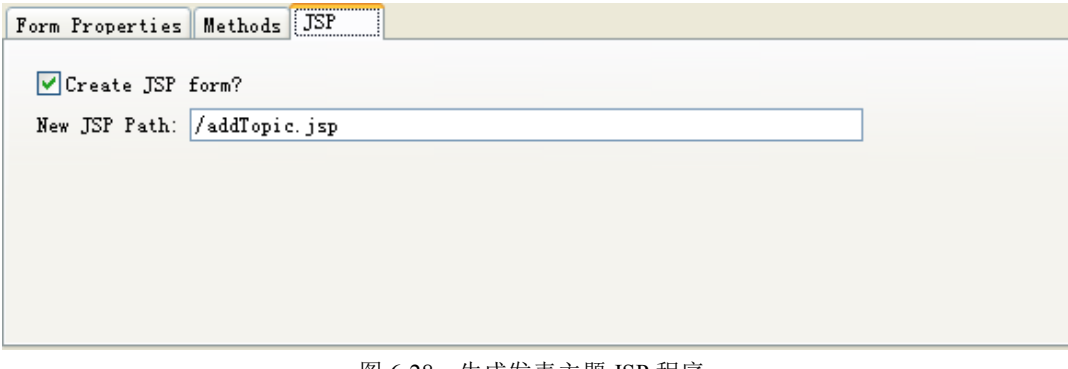


图 6-28　生成发表主题 JSP 程序

注意：主题 ActionForm 中是需要有 id 属性的，因为执行删除操作时会对主题 id 属性值进行接收。但是在发表主题页面中主题 id 是并不需要用户输入的，所以这里设置主题 id 对应的表单元素为隐藏域。

在创建主题 ActionForm 界面中，单击“Next”按钮，将完成主题 ActionForm 的创建，并进入到创建发表主题 Action 中，输入相应信息后，如图 6-29 所示。

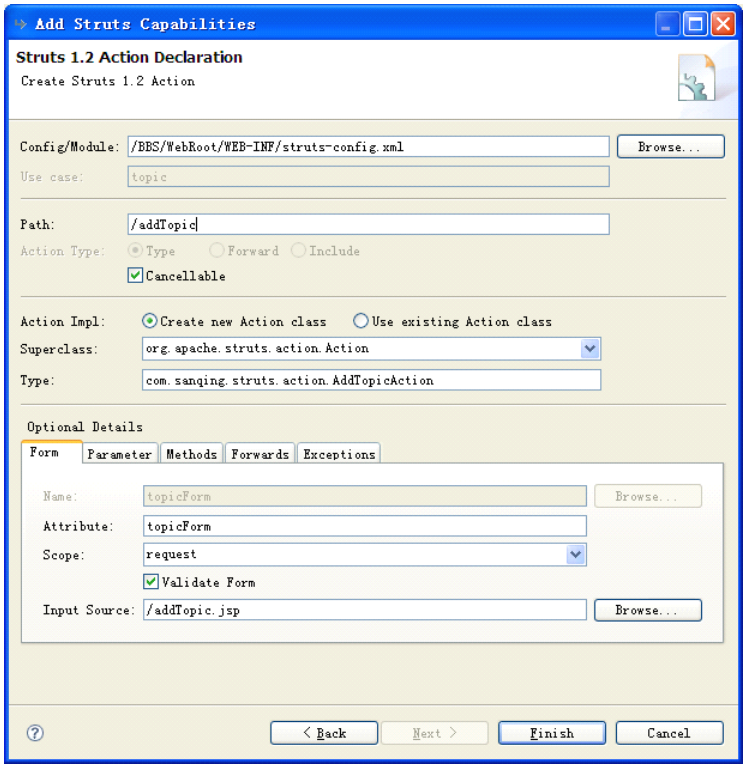


图 6-29　创建发表主题 Action

其中“Path”选项中进行修改，使用“/addTopic”更能表达 Action 的功能。在“Forwards”选项卡中还要加入成功和失败跳转，如图 6-30 所示。

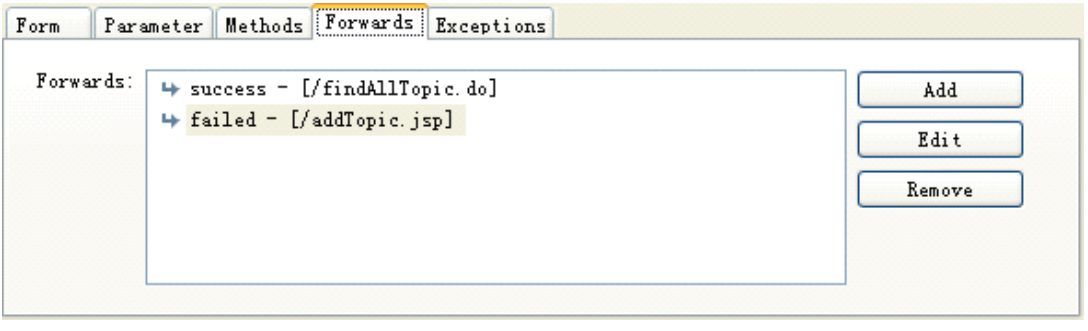


图 6-30　发表主题 Action 的跳转

当发表成功后，自动跳转到查询所有主题 Action 中，从而再执行查询所有主题的功能。当发表失败后，跳转回发表主题页面。

在创建发表主题 Action 中，单击“Finish”按钮，将完成程序的创建。打开发表主题 Action 程序，在其中加入操作数据访问层的代码，其代码内容为：

```
01　package com.sanqing.struts.action;
02　//省略导入接口和类的代码
03　public class TopicAction extends Action {
04　　public ActionForward execute(ActionMapping mapping, ActionForm form,
```

```
05                  HttpServletRequest request, HttpServletResponse response) {
06              TopicForm topicForm = (TopicForm) form;          //获取主题 ActionForm
07              Topic topic=new Topic();                         //创建主题对象并设置获取值
08              topic.setTitle(topicForm.getTitle());
09              topic.setContent(topicForm.getContext());
10              topic.setUsername(topicForm.getUsername());
11              topic.setDate(new Date());
12              try {
13                  TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
14                  topicDAO.addTopic(topic);        //执行发表主题方法
15                  return mapping.findForward("success");
16              } catch (Exception e) {
17                  e.printStackTrace();
18                  return mapping.findForward("failed");
19              }
20          }
21      }
```

在其中第 6 行中获取主题 ActionForm 对象，在第 7 行创建主题对象后，将 ActionForm 对象中的属性值设置到主题对象中。

注意：在主题 ActionForm 对象中并没有发表时间属性值，在第 11 行中使用当前日期设置到主题对象中。

在第 13 行中通过 DAO 工厂类获取 DAO 实现类对象，然后调用 addTopic 发表主题方法，如果发表成功，则返回“success”。如果发生异常，则发表失败，返回“failed”。

对 Action 的代码调整后，打开发表主题 JSP 程序，也就是“addTopic.jsp”程序。可以看到所有的信息都是英文书写的，将它们修改成中文，而且 id 是并不需要显示的。调整后的 JSP 重要代码为：

```
01  <html:form action="/addTopic">
02      <html:hidden property="id"/><html:errors property="id"/><br/>
03      发表用户: <html:text property="username"/>
04                              <html:errors property="username"/><br/>
05      主题标题: <html:text property="title"/><html:errors property="title"/><br/>
06      主题内容  <html:textarea property="context" cols="20" rows="5"/>
07                              <html:errors property="context"/><br/>
08      <html:submit value="发表"/><html:cancel value="重置"/>
09  </html:form>
```

到这里，我们的发表主题功能就开发完成了。

6.4.3 开发查询所有主题功能

当发表主题成功后，将自动跳转到查询所有主题 Action 中，在本小节中就来开发该 Action。因为完成查询所有主题的功能，并不需要接收参数，所以在 Action 中也并不需要使用到 ActionForm。

选中“action”子包，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，

将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Action”选项，单击“Next”按钮，将弹出创建 Action 程序的界面，输入相关信息后，如图 6-31 所示。

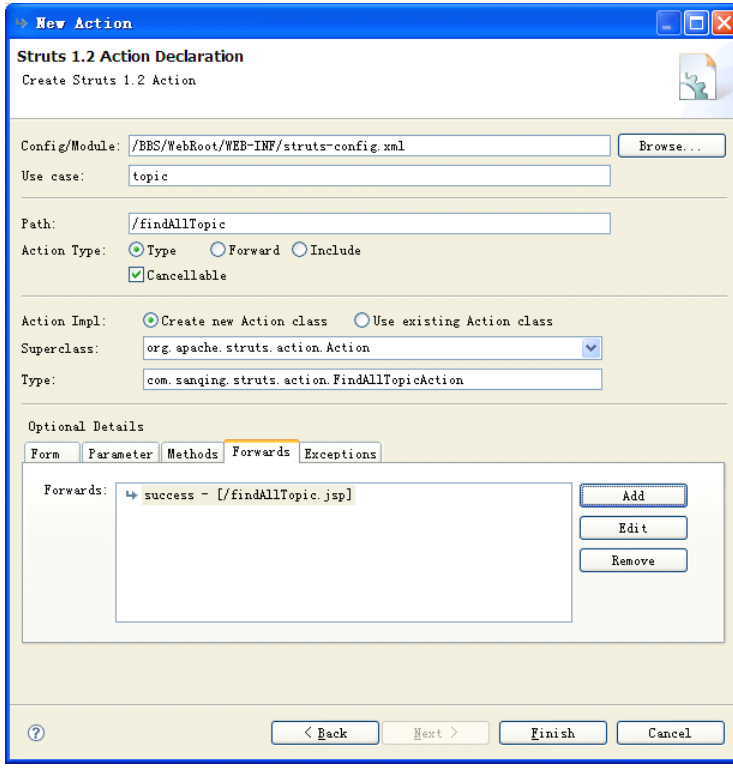


图 6-31　创建查询所有主题 Action

因为不需要使用任何 ActionForm，所以其中直接显示“Forwards”选项卡，在其中定义了当查询成功后，将跳转到显示所有主题的页面。单击“Finish”按钮，将完成该 Action 的创建。

打开查询所有主题的 Action 程序，在其中加入相应的功能代码，其调整后的代码内容为：

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class FindAllTopicAction extends Action {
04      public ActionForward execute(ActionMapping mapping, ActionForm form,
05              HttpServletRequest request, HttpServletResponse response) {
06          TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
07          List<Topic> topicList=topicDAO.findAllTopic();
08          request.setAttribute("topicList", topicList);
09          return mapping.findForward("success");
10      }
11  }
```

可以看到该程序是非常简单的，在第 8 行中将查询到的主题集合保存到 request 范

围内，这样在显示页面中就可以对它进行遍历，从而得到每一条主题记录。

接下来就是开发显示所有主题的 JSP 程序。选中“WebRoot”目录，单击鼠标右键，在弹出的菜单中选择“New”|“JSP（Avanced Templates）”命令，将弹出创建 JSP 程序的界面，输入相关信息后，如图 6-32 所示。

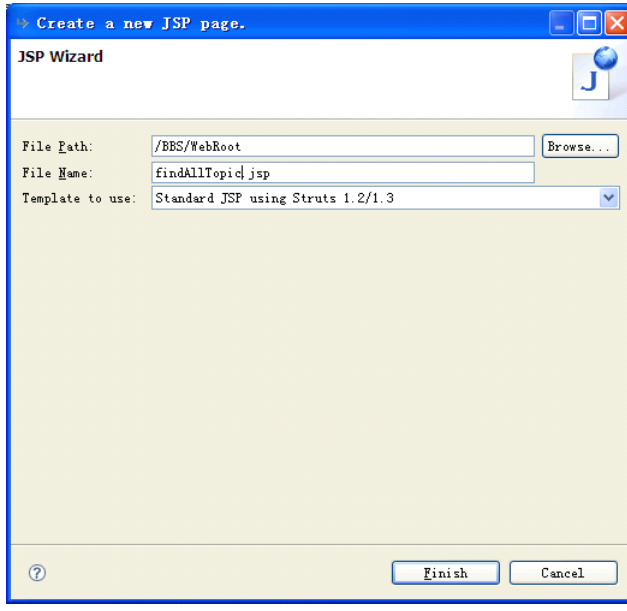


图 6-32　创建 JSP 程序

单击“Finish”按钮，将完成显示所有主题 JSP 程序的创建。在编辑区中自动打开生成的 JSP 程序，在其中加入相应代码后，其主要代码如下所示：

```
01  <table border="1">
02  <tr>
03      <td>主题标题</td>
04      <td>发表用户</td>
05      <td>发表时间</td>
06      <td>是否删除</td>
07  </tr>
08  <logic:empty name="topicList">
09      <tr>
10          <td colspan="3">数据库中还没有主题</td>
11      </tr>
12  </logic:empty>
13  <logic:notEmpty name="topicList">
14      <logic:iterate id="topic" name="topicList">
15          <tr>
16              <td>
17                  <bean:write name="topic" property="title"/>
18              </td>
19              <td>
20                  <bean:write name="topic" property="username"/>
```

```
21                      </td>
22                      <td>
23                          <bean:write name="topic" property="date"/>
24                      </td>
25                      <td>
26          <a href="deleteTopic.do?id=<bean:write name='topic' property='id'/>">删除</a>
27                      </td>
28                  </tr>
29          </logic:iterate>
30      </logic:notEmpty>
31      </table>
```

其中在第 8 行中使用使用<logic:empty>标签判断主题集合是否为空，如果为空，则显示目前没有主题。在第 14 行中使用<logic:iterate>标签对不为空的集合进行循环遍历，从而得到主题的具体信息，包括主题标题、发表用户和发表时间，主题内容通常是比较多的，并不会显示在这里。在第 26 行中定义了删除超链接，在其中以当前行主题 id 为参数调用删除主题 Action，从而将该行主题删除。

说明：在删除超链接的路径中可以看到，在其中也可以使用 Struts 的标签，将得到的值做为参数值，也和 EL 表达式是一样的。

6.4.4 开发删除主题功能

在显示所有主题页面中，单击“删除”超链接，就会将该行的主题记录删除。在本小节中就来完成删除主题的功能。

在删除主题 Action 中要使用主题 ActionForm 来接收传递的 id 属性值，主题 ActionForm 已经存在了，所以不需要再次创建。在选择创建程序界面中，选择“struts 1.2 Action”选项，单击“Next”按钮，将弹出创建 Action 程序的界面，输入删除 Action 的相关信息后，如图 6-33 所示。

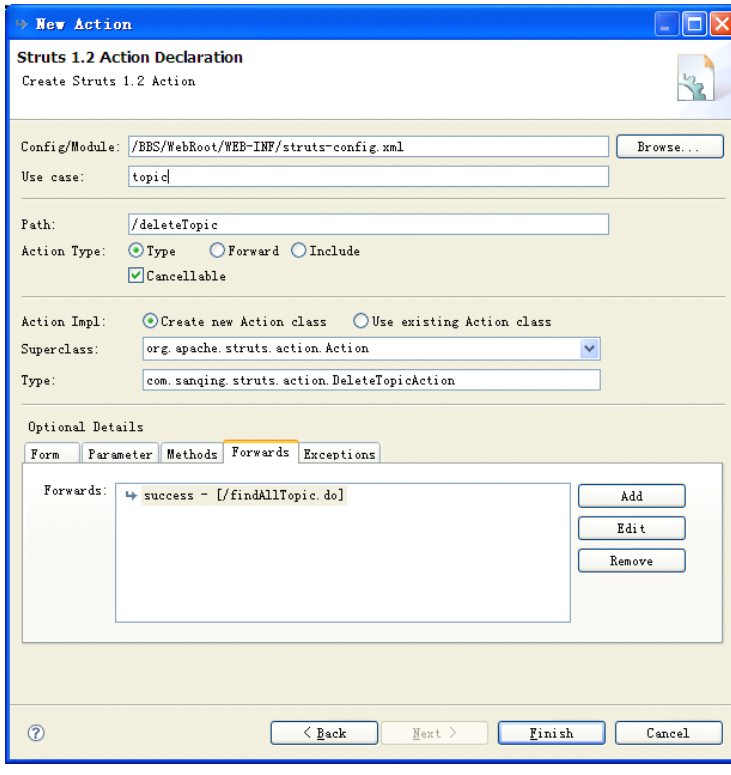


图 6-33　创建删除主题 Aciton

单击“Finish”按钮，将完成删除主题 Action 的创建。打开该程序，加入相应的功能代码，其程序代码为：

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class DeleteTopicAction extends Action {
04      public ActionForward execute(ActionMapping mapping, ActionForm form,
05              HttpServletRequest request, HttpServletResponse response) {
06          TopicForm topicForm=(TopicForm)form;
07          int id=topicForm.getId();                    //获取接收到的 id
08          TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
09          topicDAO.deleteTopic(id);                    //调用删除方法
10          return mapping.findForward("success");
11      }
12  }
```

在其中第 7 行中首先获取主题 id，然后以主题 id 为参数在第 9 行调用 deleteTopic 方法，从而将指定的主题删除。

说明：有些读者可能会认为为发表主题和删除主题分别定义 ActionForm 会更好，这是不对的，因为如果为每一个 Action 都定义一个 ActionForm，那就不再具有程序的重用性，同时也会造成程序过分膨胀。

6.4.5　运行论坛项目

经过上面的开发，该论坛项目已经能够发布并运行了，但是这样的项目是不能够处理中文提交的，要想让该项目支持中文，还需要将前面学习的编码过滤器拷贝到该项目中，然后在“web.xml”文件中进行配置。在“web.xml”文件中同时对项目的首页进行修改成“addTopic.jsp”。

发布 Struts 项目是和发布基本的 Web 项目相同的。在“Servers”界面中选择“Tomcat 6.x”服务器，单击鼠标右键，选择“Add Deployment”命令，将弹出发布项目界面，填入“BBS”项目后，如图 6-34 所示。

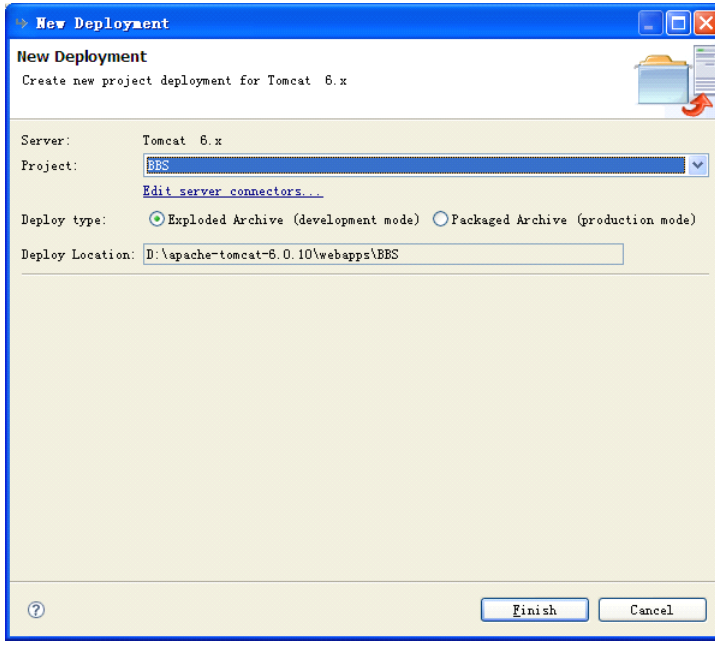


图 6-34　发布论坛项目

单击“Finish”按钮将完成项目的发布。再次选中“Tomcat 6.x”服务器，在右键弹出菜单中，选择“Run Server”命令，将开始运行服务器。启动服务器完成后，打开浏览器，输入如下地址：

http://localhost:8080/BBS/

将进入论坛项目首页，也就是发表主题页面，在其中发表一条主题，运行结果如图 6-35 所示。

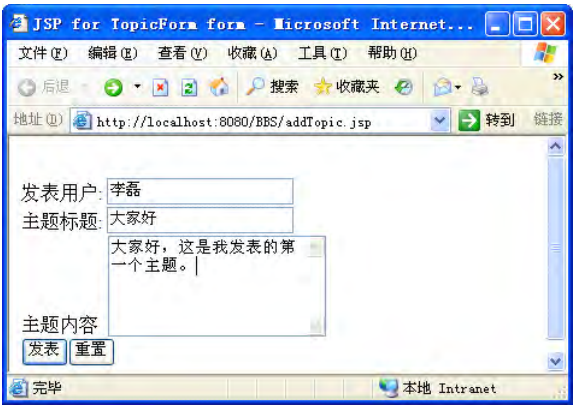


图 6-35　发表主题

单击“发表”按钮后会将这些填写的信息发送到发表主题 Action 中，发表完成后，将自动进入到查询所有主题 Action 中，然后跳转到显示所有主题页面，如图 6-36 所示。

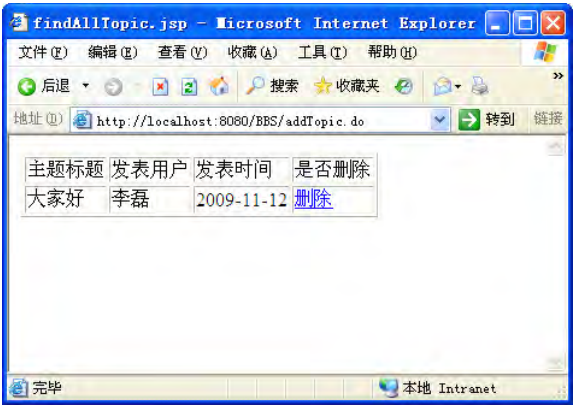


图 6-36　显示所有主题

在页面中，会将主题的标题、发表用户和发表时间显示出来。在每一条主题记录后面都有一个“删除”超链接，单击该超链接将把该条记录删除，删除后将再次跳转到显示所有主题的页面中，如图 6-37 所示。

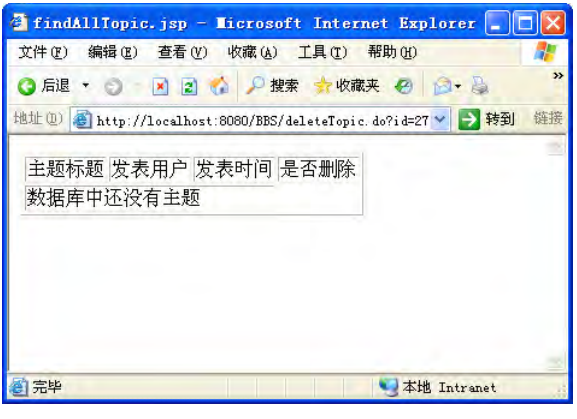


图 6-37　删除主题

因为目前我们只有一条主题，所以删除该主题后，将在页面中显示“数据库中还没有主题”的信息。

6.4.6　使用 DispatchAction 简化开发

在本节的论坛项目中，可以看到每完成一个功能，都要定义一个 Action，这对于一个功能复杂的项目来说，就要定义很多个 Action。不管是代码量上，还是配置文件中的配置都是非常多的。在 Struts 框架中，为了解决这个问题，定义了 DispatchAction 类，它是 Action 类的子类。当创建自己的功能 Action 时，可以继承 DispatchAction 类，从而将每一个功能定义一个方法，而不是一个 Action。

在 MyEclipse 中，也集成了创建这种 Action 的功能。在选择创建程序的界面中，选择“struts 1.2 Action”选项，单击“Next”按钮，将弹出创建 Action 的界面，在其中选择 Action 继承的父类为 DispatchAction，创建界面如图 6-38 所示。

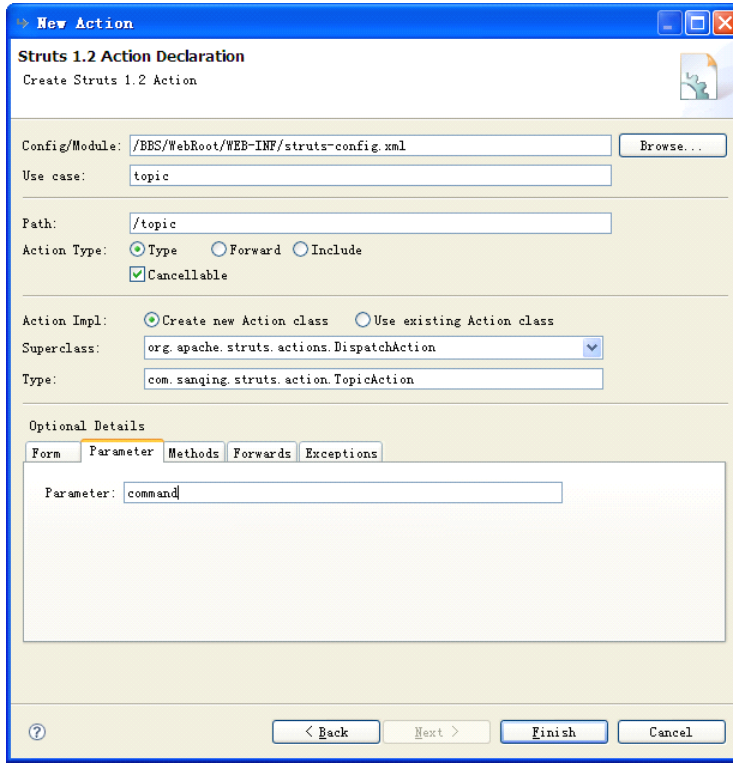


图 6-38　创建 DispatchAction

在其中“Superclass”选项中选择“org.apache.struts.actions. DispatchAction”，然后在下面的“Parameter”选项卡中填写访问主题 Action 时，使用到的参数。单击“Finish”按钮，将完成主体 Action 的创建。打开 Action 程序，加入相应的功能代码后，具体代码如下所示。

```
01    package com.sanqing.struts.action;
```

```
02  //省略导入接口和类的代码
03  public class TopicAction extends DispatchAction {
04      public ActionForward add(ActionMapping mapping, ActionForm form,
05              HttpServletRequest request, HttpServletResponse response) {
06          TopicForm topicForm = (TopicForm) form;          //获取主题 ActionForm
07          Topic topic=new Topic();                         //创建主题对象并设置获取值
08          topic.setTitle(topicForm.getTitle());
09          topic.setContent(topicForm.getContext());
10          topic.setUsername(topicForm.getUsername());
11          topic.setDate(new Date());
12          try {
13              TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
14              topicDAO.addTopic(topic);        //执行发表主题方法
15              return mapping.findForward("success");
16          } catch (Exception e) {
17              e.printStackTrace();
18              return mapping.findForward("failed");
19          }
20      }
21      public ActionForward delete(ActionMapping mapping, ActionForm form,
22              HttpServletRequest request, HttpServletResponse response) {
23          TopicForm topicForm = (TopicForm) form;
24          int id=topicForm.getId();                    //获取接收到的 id
25          TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
26          topicDAO.deleteTopic(id);                    //调用删除方法
27          return mapping.findForward("success");
28      }
29      public ActionForward findAll(ActionMapping mapping, ActionForm form,
30              HttpServletRequest request, HttpServletResponse response) {
31          TopicDAO topicDAO = TopicDAOFactory.createTopicDAOImpl();
32          List<Topic> topicList=topicDAO.findAllTopic();
33          request.setAttribute("topicList", topicList);
34          return mapping.findForward("success");
35      }
36  }
```

从代码上可以看到，我们将论坛项目中的发表主题、查询所有主题和删除主题的方法都写到了该 Action 中。当访问该 Action 时，需要给出相应的参数，例如要访问查询所有主题方法，则访问地址为：

http://localhost:8080/BBS/topic.do? command= findAll

其中“topic.do”就是对 Action 进行的路径，“command”是我们在界面中配置的 Action 参数，“findAll”指定调用的是哪一个方法。读者可以自己来将本节中开发的论坛项目修改成 DispatchAction 的形式。

说明：DispatchAction 在实际开发中使用是非常多的，但是也不是将所有的功能都写到一个 Action 中，而是将一类功能放在一起，例如对主题的操作功能就都写在

TopicAction 中，而对用户的操作就应该写在 UserAction 中。

第7章　进行Hibernate开发

在上一章中学习了 Struts 框架的开发，它是一种 MVC 框架。在本章中将学习到 Hibernate 框架，它是一种开发源代码的专注于数据访问层的对象关系映射框架，仅使用 Hibernate 框架是不能开发项目的。在 Hibernatez 中对 JDBC 进行轻量级的封装，将对象模型表示的数据映射到 SQL 表示的关系模型上，从而使得 Java 开发人员能够通过使用对象的变成思维来操作数据库。

在 MyEclipse 中，对 Hibernate 做了非常好的集成，通过界面化的操作能够让数据在 Java 程序和数据库之间进行转换。在本章中将把具体操作做为重点，从而让读者能够进行 Hibernate 相关程序的快速开发。

7.1　开发 Hibernate 项目

Hibernate 框架是一种专注于数据库操作的框架，所以它并不仅仅应用在 Java Web 的框架中，同时也可以应用在普通的 Java 项目中。本本节中，我们就以简单的 Java 项目来看一下在 MyEclipse 中如何开发 Hibernate 项目。

7.1.1　准备工作

因为 Hibernate 是一种数据库框架，所以在开发该项目的时候一定要有相应的数据库连接。在前面管理数据库一章已经讲解了，如何在数据库视图中创建数据库连接 。其中已经建立了“MySQL”的数据库连接，在该 Hibernate 项目中就要使用到该连接。如果读者的本地计算机中，不存在该连接，需要先建立后，才能够创建 Hibernate 连接 MySQL 数据库的项目。

在 MyEclipse 开发 Hibernate 项目是和开发 Struts 项目类似的，都需要首先创建一个基本项目，然后为基本项目加入相应支持。在 MyEclipse 的菜单中，选择“File”|“New”|“Java Project”命令，就会弹出创建 Java 项目的界面，在其中输入项目的名称为“FirstHibernate”，如图 7-1 所示。

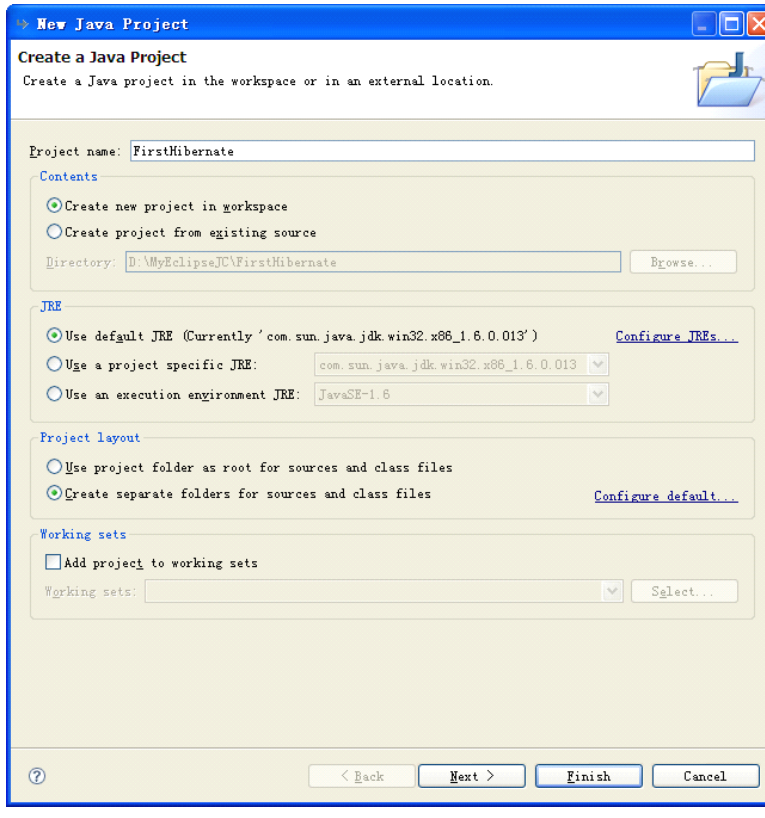


图 7-1　创建 Java 项目

单击“Finish”按钮将完成普通 Java 项目的创建。

7.1.2　加入 Hibernate 框架支持

在包资源管理器中，选中要加入 Hibernate 框架支持的项目，这里选择“FirstHibernate”项目。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Hibernate Capabilities”命令，将弹出加入支持操作界面，如图 7-2 所示。

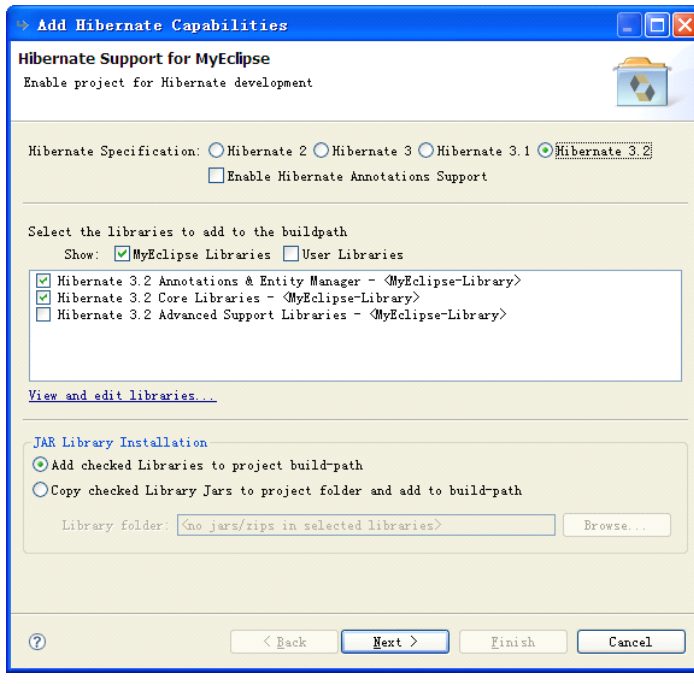


图 7-2　加入框架支持

其中“Hibernate Specification”表示要加入 Hibernate 框架的版本，默认是选中最高版本，通常也是这样的。“Enable Hibernate Annotations Support”表示是否让 Hibernate 支持注解方式，Hibernate 3 版本中多出了注解方式的开发方法，因为后面要进行该操作，所以这里选中它。

“Select the libraries to add to the buildpath”表示将向项目中加入哪些类库，对于基本的 Hibernate 项目而言，采用默认选择就可以。其中第 1 个选项就是注解开发所用到的类库。当 Hibernate 整合其他技术时，可能需要更多的类库。

“JAR Library Installation”表示是否将类库复制到项目中。其中“Add checked Libraries to project build-path”选项表示不复制 JAR 文件类库到项目中，只在发布项目时进行复制，这是默认的选项，也是经常采用的选项。“Copy checked Library Jars to project folder and to build-path”选项表示复制 JAR 文件类库到项目中，选择该选项后还需要填写将这些类库放置的目录，这种操作的好处是使开发的项目不依赖于 MyEclipse。

单击“Next”按钮，将完成 JAR 包的加入，并且弹出创建文件目录和配置文件的界面，如图 7-3 所示。

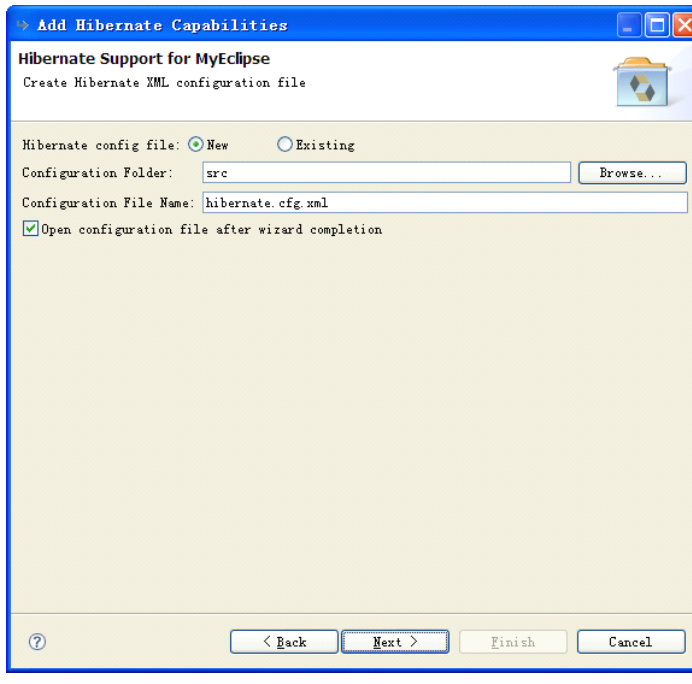


图 7-3　定义目录和配置文件

其中“Hibernate config file”表示配置文件的选择，“New”选项表示新创建，“Existing”选项表示使用已经存在的。“Configuration Folder”表示配置文件的所在目录，通常将它直接放在“src”目录下。“Configuration File Name”表示配置文件的名称，默认名称为“hibernate.cfg.xml”，通常也是采用该名称。“Open configuration file after wizard completion”表示创建完成后是否打开该文件，默认是打开的。

单击“Next”按钮，将完成 Hibernate 配置文件的创建，并进入建立数据库连接的界面，如图 7-4 所示。

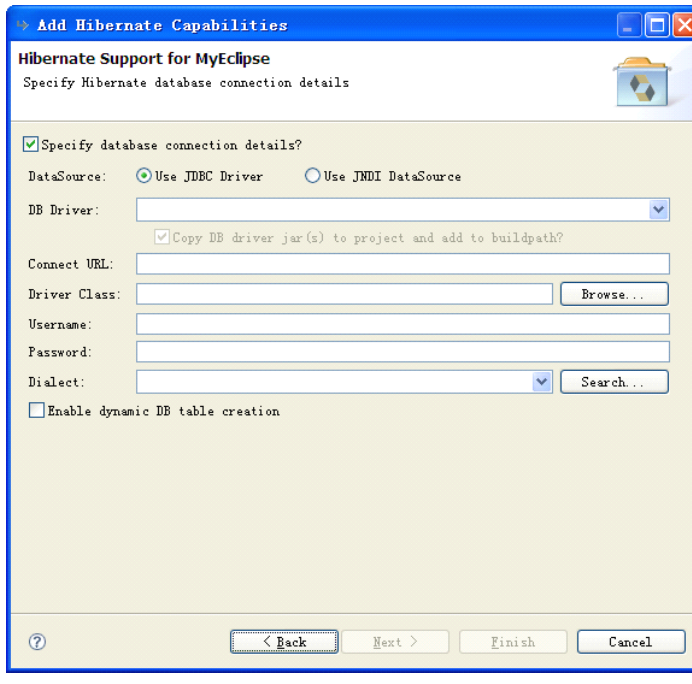


图 7-4　建立连接

其中“Specify database connection details”表示是否设置数据库连接属性，如果不选择，将跳过该界面的填写，通常是选择。“DataSource”表示数据源类型，“Use JDBC Driver”表示使用 JDBC 驱动，“Use JNDI DataSource”表示使用 JNDI 数据源，默认是使用 JDBC 驱动，这也是本书中讲解的。

“DB Driver”表示能够建立的连接，目前有两个选项，“MyEclipse Derby”表示 MyEclipse 中集成的数据库连接，“MySQL”是自己创建的，这里选择“MySQL”。选择数据库连接后，将会把下面的选项内容自动填写好。

注意：“DB Driver”选项是和数据库视图数据库浏览器中的已有数据库连接相对应的，如果其中没有“MySQL”连接，这里也就没法选择该项。

其中“Connect URL”表示连接数据库的 URL；“Driver Class”表示驱动类；“Username”和“Password”分别表示连接数据库用到的用户名和密码；“Dialect”表示数据库方言。最后的“Enable dynamic DB table creation”表示是否动态生成建表语句，默认是不选中的。

单击“Next”按钮，将完成数据库的连接，并进入 Hibernate 会话工厂设置界面，如图 7-5 所示。

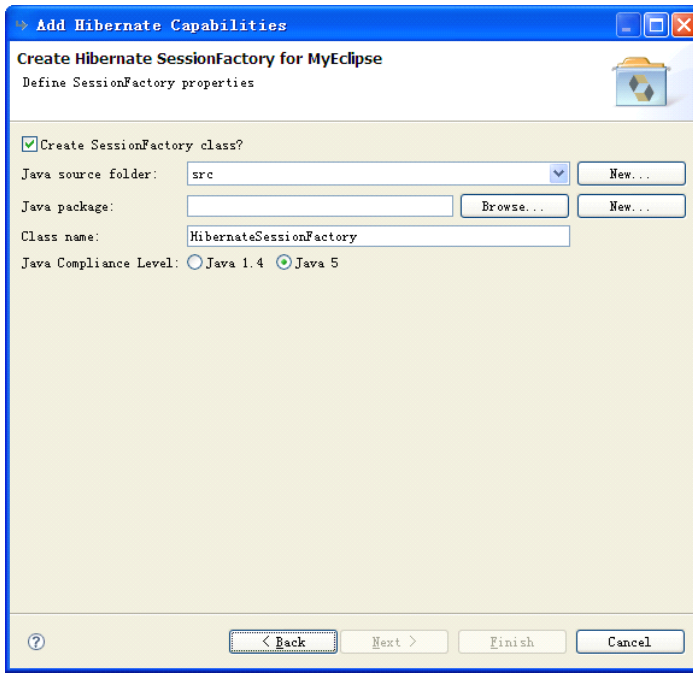


图 7-5　设置会话工厂

其中“Create SessionFactory class”表示是否创建会话工厂类，如果取消选中将跳过该界面的设置。“Java source folder”表示会话工厂类创建在哪一个目录中。“Java package”表示会话工厂类创建在哪一个包中，单击“Browse”按钮可以选择包，单击“New”按钮可以创建一个包。

说明：在实际开发中通常将它放在“hb”包中，所以这里我们创建“com.sanqing.hb”包，该包名并不是规范中所固定的。

“Class name”表示会话工厂类的类名，默认是“HibernateSessionFactory”，通常也是采用该默认名称。“Java Complliance Level”表示使用 Java 的版本，这里选择“Java 5”版本。

单击“Finish”按钮，将完成向项目中加入 Hibernate 框架支持的操作。

7.1.3　认识 Hibernate 项目结构

加入 Hibernate 框架支持后，在项目中会自动加入 Hibernate 的相关 JAR 包，以及 Hibernate 的配置文件和会话工厂类。在包资源管理器中将项目展开，如图 7-6 所示。

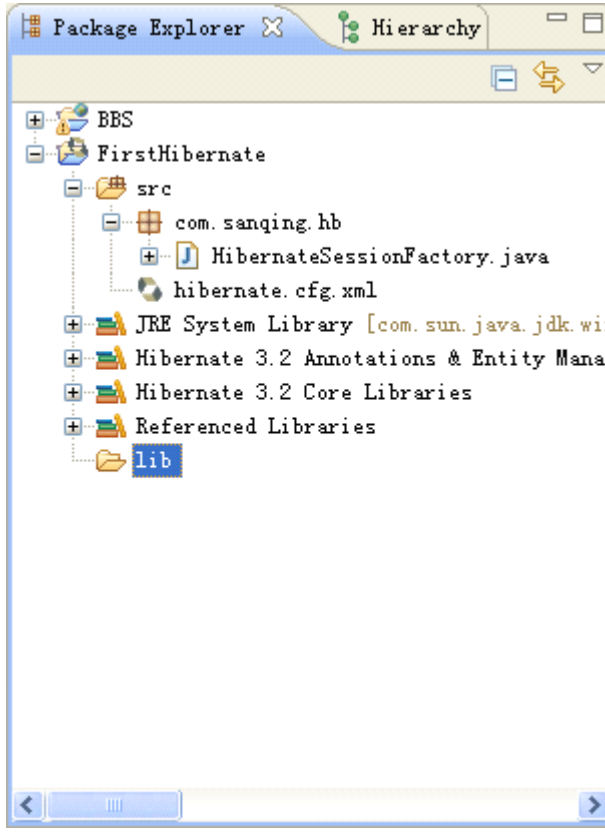


图 7-6　Hibernate 项目结构

其中在“Referenced Libraries”目录下包含着 MySQL 数据库的驱动包，该包来自建立连接时导入的包，当使用连接建立 Hibernate 项目后会自动导入。

在“com.sanqing.hb”包下会创建会话工厂类，该类中的代码都是自动生成的。通过会话工厂类可以很容易获取到 Session 对象。通过使用 Session 对象调用方法才能完成数据库的相关操作。

注意：这里所指的 Session 对象不是 JSP 中的 Session 对象，而是 Hibernate 中用来操作数据库的 Hibernate Session 对象。

为了更好的了解 Hibernate 中的数据库操作，我们需要对该会话工厂类有所了解，它的主要代码为：

```
01  public class HibernateSessionFactory {
02      private static String CONFIG_FILE_LOCATION = "/hibernate.cfg.xml";//配置文件
03      private static final ThreadLocal<Session> threadLocal
04                              = new ThreadLocal<Session>();//定义 ThreadLocal 对象
05      private   static Configuration configuration
06                              = new Configuration();//定义 Configuration 对象
07      private static org.hibernate.SessionFactory sessionFactory;//定义 SessionFactory 对象
08      private static String configFile = CONFIG_FILE_LOCATION;
09      static {
10          try {
```

```
11                  configuration.configure(configFile);//读取配置文件
12                  sessionFactory =
13                          configuration.buildSessionFactory();//根据配置文件创建
14              } catch (Exception e) {
15                  System.err
16                          .println("%%%% Error Creating SessionFactory %%%%");
17                  e.printStackTrace();
18              }
19          }
20          private HibernateSessionFactory() {
21          }
22          public static Session getSession() throws HibernateException {
23              Session session = (Session) threadLocal.get();//从 ThreadLocal 对象中获得
24              if (session == null || !session.isOpen()) {//判断 Session 对象是否为空者未打开
25                  if (sessionFactory == null) {       //如果没有创建，则首先创建
26                      rebuildSessionFactory();
27                  }
28                  //如果不为空，则调用其 openSession 方法创建 Session 对象
29                  session = (sessionFactory != null) ? sessionFactory.openSession(): null;
30                  threadLocal.set(session);//在 ThreadLocal 对象中保存该 Session 对象
31              }
32              return session;
33          }
34          public static void rebuildSessionFactory() {
35              try {
36                  configuration.configure(configFile);//读取配置文件
37                  sessionFactory =
38                          configuration.buildSessionFactory();//根据配置文件创建
39              } catch (Exception e) {
40                  System.err
41                          .println("%%%% Error Creating SessionFactory %%%%");
42                  e.printStackTrace();
43              }
44          }
45          public static void closeSession() throws HibernateException {
46              Session session = (Session) threadLocal.get();//从 ThreadLocal 对象中获得
47              threadLocal.set(null);//将当前线程 Session 对象从 ThreadLocal 对象中移除
48              if (session != null) {
49                  session.close();
50              }
51          }
52          public static org.hibernate.SessionFactory getSessionFactory() {//取得 SessionFactory
53              return sessionFactory;
54          }
55          public static void setConfigFile(String configFile) {//设置新的配置文件
```

```
56                  HibernateSessionFactory.configFile = configFile;
57                  sessionFactory = null;
58              }
59              public static Configuration getConfiguration() {//获得 Configuration 对象
60                  return configuration;
61              }
62      }
```

代码第 3 行定义了一个 ThreadLocal 对象，该对象可以用来保存不同线程下的 Session 对象。ThreadLocal 对象的内容结构就是一个 Map 对象，其 key 值用来保存线程 ID，value 值用来保存 Session 对象。这样就能保证每个线程有不同的 Session 对象，同时还可以保证了每一个线程只拥有一个 Session 对象实例。

代码第 9 行至第 19 行定义一个静态代码段，在该代码段中首先通过 configuration 对象读取配置文件，然后调用其 buildSessionFactory 方法来创建 SessionFactory 对象。代码第 22 行定义了一个 getSession 方法，在该方法中首先调用 ThreadLocal 对象来获得当前线程的 Session 对象，然后判断是否存在该 Session 对象。如果不存在该 Session 对象，或者该 Session 对象未打开，就调用 SessionFactory 对象的 openSession 方法创建 Session 对象。创建完 Session 对象后，还需要在 ThreadLocal 对象中保存该 Session 对象，并最终返回该 Session 对象。

代码第 45 行定义了一个 closeSession 方法，该方法首先从 ThreadLocal 对象中获得当前线程对象的 Session 对象，然后将当前线程 Session 对象从 ThreadLocal 对象中移除。如果该 Session 对象不为 null，则调用其 close 方法将其关闭。

说明：从代码量和讲解上可以看到，会话工厂类并不简单的，但是值得庆幸的是该类会由 MyEclipse 自动创建，不需要自己手动编写。这里只需要对它有所了解，它们会使用会话工厂类就可以了。

7.1.4 认识 Hibernate 的配置文件

Hibernate 配置文件也是 Hibernate 项目的重要组成部分，由于它的内容比较多，将它单独放在一小节中进行讲解。双击“hibernate.cfg.xml”文件在编辑打开后，在编辑区的左下角可以看到有三个选项，它们对应着三种显示文件的方式。“Configuration”选项表示界面化的显示 Hibernate 配置信息，选择该选项后，界面如图 7-7 所示。

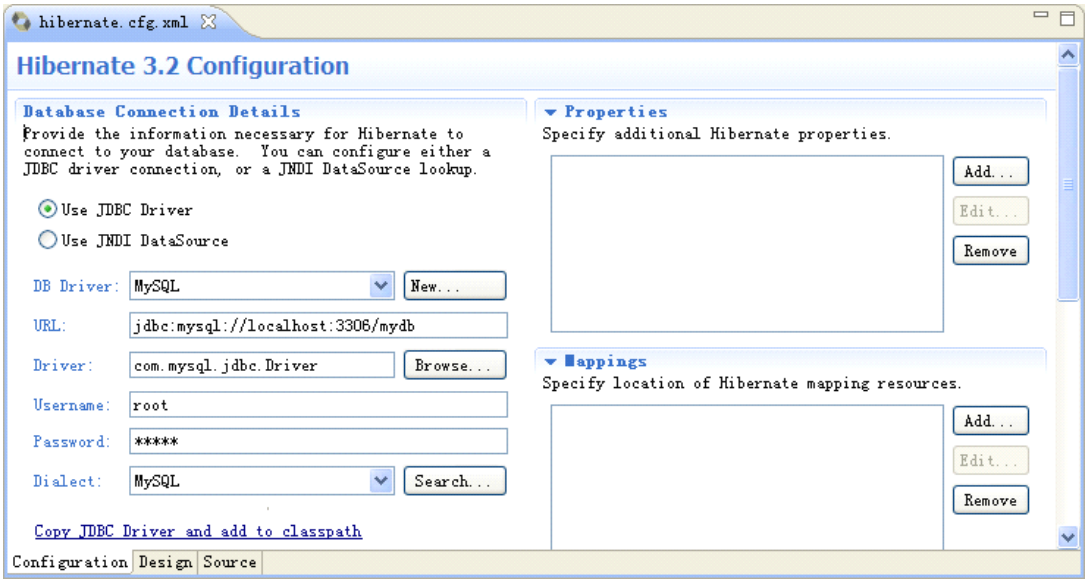


图 7-7　界面化显示配置信息

在界面的左边是对数据源进行的配置，和前面加入 Hibernate 框架支持操作中的类似的。在其中配置了数据库连接、URL、驱动类、用户名、密码和数据库方言等信息。

除了这些信息后，还可以在右边“properties”选项中添加其他配置信息，例如这里添加显示 SQL 语句的配置。单击“Add”按钮，将弹出添加属性配置界面，在其中填写相应信息，如图 7-8 所示。

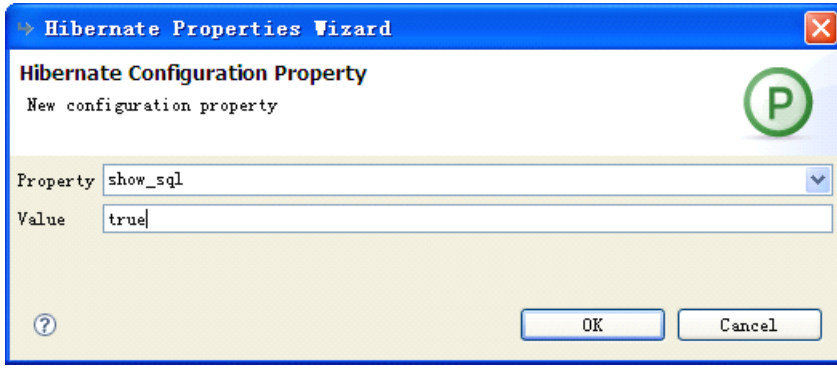


图 7-8　添加属性配置

说明：在项目开发阶段，显示 SQL 语句的配置是经常被使用到的。在 Hibernate 中执行操作的并不是 SQL 语句，但是在运行项目时想看到具体的数据库操作，这时候就可以进行该配置，从而将 Hibernate 框架中执行的 SQL 语句显示在控制台中。

在“Mappings”选项中是对实体类映射文件进行配置，在配置在后面的讲解中将详细介绍。

编辑区的左下角的第二个选项是“Design”选项，选择该选项，则会将配置文件中的配置以 XML 文件结构的形式显示出来，如图 7-9 所示。

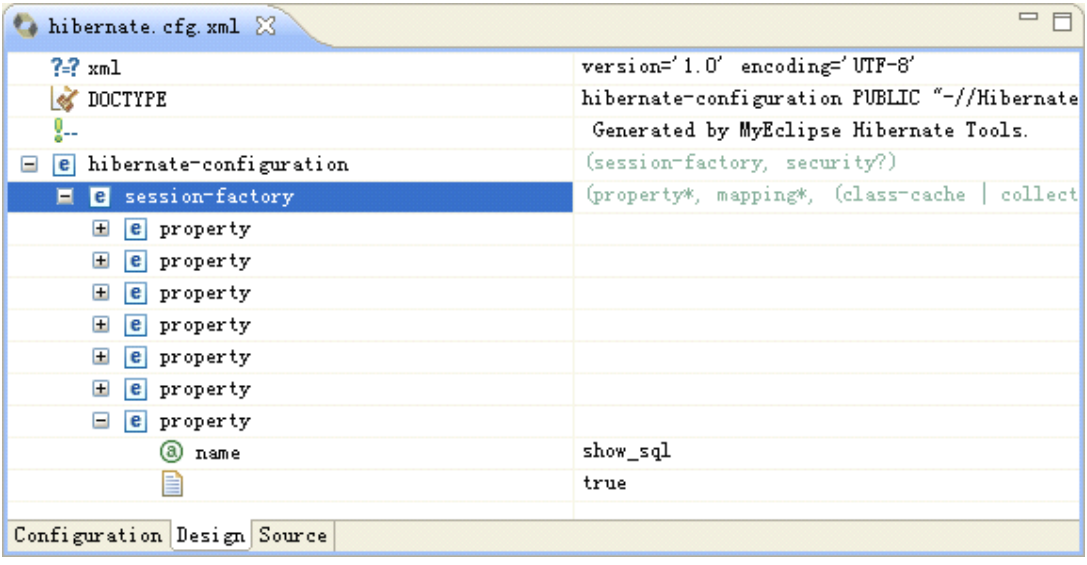


图 7-9　XML 文件形式显示

其中每一个“property”节点就对应着一条属性配置。在“property”节点下又分为两个节点，分别对应着属性名和属性值。例如显示 SQL 语句的配置，属性名为“show_sql”，属性值为“true”，表示在控制台中显示 SQL 语句。如果设置值为“false”，则表示不显示，它也是默认值。

编辑区的第三个“Source”选项就是对应着以代码的形式显示 Hibernate 的配置文件，其中的重要代码为：

```
01  <hibernate-configuration>
02  <session-factory>
03      <property name="dialect">
04          org.hibernate.dialect.MySQLDialect          <!-- 数据库方言 -->
05      </property>
06      <property name="connection.url">
07          jdbc:mysql://localhost:3306/mydb          <!-- 数据库连接 URL -->
08      </property>
09      <property name="connection.username">root</property><!-- 数据库用户名 -->
10      <property name="connection.password">admin</property><!-- 数据库密码 -->
11      <property name="connection.driver_class">
12          com.mysql.jdbc.Driver                     <!-- 数据库驱动类 -->
13      </property>
14      <property name="show_sql">true</property>          <!--显示 SQL 语句-->
15  </session-factory>
16  </hibernate-configuration>
```

Hibernate 配置文件的根节点为<hibernate-configuration>，Hibernate 的所有配置都必须放置在该节点下。代码第 2 行定义了一个<session-factory>标签，可以在该标签下进行数据源等相关属性的配置。代码第 3 行定义了一个<property>标签，该标签用来指定 Hibernate 中需要配置的属性和属性值。

其中 dialect 属性用来设置数据库方言，这里使用的是 MySQL 数据库，它的值为 org.hibernate.dialect.MySQLDialect。后面依次对连接 URL、用户名、密码、驱动类等属性进行了配置。show_sql 属性是我们自己添加的，用来设置是否显示 SQL 语句，当值为 true，表示显示 SQL 语句，否则不显示。

说明：Hibernate 中的数据库方言是非常重要的，因为不同数据库的数据库操作是有所不同的。在 Hibernate 框架中通过数据库方言，来根据不同数据库进行不同的操作，从而使程序员不用关心数据库，而只需要关心操作代码。

7.2 开发 Hibernate 项目中的程序

开发完 Hibernate 项目后，就可以在该项目中创建 Hibernate 的相关程序。Hibernate 程序包括实体类、映射文件和 DAO 数据访问类，在 MyEclipse 中集成了界面化创建这些程序的功能。

7.2.1 创建数据表

界面化创建 Hibernate 程序都是基于数据表操作的，通过数据表可以生成对该表进行操作的相关类。所以我们在开发 Hibernate 程序之前，首先要在“mydb”数据库下创建一个数据表，这里以一个学生表举例。

说明：数据库相关的操作已经在管理数据库一章进行了详细讲解，在本节中只对重要的步骤进行讲解。

在 MyEclipse 中，首先要切换到数据库操作视图。如果前面创建的“MySQL”已经被删除，则需要首先创建该连接。在数据库浏览器中，选中“MySQL”连接，单击鼠标右键，在弹出的菜单中选择“Open connection”命令，将打开该连接。

打开连接后，选中“mydb”数据库节点，单击鼠标右键，在弹出菜单中选择“New Table”命令，将弹出创建数据表的界面，在其中填写必要信息后，界面如图 7-10 所示。

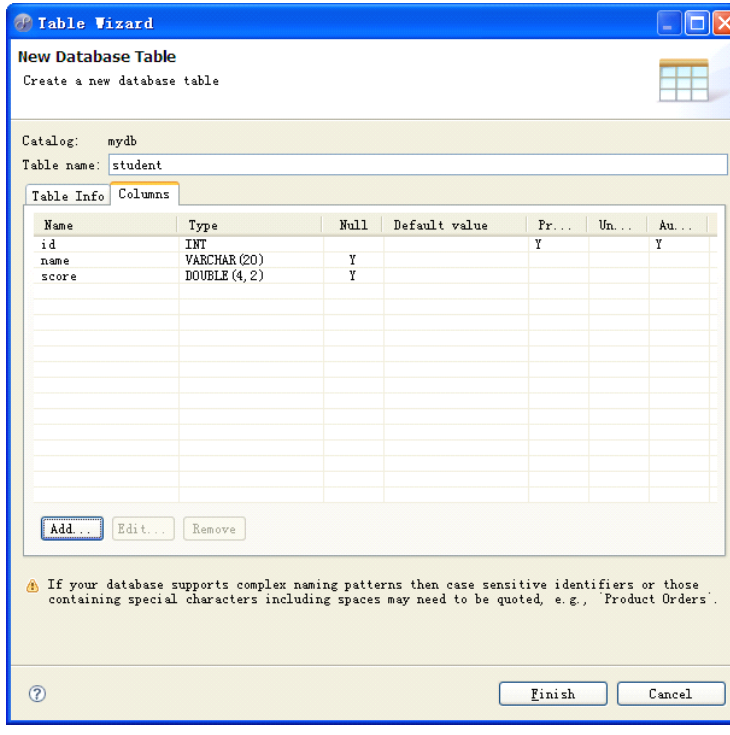


图 7-10　创建学生表

单击“Finish”按钮，将完成学生数据表的创建。在“mydb”|“TABLE”节点下将多出一个学生表节点。

技巧：如果创建数据表没有发生错误，但是没有在表格节点下显示，可以选择“TABLE”节点，在右键弹出菜单中选择“Refresh”命令，执行刷新操作，从而显示新创建的数据表。

7.2.2　创建映射文件程序

创建完成学生表后，我们就可以继续开发操作该数据表的 Hibernate 程序。选中“student”数据表节点，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，如图 7-10 所示。

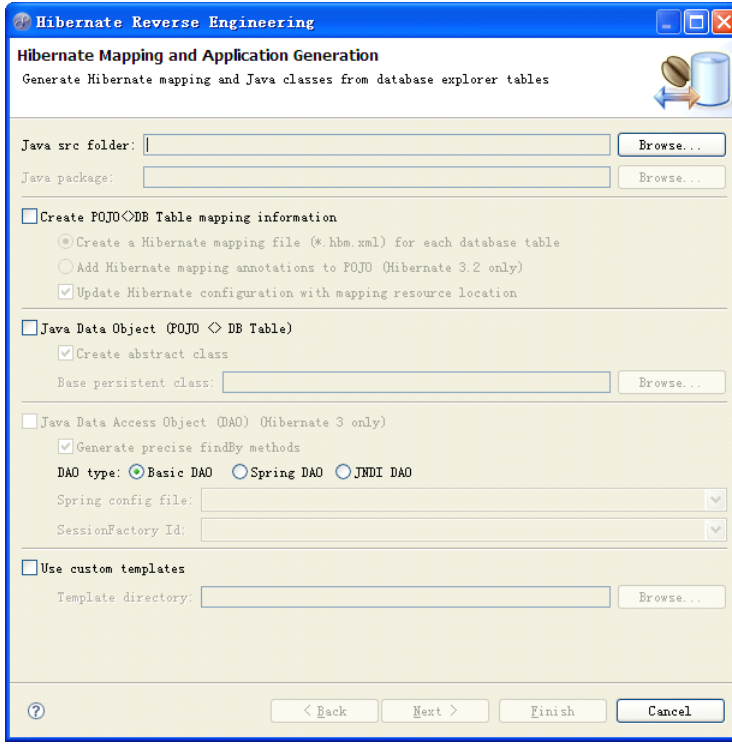


图 7-10　创建 Hibernate 程序界面

其中“Java src folder”表示选项创建的 Hibernate 程序放在哪一个项目的“src”目录下，单击“Browse”可以选择，这里选择“FirstHibernate”|“src”目录。“Java package”表示创建程序所在的包名，如果在操作前已经创建了包，则可以单击“Browse”按钮选择；如果没有创建，则可以自己输入包名，当创建程序时会将该包同时创建。这里输入“com.sanqing.po”，在 Hibernate 项目中通常将实体类和映射文件都放在“po”包下。

在接下来的界面中，可以分为三部分，它们分别创建映射文件、实体类和 DAO 类，它们可以通过该界面同时创建，这里为了更详细的讲解，我们将它们分开创建。

“Create POJO<>DB Table mapping information”表示是否创建映射文件，选择该选项，表示创建。在该选项下，还有具体设置的选项。“Create a Hibernate mapping file (*.hbm.xml) for each database table”选项表示为每一个数据表创建一个以“.hbm.xml”为结尾的映射文件；“Add hibernate mapping annotations to POJO (Hibernate 3.2 only)”选项表示通过注解的方式创建数据表映射，它只能在 3.2 版本的 Hibernate 框架中使用。“Update Hibernate configuration with mapping resource location”表示是否在 Hibernate 的配置文件中对映射文件进行配置，默认是选择的，这样就不再需要自己手动配置。

后面创建其他 Hibernate 程序的选项先不进行选择，单击“Finish”按钮，将完成映射文件的创建，并且弹出一个询问是否进入 Hibernate 操作视图的对话框。

说明：Hibernate 操作视图是 MyEclipse 中为了开发 Hibernte 项目和程序定义的视图，由于习惯问题，该视图使用的并不多的。这里同样不使用该视图，而是切换到最常用的 MyEclipse 通用开发视图中。

切换到 MyEclipse 通用视图中后，可以在包资源管理器中看到，在“FirstHibernate”|“src”目录下创建了“com.sanqing.po”包，并且在该包下创建了一个名称为“student.hbm.xml”的文件，它就是和数据表对应的映射文件。双击打开该文件，在编辑区中有两种查看方法，选择左下角的“Design”选项，将以界面化的形式显示映射信息，如图 7-11 所示。

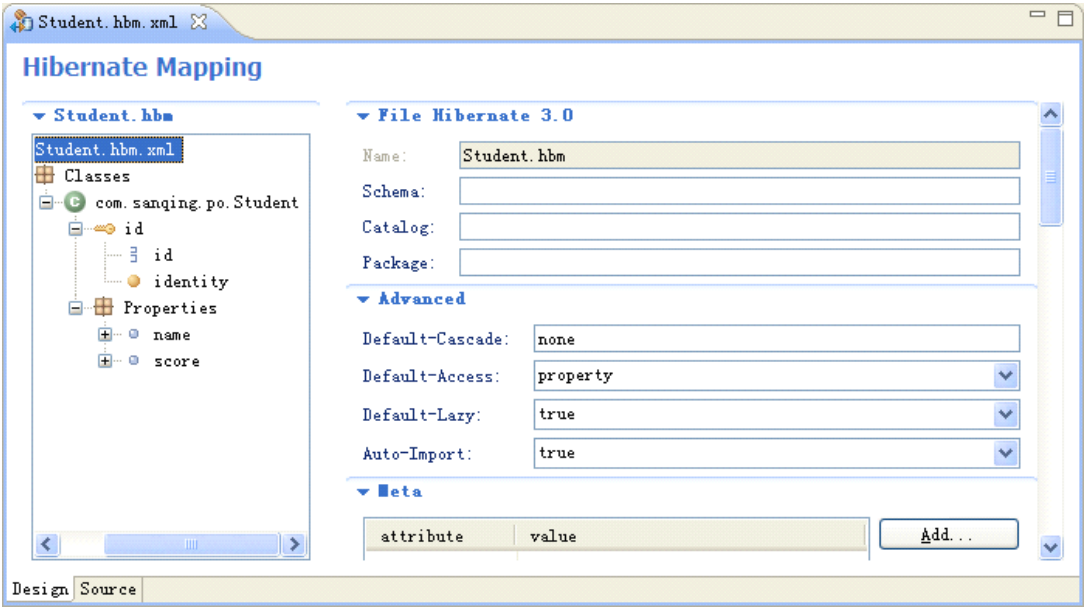


图 7-11　界面化查看映射文件

当在映射文件增加更多配置信息时，可以通过该界面进行添加。这里我们主要看一下映射文件的代码，选择编辑区左下角的“Source”选项，将可以看到该程序的代码。

```
01  <hibernate-mapping>
02      <class name="com.sanqing.po.Student" table="student" catalog="mydb">
03          <id name="id" type="java.lang.Integer">
04              <column name="id" />
05              <generator class="identity" />
06          </id>
07          <property name="name" type="java.lang.String">
08              <column name="name" length="20" />
09          </property>
10          <property name="score" type="java.lang.Double">
11              <column name="score" precision="4" />
12          </property>
13      </class>
14  </hibernate-mapping>
```

映射文件的根节点为<hibernate-mapping>，所有的配置代码都必须放置在该节点之下。在第 2 行定义了一个<class>节点，通过该节点可以让实体类和数据表建立映射关系，其中 table 属性指定的就是前面创建的学生数据表，catalog 属性指定的是数据库。name 属性指定的就是实体类名称，实体类会在下一小节中讲解如何创建。

在第 3 行定义了一个<id>节点，其中 name 属性指定主键为实体类中的 id 属性，type 属性指定的是 id 的类型。<column>节点指定的是数据表中的主键字段名，<generator>节点用来指定主键的生成策略，生成策略的知识会在后面进行讲解。代码第 6 行至第 8 行通过<property>节点配置了学生表中的其他字段，分别是姓名和成绩字段。

在创建映射文件界面中，如果选择了“Update Hibernate configuration with mapping resource location”选项，还会在“Hibernate.cfg.xml”配置文件中对映射文件进行配置，配置代码为：

```
<mapping resource="com/sanqing/po/Student.hbm.xml" />
```

每有一个映射文件，就应该多出一条这样的配置。

注意：对映射文件的配置是可以自动生成的，如果是手动加入该配置一定要注意其中的“resource”属性值的格式，包名之间不再使用“.”连接，而是使用“/”连接。

7.2.3 创建实体类程序

创建映射文件后，继续来看一下如何创建实体类程序。使用上一小节同样的操作方式，打开创建 Hibernate 程序的界面，在其中“Java src folder”选项中选择“FirstHibernate/src”目录，在“Java package”选项中填写“com.sanqing.po”。

创建 Hibernate 程序的界面中，“Java Data Object (POJO <> DB Table)”选项表示是否创建和数据表相对应的实体类，选中该选项，表示创建实体类。在该选项中，还有更详细的创建设置选项。“Create abstract class”表示是否创建抽象类，通常取消该选项。“Base persistent class”表示基本的持久化类，在普通的 Hibernate 项目中也是不需要选择该类的。选择后的界面如图 7-12 所示。

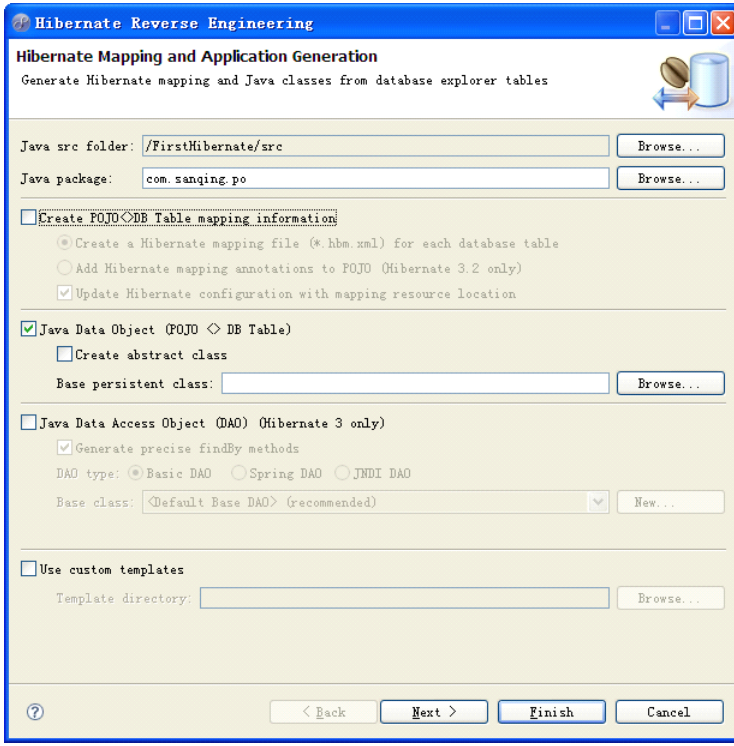


图 7-12　创建实体类

单击“Finish”按钮，将完成实体类的创建，并且将弹出一个对话框，如图 7-13 所示。

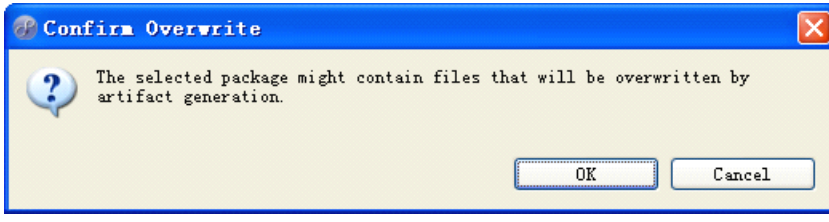


图 7-13　确认是否创建对话框

注意：弹出该对话框是因为在选择的包中已经创建了映射文件，而映射文件是和实体类配对出现的，所以这时候会提醒可能覆盖原有程序。在实际开发中通常将这两个程序同时创建，也就不会出现该对话框。

在其中单击“OK”按钮，跳过该对话框。还会弹出选择是否更换操作视图的对话框，这里单击“NO”按钮，然后手动切换到 MyEclipse 通用视图中。

在包资源管理中可以看到，在“com.sanqing.po”包下自动生成了名称为“Student”的 Java 程序，打开该程序后，它的代码如下：

```
01  package com.sanqing.po;
02  public class Student implements java.io.Serializable {
03      private Integer id;
04      private String name;
05      private Double score;
06      public Student() {
07      }
08      public Student(String name, Double score) {
09          this.name = name;
10          this.score = score;
11      }
12      public Integer getId() {
13          return this.id;
14      }
15      public void setId(Integer id) {
16          this.id = id;
17      }
18      public String getName() {
19          return this.name;
20      }
21      public void setName(String name) {
22          this.name = name;
23      }
24      public Double getScore() {
```

```
25              return this.score;
26          }
27          public void setScore(Double score) {
28              this.score = score;
29          }
30      }
```

其中从第 3 行到第 5 行生成了 id、姓名和成绩三个属性，它们都是根据数据表中的字段自动生成的。在第 8 行中使用非主键属性生成了构造函数。从第 12 行开始为所有的属性生成了 Getter 和 Setter 方法，这些都是自动生成的。

说明：在自动生成的实体类中都是使用的基本数据类型的封装类类型，这和我们手动完成有所不同的。但是在功能和调用上它们是没有差别的，因为在 Java 5.0 版本后有自动装箱和拆箱的特性。

7.2.4 创建 DAO 程序

DAO 程序中通常不只一个 Java 类程序，在 DAO 程序完成对数据库的增删改查等基本操作。在创建 Hibernate 创建程序中，同时也能够完成 DAO 程序的创建。

在创建 Hibernate 程序界面中，创建 DAO 程序时，在其中“Java src folder”选项中仍然选择“FirstHibernate”|“src”目录，而在“Java package”选项中，也就是选择包时，通常将 DAO 程序放在“dao”包中，所以这里填写“com.sanqing.dao”。

然后看一下创建 DAO 程序时所用到的选项。“Java Data Access Object (DAO)（Hibernate 3 only）”表示是否创建 DAO 程序，选中表示创建。在该选项下，还有进行更详细设置的选项。其中“Generate precise findBy methods”表示是否为每一个属性生成“findBy”方法，默认是选中的情况。如果选中，例如在学生表中，就会产生两个分别根据姓名和成绩属性查询学生的方法，具体看后面的代码。

“DAO type”表示生成 DAO 程序的类型，“Basic DAO”就是最基本的 DAO 程序，也是这里要创建的。“Spring DAO”也是比较常用的，但是只有 Hibernate 和 Spring 整合时才可以创建它。“JNDI DAO”使用的比较少，在 EJB 开发中可能会用到它。“Base class”表示基础 DAO 类，这里采用默认的类就可以。填写好这些信息后，界面如图 7-14 所示。

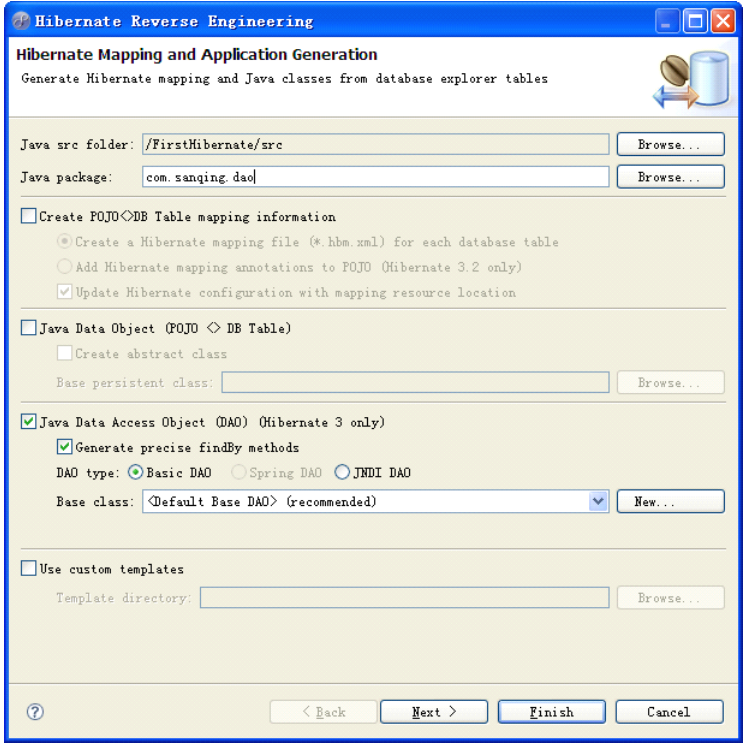


图 7-14　创建 DAO 程序

单击“Finish”按钮，将完成 Hibernate DAO 程序的创建。回到 MyEclipse 通用视图中，在包资源管理器中可以看到在“FirstHibernate”项目中创建了“com.sanqing.dao”包，并且在该包下创建了三个程序。首先双击打开“IBaseHibernateDAO.java”程序，它的代码为：

```
01  package com.sanqing.dao;
02  import org.hibernate.Session;
03  public interface IBaseHibernateDAO {
04      public Session getSession();
05  }
```

这是一个非常简单的接口，在其中定义了获取 Session 会话的方法。再来双击打开“BaseHibernateDAO.java”程序，它的代码为：

```
01  package com.sanqing.dao;
02  import com.sanqing.hb.HibernateSessionFactory;
03  import org.hibernate.Session;
04  public class BaseHibernateDAO implements IBaseHibernateDAO {
05      public Session getSession() {
06          return HibernateSessionFactory.getSession();
07      }
08  }
```

在这个类中首先实现了上面的接口，在第 5 行中实现了获取 Session 会话的方法。在第 6 行中使用到了 HibernateSessionFactory 类，它就是在创建 Hibernate 项目时自动创建的会话工厂类，在该类中定义了 getSession 获取 Session 对象的方法，从而获取到操

作数据库的 Session 对象。

最后来看一下实际操作数据库的“StudentDAO.java”程序，它的代码是由完成不同数据库操作的方法组成，这里我们只给出其中的常用方法。

注意：初始创建 DAO 程序后，程序中是由错误的，这是因为我们的 DAO 程序和实体类并不在一个包中，这时候首先要将实体类导入到 DAO 程序中，快捷键为“Ctrl+Shift+O”。

```
01  package com.sanqing.dao;
02  //省略导入接口和类的代码
03  public class StudentDAO extends BaseHibernateDAO {
04      private static final Log log = LogFactory.getLog(StudentDAO.class);
05      public static final String NAME = "name";
06      public static final String SCORE = "score";
07      public void save(Student transientInstance) {//增加学生方法
08          log.debug("saving Student instance");
09          try {
10              getSession().save(transientInstance);//调用 save 保存方法
11              log.debug("save successful");
12          } catch (RuntimeException re) {
13              log.error("save failed", re);
14              throw re;
15          }
16      }
17      public void delete(Student persistentInstance) {//删除学生方法
18          log.debug("deleting Student instance");
19          try {
20              getSession().delete(persistentInstance);//调用 delete 删除方法
21              log.debug("delete successful");
22          } catch (RuntimeException re) {
23              log.error("delete failed", re);
24              throw re;
25          }
26      }
27      public Student findById(java.lang.Integer id) {     //根据 id 主键查询
28          log.debug("getting Student instance with id: " + id);
29          try {
30              Student instance = (Student) getSession().get(
31                      "com.sanqing.dao.Student", id);
32              return instance;
33          } catch (RuntimeException re) {
34              log.error("get failed", re);
35              throw re;
36          }
37      }
38      public List findAll() {         //查询全部
```

```
39                  log.debug("finding all Student instances");
40                  try {
41                        String queryString = "from Student";
42                        Query queryObject = getSession().createQuery(queryString);
43                        return queryObject.list();
44                  } catch (RuntimeException re) {
45                        log.error("find all failed", re);
46                        throw re;
47                  }
48            }
49      }
```

其中第 7 行定义了 save 保存学生信息方法，其中重要代码是第 10 行调用 Session 会话对象中的 save 方法，从而完成保存功能。第 17 行定义了删除学生学生信息的方法，其中第 20 行中调用 Session 会话对象中的 delete 方法，从而完成删除功能。在第 27 行定义了 findById 根据 id 查询学生的方法，在第 30 行中调用 get 方法完成相应功能。

注意：get 方法中给出的参数并不是数据表，而是实体类的全称类名。后面的 id 参数也是实体类中对应的属性名。

在第 38 行中定义了 findAll 查询所有学生信息的方法，在 Session 会话中并没有定义相应功能，而是首先调用 createQuery 方法获得 Query 对象，然后使用 Query 对象调用 list 方法，从而获得由所有查询到的学生对象组成的集合。第 41 行的字符串就是 HQL 语句，在后面会对它进行详细的讲解。

说明：在 Session 会话类中对数据库的操作进行了非常好的封装，从而在我们调用时只需要调用一个方法就可以完成数据库操作。在 StudentDAO 中存在大量的日志操作代码，但是它们是不影响我们调用的。

注意：在实际开发中，通常会将 StudentDAO 程序做为超类，然后自定义一个子类，对其中必要的方法进行重写，因为在自动生成的程序的方法中并没有进行事务操作，这些需要手动完成。

7.3 配置映射文件

在前面已经讲解了 Hibernate 项目和程序的创建，它们对于一般功能的项目来说已经足够了。但是对于一些功能比较多，或者数据表关系比较复杂的项目来说，就还需要学习更多的内容。它们主要体现在映射文件中，在该文件中进行更详细的配置。

7.3.1 细节配置

细节配置包括映射类、主键生成策略、多对多、一对一等配置。在创建 Hibernate 创建界面中，不管是创建映射文件，还是创建实体类、DAO 程序等都是直接单击“Finish”按钮完成操作。在该界面中，也可以单击“Next”按钮，从而进入细节配置的界面，如

图 7-15 所示。

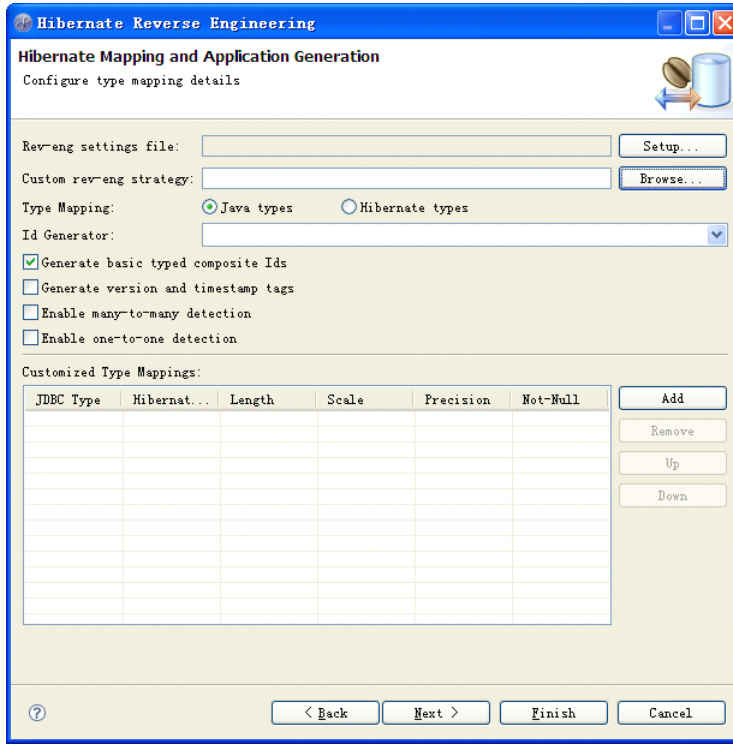


图 7-15　细节配置

对于一般项目来说，这里的选项都是可以采用默认值的。其中“Rev-eng settings file”表示通过一个文件保存配置和选项，如果不指定这样的一个文件，系统也会自动创建。“Custom rev-eng strategy”表示指定自定义的生成策略类，在该类中定义了创建 Hibernate 程序时处理过程的选项。“Type Mapping”表示在映射属性中使用的是什么类型。

“Id Generator”表示主键 id 的生成策略，如果不进行设置，Hibernate 框架会自动指定一个最合适的。在下拉列表中，有很多个选项，这里我们对其中几个常用的生成策略进行讲解，读者可以根据实际情况和开发要求来选择使用。

- increment 标识生成策略：该生成策略是由 Hibernate 在内存中生成主键，每次增量为 1，不依赖于底层的数据库，所以所有的数据库都可以使用。
- UUID 标识生成策略：该生成策略使用一个 128-bit 的 UUID 算法生成字符串类型的标识符，UUID 被编码成一个 32 位 16 进制数字的字符串，该 id 的值将会非常大的。
- identity 标识生成策略：该生成策略使用底层数据库来生成标识符，适用于 MySQL、DB2、MS SQL Server 数据库。该生成策略用于为 long、short、int 类型生成唯一标识。
- sequence 标识生成策略：该生成策略使用底层数据库提供的序列来生成标识符。适用于 DB2 和 Oracle 数据库，该生成策略同样用于为 long、short 或 int 生成唯一标识。

- hilo 标识生成策略：该生成策略使用一个高/低位算法生成标识符，给定一个表和字段做为高位值的来源 i。该策略将 id 主键值的产生源分成两部分，分别是 DB 和内存。
- native 标识生成策略：该生成策略将根据数据库的支持能力，来选择使用 identity，sequence 或 hilo 标识生成策略中的一个。如果为 MySQL 或 MS SQL Server 就选择 identity 标识符生成器。

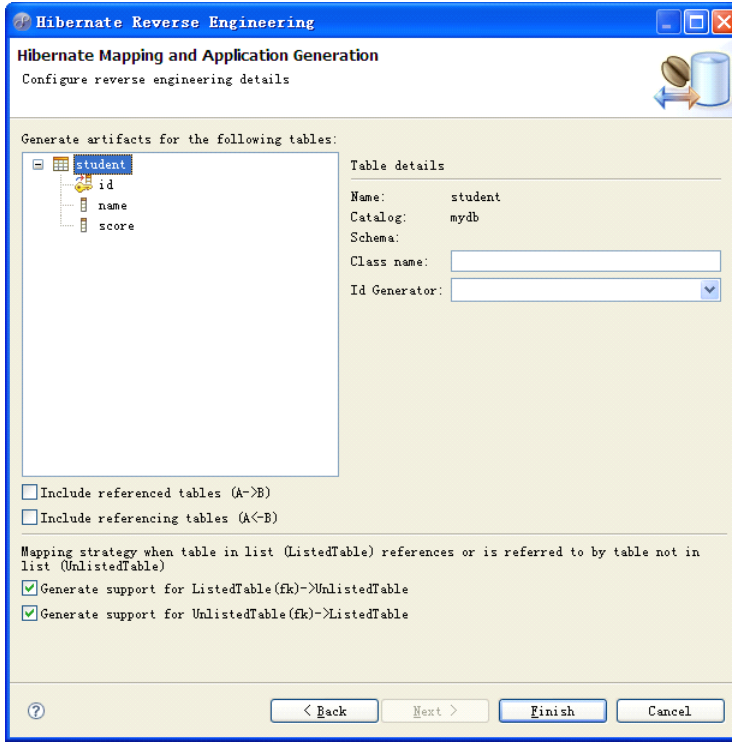
“Generate basec typed composite Ids”表示是否仍然使用基本类型来表示复合主键。“Generate version and timestamp tags”表示名称为“version”和“timestamp”的数据表字段是否以<version>和<timestamp>标记出现。“Enable many-to-many detection”和“Enable one-to-one detection”选项表示是否能够进行多对多和一对一探测。“Customized Type Mappings”表示将指定自定义 JDBC 类型转换成 Hibernate 类型。

说明：在这些细节配置中，id 主键生成策略的选择是最重要的，在实际开发中要灵活使用这些生成策略，并能够看懂项目使用的生成策略。

7.3.2 数据表和实体类的映射配置

在前面自动生成实体类时，都是根据数据表的表名和字段名自动生成实体类的类名和属性名。但是在实际开发中，开发数据库和开发 Java 项目可能并不是一个团队，这时候可能自动生成的类名和属性名并不符合 Java 程序命名规范的。例如数据表的名称为“db_student”，则使用该名称做为实体类的类名，就会不适合。在这种情况下就需要使用到本节讲解的映射配置。

在上一小节的细节配置界面中，单击“Next”按钮，将进入数据表和实体类映射配置界面，在其中选中“student”数据表节点，界面如图 7-16 所示。



7-16　数据表和实体类映射

该界面大致可以分为三部分，左上部分是显示数据表和表中的字段，右上部分中是进行的实体类映射，先来看这两部分。

选中“student”数据表节点，在右边将出现对数据表进行映射的配置。其中“Name”表示数据表名，“Catalog”表示数据表所在的数据库名，“Class name”表示映射实体类的包含包名的完整类名，“Id Generator”表示该表主键的生成策略，主键生成策略在上一小节中已经详细的讲解。

选中“id”主键节点，则右半部分将变为对主键进行映射配置的界面，如图 7-17 所示。

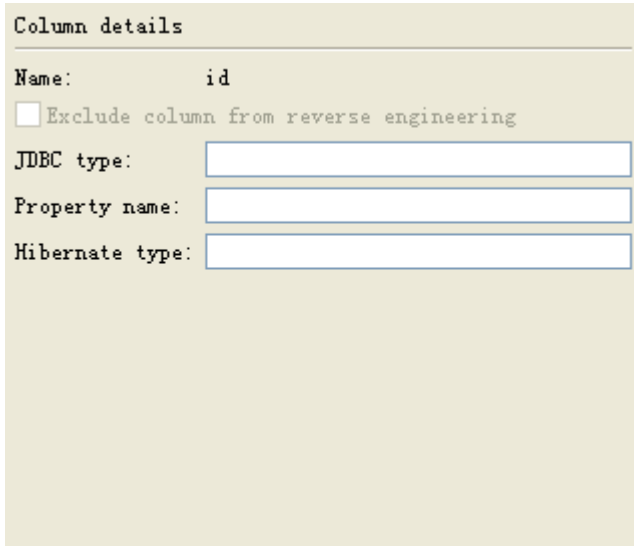


图 7-17　对主键进行映射配置

其中“Name”表示 id 主键的名称。“Exclude column from reverse engineering”表示是否在实体类中忽略主键对应属性的创建。“Property name”表示 id 主键对应的实体类属性名。“JDBC type”和“Hibernate type”表示 id 主键对应的 JDBC 和 Hibernate 的类型。

选中普通字段节点，这里以“name”字段为例，则右半部分将变为对普通字段进行映射配置的界面，其中选项是和设置 id 主键相同的，这里就不再重复讲解。

在具体映射配置界面的下半部分还有几个非常重要的选项。其中“Include referenced tables (A->B)”和“Include referencing tables (A<-B)”表示当数据表中具有外键连接其他数据表时，是否为连接的数据表也进行映射配置。“Generate support for ListedTable(fk)->UnlistedTable”和“Generate support for UnlistedTable(fk)->ListedTable”选项表示是否生成关联到当前数据表的数据表对应的实体类。

7.4　使用注解方式开发 Hibernate 程序

在 JDK 5.0 版本中多出了一种新特性技术，那就是注解，它的英文名称为 Annotations。在 Hibernate 3.2 版本中也增加了通过注解方式开发 Hibernate 程序的方式。通过使用注解，能够代替映射文件，从而也能够让实体类和数据表建立连接。

注意：注解是一把“双刃剑”，通过注解可能节省大量的代码，但是大部分的程序员已经习惯了通过映射文件的方式开发，所以在开发中要按照开发要求来进行选择使用。

7.4.1　创建数据表

通过注解方式开发 Hibernate 程序，改变的只是实体类和映射文件，映射文件将不

再需要，而是在实体类中加入注解。对于数据库和 DAO 程序而言，是没有任何改变的。在使用这种方式开发 Hibernate 程序时，同样要首先创建数据表，这里以老师表为例，它的创建界面如图 7-18 所示。

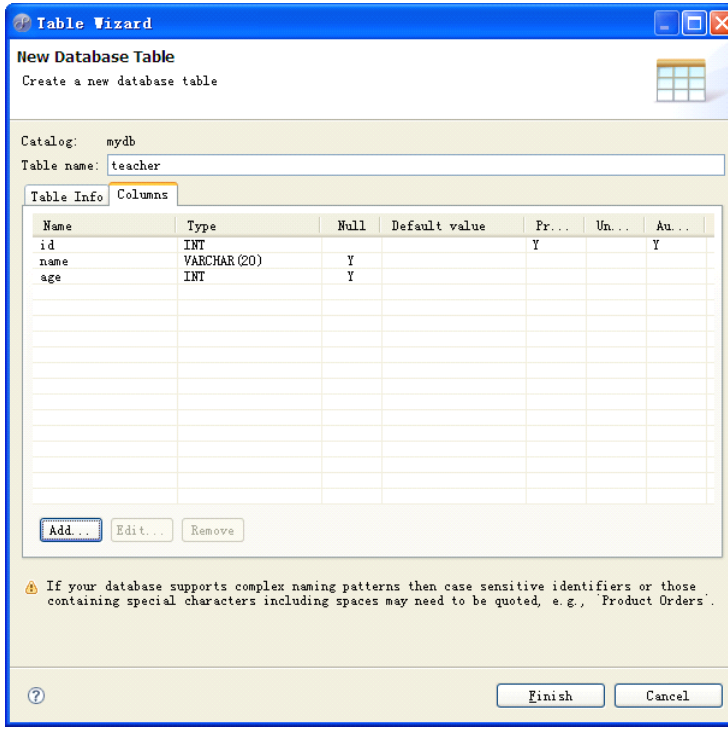


图 7-18　创建老师表

在老师表中创建了 id 主键，以及 name 姓名字段和 age 年龄字段。

7.4.2　使用注解开发

在 MyEclipse 的数据库视图中，选中“teacher”数据表字节，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，在其中选择使用注解选项，界面如图 7-19 所示。

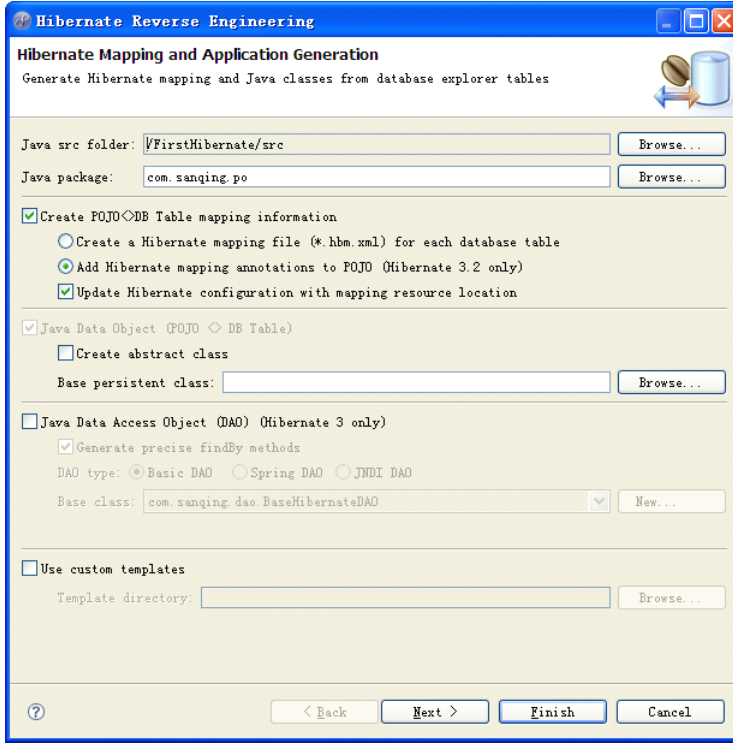


图 7-19　使用注解开发

其中选中“Add hibernate mapping annotations to POJO (Hibernate 3.2 only)”选项，就表示通过注解方式建立和数据表的映射配置。选中该选项后，则“Java Data Object (POJO <> DB Table)”选项将自动变为选中的状态，并且不能取消。这是因为注解必须配置在实体类中。

单击“Finish”按钮，将完成通过注解方式创建的实体类。切换到 MyEclipse 通用视图中，在“com.sanqing.po”包下可以看出多出了一个“teacher.java”程序，它的代码为：

```
01  package com.sanqing.po;
02  //省略导入接口和类的代码
03  @Entity          //注解该类为 Hibernate 实体类
04  @Table(name = "teacher", catalog = "mydb")       //注解映射的数据表
05  public class Teacher implements java.io.Serializable {
06      private Integer id;
07      private String name;
08      private Integer age;
09      public Teacher() {
10      }
11      public Teacher(String name, Integer age) {
12          this.name = name;
13          this.age = age;
14      }
15      @Id        //注解实体类标识
16      @GeneratedValue(strategy = IDENTITY)     //注解主键生成策略
```

```
17          @Column(name = "id", unique = true, nullable = false)   //注解字段
18          public Integer getId() {
19                  return this.id;
20          }
21          public void setId(Integer id) {
22                  this.id = id;
23          }
24          @Column(name = "name", length = 20)
25          public String getName() {
26                  return this.name;
27          }
28          public void setName(String name) {
29                  this.name = name;
30          }
31          @Column(name = "age")
32          public Integer getAge() {
33                  return this.age;
34          }
35          public void setAge(Integer age) {
36                  this.age = age;
37          }
38  }
```

其中第 3 行使用“@Entity”注解实体类，从而将一个普通类定义成 Hibernate 的实体类，也就是和数据表相对应的类。在该注解中可以有 name 属性，用来设置实体名，默认值是类名。

在第 4 行中使用“@Table(name = "teacher", catalog = "mydb")”对实体类进行进一步注解，其中 name 属性指定的是映射的数据表，catalog 属性指定数据表所在的数据库。

第 15 行中使用“@Id”对实体类中的标识进行注解，它对应数据表中的主键。第 16 行中使用“@GeneratedValue(strategy = IDENTITY)”来覆盖主键的默认生成策略，其中 strategy 属性指定主键标识生成策略，IDENTITY 表示由数据库自动设置。

在第 17 行、24 行和 31 行都是使用到了“@Column”注解，通过该注解可以让实体类中的属性和数据表中的字段建立映射关系，其中 name 属性指定的就是字段名。

7.4.3 对注解方式的实体类进行配置

在使用映射文件的方式开发 Hibernate 程序时，需要在“hibernate.cfg.xml”配置文件中对映射文件程序进行配置。使用注解方式开发 Hibernate 程序时，并不会产生映射文件，但是在“hibernate.cfg.xml”配置文件仍然要进行配置，这里是对实体类进行配置。

通过上面的自动生成操作，将自动在“hibernate.cfg.xml”配置文件中对实体类进行配置，自动生成的配置代码为：

```
<mapping class="com.sanqing.po.Teacher" />
```

在这里同样也是使用<mapping>标记，然后使用 class 属性指定实体类的包含包名的全称类名。在包名和类名之间使用“.”就可以，而不需要使用“/”。

对于 Hibernate 中的 DAO 程序来说，不管使用映射文件方式，还是注解方式，是没有区别的。这里就不在讲解 DAO 程序的创建。

7.5 进行 HQL 语句编辑

HQL 的全称为 Hibernate Query Language，直接翻译就是 Hibernate 查询语言。HQL 语句和 SQL 语句非常类似，不同的是 HQL 是面向对象的，HQL 所操作的都是持久对象，而不是数据库表。在 MyEclipse 中，集成了用于对 HQL 语句进行编写的编辑器，在本节中主要讲解该编辑器的使用。

7.5.1 启动 HQL 编辑器

在包资源管理器中，选中要进行 HQL 语句编辑的项目节点，例如这里选中“FirstHibernate”项目节点。单击鼠标右键，在弹出的菜单中，选择“MyEclipse”|“Open HQL Editor”命令，将首先弹出确认是否切换到 Hibernate 操作视图的对话框，如图 7-20 所示。

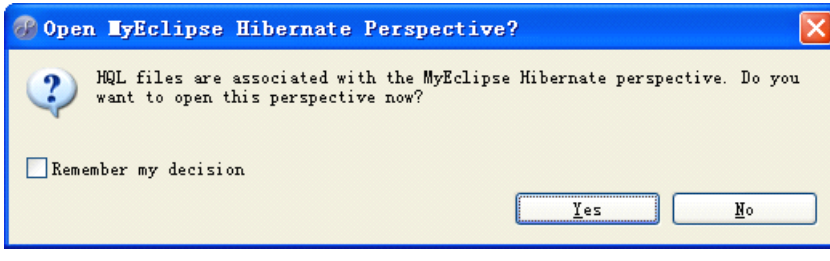


图 7-20 是否切换视图

在前面的讲解中，当弹出该对话框时，通常选择不切换，但是当进行 HQL 语句编辑时最好选择切换，因为在原视图中很多界面是看不到的。切换后的视图界面如图 7-21 所示。

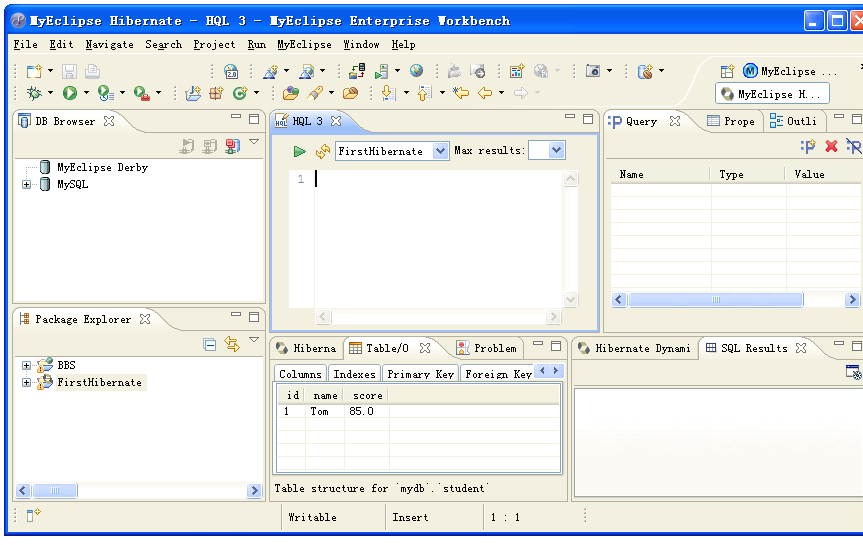


图 7-21　Hibernate 开发视图

该视图界面是比较复杂的，大致可以分为六部分，位于中间的就是 HQL 语句编辑器。在编辑器中，“ ”是运行 HQL 语句按钮，“ ”是刷新按钮，“FirstHibernate ”是用于选择操作项目的下拉列表，“Max results: ”是选择得到最大运行记录数选项。

而在左部分中，左上部分是数据库连接浏览器界面，坐下部分是 Java 项目的包资源管理器界面。其他部分在讲解知识时进行讲解。

7.5.2　执行 HQL 语句

在 Hibernate 的 DAO 程序中已经看过 HQL 语句的使用，在其中做为 Session 对象调用 createQuery 方法时的参数，从而获取 Query 对象。然后使用 Query 对象调用 list 方法从而获取查询结果对象组成的集合。

和编辑 SQL 语句一样，编辑 HQL 语句的作用就是首先验证 HQL 没有问题后，然后将它加入到 DAO 程序中。在执行 HQL 语句之前，我们首先要向数据库中加入测试使用的数据，例如这里以学生表为例插入测试数据。

然后在 HQL 语句编辑器中，输入如下代码：

```
from Student
```

其中“from”是固定的，“Student”对应的是 FirstHibernate 项目中的学生实体类类名。

注意：HQL 语句是对持久化对象进行操作的，而不是对数据表中的记录。所以在 HQL 语句中都是和实体类相关的内容。

在编辑区中，单击“ ”按钮，将运行 HQL 语句，并在中下部分界面的“Hibernate

Query Result”选项卡中将查询得到学生对象显示出来，如图 7-22 所示。

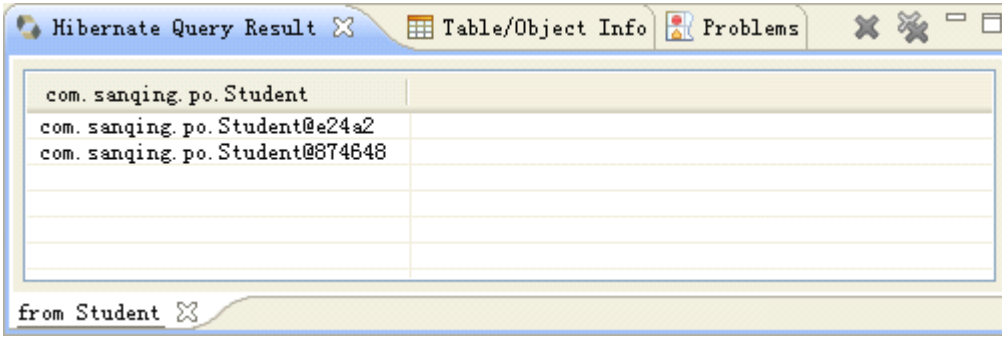


图 7-22　查询得到对象

其中“com.sanqing.po.Student@e24a2”等信息就是查询出来的学生对象，选中该对象。在右上部分的“Properties”选项卡中将把该对象的具体属性值显示出来，如图 7-23 所示。

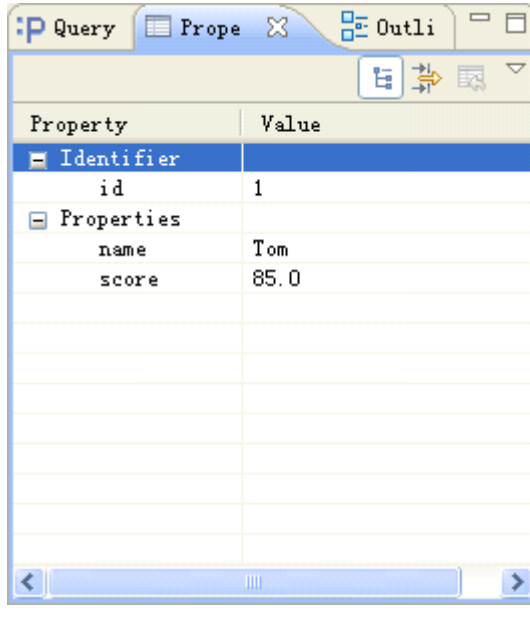


图 7-23　查询具体信息

其中“Identifier”表示对象中的标识属性。“Properties”表示普通属性，这里包括 name 名称和 score 成绩两个属性。在 Value 栏中将这些属性值显示出来。

HQL 语句并不能直接操作数据库，它需要在 Hibernate 框架中被转换为 SQL 语句才能够操作数据库。在 Hibernate 视图中还会将转换后的 SQL 语句显示出来。在右下部分的“Hibernate Dynamic Query Translator”选项卡中将显示出转换后的 SQL 语句，如图 7-24 所示。

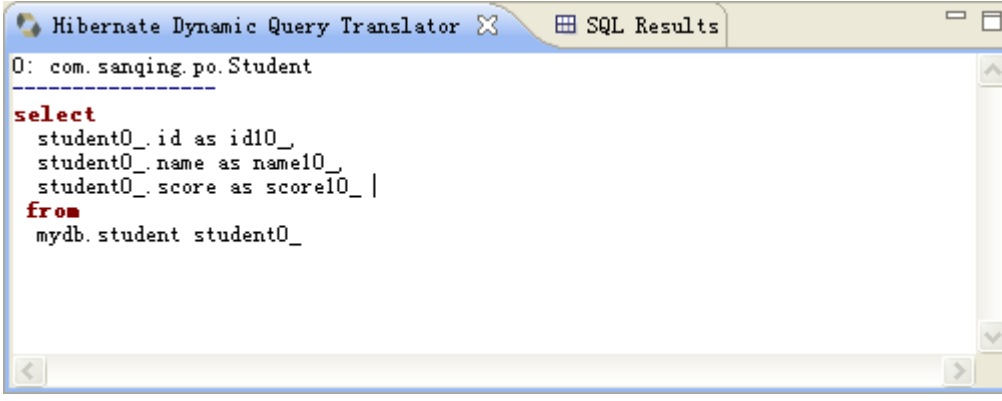


图 7-24　转换后 SQL 语句

说明：由于是框架自动转换，所以转换后的 SQL 语句是和程序员写出的 SQL 语句有所不同的。在自动生成的 SQL 语句中，大量使用到别名，例如“student0_”是数据表的别名，“name10_”是 name 字段的别名。

7.5.3　HQL 语句中的子句设置

在上一小节讲解的是一条最简单的 HQL 语句，通过该语句讲解了该如何运行 HQL 语句。我们可以在 HQL 语句中加入元素，从而进行更详细的查询。在 HQL 语句中可以加入 select 子句，从而查询指定属性的信息，例如：

```
select name,score from Student
```

将只查询学生对象中的姓名和成绩。

在 HQL 语句中，也可以为实体类类名定义别名，在查询属性时就可以使用该别名。例如：

```
select stu.name,stu.score from Student as stu
```

写 HQL 语句时，和写 Java 程序非常类似的。当在实体类别名后写入“.”后，会给出可能操作属性的提示。

在 HQL 语句中也可以使用 where 子句，从而给出查询条件。例如：

```
select name,score from Student where id=1
```

将只查询 id 值为 1 的学生对象，从而将该对象中的姓名和成绩属性值查询出来。

说明：这里只给出常用的几种字句，在大部分项目中已经足够了。如果想了解更多的子句，例如分组、排序等可以查看 Hibernate 相关资料。

7.5.4　为 HQL 语句设置参数

在实际开发中，通过给出的查询条件并不是固定的，而是通过参数传递条件值 。例如查询某学生的具体信息，该学生的名称并不是固定的。这时候就需要为 HQL 语句设置参数。

在 SQL 语句中可以通过位置和属性名两种方式完成参数设置，但是在 HQL 语句中只支持参数名的方式。参数的基本格式为“:name”，例如：

from Student where id= :idname

在 Hibernate 的 DAO 程序中，就可以通过 Query 对象中的 setXXX 方法来完成参数的赋值，其中 XXX 表示为不同的查询参数类型。

在 MyEclipse 的 HQL 语句编辑器中也可以对具有参数的 HQL 语句进行测试，这时候就需要为它定义测试参数值。在 HQL 编辑区中输入上面的 HQL 语句，然后选择右上部分的“Query Parameters”选项卡，如图 7-25 所示。

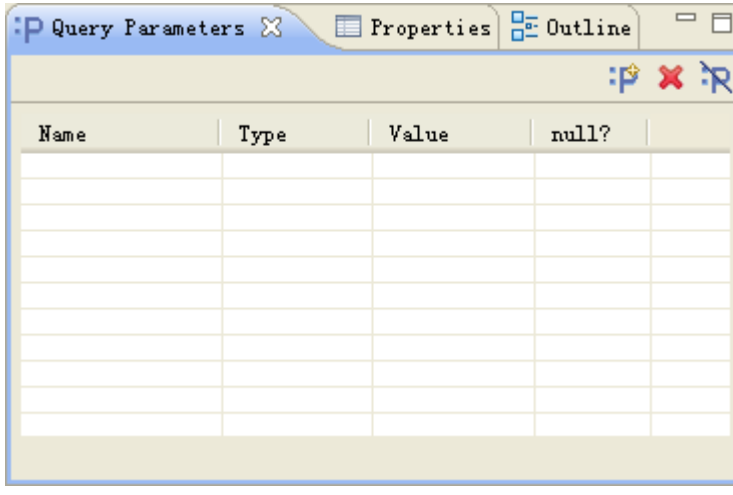


图 7-25　设置参数界面

在界面的右上角中有三个按钮，“ ”是添加参数按钮，“ ”是删除参数按钮，“ ”是禁用参数按钮。在其中单击添加参数按钮，在其中将自动将“Name”栏和“Type”栏填写好，在“Value”栏中可以填写用于测试的参数值，在这里输入“1”。

在 HQL 编辑器中，单击运行按钮，将执行具有参数的 HQL 语句，运行结果如图 7-26 所示。

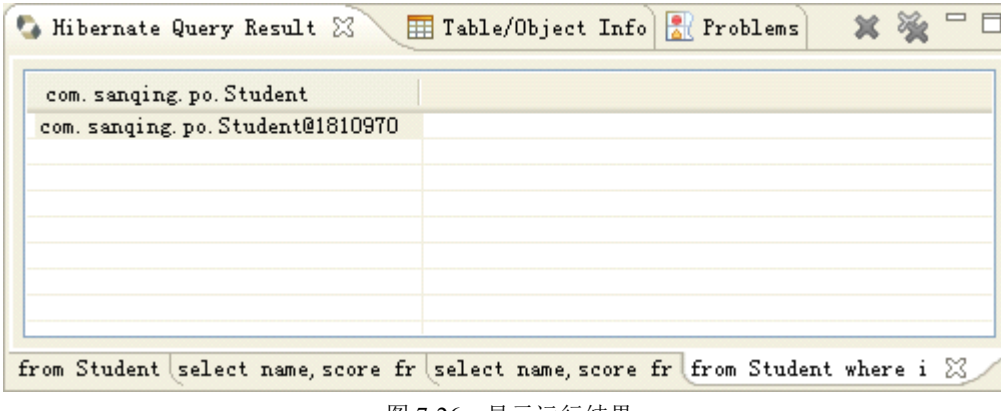


图 7-26　显示运行结果

从运行结果中可以看到只将 id 为 1 的学生对象查询出来。选中该对象，可以在在右

上部分的“Properties”选项卡中查看到该对象的 id 值为 1。

7.6 使用 Hibernate 替换论坛项目的数据访问层

在 Struts 一章的最后开发了一个论坛项目，其中的数据访问层是使用 JDBC 开发的。在本章中学习完 Hibernate 后，我们就可以将其中的数据访问层替换成 Hibernate 技术开发。

7.6.1 准备工作

在执行该操作前，我们首先复制一下项目。在包资源管理器中，选中“BBS”项目节点，使用“Ctrl+C”快捷键进行复制，然后在空白处使用“Ctrl+V”快捷键粘贴，将弹出输入复制后项目名称的界面，如图 7-27 所示。

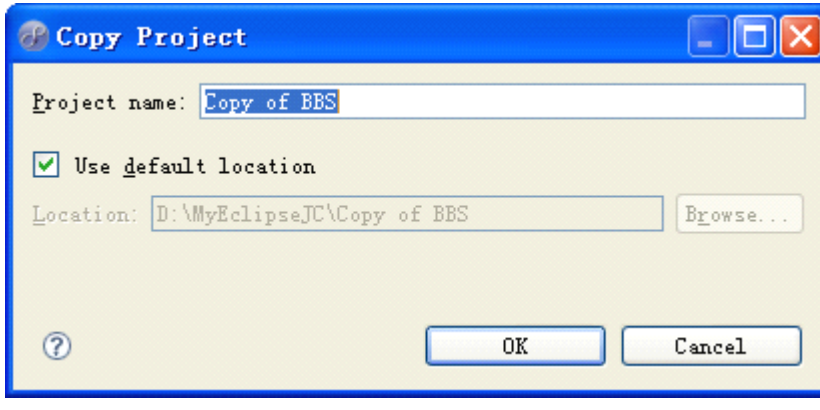


图 7-27 复制项目

在其中“Project name”选项中输入“BBSOfHibernate”做为新项目的名称。单击“OK”按钮后将完成项目的复制，在包资源管理器中可以看到多出了一个名称为“BBSOfHibernate”，这里我们就对它进行操作。

因为我们是替换其中的数据访问层，所以要将其中的原有 JDBC 开发的数据访问层删除掉。选项将其中的 bean、dao、util 包，单击鼠标右键，在弹出的菜单中，选择“Delete”命令，将弹出确定删除的对话框，如图 7-28 所示。

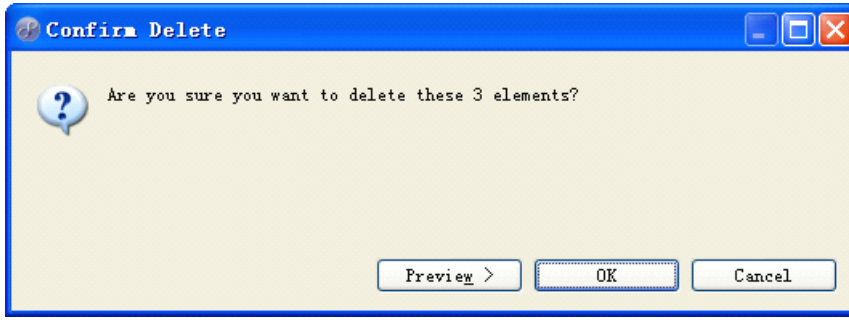


图 7-28　确定是否删除

单击“OK”按钮，将把选中的包和包下的类全部删除。删除后，项目将会发生错误，这是以为其中调用的数据访问层类已经消失。

因为我们这个项目将对 MySQL 中的“bbs”数据库进行操作，所以我们需要对数据库视图中的连接进行修改，选中“MySQL”连接，单击鼠标右键，在弹出的菜单中选择“Edit”命令，将弹出的对连接信息修改的操作界面，修改其中的 URL，界面如图 7-29 所示。

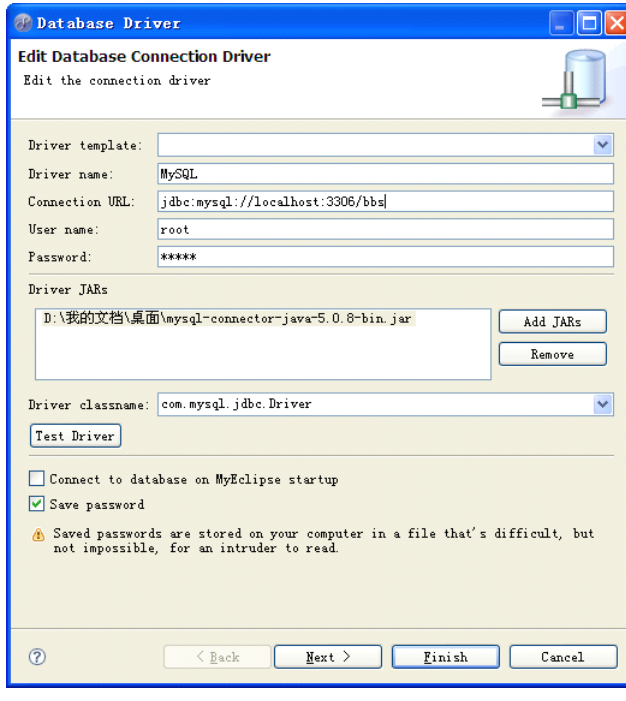


图 7-29　修改连接信息

注意：修改连接时，必须要首先关闭连接。当然修改后，不要忘记再打开。

7.6.2　为论坛项目加入 Hibernate 框架支持

在前面讲解 Hibernate 项目创建时，是对 Java 项目进行操作的。同样对于 Java Web 项目来说，也需要加入 Hibernate 框架支持。

选中“BBSOfHibernate”项目，在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Hibernate Capabilities”命令，将弹出加入支持操作的界面。在该项目中我们采用最通用的映射文件方式创建，所以不选中“Enable Hibernate Annotations Support”选项和“Select the libraries to add to the buildpath”中的第一个选项，选择后的界面如图 7-30 所示。

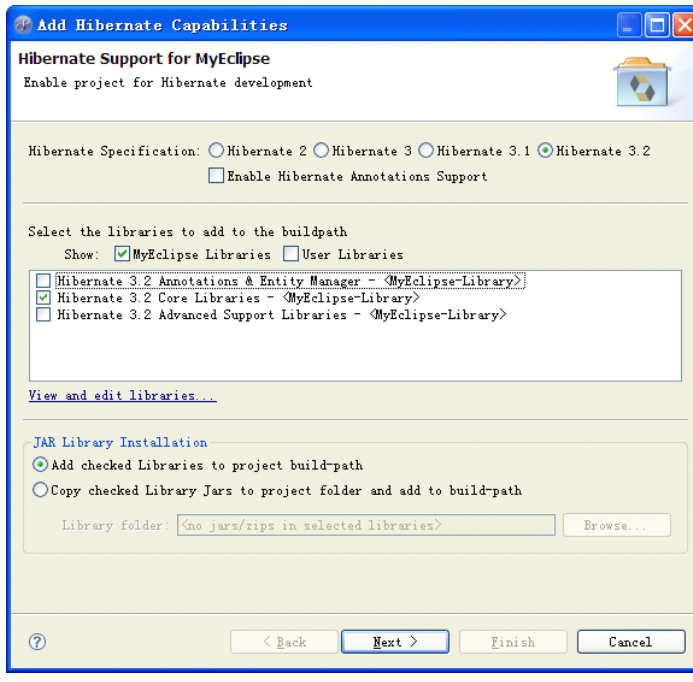


图 7-30　向论坛项目中加入类库

单击“Next”按钮，将弹出创建 Hibernate 配置文件的界面，保持默认选择，如图 7-31 所示。

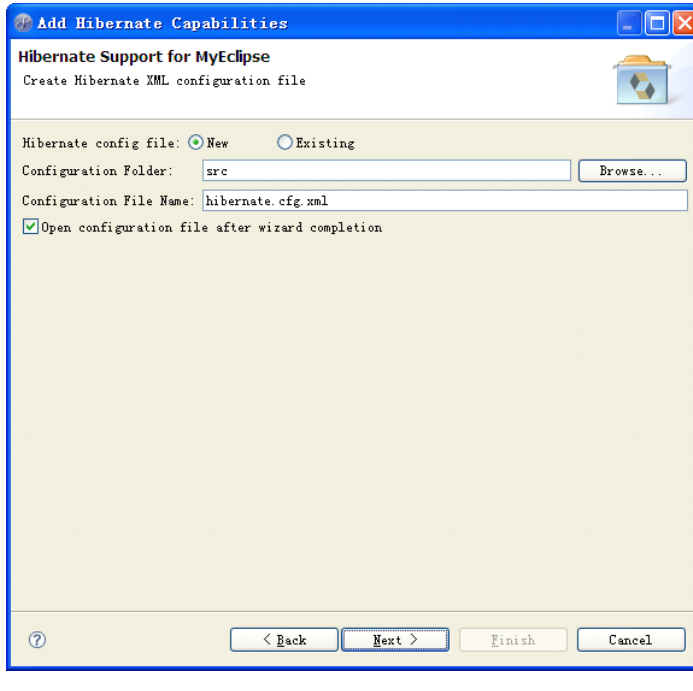


图 7-31　Hibernate 配置文件

注意：虽然 Hibernate 配置文件目前在 src 目录下，但是当发布 Web 项目后，该文

件将位于“WebRoot”|“WEB-INF”|“classes”目录下。这是由 MyEclipse 中的 Web 项目的特性决定的。

单击“Next”按钮，将弹出建立数据库连接的界面，在其中选择“MySQL”连接，如图 7-32 所示。

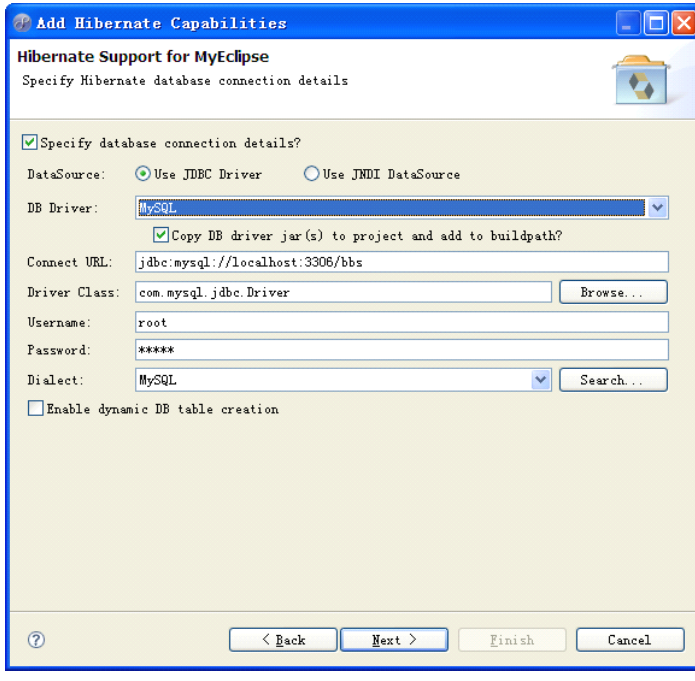


图 7-32　建立和论坛数据库的连接

其中“Commect URL”选项中将变为和 bbs 数据库连接的 URL，如果不修改数据库视图中的视图，这里可能指向其他数据库。单击“Next”按钮，将弹出创建 Hibernate 会话工厂的程序的界面，在其中创建它所在的“com.sanqing.hb”包后，界面如图 7-33 所示。

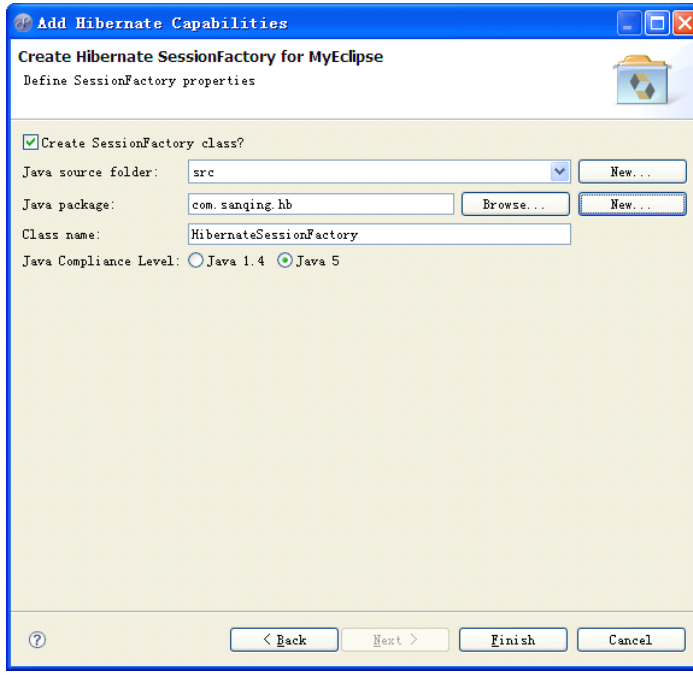


图 7-33　创建 Hibernate 会话工厂

单击“Finish”按钮，将完成 Hibernate 框架支持的操作。

7.6.3　创建 Hibernate 数据访问层

接下来就是开发最关键的 Hibernate 技术的数据访问层。将 MyEclipse 的视图切换到数据库视图，在数据库浏览器中，选中“MySQL”|“Connected to MySQL”|“bbs”|“TABLE”|“topic”节点，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，在其中选择创建映射文件、实体类和 DAO 程序，其界面如图 7-34 所示。

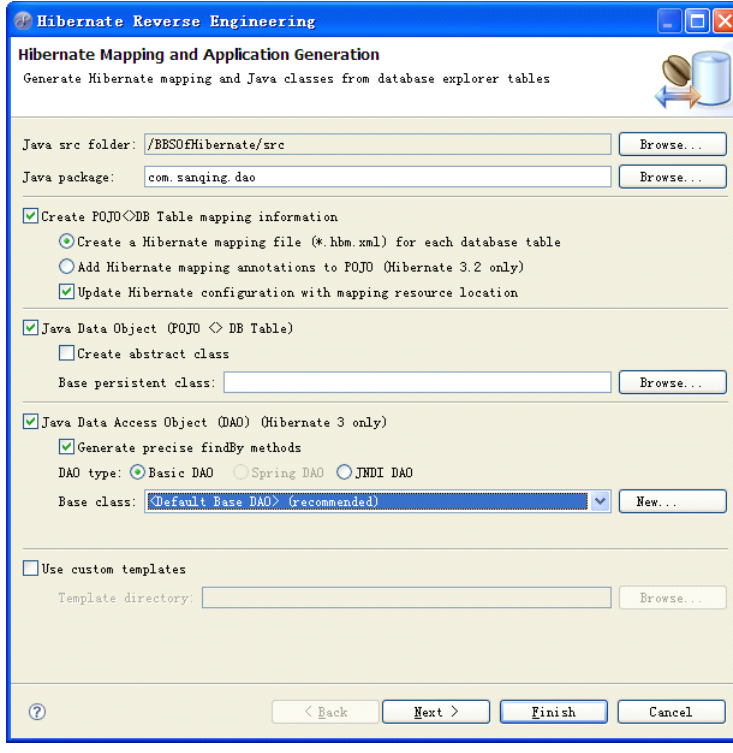


图 7-34　创建数据访问层

这里并不需要进行更详细的配置，实体类和数据表的映射也采用默认选择，所以这里可以直接单击“Finish”按钮，完成 Hibernate 程序的创建。

注意：在有些公司和规范中，规定 DAO 程序和实体类不放在同一个包下，当有该规定时，就需要将它们分开创建，这个在前面已经讲解。

7.6.4　为 DAO 程序加入事务操作

Hibernate 是对 JDBC 的轻量级封装，它本身不支持事务的，所以在进行数据库操作时，要进入事务处理代码。在 Hibernate 中，当打开 Session 会话后，将关闭事务自动提交，这时候需要自己手动提交，这样才能够使数据保存到数据库中。

对于增删改查数据库操作而言，其中查询时不需要进行事务处理，但是对于增加、删除和修改来说，就必须进行事务处理。在论坛项目中有发表主题和删除主题两个必须要进行事务处理的方法。所以这里定义一个子类，让它继承自动生成的“TopicDAO”程序，在其中重写需要事务处理的方法。

在包资源管理器中，选中“com.sanqing.dao”包，单击鼠标右键，在弹出的菜单中选择“New”|“Class”命令，将弹出创建类的界面，在其中输入类名为“TopicDAOImpl”，选择继承 TopicDAO 类，界面如图 7-35 所示。

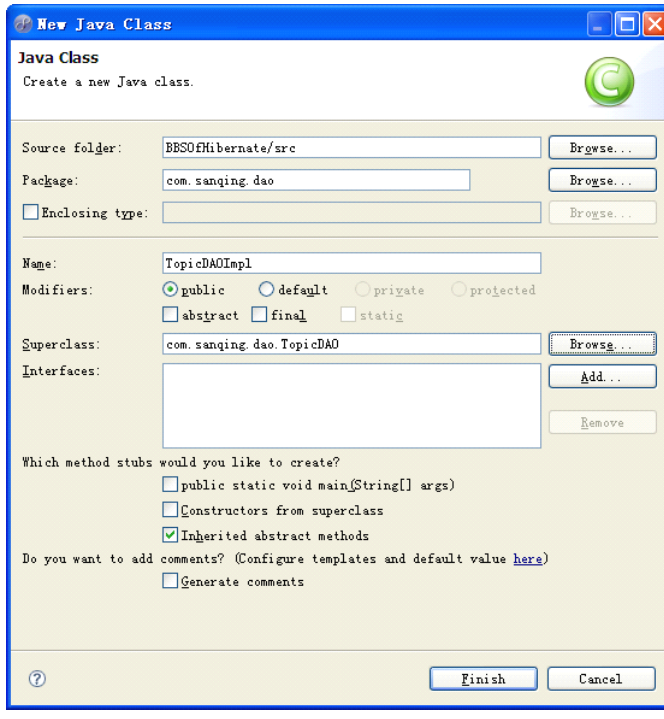


图 7-35　创建 DAO 程序子类

单击“Finish”按钮，将完成“TopicDAOImpl”程序的创建，在编辑区中将打开该程序。将鼠标焦点放在类体中，单击鼠标右键，在菜单中选择“Source”|“Override/Implement Methods”命令，将弹出重写方法的界面，在其中选择 delete 和 save 方法，如图 7-36 所示。

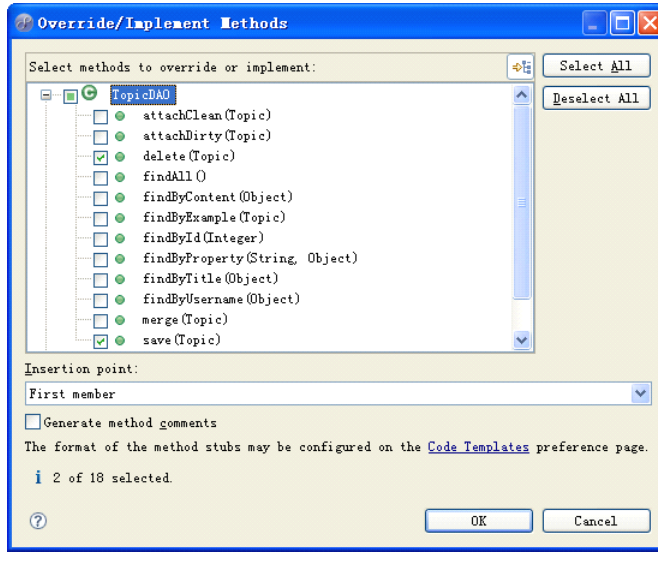


图 7-36　选择重写的方法

单击“OK”按钮，将完成两个方法的重写，修改其中的方法，加入事务操作的代

码，则“TopicDAOImpl”程序的完整代码如下：

```
01  package com.sanqing.dao;
02  import org.apache.commons.logging.Log;
03  import org.apache.commons.logging.LogFactory;
04  public class TopicDAOImpl extends TopicDAO {
05      private static final Log log = LogFactory.getLog(TopicDAO.class);
06      @Override
07      public void save(Topic transientInstance) {
08          log.debug("saving Topic instance");
09          try {
10              getSession().beginTransaction();          //开启事务
11              getSession().save(transientInstance);
12              getSession().getTransaction().commit();       //提交事务
13              log.debug("save successful");
14          } catch (RuntimeException re) {
15              log.error("save failed", re);
16              getSession().getTransaction().rollback();       //回滚
17              throw re;
18          }
19      }
20      @Override
21      public void delete(Topic persistentInstance) {
22          log.debug("deleting Topic instance");
23          try {
24              getSession().beginTransaction();                //开启事务
25              getSession().delete(persistentInstance);
26              getSession().getTransaction().commit();       //提交事务
27              log.debug("delete successful");
28          } catch (RuntimeException re) {
29              log.error("delete failed", re);
30              getSession().getTransaction().rollback();       //回滚
31              throw re;
32          }
33      }
34  }
```

其中第 10 行使用获取到的 Session 会话对象调用 beginTransaction 方法从而开启事务。当执行完数据库操作后，第 12 行中调用 commit 方法进行提交事务，从而将数据保存到数据库中。当中间发生异常后，将进行异常处理，将执行第 16 行的 rollback 方法，从而进行回滚操作。

说明：这时候项目中还是会有错误的，这是因为原 Topic 主题类和目前的 Topic 实体类并不在同一个包下。打开出错的程序，执行组织导入操作就可以了，快捷键是“Ctrl+Shift+O”。由于原 Topic 类中表示发表时间时使用的是 Date 类型，而 Hibernate 自动生成的 Topic 类中使用的 Timestamp 类型。Timestamp 类是 Date 类的子类，必要的地方进行修改，具体修改看光盘代码。

7.6.5 修改项目的访问 URL

因为我们的项目是复制原有项目的，虽然项目的名称修改了，但是当该项目发布到服务器后，访问的 URL 地址是修改的。这时候访问“BBSOfHibernate”项目时，访问地址将是：

http://localhost:8080/BBS

这并不是我们想要的，所以接下来将修改该项目的 URL。在包资源管理器中，选中“BBSOfHibernate”项目节点，单击鼠标右键，在弹出的菜单中选择“Properties”命令，将弹出对项目设置的界面，在其中选择“MyEclipse”|“Web”节点，界面如图 7-37 所示。

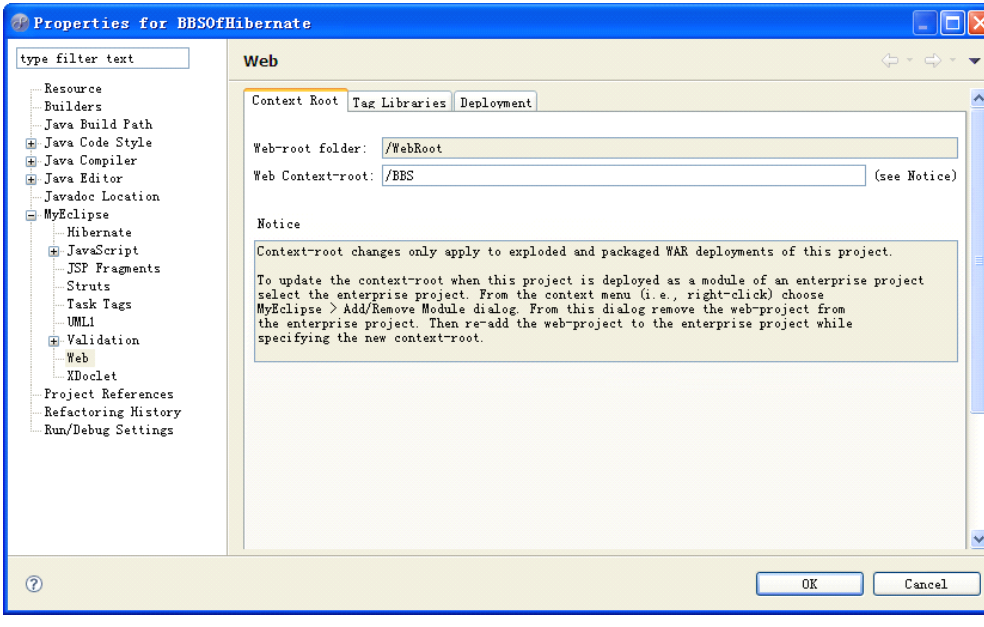


图 7-37　修改访问 URL

其中“Web Context-root”选项就是访问项目的 URL，将其中的“/BBS”修改为“/BBSOfHibernate”，单击“OK”按钮，将完成项目 URL 的修改。

7.6.6 发布和运行项目

发布和运行 Java Web 的项目都是非常类似的，不管使用什么框架开发。首先要将项目部署到应用服务器中，例如“Tomcat 6.x”。运行“Tomcat 6.x”应用服务器，启动成功后，打开浏览器中，输入如下地址：

http://localhost:8080/BBSOfHibernate/

其中“BBSOfHibernate/”就是上一小节修改的项目访问 URL。因为开发原 BBS 项目时已经修改了项目首页，复制后将不会发生改变。该地址将直接访问发表主题页面，浏览器如图 7-38 所示。

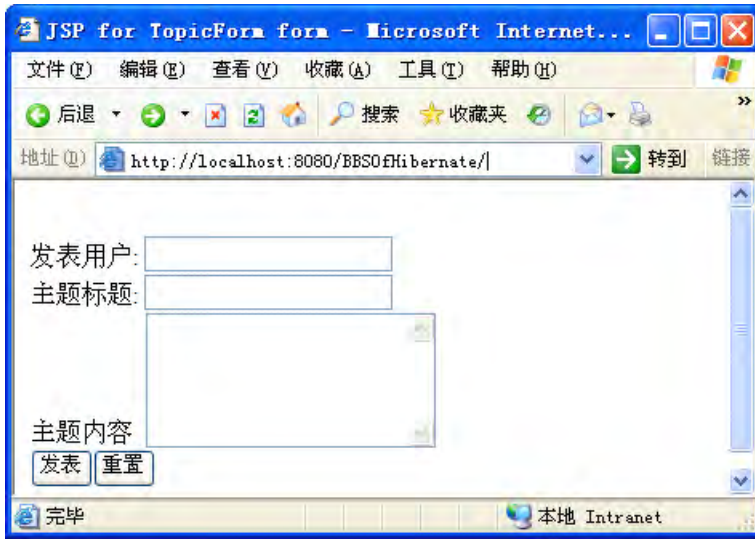


图 7-38　发表主题

在其中输入发表用户、主题标题和主题内容后，单击“发表”按钮，将进入到显示所有主题的页面，如图 7-39 所示。

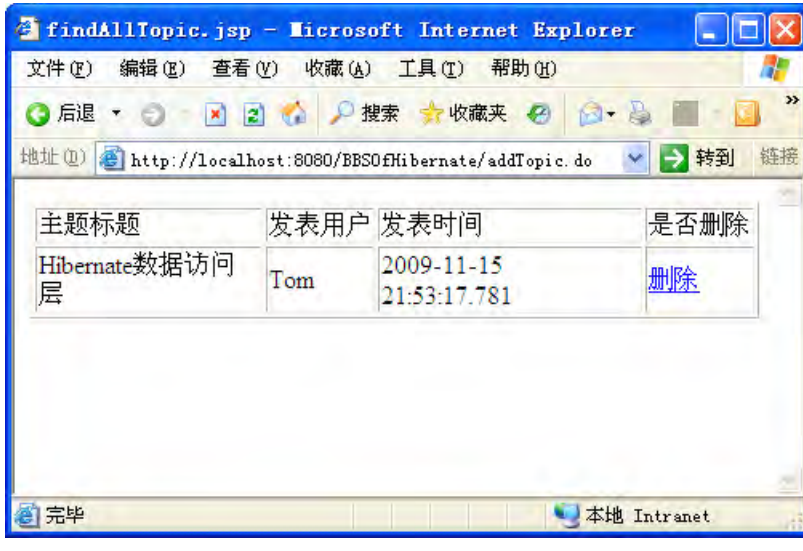


图 7-39　显示所有主题

单击“删除”超链接后，将把当前主题删除，并且再次跳转会显示所有主题页面，如图 7-40 所示。

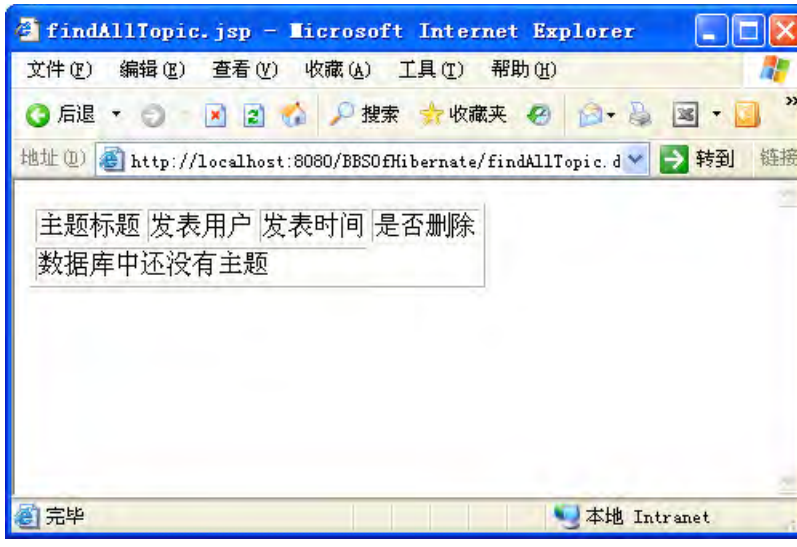


图 7-40　删除主题

说明：该项目开发中有两个需要注意的地方，首先就是必须重写自动生成的 DAO 程序，在重写方法中加入事务处理的代码。第二点就是不要忘记修改项目的访问 URL。

第8章　进行Spring开发

在前面的讲解中，已经对 Struts 和 Hibernate 框架进行了详细的讲解，在本章中继续学习 Spring 框架。Spring 是一种开源框架，通过使用它可以大大降低企业应用程序的复杂性。Spring 是一种非常完善的框架，几乎涉及 WEB 开发中的每一层，但是在开发中通常使用 Spring 开发业务逻辑层。

SSH 框架是目前最流行的软件开发技术框架，其中 Spring 的作用就是使 Struts 和 Hibernate 建立连接，使它们更好的分层。

8.1　Spring 概述

Spring 是一种轻量级框架，轻量级是指该框架技术是非入侵式的，用 Spring 中开发的系统类不需要依赖于 Spring 中的类，它不需要容器支持。

下面我们主要来讲解一下在项目中为什么要使用 Spring 框架，其中最主要的一个原因就是很好的进行分层操作。在前面的项目中，我们都是通过工厂类的方式进行不同层之间的调用，例如在 Action 中调用 DAO 工厂类中的方法，从而获取 DAO 实现类对象。这种方式虽然能够实现了分层，但是并不彻底的。

Spring 的作用就是更彻底的进行分层操作，它采用注入的方式进行对象的引入。例如在业务逻辑层中调用数据访问层，在业务逻辑层中的代码仅出现数据访问层的接口，具体如何使用，使用哪一个实现类对象，这些都是以配置文件注入的方式操作的，从而使业务逻辑层并不需要关心数据访问层是如何实现的。

Spring 是一种非常完整的技术，它涉及 Web 开发中的各个方面。但是在实际开发中，并不会全用到，用到最多的就是 IoC 技术和 AOP 技术。上面所说的就是 Spring 中非常重要的 Ioc 技术，也就是控制反转。除了该技术外，还有 AOP 面向切面编程，在本章的后面将会讲解它。

8.2　开发 Spring 项目

Spring 框架和 Hibernate 框架一样，也是不仅仅在 Java Web 项目中使用的。所以这里我们以一个简单的 Java 项目讲解 Spring 的应用。在 MyEclipse 中很好的集成了 Spring 项目和程序的开发，在本节就先来看在其中是如何开发 Spring 项目的。

8.2.1　开发普通的 Java 项目

Spring 的项目是基于普通项目的，然后在普通项目的基础上加入 Spring 的框架支持。在 MyEclipse 的菜单中，选择“File”|“New”|“Java Project”命令，就会弹出创建 Java

项目的界面，在其中输入项目的名称为“FirstSpring”，如图 8-1 所示。

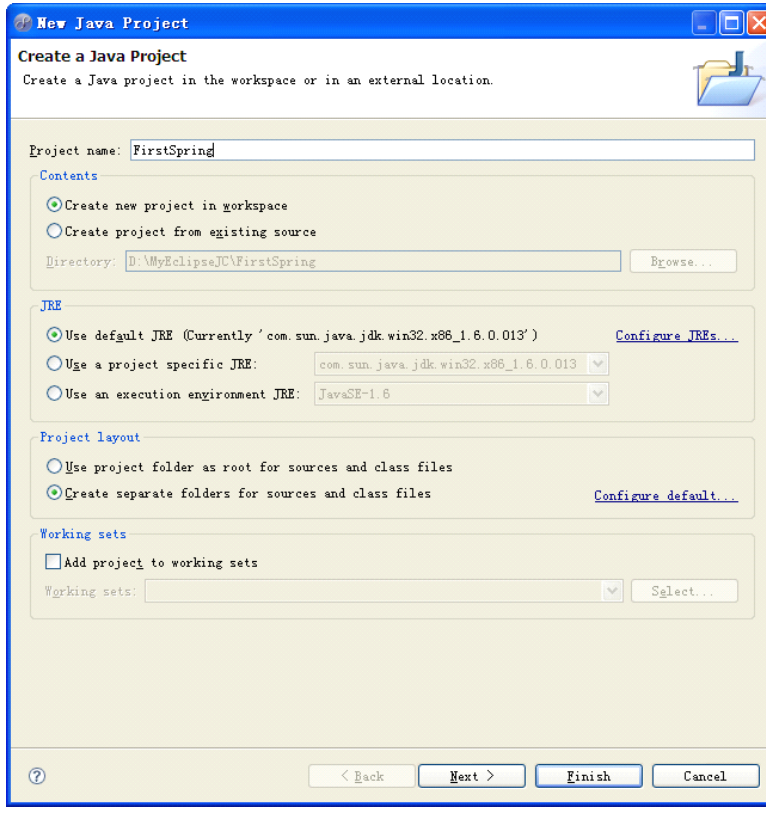


图 8-1　创建普通 Java 项目

单击“Finish”按钮，将完成普通项目的创建，它的项目名称为“FirstSpring”。

8.2.2　加入 Spring 框架支持

在包资源管理器中，选中要加入 Spring 框架支持的项目，这里选择“FirstSpring”项目。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Spring Capabilities”命令，将弹出加入 Spring 类库支持操作界面，如图 8-2 所示。

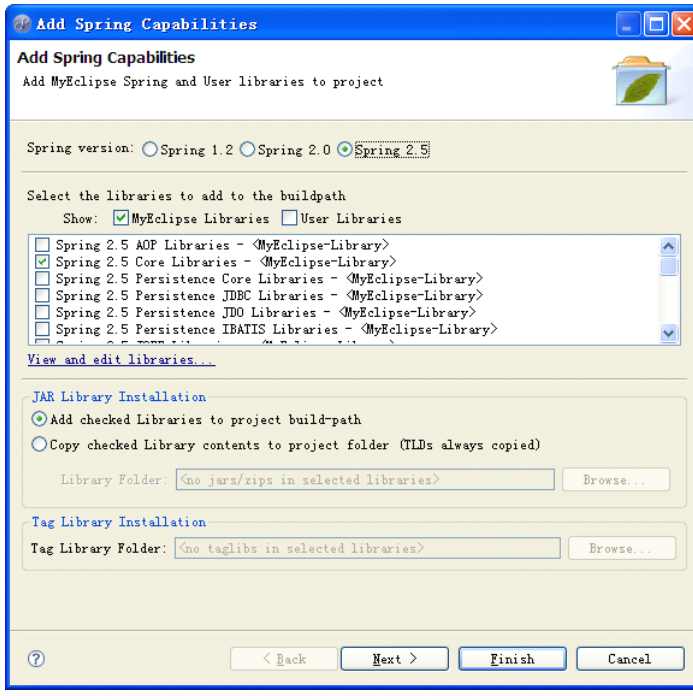


图 8-2　加入 Spring 类库

其中“Spring version”表示加入项目的 Spring 框架的版本，通常选择目前最高的 Spring 2.5 版本。“Select the libraries to add to the buildpath”表示将向项目中加入哪些类库。由于 Spring 的技术非常全面，并且能够整合很多种技术，所以其中的类库也是非常多的。这里我们先采用默认的选择，也就是仅选中“Spring 2.5 Core Libraries”选项，它是 Spring 开发的核心类库，当进行其他操作，再添加其他的类库。

“JAR Library Installation”表示是否将类库复制到项目中。其中“Add checked Libraries to project build-path”选项表示不复制 JAR 文件类库到项目中，只在发布项目时进行复制，这是默认的选项，也是经常采用的选项。“Copy checked Library contents to project folder(TLDs always copied)”选项表示复制 JAR 文件类库到项目中，选择该选项后还需要填写将这些类库放置的目录，这种操作的好处是使开发的项目不依赖于 MyEclipse。

“Tag Library Folder”选项仅对 Web 项目起作用，表示指定标签库文件的安装目录。单击“Next”按钮将进入到创建 Spring 配置文件的界面，如图 8-3 所示。

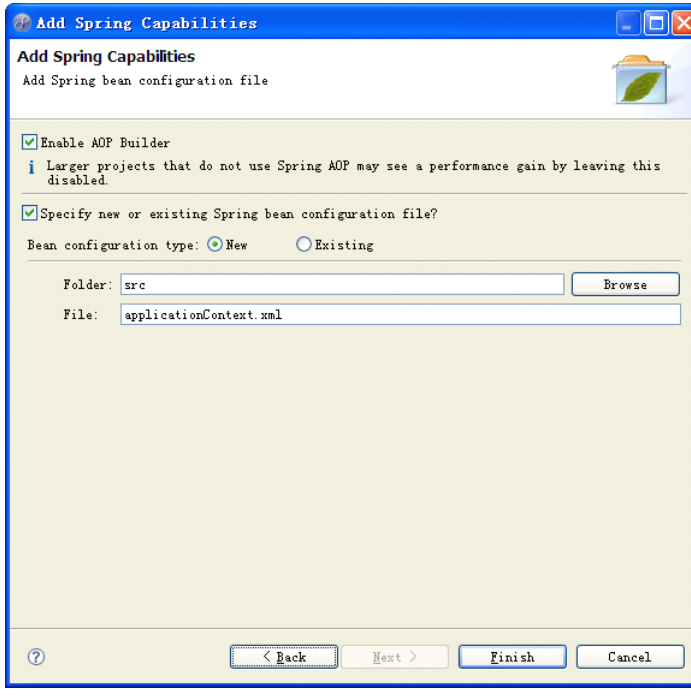


图 8-3　创建 Spring 配置文件

其中“Enable AOP Builder”选项表示是否能够加入 AOP 技术支持，保持默认的状态就可以。“Specify new or existing Spring bean configuration file”表示是否创建 Spring 配置文件，这里采用默认的选中状态。“Bean configuration type”表示 Spring 配置文件的类型，“New”表示新建，“Existing”表示使用已经存在的。

“Folder”表示 Spring 配置文件所在的目录，默认是 src 目录，保持默认就可以。“File”表示配置文件的名称，通常也采用默认的“applicationContext.xml”名称。单击“Finish”按钮，将完成加入 Spring 框架支持的操作。

说明：和前面学习 Hibernate 配置文件一样，在 Java Web 项目中，当部署 Web 项目时，当会将该配置文件放到 classes 目录下，而不是在目前的 src 目录下。

8.2.3　认识 Spring 的项目结构

向普通的 Java 项目加入 Spring 框架支持后，将向项目加入 Spring 核心类库和 S 的配置文件，在包资源管理器中，将项目展开，如图 8-4 所示。

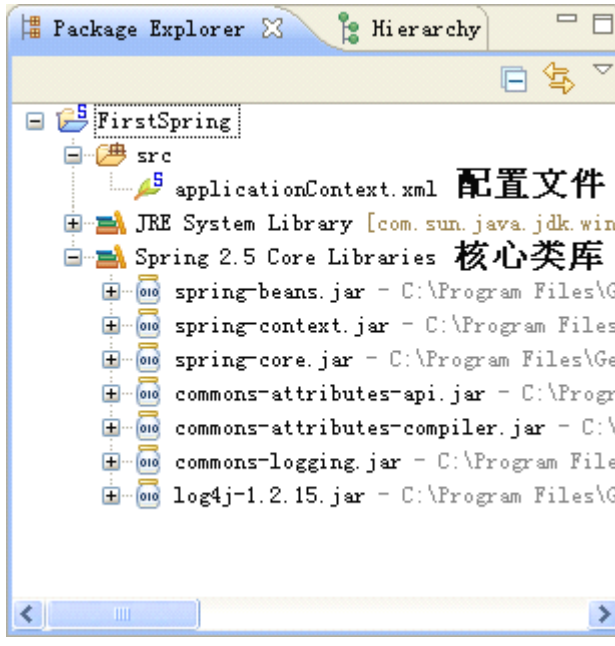


图 8-4　Spring 项目结构

从项目结构上可以看出，Sring 的项目结构要比 Hibernate 的项目结构简单的多。其中“Spring 2.5 Core Libraties”就是加入框架支持时使用的核心类库。“applicationContext.xml”就是 Spring 的配置文件，双击在编辑区中打开该文件。

在编辑区的左下角有两种显示程序的选项，选择“Design”选项将以 XML 的形式显示 Spring 的配置文件，如图 8-5 所示。

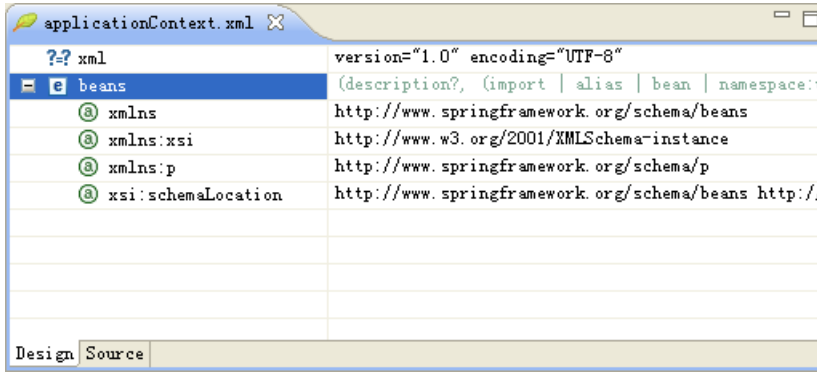


图 8-5　XML 文件形式显示

选择编辑区左下角的“Source”选项，会将 Spring 配置文件的代码显示出来，具体代码为：

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans
03      xmlns="http://www.springframework.org/schema/beans"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xmlns:p="http://www.springframework.org/schema/p"
```

```
06          xsi:schemaLocation="http://www.springframework.org/schema/beans
07                  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
08  </beans>
```

其中<beans>标记就是 Spring 配置文件的根标记，在该标记对中可以对 Spring 项目中的 Bean 进行配置。

说明：这里所说的 Bean 其实就是普通的类，在 Spring 中通过将这些类称为 Bean，它和通常所说的规范 JavaBean 是不一样的。

8.3 开发控制反转程序

控制反转是 Spring 的核心技术，它是英文名称为“Inversion of Control”，所以在很多地方将控制反转缩写为 IoC。控制反转还有另外一个名称，叫做依赖注入，它的英文名称为 Dependency Injection，简称是 DI。它们指的是一种技术，就是在框架中将对象创建后，直接注入到调用的程序中。

8.3.1 创建 Bean 程序

在 Spring 的控制反转项目中，会自动生成一个 IOC 容器，在该容器中包含所有创建的 Bean，并对这些 Bean 进行管理。开发 Bean 程序分为两步，首先要创建 Bean 程序类，然后对它进行配置。Bean 程序就是一个普通的类，但是在该类中要有 Setter 方法或者构造函数，从而能够将值或者 Bean 对象注入。

这里我们就以 Web 开发中的业务逻辑层进行举例。在实际开发中，通常将业务逻辑层的程序放在“service”包中，所以首先在“src”目录下创建“com.sanqing.service”包，然后在该包下创建名称为“HelloServiceImpl”的业务逻辑实现类，它的代码为：

```
01  package com.sanqing.service;
02  public class HelloServiceImpl {
03      private String message;              //打招呼的信息
04      public void setMessage(String message) {    //通过 Setter 方法注入
05          this.message = message;
06      }
07      public void sayHello(){      //打招呼方法
08          System.out.println("内容为："+message);
09      }
10  }
```

其中第 3 行中定义了打招呼的信息，在第 4 行中定义 Setter 方法将值注入。第 7 行是业务方法，完成打招呼的功能。

注意：其中的 Setter 方法是非常重要的，在后面的讲解中会了解到。有些程序员习惯也生成 Getter 方法，但是它对于 Spring 的操作来说，是没有任何用处的。

8.3.2 配置 Bean 程序

在上一小节中通过创建类的方式创建了 Bean 程序，在 MyEclipse 中并没有集成该功能。在本节中将继续讲解如何将 Bean 程序配置到 Spring 的配置文件中，在 MyEclipse 中很好的集成了该功能。

双击“ApplicationContext.xml”文件，在编辑区中打开该文件，将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，如图 8-6 所示。

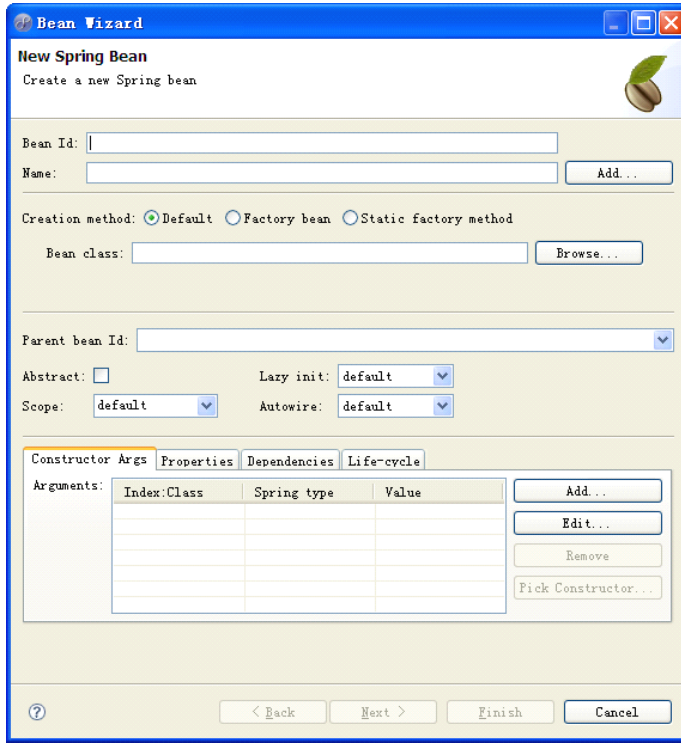


图 8-6 配置 Bean 程序

其中“Bean Id”表示 Bean 程序的 id 名称，相当于为程序起一个名称，在后面的使用 Bean 程序时使用它，这里填写“hello”。“Name”选项在普通的 Spring 程序中通常不填，当和 Struts 框架整合时，通常才配置它。

“Creation method”表示实例化 Bean 的方式，“Default”选项表示默认的方式，也就是通过构造函数来实例化；“Factory bean”选项表示通过实例工厂方法方式实例化 Bean；“Static factory method”选项表示使用静态工厂方法实例化 Bean。第一个默认选项只需要给出 Bean 程序的完整类名就可以，如果使用后两种选项，还需要给出其他配置，这里采用最简单的默认方式实例化 Bean。“Bean class”表示 Bean 程序的完整类名，这里选择填写“com.sanqing.service.HelloServiceImpl”。

说明：在后面的配置中，还有一些选项，它们都对应相应的知识点，这里我们先采用默认值，在后面的讲解中再详细的讲解它们。

在界面的最下面，选择“Properties”选项卡，它是对以 Setter 方法方式注入时的设

置。单击“Add”按钮，将弹出设置 Setter 方法属性界面，如图 8-7 所示。

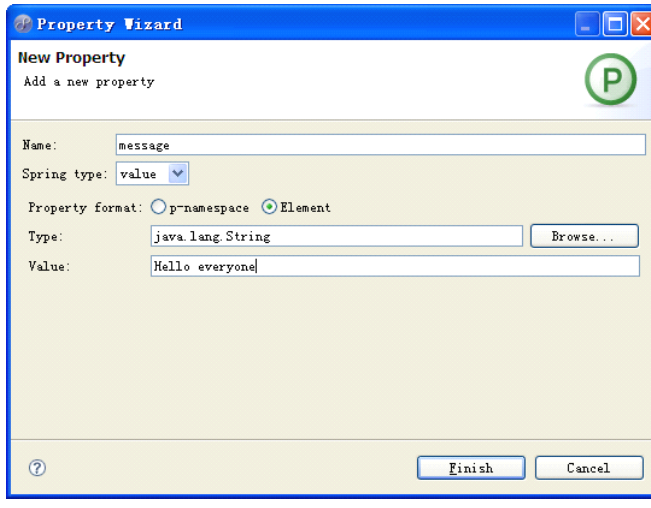


图 8-7　设置属性

其中“Name”就对应 Bean 程序中的属性名称，这里填写其中的“message”属性。“Spring type”对应注入类型，这里选择“value”选项，表示进行基本类型注入。“Property format”表示注入格式，使用最多的就是“Element”选项。“Type”表示注入属性的类型，“Value”表示注入值内容。单击“Finish”按钮，将完成属性的设置，并回到配置 Bean 程序界面，单击“Next”按钮，将进入注入方法的界面，如图 8-8 所示。

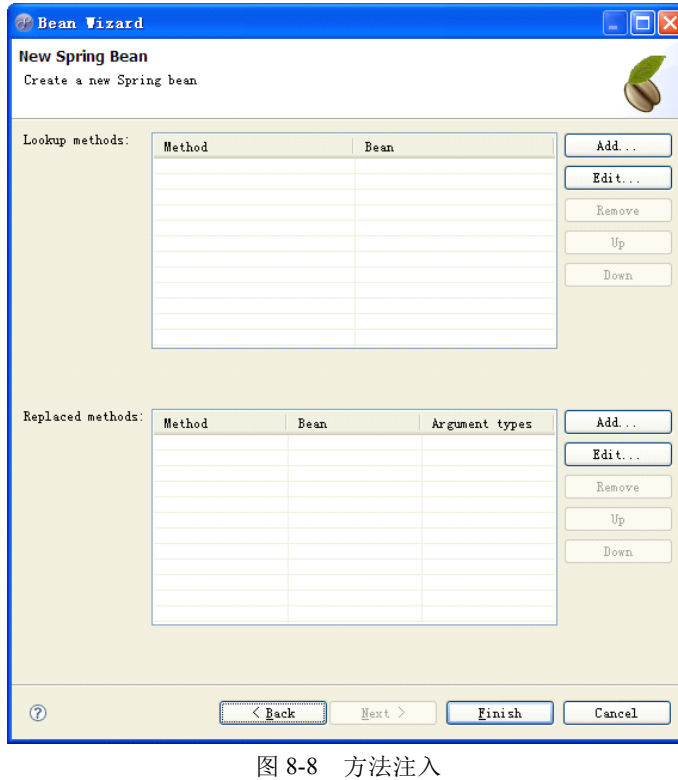


图 8-8　方法注入

在该界面中可以完成 Lookup 方法和自定义方法的注入，对于普通的 Spring 项目而说，可以忽略这一步操作。单击“Finish”按钮，将完成配置 Bean 程序的操作。

在“ApplicationContext.xml”Spring 的配置文件中，将自动添加配置代码，具体内容为：

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans
03      xmlns="http://www.springframework.org/schema/beans"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xmlns:p="http://www.springframework.org/schema/p"
06      xsi:schemaLocation="http://www.springframework.org/schema/beans
07          http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
08      <bean id="hello" class="com.sanqing.service.HelloServiceImpl"
09          abstract="false" lazy-init="default" autowire="default"
10          dependency-check="default">
11          <property name="message">
12              <value type="java.lang.String">Hello everyone</value>
13          </property>
14      </bean>
15  </beans>
```

其中我们最关心的是第 8 行和第 11 行到第 13 行的代码。在第 8 行中配置 Bean 程序类，并为它起了一个 id 名称。在第 11 行中使用<property>标记对配置了属性值，它只对于通过 Setter 方法方式注入起作用，其中使用 name 属性指定要配置的属性名称。在第 12 行中给出属性值，其中 type 属性指定属性类型，标记对中位属性值。

说明：在配置文件中，有多个属性值是“default”，它们都是 MyEclipse 自动生成的，也可以手动去掉它们，是不影响功能的，在后面的讲解中会说明它们的作用。

8.3.3 编写测试类

在实际开发中，测试类并不是必须的，但是通过都会有该程序，通过它可以判断开发的局部程序是否正确。在本节中就来开发这样的测试类，通过它来测试是否能够将配置文件中的值注入到 Bean 程序中。测试类的具体代码如下：

```
01  package com.sanqing.test;
02  import org.springframework.beans.factory.BeanFactory;
03  import org.springframework.context.support.ClassPathXmlApplicationContext;
04  import com.sanqing.service.HelloServiceImpl;
05  public class HelloTest {
06      public static void main(String[] args) {
07          BeanFactory factory =
08                  new ClassPathXmlApplicationContext("applicationContext.xml");
09          HelloServiceImpl hello=
10                  (HelloServiceImpl)factory.getBean("hello"); //获取 Bean 实例
11          hello.sayHello();//调用业务方法
12      }
```

```
13    }
```

其中第 7 行是实例化 IoC 容器的代码，第 9 行中通过容器实例化指定的 Bean，在第 11 行中调用 sayHello 业务方法，从而完成相应功能。运行该程序，运行结果为：

```
内容为：Hello everyone
```

从运行结果中可以看到，已经将 Spring 配置文件中的值注入到 Bean 程序中。

说明：从测试类中也可以看到，我们并没有通过 new 来创建 Bean 实例，就可以调用其中的业务方法，这就是控制反转的作用。

8.4 依赖注入

在前面已经讲解过，控制反转和依赖注入指的是同一个技术。但是使用名称时又有些小的不同，当指整个技术时，通常叫做控制反转；当指注入操作时，通常叫做依赖注入。在本节中，就主要来讲解一下注入的操作，包括注入方式和注入类型等，看一下 MyEclipse 中是如何操作它们的。

8.4.1 注入方式

依赖注入有两种常用的方式，分别是 Setter 方法注入和构造函数注入。使用不同方式时，Bean 程序中对属性设置的方式不同，同样对应的配置也是所有不同的。在上一节中，已经见过了 Setter 方法的方式注入，在本小节中再来看一下如何使用构造函数注入。

说明：这两种方式中，Setter 方法的方式使用的是比较多的，这也是 Spring 框架中提倡的方式。

当使用构造函数注入属性值时，需要定义以相应属性为参数的构造函数。使用构造函数注入的一个优点就是不需要为每一个属性都定义方法，只通过一个多参构造函数就可以一次注入。同样需要首先创建 Bean 程序代码，其具体代码为：

```
01  package com.sanqing.service;
02  public class Hello2ServiceImpl {
03      private String message;              //打招呼的信息
04      public Hello2ServiceImpl(String message) {  //具有属性参数的构造函数
05          super();
06          this.message = message;
07      }
08      public void sayHello(){      //打招呼方法
09          System.out.println("内容为："+message);
10      }
11  }
```

接下来就是使用 MyEclipse 中集成的功能在 Spring 的配置文件中对该 Bean 程序进行配置。同样将鼠标焦点定义在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，在其中输入必要信息后，如图 8-9 所示。

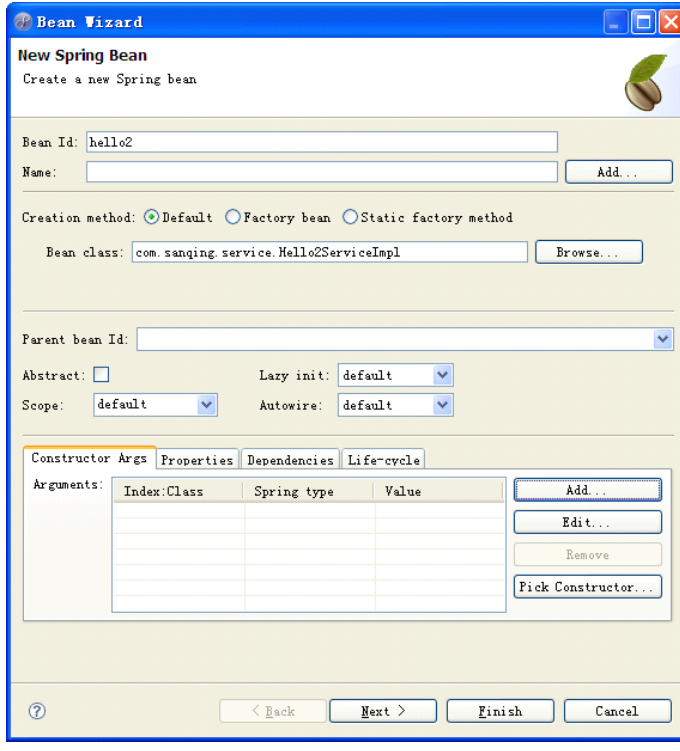


图 8-9　配置构造函数注入的 Bean

在其中填写“Bean Id”和“Bean Class”后，在最下面选择“Constructor Args”选项卡，单击其中的“Add”按钮，将弹出配置构造函数属性的界面，在其中选择配置类型为“value”，并输入要注入的值，如图 8-10 所示。

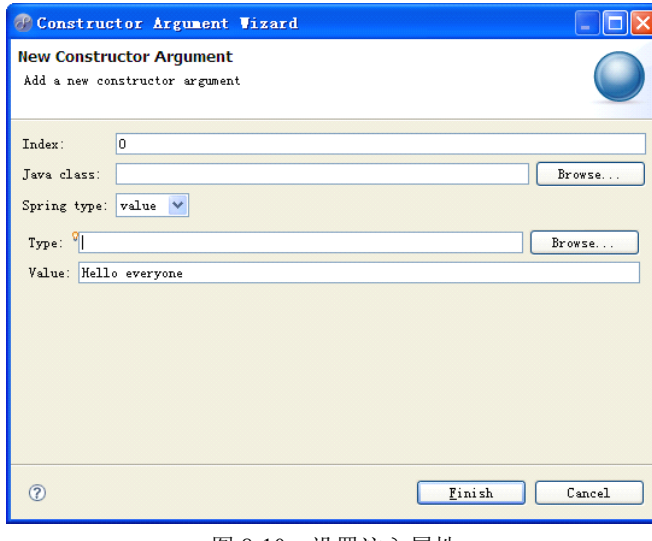


图 8-10　设置注入属性

其中“Index”表示属性对应构造函数中的第几个参数，它是从 0 开始的。“Java class”表示属性对应构造函数中参数的类型。

技巧：上述两种方式选择一种就可以，建议选择“Index”的方式，从而可以避免

参数类型相同情况的麻烦。这两个选项也可以同时填写，但是一定要全部和构造函数中的参数对应正确。

“Spring type”表示注入类型，因为这里是注入基本数据类型，所以选择“value”。“Type”表示注入值的类型，对于基本数据类型和 String 类型来说可以不填，它们是能够根据属性类型自动转换的。“Value”表示注入基本类型值。单击“Finish”按钮，将完成属性的设置。

回到配置 Bean 界面后，单击“Finish”按钮，将完成配置操作，将自动向 Spring 配置文件中加入配置代码，其加入的代码为：

```
<bean id="hello2" class="com.sanqing.service.Hello2ServiceImpl"
    abstract="false" lazy-init="default" autowire="default"
    dependency-check="default">
    <constructor-arg index="0">
        <value>Hello everyone</value>
    </constructor-arg>
</bean>
```

和通过 Setter 方法的方式注入不同的是从第 4 行到第 6 行，构造函数方式注入将采用“<constructor-arg>”标记，其中 index 属性对应的就是构造函数第几个参数，在标记对中使用“<value>”标记，在其中给出具体值。

开发完通过构造函数注入的 Bean 程序后，同样要创建测试类程序，测试程序的正确性，测试类代码如下：

```
package com.sanqing.test;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.sanqing.service.Hello2ServiceImpl;
public class Hello2Test {
    public static void main(String[] args) {
        BeanFactory factory =
            new ClassPathXmlApplicationContext("applicationContext.xml");
        Hello2ServiceImpl hello2=
            (Hello2ServiceImpl)factory.getBean("hello2");    //获取 Bean 实例
        hello2.sayHello();    //调用业务方法
    }
}
```

从代码上可以看到在测试类中只需要改动实例化 Bean 的 id，然后转换为相应的 Bean 类就可以。运行该程序，运行结果为：

```
内容为：Hello everyone
```

可以看到，通过构造函数注入也是正确的。

8.4.2 注入其他 Bean

在前面的讲解中都是向 Bean 中注入字符串类型，对于基本数据类型来说，是和字符串类型完全一样的。在本小节中将继续来讲解如何向 Bean 中注入其他 Bean，在实际

开发中，使用最多的也是注入其他 Bean。

在上一小节中，已经通过基本类型讲解了注入的方式，对于其他类型来说，都是存在这两种注入方式的，它们是非常类似的，这里我们只通过 Setter 方法的方式进行注入讲解。

在实际开发中，使用 Spring 最多的地方就是将数据访问层注入到业务逻辑层中，在其中通过接口操作，从而达到分层的效果。数据访问层一定要有 DAO 接口和 DAO 实现类，其中 DAO 接口的代码是非常简单的，这里就不再给出，读者可以查看光盘。DAO 实现类代码如下：

```
01  package com.sanqing.dao;
02  public class UserDAOImpl implements UserDAO {
03      public boolean findUser(String name, String password) { //定义查询数据库方法
04          if ("Tom".equals(name) && "456123".equals(password)) { // 判断是否正确
05              return true;
06          } else {
07              return false;
08          }
09      }
10  }
```

其中第 3 行定义了查找用户的方法，这里仅仅是演示数据库的操作，判断参数值是否为指定内容。当内容相同时，表示能够查询到用户，从而能够正确登录；当内容不同时，表示不能查询到用户，也就是不能完成登录。

接下来继续开发业务逻辑层的程序，在其中定义登录业务方法，在方法中调用数据访问层中的方法，其代码为：

```
01  package com.sanqing.service;
02  import com.sanqing.dao.UserDAO;
03  public class UserServiceImpl {
04      private UserDAO userDAO;
05      public void setUserDAO(UserDAO userDAO) {
06          this.userDAO = userDAO;
07      }
08      public boolean Login(String name, String password) { // 定义登录业务方法
09          boolean b = userDAO.findUser(name, password); // 调用 DAO 方法
10          return b;
11      }
12  }
```

开发完数据访问层和业务逻辑层的 Bean 程序后，接下来就是最关键的在 Spring 配置文件中对它们进行配置。首先对数据访问层 Bean 程序进行配置。将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，在其中输入配置数据访问层程序的信息，如图 8-11 所示。

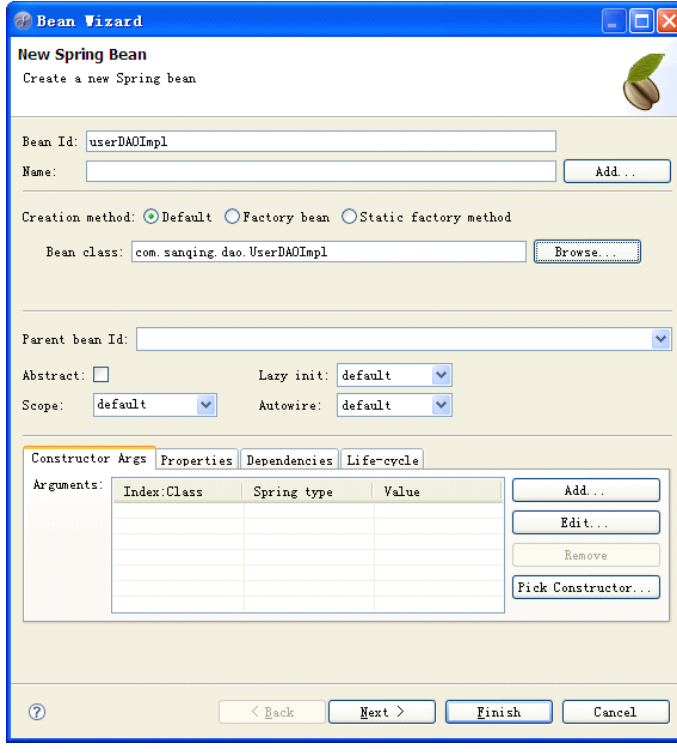


图 8-11　配置数据访问层 Bean

因为在数据访问层中并不需要注入任何属性内容，所以仅给出“Bean Id”和“Bean class”选项值就可以。单击“Finish”按钮，从而完成配置。

再次进行同样的操作，弹出对 Bean 程序进行配置的界面，在其中输入配置业务逻辑层程序的信息，如图 8-12 所示。

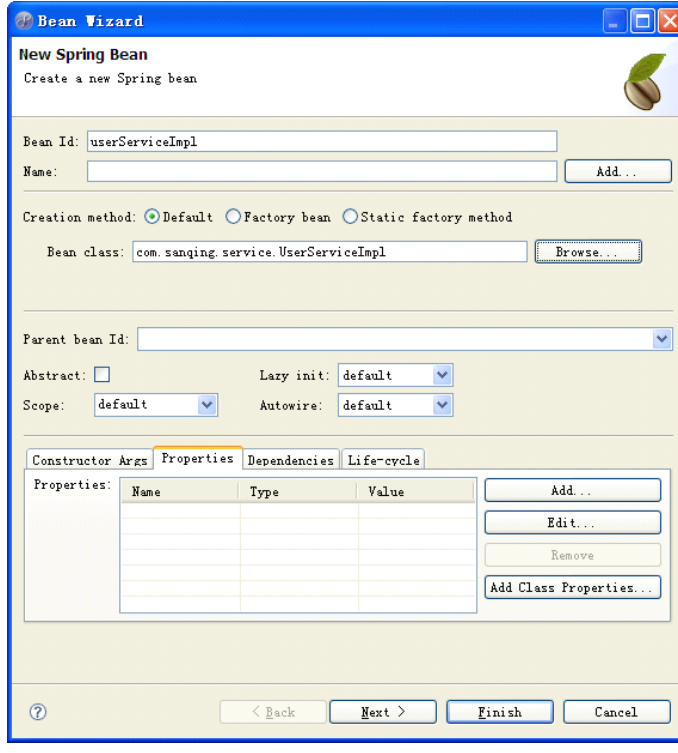


图 8-12　配置业务逻辑层 Bean

在其中配置“Bean Id”和“Bean class”选项，选择“Properties”选项卡，单击“Add”按钮，将弹出配置属性的界面。其中最关键的就是配置类型，“ref”对应的就是注入其他 Bean，选择该选项后，并填写其他信息，界面如图 8-13 所示。

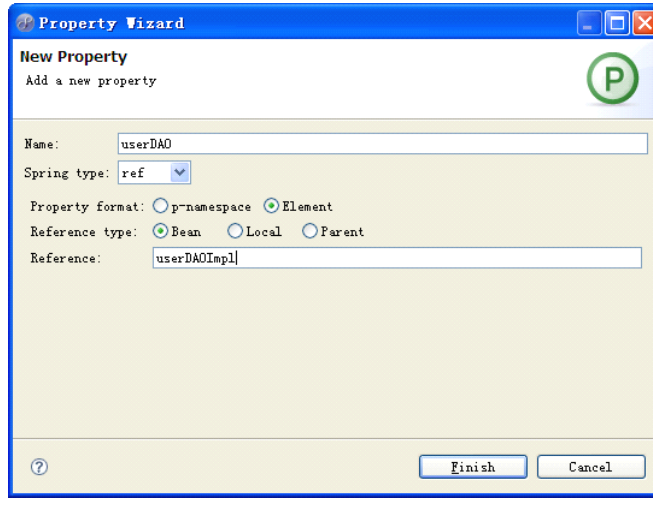


图 8-13　配置注入的 Bean

其中“Name”选项对应的属性名称，这和注入基本类型是一样的。“Reference type”表示注入其他 Bean 的形式，其中包含三种形式。“Bean”选项对应最常用的形式，通过它可以注入同一容器或者父容器中的 Bean；“Local”表示只能注入同一文件中的

Bean；“Parent”表示注入父容器中的 Bean。

说明：在 Spring 框架中可以定义多个配置文件，在每一个配置文件中完成一类 Bean 的配置，这对于大型项目来说是非常必要的。在配置文件中，可以使用<import>标记导入其他配置文件，从而产生父容器和子容器。

“Reference”表示要注入到当前 Bean 中的 Bean，这里填写注入 Bean 的 id。单击“Finish”按钮，将完成注入其他 Bean 的操作。回到配置业务逻辑层 Bean 的界面，单击“Finish”按钮，将完成所有配置。在 Spring 配置文件中，自动生成的代码为：

```
01      <bean id="userDAOImpl" class="com.sanqing.dao.UserDAOImpl"
02          abstract="false" lazy-init="default" autowire="default"
03          dependency-check="default">
04      </bean>
05      <bean id="userServiceImpl"
06          class="com.sanqing.service.UserServiceImpl" abstract="false"
07          lazy-init="default" autowire="default" dependency-check="default">
08          <property name="userDAO">
09              <ref bean="userDAOImpl" />
10          </property>
11      </bean>
```

其中第 1 行中完成数据访问层实体类的配置。从第 5 行到第 11 行完成业务逻辑层实体类的配置，其中第 8 行使用<property>标记指定为 Setter 方法的方式注入，注入的属性是“userDAO”，第 9 行中定义注入 Bean 的 id，也就是上面配置的数据访问层实体类。

和前面一样，最后创建测试类来验证注入其他 Bean 是否正确，它的代码为：

```
01  package com.sanqing.test;
02  import org.springframework.beans.factory.BeanFactory;
03  import org.springframework.context.support.ClassPathXmlApplicationContext;
04  import com.sanqing.service.UserServiceImpl;
05  public class OtherBeanClient {
06      public static void main(String[] args) {
07          BeanFactory factory =
08                  new ClassPathXmlApplicationContext("applicationContext.xml");
09          UserServiceImpl userService=
10              (UserServiceImpl)factory.getBean("userServiceImpl");//获取业务实现类
11          String name="Tom";                  //测试使用用户名
12          String password="456123";       //测试使用密码
13          System.out.println(name+"是否能够正常登录："
14                  +userService.Login(name, password));            //调用业务方法
15      }
16  }
```

其中第 9 行获取业务逻辑层实现类，然后给出测试数据，在第 14 行中调用 Login 业务方法判断用户是否能够登录。在 Login 方法中将调用数据访问层实现类中的方法，如果注入成功，将能够成功调用。运行该程序，运行结果为：

Tom 是否能够正常登录：true

从运行结果中可以看到，向业务逻辑层中注入数据访问层成功。

8.4.3 注入集合

在 Java 中，集合包括 List、Set 和 Map 三种，在 Spring 除了能够对这三种集合能够注入操作外，还包括 Properties 映射文件。在 Spring 配置文件中，使用<list/>、<set/>、<map/>及<props/>标记来对应集合进行配置。首先创建 Bean 程序，在其中具有着四种集合属性，其具体代码为：

```
01  package com.sanqing.service;
02  import java.util.List;
03  import java.util.Map;
04  import java.util.Properties;
05  import java.util.Set;
06  public class CollectionBean {
07      private List list;
08      private Set set;
09      private Map map;
10      private Properties props;
11      public void setList(List list) {
12          this.list = list;
13      }
14      public void setSet(Set set) {
15          this.set = set;
16      }
17      public void setMap(Map map) {
18          this.map = map;
19      }
20      public void setProps(Properties props) {
21          this.props = props;
22      }
23  }
```

然后在 Spring 配置文件中进行注入配置。将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，在其中输入 id 和 class 后，如图 8-14 所示。

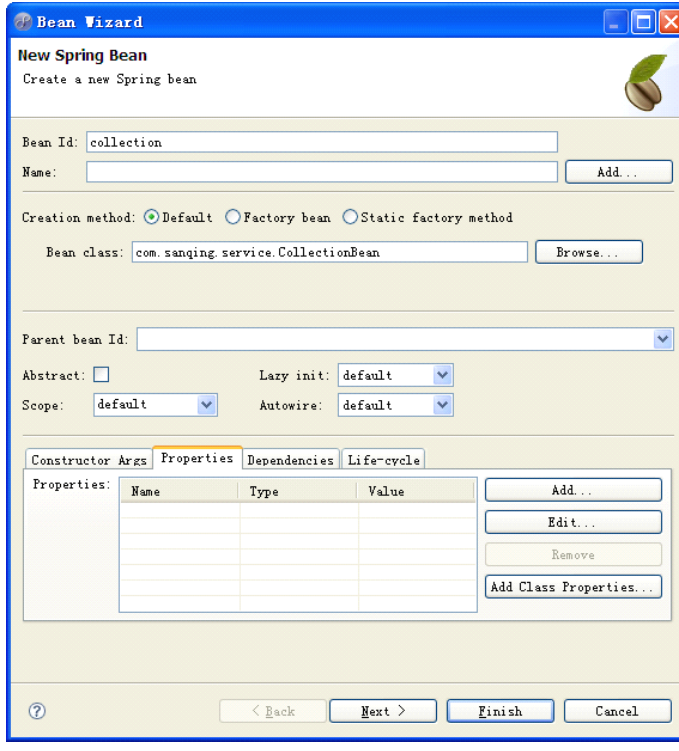


图 8-14　注入集合 Bean

在其中选择“Properties”选项卡，单击“Add”按钮，将弹出配置属性界面，通过该操作依次配置四种属性。首先来看配置 List 的界面，如图 8-15 所示。

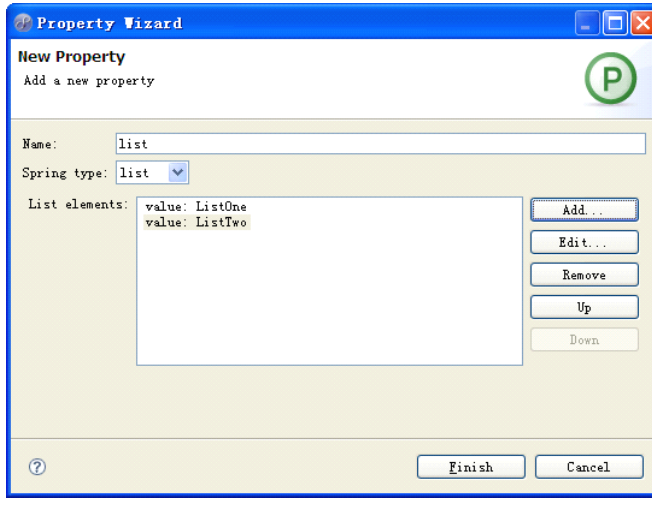


图 8-15　配置 List 属性

其中单击“Add”按钮将弹出向 List 中加入元素的界面，在其中可以添加基本类型的 value 值，也可以添加其他 Bean，具体操作界面是和前面注入方式是相同的。这里为了简单，就采用基本类型演示。选中集合元素后，还可以通过右边的按钮进行操作，通过“Up”和“Down”按钮可以调整元素顺序。单击“Finish”按钮，将完成 List 的配置。

继续单击注入集合 Bean 界面中的“Add”按钮，对 Set 进行配置，界面如图 8-16 所示。

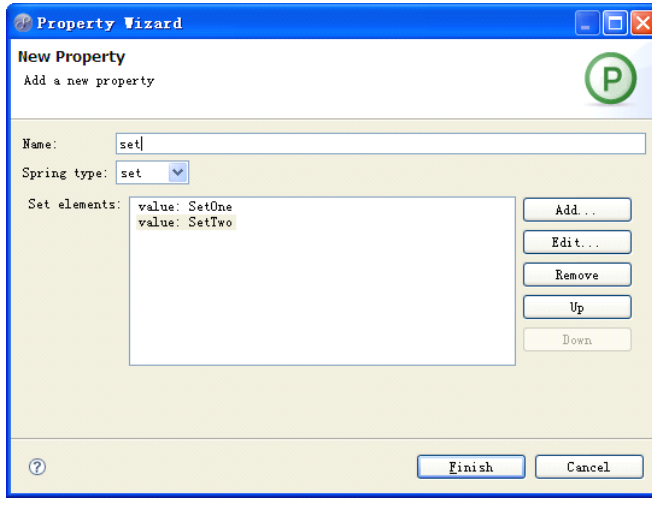


图 8-16　配置 Set 属性

向 Set 中添加元素的操作是和向 List 添加元素操作相同的，这里就不再重复讲解。继续单击注入集合 Bean 界面中的“Add”按钮，对 Map 进行配置，界面如图 8-17 所示。

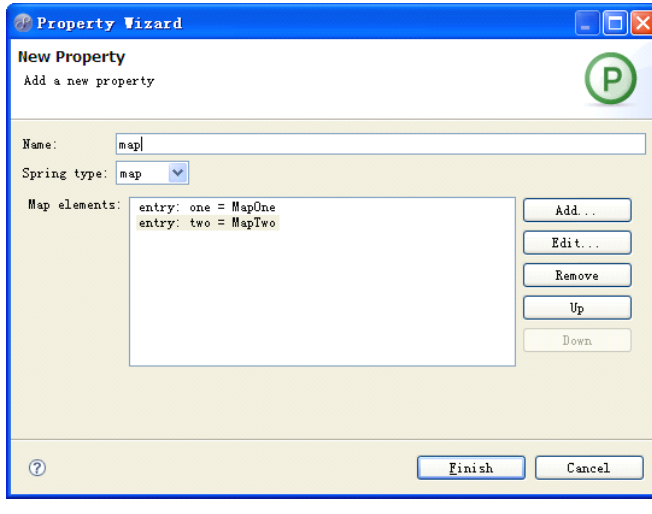


图 8-17　配置 Map 属性

在其中单击“Add”按钮，将弹出添加 Map 属性的界面，因为 Map 是以键值对存在的，所以要填写键和值。在配置 Map 属性界面中，单击“Finish”按钮，将完成配置操作。

继续单击注入集合 Bean 界面中的“Add”按钮，对 Properties 进行配置，界面如图 8-18 所示。

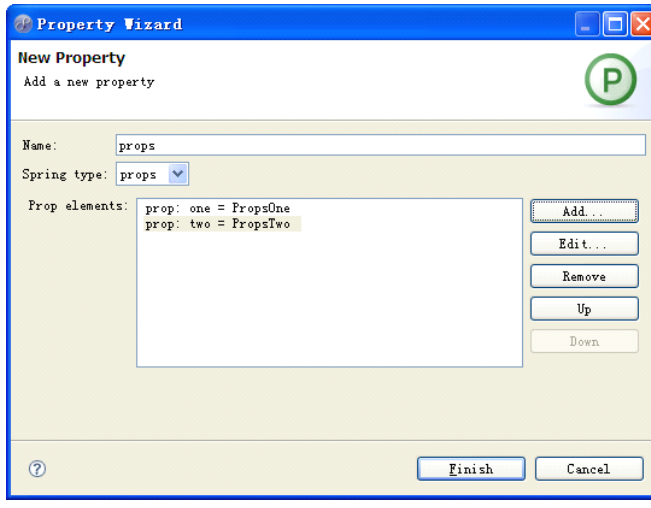


图 8-18　配置映射文件属性

配置映射文件属性和配置 Map 属性非常类似的，它也是键值对的形式。单击“Finish”按钮，完成映射文件的配置。回到注入集合 Bean 的界面，单击“Finish”按钮，完成注入操作。在配置文件中将自动添加如下代码：

```
<bean id="collection" class="com.sanqing.service.CollectionBean"
    abstract="false" lazy-init="default" autowire="default"
    dependency-check="default">
    <property name="list">
        <list>
            <value>ListOne</value>
            <value>ListTwo</value>
        </list>
    </property>
    <property name="set">
        <set>
            <value>SetOne</value>
            <value>SetTwo</value>
        </set>
    </property>
    <property name="map">
        <map>
            <entry key="one" value="MapOne"></entry>
            <entry key="two" value="MapTwo"></entry>
        </map>
    </property>
    <property name="props">
        <props>
            <prop key="one">PropsOne</prop>
            <prop key="two">PropsTwo</prop>
        </props>
```

```
27              </property>
28          </bean>
```

注意：不同的集合之间使用的标记对是不一样的，包括标记对中使用的元素标记。所以使用 MyEclipse 集成的功能，要比手动写容易的多。

在测试类中，首先获取 Bean，然后通过 Bean 获取集合属性后，可以通过循环对它们进行遍历，从而得到每一个元素值，读者可以自己来完成测试类的编写。

8.4.4 不完全依赖

在前面学习注入其他 Bean 时，当 Bean 甲被依赖注入到 Bean 乙中时，在 Bean 乙中就可以使用 Bean 甲进行调用方法。在其中 Bean 甲是要在 Bean 乙之前实例化的。

但是实际开发中有这样一种情况，需要 Bean 甲在 Bean 乙之前实例化，但是在 Bean 乙中是不需要使用 Bean 甲的，这种情况下就可以使用不完全依赖注入。

在 MyEclipse 中也集成了进行不完全依赖的功能。例如这里完成这样一个功能，在执行登录前，要先执行打招呼的功能，这时候就需要在用户业务逻辑层中配置完成打招呼功能的 Bean。

将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，在其中选择“Dependencies”选项卡，如图 8-19 所示。

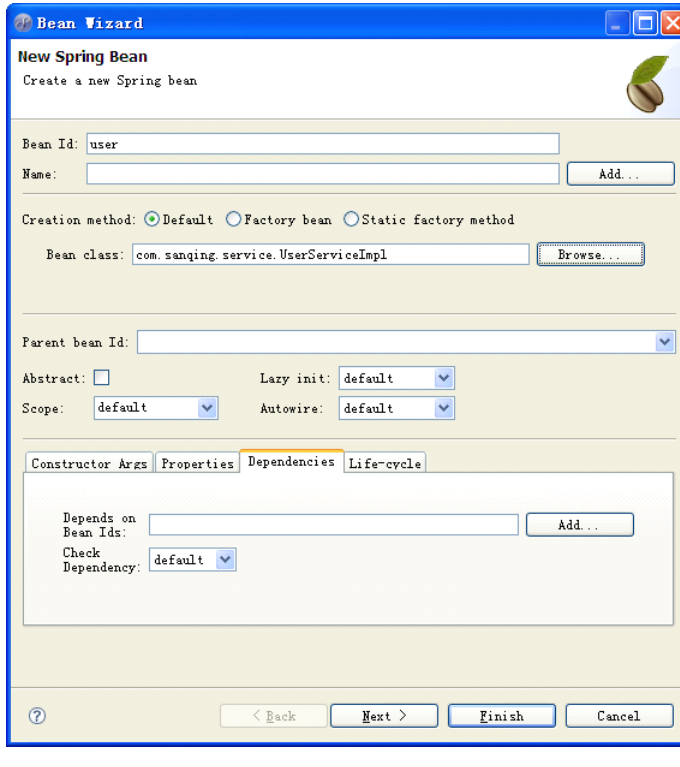


图 8-19　不完全依赖配置

在“Dependencies”选项卡中，“Depends on Bean Ids”选项就表示不完全依赖 Bean 的 id。这里可以自己填写，也可以单击“Add”方法添加，如图 8-20 所示。

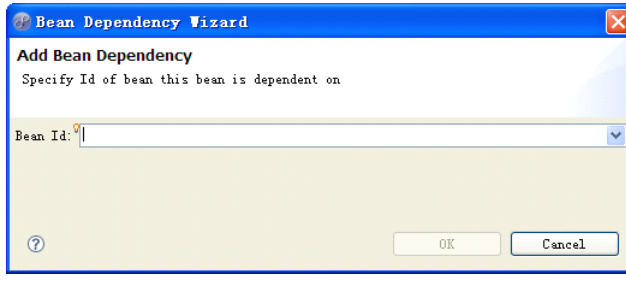


图 8-20　选择不完全依赖 id

在界面的下拉列表中，将出现目前 Spring 配置文件中所有的 id 值，选择“Hello”后，单击“OK”按钮，将完成 id 的选择，并将它添加到配置界面的“Depends on Bean Ids”选项中。id 操作可以添加多次，例如再添加“hello2”，id 之间将以逗号分隔。单击“Finish”按钮，将完成不完全依赖的配置。在 Spring 中，自动生成的代码为：

```
01        <bean id="user" class="com.sanqing.service.UserServiceImpl"
02              abstract="false" lazy-init="default" autowire="default"
03              dependency-check="default" depends-on="hello,hello2">
04        </bean>
```

其中“depends-on”属性配置的就是不完全依赖 Bean 的 id 值。

说明：不完全依赖的情况在数据库开发中比较常见，例如注册数据驱动必须在所有操作之前完成，但是在操作中是不会调用驱动的。

8.4.5　延迟初始化 Bean

在 Spring 框架中，默认情况下当启动项目后，会将所有已经配置的单例 Bean 全部实例化，其中包括配置的和注入的。这样做的好处就是在启动时，如果配置文件中有什么错误，将会立刻发现。

但是这样做，并不是适合所有情况的，在特殊情况下，我们并不想让 Bean 在启动的时候就马上实例化。这时候也可以通过配置，将它设置为延迟初始化。在 MyEclipse 的配置界面中，集成了该功能，例如我们将 Hello 程序设置为延迟初始化 Bean，它的配置界面如图 8-21 所示。

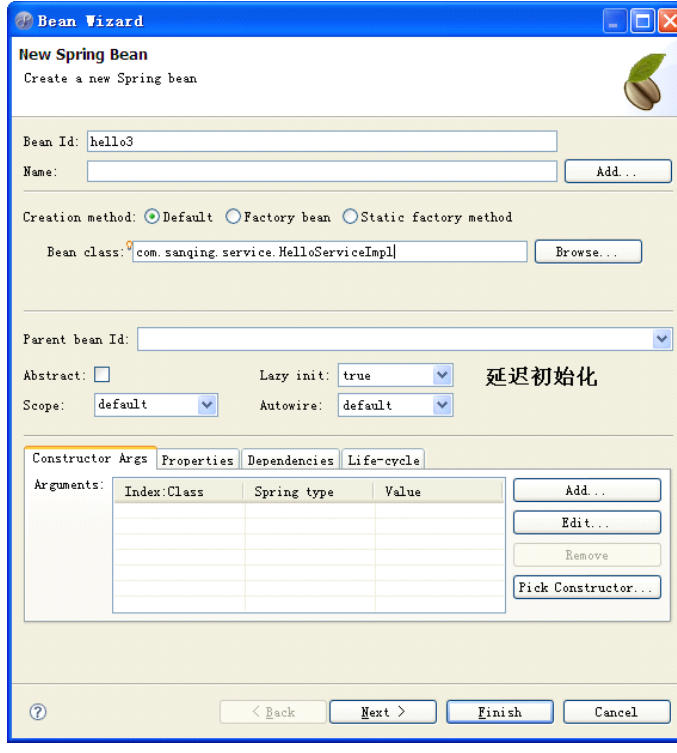


图 8-21　延迟初始化 Bean

其中“Lazy init”选项表示是否延迟初始化 Bean，默认选项是“default”，表示根据实际 Bean 类型决定。在其中选择“true”表示延迟初始化，选择“false”表示提前初始化。

单击“Finish”按钮，将完成初始化的设置，在 Spring 配置文件中自动生成的代码为：

```
01        <bean id="hello3" class="com.sanqing.service.HelloServiceImpl"
02              abstract="false" lazy-init="true" autowire="default"
03              dependency-check="default">
04        </bean>
```

其中“lazy-init”属性就表示是否延迟初始化，设置为“true”，就表示延迟初始化。

注意：延迟初始化的设置有时可能会失效。当 Bean 甲设置为延迟初始化时，但是非延迟初始化的 Bean 乙依赖于 Bean 甲，那 Bean 甲的延迟初始化的设置就会失效。

8.4.6　自动装配

在前面讲解注入其他 Bean 时，都是通过直接配置的方式注入的。在 Spring 框架的容器中，还可以通过自动装配的方式，让容器自动进行注入其他 Bean 的操作。在 MyEclipse 中，也集成了自动装配的功能。我们再次以用户业务逻辑实现类为例，在配置它的弹出界面，选择自动装配，界面如图 8-22 所示。

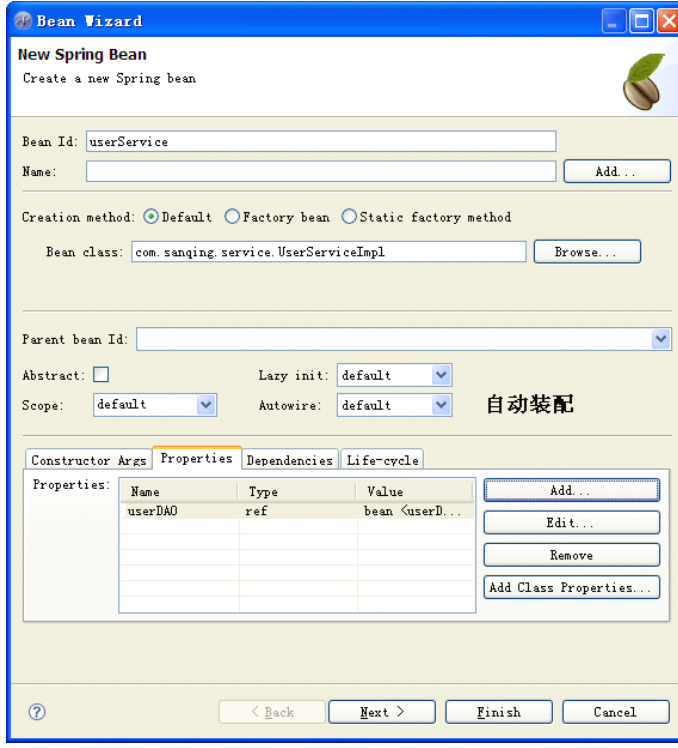


图 8-22　自动装配

其中“Autowire”表示的就是自动装配的方式。其中包括 5 个选项，“no”选项表示不使用自动装配，它是默认方式，当选择该方式时，必须对 Bean 进行配置。除此之外，还有四个选项，它们分别表示内容如下：

- byName：根据属性名自动装配。此选项将检查容器并根据名字查找与属性完全一致的 bean，并将其与属性自动装配。
- byType：如果容器中存在一个与指定属性类型相同的 bean，那么将与该属性自动装配。如果存在多个该类型的 bean，那么将会抛出异常，并指出不能使用 byType 方式进行自动装配。若没有找到相匹配的 bean，则什么事都不发生，属性也不会被设置。
- constructor：与 byType 的方式类似，不同之处在于它应用于构造器参数。如果在容器中没有找到与构造器参数类型一致的 bean，那么将会抛出异常。
- autodetect：通过 bean 类的自省机制（introspection）来决定是使用 constructor 还是 byType 方式进行自动装配。如果发现默认的构造器，那么将使用 byType 方式。

这里先选择“no”选项，单击“Finish”按钮，将完成自动装配的配置，在 Spring 配置文件中将自动生成如下代码：

```
01        <bean id="userService" class="com.sanqing.service.UserServiceImpl"
02            abstract="false" lazy-init="default" autowire="no"
03            dependency-check="default">
04            <property name="userDAO">
05                <ref bean="userDAOImpl" />
```

```
06              </property>
07          </bean>
```

其中第 2 行的“autowire”属性配置的就是自动装配的方式。

注意：目前自动装配仅对于注入其他 Bean 的操作有效，对于基本类型是没有任何作用的。

说明：自动装配既有优点又有缺点。优点是能够减少配置数量，缺点是使配置文件不再能够明显 Bean 之间的注入关系。

8.4.7 依赖注入检查

通过依赖注入检查可以，可以对 Spring 配置文件中的注入配置进行检查，当需要确保所有属性值或者属性类型正确配置时，使用该操作是非常必要的。和它相对的，在 Bean 中一些属性具有默认值，或者该属性并不在所有环境下都被调用，这时候就不需要进行依赖注入检查。所以就需要对每一个 Bean 进行各自特有的依赖注入检查设置。

在 MyEclipse 中，也集成了对依赖注入检查设置的功能。在弹出的配置 Bean 程序的界面中，选择“Dependencies”选项卡，如图 8-23 所示。

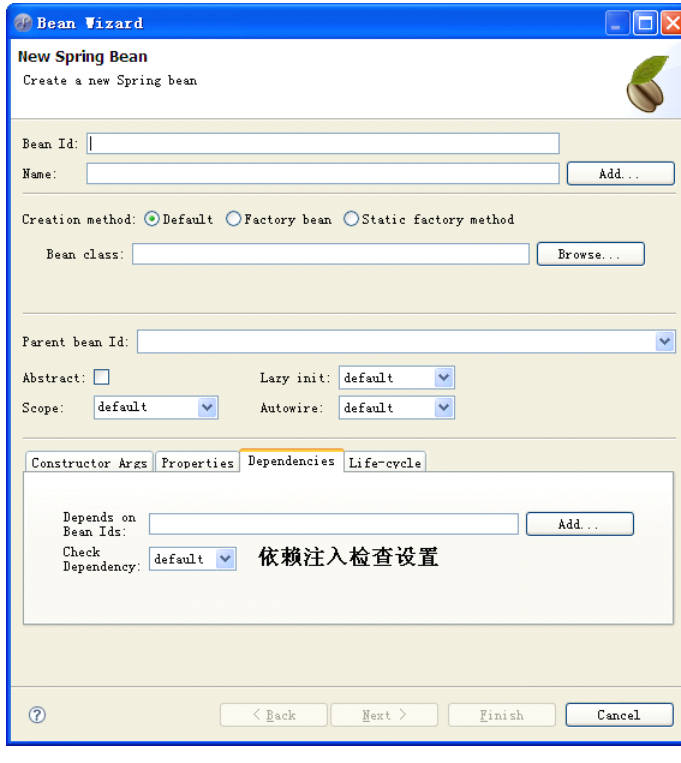


图 8-23　依赖注入检查

其中“Check Dependency”表示的就是依赖注入检查设置，其中包括 4 个选项。它们分别表示的含义如下：

- none：不进行依赖注入检查。

- simple：对基本类型和集合进行依赖注入检查。
- object：当注入其他 Bean 时，进行依赖注入检查。
- all：对所有类型都进行依赖注入检查。

在 Spring 的配置文件中，将采用“dependency-check”属性对依赖注入检查进行配置。

8.4.8 Bean 作用域设置

在 Spring 配置文件中创建 Bean 时，有时候还要设置 Bean 的作用域。Bean 的作用域就是指一个 Bean 对象的使用方式和使用范围。Spring 中的这种设计使定义 Bean 的作用域不用在类程序中进行，而是在配置文件中进行，大大提高了灵活性。

在 MyEclipse 中，也集成了对 Bean 作用域进行设置的功能。在配置 Bean 的界面中，“Scope”选项就是用来对 Bean 作用域进行设置，如图 8-24 所示。

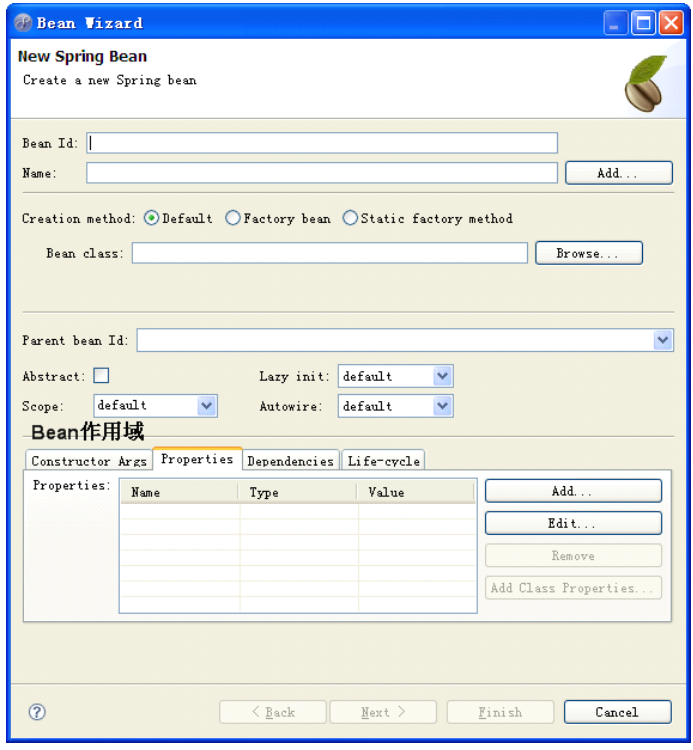


图 8-24　设置 Bean 作用域

在“Scope”的下拉列表中，有 5 个选项，它们分别表示的含义如下：

- singleton：它是 Bean 的默认作用域，使用该作用域时，在 Spring IoC 容器中将只会存在一个共享的 Bean 实例，所有的请求都只返回这个 Bean 实例。
- prototype：它和 singleton 作用域相对应，将为每一个请求返回不同的 Bean 实例。
- request：在一次 HTTP 请求中，一个 bean 定义对应一个实例；即每次 HTTP 请求将会有各自的 bean 实例，它们依据某个 bean 定义创建而成。
- session：在一个 HTTP Session 中，一个 bean 定义对应一个实例。
- global session：在一个全局的 HTTP Session 中，一个 bean 定义对应一个实例。

典型情况下，仅在使用 portlet context 的时候有效。

注意：其中 request、session 和 global session 的作用域只在 WEB 的项目中有效。

注意：单例 Bean 和设计模式中的单例模式是完全不同的。单例模式是指类只能创建一个对象。单例 Bean 是指在一个 IoC 容器中，创建一个 Bean 对象放在缓存中，所有的请求就是访问这一个对象。不同容器之间的对象是不同的。

这里以对 Hello Bean 的设置为例，将它设置为 prototype 作用域，在 Spring 的配置文件中自动生成的配置代码为：

```
01      <bean id="hello4" class="com.sanqing.service.HelloServiceImpl"
02          abstract="false" scope="prototype" lazy-init="default"
03          autowire="default" dependency-check="default">
04      </bean>
```

其中第 2 行的“scope”属性设置的就是 Bean 程序的作用域。

说明：如果配置不是在 Web 项目中，也可以使用 singleton 属性进行配置。当属性值为“true”时，表示使用 singleton 作用域；当属性值为“false”时，表示使用 prototype 作用域。

8.4.9 方法注入

在上一小节中，已经学习了 Bean 作用域的知识。在默认情况下，Ioc 容器中的所有 Bean 的作用域都是 singleton，我们也可以手动改变作用域。当改变 Bean 的作用域后，进行注入其他 Bean 操作时就可以产生问题，例如向一个作用域为 singleton 的 Bean 甲中注入作用域为 prototype 的 Bean 乙。对于 Bean 甲来说，容器只会创建一个 Bean 甲实例，这样就没法让容器提供不同的 Bean 乙实例。

为了解决这个问题，就需要放弃控制反转，而是采用方法注入的方式。方法注入分为两种，分别是 Lookup 方法注入和自定义方法注入。在 MyEclipse 中也集成了方法注入的操作，在配置 Bean 界面中，单击“Next”按钮，将进入方法注入界面，如图 8-25 所示。

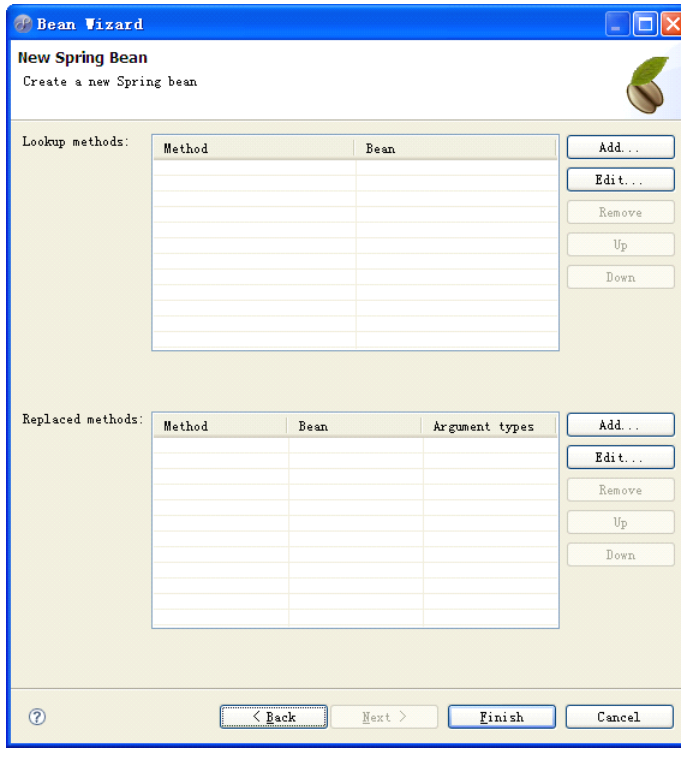


图 8-25　方法注入

其中“Lookup methods”栏表示的就是 Lookup 方法注入，而“Replaced methods”栏表示的就是自定义方法注入。单击“Lookup methods”栏中的“Add”按钮，将弹出添加 Lookup 方法的界面，如图 8-26 所示。

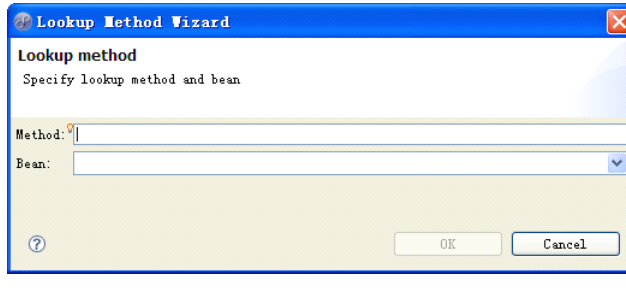


图 8-26　添加 Lookup 方法

其中“Method”表示获取 Bean 的方法名称，“Bean”表示要注入 Bean 的 id，也就是前面举例中的 Bean 乙的 id。

说明：在实际开发中，方法注入使用的并不多。并且自定义方法注入使用的比 Lookup 方法注入更少，所以这里就不在讲解自定义方法注入，有兴趣的读者可以自己参考相应资料。

8.4.10 定义 Bean 的生命周期方法

在 Spring 框架中的 Bean 程序和普通的 Java 类一样，也是具有生命周期的，包括初始化和销毁。在 Spring 框架中定义了相关生命周期接口，在定义 Bean 时可以实现这些接口，然后实现其中的生命周期方法，从而可以完成相应功能。但是这并不是建议使用的，因为这样就使程序和框架耦合在一起。

为了不让程序和框架耦合在一起，可以通过配置的方式定义自己的初始化和销毁等生命周期方法。在 MyEclipse 中，也集成了该功能。首先开发一个 Bean 程序，它的具体代码为：

```
01  package com.sanqing.service;
02  public class LifeBean {
03      private String message;              //打招呼的信息
04      public void setMessage(String message) {    //通过 Setter 方法注入
05          this.message = message;
06      }
07      public void sayHello(){      //打招呼方法
08          System.out.println("内容为："+message);
09      }
10      public void init(){              //初始化方法
11          System.out.println("进行 Bean 初始化操作");
12      }
13      public void destroy(){ //销毁方法
14          System.out.println("进行 Bean 销毁操作");
15      }
16  }
```

其中第 10 行的 init 方法就是自定义的初始化方法，第 13 行的 destrory 方法就是自定义的销魂方法。开发完 Bean 程序后，在 Spring 的配置文件中对生命周期方法进行配置，在配置界面中，选择“Life-cycle”选项卡，在其中填入相应的方法名，如图 8-27 所示。

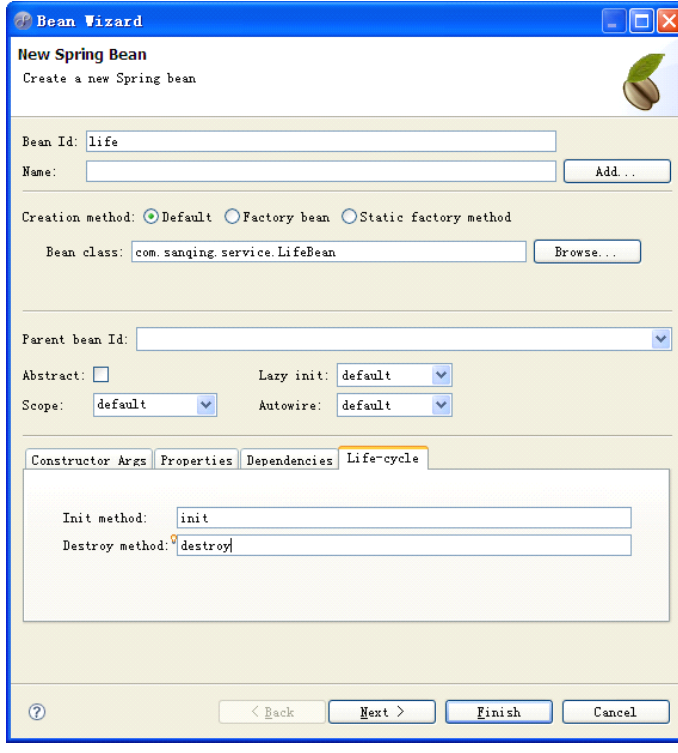


图 8-27　定义 Bean 生命周期方法

在“Life-cycle”选项卡中，“Init method”对应的就是初始化方法，这里选择定义在 Bean 程序中的“init”方法。“Destroy method”对应的就是销毁方法，这里选择同样定义在 Bean 程序中的“destroy”方法。单击“Finish”按钮，将完成 Bean 程序的配置，自动生成的代码为：

```
01      <bean id="life" class="com.sanqing.service.LifeBean"
02          abstract="false" lazy-init="default" autowire="default"
03          dependency-check="default" init-method="init"
04          destroy-method="destroy">
05          <property name="message">
06              <value>Hello everyone</value>
07          </property>
08      </bean>
```

其中第 3 行的“init-method”属性配置的就是初始化方法名。第 4 行中的“destroy-method”属性配置的就是销毁方法名。

最后就是开发测试类，来看一下开发的 Bean 程序和对它的配置是否正确，测试类代码为：

```
01  package com.sanqing.test;
02  import org.springframework.beans.factory.BeanFactory;
03  import org.springframework.context.support.ClassPathXmlApplicationContext;
04  import com.sanqing.service.LifeBean;
05  public class LifeTest {
06      public static void main(String[] args) {
```

```
07              BeanFactory factory =
08                      new ClassPathXmlApplicationContext("applicationContext.xml");
09              LifeBean life=(LifeBean)factory.getBean("life");   //获取 Bean 实例
10              life.sayHello();  //调用业务方法
11          }
12      }
```

运行该程序，运行结果为：

```
进行 Bean 初始化操作
内容为：Hello everyone
```

可以看到，当执行 Bean 程序中的业务方法前，首先执行了“init”方法，从而可以得知已经将该方法定义成初始化方法。当 Bean 被销毁时，还会执行“destroy”方法。

说明：在实际开发中在某些情况下，初始化和销毁方法是必须的，例如在非 Web 项目中关闭 IoC 容器操作，就需要在销毁方法中完成。

8.4.11 Bean 中的继承

继承是 Java 编程语言的三大特性之一，继承的类称为子类，被继承的类称为父类。在 Spring 的配置文件中，对 Bean 程序的配置也存在父子关系。在配置 Bean 时可以通过操作让它继承已有的配置 Bean 信息，它们分别被称为子 Bean 配置和父 Bean 配置。在子 Bean 配置中可以继承父 Bean 配置的数据，也可以覆盖原有数据或者添加新配置。

在 MyEclipse 中，也集成了进行集成配置的功能。在配置 Bean 界面中，“Parent bean Id”选项就是用来选择父 Bean 配置的 id，如图 8-28 所示。

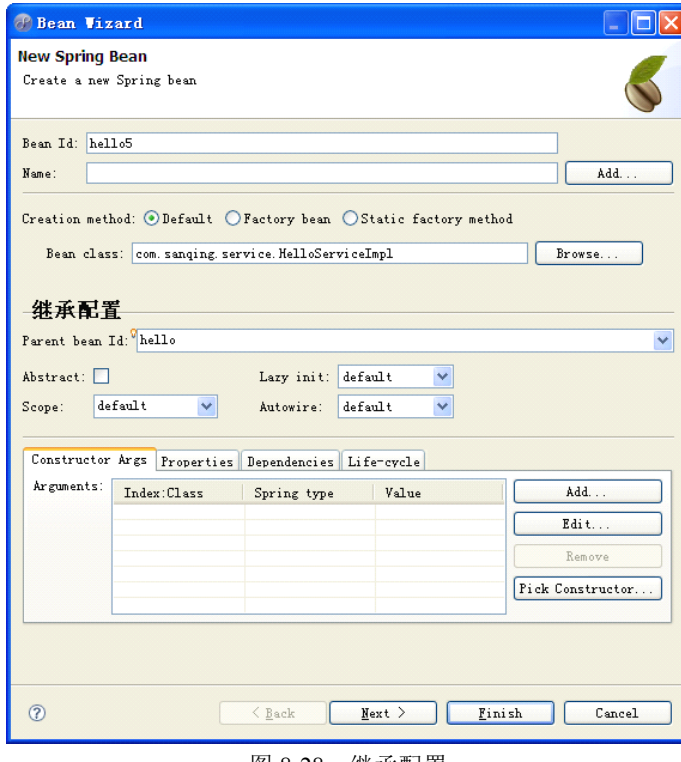


图 8-28　继承配置

单击“Finish”按钮，将完成继承配置的操作。在 Spring 的配置文件中，将自动生成如下代码：

```
01      <bean id="hello5" class="com.sanqing.service.HelloServiceImpl"
02          parent="hello" abstract="false" lazy-init="default" autowire="default"
03          dependency-check="default">
04      </bean>
```

其中第 2 行的“parent”属性就是进行的父 Bean 配置。使用该配置后，就不需要在其中对 message 属性进行配置，它会自动集成“hello”Bean 中对 message 属性的配置。

注意：本节中所说的继承，并不是指 Bean 类之间的继承关系，而是仅仅只配置文件中配置之间的继承关系。

在对 Bean 程序进行配置时，还可以选择其中的“Abstract”选项，当把该选项选中时，表示它的配置是抽象的。则该 Bean 将做为模板，被其他 Bean 配置继承。

注意：标识为“abstract”的 Bean 配置是不能够进行实例化的，当调用 getBean 方法时将发生错误。这种 Bean 的唯一作用就是让其他 Bean 配置继承。

8.5 面向切面编程

面向切面编程是 Spring 中的第二大功能，它的英文名称缩写是 AOP。使用面向对象编程，在指定方法的前面执行另一个方法是不难实现的，但是如果要求在同一个包下的所有类的方法前面都执行同一方法，就是一件比较复杂的工作。在这种情况下，如果使用面向切面编程就很容易实现。

在目前最新版本的 MyEclipse 中，并没有集成开发 AOP 项目的功能，所以在本节中主要通过手动编程程序的方式进行讲解。

8.5.1 准备工作

在“FirstSpring”项目中是不能够继续开发 AOP 项目的，因为在其中并没有加入 AOP 开发所需要的类库，所以这里单独再创建一个名称为“SpringAOP”的项目。

开发完普通的 Java 项目后，将执行加入 Spring 框架支持的操作，在添加类库界面中，不但要加入 Spring 核心类库，还要加入 AOP 开发的类库，如图 8-29 所示。

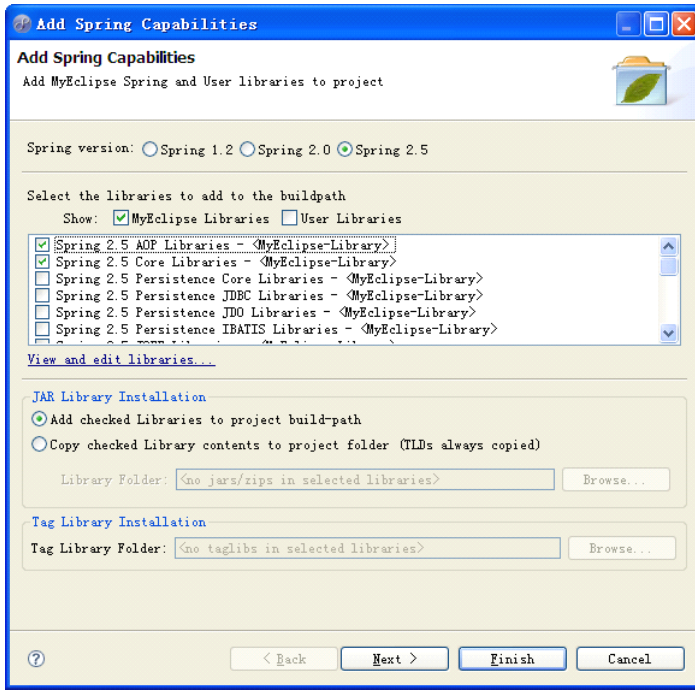


图 8-29　加入 AOP 类库

其中“Spring 2.5 AOP Libraries”选项就是 AOP 开发类库。因为并不需要对配置文件进行特殊设置，所以直接采用默认值。在该界面中单击“Finish”按钮，将完成加入 Spring 框架支持的操作。在包资源管理中展开“SpringAOP”项目，可以看到其中多出了 AOP 类库。

双击打开“ApplicationContext.xml”文件，也就是 Spring 的配置文件，在其中要加入 AOP 相关的 schemaLocation，加入后的完整代码为：

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans
03      xmlns="http://www.springframework.org/schema/beans"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xmlns:p="http://www.springframework.org/schema/p"
06      xmlns:aop="http://www.springframework.org/schema/aop"
07      xsi:schemaLocation="http://www.springframework.org/schema/beans
08          http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
09              http://www.springframework.org/schema/aop
10          http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">
11  </beans>
```

其中新加入的是第 6 行、第 9 行和第 10 行。这些代码可以通过 Spring 的下载安装包中找到对应文件，不过为了方便，读者只需要复制本书光盘中的该文件就可以进行开发了。

8.5.2　通过配置方式开发 AOP 项目

开发 AOP 项目由两种方式，分别是通过配置方式和通过注解方式，注解方式是 Java

5.0 版本以后增加的，它在下一小节中进行讲解。

面向切面编程的作用就是在某一类方法的特定位置，执行相同的功能。例如执行数据库时，不管执行什么操作都要执行关闭数据库的操作。首先来开发执行数据库操作的数据访问层程序，它的程序代码如下：

```
01  package com.sanqing.dao;
02  public class BookDAOImpl implements BookDAO {
03      public void addBook(String name, double price) {  //增加图书
04          System.out.println("执行增加图书操作");
05      }
06      public void deleteBook(int id) {                    //删除图书
07          System.out.println("执行删除图书操作");
08      }
09      public void updateBook(int id, String name, double price) {    //更新图书
10          System.out.println("执行更新图书操作");
11      }
12      public void findBookByBook(String name, double price) {    //根据 ID 查找图书
13          System.out.println("执行根据用户信息查询图书操作");
14      }
15      public void findBookById(int id) {                  //根据信息查找图书
16          System.out.println("执行根据用户 ID 查询图书操作");
17      }
18  }
```

在其中并没有写具体的数据库操作，而是通过显示信息来演示，这是不难理解的。继续开发面向切面编程类，它的代码为：

```
01  package com.sanqing.service;
02  public class AspectJXML {
03      public void myAfterMethod(){          //定义通知方法
04          System.out.println("数据库操作后，释放数据库资源");
05      }
06  }
```

在该类中第 3 行定义了通知方法，不管执行什么数据库操作方法时，都执行该方法。为了实现该功能，还要进行必要的配置，它是面向切面编程的核心，其中的配置代码为：

```
01  <bean id="bookDAOImpl" class="com.sanqing.dao.BookDAOImpl"
02      abstract="false" lazy-init="default" autowire="default"
03      dependency-check="default">
04  </bean>
05  <bean id="aspectJXML" class="com.sanqing.service.AspectJXML"
06      abstract="false" lazy-init="default" autowire="default"
07      dependency-check="default">
08  </bean>
09  <aop:config>
10      <aop:aspect id="myAspect" ref="aspectJXML">
11      <aop:pointcut id="allMothod" expression="execution(* com.sanqing.dao.*.*(..))" />
12          <aop:after pointcut-ref="allMothod" method="myAfterMethod" />
```

```
13          </aop:aspect>
14    </aop:config>
```

在其中第 1 行对数据访问层 Bean 进行配置，第 5 行对面向切面编程 Bean 进行配置，这些配置在前面讲解过。从第 9 行开发就是对面向切面编程进行配置，使用<aop:config>标记对包含相关配置。在第 10 行中使用<aop:aspect>标记对声明上面开发的面向切面编程类为切面。在第 11 行中声明了对哪些方法执行操作，这里指定对 dao 包下的所有方法执行。第 12 行中使用<aop:after>标记指定后置通知方法。

最后我们通过一个测试类，来验证通过配置方式是否达到我们想要的结果，测试类代码如下：

```
01    package com.sanqing.test;
02    import org.springframework.beans.factory.BeanFactory;
03    import org.springframework.context.support.ClassPathXmlApplicationContext;
04    import com.sanqing.dao.BookDAO;
05    public class BookTest {
06          public static void main(String[] args) {
07                BeanFactory factory =
08                      new ClassPathXmlApplicationContext("applicationContext.xml");
09                BookDAO bookDAO=
10                      (BookDAO)factory.getBean("bookDAOImpl");     //获取 DAO
11                bookDAO.updateBook(3,"Java", 58.5);         //执行更新图书操作
12                bookDAO.findBookByBook("Linux",68.5);  //执行根据信息查询图书操作
13          }
14    }
```

在代码中第 11 行执行了更新方法，在第 12 行中执行了查询方法，都是非常简单的。重要看一下执行结果，它的运行结果为：

```
执行更新图书操作
数据库操作后，释放数据库资源
执行根据用户信息查询图书操作
数据库操作后，释放数据库资源
```

可以看到，不管什么操作，在后面都会显示“数据库操作后，释放数据库资源”，它就是面向切面编程类中方法的执行结果，从这里可以看出通过配置方式开发 AOP 项目运行正确。

8.5.3 通过注解方式开发 AOP 项目

注解是 Java 5.0 版本中提出的新特性技术，通过注解可以是一个普通的 Java 程序完成特定的功能。也就是说当需要完成一个特定功能时，只需要对普通 Java 程序进行相应注解就可以完成。当使用注解方法开发时，要首先在配置文件中，加入如下配置代码：

```
<aop:aspectj-autoproxy/>
```

通过它可以启动 AspectJ 组件技术。在 Spring 的开发中要结合它来进行操作。

使用注解方式开发和使用配置方法开发最大的不同就是面向切面编程类，在该类中进行了面向切面中每一部分的注解，从而不再需要在配置文件中进行配置。这里我们来

看一个简单的面向切面编程类，它的代码为：

```
01  package com.sanqing.service;
02  import org.aspectj.lang.annotation.Aspect;
03  import org.aspectj.lang.annotation.Before;
04  import org.aspectj.lang.annotation.Pointcut;
05  @Aspect                          //声明切面
06  public class AspectJAnnotation {
07      @Pointcut("execution(* com.sanqing.dao.*.*(..))") //声明切入点
08      public void allMothod(){}                  //定义切入点名称
09      @Before("allMothod()")                     //声明通知
10      public void myBeforeMethod(){              //定义通知方法
11          System.out.println("开始进行数据库操作");
12      }
13  }
```

在该程序第 5 行中使用“@Aspect”注解声明该程序是一个切面。在第 7 行中使用“@Pointcut”声明了切入点，也就是对哪些方法进行操作，下面的第 8 行并不是一个类方法，而是切入点的名称，在通知中使用。第 9 行中使用“@Before”声明了前置通知，在其中指定切入点的名称，从而表明当执行该切入点中的方法时，执行第 10 行的通知方法。

通过这种方式完成面向切面的功能。其他程序是和前面配置方式的方法相似的，这里就不再详细讲解。

技巧：对于普通的 AOP 项目来说，最好使用注解的方式开发，它能够节省大量大量。但是当 Spring 需要整合其他技术时，所必须的 AOP 操作，习惯于使用配置文件的方式。

第9章　SSH框架整合开发

SSH 框架整合指的就是前面所学的 Struts、Spring 和 Hibernate 框架的整合，它们每两种框架之间都可以整合进行开发，在 Hibernate 一章中已经讲解了 Struts +Hibernate 的整合。在本章中将继续来讲解在 MyEclipse 中是如何集成整合开发功能的。

9.1　Spring+Hibernate 整合开发

Spring 框架和 Hibernate 框架在前面都已经讲解过了，它们都不局限在 Web 项目中。在 Hibernate 技术中，自动生成的 DAO 是不具有事务能力的，通过和 Spring 整合可以提供事务操作。在 Spring 中还定义了用于简化 Hibernate 操作的封装类，在本节中就对它们进行讲解。

9.1.1　创建整合项目

因为 Spring 和 Hibernate 技术都不局限于 Web 项目，所以这里我们通过一个简单的 Java 项目来演示它们之间的整合。首先创建一个名称为“SpringAndHibernate”的 Java 项目，然后就是向该项目中加入两种框架的技术支持。如果想让 Spring 来管理 Hibernate，它们的加入顺序是有严格规定的，必须先加入 Spring 框架支持，再加入 Hibernate 框架支持。

在包资源管理器中，选中“SpringAndHibernate”项目节点。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Spring Capabilities”命令，将弹出加入 Spring 类库支持操作界面，在其中选择 Spring 核心类库，如图 9-1 所示。

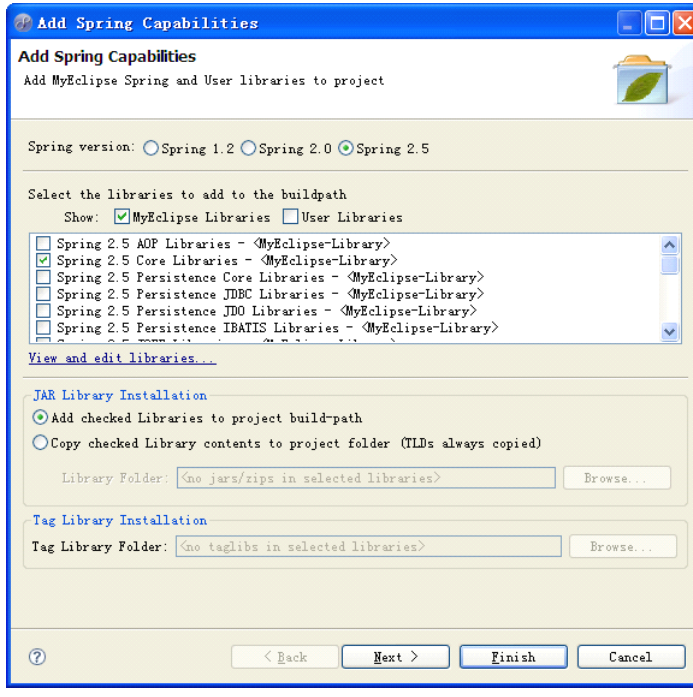


图 9-1　加入 Spring 类库

单击“Next”按钮，将进入对 Spring 配置文件进行设置的界面，如图 9-2 所示。

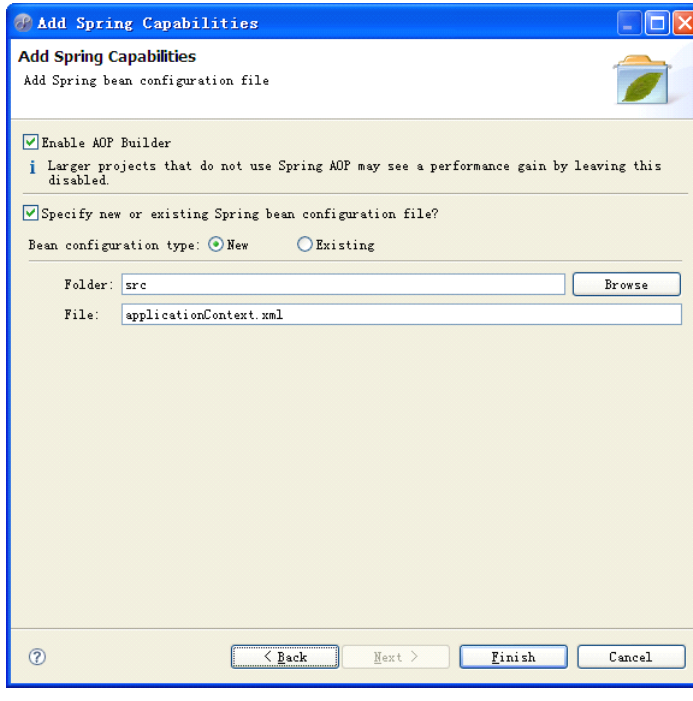


图 9-2　设置 Spring 配置文件

单击“Finish”按钮，完成 Spring 框架的技术支持。在包资源管理器中，再次选择“SpringAndHibernate”项目节点。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project

Capabilities”|“Add Hibernate Capabilities”命令，将弹出加入 Hibernate 类库支持的界面。在其中不但要加入 Hibernate 的核心类库，因为要整合 Spring，所以还要加入 Spring 的相关类库，如图 9-3 所示。

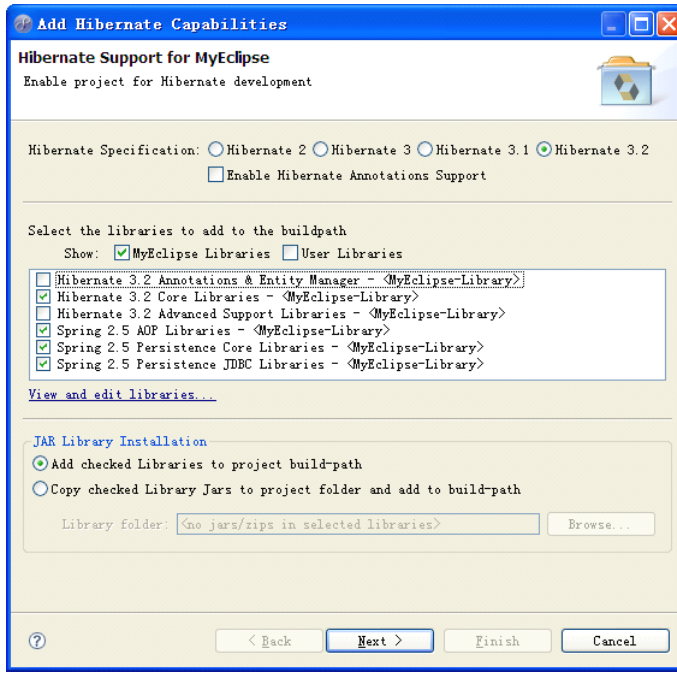


图 9-3　加入 Hibernate 相关类库

单击“Next”按钮，将进入对 Hibernate 相关信息进行管理的界面，如图 9-4 所示。

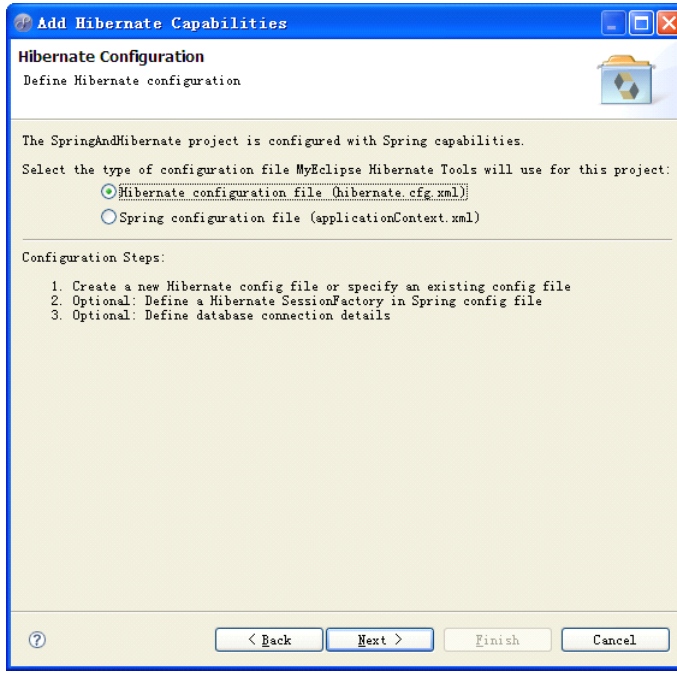


图 9-4　管理 Hibernate

其中“Hibernate configuration file (hibernate.cfg.xml)”选项表示创建 Hibernate 的配置文件，将 Hibernate 的相关配置内容放在其中。“Spring configuration file (applicationContext.xml)”选项表示将 Hibernate 的相关配置放到 Spring 配置文件中。

这两个选项的不同选择，将造成后面的配置和生成的项目结构有很大不同。在实际开发中，如果让 Hibernate 和 Spring 整合，通常让 Spring 管理 Hibernate 配置信息，所以选择第二个选项。单击“Next”按钮，将进入配置会话工厂类的界面，如图 9-5 所示。

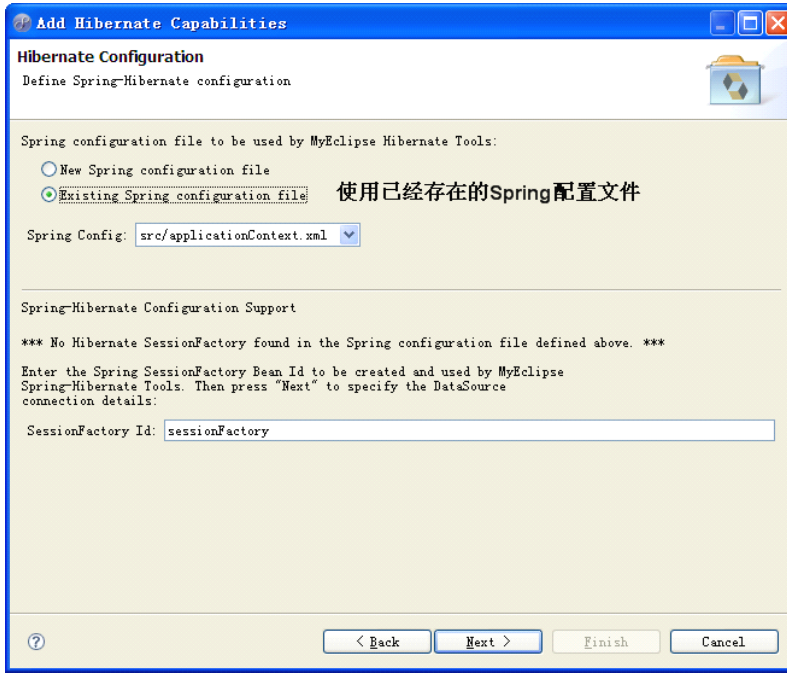


图 9-5　配置会话工厂类

界面最上面是两个单选按钮，“New Spring configuration file”表示新创建 Spring 的配置文件。“Existing Spring configuration file”表示使用已经存在的 Spring 配置文件，因为在前面已经加入过 Spring 的框架支持，所以选择该选项，并选择该文件的位置。

“SessionFactory Id”表示在 Spring 配置文件中为 Hibernate 中的会话工厂类配置的 Bean 程序 id，默认是“sessionFactory”，采用这个默认值就可以。单击“Next”按钮，将进入配置数据源的界面，选择数据库连接后，如图 9-6 所示。

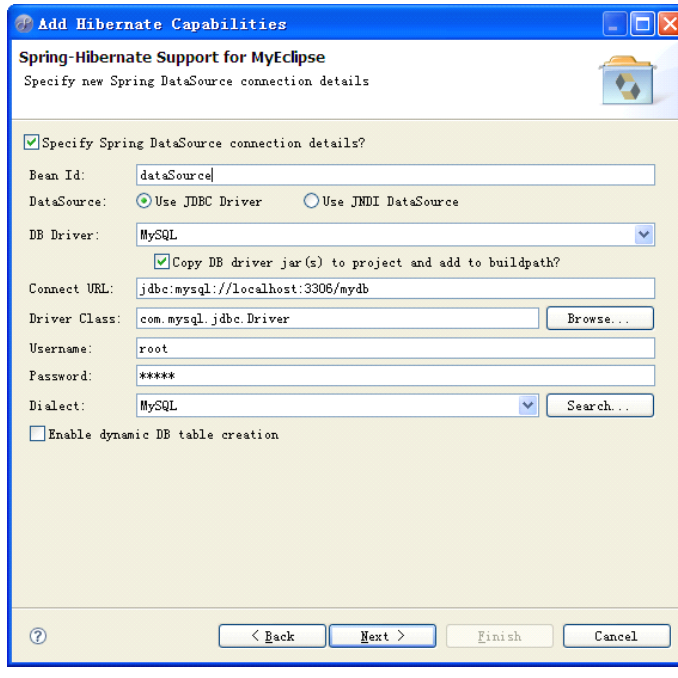


图 9-6　配置数据源

其中“Bean Id”表示数据源的 id 名，采用默认的“dataSource”就可以。“DataSource”表示数据源类型，这里选择最常用的“Use JDBC Driver”JDBC 驱动。“DB Driver”表示数据库连接选择，它是和数据库视图中数据库浏览器中的连接相对应的，这里选择“MySQL”连接。选择后，会自动将下面的 URL、驱动串、用户名、密码和方言填写好。单击“Next”按钮，将进入创建会话工厂类的界面，如图 9-7 所示。

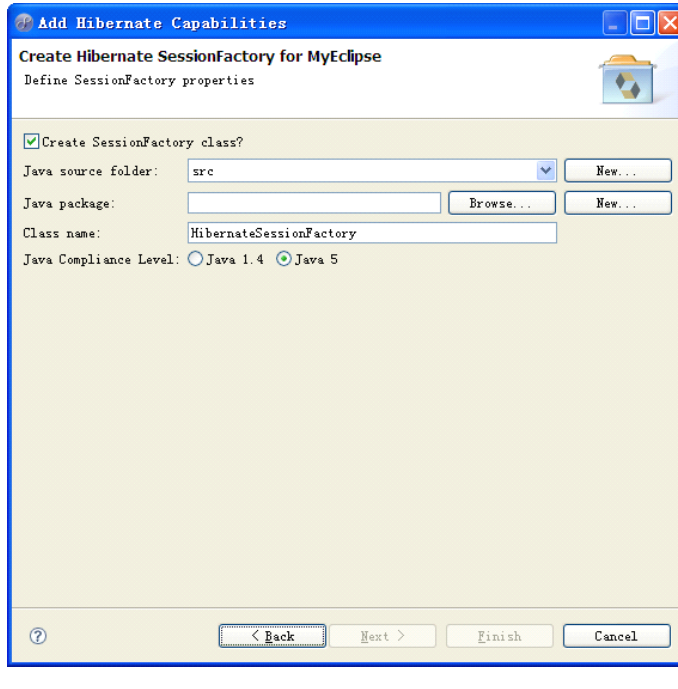


图 9-7　创建会话工厂类

该创建界面是和 Hibernate 项目中创建界面相同的。在其中需要为会话工厂类定义新创建一个包，这里仍然采用“hb”的包名。单击“Finish”按钮，将完成加入 Hibernate 框架支持的操作。

9.1.2　认识整合后的项目

向“SpringAndHibernate”项目中，依次加入 Spring 框架支持和 Hibernate 框架支持后，则 Spring+Hibernate 整合的项目就创建完成了。在包资源管理器中展开该项目，结构如图 9-8 所示。

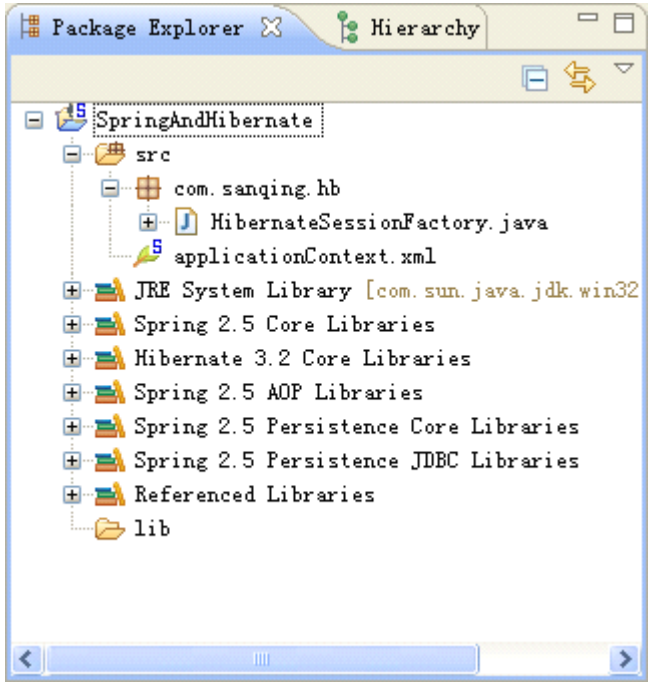


图 9-8　整合项目结构

在项目结构中可以看出，在其中导入了多个类库，这些都是前面操作中加入的。在“src”目录下，仅有 Spring 的配置文件，而没有 Hibernate 的配置文件，这是因为在加入框架支持操作中选择使用 Spring 配置文件对 Hibernate 进行管理。

其中的“HibernateSessionFactory”文件是会话工厂类，在 Hibernate 一章中已经对它进行了了解，这里就不再进行讲解。双击在编辑区中打开“applicationContext.xml”文件，它的具体代码如下：

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans
03      xmlns="http://www.springframework.org/schema/beans"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xmlns:p="http://www.springframework.org/schema/p"
06      xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
07                  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
08          <bean id="dataSource"
09                  class="org.apache.commons.dbcp.BasicDataSource">
10                  <property name="driverClassName"
11                          value="com.mysql.jdbc.Driver">
12                  </property>
13                  <property name="url" value="jdbc:mysql://localhost:3306/mydb"></property>
14                  <property name="username" value="root"></property>
15                  <property name="password" value="admin"></property>
16          </bean>
17          <bean id="sessionFactory"
18                  class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
19                  <property name="dataSource">
20                          <ref bean="dataSource" />
21                  </property>
22                  <property name="hibernateProperties">
23                          <props>
24                                  <prop key="hibernate.dialect">
25                                          org.hibernate.dialect.MySQLDialect
26                                  </prop>
27                          </props>
28                  </property>
29          </bean>
30  </beans>
```

其中的配置信息都是创建整合项目后自动生成的。从第 8 行到第 16 行是配置了数据源，向“org.apache.commons.dbcp.BasicDataSource”类中通过 Setter 方法的方式注入驱动类、URL、用户名和密码的信息值，这样在 Sprnig 的 IoC 容器中就可以创建数据源对象。

从第 17 行到第 29 行配置了会话工厂，向“org.springframework.orm.hibernate3.LocalSessionFactoryBean”类中通过 Setter 方法的方式注入数据源 Bean 和 Hibernate 属性信息。数据源 Bean 在上面已经配置了，Hibernate 属性的类型是映射文件集合，目前只有一个值，那就是数据库方言。当有其他属性配置时，可以以元素的形式加到该集合中。

9.1.3 开发整合的 DAO 程序

当 Spring 和 Hibernate 整合后，再来开发 DAO 程序，则生成的代码是和原只有 Hibernate 时生成的不一样的。同样要首先进入 MyEclipse 的数据库视图中，打开“MySQL”连接，在其中选择操作的数据表节点。例如这里仍然选择“mydb”数据库下的“student”数据表。单击鼠标右键，在其中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，填写必要信息后，如图 9-9 所示。

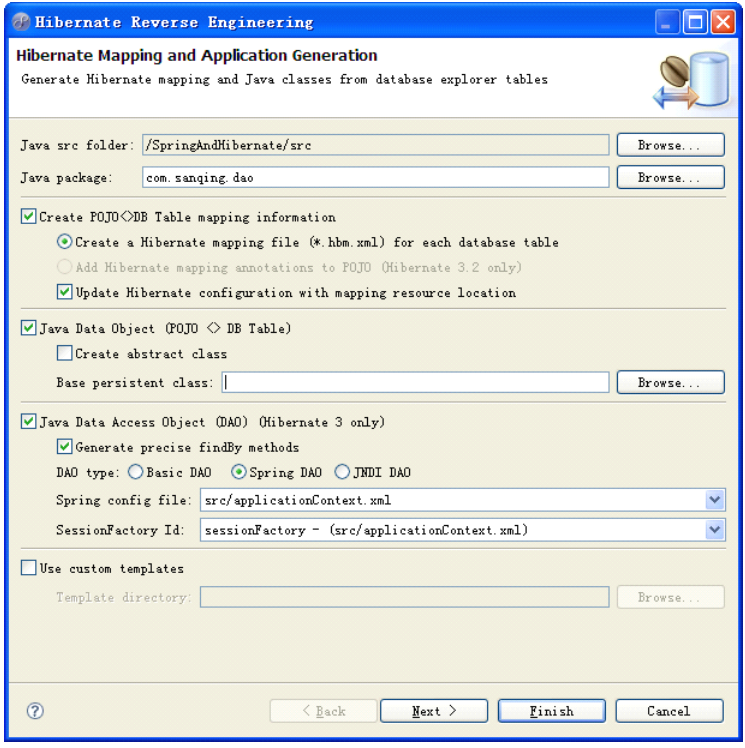


图 9-9　创建整合 DAO 程序

界面中的上半部分是和普通的 Hibernate DAO 程序没有区别的，这样就不再重复讲解。在“Java Data Access Object (DAO) (Hibernate 3 only)”选项中，在其中的 DAO 程序类型中，可以看到原来不能选择的“Spring DAO”选项，现在已经可以选择了。

“Spring config file”表示在哪个文件中对该 DAO 程序进行配置，默认的就是在 src 目录下的 Spring 配置文件。“SessionFactory Id”表示在 DAO 程序中注入的会话工厂类 id 值，并标明是在哪一个配置文件中。单击“Finish”按钮，将完成 Hibernate 程序的创建。

创建完成后，将在“SpringAndHibernate”项目中，创建一个名称为“com.sanqing.dao”的包，在该包下包括刚创建的实体类、映射文件和 DAO 程序。实体类和映射文件是没有变化的，我们主要来看一下 DAO 程序，它的部分代码为：

```
01  package com.sanqing.dao;
02  //省略导入接口和类的代码
03  public class StudentDAO extends HibernateDaoSupport {
04      private static final Log log = LogFactory.getLog(StudentDAO.class);
05      public static final String NAME = "name";
06      public static final String SCORE = "score";
07      protected void initDao() {
08          // do nothing
09      }
10      public void save(Student transientInstance) { //保存方法
11          log.debug("saving Student instance");
12          try {
```

```
13                  getHibernateTemplate().save(transientInstance);
14                  log.debug("save successful");
15              } catch (RuntimeException re) {
16                  log.error("save failed", re);
17                  throw re;
18              }
19          }
20          public void delete(Student persistentInstance) {      //删除
21              log.debug("deleting Student instance");
22              try {
23                  getHibernateTemplate().delete(persistentInstance);
24                  log.debug("delete successful");
25              } catch (RuntimeException re) {
26                  log.error("delete failed", re);
27                  throw re;
28              }
29          }
30          public Student findById(java.lang.Integer id) {      //根据 ID 查询
31              log.debug("getting Student instance with id: " + id);
32              try {
33                  Student instance = (Student) getHibernateTemplate().get(
34                          "com.sanqing.dao.Student", id);
35                  return instance;
36              } catch (RuntimeException re) {
37                  log.error("get failed", re);
38                  throw re;
39              }
40          }
41          public List findAll() {                  //查询全部
42              log.debug("finding all Student instances");
43              try {
44                  String queryString = "from Student";
45                  return getHibernateTemplate().find(queryString);
46              } catch (RuntimeException re) {
47                  log.error("find all failed", re);
48                  throw re;
49              }
50          }
51          public static StudentDAO getFromApplicationContext(ApplicationContext ctx) {
52              return (StudentDAO) ctx.getBean("StudentDAO");
53          }
54      }
```

其中在第 3 行中自动生成的 DAO 程序将继承“HibernateDaoSupport”类，它是 Spring 框架中用于操作 Hibernate 的类。在该类中提供了一个 setSessionFactory 方法来接收一个 SessionFactory 对象，从而完成 SessionFactory 的注入。在 setSessionFactory 方法中，通

过 SessionFactory 对象创建了 HibernateTemplate 对象。

HibernateTemplate 类中提供了那些简单的诸如 find、load、saveOrUpdate 或者 delete 操作的方法，还提供了可选择的快捷函数来替换回调的实现。

在 HibernateDaoSupport 类中还提供了 getHibernateTemplate 方法，所以在 DAO 程序的操作数据库的方法中，都会调用继承而来的 getHibernateTemplate 方法首先获取 HibernateTemplate 对象，然后调用相应的方法完成数据库操作。

自动生成 Hibernate 程序的同时，还会自动完成在 Spring 配置文件中的配置，配置代码如下：

```
01  <bean id="sessionFactory"
02      class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
03      <property name="dataSource">
04          <ref bean="dataSource" />
05      </property>
06      <property name="hibernateProperties">
07          <props>
08              <prop key="hibernate.dialect">
09                  org.hibernate.dialect.MySQLDialect
10              </prop>
11          </props>
12      </property>
13      <property name="mappingResources">
14          <list>
15              <value>com/sanqing/dao/Student.hbm.xml</value>
16          </list>
17      </property>
18  </bean>
19  <bean id="StudentDAO" class="com.sanqing.dao.StudentDAO">
20      <property name="sessionFactory">
21          <ref bean="sessionFactory" />
22      </property>
23  </bean>
```

其中第 15 行配置了映射文件，可以看到它做为集合中的一个元素注入到会话工厂类中。当有多个配置文件时，都会做为该集合的元素。在第 19 行中对生成的 DAO 程序 Bean 进行了配置，向其中注入了会话工厂 Bean，从而能够获得 HibernateTemplate 对象进行数据库操作。

9.1.4 事务管理

自动生成的 DAO 程序中是没有事务操作的，以前的做法是继承自动生成的 DAO 程序，在重写方法中加入事务操作。当整合 Spring 技术后，完全没有必要这样操作，我们可以通过 Spring 来对事务进行管理。

在 Spring 中进行事务管理有两种方式，分别是编程式事务管理和声明式事务管理，这里我们主要讲解最常用的声明式事务管理。使用声明式事务管理，可以无需修改项目

中的原代码，只需要在 Spring 配置文件中进行相应配置就可以。声明式事务管理的配置代码如下所示。

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans
03      xmlns="http://www.springframework.org/schema/beans"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xmlns:p="http://www.springframework.org/schema/p"
06      xmlns:aop="http://www.springframework.org/schema/aop"
07      xmlns:tx="http://www.springframework.org/schema/tx"
08      xsi:schemaLocation="http://www.springframework.org/schema/beans
09                  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
10                      http://www.springframework.org/schema/aop
11                  http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
12                      http://www.springframework.org/schema/tx
13                  http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">
14      <bean id="transactionManager"
15          class="org.springframework.orm.hibernate3.HibernateTransactionManager">
16          <property name="sessionFactory">
17              <ref bean="sessionFactory"/>
18          </property>
19      </bean>
20      <tx:advice id="txAdvice" transaction-manager="transactionManager">
21          <tx:attributes>
22              <tx:method name="*" propagation="REQUIRED"/>
23          </tx:attributes>
24      </tx:advice>
25      <aop:config>
26          <aop:pointcut id="allDaoMethod"
27                      expression="execution(* com.sanqing.dao.*.*(..))"/>
28          <aop:advisor pointcut-ref="allDaoMethod" advice-ref="txAdvice"/>
29      </aop:config>
30  </beans>
```

在进行声明式事务配置之前，首先要在 XML 文件头中加入相关的配置，不然下面的配置是会发生错误的。从第 14 行到第 19 行对 Hibernate 中的事物管理器进行配置。从第 20 行到第 24 行配置了对哪些方法进行事务管理，在其中第 22 行使用“*”表示所有方法。

说明：如果在 DAO 程序中有严格的方法命名规则，可以分别对增加、删除和修改方法进行事务管理，例如填写“delete*”就表示所有删除方法。对于查询方法来说是不需要进行事务管理的。

从第 25 行到第 29 行使用 AOP 配置方式配置了对哪一个包下的所有方法进行事务管理，这里填写的是 dao 包下。这一段配置在开发中可以复制使用，其中可能改变的只有第 27 行定义的需要事务处理的方法。

9.1.5 测试 Spring+Hibernate 整合项目

因为在 Spring 中大量使用到的代理模式，所以当获取实体类 Bean 对象时获取的是该类的代理对象，所以在测试前要首先进行抽取接口的操作。

在包资源管理器中，选中“StudentDAO”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，在其中输入接口名为“IStudentDAO”，然后将其中的所有方法选中，界面如图 9-10 所示。

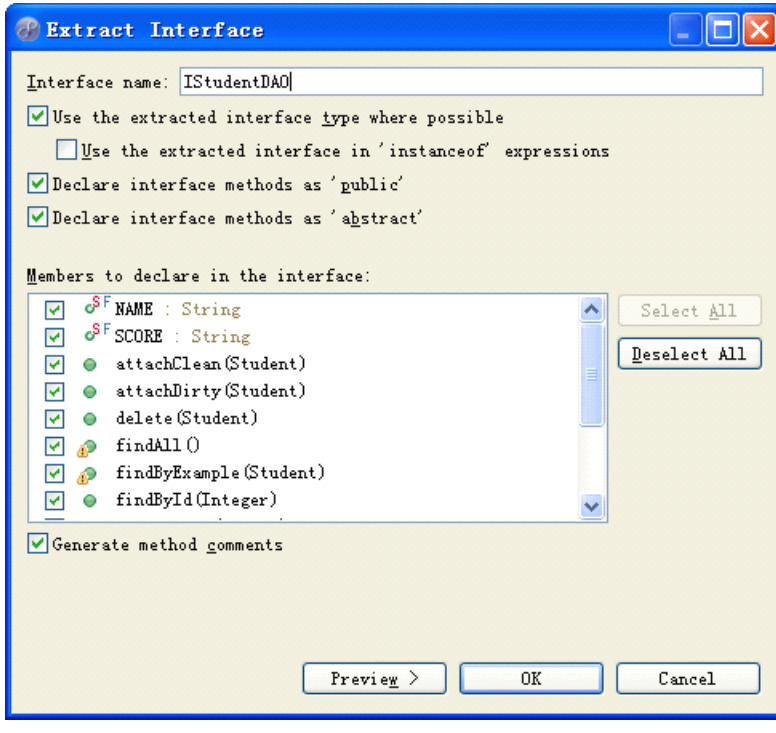


图 9-10　抽取接口

单击“OK”按钮，将完成抽取接口的操作。接下来就可以开发测试类了，它的代码为：

```
01  package com.sanqing.test;
02  import org.springframework.beans.factory.BeanFactory;
03  import org.springframework.context.support.ClassPathXmlApplicationContext;
04  import com.sanqing.dao.IStudentDAO;
05  import com.sanqing.dao.Student;
06  public class SHTest {
07      public static void main(String[] args) {
08          BeanFactory factory =
09                  new ClassPathXmlApplicationContext("applicationContext.xml");
10          IStudentDAO studentDAO=(IStudentDAO)factory.getBean("StudentDAO");
11          Student student=new Student();    //创建学生
12          student.setName("Tom");           //设置学生姓名
13          student.setScore(85.5);           //设置学生成绩
```

```
14                studentDAO.save(student);        //保存学生
15            }
16    }
```

上述代码中第 10 行获取学生 DAO 程序对象。在第 11 行中创建了学生对象，并依次将姓名和成绩设置到对象中。在第 14 行中调用 DAO 程序中的 save 方法，从而将学生对象信息保存到数据库中。

切换到 MyEclipse 中的数据库视图中，查看学生数据表可以发现多出了一条学生记录，字段内容就是测试类中设置的，从而表明 Spring 和 Hibernate 整合的项目开发完成。

9.1.6 通过注解方式进行事务管理

在学习 Spring 的面向切面编程时，已经知道可以通过注解的方式进行开发。在进行事务管理中，也可以通过注解的方式完成该功能，而且要比配置的方式简单很多。

在 MyEclipse 的数据库视图中，再以 teacher 老师数据表为例进行操作，从而生成 Hibernate 的程序，包括实体类、映射文件和 DAO 程序。打开自动生成的 DAO 程序，在类名上加入如下注解代码：

@Transactional

通过该代码，标注对该类中的方法进行事务管理。然后在 Spring 的配置文件中加入如下配置代码：

```
01    <tx:annotation-driven transaction-manager="transactionManager"
02                                                    proxy-target-class="true"/>
```

在其中使用“transaction-manager”属性指定事务管理器。这样对事务管理配置就已经完成了。最后进行通过测试类进行测试，它的代码为：

```
01    package com.sanqing.test;
02    import org.springframework.beans.factory.BeanFactory;
03    import org.springframework.context.support.ClassPathXmlApplicationContext;
04    import com.sanqing.dao.Teacher;
05    import com.sanqing.dao.TeacherDAO;
06    public class TeacherTest {
07        public static void main(String[] args) {
08            BeanFactory factory =
09                    new ClassPathXmlApplicationContext("applicationContext.xml");
10            TeacherDAO teacherDAO=(TeacherDAO)factory.getBean("TeacherDAO");
11            Teacher teacher=new Teacher();  //创建老师对象
12            teacher.setName("Tom");         //设置老师姓名
13            teacher.setAge(36);             //设置老师年龄
14            teacherDAO.save(teacher);       //保存老师
15        }
16    }
```

运行该程序，是不会发生错误的。在数据库视图中查看数据库数据，可以看到测试类中的测试数据已经保存到数据库中，表明事务处理成功。

说明：通过本节的程序，可以看到使用注解方式进行事务管理要比配置方式简单很多，但是目前开发中使用最多的还是配置方式，所以读者要将这两种方式都掌握。

9.2 Struts+Spring 整合开发

Struts 和 Spring 整合主要的目的就是让 Spring 管理 Struts 中的 Action，并且将业务逻辑层注入到 Action 中。Struts 和 Spring 的整合有多种方式，这里选择一种最通用、最优秀的整合方式进行讲解。

9.2.1 创建 Struts+Spring 的整合项目

因为 Struts 技术时仅仅使用在 Java Web 项目中的，所以我们首先要开发一个基本的 Java Web 项目，项目的名称叫做“StrutsAndSpring”。在加入框架支持时，Struts 和 Spring 的前后顺序是没有关系的，我们先来加入 Struts 框架支持。

在包资源管理器中，选中“SpringAndHibernate”项目节点。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Struts Capabilities”命令，将弹出加入 Struts 框架支持的界面，修改其中的包名后界面如图 9-11 所示。

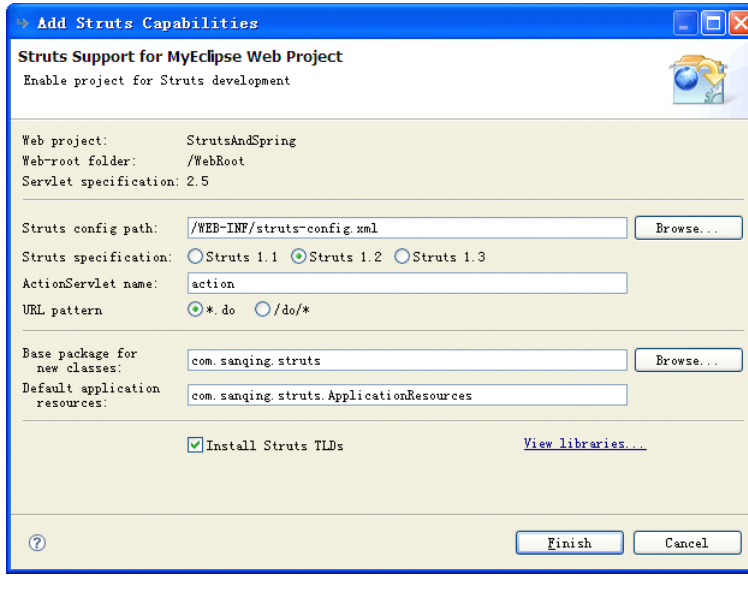


图 9-11　加入 Struts 支持

单击“Finish”按钮，将完成 Struts 框架的支持。再次在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Spring Capabilities”命令，将弹出加入 Spring 类库的界面，在其中不但要选择 Spring 的核心类库，还要选择用于 Web 项目开发的 Web 类库，如图 9-12 所示。

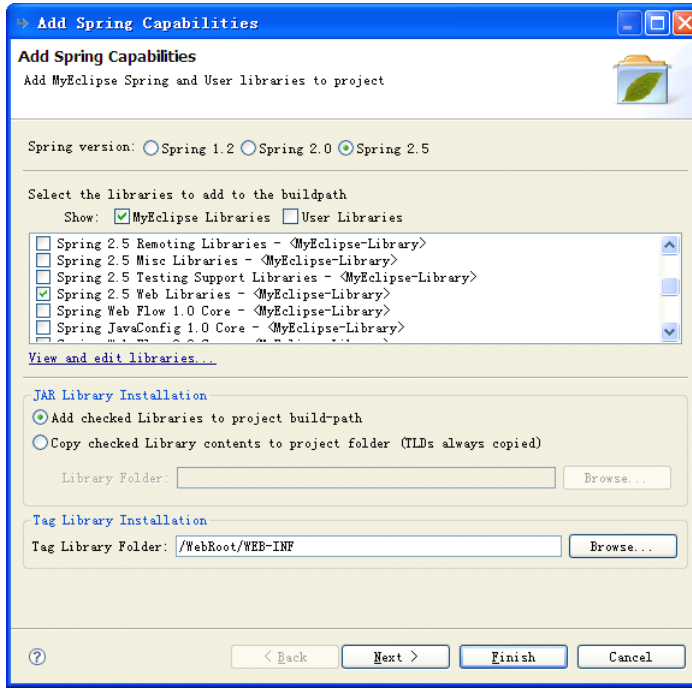


图 9-12　加入 Spring 类库

单击“Next”按钮，将完成 Spring 类库的添加。单击“Next”按钮，将进入对 Spring 配置文件进行设置的界面，默认情况下是将它创建在“src”目录下，但是在 Web 项目中，通常将它放在“WebRoot”|“WEB-INF”目录下，选择该目录后，界面如图 9-13 所示。

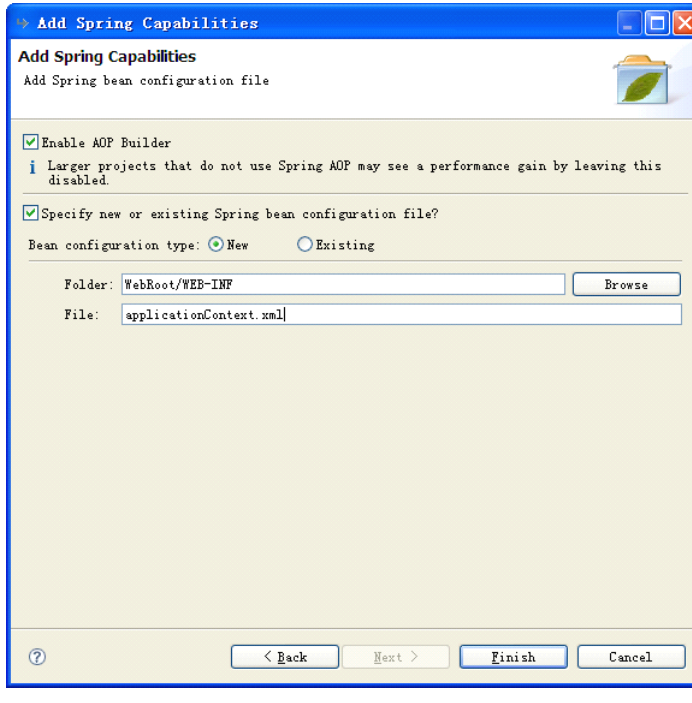


图 9-13　设置 Spring 配置文件

单击“Finish”按钮，将完成加入 Spring 框架支持的操作。

9.2.2 认识 Struts+Spring 的整合项目

依次加入 Struts 和 Spring 的框架支持后，在 MyEclipse 的包资源管理器中展开“StrutsAndSpring”项目，项目结构如图 9-14 所示。

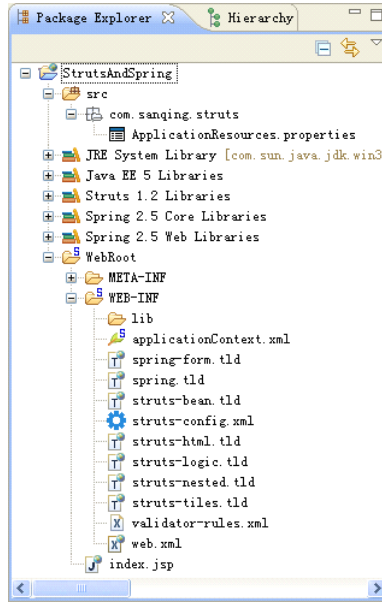


图 9-14　整合项目结构

整合后的项目结构是和 Struts 项目结构相差不大的，只是在其中加入了 Spring 的两个类库，并且 Spring 的配置文件并不直接放在 src 目录下，而是放在了“WebRoot”|“WEB-INF”目录下。

注意：将 Spring 的配置文件放在 src 目录下也是可以的，当部署项目后，它会自动放在“WebRoot”|“WEB-INF”|“classes”目录下。这两种放置方式可能造成后面的配置有所不同。

9.2.3 整合插件配置

在 Spring 的 web 开发类库中定义了“ContextLoaderPlugIn”类，通过对该类的配置可以让 Struts 框架找到 Spring 框架的支持。这里是通过 Struts 插件的形式加入的，它的配置代码如下所示：

```
01    <plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
02        <set-property property="contextConfigLocation"
03                                value="/WEB-INF/applicationContext.xml" />
04    </plug-in>
```

该配置代码要放在 Struts 的配置文件的“<struts-config>”标记对中。在该配置代码中，通过 ContextLoaderPlugIn 类可以指定哪些配置文件需要加载，从而可以在 Struts 其中时就创建 Spring 的 IoC 容器。

注意：Spring 的配置文件也可以有多个，在配置时只需要将这些配置文件到放在第 3 行的 value 属性中就可以，之间使用逗号分隔。这里也可以使用通配符，例如“/WEB-INF/applicationContext-*.xml”就表示所有以“applicationContext”开头的配置文件。

注意：当项目中的 Spring 配置文件不在“WEB-INF”目录下时，这里也需要改动这里的配置。例如直接放在 src 目录下，则 value 的值应该为“/WEB-INF/classes/applicationContext.xml”。

9.2.4 开发业务逻辑层并配置

这里就通过一个简单的登录程序来举例，在其中定义完成登录功能的业务方法。业务逻辑层实现类的代码如下所示。

```
01  package com.sanqing.service;
02  public class UserServiceImpl {
03      public boolean login(String username,String password){
04          if("Tom".equals(username)&&"456123".equals(password)){
05              return true;
06          }else{
07              return false;
08          }
09      }
10  }
```

在其中第 3 行中定义了登录业务方法，这里只是简单的通过固定字符来判断用户输入的是否为指定内容。

在包资源管理中，选中“UserServiceImpl”类节点，单击鼠标右键，在弹出的菜单中选择“Refector”|“Extract Interface”命令，将弹出抽取接口的界面，填写接口名和抽取的方法后，界面如图 9-15 所示。

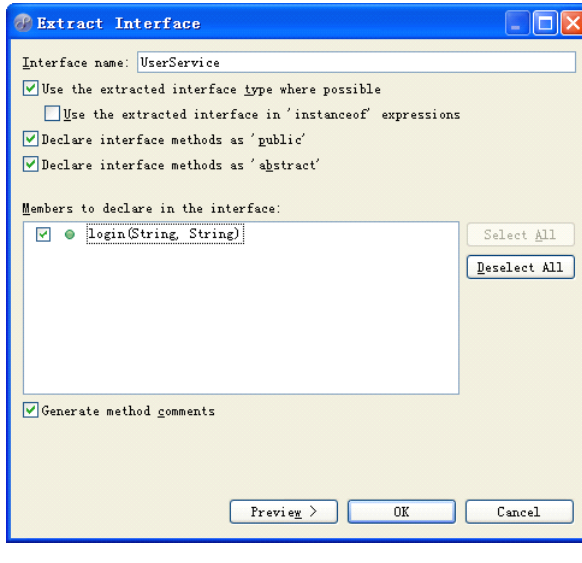


图 9-15　抽取接口

单击“OK”按钮，完成业务逻辑接口的抽取。接下来就需要在 Spring 的配置文件中，对业务逻辑层实现类进行配置。双击“ApplicationContext.xml”文件，在编辑区中打开该文件，将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Bean 程序进行配置的界面，填写配置信息后，界面如图 9-16 所示。

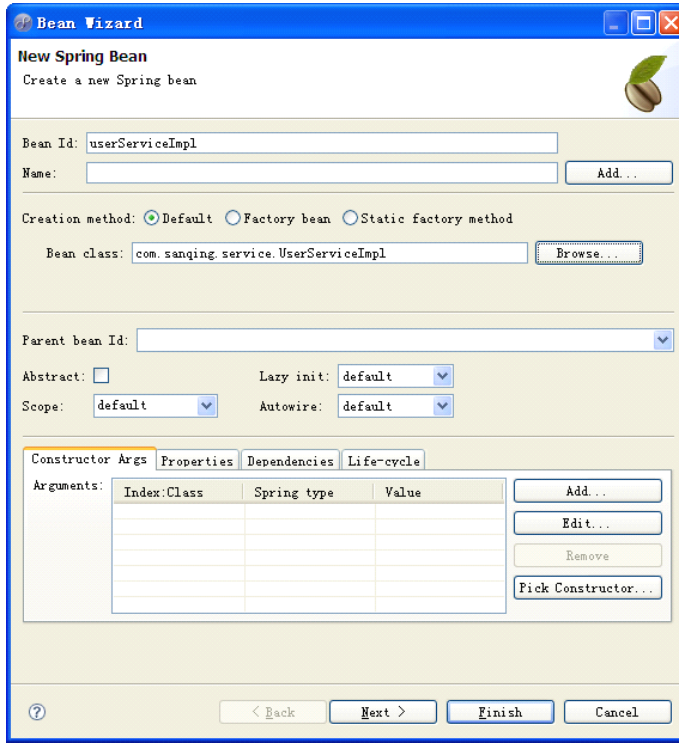


图 9-16　配置业务逻辑层

单击“Finish”按钮，将完成对业务逻辑层的配置，并在 Spring 的配置文件中自动生成配置代码。

9.2.5　开发 Struts 相关程序

开发完业务逻辑层后，我们就继续来开发控制层和显示层，它们都是通过 Struts 技术完成的。在 MyEclipse 的包资源管理器中，选中“com.sanqing.struts”包节点，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“struts 1.2 Form,Action & JSP”选项，单击“Next”按钮，将弹出创建 ActionForm 的界面，在其中输入信息后，界面如图 9-17 所示。

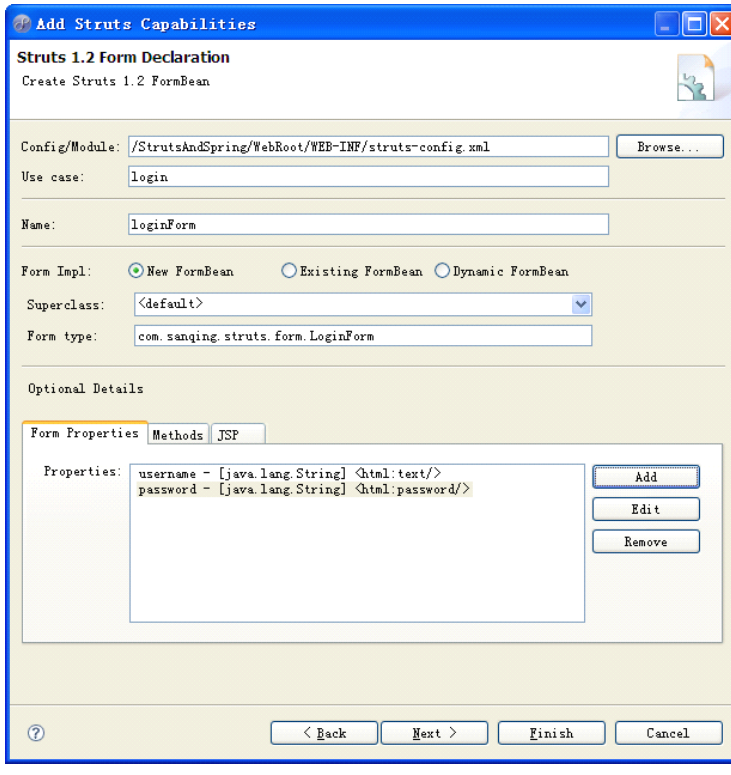


图 9-17　创建 ActionForm

除了界面中的配置外，还要选择“JSP”选项卡，其中的配置如图 9-18 所示。

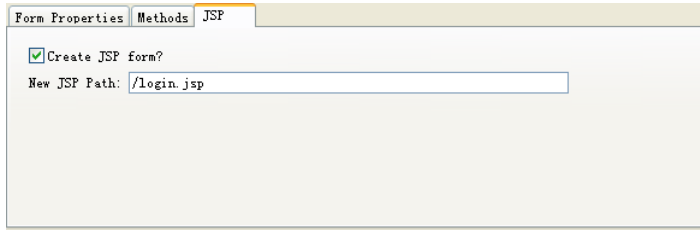


图 9-18　创建 JSP 页面

单击“Next”按钮，将进入创建 Action 的界面，在其中填写创建信息后，如图 9-19 所示。

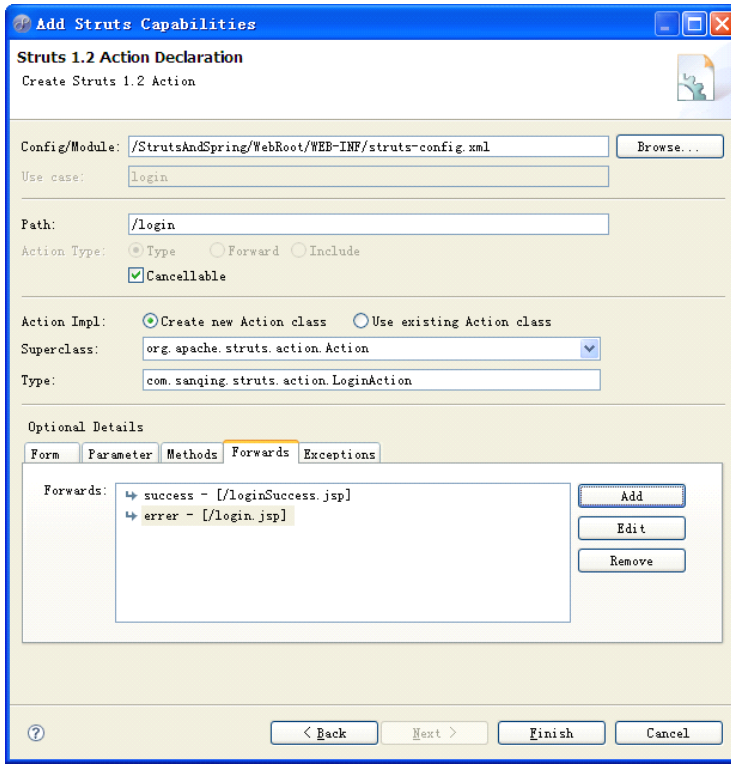


图 9-19　创建 Action

单击“Finish”按钮，将完成 Action 的创建，也就完成全部 Struts 程序的创建。并且同时在 Struts 配置文件中完成对 ActionForm 和 Action 的创建。

9.2.6　开发 Action 程序代码

通过上一小节的创建，是不会在 Action 中加入任何功能代码的。在登录 Action 中要通过 ActionForm 获取登录页面的信息，并且调用业务逻辑层。因为本项目采用到了和 Spring 的整合，所以可以以注入的方式，将业务逻辑层实现类对象注入到 Action 中。登录 Action 的具体代码为：

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class LoginAction extends Action {
04      private UserService userService;
05      public void setUserService(UserService userService) {
06          this.userService = userService;
07      }
08      public ActionForward execute(ActionMapping mapping, ActionForm form,
09              HttpServletRequest request, HttpServletResponse response) {
10          LoginForm loginForm = (LoginForm) form;
11          String username=loginForm.getUsername();        //获取用户名
12          String password=loginForm.getPassword();        //获取密码
13          try {
```

```
14                      boolean b=userService.login(username, password); //调用业务方法
15                      if(b==true){
16                          return mapping.findForward("success");
17                      }else{
18                          return mapping.findForward("errer");
19                      }
20                  } catch (Exception e) {
21                      return mapping.findForward("errer");
22                  }
23              }
24      }
```

在上述代码中第 4 行定义了业务逻辑层接口引用，并为它定义了 Setter 方法，通过这个方式将业务逻辑实现类对象注入。在第 14 行中调用 login 登录方法，当正确登录时，将返回“success”，从而跳转到登录成功页面，这是在配置文件中配置的。所以在项目中还要创建登录成功页面，该页面是非常简单的，读者可以自己查看光盘。当登录失败，或者发生异常时，将重新回答登录页面。

9.2.7 对 Action 进行配置

要想要 Spring 框架来管理 Action 程序，不但要在 Spring 的配置文件中进行配置，还要修改原有的 Struts 配置。在 Struts 配置文件中，要修改原 Action 配置的 type 属性值为“org.springframework.web.struts.DelegatingActionProxy”，该类定义在 Spring 框架中，它是“org.apache.struts.action.Action”的子类，通过它可以将请求转发给真正的 Action 来进行处理。

除此之外，还要在 Spring 配置文件中对 Action 进行配置，从而可以将 Action 看成普通的 Bean 来进行管理。将鼠标放在<beans>标记对中，单击鼠标右键，在弹出的菜单中选择“Spring”|“New Bean”命令，将弹出对 Action 这种特殊 Bean 程序进行配置的界面。因为 Action 并不会注入到其他 Bean 中，而仅仅是注入给 Struts 框架。所以这里可以只填写 name 选项，不填写 id 选项。在 name 选项中要填写 Action 的请求路径，也就是和 Struts 配置中的 path 属性相同。这样当调用 DelegatingActionProxy 类时，根据 path 属性值查找指定的 Spring 中的 Bean。配置界面如图 9-20 所示。

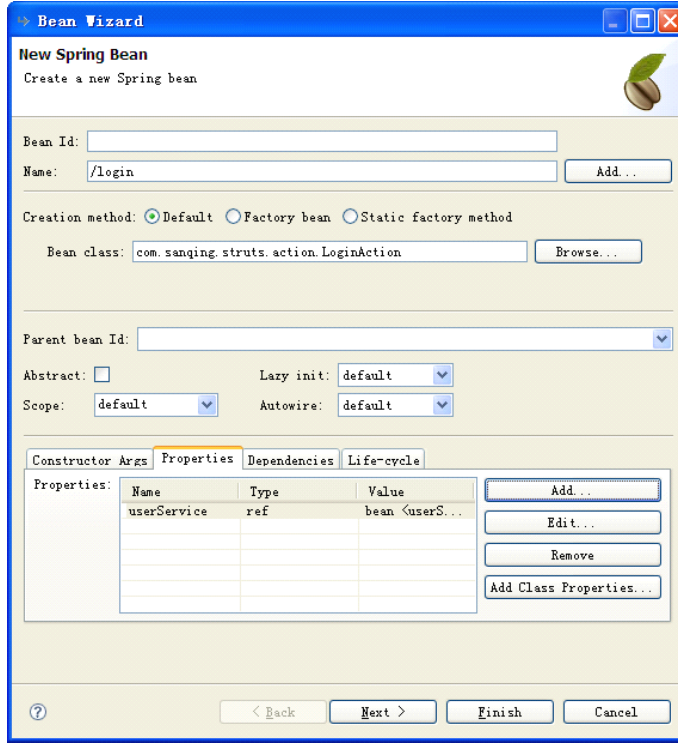


图 9-20　配置 Action

在其中还要将业务逻辑层注入到 Aciton 中，因为是通过 Setter 方法的方式注入的，所以选择“Properties”选项卡。单击“Finish”按钮，将完成对登录 Action 的配置。

9.2.8　运行项目

到目前为止，我们的 Struts+Spring 整合项目就已经开发完成了，现在就来看一下开发的项目是否正确。将项目发布到“Tomcat 6.x”服务器中，启动该服务器，打开浏览器，在其中输入如下地址：

http://localhost:8080/StrutsAndSpring/login.jsp

页面显示界面如图 9-21 所示。

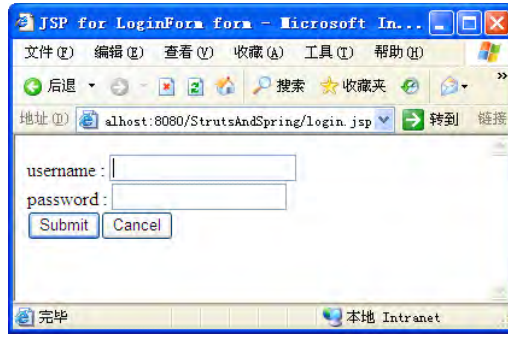


图 9-21　登录页面

这是自动生成的页面，所以其中都是以英文出现的，如何修改该页面，在 Struts 一

章已经讲解过，这里就不再重复讲解。在用户文本框中输入“Tom”，在密码框中输入“456123”，单击“Submit”按钮，将进入到登录成功页面，如图 9-22 所示。

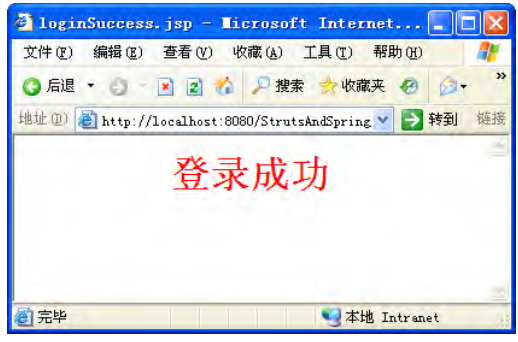


图 9-22　登录成功

如果在登录页面中，输入的是错误信息，则会执行 Action，并且再次跳回到登录页面，从而表明该 Struts 和 Spring 整合的项目开发成功。

9.3　SSH 框架整合开发博客网站

在前面的学习中已经分别讲解了 Struts、Hibernate 和 Spring 技术，并且还讲解了它们两两之间的整合，在本节中将通过一个博客网站项目来演示如何将这三种框架技术整合在一起进行开发。

9.3.1　博客网站项目功能分析

在进行大型项目开发之前，一定要对项目的功能进行分析，从而进行系统分析。用户进入博客网站后，可以通过注册可以拥有自己的博客；登录后将进入自己的博客。在个人博客中，可以查看自己发表的所有文章，发表文章，修改已发表文件和删除文章。

如果用户不注册，将以游客的身份出现。不管游客还是注册用户，都可以查看所用用户的文章，并能够对文件进行评论。

9.3.2　创建数据表

从系统功能上可以看出，博客网站有三种主体对象，分别是用户、文章和评论，所以要为它们分别定义三张数据表，我们将这些数据表都放在“blog”数据库中。在 MyEclipse 的数据库视图中，打开“MySQL”连接，在“Connected to MySQL”节点上，单击鼠标右键，在菜单中选择“Create database”命令，在弹出的创建数据库界面中填写相关信息后，如图 9-23 所示。

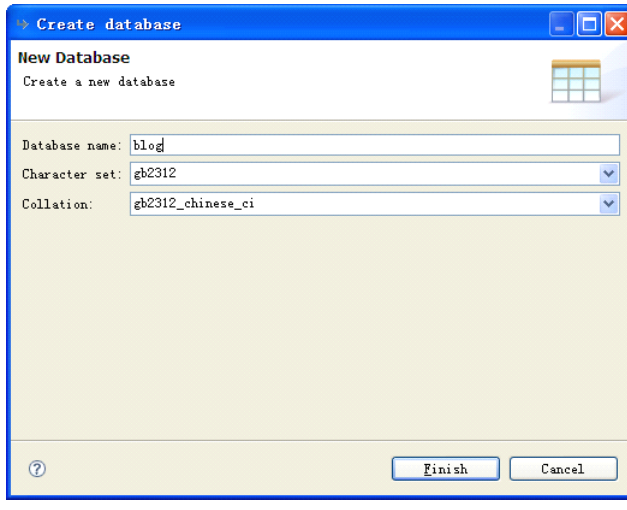


图 9-23　创建博客数据库

单击“Finish”按钮，将完成数据库的创建。选中“blog”数据库节点，单击鼠标右键，选择“New Table”命令，将弹出创建数据表界面。通过该界面，依次创建用户、文章和评论三张数据表。首先来创建用户数据表，它的创建界面如图 9-24 所示。

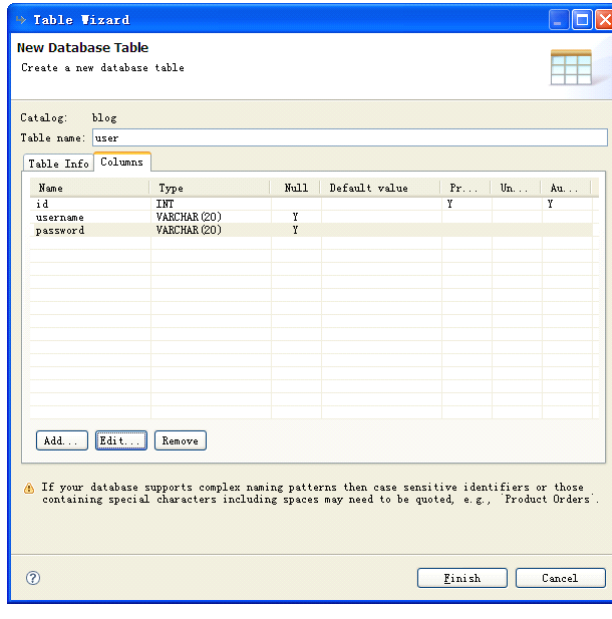


图 9-24　创建用户表

在用户表中创建 id 主键、用户名和密码两个字段。如果网站中可以设置用户昵称，或者需要设置验证问题和答案，都可以添加这些字段，这里为了简单只填写这两个字段。

再来创建文章数据表，创建界面如图 9-24 所示。

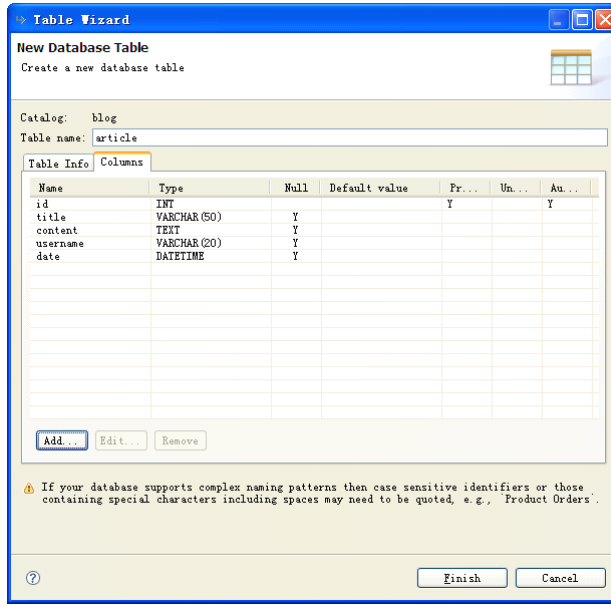


图 9-25　创建文章表

在文章表中，除了定义了 id 主键外，还定义了标题、内容、发表用户和发表时间等四个字段。其中文章内容不应该使用“VARCHAR”类型，而应该使用“TEXT”类型，从而保存大文本内容。

最后创建评论数据表，创建界面如图 9-26 所示。

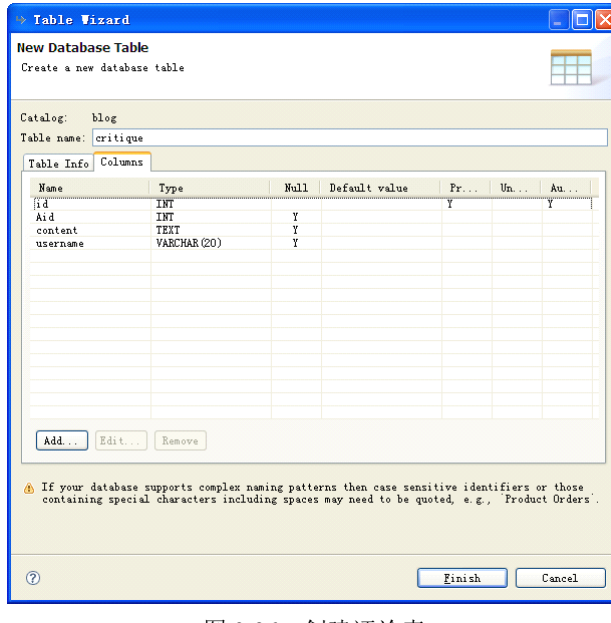


图 9-26　创建评论表

其中 id 是评论表的主键。并且并且定义了“Aid”字段，用来表示所属文章 id，也就是该评论是发表在哪一个文章下的。除此之外还有发表内容和发表用户两个字段。

创建数据库和数据表完成后修改“MySQL”数据库连接，让其中的 URL 指向“blog”

数据库。

9.3.3 创建博客项目

首先创建一个普通的 Java Web 项目，项目名称为“BLOG”，然后就是向其中加入框架支持。如果想让 Spring 对 Hibernate 进行管理，就一定要先加入 Spring 框架支持，其他顺序是没有要求的。

技巧：通常情况下，加入顺序是 Struts、Spring 和 Hibernate，这也是 SSH 框架整合缩写的由来。

先向“BLOG”项目中加入 Struts 框架支持，加入界面，如图 9-27 所示。

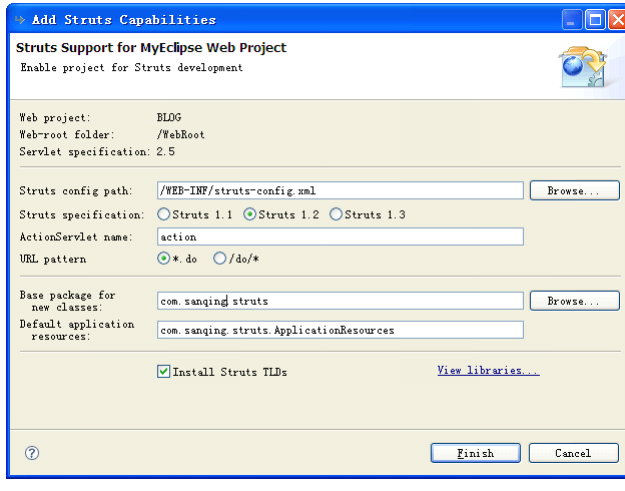


图 9-27　加入 Struts 支持

然后加入 Spring 框架支持，其中不要忘记加入 web 开发所用到的类库，如图 9-28 所示。

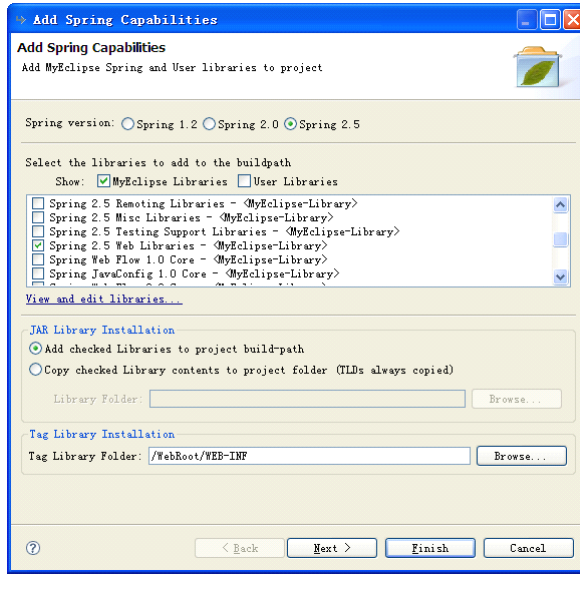


图 9-28　加入 Spring 类库支持

单击“Next”按钮，将进入对 Spring 配置文件的设置，将该配置文件放在“WEB-INF”目录下，如图 9-29 所示。

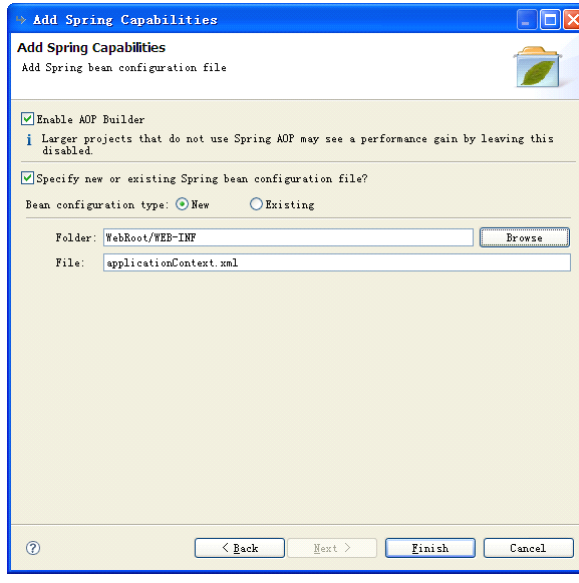


图 9-29　设置 Spring 配置文件

单击“Finish”按钮，将完成 Spring 框架的加入支持操作。最后加入 Hibernate 的框架支持，在其中不要忘记 Spring 整合 Hibernate 时所用到的类库，如图 9-30 所示。

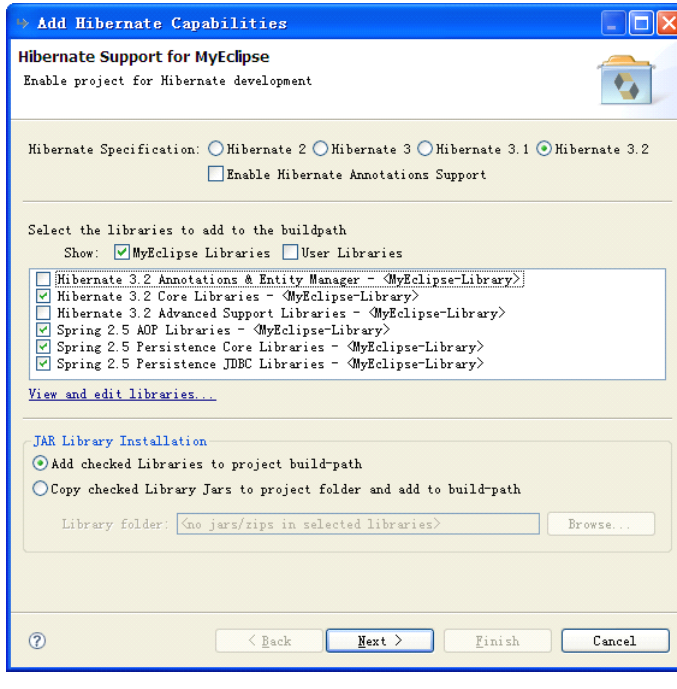


图 9-30　加入 Hibernate 类库支持

单击“Next”按钮，将进入选择 Hiibernate 配置信息支持的界面，在其中选择第 2 个选项，让 Spring 的配置文件对数据库的相关信息管理。单击“Next”按钮，将进入配置会话工厂类的界面，选择使用已有 Spring 配置文件，其他采用默认值。单击“Next”按钮，将进入配置数据库连接的界面，在其中选择“MySQL”，如图 9-31 所示。

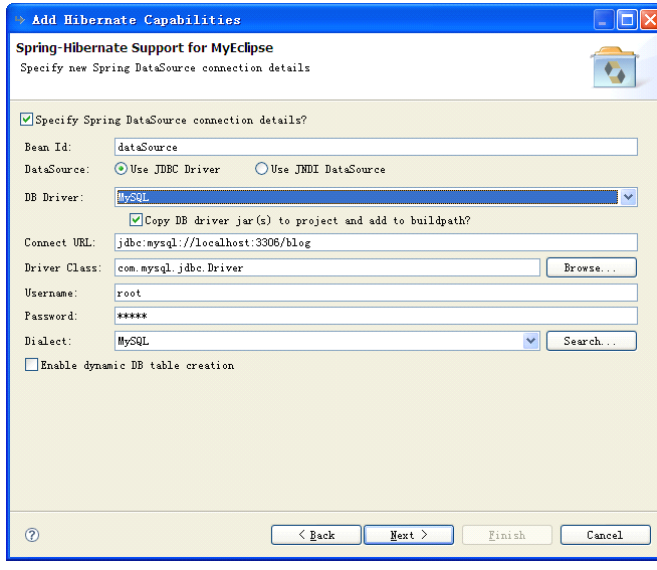


图 9-31　建立数据库连接

单击“Next”按钮，将进入创建会话工厂类的界面，将它定义在“com.sanqing.hb”包中，单击“Finish”按钮，将完成加入 Hibernate 框架支持的操作。

加入三种框架支持后，读者可以自己简单的了解一下项目结构，这里就不在抓图讲解了。

注意：加入三种框架后，经常会出现相同包之间重复的错误，这是因为有可能三个框架中都引入了用一个 JAR 包，这时候只需要将重复的 JAR 包去掉就可以。最常见的就是 asm.jar 包重复。

9.3.4　进行整合配置

仅仅加入框架支持后，是还不能进行整合开发的，还需要进行必要的配置。首先是在 Struts 配置文件中加入 Struts 和 Spring 整合所需要的插件配置代码，具体代码为：

```
01    <plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">
02        <set-property property="contextConfigLocation"
03                                    value="/WEB-INF/applicationContext.xml" />
04    </plug-in>
```

当 Spring 和 Hibernate 整合时，通常将事务将给 Spring 进行管理，所以要在 Spring 配置文件中对事务进行配置，配置代码如下所示。

```
01    <bean id="transactionManager"
02            class="org.springframework.orm.hibernate3.HibernateTransactionManager">
03        <property name="sessionFactory">
```

```
04              <ref bean="sessionFactory"/>
05          </property>
06      </bean>
07      <tx:advice id="txAdvice" transaction-manager="transactionManager">
08          <tx:attributes>
09              <tx:method name="*" propagation="REQUIRED"/>
10          </tx:attributes>
11      </tx:advice>
12      <aop:config>
13          <aop:pointcut id="allDaoMethod"
14                                  expression="execution(* com.sanqing.dao.*.*(..))"/>
15          <aop:advisor pointcut-ref="allDaoMethod" advice-ref="txAdvice"/>
16      </aop:config>
```

在其中不要忘记加入 XML 文件头信息。

9.3.5 开发数据访问层

在实际开发中，Java 项目的开发顺序通常是固定的，从数据访问层开始，然后是业务逻辑层，最后是控制层和显示层。在 SSH 框架整合项目也是按照这个顺序的，我们先来开发数据访问层，通过它可以进行基本的数据库操作。

在 MyEclipse 的数据库视图中，选中要操作的数据表节点，例如这里选中“MySQL”|“Connection to MySQL”|“blog”|“TABLE”|“user”节点，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，在其中选择创建实体类、映射文件和 DAO 程序，如图 9-32 所示。

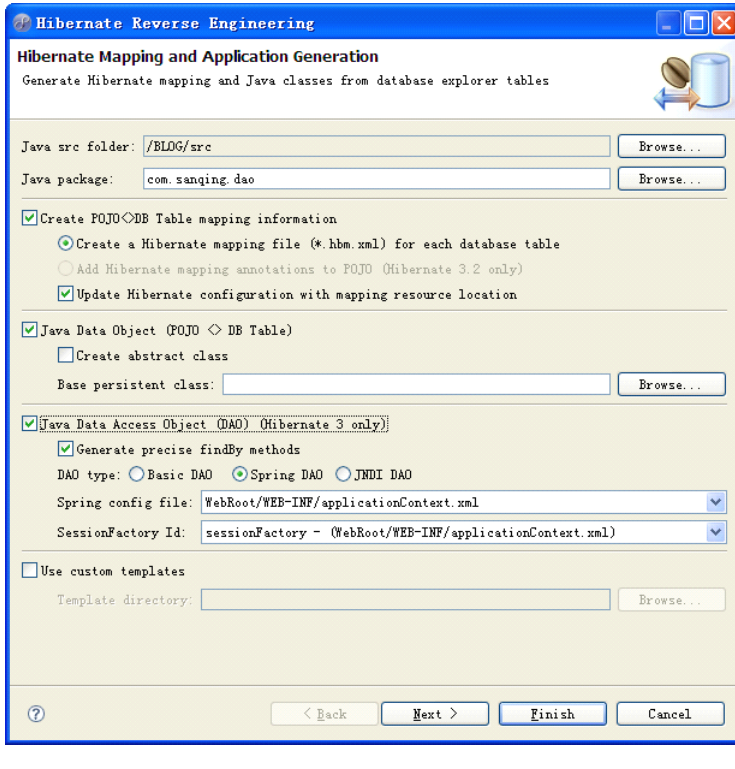


图 9-32　创建 Hibernate 程序

单击“Finish”按钮，将完成用户数据访问层程序。这里只生成了 DAO 实现类程序，为了使用 Spring 的注入技术，所以还需要为它抽取接口。

在包资源管理器中，选中“UserDAO”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，在其中输入接口名为“IUserDAO”，然后将其中的所有方法选中，界面如图 9-33 所示。

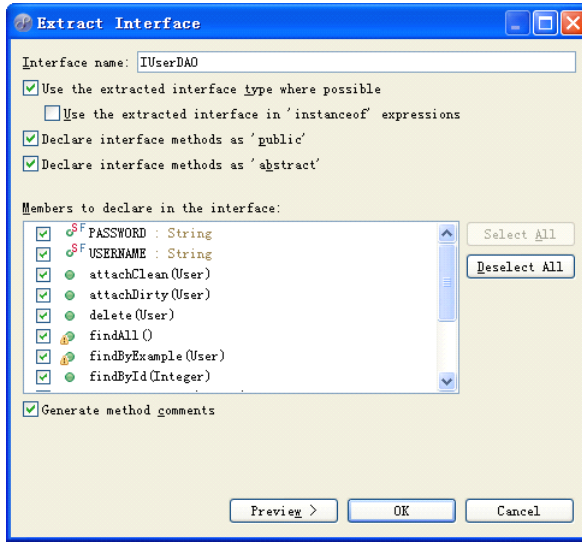


图 9-33　抽取用户 DAO 接口

抽取接口后，就可以使用该接口来引用实现类对象。通过上述方法，依次再对 article 文章数据表和 critique 评论数据表进行操作，从而生成它们的 DAO 程序，并也进行抽取方法操作。操作后的具体代码这里就不再给出，读者可以自己查看光盘中的代码。

说明：使用 Spring DAO 类型创建的数据访问层程序，在创建完成后，也会在 Spring 配置文件中自动进行配置，是不需要手动配置的。

9.3.6　开发用户业务逻辑层

业务逻辑层的主要作用就是定义相应的功能方法，从而完成项目中的功能需求。在项目中，业务逻辑层将被控制层调用，在业务逻辑层的方法中又将调用数据访问层，从而进行数据库操作。

业务逻辑层也是按照项目操作主体来划分的，所以本博客项目中，将分为用户业务逻辑实现类、文章业务逻辑实现类和评论业务逻辑实现类。在业务逻辑实现类中开发哪些方法是由项目开始之前的功能分析得到的，例如用户业务逻辑实现类中就应该定义注册和登录方法。因为在项目中使用到了 Spring 的控制反转技术，所以要在业务逻辑层中定义数据访问层的引用，并为它定义 Setter 方法，通过这种方式注入。用户业务逻辑层的具体代码如下：

```
01    package com.sanqing.service;
```

```
02  import java.util.List;
03  import com.sanqing.dao.IUserDAO;
04  import com.sanqing.dao.User;
05  public class UserServiceImpl implements UserService {
06      private IUserDAO userDAO;
07      public void setUserDAO(IUserDAO userDAO) {
08          this.userDAO = userDAO;
09      }
10      public boolean register(User user){          //注册方法
11          if(userDAO.findByUsername(user.getUsername()).size()>0){
12              return false;
13          }else{
14              userDAO.save(user);
15              return true;
16          }
17      }
18      public boolean login(User user){          //登录方法
19          if(userDAO.findByUsername(user.getUsername()).size()==0){
20              return false;
21          }else{
22              User findUser=
23                  (User)userDAO.findByUsername(user.getUsername()).get(0);
24              if(findUser.getPassword().equals(user.getPassword())){
25                  return true;
26              }else{
27                  return false;
28              }
29          }
30      }
31      public List<User> showAllUser(){
32          return userDAO.findAll();
33      }
34  }
```

其中第 5 行中定义了用户数据访问接口引用，并它定义了 Setter 方法，从而能够将数据访问实现类对象注入。在第 9 行中定义了注册方法，在其中要首先判断该用户名是否已经被注册，如果没有则可以执行保存操作。第 17 行中定义了登录方法，在其中同样要判断用户名是否存在，如果不存在时肯定不能够登录的。当存在时，并且密码也相同的话，就可以进行登录。在第 31 行中还定义了获取所有用户的方法，从而可以将所有用户显示在首页中。

开发完业务逻辑实现类后，执行抽取接口操作。抽取接口的具体操作就不介绍了，它的操作界面如图 9-34 所示。

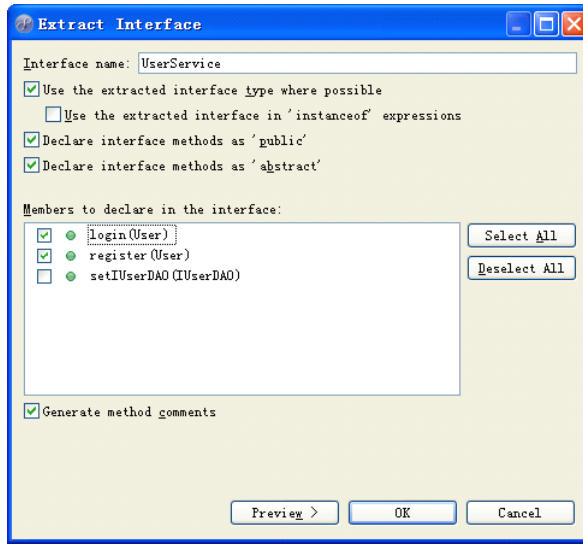


图 9-34　抽取业务逻辑接口

其中的数据访问接口的 Setter 方法不需要抽取。单击“OK”按钮，完成用户业务逻辑接口的抽取。由于业务逻辑层程序是手动开发的，是不会在 Spring 配置文件中自动配置的，所以要进行手动配置。配置界面如图 9-35 所示。

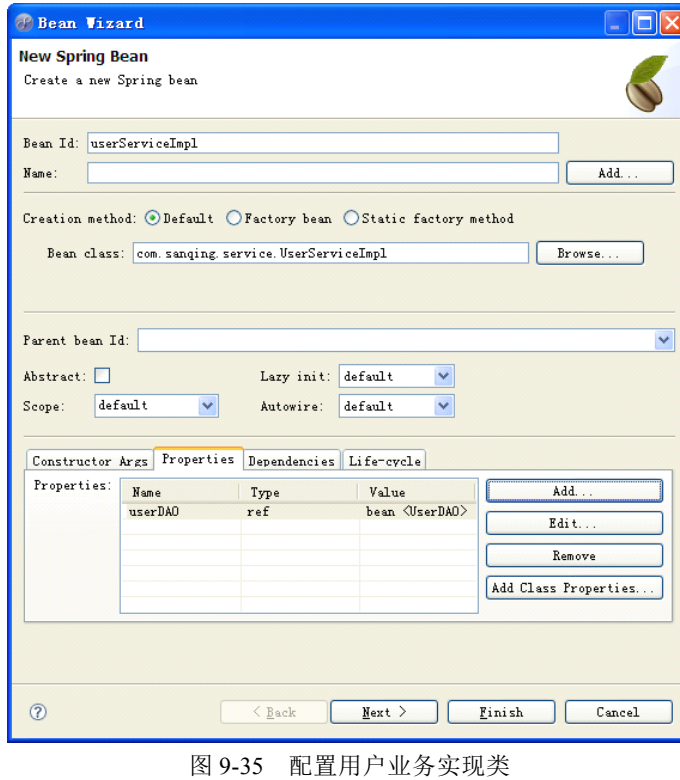


图 9-35　配置用户业务实现类

单击“Finish”按钮，完成用户业务实现类的配置，到这里用户业务逻辑层就开发完成。

9.3.7 开发文章业务逻辑层

在文章业务逻辑层中要完成对文章进行操作的功能，包括显示网站所有文章、显示某用户所有文章、显示某一文章详细内容、发表文章、删除文章和对文章进行修改等功能。同样在文章业务逻辑实现类中定义数据访问接口引用，并为它定义 Setter 方法。文章业务实现类的具体代码如下：

```
01  package com.sanqing.service;
02  import java.util.List;
03  import com.sanqing.dao.Article;
04  import com.sanqing.dao.IArticleDAO;
05  public class ArticleServiceImpl {
06      private IArticleDAO articleDAO;
07      public void setArticleDAO(IArticleDAO articleDAO) {
08          this.articleDAO = articleDAO;
09      }
10      public List<Article> showALLArticle(){      //显示所有文章
11          return articleDAO.findAll();
12      }
13      public List<Article> showUserArticle(String username){//显示指定用户文章
14          return articleDAO.findByUsername(username);
15      }
16      public Article showArticle(int id){             //显示文章具体内容
17          return articleDAO.findById(id);
18      }
19      public void addArticle(Article article){        //发表文章
20          articleDAO.save(article);
21      }
22      public void deleteArticle(Article article){     //删除文章
23          articleDAO.delete(article);
24      }
25      public void updateArticle(Article article){     //更新文章
26          articleDAO.attachDirty(article);
27      }
28  }
```

其中第 6 行定义了文章数据访问接口引用，并为它定义了 Setter 方法，通过这种方式将数据访问实现类对象注入。并且在后面依次定义了显示所有文章、显示指定用户发表的文章、显示文章具体信息、发表文章、删除文章和更新文章等功能方法。

开发完文章业务逻辑实现类后，为它抽取接口，抽取接口操作界面如图 9-36 所示。

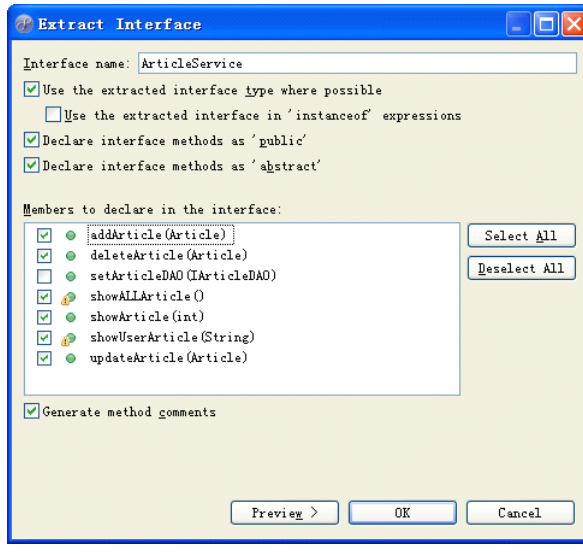


图 9-36　抽取文章业务逻辑接口

单击“OK”按钮，完成抽取文章业务逻辑接口操作。最后就是在 Spring 配置文件中对文章业务实现类进行配置，配置界面如图 9-37 所示。

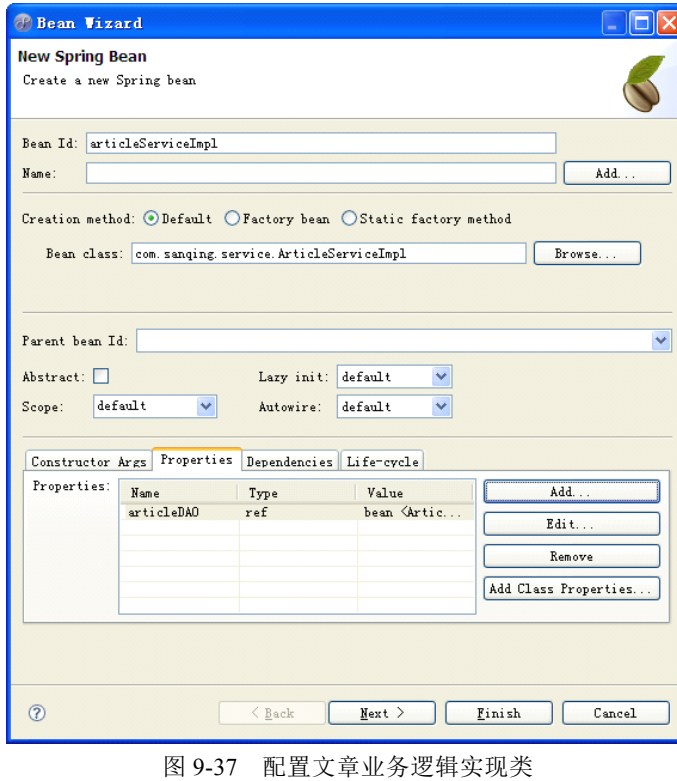


图 9-37　配置文章业务逻辑实现类

单击“Finish”按钮，完成文章业务逻辑实现类的配置。

9.3.8 开发评论业务逻辑层

在评论业务逻辑层中要完成对评论进行操作的功能，包括发表评论，显示文章所有评论和获取文章评论数量等功能。在评论业务逻辑实现类中要定义评论数据访问接口引用，并为它定义 Setter 方法，通过这种方式注入。评论业务逻辑实现类的具体代码如下所示。

```
01  package com.sanqing.service;
02  import java.util.List;
03  import com.sanqing.dao.Critique;
04  import com.sanqing.dao.ICritiqueDAO;
05  public class CritiqueServiceImpl {
06      private ICritiqueDAO critiqueDAO;
07      public void setCritiqueDAO(ICritiqueDAO critiqueDAO) {
08          this.critiqueDAO = critiqueDAO;
09      }
10      public void addCritique(Critique critique){    //发表评论
11          critiqueDAO.save(critique);
12      }
13      public List<Critique> showAllCritique (int aId){    //显示文章所有评论
14          return critiqueDAO.findByAid(aId);
15      }
16      public int getCritiqueCount(int aId){    //获取文章评论数量
17          return showAllCritique (aId).size();
18      }
19  }
```

上述代码中第 6 行定义了评论数据访问接口引用，并为它定义了 Setter 方法，从而将评论数据访问实现类对象注入。在后面依次定义了发表评论、显示文章所有评论和获取文章评论数量等方法。

同样，开发完评论业务逻辑实现类后，要对它进行抽取接口的操作，抽取接口操作界面如图 9-38 所示。

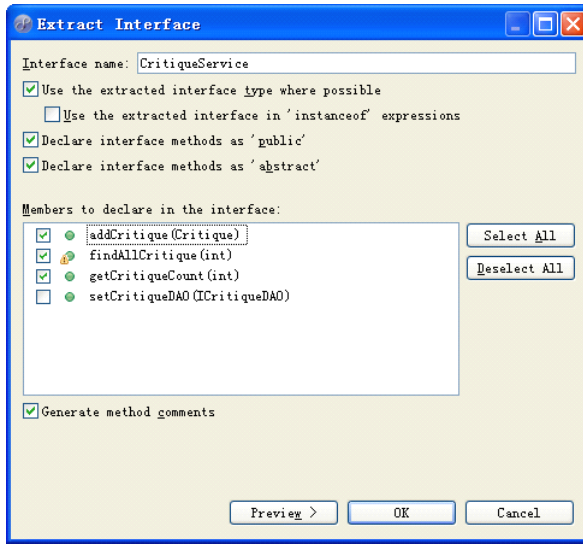


图 9-38　抽取评论业务逻辑接口

单击“OK”按钮，将完成抽取评论业务逻辑接口的操作。同样开发完评论业务逻辑实现类后，要在 Spring 配置文件中对它进行配置，配置界面如图 9-39 所示。

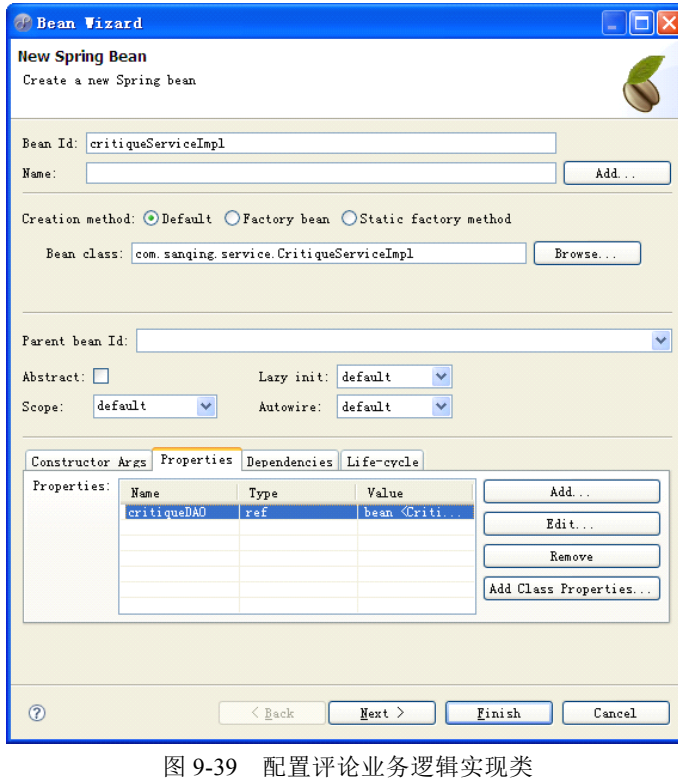


图 9-39　配置评论业务逻辑实现类

单击“Finish”按钮，完成对评论业务逻辑实现类的配置。

9.3.9 开发首页控制层

因为本节中讲解的博客显示是 SSH 框架整合开发的，所以项目中的控制层将由 Struts 中的 Action 完成。在博客网站的首页中将把网站中所有的文章和用户显示出来，这些都不是固定的，所以可以定义一个首页控制层，在其中获取显示数据后跳转到首页中。

因为在首页控制层中，并不需要接收任何参数，所以是不需要定义相应的 ActionForm 的。在创建博客项目后，会在项目中自动创建“com.sanqing.struts”包。选中该包，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将弹出选择创建程序的界面，在其中选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Action”选项，单击“Next”按钮，将弹出创建 Action 程序的界面，在其中输入必须信息，如图 9-40 所示。

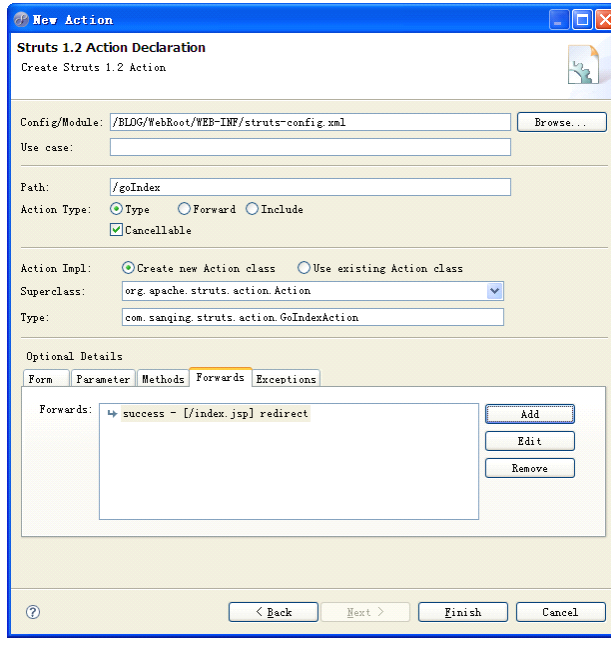


图 9-40　创建首页控制层

单击“Finish”按钮，将完成首页控制层的创建。因为本项目中将采用 SSH 框架整合开发，所以在 Action 中调用业务逻辑层时采用注入的方式。在首页 Action 中将定义所需要的业务逻辑接口引用，并为它们定义 Setter 方法，使用这种方式将业务逻辑实现类对象注入。首页 Action 的具体代码如下所示。

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class GoIndexAction extends Action {
04      private UserService userService;
05      private ArticleService articleService;
06      private CritiqueService critiqueService;
07      public void setUserService(UserService userService) {
```

```
08              this.userService = userService;
09          }
10          public void setArticleService(ArticleService articleService) {
11              this.articleService = articleService;
12          }
13          public void setCritiqueService(CritiqueService critiqueService) {
14              this.critiqueService = critiqueService;
15          }
16
17          public ActionForward execute(ActionMapping mapping, ActionForm form,
18                  HttpServletRequest request, HttpServletResponse response) {
19              List<User> userList=userService.showAllUser();
20              List<Article> articleList=articleService.showALLArticle();
21              List<Integer> countList=new ArrayList<Integer>();
22              for(int i=0;i<articleList.size();i++){
23                  Article article=articleList.get(i);
24                  int count=critiqueService.getCritiqueCount(article.getId());
25                  countList.add(count);
26              }
27              request.setAttribute("userList", userList);
28              request.setAttribute("articleList", articleList);
29              request.setAttribute("countList", countList);
30              return mapping.findForward("success");
31          }
32      }
```

因为在首页中将显示所有用户、所有文章和每一个文章的评论数，所以定义用户业务逻辑接口、文章业务逻辑接口和评论业务逻辑接口，并为它们定义 Setter 方法，从而通过这种方式将相应的实现类对象注入。

在第 19 行中调用用户业务逻辑实现类中的 showAllUser 方法，从而获取目前网站中所有的用户，在第 27 行中将获取的用户集合保存到 request 范围内。在第 20 行中调用文章业务逻辑实现类中的 showALLArticle 方法，从而获取已发表的所有文章组成的集合，在第 28 行中同样将它保存到 request 范围内。

在第 24 行中调用评论业务逻辑实现类中的 getCritiqueCount 方法，从而获取每一文章的评论数，并依次保存到集合中，在显示时可以按照顺序依次得到每一文章评论数。

开发完首页 Action 后，要对它进行配置。首先要修改 Struts 配置文件中的配置，将 type 属性值修改为“org.springframework.web.struts.DelegatingActionProxy”，从而将 Action 交给 Spring 进行管理。然后再在 Spring 的配置文件中对 Action 进行配置，配置界面如图 9-41 所示。

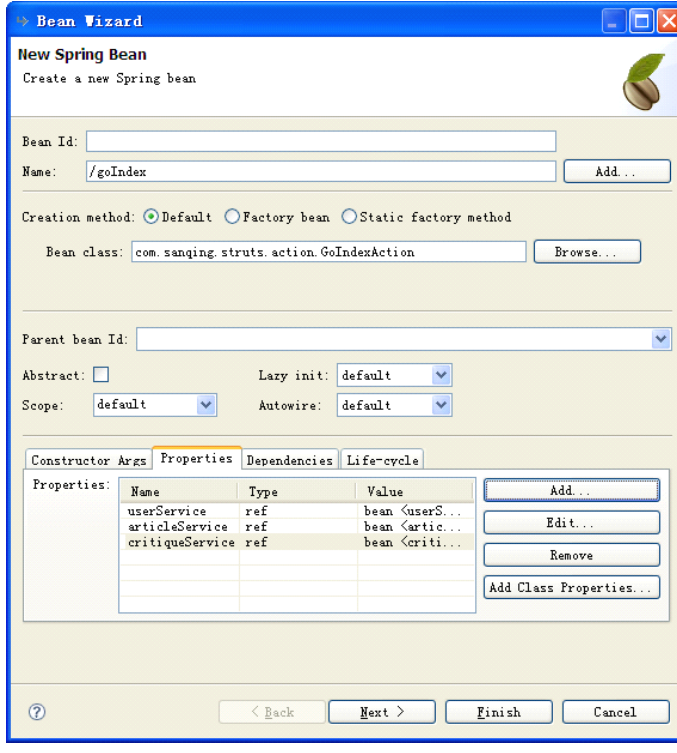


图 9-41　配置首页 Action

单击“Finish”按钮，将完成对指向首页 Action 的 Spring 配置。在其中将所需要的业务逻辑实现类对象注入。

9.3.10　开发用户控制层

在 Struts 框架中，通常将对某一主体进行操作的 Action 放在一个 DispatchAction 中，在 Struts 一章中已经讲解过。对于注册和登录来说，都是对用户进行操作，所以就可以定义一个 DispatchAction，在其中定义分别执行注册和登录的方法。

因为在注册和登录 Action 中都需要接收来自页面的信息，所以要定义用户 ActionForm，所以这里选择同时创建 JSP 页面、ActionForm 程序和 Action 程序的命令。

在选择创建 Struts 程序的界面中，选择“MyEclipse”|“Web-Struts”|“Struts 1.2”|“struts 1.2 Form,Action & JSP”选项，单击“Next”按钮，将首先弹出创建 ActionForm 的界面，填写用户 ActionForm 后，界面如图 9-41 所示。

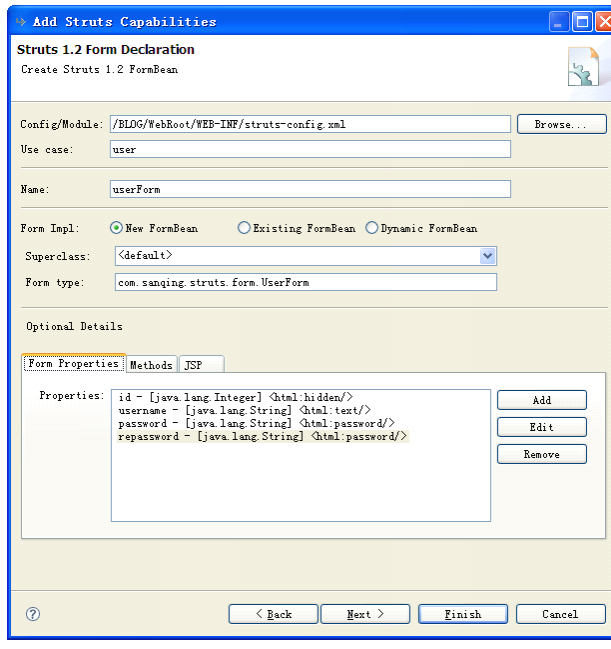


图 9-41　创建用户 ActionForm

除了界面中的配置信息外，在“Methods”选项卡中取消验证和重置两个方法选中状态，在“JSP”选项卡中选择创建 JSP 程序，JSP 程序的名称为“/register.jsp”。单击“Next”按钮完成用户 AtionForm 的创建，并进入到创建用户 Action 的界面，在其中填写相关信息后，如图 9-42 所示。

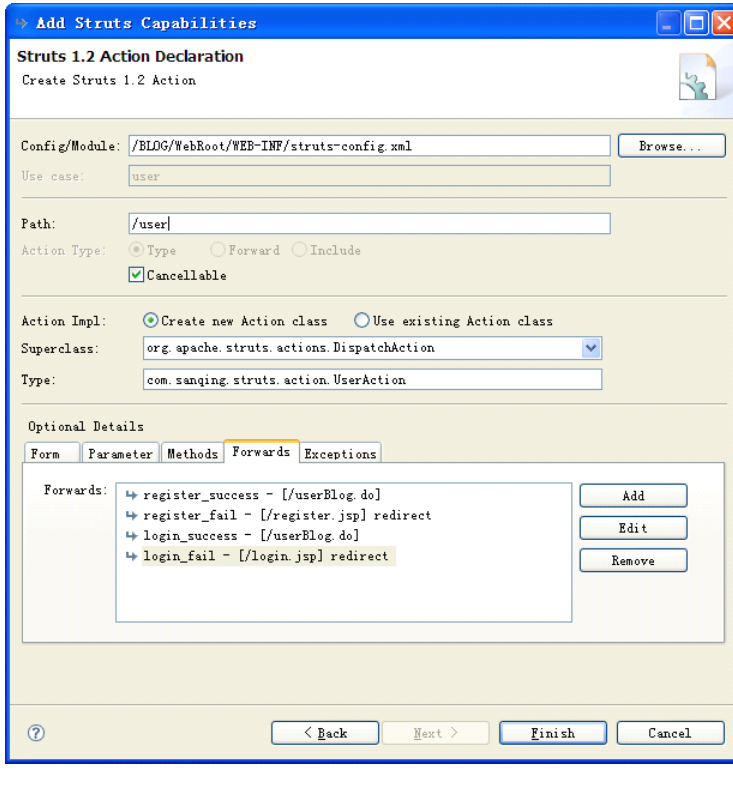


图 9-42　创建用户 Action

除了界面中的配置外，在“Form”选项卡中自动配置了用户 ActionForm，在“Parameter”选项卡中还将配置所必须的参数，这里设置参数名为“command”。单击“Finish”按钮，将完成用户 Action 的创建。

打开创建的用户 Action，在其中分别定义完成注册和登录的控制方法，其具体代码如下所示。

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class UserAction extends DispatchAction {
04      private UserService userService;
05      public void setUserService(UserService userService) {
06          this.userService = userService;
07      }
08      public ActionForward register(ActionMapping mapping, ActionForm form,
09              HttpServletRequest request, HttpServletResponse response) {
10          UserForm userForm = (UserForm) form;// TODO Auto-generated method stub
11          User user=new User();
12          user.setUsername(userForm.getUsername());
13          user.setPassword(userForm.getPassword());
14          try {
15              if(userService.register(user)){
16              request.getSession().setAttribute("username", userForm.getUsername());
17                  return mapping.findForward("register_success");
18              }else{
19                  return mapping.findForward("register_fail");
20              }
21          } catch (Exception e) {
22              return mapping.findForward("register_fail");
23          }
24      }
25      public ActionForward login(ActionMapping mapping, ActionForm form,
26              HttpServletRequest request, HttpServletResponse response) {
27          UserForm userForm = (UserForm) form;// TODO Auto-generated method stub
28          User user=new User();
29          user.setUsername(userForm.getUsername());
30          user.setPassword(userForm.getPassword());
31          try {
32              if(userService.login(user)){
33              request.getSession().setAttribute("username", userForm.getUsername());
34                  return mapping.findForward("login_success");
35              }else{
36                  return mapping.findForward("login_fail");
37              }
38          } catch (Exception e) {
```

```
39                      return mapping.findForward("login_fail");
40                  }
41              }
42      }
```

上述代码中第 4 行定义了用户业务逻辑接口引用，并为它定义了 Setter 方法，通过这种方式将用户业务逻辑实现类对象注入。在第 8 行中定义了注册方法，在其中调用用户业务逻辑实现类中的 register 方法，当返回结果为 true 时，表示能够注册。当返回结果为 false，或者操作发生异常，将不能够进行注册，返回注册页面。登录方法也是类似的。

开发完代码程序后，就要对用户 Action 进行配置。首先将 Struts 配置文件中对用户 Action 配置的 type 属性修改为“org.springframework.web.struts.DelegatingActionProxy”。然后在 Spring 配置文件中对 Action 进行配置，配置界面如图 9-43 所示。

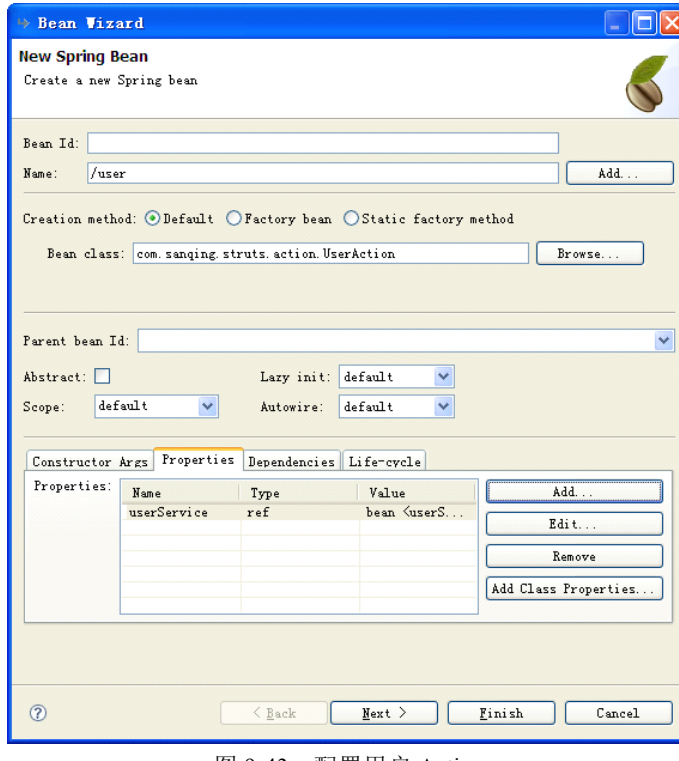


图 9-43　配置用户 Action

单击“Finish”按钮，将完成对用户 Action 的配置。

9.3.11　开发个人博客控制层

在博客网站中成功注册后，就会拥有属于自己的个人博客。注册和登录成功后，就会自动进入自己的博客。在首页中的用户栏中，单击用户超链接，或者在文章标题后单击用户超链接都会进入该用户的个人博客。在本小节中就来开发完成该功能的个人博客控制层。

在创建 Ation 的界面中，输入进入个人博客的必要信息，如图 9-44 所示。

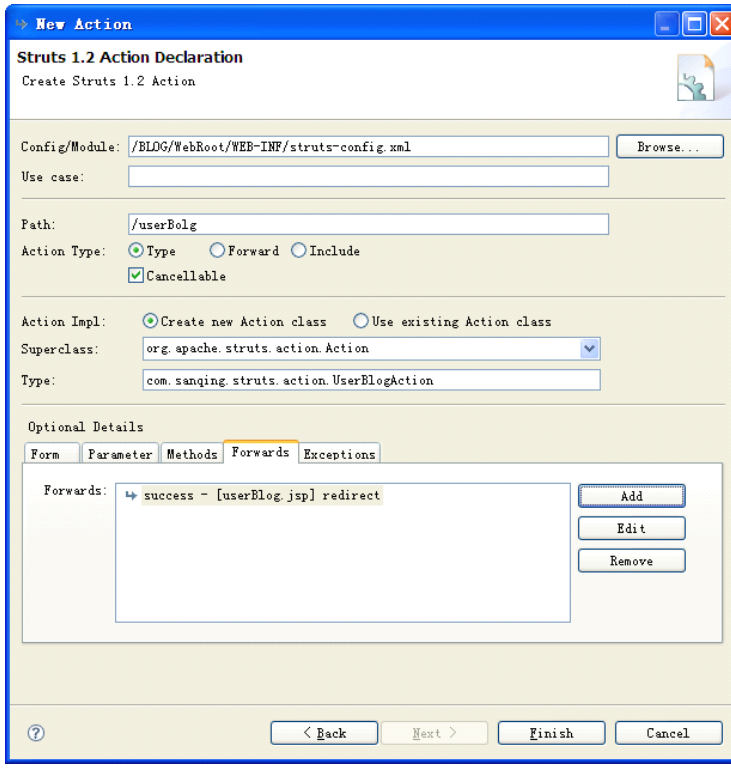


图 9-44　创建个人博客 Action

单击“Finish”按钮，将完成进入个人博客 Acton 的创建。在进入个人博客 Action 将完成获取指定用户所发表文章的集合，并且同样也会将文章的评论数获取出来，它的具体代码如下所示。

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class UserBlogAction extends Action {
04      private ArticleService articleService;
05      private CritiqueService critiqueService;
06      public void setArticleService(ArticleService articleService) {
07          this.articleService = articleService;
08      }
09      public void setCritiqueService(CritiqueService critiqueService) {
10          this.critiqueService = critiqueService;
11      }
12      public ActionForward execute(ActionMapping mapping, ActionForm form,
13              HttpServletRequest request, HttpServletResponse response) {
14          UserForm userForm = (UserForm) form;
15          String username=null;
16          if(userForm.getUsername()==null){     //判断是否接收到参数
17              username=(String)request.getSession().getAttribute("username");
18              request.getSession().setAttribute("self", new Boolean(true));
19          }else{
```

```
20                  username=userForm.getUsername();
21                  request.getSession().setAttribute("self", new Boolean(false));
22              }
23              List<Article> articleList=articleService.showUserArticle(username);
24              List<Integer> countList=new ArrayList<Integer>();
25              for(int i=0;i<articleList.size();i++){
26                  Article article=articleList.get(i);
27                  int count=critiqueService.getCritiqueCount(article.getId());
28                  countList.add(count);
29              }
30              request.setAttribute("articleList", articleList);
31              request.setAttribute("countList", countList);
32              return mapping.findForward("success");
33          }
34      }
```

上述代码中第 4 行和第 5 行中定义了文章和评论业务逻辑接口引用，并为它们定义了 Setter 方法，从而将对象的业务逻辑实现类对象注入。

因为进入个人博客的方式不同，可以是通过注册和登录后进入自己的博客，也可能是进入别人的博客，在其中要进行时哪一种方式进入的。当进入别人博客时会提交对应用户的用户名，从而可以通过获取用户名参数得到。当进入的是自己博客时，是不会提交用户名参数的，可以在 session 中获取用户名。

在第 23 行中通过获取的用户名调用文章业务实现类中的 showUserArticle 方法，从而获取该用户所发表的所有文章。使用首页 Action 中同样的方法可以获取每一个文章的评论数量。

开发完进入个人博客 Action 后，要对它进行配置。同样还是首先修改 Struts 配置文件中的 type 属性值为“org.springframework.web.struts.DelegatingActionProxy”，然后在 Spring 的配置文件中进行配置，配置界面如图 9-45 所示。

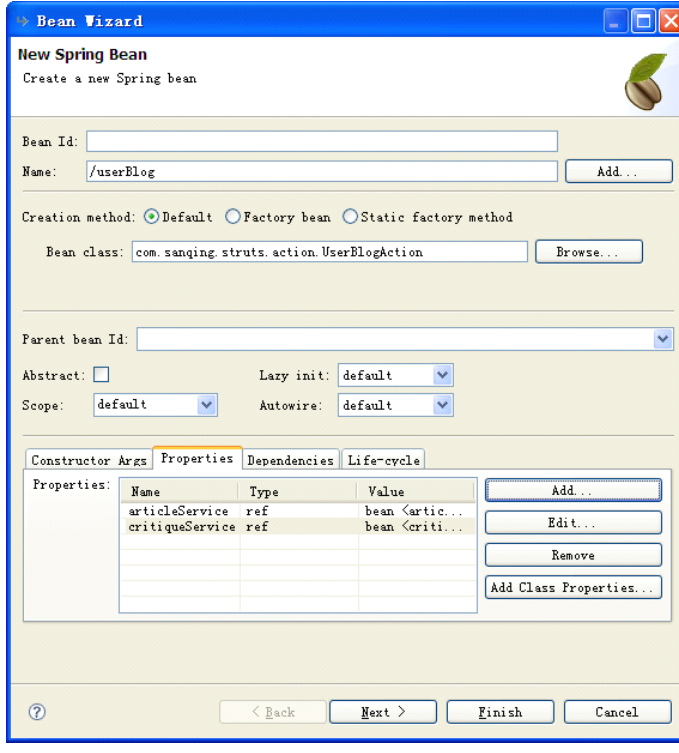


图 9-45　配置个人博客 Action

单击“Finish”按钮，将完成对进入个人博客 Action 的配置，向其中注入了文章业务逻辑实现类对象和评论业务逻辑实现类对象。

9.3.12　开发文章控制层

在博客系统中，对文章的操作是比较多的，所以我们将所有的文章操作放在一个 DispatchAction 中。对文章的操作包括发表文章、查看指定文章、修改文章和删除文章，在执行这些操作时，都要进行接收文章相关参数信息。所以在开发文章 Action 之前，要首先创建文章 ActionForm。

这里我们选择同时创建 JSP 页面、ActionForm 和 Action，首先进入创建 ActionForm 界面，在其中输入相关信息后，界面如图 9-46 所示。

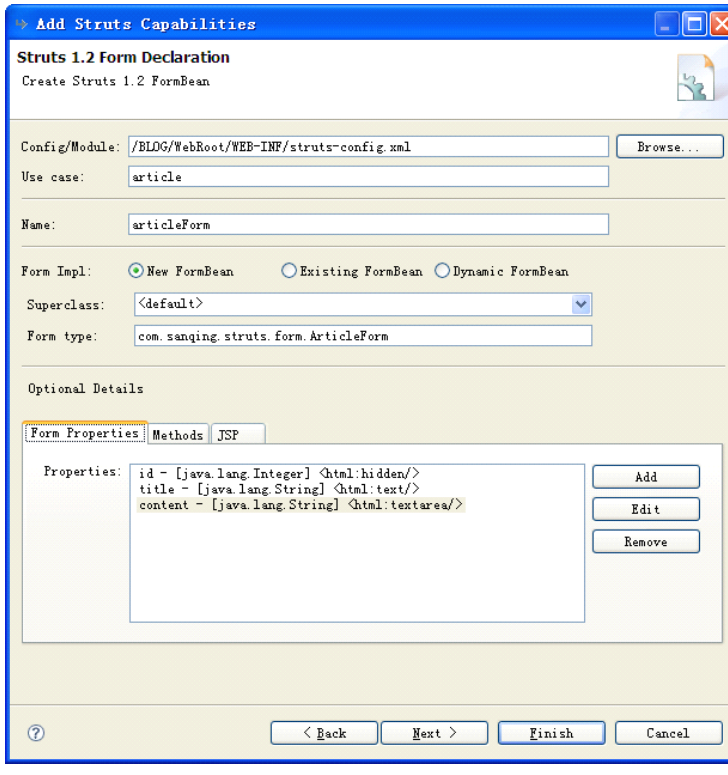


图 9-46　创建文章 ActionForm

单击“Next”按钮，将完成文章 ActionForm 的创建，并进入创建文章 Action 的界面，填写相关信息后，界面如图 9-47 所示。

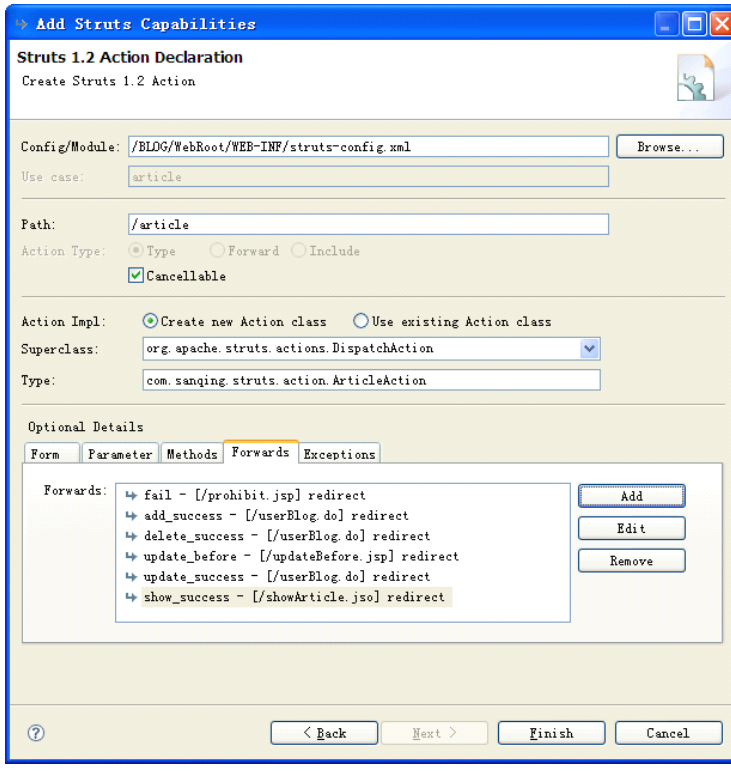


图 9-47　创建文章 Action

单击“Finish”按钮，将完成文章 Action 的创建。然后在文章 Action 中依次定义完成发表文章、查看指定文章、修改文章和删除文章的操作方法，其具体代码如下所示。

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class ArticleAction extends DispatchAction {
04      private ArticleService articleService;
05      public void setArticleService(ArticleService articleService) {
06          this.articleService = articleService;
07      }
08      public ActionForward showArticle(ActionMapping mapping, ActionForm form,
09              HttpServletRequest request, HttpServletResponse response) {//显示文章
10          ArticleForm articleForm = (ArticleForm) form;
11          Article article=articleService.showArticle(articleForm.getId());
12          request.setAttribute("article", article);
13          return mapping.findForward("show_success");
14      }
15      public ActionForward addArticle(ActionMapping mapping, ActionForm form,
16              HttpServletRequest request, HttpServletResponse response) {//发表文章
17          ArticleForm articleForm = (ArticleForm) form;
18          Article article=new Article();
19          article.setTitle(articleForm.getTitle());
20          article.setContent(articleForm.getContent());
```

```
21              article.setUsername((String)request.getSession().getAttribute("username"));
22              article.setDate(new Timestamp(new Date().getTime()));
23              articleService.addArticle(article);
24              return mapping.findForward("add_success");
25          }
26          public ActionForward deleteArticle(ActionMapping mapping, ActionForm form,
27                  HttpServletRequest request, HttpServletResponse response) {//删除文章
28              ArticleForm articleForm = (ArticleForm) form;
29              Article article=articleService.showArticle(articleForm.getId());
30              articleService.deleteArticle(article);
31              return mapping.findForward("delete_success");
32          }
33          public ActionForward updateBefore(ActionMapping mapping, ActionForm form,
34                  HttpServletRequest request, HttpServletResponse response) {//更新前
35              ArticleForm articleForm = (ArticleForm) form;
36              Article article=articleService.showArticle(articleForm.getId());
37              request.setAttribute("article", article);
38              return mapping.findForward("update_before");
39          }
40          public ActionForward updateArticle(ActionMapping mapping, ActionForm form,
41                  HttpServletRequest request, HttpServletResponse response) {//更新文章
42              ArticleForm articleForm = (ArticleForm) form;/
43              Article article=new Article();
44              article.setTitle(articleForm.getTitle());
45              article.setContent(articleForm.getContent());
46              articleService.updateArticle(article);
47              return mapping.findForward("update_success");
48          }
49      }
```

上述代码中第 4 行定义了文章业务逻辑接口引用，并为它定义了 Setter 方法，通过这种方式将文章业务逻辑实现类对象注入。在后面操作代码中，依次定义了显示文章、发表文章、删除文章和更新文章方法，在更新操作中，要首先将原信息显示在页面中，然后执行更新，所以这里有两个方法。

开发完文章控制层后，首先要修改 Struts 配置文件中的 type 属性配置，将它的值修改为“org.springframework.web.struts.DelegatingActionProxy”，然后在 Spring 配置文件中进行配置，配置界面如图 9-48 所示。

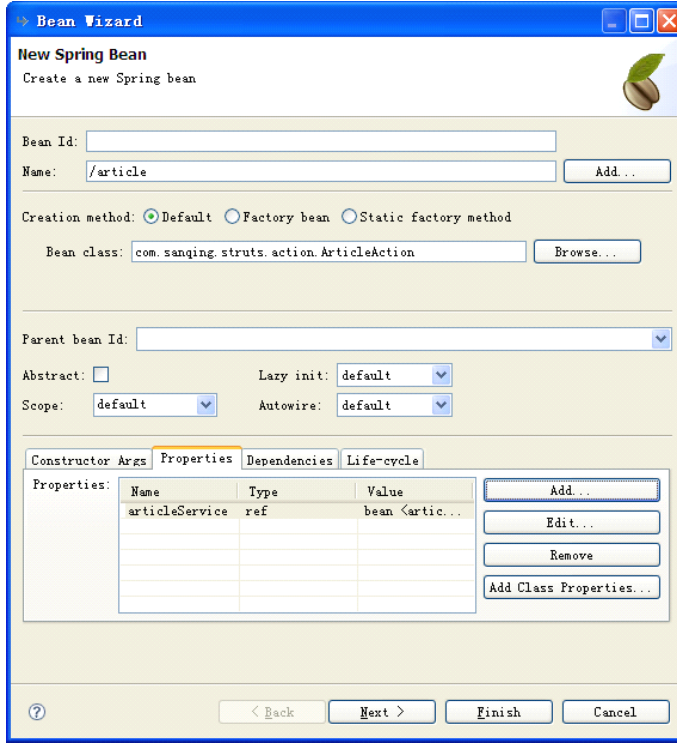


图 9-48　配置文章 Action

单击“Finish”按钮，将完成对文章 Action 的配置，向其中注入了文章业务逻辑实现类对象。

9.3.13　开发评论控制层

对评论的操作要比文章的操作少很多，在其中只需要定义显示文章所有评论和发表新评论两个操作，所以这里也为评论控制层开发一个 DispatchAction。在发表评论时，也是需要需要接收参数的，所以也要定义评论 ActionForm。

这里我们选择同时创建 JSP 页面、ActionForm 和 Action，首先进入创建 ActionForm 界面，在其中输入相关信息后，界面如图 9-49 所示。

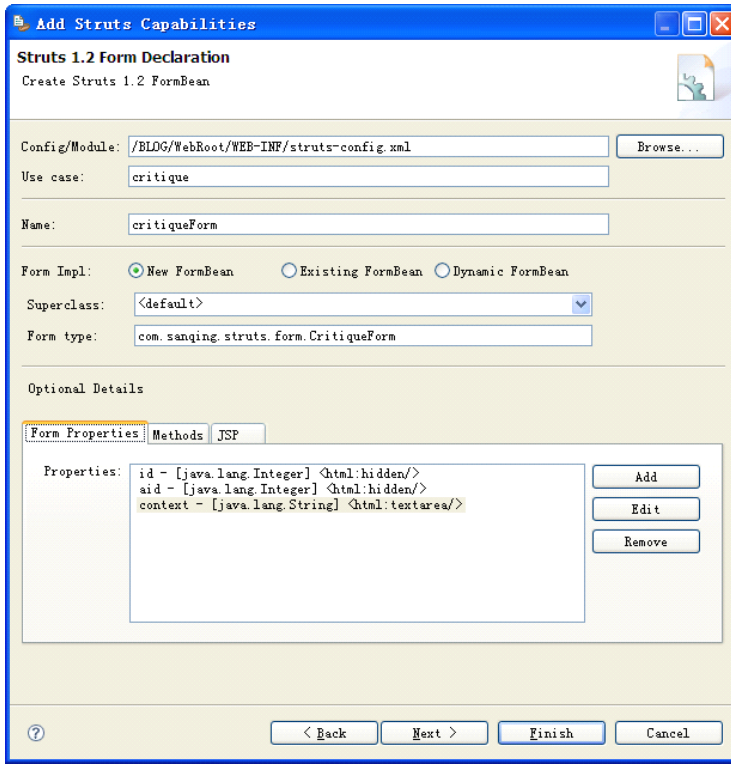


图 9-49　创建评论 ActionForm

单击“Next”按钮，将完成评论 ActionForm 的创建，并且进入创建评论 Aciton 的界面，在其中输入相关信息后，如图 9-50 所示。

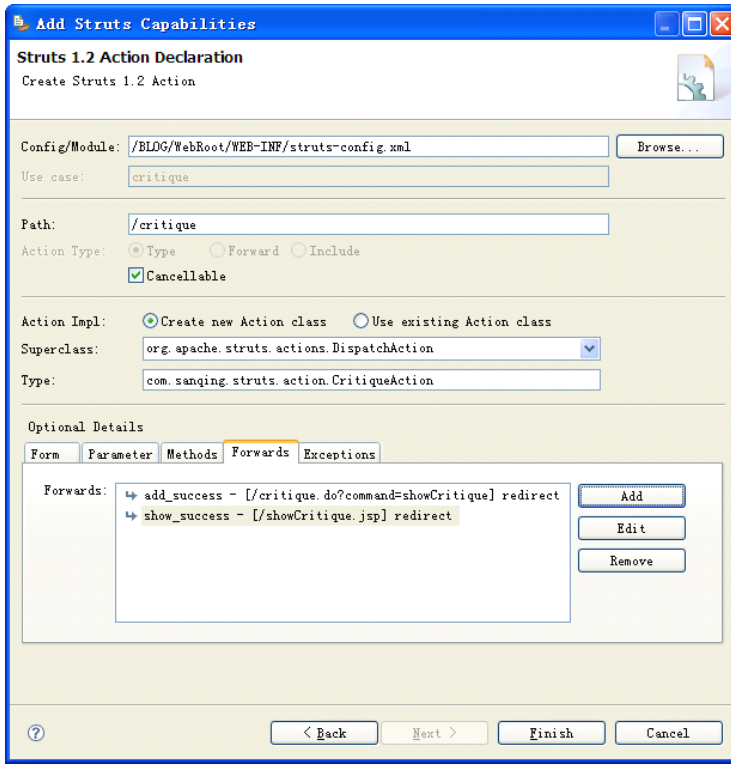


图 9-50　创建评论 Action

单击“Finish”按钮，将完成评论 Action 的创建。然后在 Action 程序中加入显示文章所有评论和发表新评论操作方法，其具体代码如下所示。

```
01  package com.sanqing.struts.action;
02  //省略导入接口和类的代码
03  public class CritiqueAction extends DispatchAction {
04      private CritiqueService critiqueService;
05      public void setCritiqueService(CritiqueService critiqueService) {
06          this.critiqueService = critiqueService;
07      }
08      public ActionForward showCritique(ActionMapping mapping, ActionForm form,
09              HttpServletRequest request, HttpServletResponse response) {//显示评论
10          CritiqueForm critiqueForm = (CritiqueForm) form;
11          List<Critique> critiqueList=
12                  critiqueService.showAllCritique(critiqueForm.getAid());
13          request.setAttribute("critiqueList", critiqueList);
14          return mapping.findForward("show_success");
15      }
16      public ActionForward addCritique(ActionMapping mapping, ActionForm form,
17              HttpServletRequest request, HttpServletResponse response) {//发表评论
18          CritiqueForm critiqueForm = (CritiqueForm) form;
19          Critique critique=new Critique();
20          critique.setContent(critiqueForm.getContext());
```

```
21                  critique.setAid(critiqueForm.getAid());
22                  String username="匿名";
23                  if(request.getSession().getAttribute("username")!=null){
24                      username=(String)request.getSession().getAttribute("username");
25                  }
26                  critique.setUsername(username);
27                  critiqueService.addCritique(critique);
28                  return mapping.findForward("add_success");
29              }
30      }
```

上述代码中第 4 行定义了评论业务逻辑接口引用，并为它定义了 Setter 方法，通过这种方式将评论业务逻辑实现类对象注入。在后面的操作中，依次定义了显示文章所有评论和发表评论的操作方法。

开发完评论 Action 代码后，就是对评论 Action 进行配置。首先修改 Struts 配置文件中的 type 属性值为“org.springframework.web.struts.DelegatingActionProxy”。然后在 Spring 配置文件中进行配置，配置界面如图 9-51 所示。

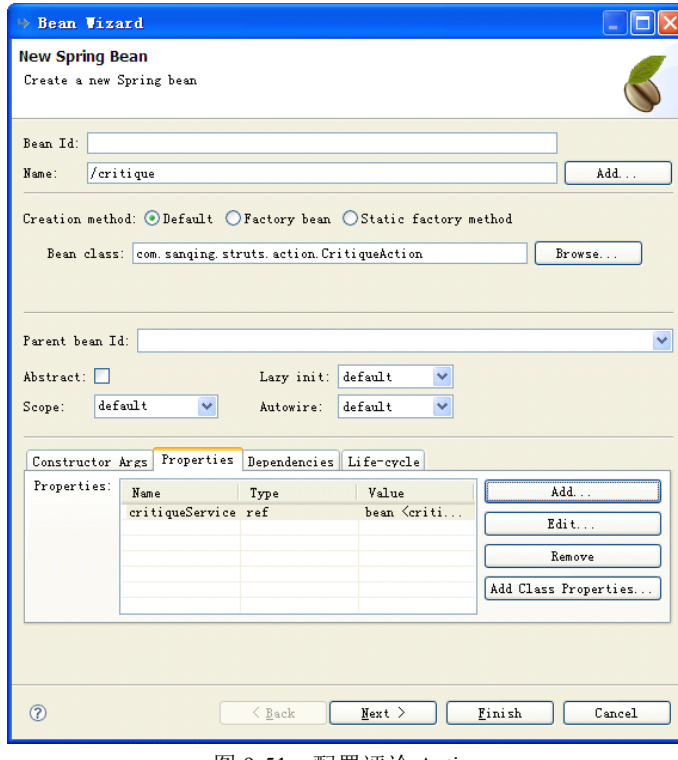


图 9-51　配置评论 Action

单击“Finish”按钮，完成对评论 Action 的配置，向其中注入了评论业务逻辑实现类对象。

到这里为止，该博客网站项目就已经开发完成了，读者可以自己开发显示层程序，也就是 JSP 程序。显示层的开发是非常简单的，这里就不再详细讲解。

第10章　进行Struts 2开发

Struts 2 框架和前面讲解的 Struts 框架有很大的差别。Struts 2 是以 WebWork 为核心，采用拦截器的机制来处理用户的请求，这样的设计也是可以使得业务逻辑控制器能够与 Selvet API 完全的脱离开，所以 Struts 2 可以理解为是 WebWork 的更新产品。但是由于 Struts 的名气，从而使新框架延续该名称。

随着 Struts 2 的发展，以及慢慢取代了 Struts 1 的霸主地位，越来越多的项目都在向 Struts 2 迁移，在本节中就来学习一下如何在 MyEclipse 中进行 Struts 2 的项目开发。

说明：在本书讲解中，使用“Struts 2”就表示的是 Struts 2 框架，如果没有特别说明，就是指原 Struts 框架。

10.1　使用插件开发 Struts 2 项目

Struts 2 是目前最流行的 MVC 框架，基于 Struts 2 来进行开发可以大大减少开发时间，提高开发效率，并降低后期维护时间和精力。安装 MyEclipse 后并没有集成开发 Struts 2 项目的功能，对于低版本的 MyEclipse 而言，只能够手动搭建 Struts 2 项目开发环境。但是在 MyEclipse 7.5 版本中，可以在线安装 Struts 2 项目开发插件。

10.1.1　安装 Struts 2 插件

在 MyEclipse 中，所有的功能都是通过插件完成的。只是有些插件在安装 MyEclipse 时就已经默认安装了，例如前面介绍的 Hibernate 和 Spring 开发。但是对于 Struts 2 来说，默认安装 MyEclipse 时，是不会自动安装它的插件的，所以需要手动安装。

安装插件有两种方式，分别是在线安装和本地安装，这里我们选择比较容易的在线安装。在 MyEclipse 的菜单中，选择“Help”|“Software update”|“Add/Remove Software”命令，经过一段时间后，将弹出如图 10-1 所示的界面。

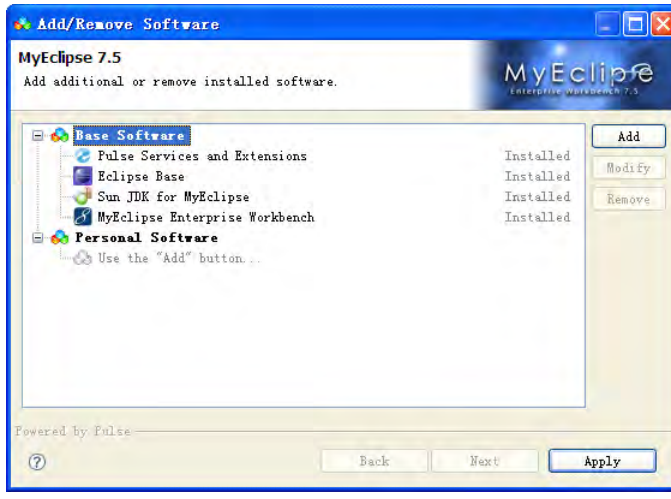


图 10-1　选择安装插件

目前是没有 Struts 2 插件的，单击“Add”按钮，将弹出增加或者删除插件的界面，如图 10-2 所示。

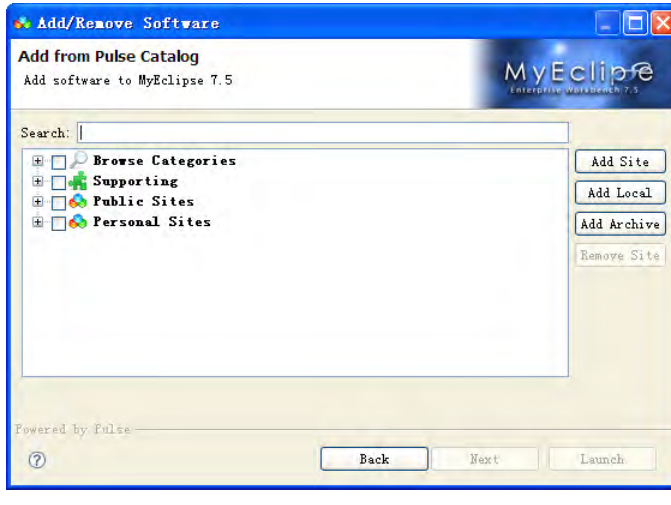


图 10-2　增加和删除插件

单击“Add Site”按钮，将弹出连接下载地址的操作界面，如图 10-3 所示。

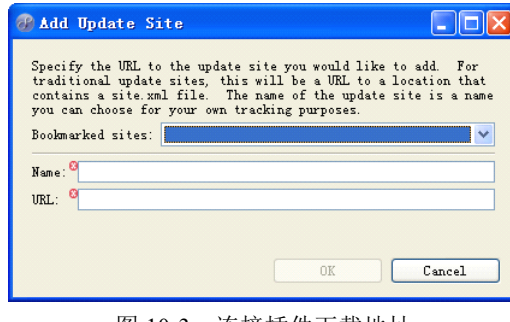


图 10-3　连接插件下载地址

在界面中 Name 文本框中输入“Struts 2”，URL 文本框中输入“http://mvcwebproject.sourceforge.net”，单击“OK”按钮，回到增加插件界面，将其中的“Personal Sites”节点展开，如图 10-4 所示。

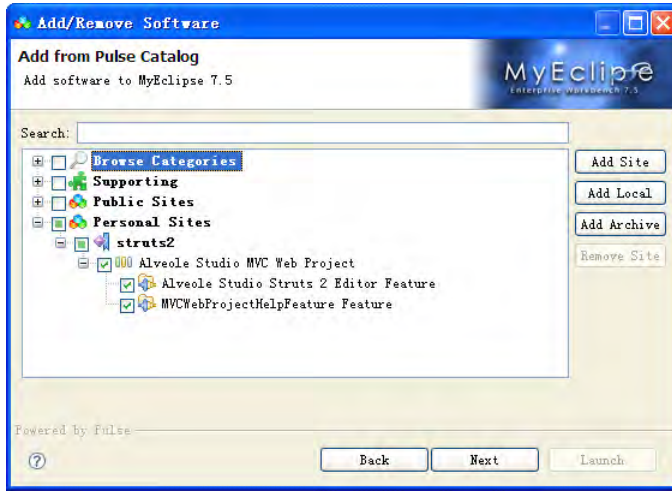


图 10-4　查找到 Struts 2 插件

单击“Next”按钮，将进入选择安装 Struts 2 插件的界面，如图 10-5 所示。

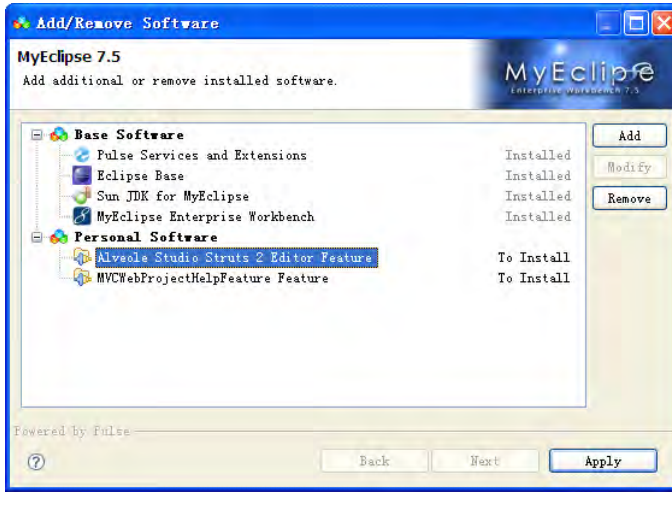


图 10-5　选择安装插件

在其中选中“Alveole Studio Struts 2 Editor Feature”节点，单击“Apply”按钮将进入更新 MyEclipse 的界面，也就是安装 Struts 2 插件的界面，如图 10-6 所示。

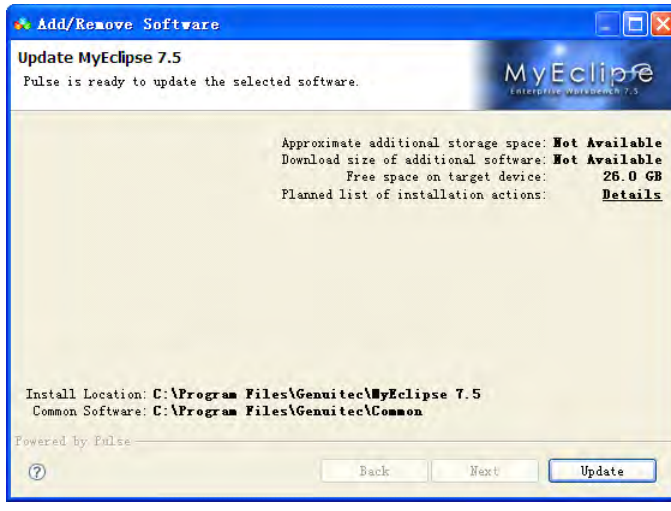


图 10-6　更新 MyEclipse

单击“Update”按钮，将开始更新 MyEclipse，主要是安装 Struts 2 插件。该步骤根据本地的网速不同，花费的时间也不同。更新完成后，将出现一个选择是否重启 MyEclipse 的对话框，在其中单击“Yes”按钮，重启 MyEclipse，重启后 Struts 2 插件就安装完成。

说明：所有的在线安装操作都是非常类似的，在后面的学习中还会使用到其他插件。如果本地网速并不好，笔者建议采用本地安装的方式，将插件文件先放置在 MyEclipse 文件中，然后重启 MyEclipse。具体操作读者可以在网络上搜索一下，是不难操作的。

10.1.2　开发 Struts 2 项目

Struts 2 框架也是只使用在 Web 项目中的，所以要首先创建一个普通的 Java Web 项目。在 MyEclipse 的菜单中，选择“File”|“New”|“Project…”命令，将弹出选择创建程序界面。选中“Show all Wizards”选项，在其中选择“Web”|“Dynamic Web Project”节点，界面如图 10-7 所示。

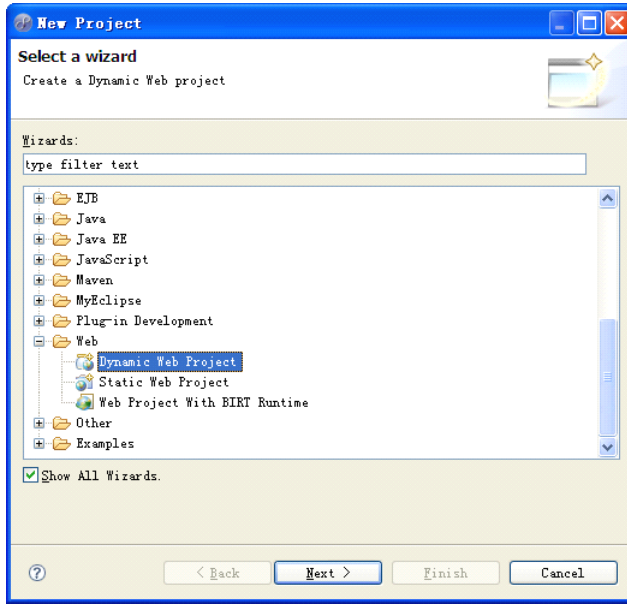


图 10-7　选择创建程序

单击“Next”按钮，将弹出创建动态 Web 项目的界面，在其中输入项目的名称为“FirstStruts 2”，界面如图 10-8 所示。

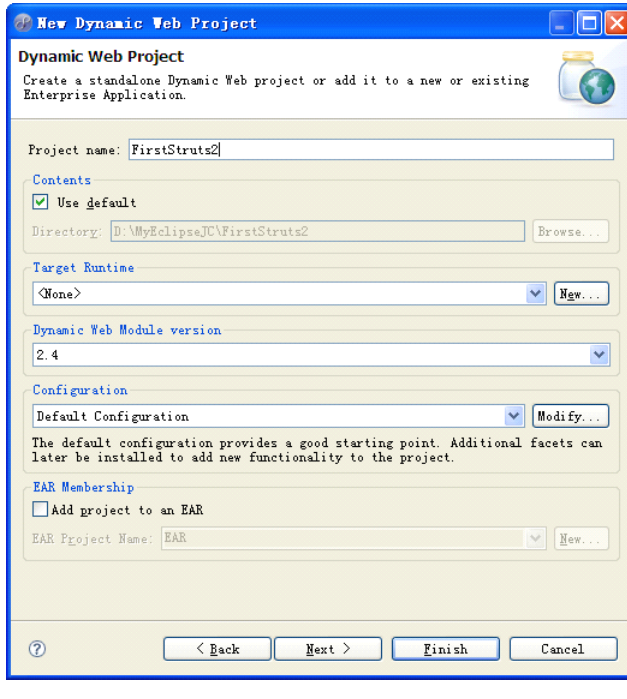


图 10-8　创建动态 Web 项目

单击“Finish”按钮，将完成动态 Web 项目的创建。在包资源管理器中，选中“FirstStruts 2”项目节点，单击鼠标右键，在弹出的菜单中选择“New”|“Other”命令，将再次弹出选择创建程序界面，在其中选择“Web”|“Alveole Studio MVC Web Project”节点，

界面如图 10-9 所示。

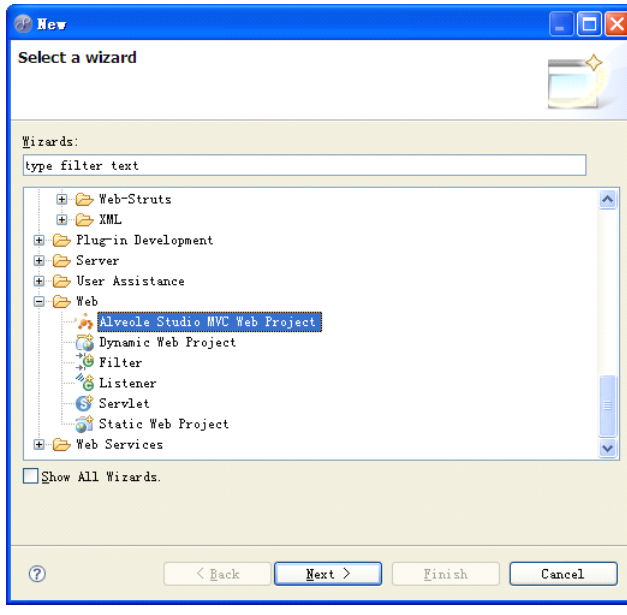


图 10-9　创建 Struts 2 项目

单击“Next”按钮，将进入创建 Struts 2 项目工厂文件的界面，如图 10-10 所示。

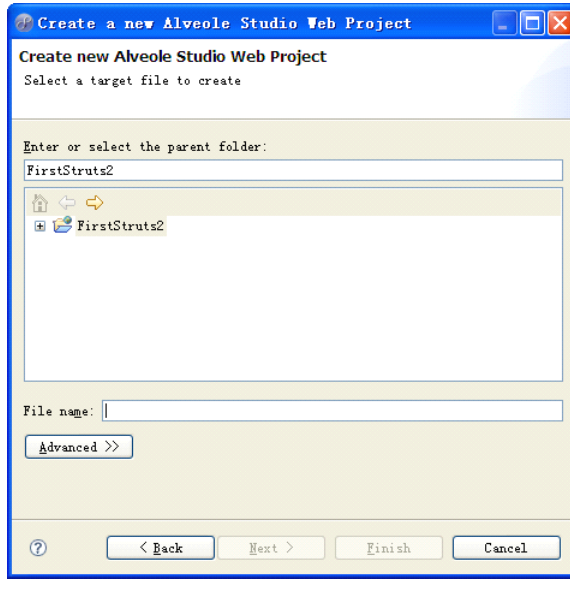


图 10-10　创建工程文件

在其中选择“FirstStruts2”项目，然后在“File name”文本框中输入工厂文件名称，这里输入“project.aswp”。单击“Next”按钮，将进入填写项目描述信息的界面，如图 10-11 所示。

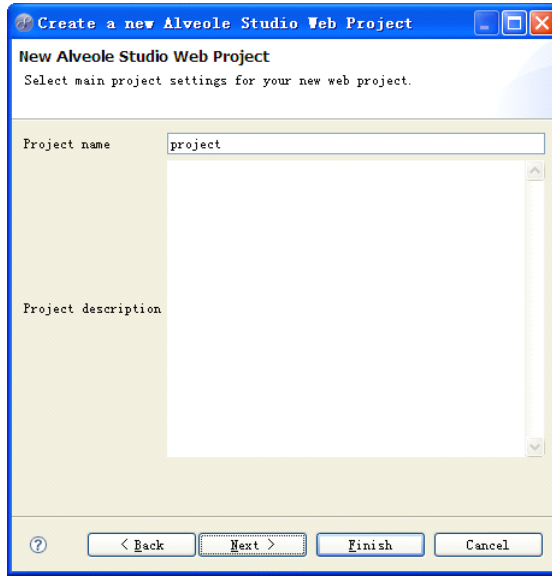


图 10-11　项目描述

这里保持默认的不填写任何内容，单击“Next”按钮，进入到添加 Struts 2 相关类库的界面，如图 10-12 所示。

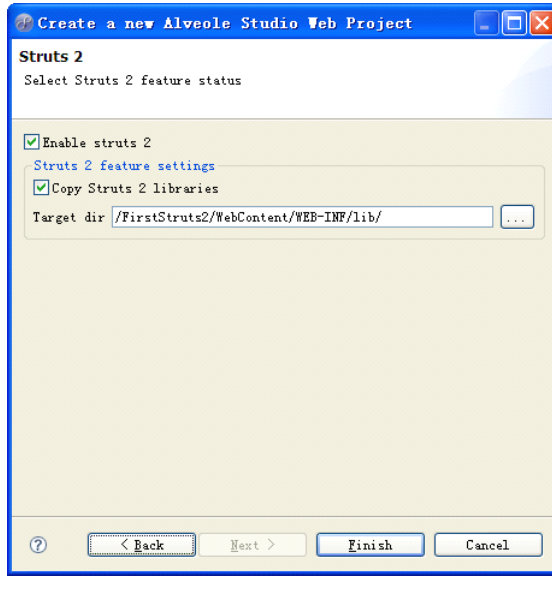


图 10-12　添加 Struts 2 类库

其中“Target dir”表示向哪一个目录中添加类库，在默认情况下会指定放置目录的，默认值是“/FirstStruts2/WebContent/WEB-INF/lib/”。单击“Finish”按钮，将完成 Struts 项目的创建。

10.1.3　认识 Struts 2 项目

开发完 Struts 2 项目后，我们来简单的认识一下 Struts 2 项目结构。在包资源管理器中，将它展开，如图 10-13 所示。

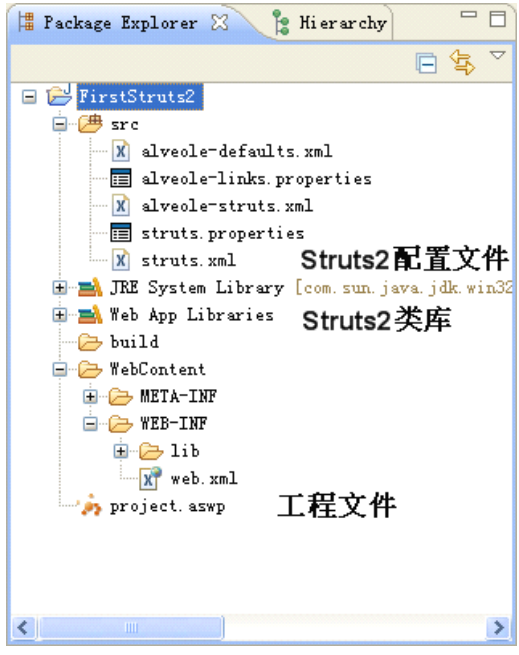


图 10-13　Struts 2 项目结构

其中“Web App Libraries”中包含了 Struts 2 开发中所用到的所有 JAR 包，其中是比较多了，如果仅进行基本的 Struts 2 项目开发，将只使用到其中的 Struts 核心包。“web.xml”文件是 Web 项目中都会有的，双击打开该文件，可以发现其中对 Struts 2 的核心控制器进行了配置，具体配置代码如下所示。

```
01　<filter>
02　　<filter-name>org.apache.struts2.dispatcher.FilterDispatcher</filter-name>
03　　<filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
04　</filter>
05　<filter-mapping>
06　　<filter-name>org.apache.struts2.dispatcher.FilterDispatcher</filter-name>
07　　<url-pattern>*.action</url-pattern>
08　</filter-mapping>
```

在其中第 7 行中配置了当所有以“action”结尾的请求都由和核心控制器进行处理。

注意：当自己手动搭建 Struts 2 开发环境时，经常在第 7 行中写入“/*”，表示处理所有请求，它们两种都是可以的。但是不要写入“/*.action”，它只表示根目录下的以“action”结尾的请求。

“struts.xml”是 Struts 2 项目中的核心配置文件，在后面开发 Action 程序时，都要在该文件中进行配置。“project.aswp”文件就是在使用 Struts 2 插件开发 Struts 2 项目时创建的工程文件，通过该文件可以进行视图化的创建 Struts 2 程序。双击打开该文件，

它有多种视图选项，主要使用其中的“Tree”选项，选择它，操作界面如图 10-14 所示。

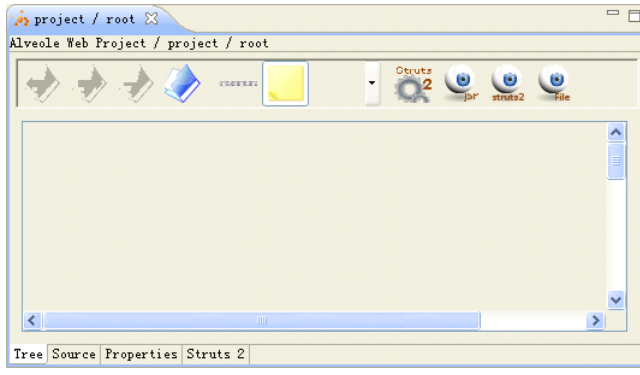


图 10-14　视图创建 Struts 2 程序

在后面的操作中，我们主要通过该界面进行各种程序的开发。

10.2　使用插件开发 Struts 2 程序

开发完 Struts 2 项目后，就继续来学习如何通过插件开发 Struts 2 项目中的程序。Struts 2 中最重要的程序那就是 Action，它是 Struts 2 项目的核心，它从 JSP 页面程序中接收参数，然后处理后跳转到其他页面。

10.2.1　创建 Action 程序

双击打开“project.aswp”文件，选择编辑区左下角的“Tree”选项，将以界面的形式打开该文件。在界面的最上面是用于创建 Struts 2 程序的功能按钮，其中“ 

”就是用来创建 Action 的按钮。单击 Action 按钮，将打开用于创建 Action 的视图界面，如图 10-15 所示。

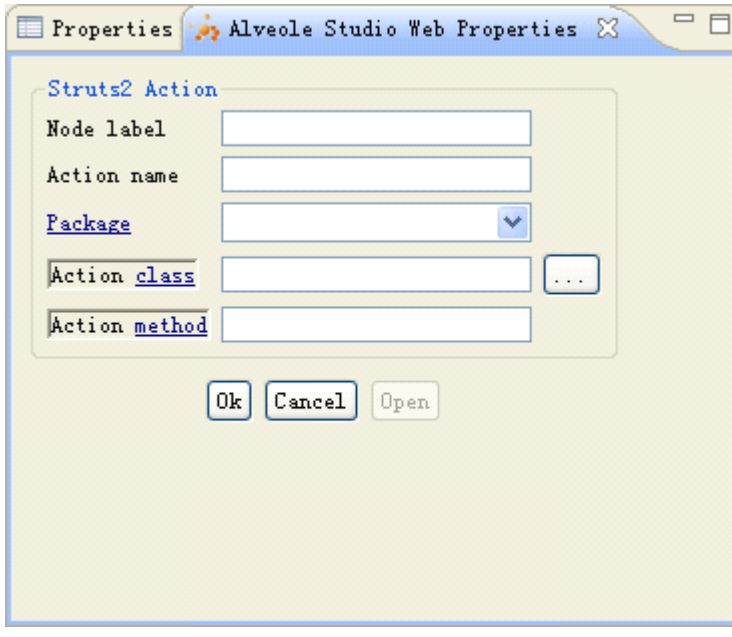


图 10-15　创建 Action 视图界面

其中“Node label”表示节点标签，也就是在界面中显示的内容，这里填写“register action”。“Action name”表示访问创建的 Action 时所使用的 URL，这里填写“register”。

注意：请求 URL 中，既没有前面的“/”，也没有后面的“.action”，只需要给出名称就可以。

“Package”表示创建 Action 所在的配置包，不填写表示使用默认内容。“Action class”表示创建 Action 的类名，这里一定要填写包括包名的完整类名，如“com.sanqing.action.RegisterAction”。“Action method”表示 Action 中创建的控制方法，这里填写“register”。单击“OK”按钮，将完成注册 Action 的创建，并将其显示在创建操作界面中，如图 10-16 所示。

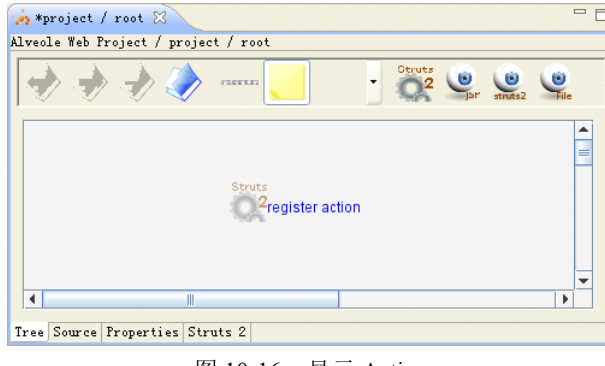


图 10-16　显示 Action

双击其中的图标将弹出创建 Action 类的界面，如图 10-17 所示。

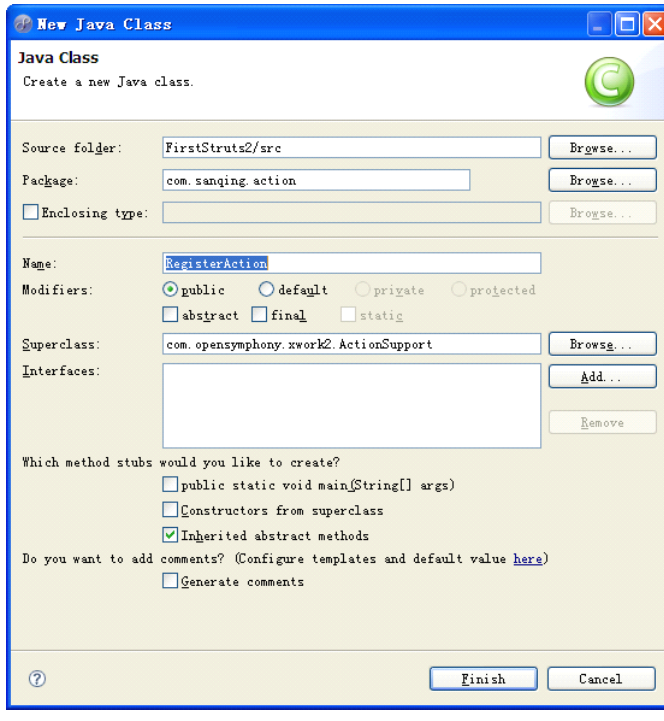


图 10-18　创建 Action 类

从界面中可以看到，创建 Action 程序是和创建普通的 Java 类是没有区别的，只是其中让创建的 Action 继承“com.opensymphony.xwork2.ActionSupport”类。单击“Finish"按钮，将完成注册 Action 的创建。

创建 Action 程序后，将自动在编辑区中打开相应的程序，自动创建的程序代码为：

```
01    package com.sanqing.action;
02    import com.opensymphony.xwork2.ActionSupport;
03    public class RegisterAction extends ActionSupport {
04        public String register() {
05            return "success";
06        }
07    }
```

在这些代码的基础上可以加入必要的功能代码，这里我们就仅让 Action 执行跳转这一简单操作。

10.2.2　创建 JSP 显示程序

在 Struts 2 插件中也集成了创建 JSP 显示程序的功能，单击功能按钮工具栏中的“  ”按钮，将打开用于创建 JSP 程序的视图界面，如图 10-19 所示。

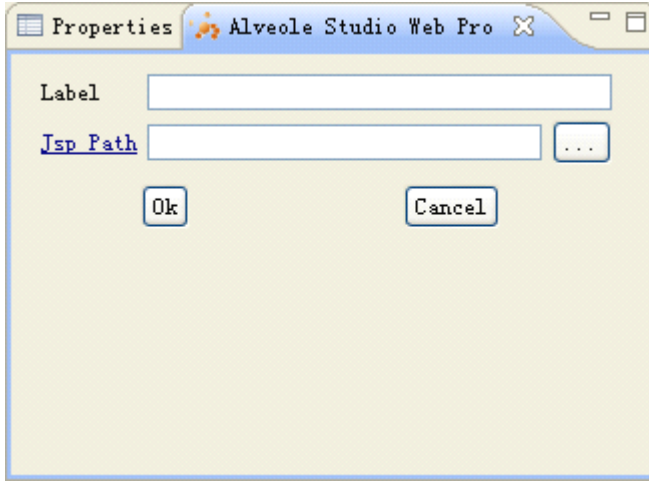


图 10-19　创建 JSP 程序

其中“Label”就是用于在界面中显示的内容，填写“register success JSP”。“Jsp Path”表示 JSP 程序的访问 URL，这里填写“registerSuccess.jsp”。单击“OK”按钮，将完成 JSP 程序的创建，从而也会将它的图标显示在界面中，如图 10-20 所示。

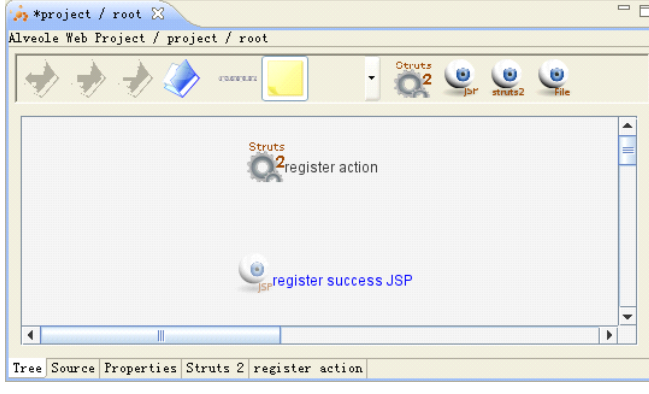


图 10-20　显示 JSP 程序

双击 JSP 程序图标，将打开创建 JSP 程序的界面，如图 10-21 所示。

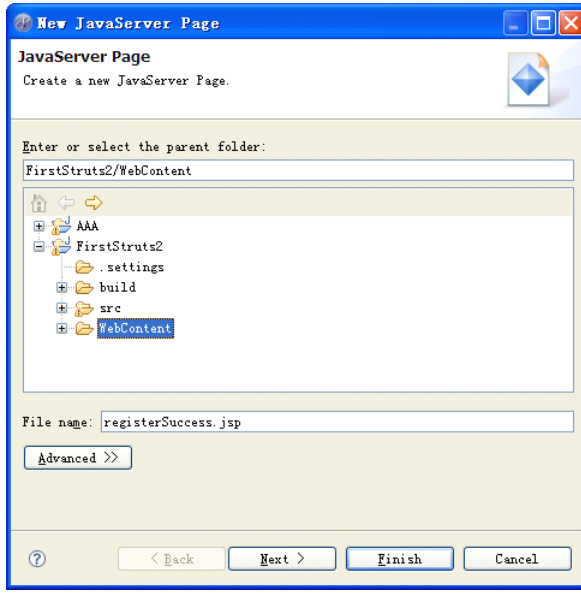


图 10-21　创建 JSP 程序文件

单击“Finish”按钮，将完成注册成功页面程序的创建。JSP 程序的编写时非常简单的，只需要让它输出“注册成功”信息就可以。

10.2.3　创建链接

当正确执行注册 Action 后将跳转到注册成功页面，在本节中就来通过插件创建 Action 和 JSP 页面的链接。在功能工具栏中，单击“  ”按钮，然后先选择 Action 图标，一直拖动到 JSP 图标，这时候将打开创建连接的视图界面，如图 10-22 所示。

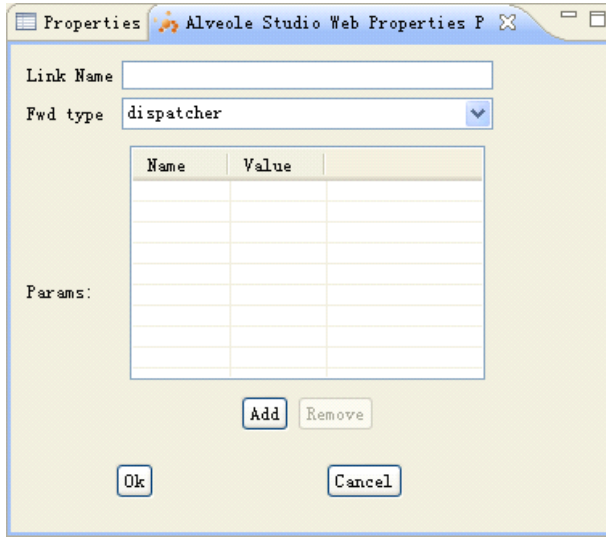


图 10-22　创建链接视图界面

其中“Link Name”表示链接的名称，也就是 Action 中的返回值，这里填写“success”。“Fwd type”表示跳转类型，采用默认的“dispatcher”就可以。“Params”表示跳转时传递的参数，不填写表示不传递参数。单击“OK”按钮，将完成链接的创建。创建链接后，显示界面如图 10-23 所示。

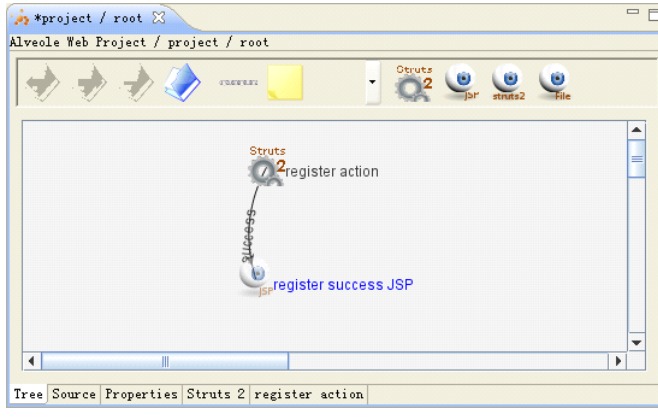


图 10-23　创建链接后界面

保存工程文件，将完成 Struts 2 程序的创建，并完成对它们的配置。打开其中的“alveole-struts.xml”文件，可以看到配置代码，如下所示。

```
01    <package extends="alveole-defaults" name="alveole" namespace="/">
02        <action class="com.sanqing.action.RegisterAction"
03                                        method="register" name="register">
04            <result name="success" type="dispatcher">
05                /registerSuccess.jsp
06            </result>
07        </action>
08    </package>
```

其中第 2 行对注册 Action 进行了配置，第 4 行中配置了当返回值为“success”是跳转到第 5 行的注册页面。

10.2.4　运行 Struts 2 项目

因为使用 Struts 2 插件使用的并不是前面 Web 一章中介绍的 Java Web 项目，所以它的部署和运行也是有所不同的。在包资源管理器中，选中“FirstStruts2”项目节点，单击鼠标右键，选择“Run Ads”|“Run on Server”命令，将弹出发布界面，在其中选择 Tomcat 服务器，如图 10-24 所示。

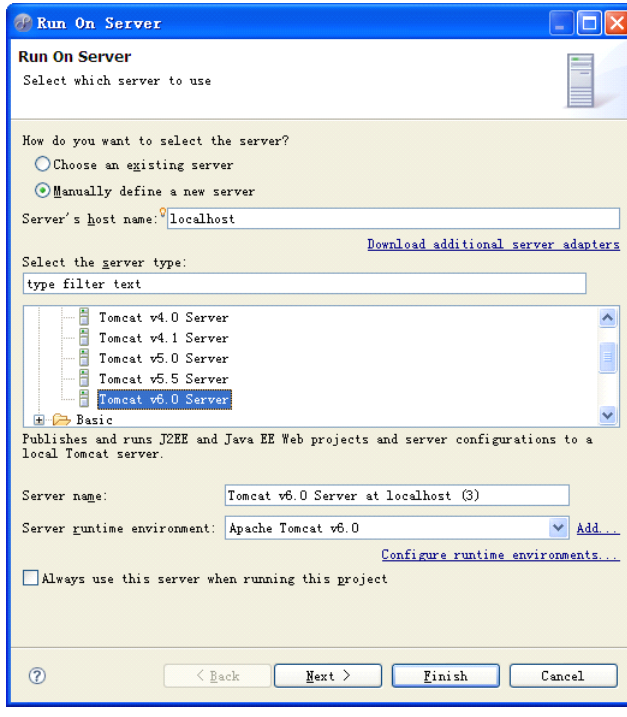


图 10-24　发布 Struts 2 项目

单击“Finish”按钮，将完成 Struts 2 项目的发布，并且在编辑区中将打开一个虚拟浏览器，在其中输入如下地址：

http://localhost:8080/FirstStruts2/register.action

则界面如图 10-25 所示。

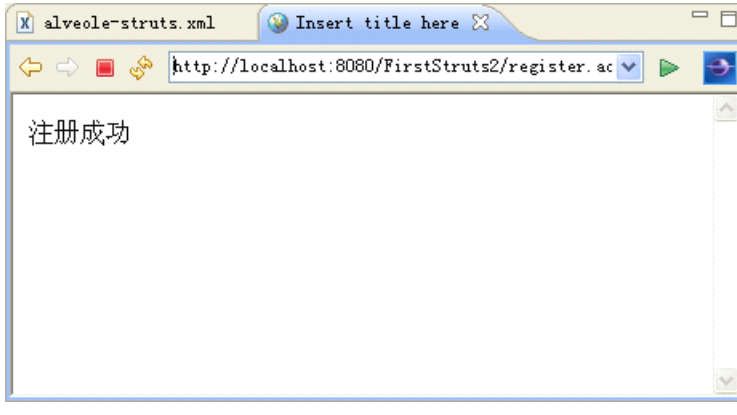


图 10-25　显示结果

当方法注册 Action 后，返回值为“success”，将执行跳转，从而跳转到注册成功 JSP 程序，从而出现该界面。到这里我们就通过插件开发了一个非常简单的 Struts 2 项目，并且运行成功。

10.3 手动搭建 Struts 2 开发环境

使用 Struts 2 插件虽然能够视图化的创建 Struts 2 项目和程序，但是操作起来远比前面三种框架的复杂。而且该插件和 MyEclipse 集成并不好，它更倾向于和 Eclipse 集成。所以在本节中将来学习一下如何手动的搭建 Struts 2 开发环境，它也是不难操作的。

10.3.1 下载和安装 Struts 2 类库

Struts 2 框架是由 Apache 开源组织开发并负责维护的，它的官方网站为“http://struts.apache.org”。目前 Struts 2 最新版本是 2.1.8，它的下载地址是“http://struts.apache.org/download.cgi#struts2181”。其中有多个版本的下载链接，如图 10-26 所示。

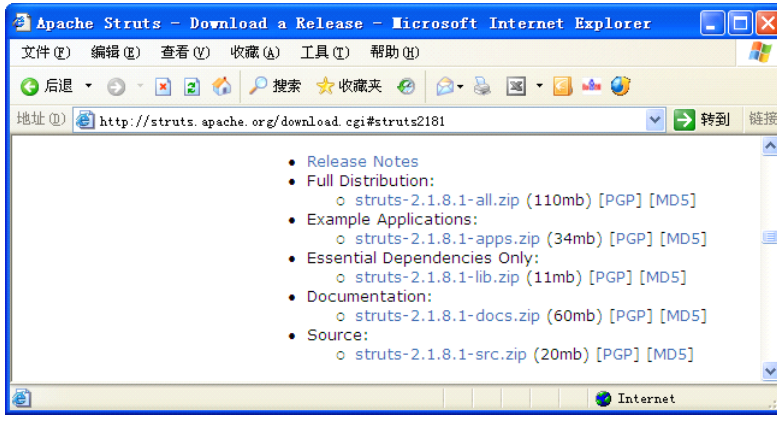


图 10-26 下载版本

在页面包括如下下载链接。

- Full Distribution：为完整版的下载，包括了下面所有的文件，建议下载这个。
- Example Applications：为 Struts 2 的示例程序，这些示例程序在以后的学习中会有很大的帮助。
- Essential Dependencies Only：为 Struts 2 的核心库。
- Documentation：为 Struts 2 的相关文档，包含 Struts 2 的使用文档、参考手册和 API 文档等。有很好的参考价值。
- Source：为 Struts 2 的源代码。因为 Struts 2 也是开源框架，提供源代码以供参考。

这里我们现在完整版本，单击“Full Distribution”下面的文件链接，将开始下载。下载完成后，对下载文件进行解压，其中包括如下几个文件夹。

- apps：该文件夹中存放 Struts 2 的示例程序。
- doc：该文件夹中存放 Struts 2 的相关文档。
- lib：该文件夹中存放 Struts 2 的核心类库，以及第三方的 JAR 文件。
- src：该文件夹中存放 Struts 2 框架的全部源代码。

在 lib 下包含了 Struts 2 开发的核心 JAR 包，其中最重要的就是 common-logging-1.0.4.jar 、 commons-fileupload-1.2.1.jar 、 freemarker-2.3.15.jar 、 struts2-core-2.1.8.1.jar、xwork-2.1.6.jar 和 ognl-2.7.3.jar 这几个 JAR 包，将它们复制到项目中。例如我们这里创建一个“SecondStruts2”目录，将核心 JAR 包复制到“WebRoot”|“WEB-INF”|“lib”目录下，这样就搭建了 Struts 2 项目的开发环境。

说明：上述导入的 JAR 包只是核心 JAR 包，如果需要进行更多的功能，还要导入其他的 JAR 包，这些都在 lib 目录下。

10.3.2 配置 Web 描述符

在 Web 项目中，“web.xml”文件是必不可少的，当进行 Struts 2 项目开发时，在该文件中需要对 Struts 2 的核心控制器进行配置。配置代码如下所示。

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <web-app version="2.5"
03      xmlns="http://java.sun.com/xml/ns/javaee"
04      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
06      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
07      <filter>
08          <filter-name>struts2</filter-name>
09          <filter-class>
10              org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
11          </filter-class>
12      </filter>
13      <filter-mapping>
14          <filter-name>struts2</filter-name>
15          <url-pattern>/*</url-pattern>
16      </filter-mapping>
17      <welcome-file-list>
18          <welcome-file>index.jsp</welcome-file>
19      </welcome-file-list>
20  </web-app>
```

从配置所使用的标记可以看到，是对过滤器进行配置。在第 10 行中配置的过滤器类是“org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter”，它是 Struts 2 框架的核心控制器类。第 15 行中使用“/*”表示对所有请求进行过滤由 Struts 2 核心控制器进行处理。

说明：控制层的框架都有有一个核心控制器来处理请求，它们的作用都是在其中分析请求，然后调用不同的处理类，在 Struts 2 中就是调用 Action。

10.3.3 创建 Struts 2 配置文件

和其他框架一样，Struts 2 框架也有自己的配置文件，在其中完成对 Action 的配置。

Struts 2 的配置文件通常命名为“struts.xml”，它和 Struts 配置文件的名称是不同的。Struts 2 配置文件要放置在项目的“WebRoot”|“WEB-INF”|“classes”目录下，对于 MyEclipse 中的 Web 项目来说，只需要放在“src”目录下就可以，当发布项目后，将自动放置到“WebRoot”|“WEB-INF”|“classes”目录下。

Struts 2 配置文件的基础模板可以参考 Struts 2 下载包中的实例中的配置文件。在包资源管理器中，选中“SecondStruts2”项目中的“src”目录，单击鼠标右键，在弹出的菜单中选择“New”|“XML(Basic Templates)”命令，将弹出创建 XML 文件的界面，如图 10-27 所示。

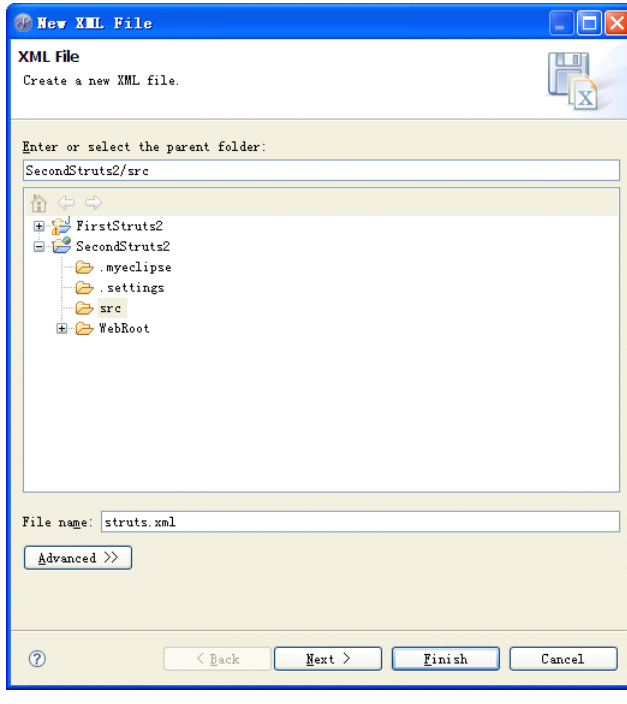


图 10-27　创建 Struts 2 配置文件

单击“Finish”按钮，将完成 Struts 2 配置文件的创建。在编辑区中打开该文件，复制如下代码到文件中。

```
01  <?xml version="1.0" encoding="UTF-8" ?>
02  <!DOCTYPE struts PUBLIC
03      "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
04      "http://struts.apache.org/dtds/struts-2.0.dtd">
05  <struts>
06      <package name="default" extends="struts-default">
07
08      </package>
09  </struts>
```

其中第 5 行的<struts>标记是 Struts 2 配置文件的根标记，对于不同类的 Action 来说可以定义不同的 package。在根标记下创建<package>标记来包含一类 Action。

10.4 手动开发 Struts 2 程序

在前面已经讲解了通过 Struts 2 插件的方式开发 Struts 2 程序，在本节中将讲解通过手动的方法开发 Struts 2 程序，这也不是困难的。在本节中通过一个简单的登录功能来进行讲解，其中包括填写登录的表单页面程序、进行处理的登录 Action 程序和显示登录结果的页面程序。

10.4.1 开发登录页面程序

Struts 2 框架也是拥有自己独立的一套标签库的，而且和 Struts 框架中的标签完全不同。但是这里为了简单，只使用 HTML 中的表单标记来定义一个用于登录的表单。如何创建 HTML 程序这里就不在详细讲解了，表单代码如下所示。

```
<body>
    <form action="login.action" method="post">
      用户名称：<input type="text" name="username"><br>
      用户密码：<input type="password" name="password"><br>
      <input type="submit" value="登录">
    </form>
  </body>
```

在该表单中创建了用于输入用户名的文本框和用于输入密码的密码框。执行登录处理后，如果登录成功，将跳转到登录成功页面；如果登录失败，将跳转到登录失败页面。在该登录项目中，在结果页面中只显示结果信息，具体代码读者可以查看光盘。

注意：在提交表单的 action 属性指定的提交 URL 中，一定要以“.action”结尾，并且前面没有“/”。

10.4.2 开发处理登录 Action

在 Struts 2 项目中，也通过 Action 程序来处理功能，但是 Struts 2 中的 Action 程序是和 Struts 项目中的有很大不同的。在 Struts 2 项目的 Action 中并不借助 ActionForm 来接收参数值，而是通过自身来接收，所以在 Action 程序中要定义和表单对应的属性，并需要为这些属性定义相应的 Setter 和 Getter 方法。

Struts 2 项目中的 Action 中的处理方法名称可以是“execute”，也可以是其他的名称。Struts 2 中的 Action 是可以和框架相分离的，也就是不需要继承框架中的类，但是在开发中通常让它继承“com.opensymphony.xwork2.ActionSupport”，是否继承都是能够完成处理功能的。

Struts 2 中的 Action 程序通常也是放在“action”包下的，选中该包，单击鼠标右键，在弹出的菜单中选择“New”|“Class”命令，将弹出创建 Java 类的界面，在其中输入 Action 的相关信息后，如图 10-28 所示。

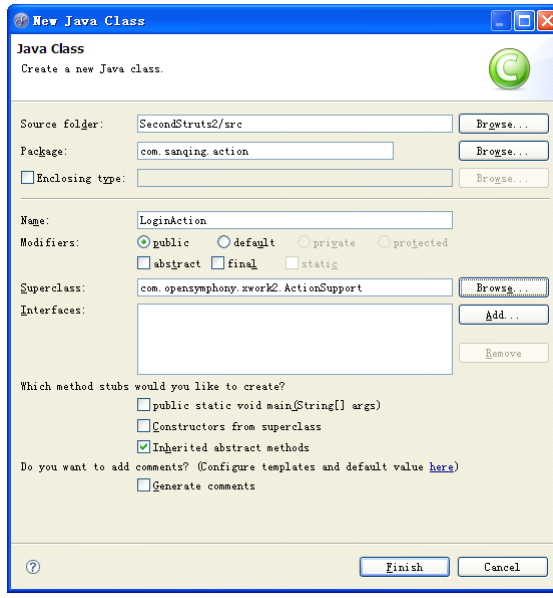


图 10-28　创建登录 Action

单击“Finish”按钮，将完成登录 Action 的创建。在编辑区打开程序后，加入必要的功能代码，具体代码如下所示。

```
01  package com.sanqing.action;
02  import com.opensymphony.xwork2.ActionSupport;
03  public class LoginAction extends ActionSupport {
04      private String username;            //用户名
05      private String password;            //密码
06      public String getUsername() {
07          return username;
08      }
09      public void setUsername(String username) {
10          this.username = username;
11      }
12      public String getPassword() {
13          return password;
14      }
15      public void setPassword(String password) {
16          this.password = password;
17      }
18      public String execute() throws Exception {
19          if("Tom".equals(username)&&"456123".equals(password)){//判断是否登录
20              return "success";      //返回成功字符串
21          }else{
22              return "fail";         //返回失败字符串
23          }
24      }
25  }
```

上述代码中第 4 行和第 5 行分别定义了用户名和密码属性，并为它们定义了 Getter 和 Setter 方法，从而能够获取提交表单中的内容。在第 18 行中的处理方法中，判断用户名和密码是否为指定内容，从而返回不同的字符串。

10.4.3 对登录 Action 进行配置

使用插件开发 Action 程序时，并不需要进行手动配置，在其中会自动完成配置。当手动开发登录 Action 后，是要自己手动配置的。配置代码就是放在 Struts 2 配置文件中，具体配置代码如下所示。

```
01  <struts>
02      <package name="default" extends="struts-default">
03          <action name="login" class="com.sanqing.action.LoginAction">
04            <result name="success">/login_success.jsp</result>
05            <result name="fail">/login_fail.jsp</result>
06          </action>
07      </package>
08  </struts>
```

其中第 3 行对登录 Action 进行配置，class 属性指定 Action 类的完整类名，name 属性指定访问该 Action 使用的名称。第 4 行中配置了当返回结果为“success”时，跳转到“/login_success.jsp”程序，也就是登录成功页面。第 5 行是配置了登录失败页面，和配置成功页面相似的。

10.4.4 发布并运行登录项目

在前面学习通过插件方式开发 Struts 2 项目时，发布项目是和本书前面讲到的发布方式有很大不同的。但是如果通过手动开发 Struts 2 项目，那么和发布普通 Web 项目没有区别的。

在 Servers 界面中选中“Tomcat 6.x”服务器，单击鼠标右键，在弹出的菜单中，选择“Add Deployment”命令，将弹出指定服务器发布界面，在其中选择“SecondStruts2”项目进行发布。

然后选中“Tomcat 6.x”服务器，单击鼠标右键，在弹出的菜单中，选择“Run Server”命令，将开始启动服务器。启动服务器成功后，打开浏览器中，在其中输入如下地址：

http://localhost:8080/SecondStruts2/login.html

运行结果如图 10-29 所示。

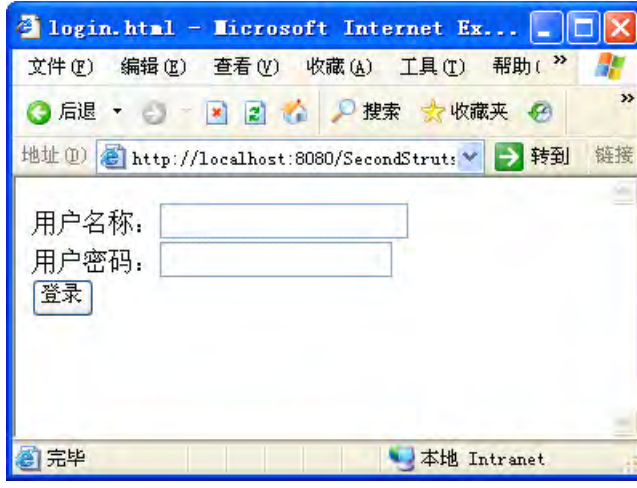


图 10-29　登录页面

在其中用户名称文本框中输入“Tom”，在用户密码框中输入“456123”，单击“登录”按钮，运行结果页面如图 10-30 所示。

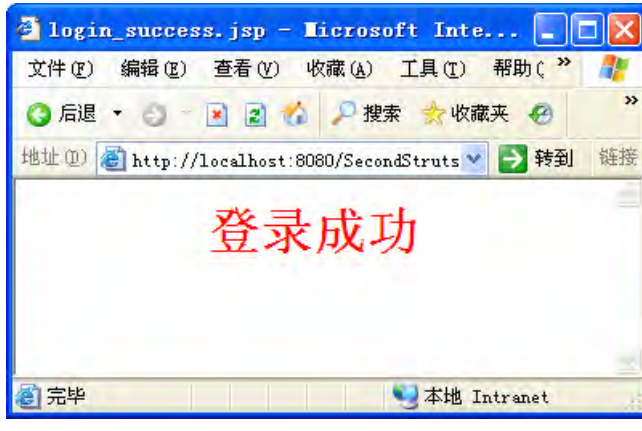


图 10-30　登录成功页面

如果输入的内容不是上述内容，将跳转到登录失败页面。从运行结果中可以看到登录项目开发成功。

10.4.5　Struts 2 程序执行流程

开发完 Struts 2 的项目后，我们简单的来看一下 Struts 程序的执行流程。在表单中进行提交操作后，将经过 Web 描述符中配置的核心控制器进行过滤操作，在其中分析提交地址，截取后的字符为“login”。

然后进入 Struts 2 的配置文件中查找以“login”为名称的 Action 配置，从而找到 Action 程序类执行。执行返回结果后，再访问 Struts2 配置文件，查找通过“<request>”标记配置的该字符串指定的返回结果，从而得到相应的页面。

第11章　SSH2框架整合开发

Struts 2 框架在实际开发中主要做控制层，所以它也可以和 Spring、Hibernate 进行整合开发，通常将它们整合后的框架称为 SSH2 框架。Struts2 框架只和 Hibernate 框架整合是比较少见的，它并不符合当前 Web 项目的分层思想，所以这里就不对它进行详细讲解。在本章中将先来讲解 Struts2 和 Spring 整合，再来讲解如何将三种框架整合在一起开发。

11.1　Struts 2+Spring 整合开发

在 Struts 和 Spring 整合一节中，可以发现修改的地方是非常多的。Struts2 和 Spring 整合是比较方便的，在 Struts 2 框架中专门开发了用于 Spring 整合的插件 JAR 包，在本节中将主要讲解它们之间的整合操作。

11.1.1　创建整合项目

因为在 MyEclipse 中并没有集成开发 Struts 2 项目的功能，所以要首先手动创建一个 Struts 2 项目。在普通的 Web 项目中执行加入 Struts 2 类库和加入、修改配置文件等操作，这些操作可以参考 Struts 2 一章进行学习。

在 Struts 2 项目的基础上，加入 Spring 框架的支持。在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Spring Capabilities”命令，将弹出加入 Spring 类库支持操作界面，在其中选择 Spring 核心类库和 Web 类库，如图 11-1 所示。

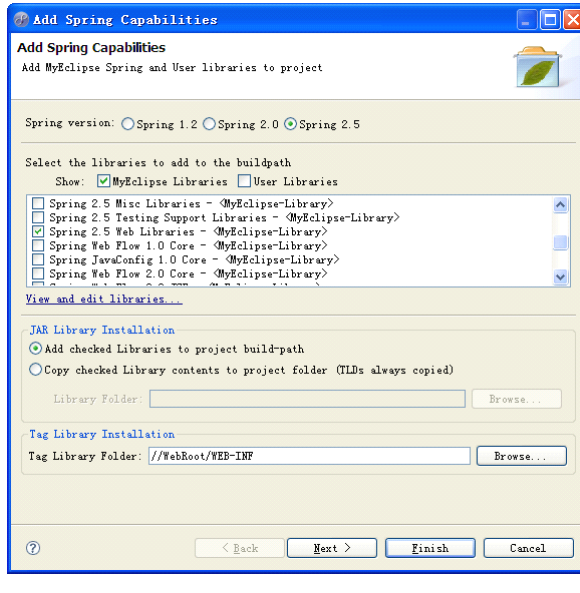


图 11-1　加入 Spring 类库

单击“Next”按钮，将进入对 Spring 配置文件进行设置的界面，如图 11-2 所示。

说明：在和 Struts 2 整合的项目中，当启动项目时将会首先查找 WEB-INF 目录下的 Spring 配置文件，如果该根目录下没有，则会自动查找子目录。所以通常直接放在 WEB-INF 目录下，放在 sec 目录下也是可以的，它将自动放置在 WEB-INF|classes 目录下。

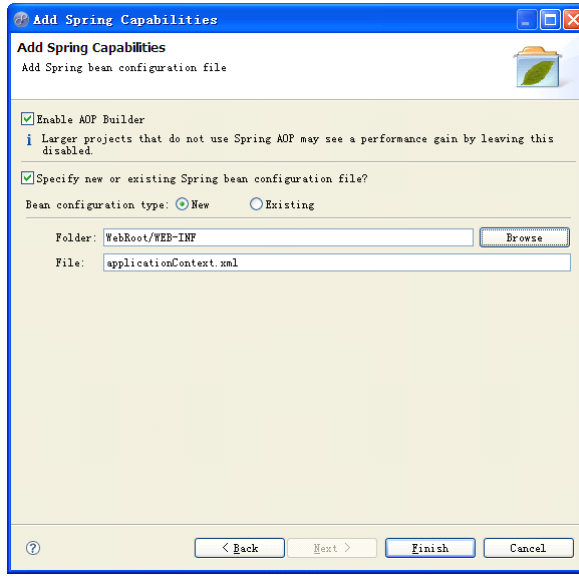


图 11-2　设置 Spring 配置文件

单击“Finish”按钮，完成 Spring 框架的技术支持。

向普通的 Web 项目中 Struts 2 框架和 Spring 框架支持后，是还不能够进行整合开发的，还要进行必要的配置。首先就是要安装 Struts 2 框架中的 Spring 插件，该 Spring 插件是一个 JAR 包，名称为“struts2-spring-plugin-2.1.8.1.jar”，在 Struts 2 框架的下载包中可以找到它。只需要将它复制到“WebRoot”|“WEB-INF”|“lib”目录下就可以。

为了让 Web 项目启动时，就能够查找到 Spring 的配置文件，还需要在 Web 描述符中添加一个 Spring 框架中的监听器类，配置代码如下所示。

```
01    <listener>
02        <listener-class>org.springframework.web.context.ContextLoaderListener
03        </listener-class>
04    </listener>
```

通过添加该监听器，使得 Web 应用启动时会自动查找 WEB-INF 目录下的 applicationContext.xml 配置文件，并根据该配置文件来创建 Spring 容器。

11.1.2　开发业务逻辑层并配置

在 Struts 2 一章中是以登录程序来讲解了如何开发 Struts 2 程序。这里我们通过通过

登录来讲解 Struts 2 和 Spring 整合开发。首先是开发业务逻辑层，在其中定义了处理登录的业务方法，它的代码为：

```
01  package com.sanqing.service;
02  public class UserServiceImpl {
03      public boolean login(String username,String password){
04          if("Tom".equals(username)&&"456123".equals(password)){
05              return true;
06          }else{
07              return false;
08          }
09      }
10  }
```

在包资源管理中，选中“UserServiceImpl”类节点，单击鼠标右键，在弹出的菜单中选择“Refector”|“Extract Interface”命令，将弹出抽取接口的界面，填写接口名和抽取的方法后，界面如图 11-3 所示。

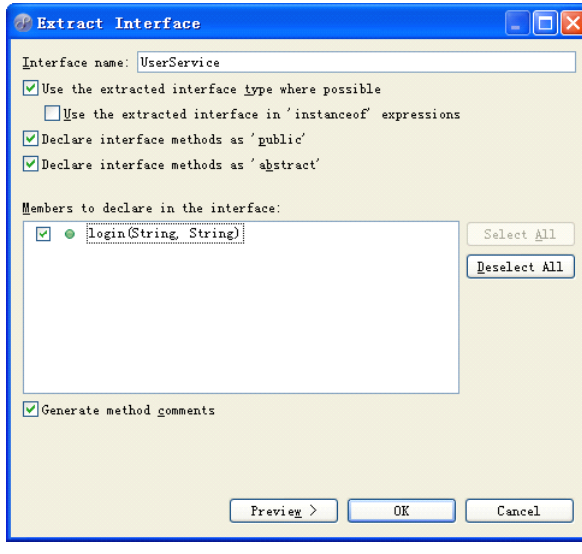


图 11-3　抽取接口

单击“OK”按钮，完成业务逻辑接口的抽取。接下来就需要在 Spring 的配置文件中，对业务逻辑层实现类进行配置，配置界面如图 11-4 所示。

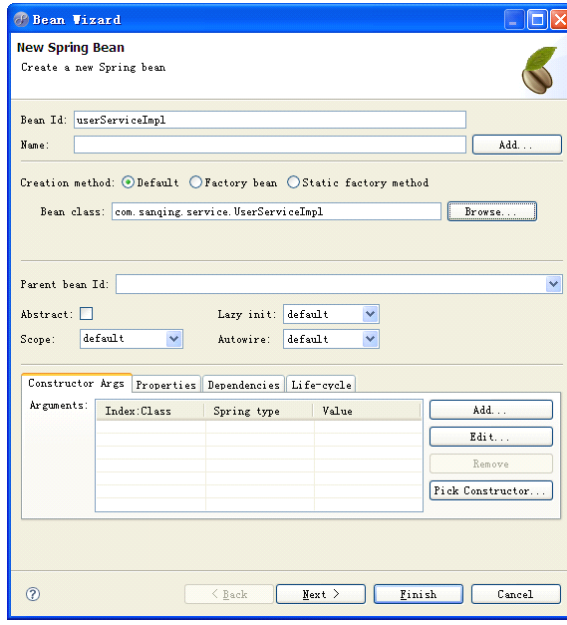


图 11-4　配置业务逻辑层

单击“Finish”按钮，将完成对业务逻辑层的配置，并在 Spring 的配置文件中自动生成配置代码。

11.1.3　开发控制层的 Action

这里我们采用手动创建的方式创建 Action 程序，因为该项目整合了 Spring 框架，所以在其中可以将业务逻辑实现类对象注入，从而需要在 Action 程序中定义相应的 Setter 方法。登录 Action 的具体代码如下所示。

```
01  package com.sanqing.action;
02  import com.opensymphony.xwork2.ActionSupport;
03  import com.sanqing.service.UserService;
04  public class LoginAction extends ActionSupport {
05      private String username;          //用户名
06      private String password;          //密码
07      private UserService userService;
08      public String getUsername() {
09          return username;
10      }
11      public void setUsername(String username) {
12          this.username = username;
13      }
14      public String getPassword() {
15          return password;
16      }
17      public void setPassword(String password) {
18          this.password = password;
```

```
19          }
20          public void setUserService(UserService userService) {
21                this.userService = userService;
22          }
23          public String execute() throws Exception {
24                if(userService.login(username, password)){   //判断是否登录
25                      return "success";//返回成功字符串
26                }else{
27                      return "fail";              //返回失败字符串
28                }
29          }
30    }
```

在其中第 7 行中定义了业务逻辑接口引用，并为它定义了 Setter 方法，通过这种方式将业务逻辑实现类对象注入。在第 24 行中调用业务逻辑实现类中的 login 登录方法，从而判断是否能够登录。

开发完 Action 程序后，需要进行两个配置，分别是 Struts 配置和 Spring 配置。首先进行 Spring 的配置，配置界面如图 11-5 所示。

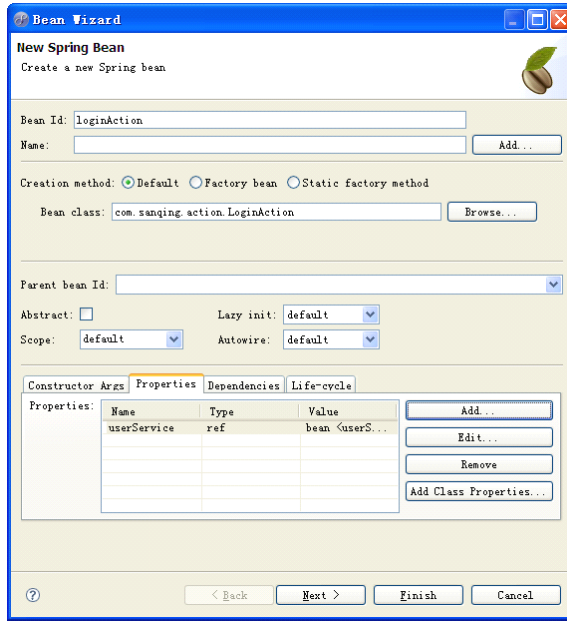


图 11-5　Spring 配置登录 Action

单击“Finish”按钮，从而完成对登录 Action 的 Spring 配置，也从而将业务逻辑实现类对象注入到 Action 中。

然后进行 Struts 2 配置，在 Struts 2 配置文件中的配置代码如下所示。

```
01    <struts>
02          <package name="default" extends="struts-default">
03                <action name="login" class="loginAction">
04                      <result name="success">/login_success.jsp</result>
```

```
05                <result name="fail">/login_fail.jsp</result>
06            </action>
07        </package>
08    </struts>
```

在该代码中主要修改了<action>节点中的 class 属性，因为这时业务控制器不再由 Struts 2 来创建，而是需要从 Spring 的 IoC 容器中取得。class 属性的属性值为进行 Spring 配置时指定的 id 值，也就是<bean>节点的 id 属性值。

技巧：对 Action 进行配置时，通常先进行 Spring 配置，再进行 Struts 2 配置。因为在 Struts 2 配置中需要使用到 Spring 配置中的 id。

11.1.4 运行整合项目

开发 Struts 2+Spring 的项目后，我们就可以来运行该项目，从而验证项目开发的是否正确。将“Struts2AndSpring”项目发布到 Tomcat 服务器中后，启动 Tomcat 服务器。打开浏览器，在其中输入如下地址：

http://localhost:8080/Struts2AndSpring/login.html

运行页面如图 11-6 所示。

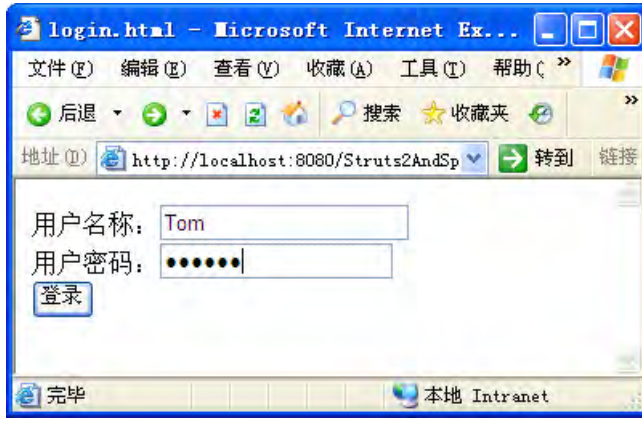


图 11-6 登录页面

这里我们输入一个错误的密码，单击“登录”按钮，运行结果如图 11-7 所示。

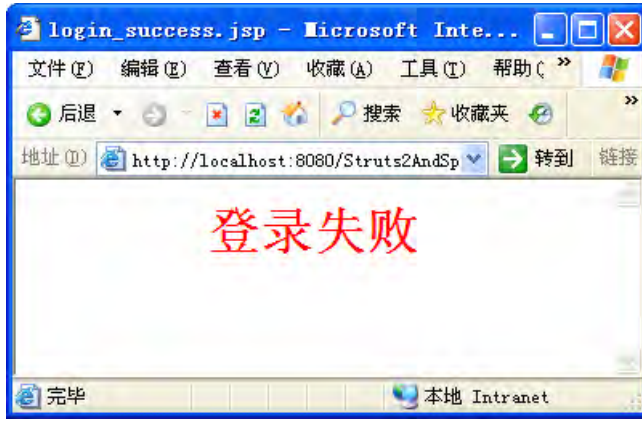


图 11-7　登录失败

从运行结果中可以看到，Struts 2 框架成功整合了 Spring 框架，项目开发正确。

11.2　SSH2 框架整合开发考试系统

SSH2 框架整合开发是目前最流行的开发技术，在前面的学习中已经学习了它们之间两两之间的整合。在本节中通过一个考试系统来学习一下如何将三种框架技术整合在一起进行开发。

11.2.1　考试系统功能分析

在一个完成的考试系统中，应该具有后台的录入试题、查看考生考试信息功能，在前台中，应该具有考生登录、获取考试试卷、提交考试试卷、获取考试成绩、显示试卷解析等功能。后台的功能实现起来是比较容易的，在本节中主要来完成系统前台考生操作的功能。

目前的考试形式有两种，一种是所有考生答同样的试卷，这种适合考试一起考试。另一种是随机从题库中获取试卷，计算机方面的考虑一般都采用这种方式，在本系统中也开发这样的功能。

11.2.2　创建数据库

在创建数据库和数据表之前，首先要分析系统中有哪些主体，由于这里我们只开始考试系统的前台，所以主体包括考生和科目考试试题，所以要为它们分别定义考生数据表和试题数据表。在创建这些数据表之前，我们为该项目单独创建一个数据库。

在 MyEclipse 的数据库视图中，打开“MySQL”连接，在“Connected to MySQL”节点上，单击鼠标右键，在菜单中选择“Create database”命令，在弹出的创建数据库界面中填写数据库的名称是“exam”，如图 11-8 所示。

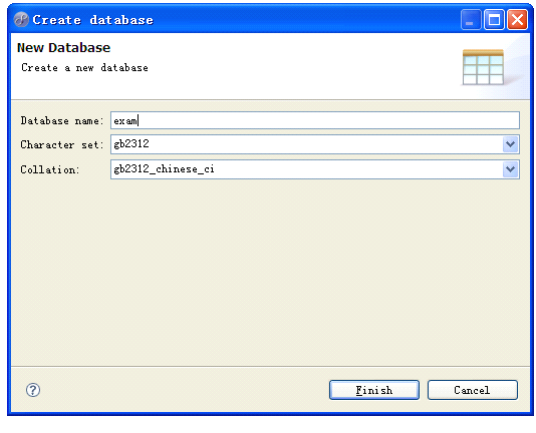


图 11-8　创建考试数据库

单击“Finish”按钮，将完成数据库的创建。选中“exam”数据库节点，单击鼠标

右键，选择“New Table”命令，将弹出创建数据表界面。通过该界面，依次创建考生和试题数据表。先来创建考生数据表，创建界面如图 11-9 所示。

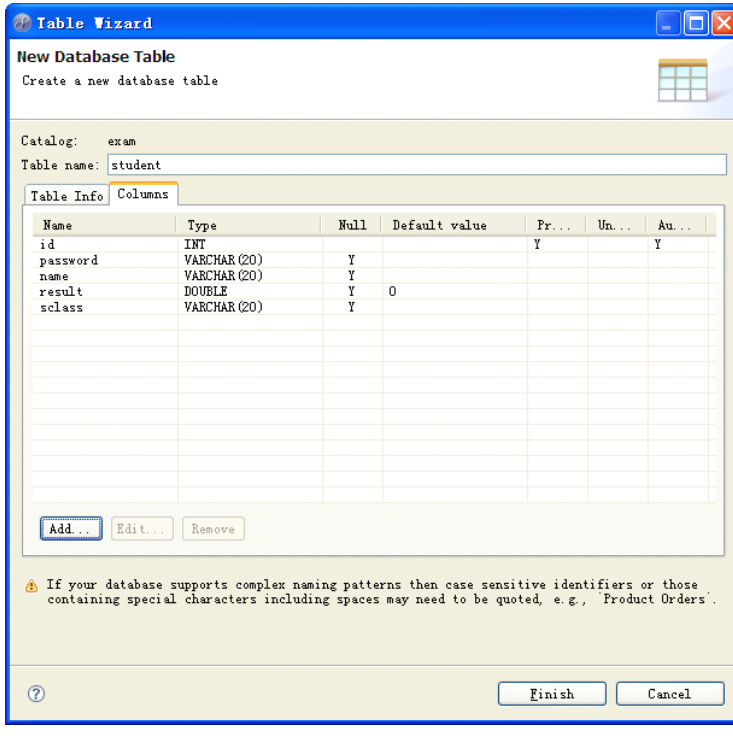


图 11-9　创建考生数据表

其中 id 是考生数据表的主键，它也是考生的学生证号。password 是考生进入考试系统的密码。name 是考生的真实姓名，因为姓名可以重复，所以不能让它做为主键。result 是考生的考试成绩，设置默认值是 0 分。sclass 是考生所在的班级，通过该字段就可以得到某一个班的所有学生成绩。

单击“Finish”按钮，将完成考生数据表的创建。再执行创建数据表操作，在界面中填写试题信息，界面如图 11-10 所示。

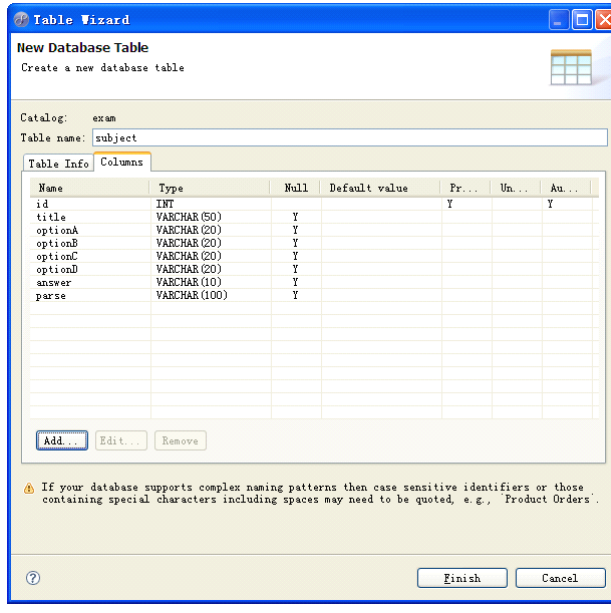


图 11-10　创建试题数据表

其中 id 是试题数据表的主键，title 是实体的题目。optionA、optionB、optionC 和 optionD 分别是试题的四个选项。answer 是试题的正确答案，目前系统中只支持选择题。parse 是试题的解析，考生考试完成后，可以查看。

单击“Finish”按钮，将完成试题数据表的创建，向其中插入一些测试数据，这里就不演示了，读者可以参考数据库管理一章进行学习。开发完成后不要忘记修改数据库连接指向 exam 数据库。

11.2.3　创建考试系统项目

首先创建一个普通的 Java Web 项目，项目的名称是“EXAM”，然后依次向其中加入 Struts 2 框架、Spring 框架和 Hibernate 框架的支持。

想项目中加入 Struts 2 框架支持时，首先要将必要的开发包复制到“WebRoot”|“WEB-INF”|“lib”目录下，其中包括 6 个基础包和 Spring 插件包。然后在“web.xml”文件中加入对 Struts 2 核心控制器配置的代码，代码如下：

```
01  <filter>
02      <filter-name>struts2</filter-name>
03      <filter-class>
04      org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
05  </filter>
06  <filter-mapping>
07      <filter-name>struts2</filter-name>
08      <url-pattern>/*</url-pattern>
09  </filter-mapping>
```

接下来在 src 目录下创建 Struts 2 的配置文件，文件名称为“struts.xml”，基础配置代码为：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="default" extends="struts-default">

    </package>
</struts>
```

加入 Struts 2 框架支持后，依次再加入 Spring 的框架支持。选中“EXAM”项目后，在 MyEclipse 的菜单中，选择“MyEclipse”|“Project Capabilities”|“Add Spring Capabilities”命令，将弹出加入 Spring 类库支持操作界面，在其中选择 Spring 核心类库和 Web 类库，如图 11-11 所示。

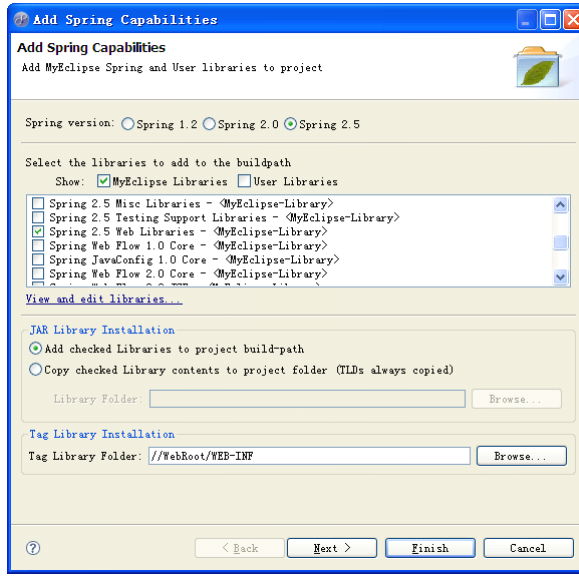


图 11-11　加入 Spring 类库

单击“Next”按钮，将进入对 Spring 配置文件进行设置的界面，在其中将 Spring 的配置文件放置在“WebRoot”|“WEB-INF”目录下，如图 11-12 所示。

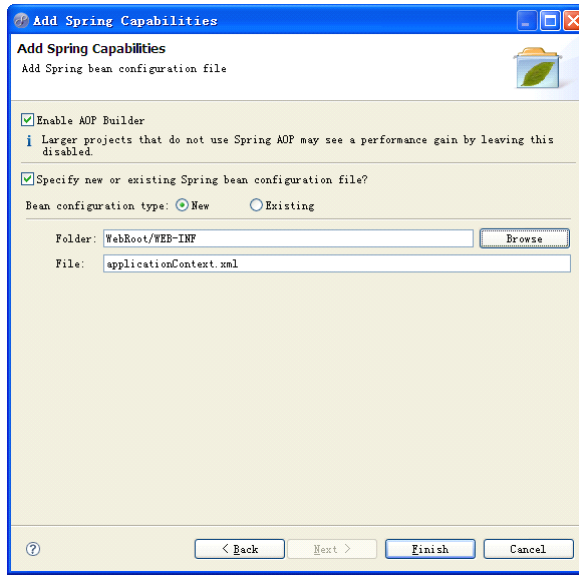


图 11-12　设置 Spring 配置文件

单击“Finish”按钮，将完成加入 Spring 框架的支持操作。最后是向项目中加入 Hibernate 框架支持。在加入 Hibernate 类库操作界面中，不要忘记 Spring 整合 Hibernate 时所用到的类库，如图 11-13 所示。

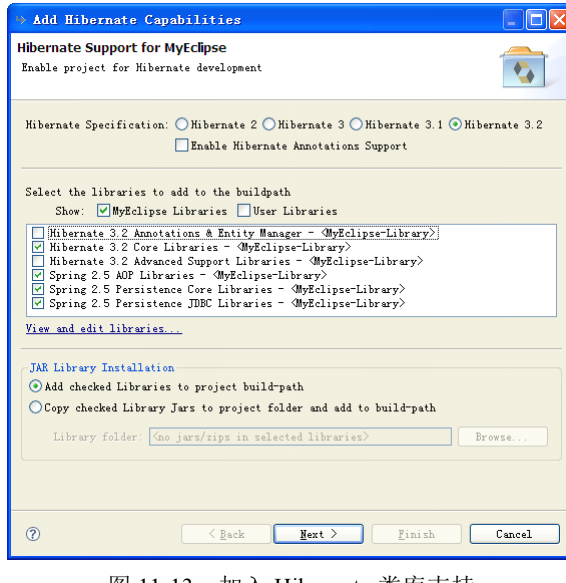


图 11-13　加入 Hibernate 类库支持

单击“Next”按钮，将进入选择 Hiibernate 配置信息支持的界面，在其中选择第 2 个选项，让 Spring 的配置文件对数据库的相关信息管理。单击“Next”按钮，将进入配置会话工厂类的界面，选择使用已有 Spring 配置文件，其他采用默认值。单击“Next”按钮，将进入配置数据库连接的界面，在其中选择“MySQL”，如图 11-14 所示。

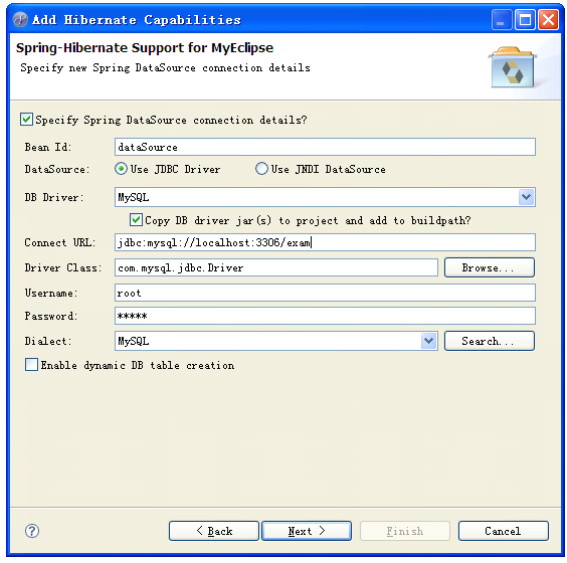


图 11-14　建立数据库连接

单击“Next”按钮，将进入创建会话工厂类的界面，将它定义在“com.sanqing.hb”包中，单击“Finish”按钮，将完成加入 Hibernate 框架支持的操作。

向项目中加入三种框架支持后，要想进行项目程序开发，还要进行必要的配置。首先是 Struts 2 和 Spring 整合时需要配置的 Spring 监听器，在“web.xml”文件中添加一个 Spring 框架中的监听器类，配置代码如下所示。

```
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

通过添加该监听器，可以在 Web 项目启动时会查找 WEB-INF 目录下的 applicationContext.xml 配置文件，通过该配置文件来创建 Spring 容器。

Spring 和 Hibernate 整合时，通常将事务交给 Spring 进行管理，从而要在 Spring 的配置文件中对事务进行配置，配置代码为：

```
<bean id="transactionManager"
        class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory">
        <ref bean="sessionFactory"/>
    </property>
</bean>
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*" propagation="REQUIRED"/>
    </tx:attributes>
</tx:advice>
<aop:config>
    <aop:pointcut id="allDaoMethod"
                  expression="execution(* com.sanqing.dao.*.*(..))"/>
```

```
15          <aop:advisor pointcut-ref="allDaoMethod" advice-ref="txAdvice"/>
16      </aop:config>
```

到此为止，我们的 SSH2 框架整合开发的环境就已经创建完成。

技巧：如果经常使用同一种框架整合技术进行开发，可以创建一个空的模板项目，在该模板项目加入必要的框架支持和进行必要配置。当创建某一项目时，直接复制模板项目，然后在其中开发项目程序。通过这种方式，可以节省创建项目所花费的时间。

11.2.4 开发数据访问层

对于 Hibernate 技术开发的数据访问层来说，是不受控制层使用什么框架影响的。在 MyEclipse 的数据访问层中，对数据表进行操作自动生成数据访问层程序。在考试系统中存在两个数据表，依次对它们进行生成数据访问层程序的操作，这里以试题数据表为例。

在 MyEclipse 的数据库视图中，选中 “MySQL”|“Connection to MySQL”|“exam”|“TABLE”|“subject”试题数据表节点，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，在其中选择创建实体类、映射文件和 DAO 程序，如图 11-15 所示。

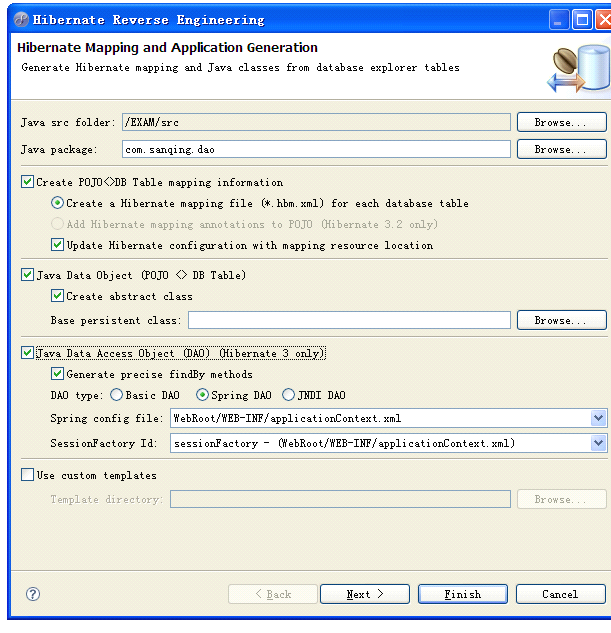


图 11-15　创建试题数据访问层

单击“Finish”按钮，将完成试题相关的实体类、映射文件和 DAO 实体类程序的创建。在自动生成的 DAO 程序中，虽然自动生成很多操作数据库的方法，但是并不能完成所有功能，例如从数据库中随机获取记录。这时候就需要手动的完成这类功能。随机获取试题记录的方法代码如下所示：

```
01          public List<Subject> randomFindSubject(int number) {
02              Query query = getSession().createQuery("from Subject as sub order by rand()");
```

```
03                query.setMaxResults(number);//设置查询记录数
04                List list = query.list(); //查询结果保存到 list 中
05                return list;
06            }
```

其中使用到了 HQL 语句，在第 2 行中调用了 rand 函数，从而获取随机记录。

说明：在 HQL 语句中有很多非常有用的函数，出了本节中介绍的随机函数外，还有获取前几条记录，获取从多少条到多少条的记录等函数。在本书中是不可能全部讲到的，有兴趣的读者可以查看相关资料。

创建 DAO 程序后，要进行抽取接口的操作。在包资源管理器中，选中“SubjectDAO”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，在其中输入接口名为“ISubjectDAO”，然后将其中的所有方法选中，界面如图 11-16 所示。

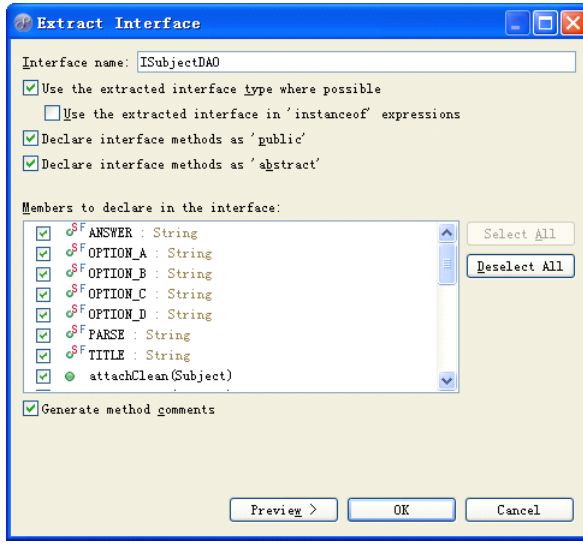


图 11-16　抽取试题 DAO 接口

单击“OK”按钮，将完成试题 DAO 接口的抽取操作。对试题数据表操作后，依次再对考生数据表进行操作，操作步骤是非常类似的，只是考生 DAO 程序中并不需要手动加入代码。

11.2.5　开发考生业务逻辑层

从前面的功能分析中可以知道，关于考生的操作功能包括登录考试系统、获取考生信息和设置考生成绩等功能，在本节中就来开发具有这些功能的考生业务逻辑实现类。

因为该系统使用了 Spring 框架技术，所以在业务逻辑实现类中调用数据访问层时，采用的是 Setter 方法注入的方式。考生业务逻辑层的具体代码如下所示：

```
01    package com.sanqing.service;
02    import com.sanqing.dao.IStudentDAO;
03    import com.sanqing.dao.Student;
```

```
04  public class StudentServiceImpl {
05      private IStudentDAO studentDAO;
06      public void setStudentDAO(IStudentDAO studentDAO) {
07          this.studentDAO = studentDAO;
08      }
09      public boolean login(Student student) {        //登录
10          if(studentDAO.findById(student.getId())==null){  //判断是否有该考生
11              return false;
12          }else{
13              Student findStudent=studentDAO.findById(student.getId());
14              if(student.getPassword().equals(findStudent.getPassword())){
15                  return true;
16              }else{
17                  return false;
18              }
19          }
20      }
21      public Student getStudent(int id) {            //获取考生信息
22          return studentDAO.findById(id);
23      }
24      public void setStudentResult(int id, double result) {
25          Student student = studentDAO.findById(id);//根据 ID 查找到该学生
26          student.setResult(result);                 //设置其成绩
27          studentDAO.attachDirty(student);           //更新学生信息
28      }
29  }
```

上述代码中第 5 行定义了考生数据访问层接口引用，并为它定义了 Setter 方法，通过这种方式将考生数据访问层实现类对象注入。在后面的代码中依次定义了登录方法、获取考生信息方法和设置考生成绩方法。

开发完考生业务逻辑实现类后，要进行抽取接口操作，抽取出一个考生业务逻辑接口。抽取操作界面如图 11-17 所示。

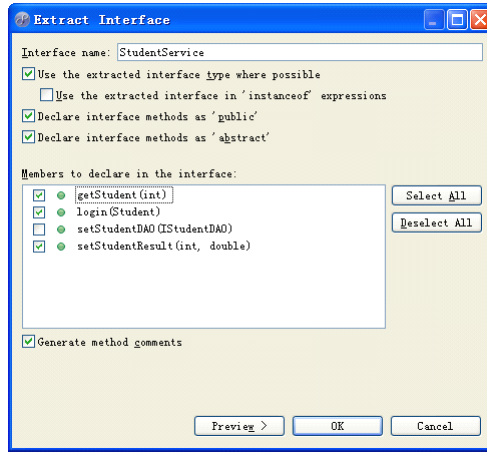


图 11-17　抽取考生业务逻辑接口

单击“OK”按钮，将完成考生业务逻辑接口的抽取。开发完考生业务逻辑实现类后，要在 Spring 配置文件中进行配置，配置界面如图 11-18 所示。

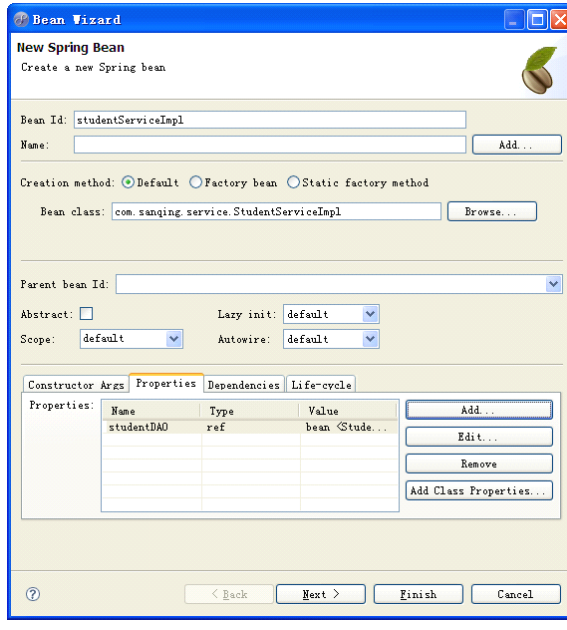


图 11-18　配置考生业务逻辑实现类

单击“Finish”按钮，将完成考生业务逻辑实现类的 Spring 配置，将考生数据访问层实现类对象注入到其中。

11.2.6　开发试题业务逻辑层

在试题业务逻辑实现类中，要完成对试题进行操作的功能方法，包括获取试题信息、随机获取一张试卷和计算考生考试成绩等功能。试题业务逻辑实现类的具体代码为：

```
01  package com.sanqing.service;
02  import java.util.List;
03  import com.sanqing.dao.ISubjectDAO;
04  import com.sanqing.dao.Subject;
05  public class SubjectServiceImpl {
06      private ISubjectDAO subjectDAO;
07      public void setSubjectDAO(ISubjectDAO subjectDAO) {
08          this.subjectDAO = subjectDAO;
09      }
10      public Subject getSubject(int id){              //获取试题信息
11          return subjectDAO.findById(id);
12      }
13      public List<Subject> randomFindSubject(int number) {  //随机获取试卷
14          return subjectDAO.randomFindSubject(number);
```

```
15          }
16          public int accountResult(List<Integer> subjectIDs,
17                  List<String> studentAnswers) {          //计算考试成绩
18              int GeneralPoint = 0;//总分
19              for(int i = 0; i < subjectIDs.size(); i++) {
20                  String rightAnswer = subjectDAO.findById(subjectIDs.get(i)).
21                                          getAnswer();     //得到正确答案，通过试题 ID
22                  if(rightAnswer.equals(studentAnswers.get(i))) {
23                      GeneralPoint += 20;   //加 20 分
24                  }
25              }
26              return GeneralPoint;
27          }
28      }
```

上述代码中第 6 行定义了试题数据访问接口引用，并为它定义了 Setter 方法，通过这种方式将试题数据访问实现类对象注入。在后面的代码中，依次定义了获取试题信息、随机获取试卷和计算考试成绩的方法。

技巧：我们的系统中是固定每一道题分值是相同的，如果不同，可以在试题数据表中定义分值字段。在计算结果业务方法中还要求每一道题的分值，从而完成累加。

开发完试题业务逻辑实现类后，要进行抽取接口操作，抽取出一个试题业务逻辑接口。抽取操作界面如图 11-19 所示。

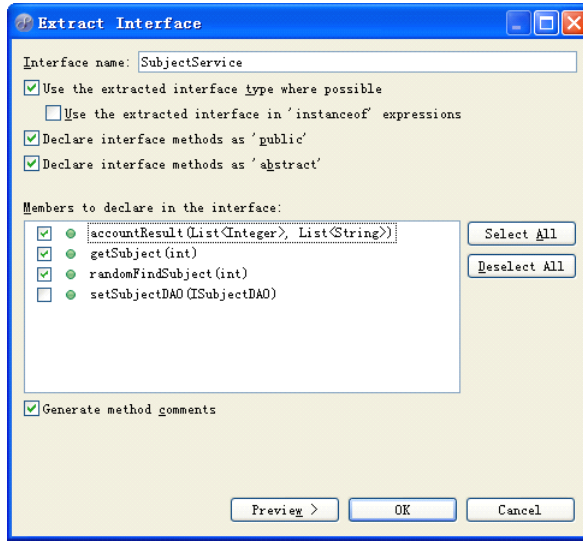


图 11-19　抽取试题业务逻辑接口

单击“OK”按钮，完成试题业务逻辑接口的抽取。最后同样还是在 Spring 配置文件中对试题业务实现类进行配置，配置界面如图 11-20 所示。

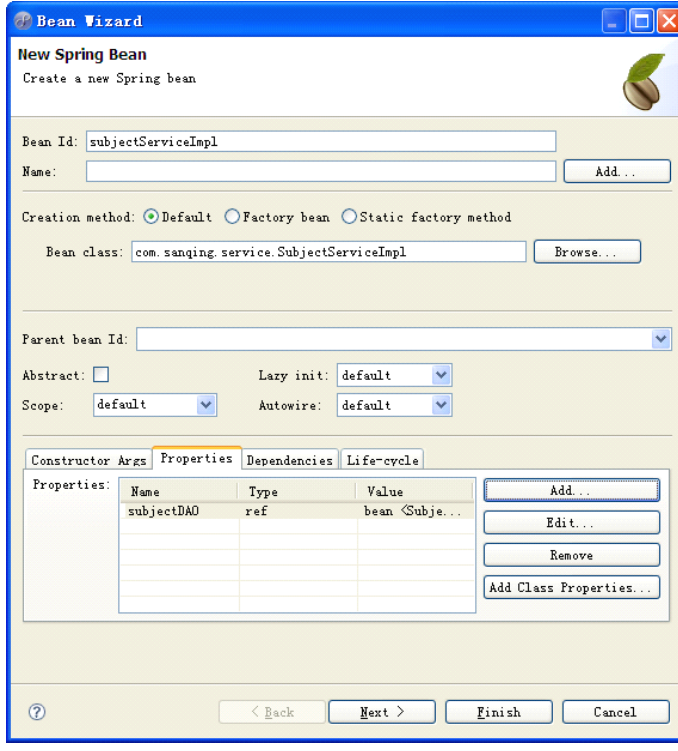


图 11-20　配置试题业务逻辑实现类

单击“Finish”按钮，将完成对试题业务逻辑实现类的配置，并将试题数据访问实现类对象注入。

11.2.7　开发登录系统 Action

考生进入考试系统以后，首先看到的就是登录页面，在其中通过输入自己的考试号和密码就可以进入系统。登录页面程序就不再给出了，读者可以查看光盘，这里主要来讲处理登录的 Action，它在系统中主要作为控制层。

登录 Action 的主要功能就是接收登录页面中的考试号和密码信息，然后调用业务逻辑层中的登录业务方法，最后根据返回结果跳转到不同页面。下面来看登录 Action 的具体代码。

```
01  package com.sanqing.action;
02  import com.opensymphony.xwork2.ActionSupport;
03  import com.sanqing.dao.Student;
04  import com.sanqing.service.StudentService;
05  public class LoginAction extends ActionSupport {
06      private int id;
07      private String password;
08      private StudentService studentService;
09      public int getId() {
10          return id;
11      }
12      public void setId(int id) {
```

```
13              this.id = id;
14          }
15          public String getPassword() {
16              return password;
17          }
18          public void setPassword(String password) {
19              this.password = password;
20          }
21          public void setStudentService(StudentService studentService) {
22              this.studentService = studentService;
23          }
24          public String execute() throws Exception {
25              Student student=new Student();          //创建考生对象
26              student.setId(id);                      //设置考生编号
27              student.setPassword(password);          //设置考生密码
28              if(studentService.login(student)){      //判断是否正确
29                  Map session = ActionContext.getContext().getSession();
30                  session.put("student", student);
31                  return "success";
32              }else{
33                  return "fail";
34              }
35          }
36      }
```

上述代码中第 6 行和第 7 行分别定义了考试的考号和密码属性，并为它们定义了 Setter 方法，从而获取登录页面中的信息。在第 8 行定义了考生业务逻辑接口引用，为为它定义了 Setter 方法，通过这种方式将考生业务逻辑实现类对象注入。

在处理方法中，第 25 行创建考生对象，并依次将获取的考号和密码信息设置到对象中。第 28 行中调用考生业务实现类中的登录业务方法，从而判断是否能够登录。

开发完 Action 程序后，要在 Spring 和 Struts 2 配置文件中进行配置，通常是先在 Spring 配置文件中进行配置。它的配置界面如图 11-21 所示。

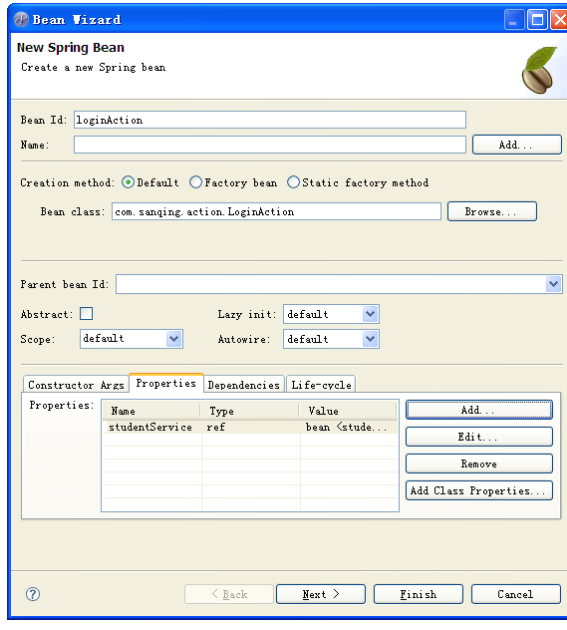


图 11-21　Spring 配置登录 Action

单击“Finish”按钮，将完成登录 Action 的配置，并将考生业务逻辑实现类对象注入进去。完成 Spring 配置后，要在 Struts 2 配置文件中进行配置，它的配置代码为：

```
01    <action name="login" class="loginAction">
02        <result name="success" type="chain" >getRandomSubject</result>
03        <result name="fail">/login.jsp</result>
04    </action>
```

其中第 1 行的 class 属性指定 Spring 配置文件中的 id 属性。第 2 行配置了当考生登录成功后，将执行获取随机试题 Action，该 Action 将在下一小节中开发。

11.2.8　开发获取随机试题 Action

考生登录成功后，进入考试系统后，将自动获取一张试卷，该试卷中的试题是从题库中随机获取的。在本小节中就来开发完成该功能的 Action 程序。

在获取随机试题 Action 中，并不会接收任何参数，在其中将调用试题业务逻辑实现类中的随机获取试题业务方法，它的具体代码如下所示。

```
01    package com.sanqing.action;
02    import java.util.List;
03    import javax.servlet.http.HttpServletRequest;
04    import org.apache.struts2.ServletActionContext;
05    import com.opensymphony.xwork2.ActionSupport;
06    import com.sanqing.dao.Subject;
07    import com.sanqing.service.SubjectService;
08    public class GetRandomSubjectAction extends ActionSupport {
09        private SubjectService subjectService;
10        public void setSubjectService(SubjectService subjectService) {
```

```
11              this.subjectService = subjectService;
12          }
13          public String execute() throws Exception {
14              List<Subject> subjectList=subjectService.randomFindSubject(5);    //随机 5 道
15              HttpServletRequest request = ServletActionContext.getRequest();
16              request.setAttribute("subjectList", subjectList);      //保存到 request 中
17              return "success";
18          }
19      }
```

上述代码中第 9 行定义了试题业务逻辑接口引用，并为它定义了 Setter 方法，通过这种方式将试题业务逻辑实现类对象注入。

在操作方法中，第 14 行调用试题业务逻辑实现类中的 randomFindSubject 随机获取试题方法，从而获取由试题对象组成的集合。并在第 16 行中将试题对象集合保存到 request 中。

说明：从获取随机试题 Action 中也可以更好的体会到 Action 的作用，Action 就是接收参数，调用业务方法，然后根据结果跳转。具体功能的实现由业务逻辑层来完成，例如获取随机试题。

开发完获取随机试题 Action 后，首先在 Spring 配置文件中进行配置，配置界面如图 11-22 所示。

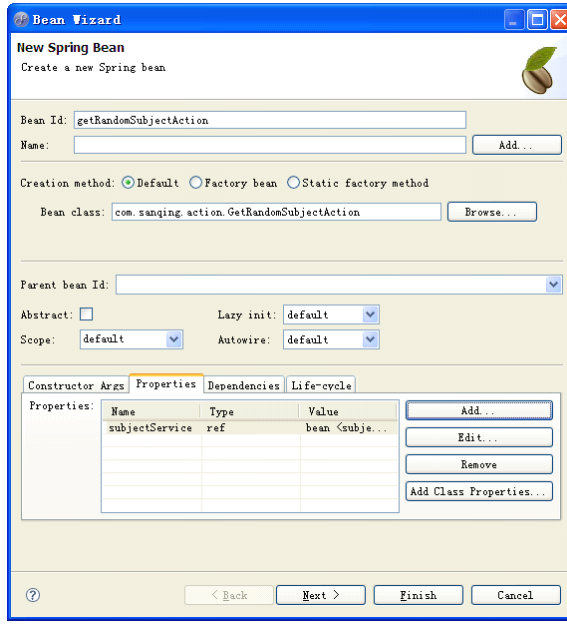


图 11-22　配置获取随机试题 Ation

单击“Finish”按钮，将完成在 Spring 配置文件中对获取随机试题 Action 的配置，从而将试题业务逻辑实现类对象注入。接下来就是在 Struts 2 配置文件中进行配置，配置代码如下所示。

```
01      <action name="getRandomSubject" class="getRandomSubjectAction">
```

```
02          <result name="success">/showSubject.jsp</result>
03      </action>
```

当执行完获取随机试题 Action 后，将跳转到显示所有试题的页面，在其中对保存在 request 中的试题对象集合遍历，从而得到一张试卷。显示获取试题的页面程序读者可以自己查看光盘代码。

技巧：在显示所有试题的页面程序中，使用到了隐藏域来获取试题 id 值，从而在计算考试成绩时可以获取这些值，从而能够调用后台获取它的正确答案。

11.2.9 开发获取考试成绩 Action

在显示试题的试卷中，单击“提交答卷”按钮，将执行获取考试成绩 Action。在获取考试成绩 Action 中，不但获取考生提交的答案，还会获取所有试题的 id 属性值。在 Action 的操作功能中计算考生的成绩，并将所得成绩保存到考生数据表中。获取考试成绩 Acton 的具体代码为：

```
01  package com.sanqing.action;
02  //省略导入接口和类的代码
03  public class SubmitExamAction extends ActionSupport{
04      private List<Integer> subjectID;  //考试考试试题 id
05      private SubjectService subjectService;
06      private StudentService studentService;
07      public List<Integer> getSubjectID() {
08          return subjectID;
09      }
10      public void setSubjectID(List<Integer> subjectID) {
11          this.subjectID = subjectID;
12      }
13      public void setSubjectService(SubjectService subjectService) {
14          this.subjectService = subjectService;
15      }
16      public void setStudentService(StudentService studentService) {
17          this.studentService = studentService;
18      }
19      public String execute() throws Exception {
20          HttpServletRequest request = ServletActionContext.getRequest();
21          List<String> studentAnswers = new ArrayList<String>();
22          for(int i = 0; i < 5; i++) {
23              String answer = request.getParameter("subjectAnswer"+i);
24              studentAnswers.add(answer);      //将每一题的答案保存到答案集合中
25          }
26          int GeneralPoint = subjectService.accountResult(subjectID, studentAnswers);
27          Map session = ActionContext.getContext().getSession();
28          Student student = (Student)session.get("student");
29          int studentID = student.getId();
30          Student student2=studentService.getStudent(studentID);
```

```
31                  studentService.setStudentResult(studentID, GeneralPoint);
32                  request.setAttribute("studentName", student2.getName());//保存学生姓名
33                  request.setAttribute("GeneralPoint", GeneralPoint);
34                  session.put("subjectIDs", subjectID);//将考试题目保存到 session
35                  return SUCCESS;
36              }
37      }
```

上述代码中第 4 行定义了试题 id 属性集合，并为它定义了 Setter 和 Getter 方法，从而获取考生试卷中所有的试题 id。在第 5 行和第 6 行分别定义了试题和考生的业务逻辑接口引用，并为它们定义了 Setter 方法，从而将相应的实现类对象注入。

在操作方法中，首先依次将用户提交的答案获取出来，并且将它们保存到集合中。在第 26 行中调用试题业务逻辑实现类中的 accountResult 获取考试成绩业务方法。

在第 31 行中调用考生业务逻辑实现类中的 setStudentResult 设置成绩业务方法，从而将考生的成绩保存到数据库中。在第 34 行中将所有的考试题目保存到 session 中，因为显示试卷正确答案和解析时将会使用到。

开发完获取考试成绩 Action 后，首先要在 Spring 配置文件中进行配置，配置界面如图 11-23 所示。

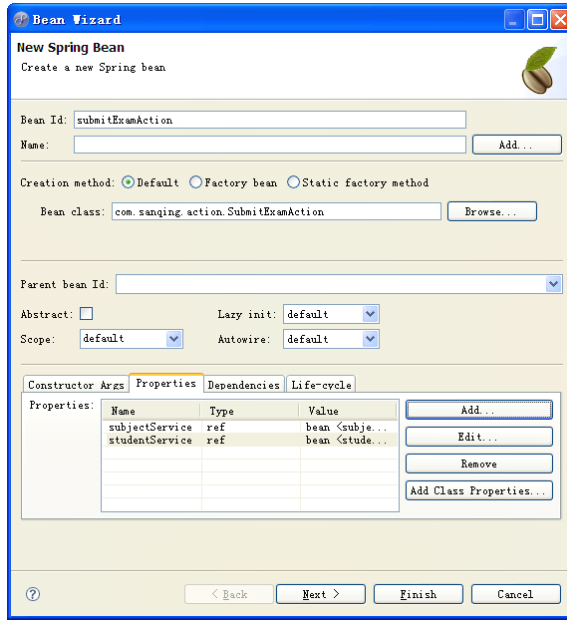


图 11-23　配置获取考试成绩 Action

单击“Finish”按钮，将完成获取考试成绩 Action 在 Spring 配置文件中的配置。接下来就是在 Struts 2 配置文件中进行配置，配置代码如下所示。

```
01    <action name="submitExam" class="submitExamAction">
02          <result name="success">/examResult.jsp</result>
03    </action>
```

当执行完获取考试成绩 Action 后，将跳转到第 2 行配置的显示考试结果的页面中，

在其中将把考生姓名和对应的成绩显示出来。在该页面中还会定义一个查看答案超链接，从而将考生所做试卷的正确答案显示出来。

11.2.10 开发显示试题答案 Action

在显示考试成绩页面中，单击“查看答案”超链接将执行显示试题答案 Action。在显示试题答案 Action 中并不会接收任何参数，只是获取保存在 session 中的试题 id 集合，从而获取对应的每一个试题对象。显示试题答案 Action 的具体代码如下所示。

```
01  package com.sanqing.action;
02  //省略导入接口和类的代码
03  public class ShowSubjectAnswer extends ActionSupport{
04      private SubjectService subjectService;
05      public void setSubjectService(SubjectService subjectService) {
06          this.subjectService = subjectService;
07      }
08      public String execute() throws Exception {
09          List<Subject> subjects = new ArrayList<Subject>();//保存学生考过的题目
10          HttpServletRequest request = ServletActionContext.getRequest();
11          Map session = ActionContext.getContext().getSession();
12          List<Integer> subjectIDs = (List<Integer>) session.get("subjectIDs");
13          for(Integer subjectID : subjectIDs) {
14              Subject subject = subjectService.getSubject(subjectID);//通过 id 查找
15              subjects.add(subject);
16          }
17          request.setAttribute("subjects", subjects);
18          return SUCCESS;
19      }
20  }
```

上述代码第 12 行中获取保存在 session 中的试题 id 组成的集成，第 13 行中对它进行遍历。在第 14 行中调用试题业务逻辑实现类中的 getSubject 获取试题方法，从而获取考生试卷中所有试题对象，并在第 15 行中将它们保存到集合中。

开发完显示试题答案 Action 后，首先在 Spring 配置文件中对该 Action 进行配置，配置界面如图 11-24 所示。

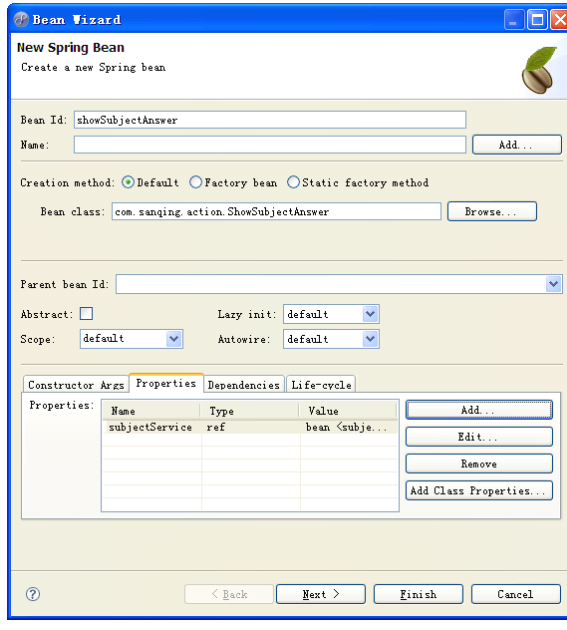


图 11-24　配置显示试题答案 Action

单击“Finish”按钮，将完成对显示试题答案 Action 的配置，在其中将试题业务逻辑实现类对象注入。接下来在 Struts 2 配置文件中对该 Action 进行配置，配置代码如下所示。

```
01　<action name="showSubjectAnswer" class="showSubjectAnswer">
02　　<result name="success">/showAnswer.jsp</result>
03　</action>
```

执行完显示试题答案 Action 后，将跳转到显示答案页面程序，从而将试题的答案和解析显示在页面中。

11.2.11　运行考试系统

将项目部署到 Tomcat 服务器中后，启动服务器。如果启动时发生错误，将项目中的“asm-x.jar”JAR 包删除，因为其中该包重复。启动成功后，打开浏览器，在其中输入如下地址：

http://localhost:8080/EXAM/login.jsp

运行结果如图 11-25 所示。

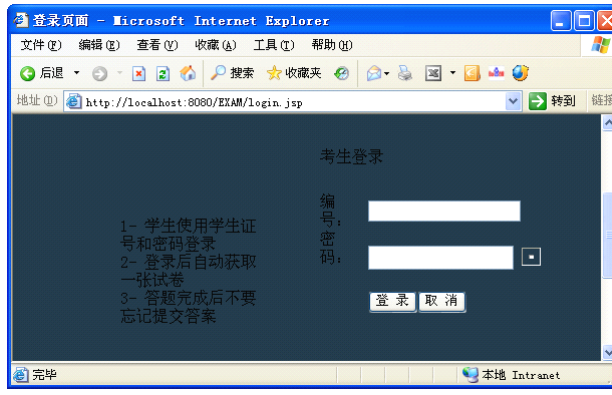


图 11-25　登录页面

注意：在执行登录操作之前，一定要向数据库中加入测试的考生信息。在实际开发中，考生的信息也是事前录入的。

在其中编号文本框中输入“20090001”，在密码框中输入“456123”，单击“登录”按钮，运行结果如图 11-26 所示。

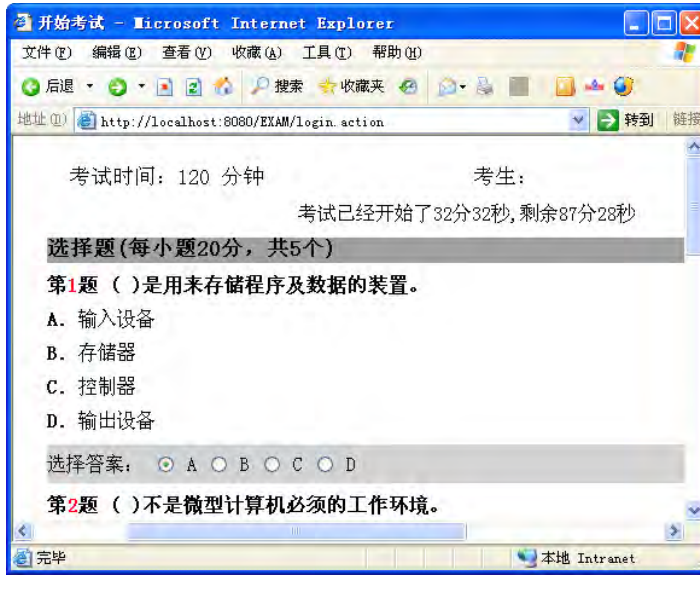


图 11-26　试卷页面

在试卷页面中将随机得到 5 道题，考生可以对这些试题进行作答。当考试时间结束，或者考试单击“提交答卷”按钮后将完成试题答案的提交，并进入到显示考试成绩页面，如图 11-27 所示。

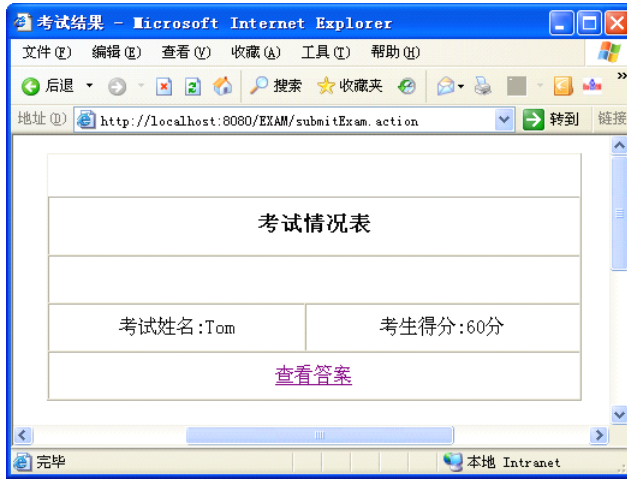


图 11-27　显示考试成绩

在显示考试成绩页面中会将学生姓名和它的考试成绩显示出来，单击其中的“查看答案”超链接，将进入显示答案页面，如图 11-28 所示。

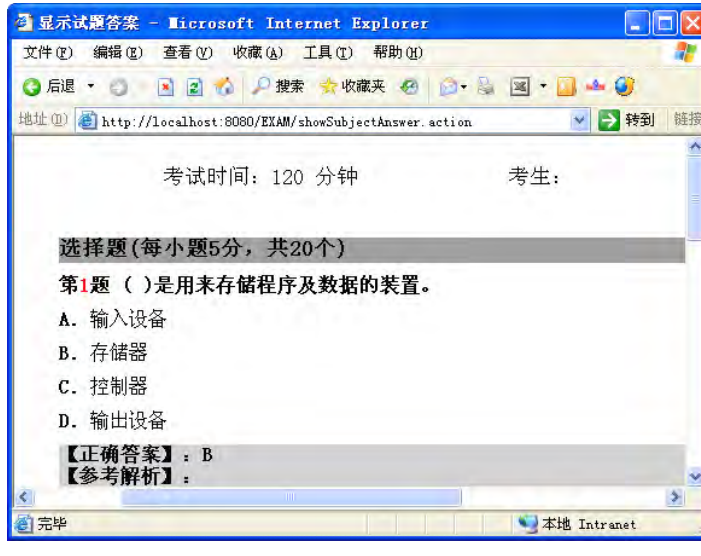


图 11-28　显示试题答案

该页面中的试题答案是和考生的考试试卷相对应的。在显示试题答案页面中，将把试题的答案显示出来，如果试题有参考解析，也会将解析内容显示出来。

第12章　调试和测试

调试和测试在项目开发中是必不可少的。通过测试可以检测程序中是否有错误，通过调试可以查找到具体错误，并进行排错操作。在 MyEclipse 中，集成了对程序进行调试和测试的功能。在 MyEclipse 中，说测试时一般是指单元测试，也就是对某一小段代码进行测试，而不是对整个项目进行测试。

12.1　调试

不管程序员的技术多么精通，工作经验多么丰富，都不能保证所写程序中不存在 BUG。在这种情况下，通过人为查错是非常复杂的，这时候就需要调试工具来对程序进行查错，最后人为的修改这些错误。在 MyEclipse 中，就集成了对程序进行调试的工具。

12.1.1　认识调试透视图

在 MyEclipse 的默认透视图下，是不能很好的看到调试效果的，所以需要切换到调试透视图下。在 MyEclipse 的菜单中，选择“Window”|“Open Perspective”|“Other”命令，将弹出选择切换透视图的界面，如图 12-1 所示。

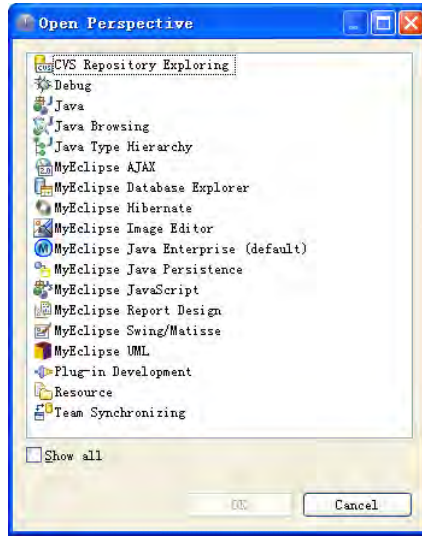


图 12-1　选择切换透视图

在其中选择“Debug”节点，单击“OK”按钮，MyEclipse 的视图将会发生变化，界面将变为如图 12-2 所示。

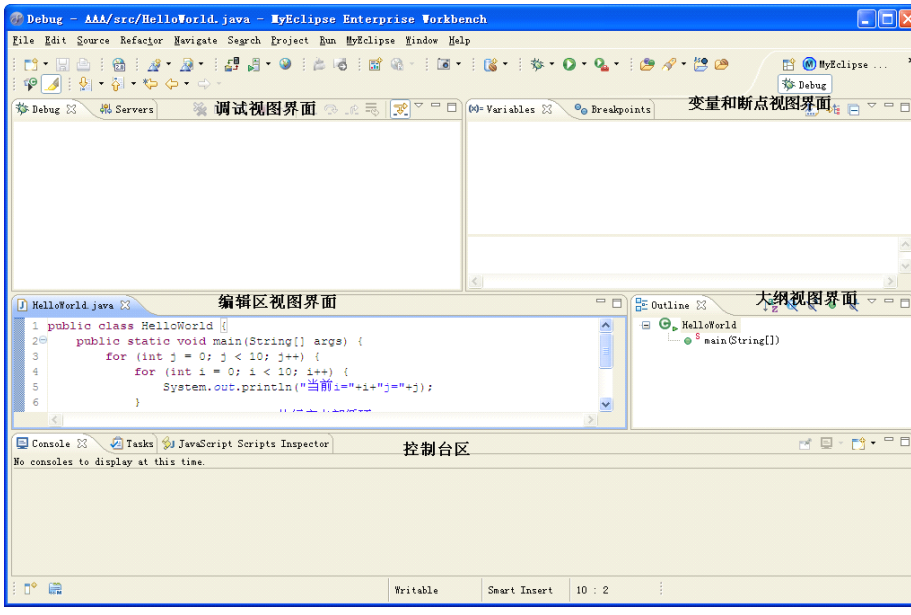


图 12-2　调试透视图界面

其中上面的四个视图界面是重要的。左上角是调试视图界面，在其中可以控制程序的运行，让程序按照程序员的意愿进行运行，从而不会一次运行结束。右上角是变量和断点视图界面，在变量视图中可以看到程序中变量值的变化；断点是调试中重要的部分，在后面将详细讲解。左下角是编辑区视图界面，在其中将显示要调试的代码内容，如果想创建和修改程序，最好回到默认透视图中。右下角是大纲视图界面，会将 Java 的文件结构显示在其中，它是比较少用的。

说明：这里只是简单的认识各个部分，在后面的讲解中还会讲解它们的具体使用和操作。在后面的讲解中将直接使用每一部分的名称。

12.1.2　启动调试配置

在正式开始调试操作之前，我们先来开发一个项目，并在项目中开发一个简单的 Java 程序，该程序的具体代码为：

```
01  public class DoubleFor {
02      public static void main(String args[]){
03          for(int i=0;i<5;i++){
04              for(int j=0;j<5;j++){
05                  System.out.println("当前 i="+i+"；j="+j);
06              }
07              System.out.println("内部循环执行结束");
08          }
09      }
10  }
```

该程序是非常简单的，例如定义了双重 for 循环语句，从而该程序来查看其中变量值的变化。开发完 Java 程序后，就可以对其进行调试。

说明：启动调试有多种方式，包括启动菜单、启动工具按钮和编辑区右键弹出菜单。笔者更倾向于使用右键弹出菜单来启动调试。

有时候在执行调试之前，需要进行必要的调试配置。在编辑区中，单数鼠标右键，在弹出的菜单中选择“Debug As”|“Debug Configurations”命令，将弹出进行调试配置的界面。在其中左面选择“Java Application”节点，然后选择要进行调试的程序节点，如图 12-3 所示。

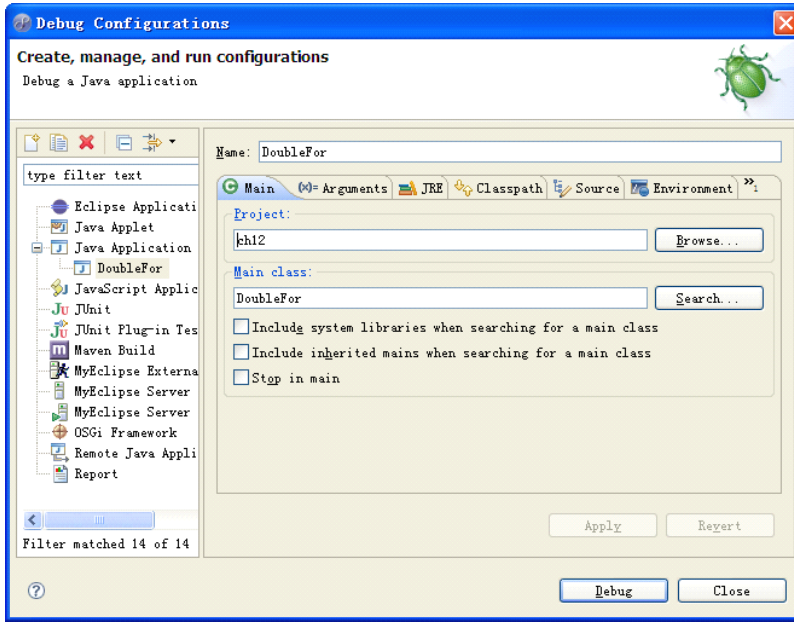


图 12-3　调试配置界面

其中“Name”表示进行调试的类的名称，下面有多个选项卡用来对该类进行必要的配置。“Main”选项卡是我们主要配置的，其中“Project”表示要调试程序所在的项目，“Main class”表示类名，单击“Search”按钮可以对类进行搜索。单击“Apply”按钮，将完成调试配置操作。单击“Debug”按钮，将开始对配置的类进行调试。

12.1.3　断点

断点是调试中的重要组成部分。如果不使用断点，进行调试时，程序在不发生异常的情况下会执行到最后，但是如果使用断点后，当程序运行到该断点，就会将当前执行线程挂起，从而能够人为的控制程序执行。

在程序中加入断点是非常简单的。在编辑区中，在行号的左边的标记栏中，双击鼠标就会为所在行的代码加入一个断点，如图 12-4 所示。

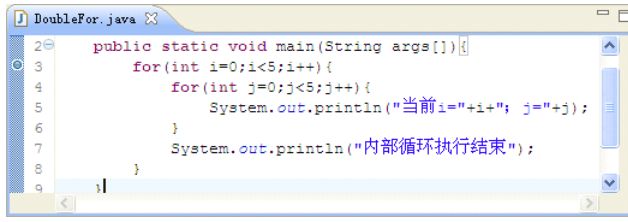


图 12-4　添加断点

添加断点后，在调试透视图的断点视图界面中也将显示该断点的信息，如图 12-5 所示。

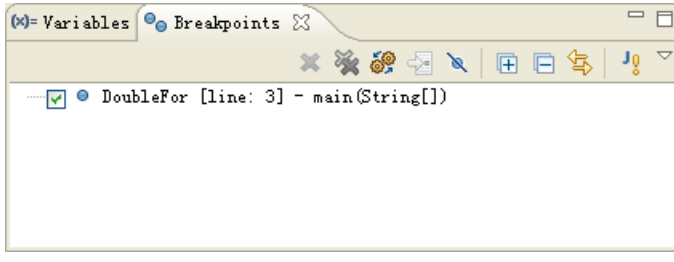


图 12-5　断点视图

在断电视图中，将把断点所在的类、行号和所在方法等信息显示出来。在一个程序中，使用同样的方法，可以同时添加多个断点，从而在断电视图中会将这些断点信息都显示出来。

当断点完成调试需求后，将需要将段断点删除。删除断点是非常容易的，在断点上双击鼠标，就可以将断点删除。

在断点视图界面中，还可以对断点进行操作。例如不想对断点进行频繁的添加和删除，就可以对断点进行启用和禁用。在断点视图界面中，选中断点信息，单击鼠标右键，在弹出菜单中选择“Disable”命令，将禁用该断点，是和删除断点一样的。禁用的断点将以空心的形式出现，如图 12-6 所示。

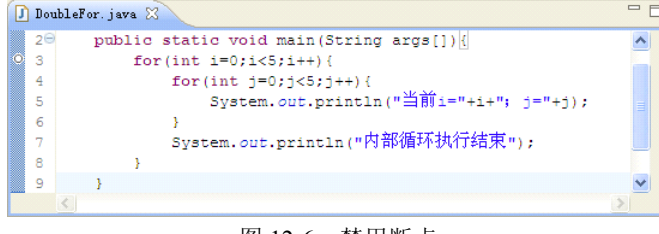


图 12-6　禁用断点

在断点视图中，选中禁用的断点信息，单击鼠标右键，在弹出的菜单中，选择“Enable”命令，将启用被禁用的断点。

12.1.4　开始调试

设置断点后，就可以进行调试。开启调试的方式有很多，除了能够在调试配置界面中单击“Debug”按钮外，还可以在编辑区中的右键弹出菜单中选择“Debug As”|“Java

Application”命令启动。

启动调试后，开始运行程序，当运行到断点处后，将会把当前线程挂起，调试视图界面如图 12-7 所示。

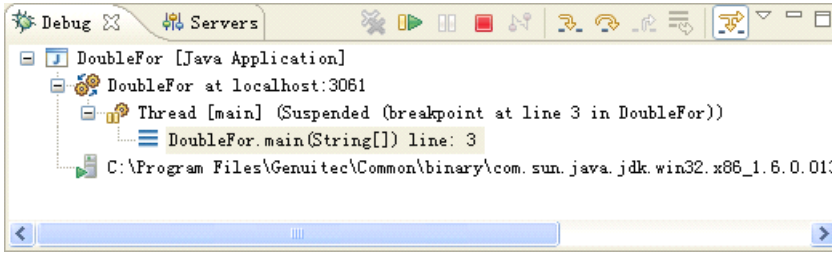


图 12-7　断点处挂起

其中三条蓝色线就表示在当前位置挂起，并给出位置是第 3 行。通过调试视图可以对程序进行多种调试操作，其中使用最多的就是单步跳过。其中“  ”就是单步跳过按钮，它的快捷键是“F6”。单击单步跳过按钮，在编辑区中将高亮显示当前运行的行，如图 12-8 所示。

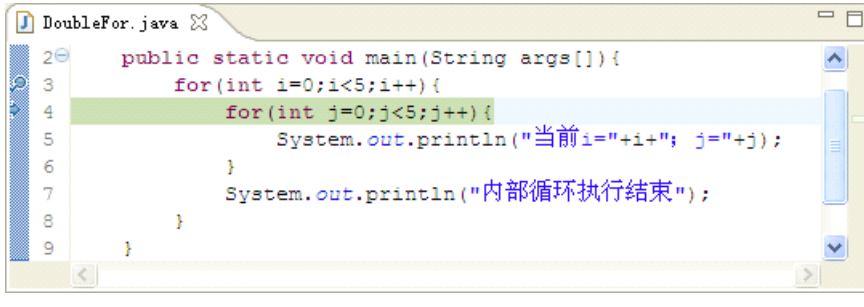


图 12-8　编辑区的变化

在变量视图界面中，还会将当前程序中的变量值显示出来，如图 12-9 所示。

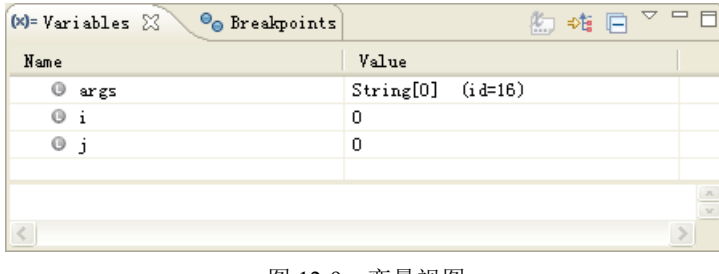


图 12-9　变量视图

多次单击单步跳过按钮，变量视图中的变量将随着变化，通过它就可以观察变化是否是我们想要的结果。如果发生异常，或者变量结果和期望结果不同，就需要及时排错，从而保证程序正常运行。

12.1.5 断点命中计数

在调试程序时，当运行到断点时，将自动会将程序挂起。但是有些时候，我们并不想每一次运行到断点时都将程序挂起，这时候就需要使用到断点命中计数。在对循环语句进行调试时，该操作是非常重要的。

例如我们将断点设置在“DoubleFor”程序的内存循环语句中，如图 12-10 所示。

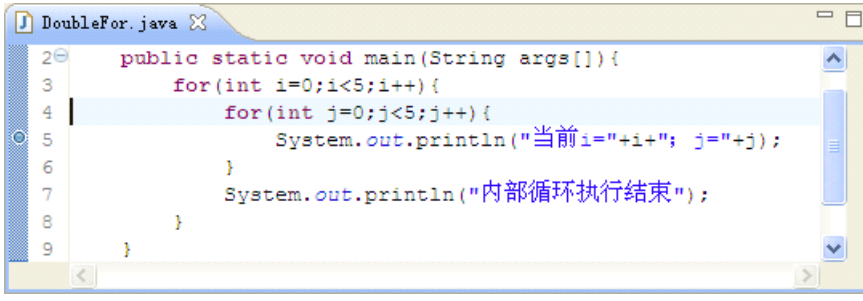


图 12-10　将断点设置在循环中

然后在断点视图中，选中当前断点信息，在右键菜单中选择“Hit Count”命令，将弹出设置命中计数的界面，如图 12-11 所示。

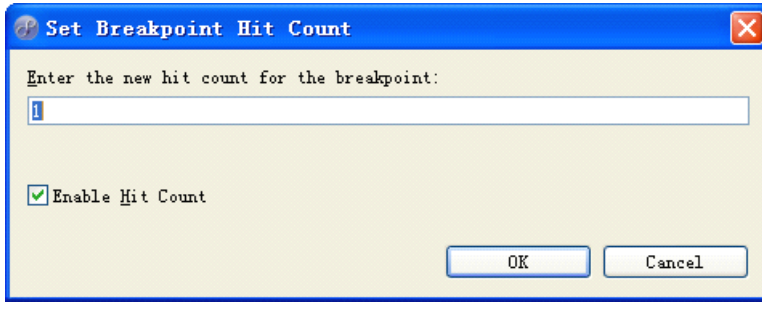


图 12-11　设置命中计数

在其中设置断点的命中计数为“3”，单击“OK”按钮，将完成设置。再次启动调试，观看控制台中的输出，输出结果为：

当前 i=0；j=0

当前 i=0；j=1

当第 3 次要运行断点所在的代码时，将把当前线程挂起，从而只输出了两条语句。

注意：在启动下一个调试之前，一定要将上一次的调试结束掉，在调试视图界面中单击“  ”停止按钮可以完成该操作。

12.1.6 断点条件

断点条件也是一种对断点的限制，是指当断点满足某一条件时，才会将当前线程挂起。设置断点条件也是在断点视图界面中完成，选中断点信息，在右键弹出菜单中，选择“Breakpoint Properties”命令，将它们对断点进行属性设置的界面，如图 12-12 所示。

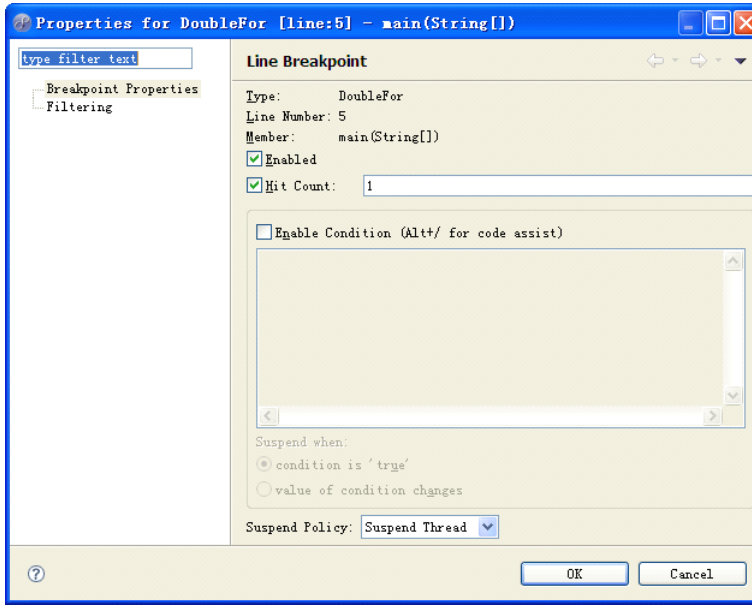


图 12-12　设置断点条件

从属性设置界面中可以看到，也可以在该界面中进行启用断点和设置断点命中计数设置。其中“Enable Condition (Alt+/ for code assist)”选项就是选择是否启用断点条件，这里选中它，然后在下面的文本域中输入断点条件，这里输入“j==2”。其他选项采用默认值，单击“OK”按钮，将完成断点条件的设置。

注意：将要其中的“Hit Count”选项修改为未选中状态，从而使断点只按照断点条件执行。

再次启动调试，可以看到控制台的运行结果为：

当前 i=0；j=0

当前 i=0；j=1

可以看到当 j 等于 2 的结果并没有输出，这是因为设置了断点的条件时“j==2”，这时候程序中的线程被挂起了。

12.2　测试

软件测试和软件开发一样，都是一种非常宽泛的概念。随着软件项目的规范越来越大，软件测试已经变的非常重要。测试分为多种，分别是综合测试、单元测试和用户测试等。在本书中主要讲解的是单元测试，它是和开发人员联系非常紧密的。通过单元测试可以确定某一段代码或者某一个方法是否正确。

JUnit 是一种开发开源的用于 Java 程序测试的框架，在实际开发中通常使用它来进行单元测试。在 MyEclipse 中，也集成了 JUnit 框架，在本节中主要来讲解 MyEclipse 中 JUnit 的使用。

12.2.1 创建测试程序

单元测试主要是对某一方法进行测试，所以这里我们首先来开发一个功能方法。被测试的程序代码如下所示。

```
01  package com.sanqing.junit;
02  public class HelloWorld {
03      public String connection(String s1,String s2){
04          return s1+s2;
05      }
06  }
```

上述代码第 3 行定义了连接字符串的方法，通过该方法可以让两个字符串连接在一起，从而组成一个新字符串，并返回。

接下来就是开发用于测试连接字符串方法的 JUnit 程序。

说明：JUnit 的测试程序类通常不和被测试程序放在同一个包下，而是在原包的基础上创建名称为“test”的子包，将所有的测试程序都放在“test”包下。

在包资源管理器中，选中“test”包，单击鼠标右键，在弹出菜单中选择“New”|“Other”命令，将弹出选择创建程序界面，选中“Show All Wizards”选项，界面如图 12-13 所示。

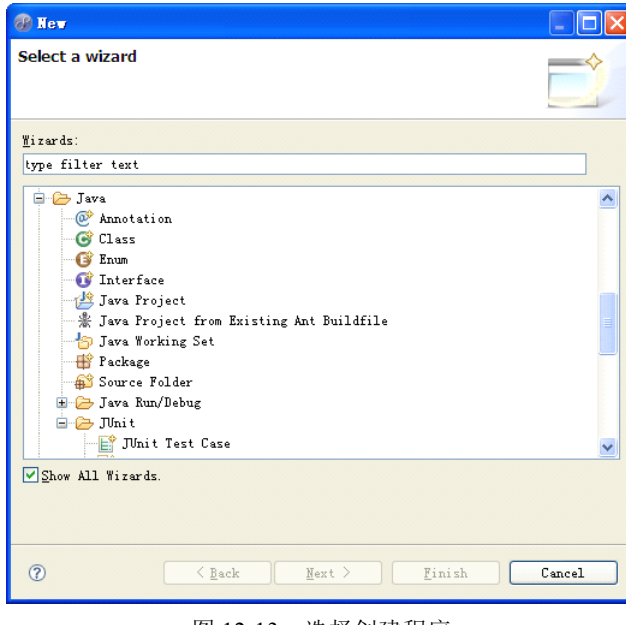


图 12-13　选择创建程序

在其中选中“Java”|“JUnit”|“JUnit Test Case”节点，单击“Next”按钮，将弹出创建 JUnit 程序的界面，如图 12-14 所示。

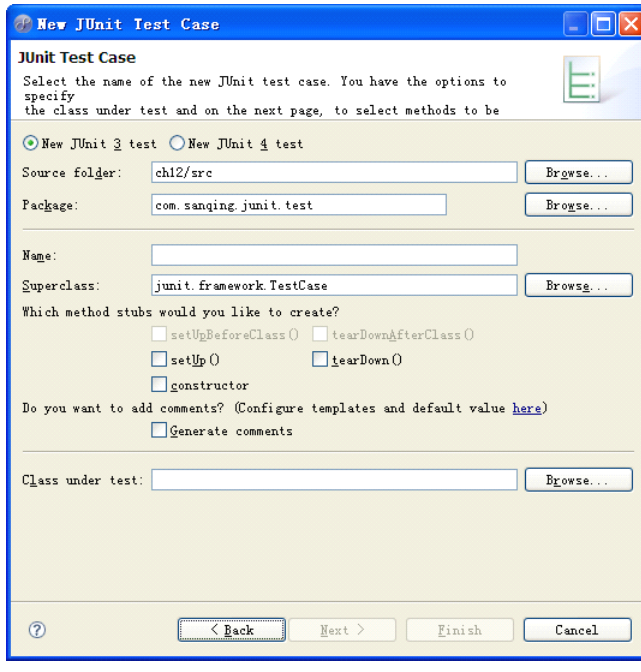


图 12-14　创建 JUnit 程序

界面最开始就是选择创建哪一个版本的 JUnit 程序，其中 JUnit 3 版本流行了很长时间，JUnit 4 版本是 JDK 5.0 以后出现的，和原版本相比变化时比较大的，这里我们直接学习最新版本的 JUnit 4 版本。

其中“Source folder”表示 JUnit 测试程序所在的目录，“Package”表示 JUnit 测试程序所在的包。“Name”表示测试程序类的名称。

注意：测试程序类的类名通常是固定格式的，为 XXXTest 形式，其中 XXX 就是被测试程序的类名。

“Superclass”表示测试类将要继承的父类，对于 JUnit 3 版本来说需要继承“junit.framework.TestCase”类，但是对于 JUnit 4 版本来说是并不需要的，直接使用注解标识就可以，在后面的代码中可以看到这一点。

“Which method stubs would you like to create”表示在测试程序中创建哪些方法，这里保持默认的不选中状态。“Class under test”表示对哪一个类进行测试，单击“Browse”按钮可以选择，这里填写本节开始创建的 HelloWorld 程序。单击“Next”按钮，将进入选择测试方法的界面如图 12-15 所示。

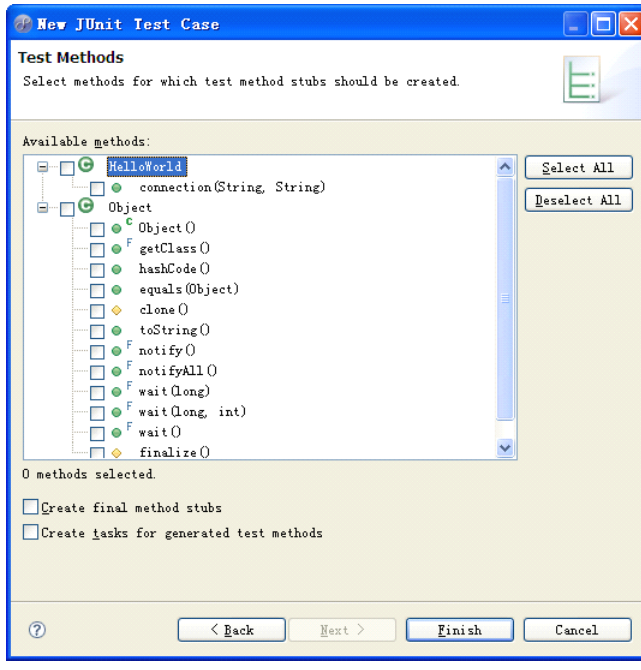


图 12-15　选择测试方法

其中不但将被测试类本身的方法列出来，还会将它的父类中的方法同样给出。这里我们只对其中的连接字符串方法进行测试，选择“connection”方法，单击“Finish”按钮，将弹出导入 JAR 包的界面，如图 12-16 所示。

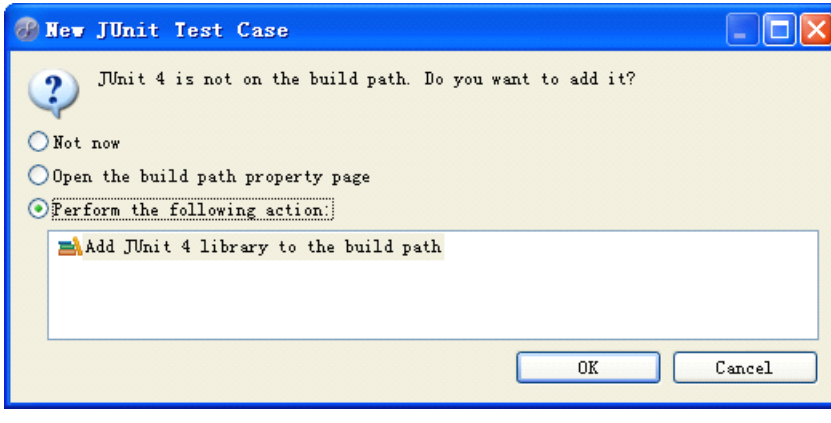


图 12-16　选择导入 JAR 包

其中有三个选项，“Not now”表示不加入 JAR 包；“Open the build path property page”表示打开构建路径界面，手动导入 JAR 包；“Paerform the following action”表示使用 MyEclipse 中自带的 JUnit 相关 JAR 包。这里采用默认的使用自带包，单击“OK”按钮，将完成测试程序的创建。

说明：MyEclipse 中自带的 JUnit 包并不是十分优秀的，当进行某些操作时，需要手动导入最新的 JAR 包。在后面使用到时进行讲解。

完成上述操作后，会在编辑区中自动打开创建的测试程序，自动生成的代码为：

```
01  package com.sanqing.junit.test;
02  import static org.junit.Assert.*;
03  import org.junit.Test;
04  public class HelloWorldTest {
05      @Test
06      public void testConnection() {
07          fail("Not yet implemented");
08      }
09  }
```

注意：第 2 行中使用的是静态导入方法，并不是通常用的导入接口或者类，它是 JDK 5.0 以后加入的，通过它可以在程序中直接使用指定类的静态方法，例如“Assert”类中的静态方法。

其中第 5 行使用到了“@Test”注解，通过它可以标明下面第 6 行的“testConnection”是一个测试方法。

注意：测试方法的名称是有固定格式的，通常为“testXXX”，其中 XXX 就是被测试方法的名称。虽然在 JUnit 4 版本中并没有严格规定，但是最好采用同样规范。

12.2.2 开发测试代码

创建测试程序后，将自动生成测试代码，但是这并不是我们想要的。我们测试方法是为了验证它是否能够完成某一个功能。例如连接字符串方法，我们就来验证是否能够达到连接字符串的效果。将自动生成的代码修改成如下代码：

```
01  package com.sanqing.junit.test;
02  import static org.junit.Assert.*;
03  import org.junit.Test;
04  import com.sanqing.junit.HelloWorld;
05  public class HelloWorldTest {
06      @Test
07      public void testConnection() {
08          String s=new HelloWorld().connection("Hello", "World");
09          assertEquals("HelloWorld", s);          //判断是否和期望值相等
10      }
11  }
```

其中第 8 行中调用了被测试程序中的被测试方法，得到一个返回的字符串变量。在第 9 行中调用了“Assert”类中的 assertEquals 静态方法，该方法中具有两个 Object 类型的参数，其中第 1 个参数表示期望值，第 2 个参数表示真实值，它的作用就是判断真实值和期望值是否相等。

在“Assert”类中除了 assertEquals 静态方法外，还有其他一些判断方法，在后面的学习中将会学习到。

12.2.3 运行测试程序

当运行普通 Java 程序时，如果有输出结果，将把输出结果输出到控制台中。运行测试程序是和运行普通程序有很大不同的，在本小节中就来看一下如何运行测试程序和查看运行结果。

在运行测试程序之前，我们首先将 MyEclipse 切换到 Java 透视图。在 MyEclipse 菜单中，选择“Window”|“Open Perspective”|“Other”命令，在弹出界面中选择“Java”节点，单击“OK”按钮，将完成 MyEclipse 视图的切换，切换到 Java 透视图下。

运行测试程序和运行普通程序一样，也是有多种操作方式，这里采用右键菜单的方式。在测试程序编辑区中，单击鼠标右键，在弹出菜单中选择“Run”|“JUnit Test”命令，将出现一个“JUnit”视图界面，界面如图 12-17 所示。

图 12-17　测试正确结果

界面中最明显的就是最上方的绿条，出现该绿条后就表示测试通过，功能代码是正确可用的。在该界面中的“  ”按钮是启动按钮，如果测试程序发生改变，需要再次执行测试，就可以单击该按钮。

例如将测试程序中第 9 行的“HelloWorld”改为“NOHelloWorld”，单击启动按钮，“JUnit”视图界面将变为如图 12-18 所示的结果。

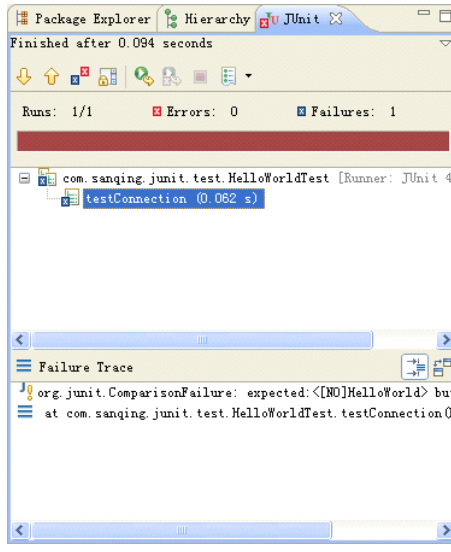


图 12-18　测试错误结果

因为我们将测试中的期望值改变了，所以将和输出值是不同的，从而将测试得到错误结果。在“JUnit”视图界面中将显示红条，表示测试不通过。在最下方的“Failure Trace”栏中将把测试不通过的原因显示出来。

12.3　测试方法

在上一节中已经讲解了一种测试方法，比较方法输出是否和期望值相等。除了该方法外，还有其他比较的方法，例如比较结果是否为 false 或者 true，比较结果是否为 null，比较结果是否相同等。这些都是 JUnit 3 版本中已经有的，在 JUnit 4 版本中又增加了 assertThat 方法，在本节中将对这些内容进行讲解。

12.3.1　输出测试错误信息

在每一种测试方法中，都有一个对应的多出一个字符串类型参数的重载方法，该字符串的值就是测试失败后，输出的错误信息。

修改上一节的测试程序，在该程序中再加入一个测试方法，该方法的代码为：

```
01    @Test
02    public void testConnection2() {
03          String s=new HelloWorld().connection("Hello", "World");
04          assertEquals("输出结果和期望值不同","NOHelloWorld", s); //判断是否相等
05    }
```

如果这时候再在编辑区中的右键弹出菜单中，选择命令进行测试，将同时测试其中的两个方法。如果只想测试这一个方法，可以在包资源管理中，选中“testConnection2”方法节点，单击鼠标右键，在弹出的菜单中选择“Run As”|“JUnit Test”命令，这时候将只测试指定的方法。测试结果如图 12-19 所示。

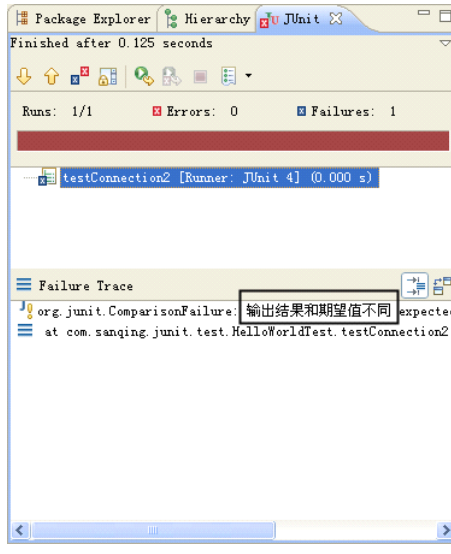


图 12-19　输出错误信息

其中上半部分是和前面相同的，在“Failure Trace”栏中将把方法中定义的信息显示在其中。该信息只有测试失败时才会出现，当测试成功后将被忽略掉，通过它能够更好的检查到哪一个方法测试没有通过。

12.3.2　测试结果 true 和 false

在“org.junit.Assert”类中定义了“assertTrue”和“assertFalse”方法，通过这两个方法可以测试参数是否为 true 或者 false。对于返回值为 booean 类型的方法来说，一般采用这两种方法进行测试。首先我们先来开发一个被测试的方法，这里以登录举例说明，它的程序代码为：

```
01  package com.sanqing.junit;
02  public class UserServiceImpl {
03      public boolean login(String username,String password){
04          if("Tom".equals(username)&&"456123".equals(password)){
05              return true;
06          }else{
07              return false;
08          }
09      }
10  }
```

其中定义了一个登录业务方法，接下来就是对该方法进行测试。开发测试程序的步骤在前面已经讲解了，这里就不再重复介绍。测试程序的代码为：

```
01  package com.sanqing.junit.test;
02  import static org.junit.Assert.*;
03  import org.junit.Test;
04  import com.sanqing.junit.UserServiceImpl;
05  public class UserServiceImplTest {
```

```
06          @Test
07          public void testLogin() {
08                boolean b=new UserServiceImpl().login("Tom", "456123");
09                assertTrue(b);
10          }
11    }
```

启动运行测试，“JUnit”视图界面如图 12-20 所示。

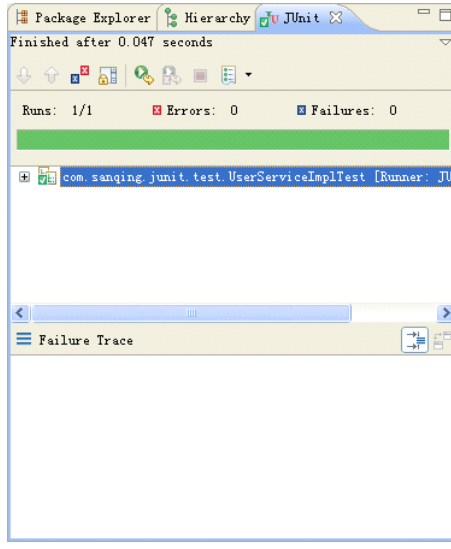


图 12-20　true 测试

上面代码第 8 行中使用测试数据调用登录测试方法，从而得到表示是否能够登录的 boolean 类型返回值。在第 9 行中使用“assertTrue”方法判断返回值是否为 true，如果是 true，则测试通过，得到上面的结果。

说明：“assertFalse”方法和“assertTrue”方法非常类似的，它是用来判断结果是否为 false。如果是，则测试成功，如果不是，则测试失败。

12.3.3　测试结果是否为 null

在实际开发中，测试方法的返回结果是否为 null 是非常重要的，因为该返回结果可能会被使用来调用其他方法。例如在 Web 开发中，在业务逻辑层中要使用数据访问层的对象来调用数据访问层中的方法，在使用之前，通常就应该对数据访问层对象进行是否为 null 测试。

接下来就来演示测试数据访问层对象是否 null，先来开发业务逻辑层程序，程序代码为：

```
01    package com.sanqing.junit;
02    public class UserServiceImpl {
03          private UserDAO userDAO;
04          public UserDAO getUserDAO() {
05                return userDAO;
```

```
06          }
07          public void setUserDAO(UserDAO userDAO) {
08                this.userDAO = userDAO;
09          }
10    }
```

如果学习过本书前面的内容，是不难理解该代码的。上述代码第 3 行中定义了数据访问接口引用，并为它定义了 Setter 方法，从而如果使用 Spring，将把数据访问实现类对象注入。使用其中第 4 行的获取对象方法，就可以获取该对象，接下来就对该获取对象方法进行测试，判断是否将对象注入进来。测试程序中的测试方法代码为：

```
01    @Test
02    public void testGetUserDAO() {
03          UserDAO userDAO=new UserServiceImpl().getUserDAO();
04          assertNull(userDAO);
05    }
```

启动运行测试，“JUnit”视图界面如图 12-21 所示。

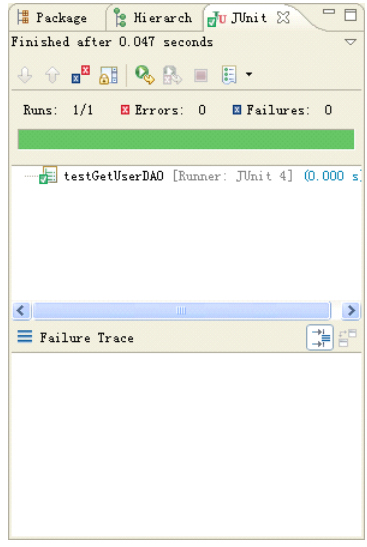


图 12-21　是否为 null 测试

因为在程序中并没有使用到 Spring，所以调用 getUserDAO 方法是不会获取到对象的，从而 userDAO 引用的值就是 null。在第 4 行中调用 assertNull 方法测试值是否为 null，从而测试通过，出现上面的结果。

说明：和 assertNull 方法类似的还有一个“assertNotNull”方法，当测试值不为 nul 时，使用该方法通过测试。

12.3.4　测试是否引用同一个对象

在前面使用“assertEquals”方法时，是用来判断两个参数的值是否相等。和它相似的还有一个“assertSame”方法，通过该方法可以判断两个参数是否引用同一个对象，也从而定义该方法的参数类型必须是对象类型。

多个引用指向一个对象的情况最多的就是出现在单例模式中，我们首先创建一个单例模式程序，它的代码为：

```
01  package com.sanqing.junit;
02  public class MyBean {
03      private static MyBean instance =null;
04      private MyBean(){
05      }
06      public static MyBean getInstance(){
07          if(instance==null){
08              instance=new MyBean();
09          }
10          return instance;
11      }
12  }
```

上述代码是一个标准的单例模式程序，其中最关键的就是第 4 行定义了私有的构造函数，从而不能在类外部创建对象，只能够通过第 6 行中的方法获取。接下来就是对其中第 6 行的方法进行测试，判断是否返回的都是同一对象。

创建测试程序，然后在其中加入测试代码，其具体代码为：

```
01  package com.sanqing.junit.test;
02  import static org.junit.Assert.*;
03  import org.junit.Test;
04  import com.sanqing.junit.MyBean;
05  public class MyBeanTest {
06      @Test
07      public void testGetInstance() {
08          MyBean myBean1=MyBean.getInstance();
09          MyBean myBean2=MyBean.getInstance();
10          assertSame(myBean1, myBean2);
11      }
12  }
```

启动运行测试，“JUnit”视图界面如图 12-22 所示。

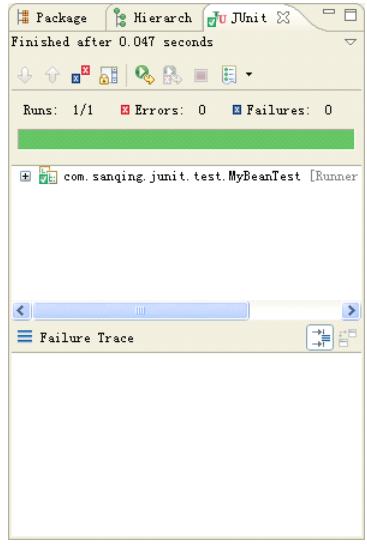


图 12-22　测试是否为一个对象

在代码中第 8 行和第 9 行都调用了单例默认程序中获取对象的方法，然后在第 10 行中调用 assertSame 方法判断获取的对象是否为同一个对象。

注意：和前面几种方法一样，也有和“assertSame”功能相反的“assertNotSame”方法，使用它时，当两个引用指向不同对象时测试通过。

12.3.5　新版本新增方法

在 JUnit 4 版本中新增了一个方法，那就是“assertThat”方法，使用该方法可以完成上面所讲的所有方法的功能。“assertThat”方法的基本语法结构为：

public static void assertThat([value], [matcher statement]);

其中 value 表示想要测试的变量值。matcher statement 是使用 Hamcrest 匹配符来表达的对前面变量所期望的值的声明，如果 value 值与 matcher statement 所表达的期望值相符，则测试成功，否则测试失败。

在 JUnit 框架中自带了有限的几个 Hamcrest 匹配符，如果想使用更多的，可以去下载 Hamcrest 匹配符的相关 JAR 包，然后将它们导入到测试项目中。Hamcrest 匹配符的官方网站为“http://code.google.com/p/hamcrest/”，在其中选择下载栏目就可以下载到最新版本的 Hamcrest 匹配符下载包。将下载包解压缩后，将根目录下的“hamcrest-library-1.2.jar”和“hamcrest-core-1.2.jar”两个文件导入到项目就可以在项目中使用所有 Hamcrest 匹配符。

注意：MyEclipse 中自带 JUnit 的 JAR 包是和下载的 Hamcrest 匹配符 JAR 包相冲突的，所以要去 JUnit 官方网站下载最新包，官方网站地址为“www.junit.org/”。

下面我们就通过“assertThat”方法使用 Hamcrest 匹配符的方式来测试最开始的连接字符串方法，测试方法的代码如下所示。

```
01    @Test
02    public void testConnection3() {
```

```
03          String s=new HelloWorld().connection("Hello", "World");
04          assertThat ("HelloWorld", is(s));          //判断是否和期望值相等
05      }
```

其中第 4 行中使用了 assertThat 方法，方法中的第二个参数“is(s)”就是一个 Hamcrest 匹配符，判断值是否等于前面的值，它的作用是和 assertEquals 方法相同的。

从上面这一个简单程序，就应该能够体现出 assertThat 方法的优点。首先在测试程序中不在需要多种方法，通过 assertThat 方法就可以代替以前所有的老方法，使得编写测试用例变得简单，代码风格变得统一，测试代码也更容易维护。

其次使用 assertThat 方法更适合代码阅读，因为在 Hamcrest 匹配符中使用了大量的符合英文的格式，例如 is、allOf、anyOf 等。

这里我们就不为每一个匹配符进行代码举例，下面给出部分重要通配符的使用格式，这里按照它们的作用不同分为几类。

- 1．一般通配符

（1）allOf 匹配符表明当后面的所有条件都成立时，测试才能通过，相当于运算符中的“与”（&&）。例如：

```
assertThat( testedNumber, allOf( greaterThan(3), lessThan(10) ) );
```

表示 testedNumber 的值必须大于 3，并且小于 10，则测试通过。

（2）anyOf 匹配符表明后面的所有条件中只要有一个成立就测试通过，相当于运算符中的“或”（||）。例如

```
assertThat( testedNumber, anyOf( greaterThan(10), lessThan(3) ) );
```

表示 testedNumber 的值大于 10，或者小于 3，则测试通过。

（3）not 匹配符和 is 匹配符正好相反，表明如果前面测试的 object 不等于后面给出的 object，则测试通过。例如：

```
assertThat( testedString, not( "developerWorks" ) );
```

- 2．字符串相关匹配符

（1）containsString 匹配符表明如果测试的字符串 testedString 包含子字符串"developerWorks"则测试通过。例如

```
assertThat( testedString, containsString( "developerWorks" ) );
```

（2）endsWith 匹配符表明如果测试的字符串 testedString 以字符串"developerWorks"结尾则测试通过。例如

```
assertThat( testedString, endsWith( "developerWorks" ) );
```

（3）startsWith 匹配符表明如果测试的字符串 testedString 以字符串"developerWorks"开始则测试通过。例如

```
assertThat( testedString, startsWith( "developerWorks" ) );
```

（4）equalTo 匹配符表明如果测试的 testedValue 等于 expectedValue 则测试通过。例如

```
assertThat( testedValue, equalTo( expectedValue ) );
```

（5）equalToIgnoringCase 匹配符表明如果测试的字符串 testedString 在忽略大小写的情况下等于"developerWorks"则测试通过。例如

assertThat(testedString, equalToIgnoringCase("developerWorks"));

（6）qualToIgnoringWhiteSpace 匹配符表明如果测试的字符串 testedString 在去除收尾的空格后等于"developerWorks"字符串，则测试通过。例如

assertThat(testedString, equalToIgnoringWhiteSpace("developerWorks"));

- 3．数值相关匹配符

（1）closeTo 匹配符表明如果所测试的浮点型数 testedDouble 在 20.0±0.5 范围之内则测试通过。例如

assertThat(testedDouble, closeTo(20.0, 0.5));

（2）greaterThan 匹配符表明如果所测试的数值 testedNumber 大于 16.0 则测试通过。例如

assertThat(testedNumber, greaterThan(16.0));

（3）lessThan 匹配符表明如果所测试的数值 testedNumber 小于 16.0 则测试通过。例如

assertThat(testedNumber, lessThan (16.0));

（4）greaterThanOrEqualTo 匹配符表明如果所测试的数值 testedNumber 大于等于 16.0 则测试通过。例如

assertThat(testedNumber, greaterThanOrEqualTo (16.0));

（5）lessThanOrEqualTo 匹配符表明如果所测试的数值 testedNumber 小于等于 16.0 则测试通过。例如

assertThat(testedNumber, lessThanOrEqualTo (16.0));

- 4．collection 相关匹配符

（1）hasEntry 匹配符表明如果测试的 Map 对象 mapObject 中，含有一个键值为"key"对应值为"value"的键值对组成，则测试通过。它的语法格式为：

assertThat(mapObject, hasEntry("key", "value"));

（2）hasItem 匹配符表明如果测试的迭代对象 iterableObject，它可以使 Set、List 或者数组，其中含有 “element” 元素，则测试通过。它的语法格式为：

assertThat(iterableObject, hasItem ("element"));

（3）hasKey 匹配符表明如果测试的 Map 对象 mapObject 中含有 “key”键，则测试通过。它的语法格式为：

assertThat(mapObject, hasKey ("key"));

（4）hasValue 匹配符表明如果测试的 Map 对象 mapObject 中含有“value”值，则测试通过。它的语法格式为：

assertThat(mapObject, hasValue ("key"));

说明：Hamcrest 匹配符并不是我们的学习重点，在其中读者只需要知道 assertThat 方法的使用方式就可以，如果对它感兴趣，可以看一下官方文档，它是非常容易学习的。

12.4 注解在测试中的使用

注解是 JDK 5.0 版本以后添加的新特性技术，通过使用它可以在程序中节省大量代码。在 JUnit 4 版本中也引入的注解，例如前面见过多次的“@Test”注解，通过它就可以标明下面的方法是一个测试方法，从而就不需要让测试类继承其他类。除了“@Test”注解外，还有其他几种常用的注解，在本节中就来对它们进行讲解。

12.4.1 测试程序是否发生异常

在前面使用“@Test”注解时只是标明下面的方法是测试方法，在该注解中还可以添加属性来进行更详细的测试，例如 expected 属性，通过它可以测试程序中是否发生指定的异常。例如在程序中定义了一个执行除法运算的业务方法，它的代码为：

```
01  package com.sanqing.junit;
02  public class DivideMath {
03      public double divide(int a,int b){
04          return a/b;
05      }
06  }
```

其中第 3 行定义了执行除法运算的业务方法，当执行该方法时，就可以发生除数为 0 的异常。现在我们就来测试使用两个数调用该方法是否发生除数为 0 的异常。测试代码如下所示：

```
01  package com.sanqing.junit.test;
02  import org.junit.Test;
03  import com.sanqing.junit.DivideMath;
04  public class DivideMathTest {
05      @Test(expected=java.lang.ArithmeticException.class)
06      public void testDivide() {
07          double d=new DivideMath().divide(3, 0);
08      }
09  }
```

启动运行测试，“JUnit”视图界面如图 12-23 所示。

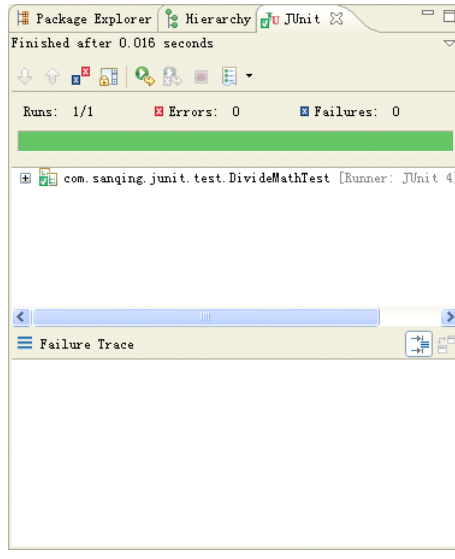


图 12-23　测试异常

在代码中的第 7 行是测试除法方法，设置其中的除数为 0，执行该方法是会发生异常的。在测试方法的上面使用“@Test”注解中的“expected”属性来测试程序中是否会发生“java.lang.ArithmeticException”。当发生该异常时，则测试通过，从而出现上面的结果。

如果在测试方法上的“@Test”注解中不使用“expected”属性，也就是将测试代码该为如下代码：

```
01    @Test
02    public void testDivide2() {
03          double d=new DivideMath().divide(3, 0);
04    }
```

启动运行测试该方法，“JUnit”视图界面如图 12-24 所示。

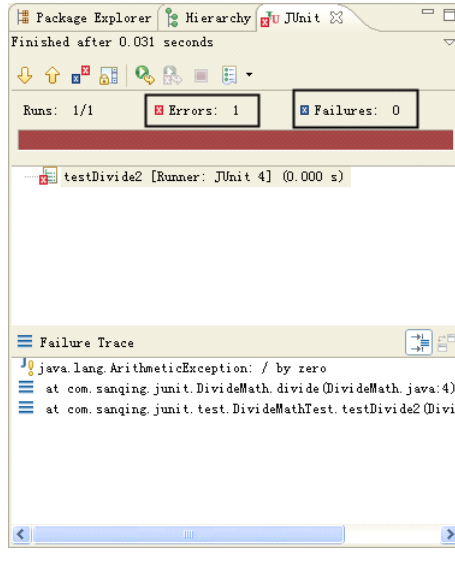


图 12-24　测试异常失败

从界面中可以看到测试没有通过。

注意：测试没有通过可能有两种原因。一种就是最常见的程序功能有问题，另一种就是测试程序中发生异常，该程序就是该原因。在界面中也会进行信息提示，在红条的上面，“Error”提示的就是错误或者异常数量，“Failures”就是测试不同的数量。

12.4.2　测试程序运行时间

在前面的学习中都是测试程序是否正确，但是对于项目而言，除了正确以外，效率也是非常重要的。在测试中，也可以对执行效率进行测试，它是通过“@Test”注解中的“timeout”属性完成的。测试程序运行时间的代码如下所示。

```
01    @Test(expected=java.lang.ArithmeticException.class,timeout=100)
02    public void testDivide3() {
03          double d=new DivideMath().divide(3, 0);
04    }
```

其中第 1 行中使用“timeout”属性指定程序所花费的时间，当执行被测试方法时，花费的时间超过“timeout”属性值，则测试失败。反之则测试成功。

启动运行测试，“JUnit”视图界面如图 12-25 所示。

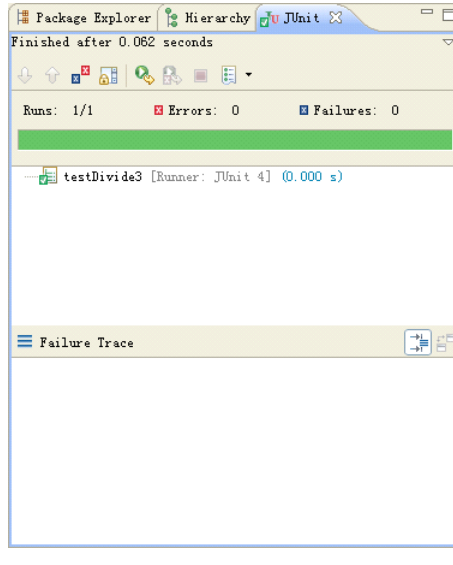


图 12-25　测试时间

出现绿条，表示测试成功，也就是执行除法运算方法所花费的时间不会超过 100 毫秒。

说明：对于本地方法调用来说，是通过不需要进行时间测试的。但是对于 Web 程序或者执行大量查询的数据库程序，测试程序的运行时间是非常重要的。如果时间不符合要求，那就需要调整开发方式。

12.4.3 测试方法的初始化和销毁

在讲解知识之前，我们来分析几个情况。例如在 Web 项目中，我们要对 Spring 管理的所有业务方法进行测试，在其中都要执行实例化 IoC 容器等操作。再例如对数据访问层的方法进行测试，执行完测试后，都要关闭数据库资源。这些情况下，每一个测试方法中都要写相同的代码。

在这种情况下就可以考虑使用“@Before”注解和“@After”注解，通过这两种注解标明的方法就会在测试方法之前和之后执行相同的代码。例如在“DivideMathTest”测试类程序中加入下面的代码：

```
01  @Before
02  public void before() {
03      System.out.println("进行 Spring 中 IoC 容器的实例化");
04  }
05  @After
06  public void after() {
07      System.out.println("释放数据库资源");
08  }
```

加入这些代码后，启动运行测试，这里我们不关心测试的结果，直接看控制台中，程序的输出结果为：

```
进行 Spring 中 IoC 容器的实例化
释放数据库资源
进行 Spring 中 IoC 容器的实例化
释放数据库资源
```

在程序中进行了两个方法的测试，从运行结果中可以看到各执行了两次上面的方法。其中“@Before”注解的方法将在每一个测试方法之前执行，“@After”注解的方法将在每一个测试方法之后执行。

12.4.4 测试类的初始化和销毁

不但可以为每一个方法定义初始化和销毁方法，还可以直接为测试类定义这样的方法。“@BeforeClass”注解标明的方法就是一个测试类初始化方法，当执行该测试类时将首先执行该方法。“@AfterClass”注解标明的方法就是测试类的销毁方法，当执行完测试类中的所有测试方法后，将执行该方法。我们在继续在“DivideMathTest”测试类程序中加入下面的代码：

```
@BeforeClass
public static void beforeClass() {
 System.out.println("开始执行用户业务逻辑层测试");
}
@AfterClass
public static void afterClass() {
 System.out.println("结束执行用户业务逻辑层测试");
}
```

加入代码后，再次启动测试，则控制台的输出结果如下所示：

开始执行用户业务逻辑层测试
进行 Spring 中 IoC 容器的实例化
释放数据库资源
进行 Spring 中 IoC 容器的实例化
释放数据库资源
结束执行用户业务逻辑层测试

从运行结果中可以看到，在执行所有测试方法之前，执行了“@BeforeClass”注解标明的方法。当执行完所有测试方法后，将执行“@AfterClass”注解标明的方法。

注意：“@BeforeClass”注解和“@AfterClass”注解标明的方法一定要是 static 修饰的静态方法，这样才能保证在类实例化之前就能够执行方法。

12.5 运行多个测试

在前面的学习中，学习了如何运行单个测试方法，也学习了如何运行一个测试类中的所有测试方法。如果仅仅能够这样，测试效率是非常低的，对于一个大型项目而言，运行测试的时间就是非常多的。

在 JUnit 测试中，还可以对整个包中的测试方法同时运行。因为项目中的所有测试程序通常放在同一包下，经过该操作后就可以同时运行一个项目中的测试。在 MyEclipse 的包资源管理器中选中测试程序所在的包，这里选择“com.sanqing.junit.test”包，单击鼠标右键，在弹出的菜单中选择“Run As”|“JUnit Test”命令，就会执行整个包的所有测试方法。执行完成后，“JUnit”视图界面如图 12-26 所示。

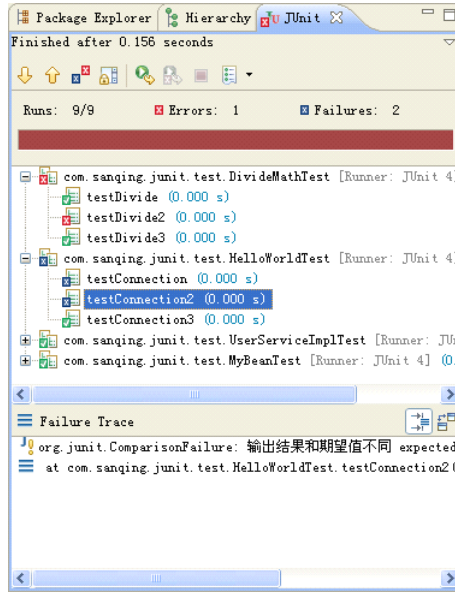


图 12-26　运行多个测试

当执行多个测试时，只要其中有一个不能通过测试，都会出现红条。在红条上面，在“Error”后会显示有多少个测试方法发生异常，在“Failures”后会显示有多少个测

试方法测试失败。

在下面选中没有通过测试的方法，将在“Failure Trace”栏中显示出它的具体信息。然后依次对没有通过测试的方法进行排错处理，直到在界面中出现绿条，所有方法通过测试。

说明：测试是一个反复的过程，没有程序员能够保证写的程序都能通过测试，也不能保证每一次排错后都不会在出错。测试的方法可以学会，但是测试的经验需要每一个读者自己总结。

第13章　CVS和SVN版本控制

在开发小型项目中，通常只保留最新的版本。对于大型项目而言，这种做法就会有一定的问题。因为在大型项目改动是比较多的，而且不能保证最新修改的版本就是最好的版本，所以经常要反复操作，这时候就要用到版本管理的概念。当工作到一个阶段，就要进行版本保存，在后面的工作中或者其他开发人员可以使用指定的某一个版本。

随着项目的发展，版本管理越来越重要，它在当前软件开发中是必不可少的。版本管理是一种功能，目前实现该功能的技术有很多，在 MyEclipse 中本身集成了标准的版本管理系统 CVS，该技术已经流行了很长时间。除了 CVS 外，SVN 也是最近流行起来的版本管理工具，它包含 CVS 的所有功能，并且修复了 CVS 的一些缺陷。在本章中将重点讲解在 MyEclipse 中如何操作 CVS 和 SVN 这两个版本管理工具。

13.1　本地版本控制

CVS 和 SVN 是在大型项目中团队开发和阶段性开发起到非常重要的作用，但是如果是对于一个人来说，或者对于某一个程序文件来说，完全没有必要使用这些工具，只使用 MyEclipse 自带的版本管理功能就可以。

13.1.1　比较

在讲解操作之前，我们先来看一下 MyEclipse 的保存机制。在 MyEclipse 中开发程序，每保存一次都会在其中保存一个历史文件副本。本地的版本管理就是对这些历史副本进行操作。先来学习一下如何比较两个历史副本。我们首先创建一个程序，程序的代码为：

```
01  public class HelloWorld {
02      public static void main(String[] args) {
03          System.out.println("HelloWorld 第 1 个版本 AAA");
04      }
05  }
```

保存该程序后，然后将其中的第 3 行代码换为如下代码：

```
System.out.println("HelloWorld 第 2 个版本 BBB");
```

然后再次保存程序，这时候就会在 MyEclipse 中产生两种历史文件副本。在编辑区中，单击鼠标右键，在弹出的菜单中选择“Compare With”|“Local History”命令，将弹出文件历史视图界面，如图 13-1 所示。

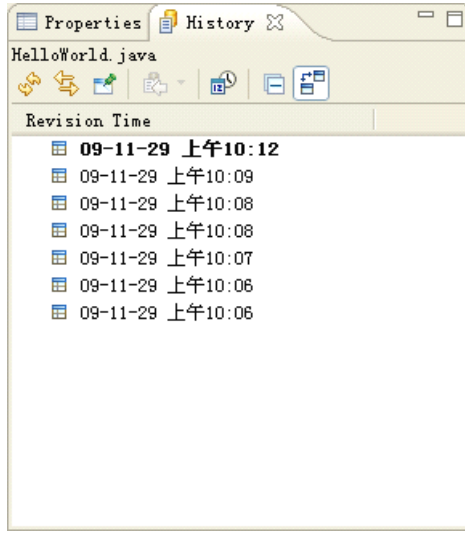


图 13-1　程序历史记录

然后在其中双击某一历史记录，例如选择“09-11-29 上午 10：09”记录，在编辑区中将弹出比较的界面，如图 13-2 所示。

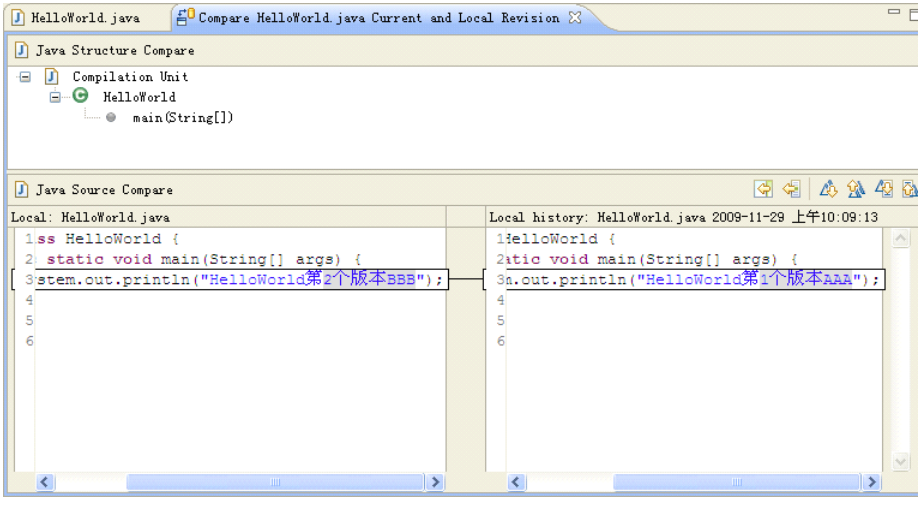


图 13-2　比较界面

界面中上部分是显示当前程序的程序结构。在左下部分中显示了当前程序的代码，右下部分中显示指定历史记录的代码，在其中将两个代码不同的地方标明。在历史记录代码的上面有几个操作按钮，其中“ ”和“ ”按钮表示查找下一个不同处和上一个不同处。“ ”按钮表示使用所有不同的地方替换左边的内容，“ ”按钮表示将当前指定的不同处替换成左边的内容。

注意：不但可以让当前程序和某一历史记录进行比较，还可以让两个历史记录之

间进行比较。在历史记录视图界面中，选中两个历史记录，单击鼠标右键，在弹出的菜单中选择“Copare with Each Other”命令，将弹出对这两个历史记录进行比较的界面。和前面的比较界面非常相似的。

13.1.2 替换

在上一小节中讲解了如何比较两个历史记录，所以对不同处进行替换操作。在MyEclipse 中，也可以直接进行替换操作，将当前程序替换成指定历史记录的副本。

在编辑区中，单击鼠标右键，在弹出的菜单中中选择“Relace With”|“Local History”命令，将弹出替换界面，在其中双击要替换的历史记录，替换界面将变为如图 13-3 所示。

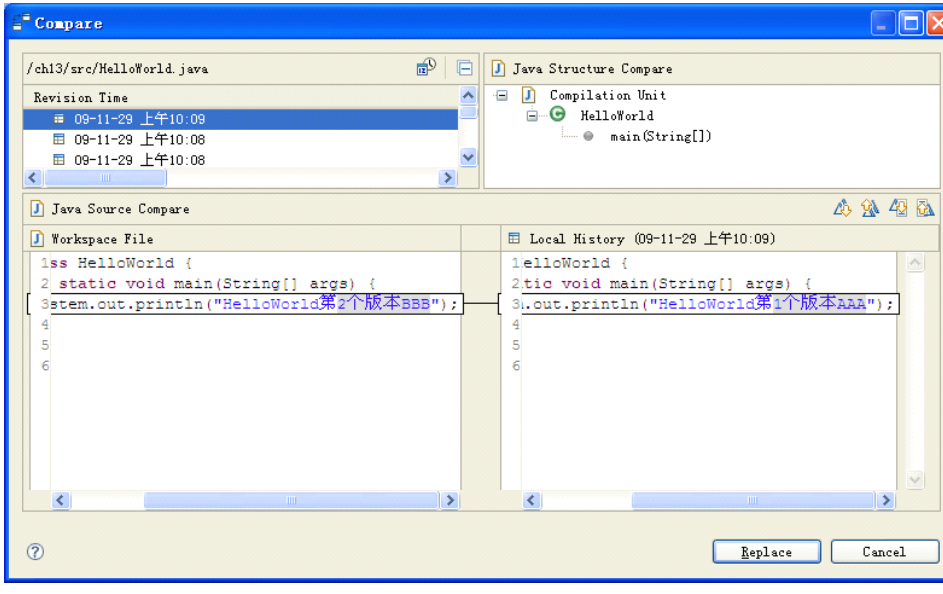


图 13-3 替换界面

在界面中将把当前代码和指定历史记录的代码都显示出来，并将其中的不同处也显示出来。单击“Replace”按钮，将使用右边指定历史记录的代码完全替换左边的当前代码。

技巧：在实际开发中，使用最多的就是替换功能，但是并不是上面的替换操作，而是替换成上一次的历史记录。在编辑区中，单击鼠标右键，在弹出的菜单中选择“Relace With”|“Previous from Local History”命令，并不会弹出任何界面，而是直接替换成上一次保存时的历史记录。使用该操作，要比使用“Ctrl+Z”撤销更快捷。

13.2 CVS 版本控制

使用 MyEclipse 中自带的版本管理功能只能对一台代码中的指定程序进行操作。如

果在团队开发中，该功能就不能达到需求。CVS 是一种控制系统，它是 C/S 结构的。对于每一个程序员来说都是一个客户端，程序员可以上传自己的程序，每次上传 CVS 都会提供一个依次增加的版本号。程序员也可以下载其他程序员开发完成的程序，在其中可以选择最新版本或者某一版本。

在 CVS 的服务器端中，可以接收程序员的上传程序，也可以让程序员下载。服务器端通常是由项目经理管理的，在 MyEclipse 中只集成了 CVS 客户端的功能，想对服务器端进行操作，还可以安装 CVS 服务器。

13.2.1　下载和安装 CVS 服务器

CVS 可以使用在多个操作系统下，对于每一个系统的软件名称是不同的。使用在 Windows 系统下的 CVS 服务器的软件名称为 CVSNT，它的官方网站为“www.cvsnt.org”，在其中下载最新版本的 CVS 服务器。进入官方网站后，如图 13-4 所示。

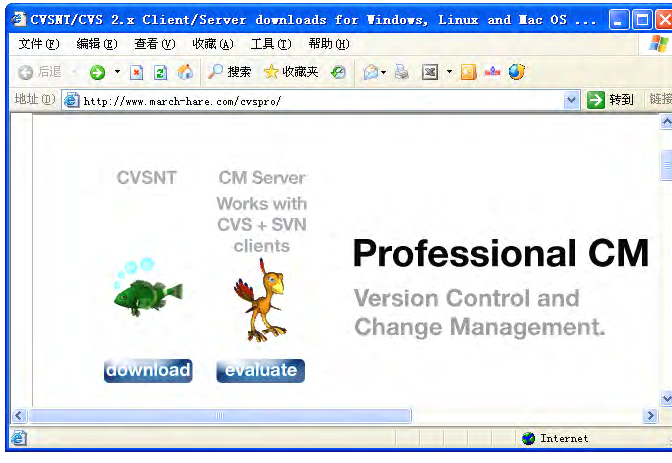


图 13-4　CVS 服务器官方网站

单击 CVSNT 图标下的“download”按钮，将显示下载 CVS 服务器的页面，如图 13-5 所示。

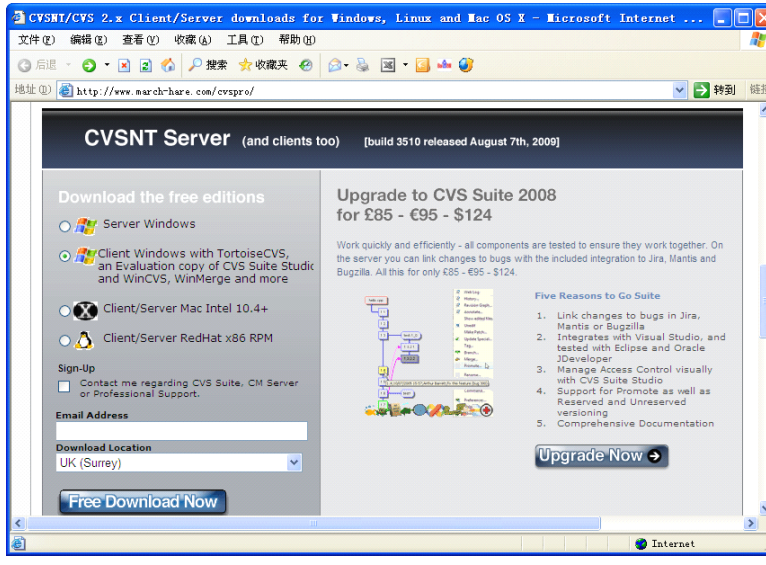


图 13-5　服务器选择页面

在其中选择“Server Windows”选项，单击“Free Download Now”按钮，将弹出下载窗口，或者在下载工具中进行下载。

注意：浏览器可能会阻止下载窗口的出现，如果出现该结果，手动在阻止工具条中单击鼠标右键，选择“下载文件命令”。

将 CVS 服务器端文件下载到本地后，就可以安装它，其中大部分操作只需要单击“Next”按钮就可以，在其中要选择同意协议。进入到安装方式选择界面中后，如图 13-6 所示。

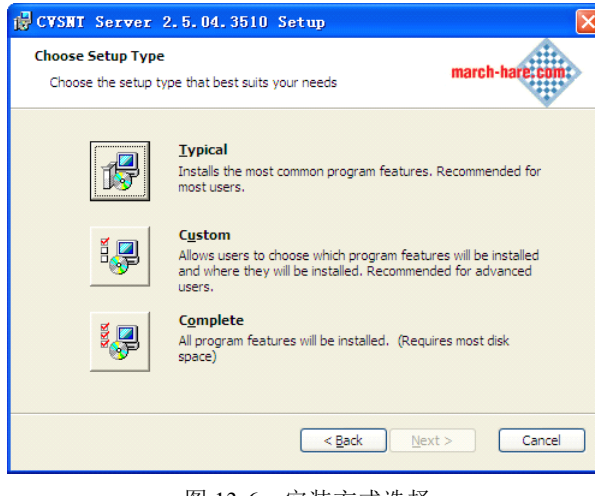


图 13-6　安装方式选择

其中第一个选项“Typical”表示典型安装，选择该选项就可以。然后在下一个界面中单击“Install”按钮将开始安装。稍等一会后，将弹出安装完成界面，单击“Finish”按钮，将完成安装。

13.2.2 配置 CVS 服务器

下载和安装 CVS 服务器后，还需要对 CVS 服务器进行配置。在系统开始菜单中，选择“程序”|“CVSNT”|“CVSNT Control Panel”命令，将弹出配置 CVS 服务器的界面，如图 13-7 所示。

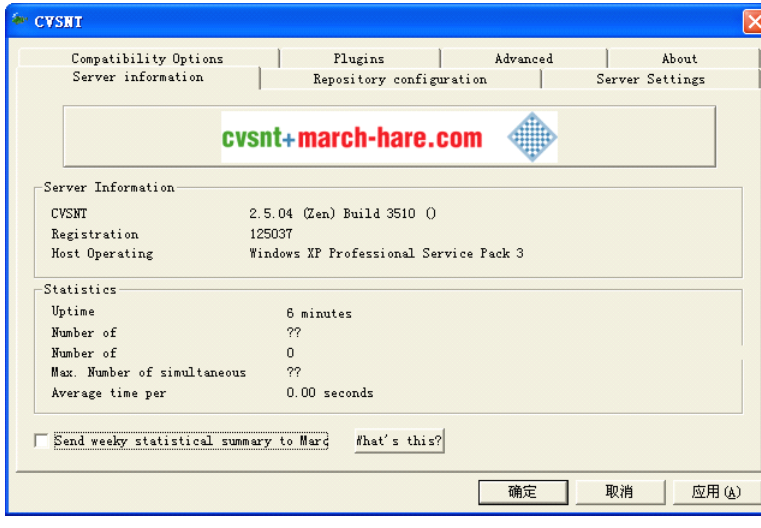


图 13-7　配置界面

在其中选择“Repository configuration”选项卡，将进入到服务器库的界面，如图 13-8 所示。

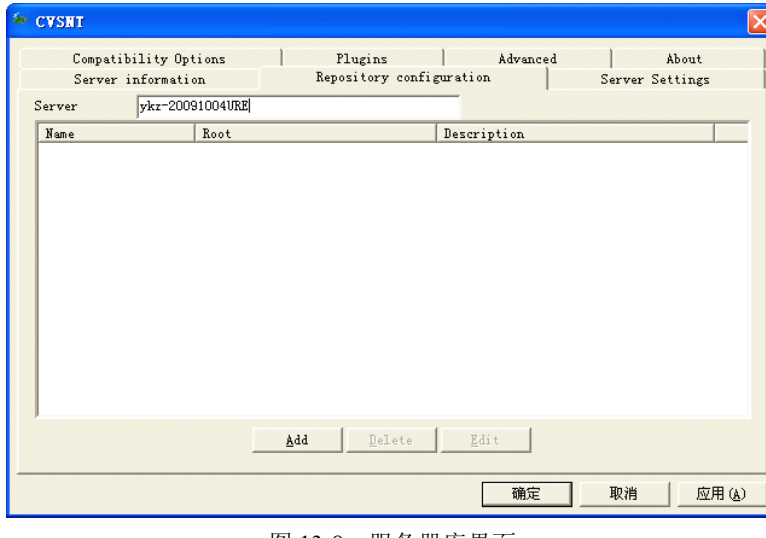


图 13-8　服务器库界面

单击“Add”按钮，将进入到服务配置界面，如图 13-9 所示。

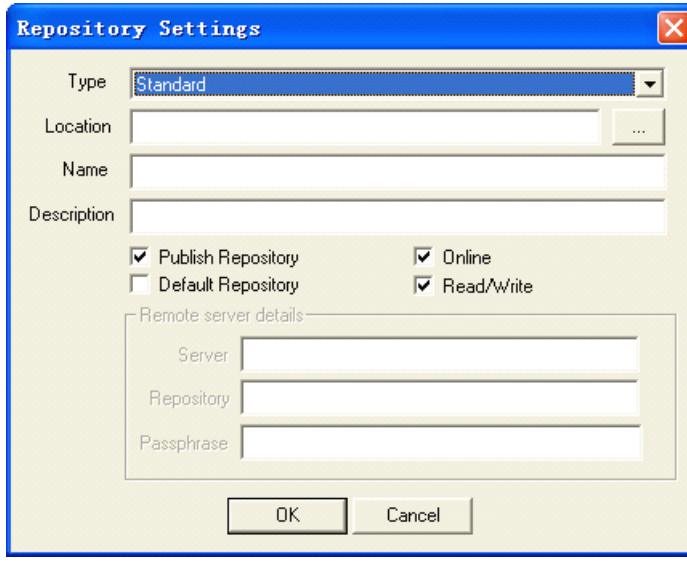


图 13-9　服务配置界面

其中“Location”表示资源库所在的路径，例如我们在 D 盘下创建了一个名称为“CVSNT”的文件夹，在这里就可以单击“…”按钮，选择该文件夹。这样我们就为资源库创建了一个存储空间。单击“OK”按钮，将完成服务器配置中资源库的配置。并且回到服务器库界面，并将新增加的库显示在其中。

在配置界面中再次选择“Server Settings”选项卡，界面如图 13-10 所示。

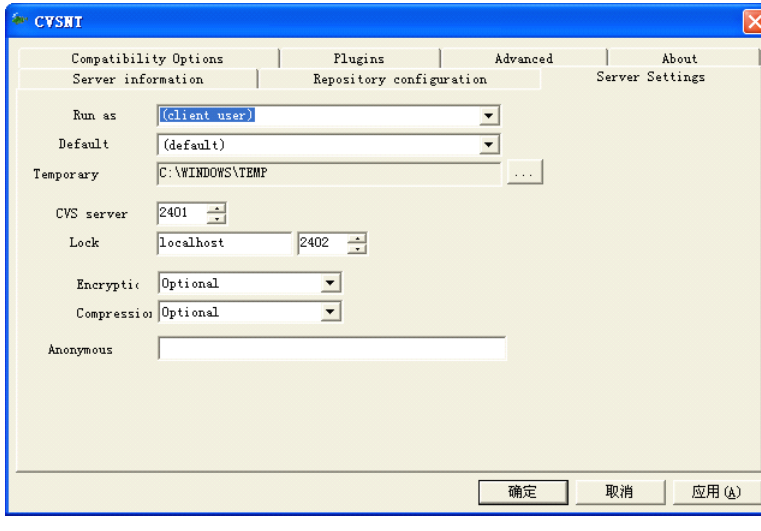


图 13-10　配置管理员

其中“Run as”用来选择管理员，这里选择“Administrator”。“Temporary”表示出错信息的存储目录，保持默认的目录就可以。单击“确定”按钮，将完成 CVS 服务器的配置。

13.2.3 建立 CVS 服务器端连接

安装和配置 CVS 服务器后，我们就可以继续来开发 CVS 的客户端。在 MyEclipse 中集成了 CVS 客户端的功能，并且定义了 CVS 操作透视图，我们通常是在该透视图下操作的。在选择透视图界面中，选择“CVS Repository Exploring”选项，就会将 MyEclipse 切换到 CVS 服务器操作的视图，如图 13-11 所示。

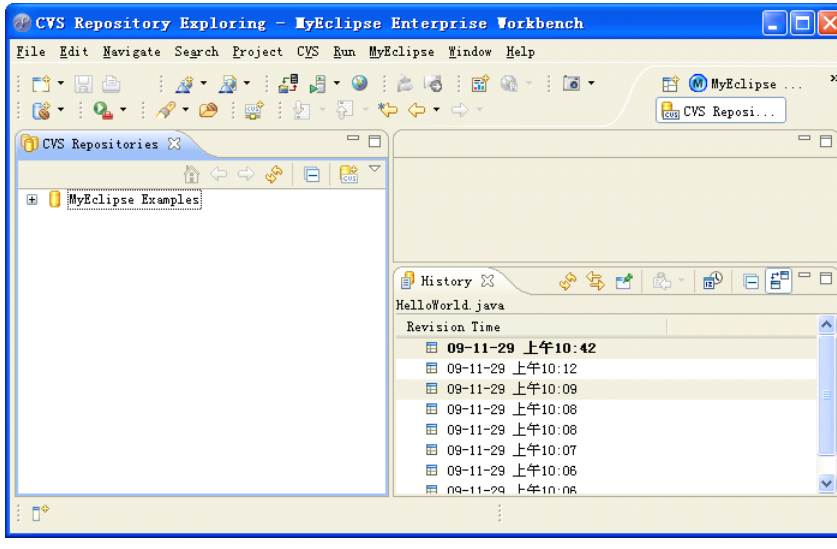


图 13-11　CVS 操作透视图

在该透视图中，我们就可以进行 CVS 客户端的操作。在客户端进行程序的操作之前，要首先和 CVS 服务器端建立连接。在“CVS Repositories”视图界面的空白处，单击鼠标右键，在弹出的菜单中选择“New”|“Repository Location”命令，将弹出添加 CVS 资源库的界面，如图 13-12 所示。

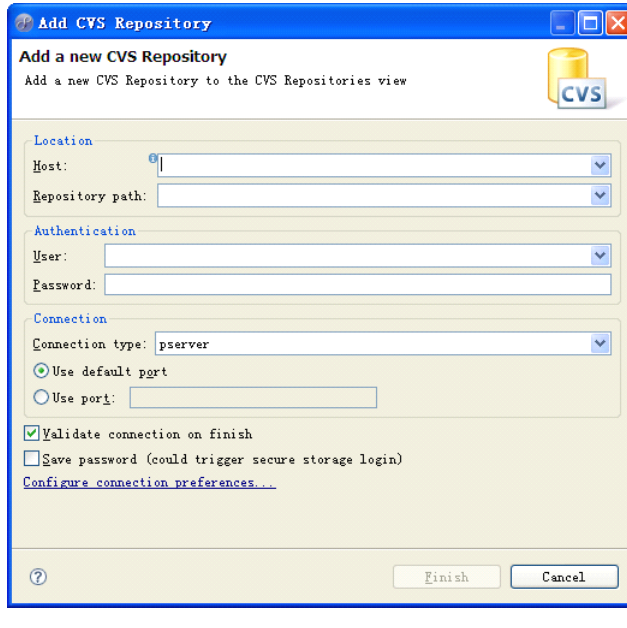


图 13-12　添加 CVS 资源库

其中“Host”表示指定安装 CVS 服务器的主机 IP 地址，因为这里我们做实验在一台计算机上，所以可以填写“LocalHost”。“Repository path”表示资源库的路径，也就是我们在配置 CVS 服务器时配置的，这里填写“/CVSNT”。“User”和“Password”分别表示系统的用户名和密码，这里可以是“Administrator”超级用户，也可以是自己创建的普通用户。其他选项采用默认值就可以，单击“Finish”按钮，将完成 CVS 服务器的连接，从而在“CVS Repositories”视图界面多出一个资源库，如图 13-13 所示。

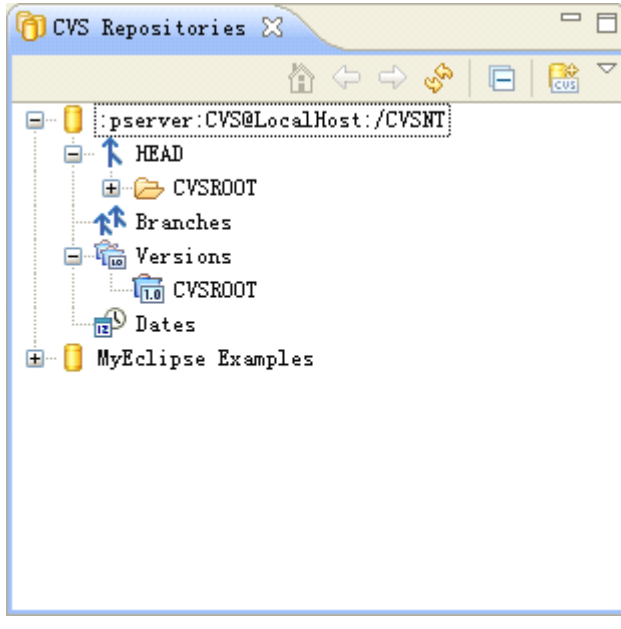


图 13-13　建立 CVS 服务器连接

13.2.4　创建共享项目

和 CVS 服务器端建立连接后，我们就可以进行客户端的操作，首先是学习如何在 CVS 服务器端创建项目。这里我们就以本章的项目为例进行操作。

在 MyEclipse 中，切换到普通透视图后，在包资源管理器中选中要提交的项目。单击鼠标右键，在弹出菜单中选择“Team”|“Share Project”命令，将弹出选择资源库的界面，如图 13-14 所示。

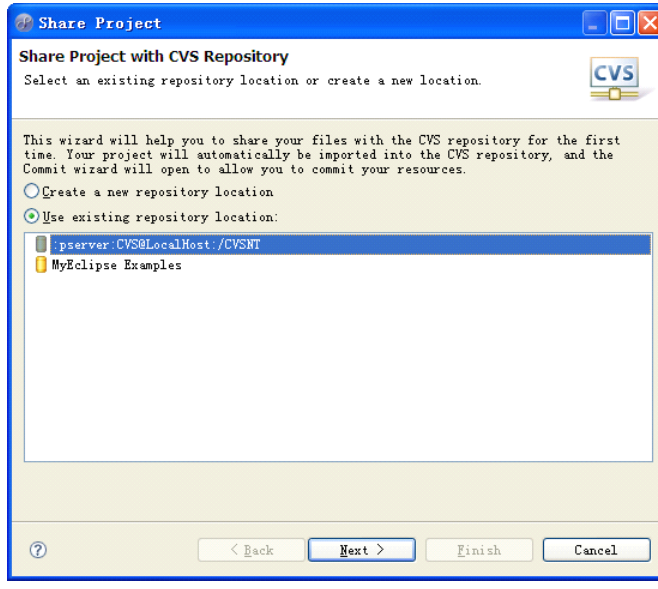


图 13-14　选择资源库

在界面中有两个选项，其中“Create a new repository location”表示创建一个新的资源库，“Use existing repository location”表示使用已经创建的资源库。因为我们在前面的操作中，已经创建了资源库，所以这里选择第 2 个选项，然后在现在选择“.CVSNT”资源库。单击“Next”按钮，将弹出设置提交后名称的界面，如图 13-15 所示。

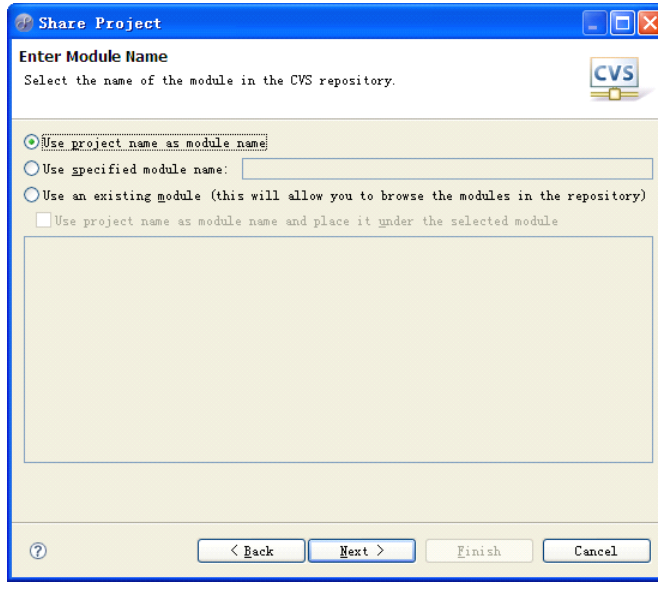


图 13-15　设置名称

其中“Use project name module name”表示使用项目的名称做为提交后名称；“Use specified module name”表示自己指定提交后的名称，选择该选项后，需要在后面填写名称；“Use an existing module”表示使用已经存在的名称。通常情况就是选择“Use project

name module name”选项，单击“Next”按钮，将弹出共享项目资源界面，如图 13-15 所示。

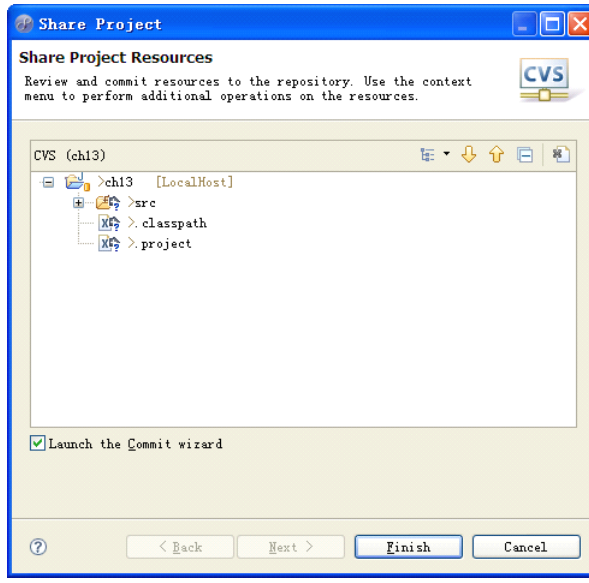


图 13-15　共享项目资源

如果在其中选中了“Launch the Commit wizard”复选框，共享后，会启动提交选项，从而完成第 1 次提交。单击“Finish”按钮，将完成共享操作，并弹出提交项目界面，如图 13-16 所示。

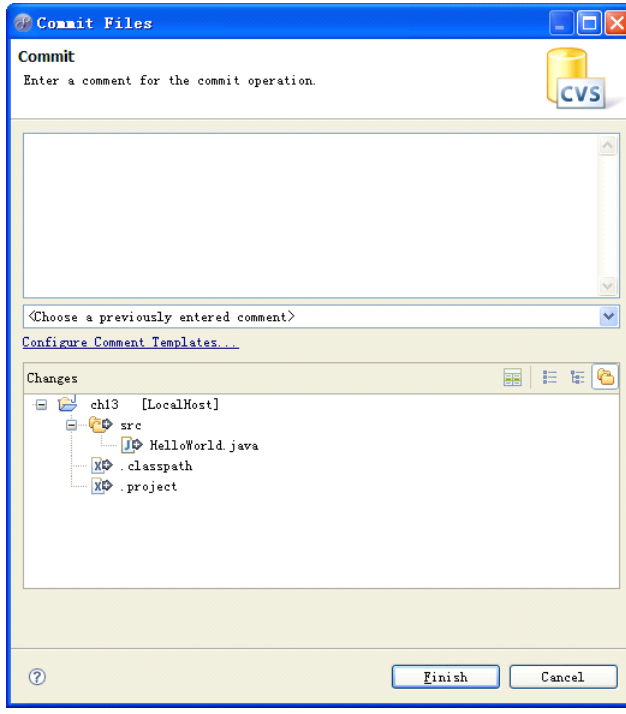


图 13-16　第 1 次提交项目

在界面的上半部分的文本域中，是用来输入关于提交操作的注释和描述。

注意：在实际开发中，它是非常重要的。通过描述，项目经理可以了解程序员的开发速度，其他程序员使用你的项目时，也可以得到必要信息。具体到描述的书写格式，每一个公司都是不同的，读者需要遵守所在公司的规范。

单击“Finish”按钮，将完成项目的第一次提交。切换到 CVS 操作透视图中，刷新其中的 CVS 服务器，然后将它展开，如图 13-17 所示。

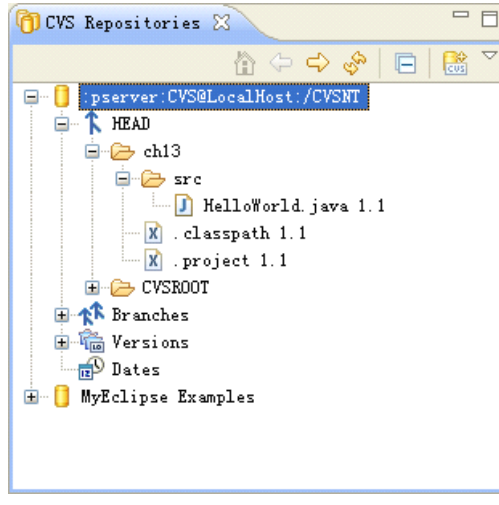


图 13-17　第一次提交结果

从结果中可以看到，已经将“ch13”项目提交到 CVS 服务器中，并且在程序文件后面多出了“1.1”的内容，这就是提交的版本。我们继续再来看一下普通透视图中包资源管理器的视图界面，如图 13-18 所示。

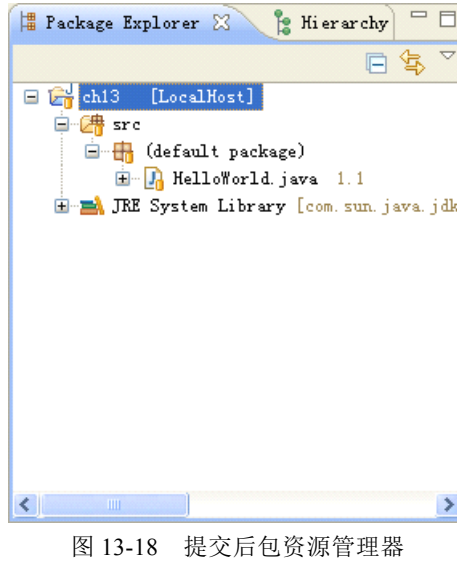


图 13-18　提交后包资源管理器

在包资源管理器中，项目名后会显示该项目提交的服务器，在每一个文件后都会显

示该文件在 CVS 服务器中的版本号。

13.2.5 提交文件

在上一小节中已经完成了共享项目的操作，并且已经完成了第 1 次提交。当项目或者某一个文件发生变化后，我们需要将新版本的程序提交到 CVS 服务器中，从而供其他程序员使用。在本小节中就来学习一下如何在 MyEclipse 中完成提交程序的操作。

首先我们将 HelloWorld 程序的代码修改成如下内容：

```
01  public class HelloWorld {
02      public static void main(String[] args) {
03          System.out.println("HelloWorld 第 3 个版本 CCC");
04      }
05  }
```

然后在包资源管理器中，选中“HelloWorld.java”程序节点，单击鼠标右键，在弹出的菜单中选择“Team”|“Commit”命令，将弹出提交文件界面，在其中输入提交描述后，界面如图 13-19 所示。

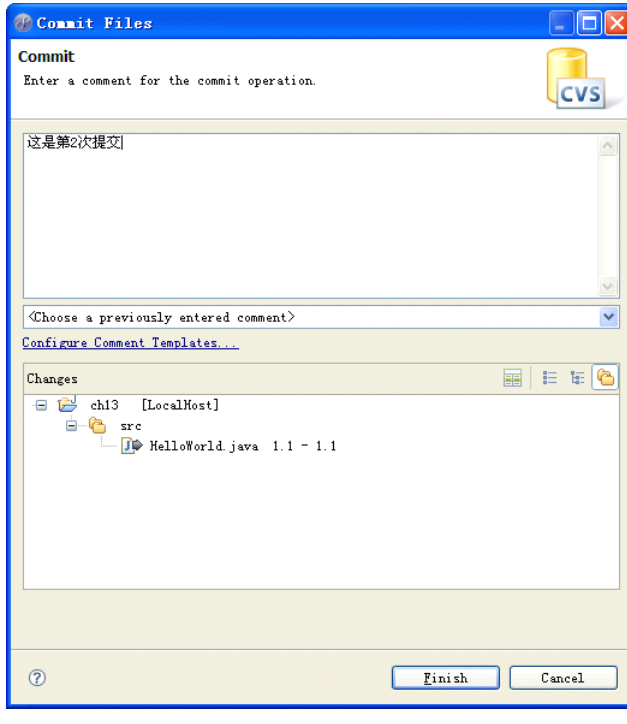


图 13-19 提交文件

单击“Finish”按钮，将完成提交文件的操作。再次展开包资源管理器中的项目，如图 13-20 所示。

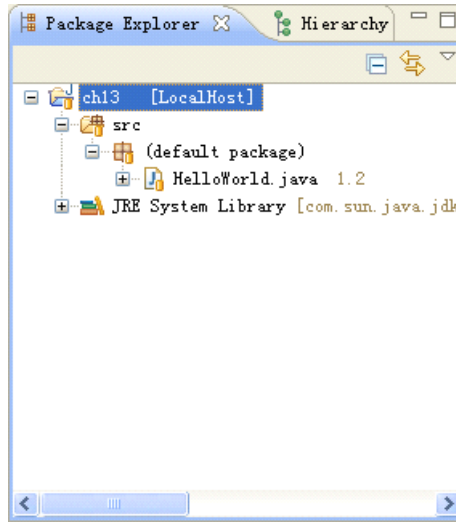


图 13-20　提交后界面

从包资源管理器中可以看到，提交程序后的版本已经变为“1.2”。

技巧：不但可以提交某一个文件，也可以提交整个项目，操作是一样的。通过这种方式可以将整个项目提交，其中所有发生改变的文件都会版本增加，没有改变的文件不会版本增加。在实际开发中，这种操作使用时最多的。

13.2.6　更新文件

当对程序修改后，发现修改的并不正确，需要回到原来的样子，这时候就要在 CVS 服务器中更新文件。例如我们将 HelloWorld 程序的代码修改为：

```
01    public class HelloWorld {
02        public static void main(String[] args) {
03            System.out.println("HelloWorld 第 4 个版本 DDD");
04        }
05    }
```

修改后，发现并不是我们想要的，需要切换到原来版本。这时候就要执行更新文件的操作，在包资源管理器中选择“Team”|“Synchronize with Repository”命令，将弹出比较不同的界面，如图 13-21 所示。

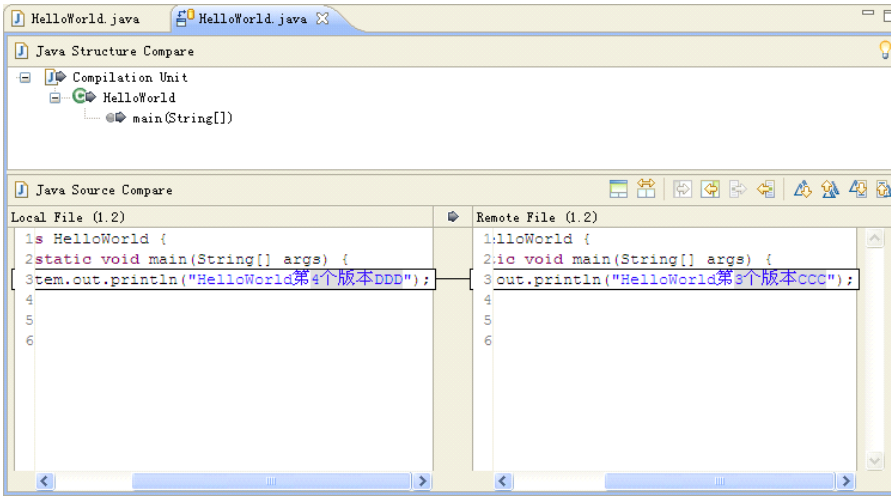


图 13-21　和原版本比较

通过该文件就可以使用当前程序内容和提交到 CVS 服务器的程序相比较，从而得到所有的不同点。单击其中的“  ”按钮和“  ”按钮可以完成更新操作。

13.2.7　检出 CVS 服务器中的项目

程序员不但可以将自己的项目提交到 CVS 服务器中，还可以在 CVS 服务器中检出其他程序员的项目。

在 MyEclipse 中，切换到 CVS 操作透视图中后，在“CVS Repositories”视图界面中选中要检出的项目，例如这里选择前面操作上传的“ch13”项目。单击鼠标右键，在弹出的菜单中有两个检出命令，分别是“Check Out”命令和“Check Out As”命令。

说明：“Check Out”命令表示检出到本地当前工作空间内，在实际开发中，通常使用该命令，因为上传者和下载者通常不是一个程序员，也就是不在同一电脑上。但是现在我们是在本地做实验，所以我们这里选择“Check Out As”命令，它的操作界面是和“Check Out”命令的操作界面非常相似的。

选择“Check Out As”命令后，将弹出修改检出项目名称的界面，如图 13-22 所示。

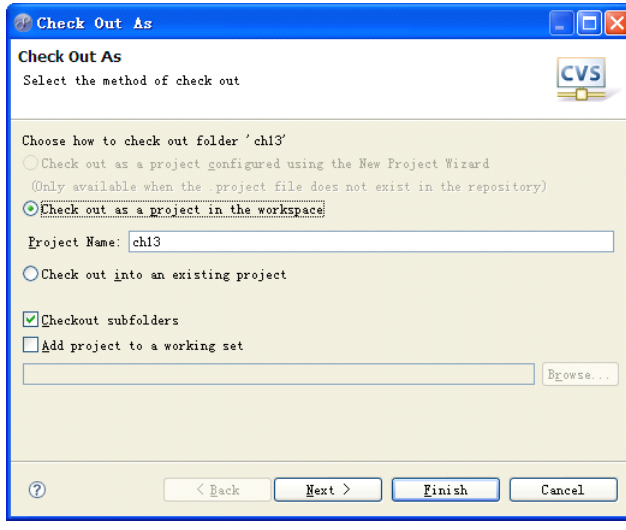


图 13-22　修改项目名称

其中“Check out as a project in the wordspace”选项表示做为工作空间中的项目检出，下面的“Project Name”表示检出项目的名称，这里将它修改为“ch13_2”。“Check out into an existing project”表示检出到一个已经存在的项目中，对于检出文件的操作，通常选中该选项。“Checkout subfolders”表示检出子文件夹，通常保持选中的状态。单击“Next”按钮，将弹出选择工作空间的界面，如图 13-23 所示。

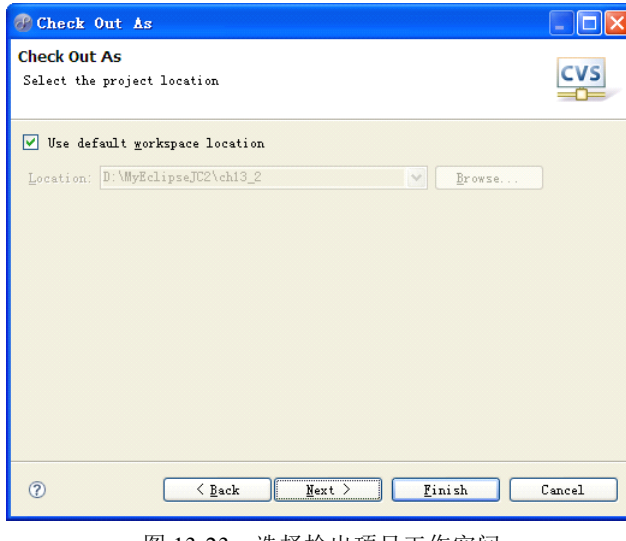


图 13-23　选择检出项目工作空间

其中“Use default workspace location”表示是否使用默认的工作空间，默认是选中的状态。取消选中状态后，可以在“Location”中选择其他工作空间。采用默认值，单击“Next”按钮，将弹出对检出项目进行描述的界面，如图 13-24 所示。

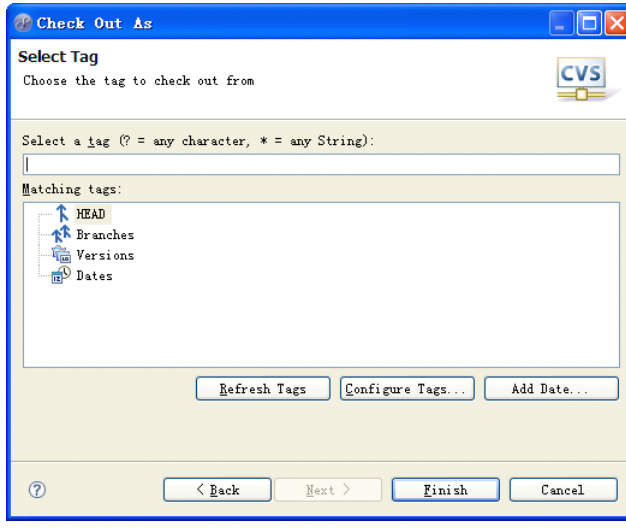


图 13-24　配置检出项目

其中“Refresh Tags”表示刷新标记，“Configure Tags”表示配置标记，“Add Date”表示为检出项目添加一个日期。单击这些按钮，都会进入相应设置的界面，这里就不详细讲解了。可以采用默认值，单击“Finish”按钮，将完成项目的检出。

在 MyEclipse 中切换到默认视图中后，在包资源管理器中可以看到多出了一个项目，如图 13-25 所示。

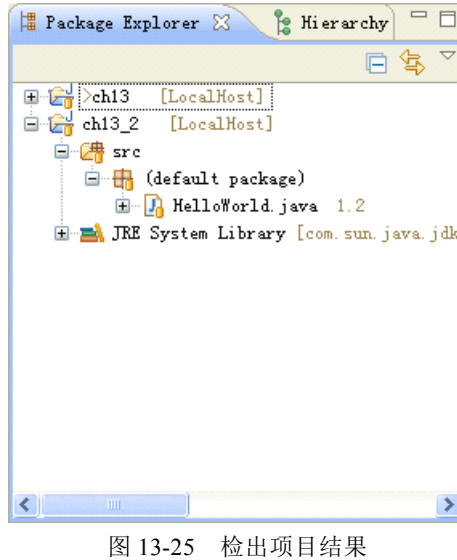


图 13-25　检出项目结果

从包资源管理器可以看到，多出了一个名称为“ch13_2”的项目，这就是从 CVS 服务器中检出的项目。在该项目中也会保留程序的版本，例如 HelloWorld 程序是以“1.2”版本提交的，从而也以该版本显示出来。

13.3 SVN 版本控制

SVN 可以说是 CVS 的“接班人”，它是比 CVS 更强大的版本控制工具。在 SVN 中实现了 CVS 的全部功能，并且修正了 CVS 的缺陷和增加了更强大的功能。但是在 MyEclipse 中并没有集合 SVN 的操作功能，所以如果想在 MyEclipse 中进行 SVN 操作，需要安装相应的插件。

13.3.1 下载和安装 SVN 服务器

SVN 和 CVS 类似，也是由服务器端和客户端组成的，服务器端需要在官方网站下载。如果不采用 MyEclipse 等开发工具，客户端也是需要下载的，但是这里我们采用 MyEclipse 的插件进行客户端操作。在本节中先来学习下载和安装 SVN 服务器。

SVN 的服务器端的官方网站为“http://subversion.tigris.org”，它的下载链接为“http://subversion.tigris.org/project_packages.html”，打开该链接后，页面如图 13-26 所示。

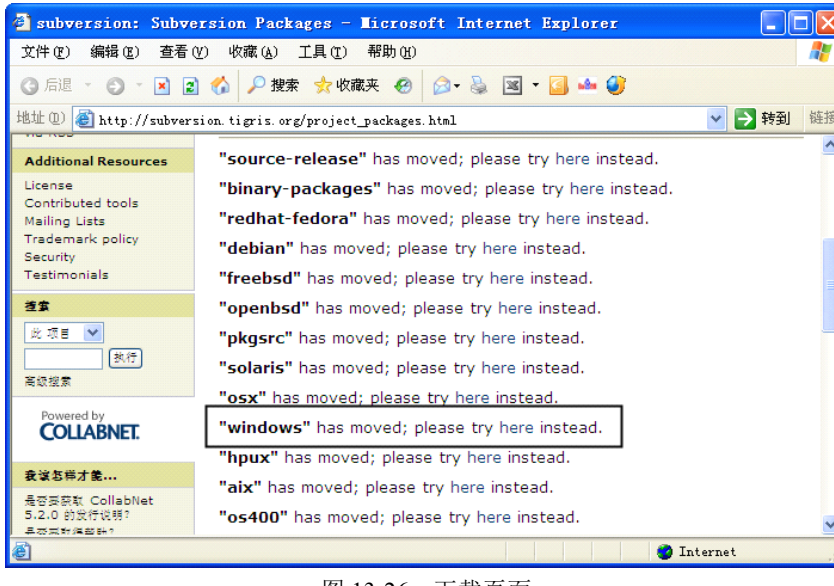


图 13-26 下载页面

在该页面中，会显示 SVN 服务器在不同操作系统下的版本，这里选择 Windows 操作系统的版本，单击后面的“here”超链接，将进入 Windows 版本下载的页面，在其中选择“Tigris.org”超链接，就会进入对应选择版本的下载页面。在其中选择最新的版本，目前最新版本为“1.6.5”版本，单击它对应的超链接，就会开始下载。

下载完成后，就可以进行双击安装。SVN 服务器的安装时非常简单的，几乎都是单击“Next”按钮，中间有一步时选择安装位置，可以采用默认地址，也可以自己指定，选择位置后，单击“Next”按钮，将弹出如图 13-27 所示的界面。

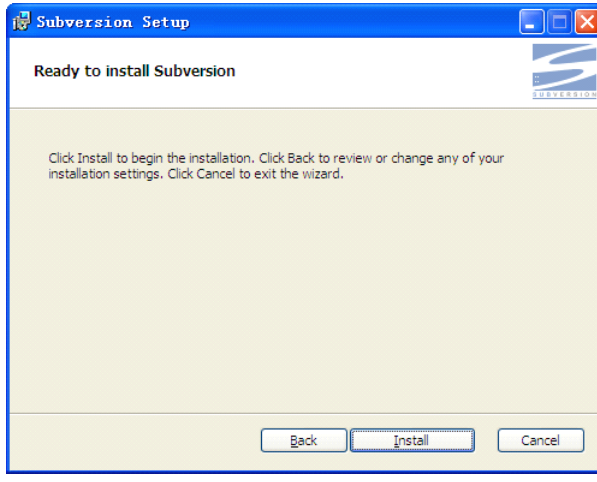


图 13-27　安装 SVN 服务器

单击“Install”按钮，将开始安装 SVN 服务器。稍等片刻后，弹出完成界面，如图 13-28 所示。

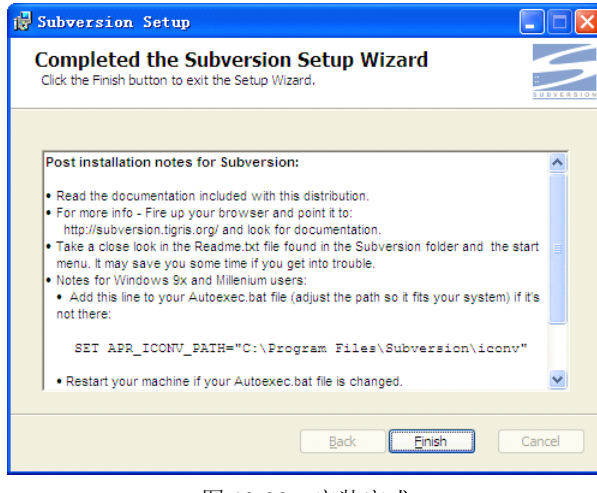


图 13-28　安装完成

单击“Finish”按钮，将完成 SVN 服务器的安装。

13.3.2　安装 SVN 插件

在前面学习 Struts 2 开发时，已经学习了如何安装 Struts 2 插件。在 MyEclipse 中，所有的插件的安装都是非常类似的。在 MyEclipse 的菜单中，选择“Help”|“Software update”|“Add/Remove Software”命令，将弹出选择安装插件的界面，在其中“Add”按钮，将弹出增加插件的界面。在增加插件界面中，单击“Add Site”按钮，将弹出连接下载地址的操作界面，在其中输入下载 SVN 插件的 URL，如图 13-29 所示。

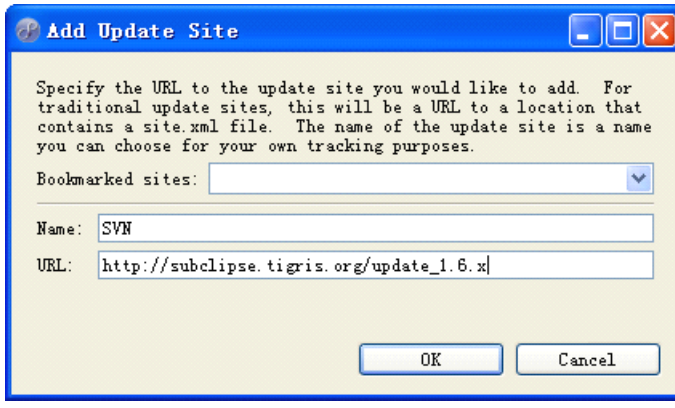


图 13-29　连接 SVN 插件 URL

SVN 插件的安装 URL 为“http://subclipse.tigris.org/update_1.6.x”，输入该 URL 后，单击“OK”按钮，将完成 SVN 插件的连接，然后在增加和删除插件的界面中展开下载的插件，如图 13-30 所示。

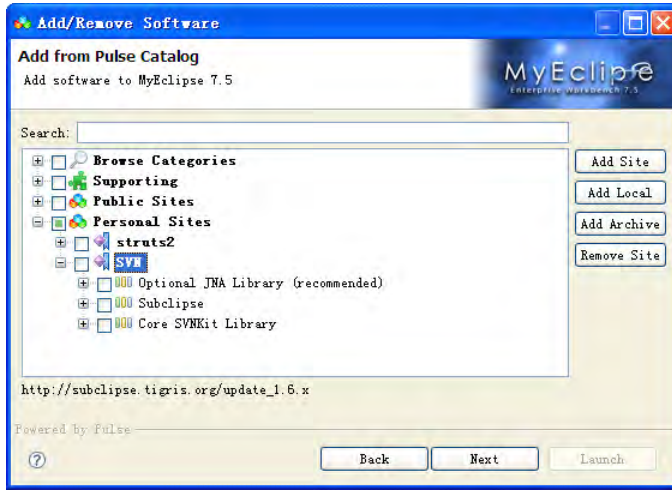


图 13-30　查找到 SVN 插件

选择“SVN”下的三个选项，单击“Next”按钮，将进入安装 SVN 插件的界面，在其中单击“Apply”按钮将开始安装这些插件。后面的操作就是和安装 Struts 2 插件完成相同的，这里就不再详细讲解。

13.3.3　创建服务器资源库

安装 SVN 服务器后，就可以创建服务器资源库。该操作一定要在系统中完成，然后在 MyEclipse 中引用，在 MyEclipse 是不能够直接完成了，因为其他并没有集成服务器的功能。

在创建服务器资源库之前，同样需要在系统中首先创建一个空文件夹，例如在 D 盘下创建“SVN”文件夹。然后在系统开始菜单中，选择命令，输入“cmd”命令，将进

入命令操作界面，在其中输入“svnadmin create d:\SVN”命令，如图 13-31 所示。

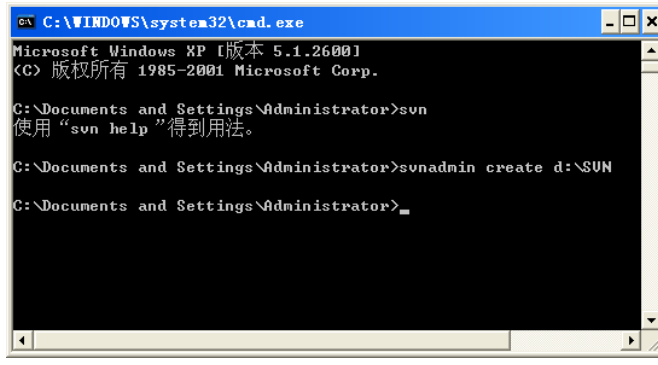


图 13-31　创建资源库

按下“Enter”回车键后，将完成该命令的执行。打开 D 盘下的“SVN”文件夹可以发现里面初始化了一些文件，这就是资源库的相关文件。

创建服务器资源库后，通常还需要对它进行比较的配置。在默认情况下，登录 SVN 服务器是匿名的形式登录的，这对于服务器而言是非常不安全的，所以我们要定义登录用户名和密码。首先打开资源库文件夹下的“conf\svnserve.conf”文件，因为我们将资源库放在 D 盘的“SVN”文件夹下，所以完整路径为“D:\SVN\conf\svnserve.conf”。

打开文件后，如果其中“anon-access = read”和“auth-access = write”行前面没有加#注释，就为它们加上#注释。然后将其中的“password-db = passwd”行前面的#注释去掉，通过该操作可以时同目录下的密码文件生效。

接下来使用记事本等文本工具打开同目录下的“passwd”文件，笔者计算机上它的完整路径为“D:\SVN\conf\passwd”。在该文件的最后加入如下内容：

Tom = 456123

其中等号左边的“Tom”就表示用户名，等号右边的“456123”就表示密码。

注意：创建服务器资源库后，要在命令控制台中输入“svnserve –d –r d:/SVN”命令，并且不要关闭窗口，这样才能够保证 SVN 服务器启动状态。

13.3.4　在 MyEclipse 中引入服务器资源库

在 MyEclipse 中安装 SVN 插件成功后，将在其中出现 SVN 透视图。在 MyEclipse 菜单中，选择“Window”|“Open Perspective”|“Other”命令，将弹出选择透视图的界面，如图 13-32 所示。

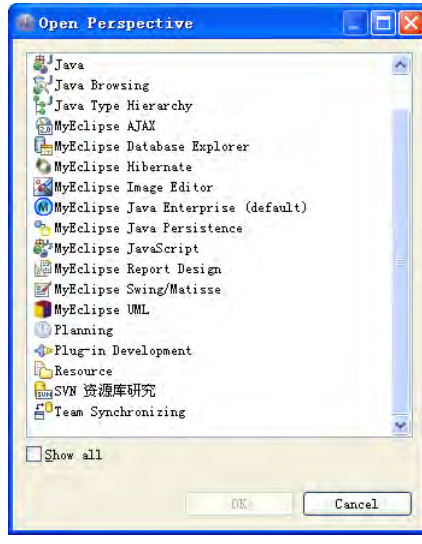


图 13-32　选择 SVN 透视图

可以看到，其中多出了一个“SVN 资源库研究”的透视图，选择该透视图，单击“OK”按钮，将切换到该透视图中，如图 13-33 所示。

说明：自动安装 SVN 插件时，会自动安装对应语言的版本，例如这里显示的是中文，在后面的操作步骤中，大部分也是中文。这和其他功能是有很大区别的。

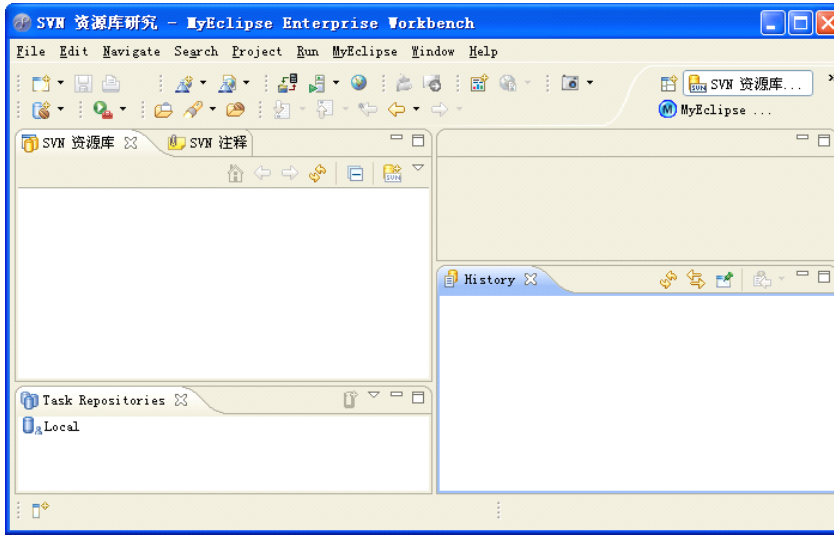


图 13-33　SVN 透视图效果

其中最重要的就是“SVN 资源库”视图界面，在其中空白处，单击鼠标右键，在弹出的菜单中选择“新建”|“资源库位置”命令，将弹出添加 SVN 资源库的界面，如图 13-34 所示。

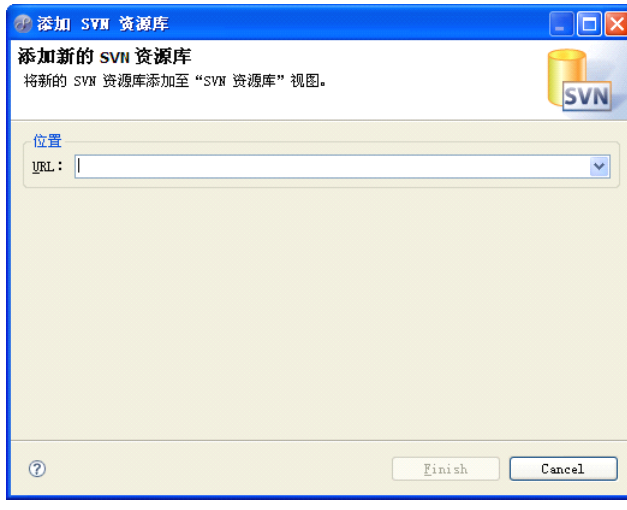


图 13-34　添加 SVN 资源库

其中 URL 表示的就是 SVN 资源库所在的地址，因为我们在本地做实验，所以这里可以填写“svn://localhost”，最后单击“Finish”按钮，将完成引入操作，从而在“SVN 资源库”视图界面中多出一个 SVN 资源库。

13.3.5　向 SVN 服务器中提交

创建 SVN 服务器资源库后，我们就可以向资源库中提交项目或者程序文件。切换到 MyEclipse 的默认透视图下，在包资源管理器中创建一个新项目，项目的名称为“ch13_SVN”，然后在该项目中创建 Java 程序，其代码内容为：

```
01  public class HelloWorld {
02      public static void main(String[] args) {
03          System.out.println("SVN 服务器第 1 次提交 AAA");
04      }
05  }
```

在讲解 CVS 时，主要是进行的程序文件操作，这里讲解 SVN 时就主要以项目进行操作，它们的操作都是一样的。在包资源管理器中，选中“ch13_SVN”项目，单击鼠标右键，在弹出的菜单中选择“Team”|“Share Project”命令，将弹出选择版本控制工具的界面，如图 13-35 所示。

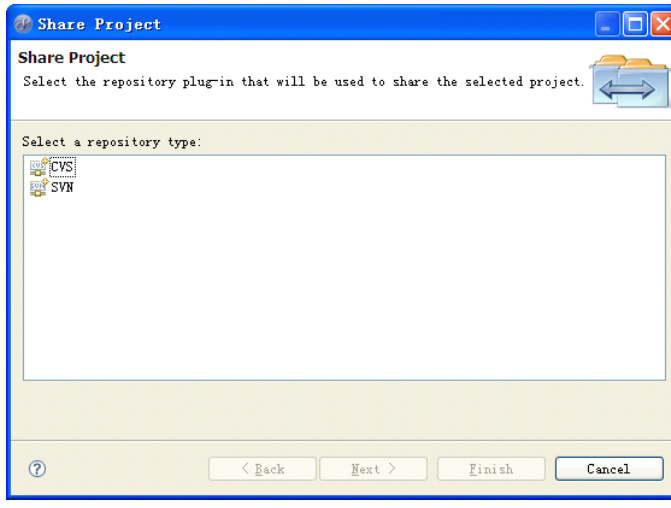


图 13-35　选择版本控制工具

因为目前 MyEclipse 中并不只 CVS 这一种版本控制工具，所以会出现该选择工具的界面。在其中选择“SVN”选项，单击“Next”按钮，将进入选择资源库的界面，如图 13-36 所示。

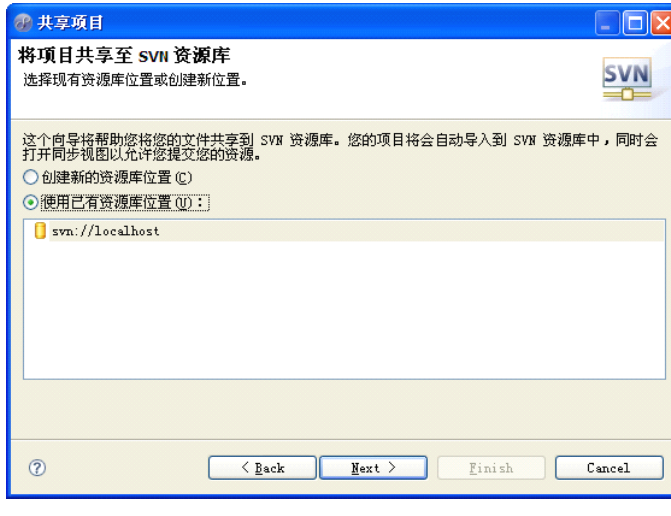


图 13-36　选择资源库

在该界面中，可以选择创建新的资源库，也可以选择使用已有的资源库。当选择第 2 个使用已有资源库的选项后，还可以在下面选择具体的哪一个资源库。选中后，单击“Next”按钮，将弹出设置文件夹名的界面，如图 13-37 所示。

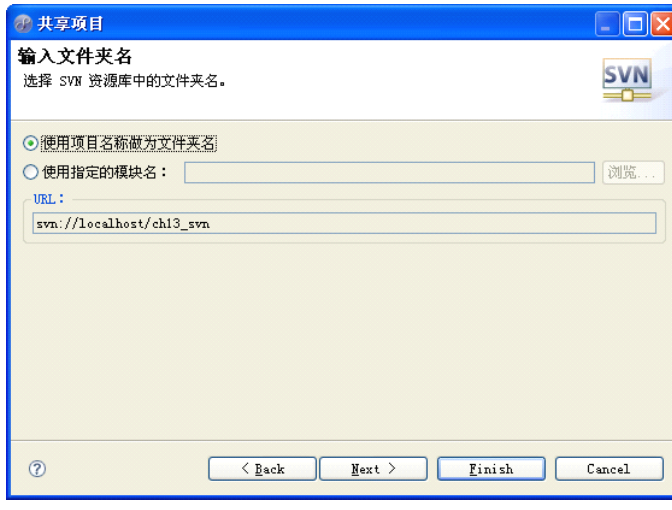


图 13-37　设置文件夹名

在该界面中，我们通常选择使用项目名作为文件夹的名称，在“URL”的文本框中海会显示该项目在 SVN 服务器中的访问地址。单击“Next”按钮，将弹出输入提交注释的界面，如图 13-38 所示。

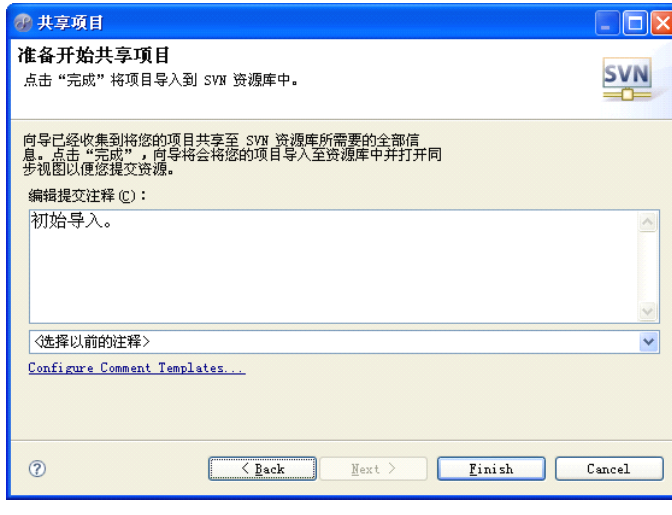


图 13-38　输入提交注释

每一种版本控制工具中都会有这一步的，这里我们输入“这是向 SVN 服务器中第 1 次提交”。单击“Finish”按钮，将完成向 SVN 服务器的第 1 次提交。

提交后，在 MyEclipse 菜单中，选择“Show View”|“Other”命令，将弹出选择视图界面的窗口，如图 13-39 所示。

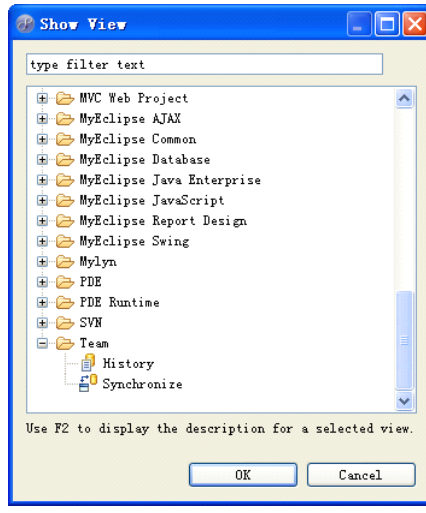


图 13-39　选择视图界面

在其中选择“Team”|“Synchronize”选项，单击“OK”按钮，将打开该视图界面，如图 13-40 所示。

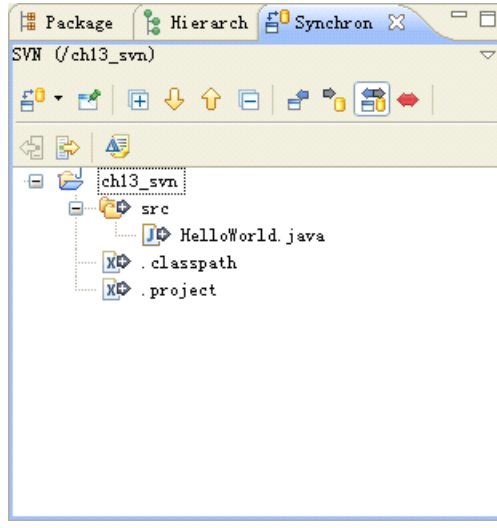


图 13-40　提交项目后结果

该视图界面就是用于显示 SVN 服务器中保存的项目，从其中可以看到前面创建“ch13_svn”项目已经被提交到 SVN 服务器中，从而可以供其他程序员下载。

说明：第 1 次提交项目是比较复杂的，需要首先在 SVN 服务器中创建对应的文件夹。在以后的提交后，就比较简单，只要在右键弹出菜单中选择“Team”|“提交”命令就可以完成新版本的提交。

13.3.6　从 SVN 服务器中检出

向 SVN 服务器中提交项目后，其他程序员就可以将该项目检出。这里我们重新启动一个 MyEclipse，并切换到不同的工作空间下，这样来模拟另一个程序员。

在 MyEclipse 中，首先切换到“SVN 资源库研究”透视图下，同样需要在其中引入服务器资源库。打开该服务器资源库，可以发现其中多出了一个名称为“ch13_svn”的项目，选中该项目，单击鼠标右键，在弹出的菜单中选择“检出为”命令，将弹出检出操作的界面，如图 13-41 所示。

图 13-41　从 SVN 检出

在其中选中“做为工作空间中的项目检出”的选项，保持项目的名称为“ch13_svn”，单击“Next”按钮，将弹出选择项目位置的界面，如图 13-42 所示。

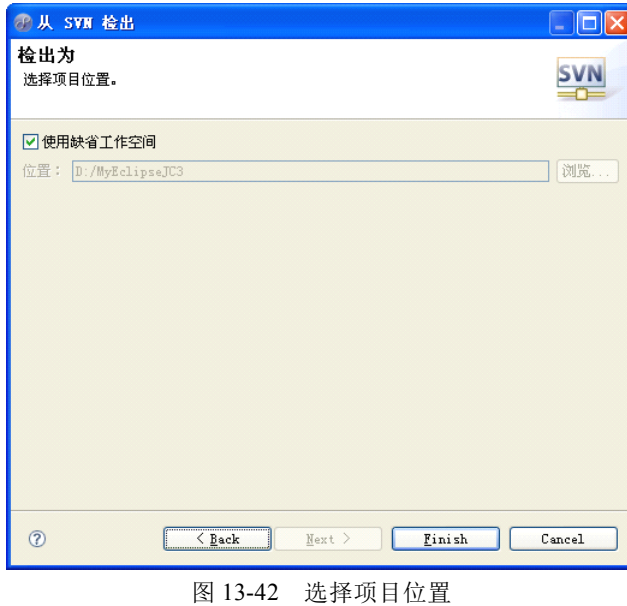


图 13-42　选择项目位置

通过该界面可以设置是否将检出的项目放置在默认工作空间内，通常是保持默认的选中状态的。单击“Finish”按钮，将把 SVN 服务器中的项目检出到当前工作空间中。

将 MyEclipse 切换到默认透视图中，在包资源管理器中展开“ch13_svn”项目，如

图 13-43 所示。

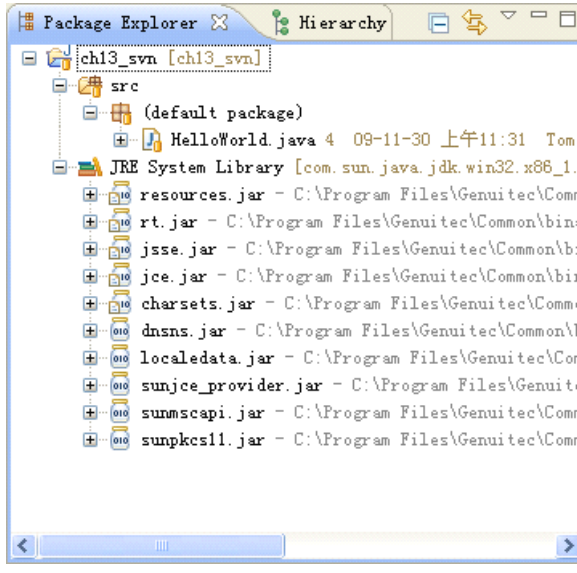


图 13-43　检出结果

在其中可以看到在项目名后将出现对应 SVN 服务器中的文件夹名称。在其中的程序文件后，将出现该文件的 SVN 版本、提交时间和提交用户。

第14章　UML建模

UML 的中文名称叫统一建模语言，它是一种对软件密集系统进行可视化建模的编程语言。随着软件的发展，UML 越来越重要。目前 UML 的开发工具有很多，例如 Rational Rose、Borland Together 等，它们大部分都是商业软件，也就是收费的。开源的 UML 开发工具比较少，最有名的是 ArgoUML， MyEclipse 中集成的 UML 建模工具就是基于 ArgoUML 工具开发的。在本章中将重要来讲解 MyEclipse 中进行 UML 建模操作。

14.1　创建 UML 模型仓库

在本节中通过一个最简单的例子来演示在 MyEclipse 中如何创建 UML 模型。在 MyEclipse 中，UML 模型也是被看做是一般的程序文件的，所以它也要放在某一项目下。首先在开发之前首先创建一个名称为“ch14”的项目，然后在该项目下创建“uml”包。

在 MyEclipse 中进行 UML 开发，第一步并不是直接创建 UML 模型，而是首先要创建一个 UML 模型仓库，所有的 UML 模型将放在该模型仓库中。

在包资源管理器中，选中“uml”包，单击鼠标右键，在弹出的菜单中选择“New”|“UML1 Model”命令，将弹出创建 UML 模型仓库的界面，如图 14-1 所示。

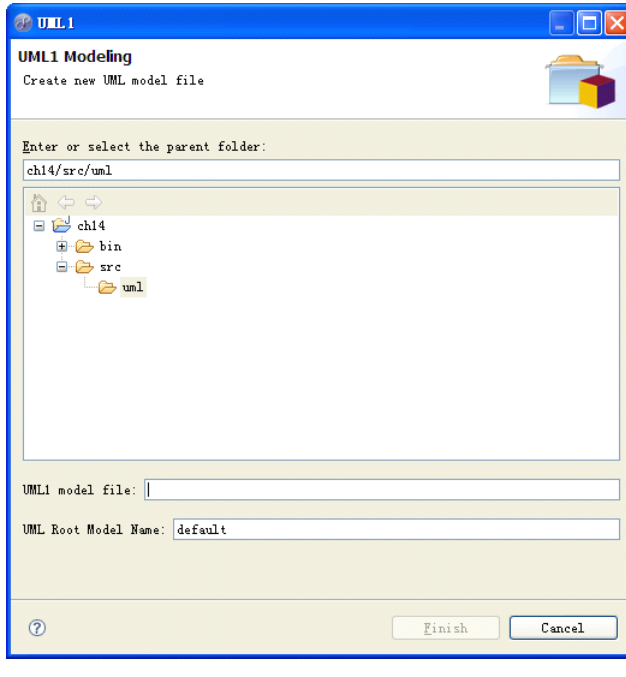


图 14-1　创建 UML 模型仓库

其中“UML1 model file”表示模型仓库的名称，这里输入“firstUML”，其他保持默认值，单击“Finish”按钮，将完成 UML 模型仓库文件的创建。并且弹出一个是否切换透视图的对象化，这里单击“Yes”按钮，将切换到 UML 透视图中。并且在编辑区中还会打开 firstUML 模型仓库文件，界面如图 14-2 所示。

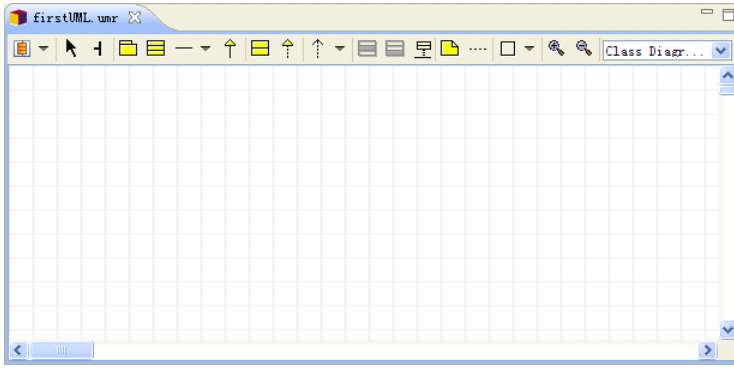


图 14-2　开发界面

在该文件界面中，就可以创建 UML 模型，主要通过上面的工具按钮来完成，在后面操作中将详细讲解它们。

注意：在 MyEclipse 中虽然显示 UML 模型仓库的扩展名为“.umr”，其实它是一个包文件，完全可以使用 WinRAR 等软件打开。

14.2　用例图

开发完 UML 模型仓库后，也就是创建模型仓库文件后，就可以在该文件中创建 UML 模型图形。UML 中的模型图形有很多种，包括用例图、类图、状态图、时序图等很多种，在本节中将对其中开发中经常用到的图进行操作学习。

用例图是在系统分析阶段经常要绘制的 UML 图，在其中通常标出系统中有哪些用户，这些用户能够完成哪些功能。在用例图中可以包含 6 个元素，分别是参与者、用例、关联关系、包含关系、扩展关系和泛化关系，在一个用例图中是不必须全部包含这六种元素。

14.2.1　参与者

在 MyEclipse 的模型仓库界面中，选择工具栏第一个下三角菜单中的“New Use Case Diagram”选项，则当前工具栏将变为进行用例图创建的形式。在其中单击“  ”按钮，然后在画布中单击，将创建一个用例图中的参与者，并在界面中显示一个参与者图标。双击参与者图标下空白处，可以添加参与者的名称，这里添加“游客”。如图 14-3 所示。

说明：为 UML 图添加例如名称这样的信息，通常是双击添加，也可以在

“Properties”视图界面中添加。

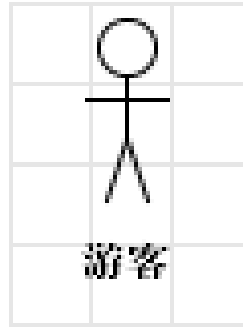


图 14-3　创建参与者

14.2.2　用例

开发完参与者后，就可以开发用例，用例就表示在系统中能够完成哪些功能。单击工具栏中的“  ”按钮，然后在画布中再次单击，将创建一个用例，双击图标可以添加用例的内容。使用这种方式，这里添加两个用例，如图 14-4 所示。

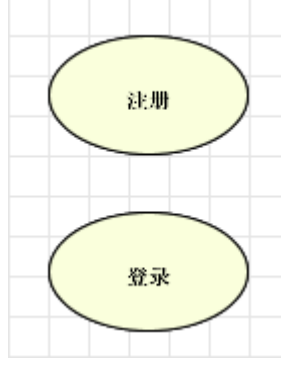


图 14-4　创建用例

14.2.3　关联关系

参与者和用例之间的关系称之为关联关系。在工具栏中单击“  ”按钮后的下三角选择“uni Association”选项，然后在下面的界面中首先选中参与者，然后拖动鼠标在选中用例，这样就添加了一条关联关系。为两个用例都建立关联关系后，如图 14-5 所示。

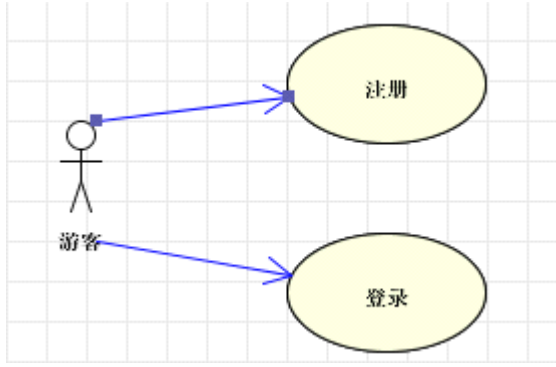


图 14-5　创建关联关系

建立关联关系后，一个基本的用例图就开发完成了。从用例图中可以知道目前系统中有一种用户是“游客”，它能够完成注册和登录的功能。

说明：用例图除了上面三种元素外，包括包含关系、扩展关系和泛化关系。包含关系用于用例和用例之间，表示一个用例包含另一个用例的全部功能，例如用户用例包含游客用例。扩展关系也是用在用例之间，表示对用例的扩展，例如发表主题时，用户积分会增加，则积分增加用例和发表主题用例之间就是扩展关系。返回关系表示将用例功能更详细化，例如查看文章用例，就可以泛化为在首页查看文章和个人博客中查看文章两个范例。

14.3　类图

类图是用来描述类相关的信息，例如类中的变量和方法、类与类之间的关系，通过它可以显示类的静态结构。类图中包含多种元素，分别是类、接口、协作、依赖关系、泛化关系和关联关系等。

注意：在不同的 UML 图中经常会出现相同的元素名称，例如泛化关系，需要注意的是它们表示的含义是完全不同的，一定要注意区分。

14.3.1　类

在工具栏中选择“New Class Diagram”选项，当前工具栏将变为创建类图的工具栏。单击其中的“  ”按钮，然后在画布中再次单击，将创建一个表示 UML 图中类的图标。类图标分为三层，依次可以向其中输入类名、属性名和方法名。通过双击可以填写相关名称，属性和方法名可以填写多个。例如这里创建一个学生类，如图 14-6 所示。

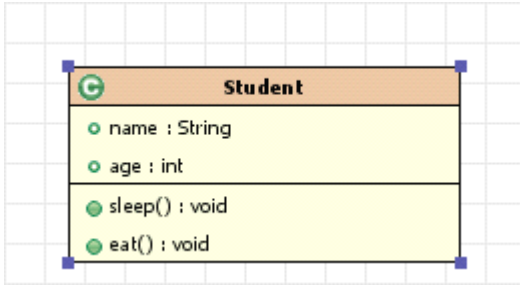


图 14-6　类

通过该类图标，程序员就可以知道在项目中要创建一个 Student 学生类，在该类中定义 name 姓名和 age 年龄属性，以及 sleep 睡觉和 eat 吃饭方法。

14.3.2　接口

接口也是类图中必不可少的一种元素。在接口中规定了一类对象行为的描述，但是并没有给出具体的实现。在接口中只包含操作，并不包含属性。在类图工具栏中单击“  ”按钮，然后在画布中再次单击，将创建一个 UML 图中的接口图标。接口图标分为两层，上层是用来填写接口名，下层用来定义接口中有哪些方法。通过双击操作同样能够输入内容，例如这里创建一个表示人的接口，如图 14-7 所示。

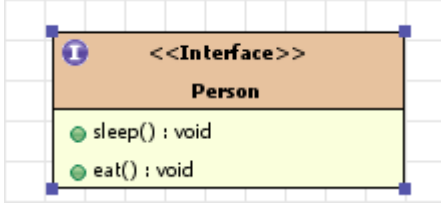


图 14-7　接口

这样就创建了一个表示人的接口，其中定义了睡觉和吃饭两个抽象方法。类与接口之间最常见的关系就是实现，单击工具栏中的“  ”按钮，然后从类拖动到接口，就建立了它们之间的实现关系，如图 14-8 所示。

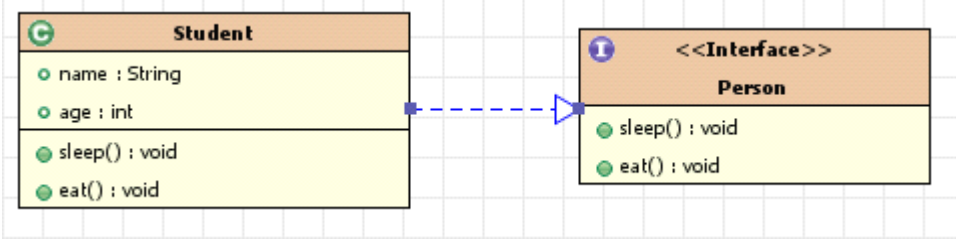


图 14-8　类实现接口

通过这种虚线闭合箭头的形式，就可以知道 Student 学生类将实现 Person 接口。

技巧：实现关系还有简略的方式，就是将接口使用圆圈图标，然后使用线段连接

类和接口就表示实现关系。但是目前这种 UML 图已经很少见了。

14.3.3 类之间的依赖关系

依赖关系是指类模型之间语义上的关系，其中包含四种依赖类型，分别是使用依赖、抽象依赖、授权依赖和绑定依赖。依赖关系通过虚线非闭合箭头表示，单击工具栏中的“  ”按钮中的倒三角可以完成，如图 14-9 所示。

图 14-9 依赖关系

该依赖关系 UML 图就表示在 Car 汽车类中将使用依赖于 Tyre 轮胎类。具体其他依赖关系，读者可以参考 UML 相关资料学习，这里就不再一一讲解。

14.3.4 类之间泛化关系

UML 中的类之间泛化关系对应的就是程序中的继承，它表示父类和子类之间的关系。在 UML 图中，使用实线闭合箭头从子类指向父类，从而表示他们之间的泛化关系，单击“  ”按钮可以完成，如图 14-10 所示。

图 14-10 泛化关系

通过该 UML 图就可以表示 Car 汽车类将继承 Vehicle 交通工具类。

说明：一个类的图标可以被多个类图标泛化，也是 Java 程序中的一个父类拥有多个子类，例如飞机类也可以和交通工具类建立泛化关系。因为 UML 不只能表示 Java 程序，当表示其他编程语言时，还可能出现一个类泛化多个类的情况。

14.3.5 类之间关联关系

严格的来说，关联关系并不是类之间的关系，而是类实例或者对象之间的关系，但是因为 UML 图中通过使用类的图标建立关系，所以也可以称之为类之间的关联关系。

关联关系是通过一个线段表示的，但是如果仅仅使用它是不能具体看出之间的关系的，所以需要为它加入一些修饰，包括名称、角色、多重性、聚合、组合和导航性 。单

击工具栏中的“ — ”按钮，可以完成关联关系的创建，通过“Properties”视图界面可以为关联关系加入修饰，如图 14-11 所示。

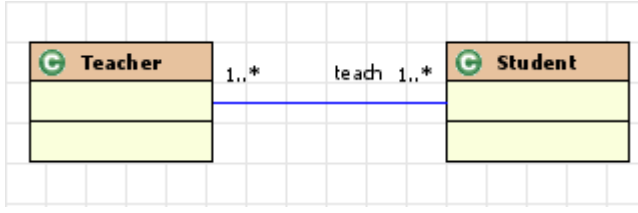


图 14-11　关联关系

该 UML 图中为 Teacher 老师类和 Student 学生类建立了关联关系，它们之间是 teach 教学的关系。类图标旁边的数字表示类的多重性，这里表示一个老师可以教多个学生，一个学生可以被多个老师教。

14.4　UML 和 Java 程序之间的转换

如果仅仅是通过 UML 图来表示程序的结构或者程序之间的关系，那 UML 也不会这种重要。UML 强大之处在于它能够和 Java 程序进行相互转换，也就是我们绘制完 UML 类图后，能够自动生成 Java 程序。从 UML 类图到 Java 程序被称为正向工程，从 Java 程序到 UML 类图被称为反向工程。

14.4.1　正向工程

正向工程就是建立 UML 类图后，自动生成和它对应的 Java 程序，在该 Java 程序中就会具有 UML 类图中定义了属性、方法等信息。

注意：自动生成是有前提条件的，必须在绘制 UML 类图时就要使用规范的形式。因为在绘制 UML 时并没有严格的规范，例如定义方法时不填写返回值也是没有问题的，但是当转换 Java 程序时就会出现问题。

这里我们先创建一个 UML 模型仓库，然后在该模型仓库中定义表示 Student 学生类的图标，如图 14-12 所示。

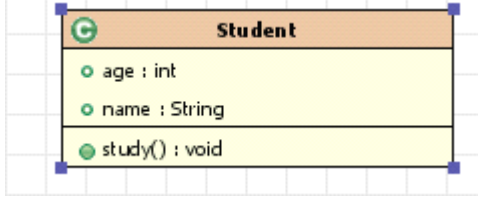


图 14-12　创建示例类

创建 UML 模型后，不要忘记保存模型仓库文件。然后就是执行正向工程的操作，在 MyEclipse 中选择“UML”|“Generate Java”命令，将弹出创建 Java 程序的界面，如

图 14-13 所示。

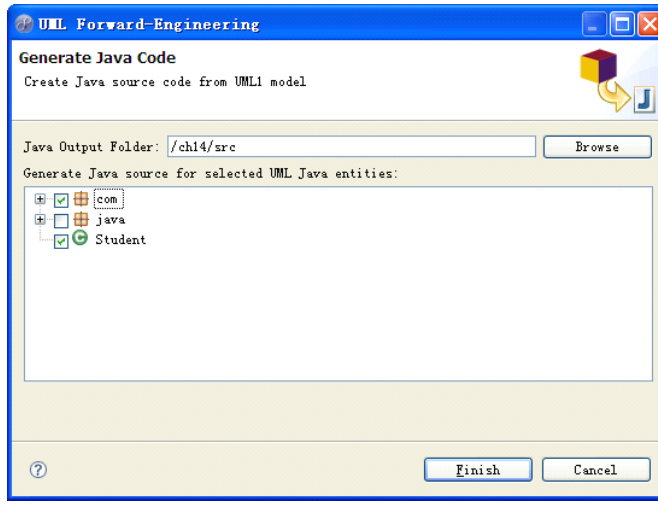


图 14-13　正向工程

其中“Java Output Folder”表示 Java 程序的输出目录，这里要选择在 src 目录下。在下面的文本域中将列出可以创建哪些 Java 程序，如果 UML 图中有多个类或者接口，都将列在这里。由于目前只创建了一个 Student 学生类的图标，所以这里只有一个 Student 节点，选中它，单击“Finish”按钮，将完成 Java 程序的创建。

双击打开自动生成的“Student.java”程序，在编辑区中将显示它的代码，如下所示。

```
01    import java.lang.String;
02    public class Student {
03        public int age;
04        public String name;
05        public void study() {
06        }
07    }
```

从代码中可以看到自动生成了 age 年龄和 name 姓名属性，并且也生成了表示 study 学习的空方法，需要手动在方法中加入功能代码。自动生成的代码也是有不合适的地方的，例如第 1 行的代码，对于“java.lang.*”包的类是不需要导入的，但是自动生成时会加上。

14.4.2　反向工程

有些读者可能有这样的疑问：UML 模型的作用就是让程序员参考进行开发，那为什么开发完 Java 程序后，还需要生成 UML 模型呢？

这是因为通过 UML 模型能够很清楚的看到程序之间的关系，这在大型项目中是非常重要的。当项目中有很多个程序时，并且程序之间关系复杂，通过读程序代码是非常麻烦的，这时候就可以将它们生成 UML 模型，通过它们来看程序之间的关系。

这里我们首先创建一个非常简单的 Java 类程序，通过它可以演示如何生成 UML 类图，它的代码为：

```
01  package com;
02  public class Teacher {
03      private String name;
04      private int age;
05      public String getName() {
06          return name;
07      }
08      public void setName(String name) {
09          this.name = name;
10      }
11      public int getAge() {
12          return age;
13      }
14      public void setAge(int age) {
15          this.age = age;
16      }
17  }
```

开发完程序后，然后回到UML模型仓库文件中。在MyEclipse的菜单中，选择“UML”|“Reverse Engineer UML from Java”命令，将弹出选择Java程序的界面，如图14-14所示。

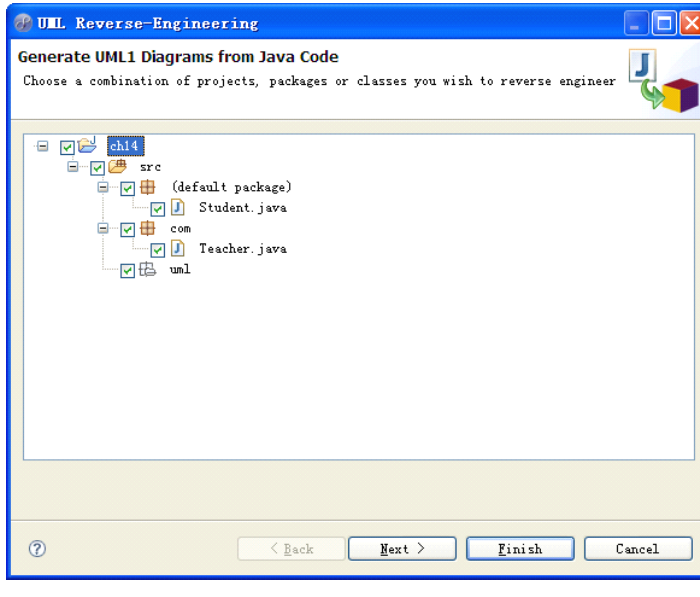


图14-14　选择Java程序

这里在其中只选择“Teacher.java”程序节点，单击“Next”按钮，将弹出设置UML类图的界面，如图14-15所示。

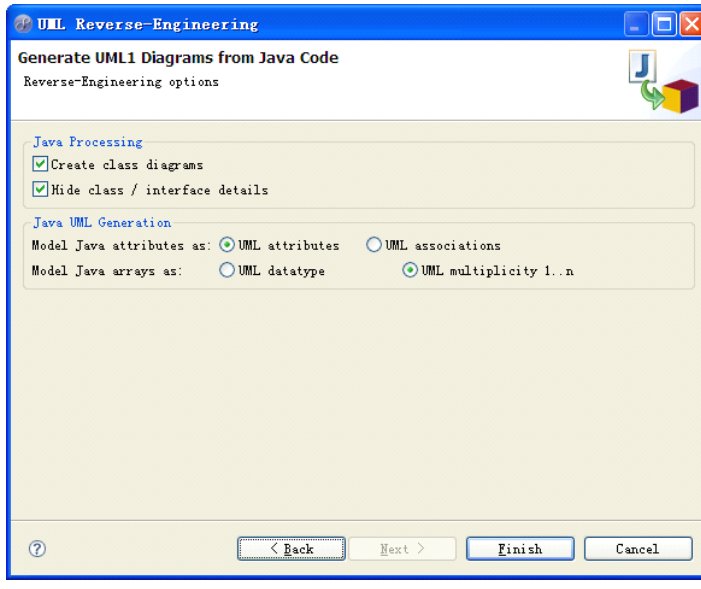


图 14-15　设置 UML 类图

界面中“Java Processing”表示 Java 处理的设置栏，其中“Creat class diagrams”表示是否生成类图，保持默认选中状态；“Hide class/interface details”表示是否隐藏类或者接口详细内容，通常取消该选项选中状态，显示详细内容。

在生成 UML 设置的“Java UML Generation”栏中，“UML attributes”选项表示将 Java 属性建模为 UML 属性，“UML associations”选项表示将 Java 属性建模为 UML 关联。“UML datatype”表示将 Java 数组建模为 UML 数据类型，“UML multiplicity 1..n"表示将 Java 数组建模为 UML 一对多关系。这些选项通常采用默认的选项就可以，单击“Finish”按钮，将完成反向工程操作，片刻后将弹出对话框，如图 14-16 所示。

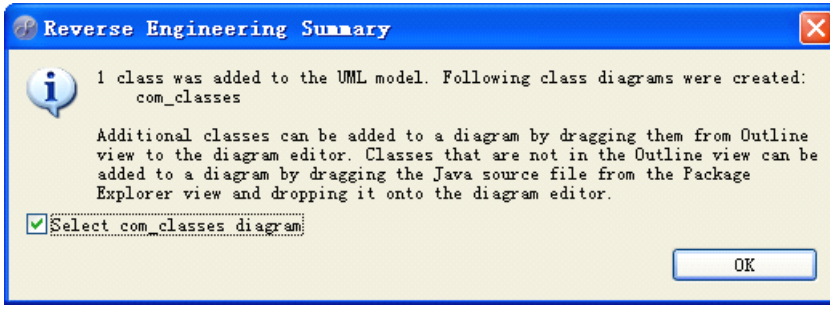


图 14-16　确定对话框

保持该对话框中选项默认的选中状态，单击“OK”按钮，将完成所有操作，在 UML 模型仓库界面中将显示创建的 Teacher 老师类图，如图 14-17 所示。

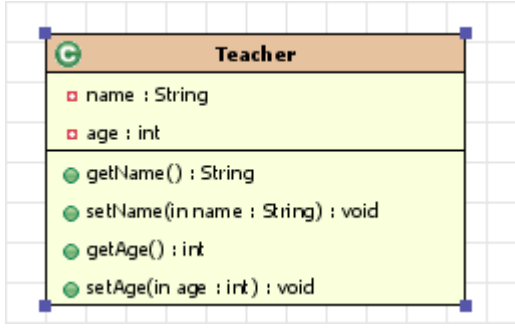


图 14-17　反向工程生成 UML 模型

从类图中可以看到，将程序中的类名、属性和方法都定义到了 Teacher 类图中。如果在设置 UML 类图界面中，保持默认的“Hide class/interface details”选项选中状态，这时候将只显示类名。

技巧：本小节中讲解的方式最实用与对多个程序同时生成 UML 模型的情况，如果仅仅对数量有限的程序生成，还有一个更简单的方式，那就是直接将程序拖动到 UML 模型仓库画布中，将执行弹出设置 UML 类图的界面。

14.5　状态图

状态图是一种系统分析时经常用到的 UML 图，通过它可以跟踪某一对象变化的动态行为。例如网上商店中的订单对象，当用户购买后，就生产订单，然后该订单就会经过销售部门、仓库部门、财务部门等，在每一个部门中都会产生一个新状态，例如已确定、已发货、已收钱等。

14.5.1　开始状态和终止状态

在 MyEclipse 的 UML 模型仓库界面中，单击工具栏中第一个按钮后的倒三角，在其中选择“New Statechart Diagram”选项，就会切换到状态图开发的工具栏。

状态图中最固定的状态，那就是开始状态和终止状态。在工具栏中使用“  ”按钮可以创建开始状态，使用“  ”按钮可以创建终止状态，创建的图标如图 14-18 所示。

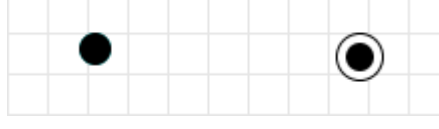


图 14-18　开始和终止状态

其中实心黑点就是开始状态，圆环黑点就是终止状态，这两个状态是固定的，在这

两个状态之间可以加入其它自定义的状态。

14.5.2 增加状态

开发完开始状态和终止状态后，就可以在其中增加自定义的状态。在工具栏中使用“  ”按钮，可以增加一个状态，单击上面的空白处可以添加状态的内容。例如这里我们增加确认、发货和回款三种状态，如图 14-19 所示。

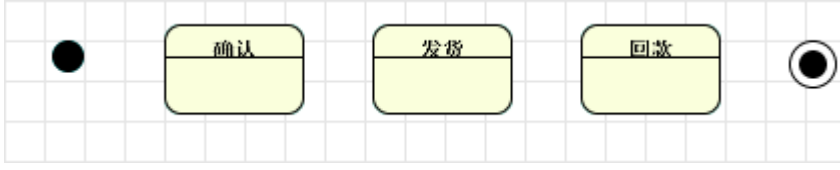


图 14-19 增加状态

14.5.3 连接状态

在状态图中创建状态后，需要将所有的状态连接起来，从而知道状态的执行顺序。在工具栏中使用“  ”按钮，可以连接两个状态，在该图标的上面还可以给出状态转换的条件，如图 14-20 所示。

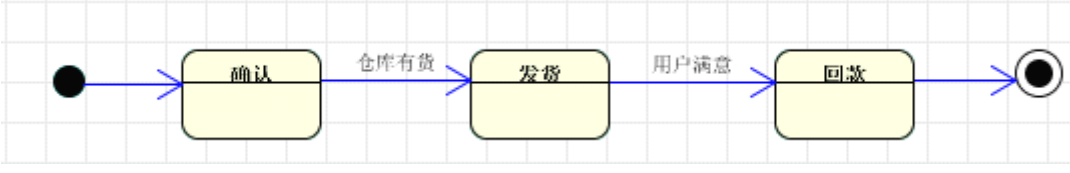


图 14-20 连接状态

当销售部门确认订单后，就可以将订单提交给仓库部分，如果仓库中有货，则订单就进入发货状态。用户收到商品后，支付商品所用的钱，则订单就进入回款状态。

14.6 活动图

活动图是 UML 技术中的一个概念，它和开发中经常说的流程图是非常类似的。当接触 UML 的读者经常会混淆状态图和活动图的。活动图是指活动间的控制流，是内部处理驱动的流程，而状态图是指某一对象不同状态间的控制流，主要是由于外部事件驱动的流程。

14.6.1 增加动作状态

在 MyEclipse 的 UML 模型仓库界面中，单击工具栏中第一个按钮后的倒三角，在其中选择“New Activity Diagram”选项，就会切换到状态图开发的工具栏。

在活动图中最基本的就是初态和终态，它们是和状态图中的开始状态和终止状态非常相似的，这里就不多介绍，我们直接学习如何增加动作状态。在活动图工具栏中，使用“  ”按钮可以增加一个动作状态，双击其中的空白处可以填写状态内容，这里以登录为例，如图 14-21 所示。

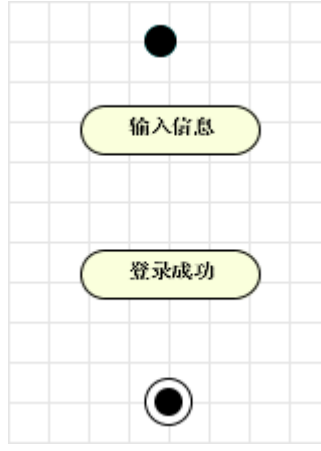


图 14-21　增加动作状态

这里增加了两个动作状态，首先进入输入信息页面，执行该动作。输入信息正确，就会执行登录成功的动作。

14.6.2　增加分支结构

通常活动图中动作状态的切换并不是“一帆风顺”的，例如输入登录所用的信息，当输入正确时将进入登录成功动作状态；但是当输入不正确时，就不能继续下面的状态，这种情况下就需要增加一个分支结构。

在活动图工具栏中，使用“  ”按钮，可以创建增加一个分支结构，例如这里我们增加一个对输入信息进行判断的分值结构，如图 14-22 所示。

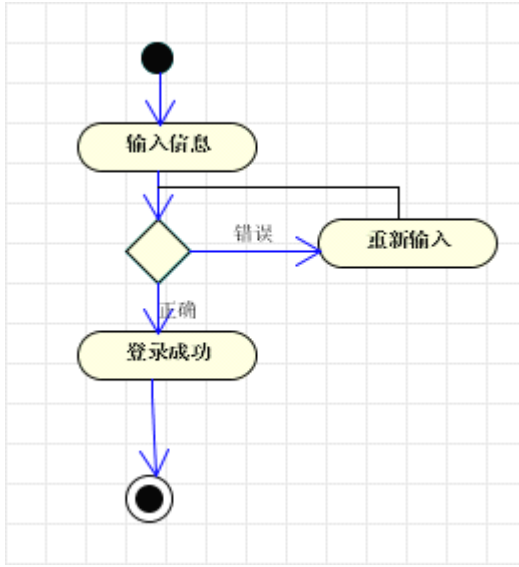


图 14-22　增加分支

其中菱形就表示分支结构，通常并不在其中输入任何内容。在分支结构上最少定义两个继续的动作状态，根据不同的结果，进入不同的动作状态。

在上图中，不但定义了分支结构，还将动作状态和分支结构之间使用实线箭头连接起来，从而组成一个简单的活动图。

14.7　时序图

时序图在有些地方也被称为序列图，它是一种以时间为序来描述对象之间传递消息。在开发中，通过时序图能够很好的描述程序的执行，在框架开发中尤其重要。时序图中包含 4 个元素，分别是对象、生命线、消息和激活。

14.7.1　创建时序图对象

在 MyEclipse 的 UML 模型仓库界面中，单击工具栏中第一个按钮后的倒三角，在其中选择“New Sequence Diagram”选项，就会切换到时序图开发的工具栏。

时序图对象时时序图中最重要的组成部分。在其中使用“ ”按钮，可以创建时序图对象，单击矩形其中的空白处就可以添加对象名称，例如这里我们创建用户和评论两个对象，如图 14-23 所示。

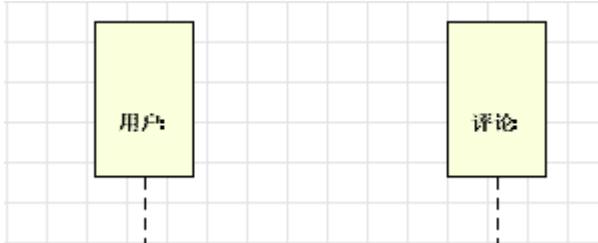


图 14-23　创建对象

从图中可以看到创建了两个对象，在这两个对象之间就可以完成一些操作，例如发表评论，查看评论等。

14.7.2　添加消息

时序图中的消息就是对象间的通信，一个对象可以请求另一个对象做某操作。在工具栏中，使用“  ”按钮可以创建时序图中的消息，这里我们就添加一个发表评论的消息，如图 14-24 所示。

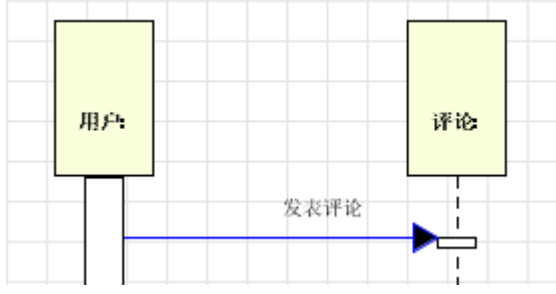


图 14-24　添加消息

添加消息后，则对象下面的生命线将被激活。在消息实现箭头上可以添加具体消息的内容，也就是具体的两个对象间的操作。

第15章　团队开发新闻发布系统

在前面的讲解中，我们已经了解了 MyEclipse 在开发中的重要功能，在本章中将通过一个综合案例，再来全面的认识 MyEclipse 这一款开发工具。在当前项目开发中，通常都是采用团队开发的方式，让每一组或者一位开发人员完成某一模块或者层的开发。在本章的讲解中就来模拟一下如何在 MyEclipse 开发工具中使用 Struts2+Spring+Hibernate 框架技术进行模拟开发。

15.1　系统分析和设计

在本章中将开发一个新闻发布系统，在进行开发之前，要首先确定在该系统中应该开发哪些功能，以及系统的性能、效率等问题。

15.1.1　需求分析

一个新闻发布系统，必须由两部分组成，分别是系统前台和系统后台。其中系统后台供新闻工作者使用，用来管理版面和新闻，其中管理板块包括新增版面、对已经板块进行修改以及删除指定板块；管理新闻包括发布新闻、浏览当前所有新闻和删除新闻。系统前台供所有网友使用，由用户自行选择感兴趣的版面，然后选择感兴趣的新闻。

15.1.2　功能分析

新闻发布系统分为前台和后台两个部分，其中前台是一个公共的平台，所有的系统访问者都可以使用。不过对于新闻发布系统的后台，则只提供给特定的用户，这里主要指新闻工作者和管理员。在该系统中根据权限将用户分成三种角色，分别为普通用户、新闻工作者和管理员。

对于普通用户，只能进入新闻发布系统的前台，可以选择自己感兴趣的新闻版面和新闻，并翻阅先问内容。

对于新闻工作者，除了具有普通用户的所有功能外，还可以进入新闻发布系统的后台。并对系统的版面进行管理，包括新增版面、查看已有版面、编辑版面和删除版面。还可以对新闻进行管理，包括发布新闻、浏览新闻和删除新闻。

对于管理员，除了拥有新闻工作者的所有功能外，还可以对系统用户进行管理，包括新增用户、更新用户、浏览所有用户和删除指定用户。

15.1.3　绘制 UML 用例图确定系统功能

从前面的系统分析中知道论坛系统一共包含三种用户角色，分别普通用户、新闻工作者和系统管理员。下面就来分析这三个角色所对应的用例图。

普通用户仅仅能选择指定版面和查看指定新闻，其用例图如图 20-1 所示。

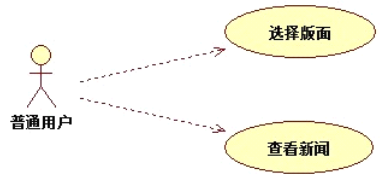


图 15-1　普通用户用例图

新闻工作者除了拥有普通用户的所有功能外，还可以对版面和新闻进行管理，其用例图如图 20-2 所示。

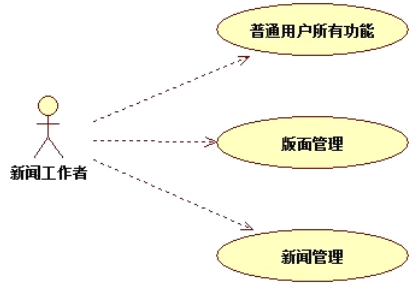


图 15-2　新闻工作者用例图

管理员拥有了新闻工作者的全部功能，还可以对所有的用户进行统一管理，包括新增用户、修改用户、删除用户以及查看所有用户。其用例图如图 20-3 所示。

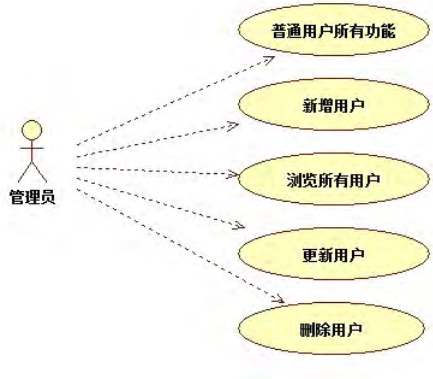


图 15-3　管理员用例图

15.1.4　绘制系统流程图

普通用户进入本系统后，将首先打开系统首页。在系统首页中，用户可以选择自己喜欢的版块，选择指定板块后将显示该板块的所有新闻。对于新闻工作者和管理员可以进入系统后台，在系统后台中可以对用户、版面、新闻进行管理。系统流程图如图 20-4

所示。

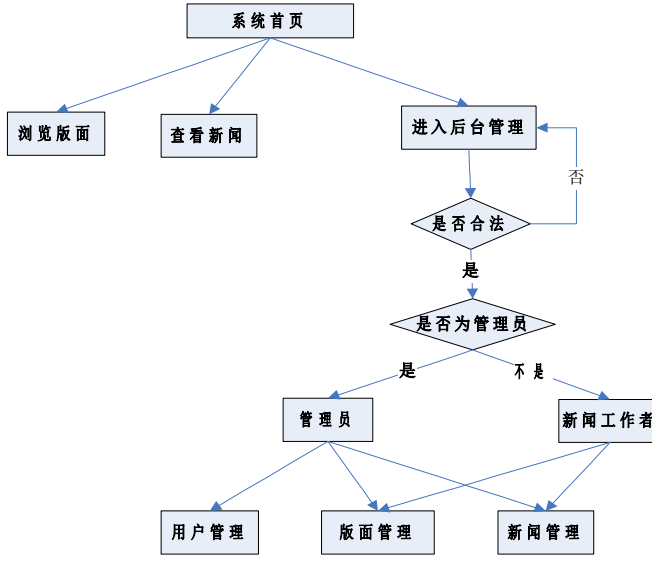


图 15-4　系统流程图

15.2　DBA 团队分析和设计数据库

DBA 是数据库管理员的英文缩写。当开发人员完成系统的设计后，就需要 DBA 来创建数据库存储结构和数据库对象。DBA 将根据系统设计分析采用何种数据库和创建什么数据表。在本书中，DBA 就采用 MySQL 数据库和 MyEclipse 来开发数据库。

15.2.1　数据库逻辑结构设计

在前面的需求和功能分析中，我们可以分析出在数据库中应该创建用户数据表、版面数据表和新闻数据表。接下来就是确定每一个数据表的逻辑结构。用户数据表用来保存用户的信息，包括用户编号、用户名、密码。该表的逻辑结果如表 15.1 所示。

表 15.1　用户数据表

字段名	数据类型	是否主键	描述
userID	整数(int)	是	用户编号
username	文本(varchar)	否	用户名
password	文本(varchar)	否	登录密码

版面表用来保存版面信息，包括版面编号、版面名称、版面描述、版面是否审核。该表的逻辑结果如表 15.2 所示。

表 15.2　版面数据表

字段名	数据类型	是否主键	描述
moduleID	整数(int)	是	版面编号
moduleName	文本(varchar)	否	版面名称

moduleDescription	文本(varchar)	否	版面描述
auditing	整数(binary)	否	是否审核

新闻表用来保存新闻信息，包括新闻编号、新闻标题、新闻内容、新闻作者、新闻发表时间、新闻关键字、新闻来源，所属版面编号，该表的逻辑结果如表 15.3 所示。

表 15.3　新闻数据表

字段名	数据类型	是否主键	描述
newsID	整数(int)	是	新闻编号
newsTitle	文本(varchar)	否	新闻标题
newsContent	文本(text)	否	新闻内容
newsAuthor	文本(varchar)	否	新闻作者
publishdDate	日期	否	新闻发表时间
newsKey	文本(varchar)	否	新闻关键字
origin	文本(varchar)	否	新闻来源
module	整数(int)	外键	所属版面编号

15.2.2　创建数据库和数据表

确定使用 MySQL 数据库和分析确定每一数据表的逻辑结构后，就可以进入 MyEclipse 的数据库视图进行数据库的开发。打开“MySQL”连接，在“Connected to MySQL”节点上，单击鼠标右键，在菜单中选择“Create database”命令，在弹出的创建数据库界面中填写相关信息后，如图 15-5 所示。

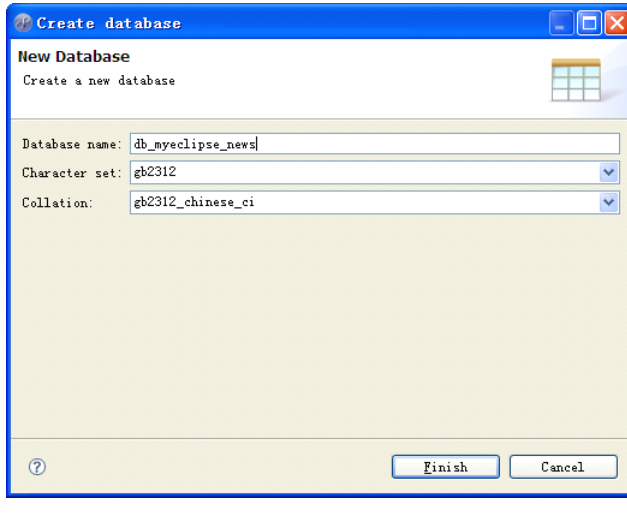


图 15-5　创建数据库

单击“Finish”按钮，将完成数据库的创建。选中“db_myeclipse_news”数据库节点，单击鼠标右键，选择“New Table”命令，将弹出创建数据表界面。通过该界面，依次创建用户、版面和新闻数据表。先来创建用户数据表，它的创建数据表界面如图 15-6 所示。

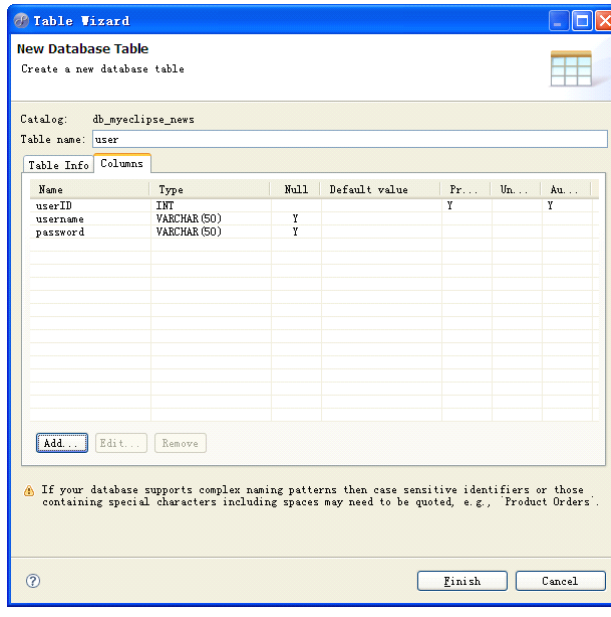


图 15-6　创建用户数据表

单击“Finish”按钮，将完成用户数据表的创建。使用同样的创建方法，继续创建版面数据表，它的创建数据表界面如图 15-7 所示。

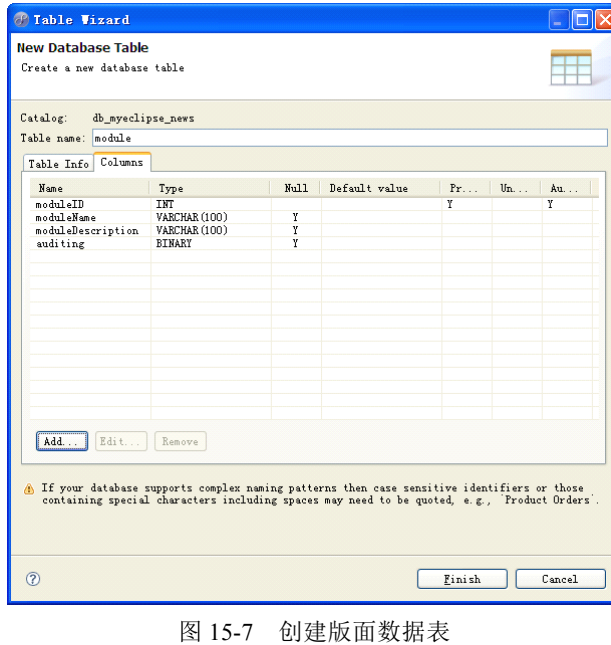


图 15-7　创建版面数据表

单击“Finish”按钮，将完成版面数据表的创建。在每一版面下面都将有多条新闻信息，新闻实体也是本章新闻发布系统的核心，所以新闻数据表的创建也是相对复杂的，它的创建数据表界面如图 15-8 所示。

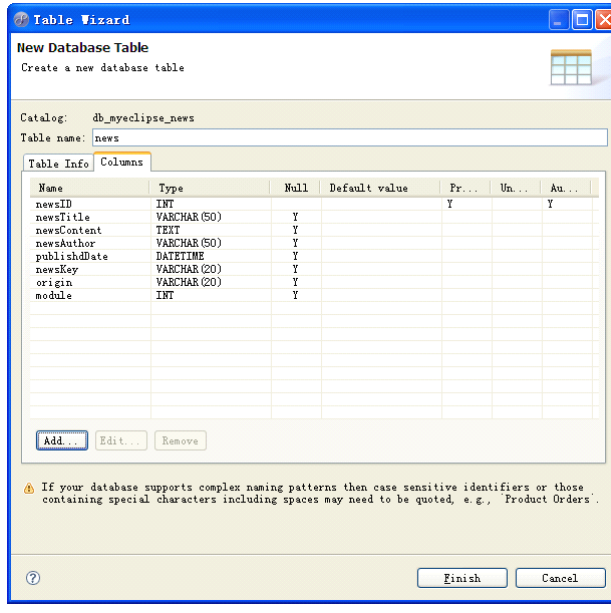


图 15-8　创建新闻数据表

单击“Finish”按钮，将完成新闻数据表的创建。到这里 DBA 将完成数据库的分析和设计，从而将创建后的数据库将给开发人员。开发人员将根据系统设计和数据库进行程序的开发。

15.3　项目经理配置版本控制服务器

因为我们采用的是团队开发的形式进行开发，所以所有的程序并不是开发在一台计算机中，这时候就需要使用到版本控制服务器。在前面的讲解中，我们了解了 CVS 和 SVN 这两种版本控制服务器，它们的操作是类似的，这里我们就采用 MyEclipse 中集成的 CVS 版本控制服务器。

使用 CVS 版本控制服务器时，首先就需要由项目经理在公司的主机上建立和配置版本控制服务器，团队中开发的所有程序都将提交到该服务器。在前面章节中已经学习了如何下载和安装 CVS 服务器，这里我们就不再重复，重点讲解如何配置版本控制服务器。

安装后，在系统开始菜单中，选择“程序”|“CVSNT”|“CVSNT Control Panel”命令，将弹出配置 CVS 服务器的界面。在其中选择“Repository configuration”选项卡，将进入到服务器库的界面。单击“Add”按钮，将进入到服务配置界面，在其中输入相关信息后，界面如图 15-9 所示。

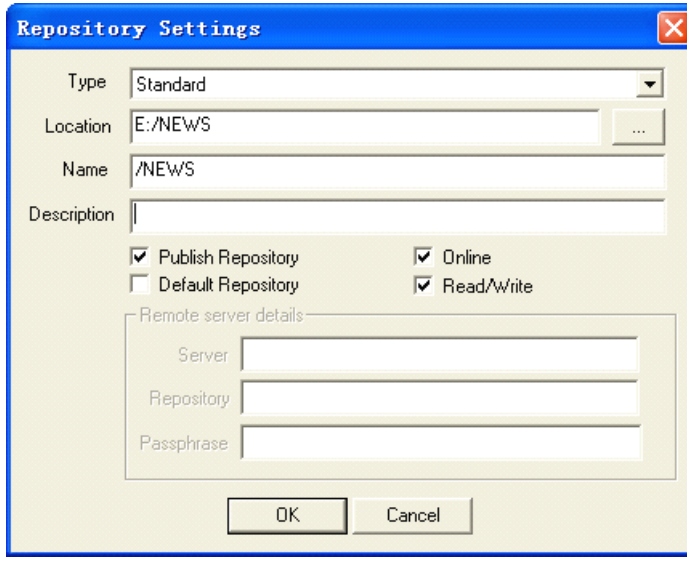


图 15-9　服务配置界面

在其中“Location”选项中选择的是 E 盘下的“NEWS”目录，从而团队中开发的所有程序都可以提交到该目录中。配置版本控制服务器还有其他一些选项，读者可以自己参考前面的章节进行学习。

15.4　开发数据访问层

在前面的讲解中，我们已经多次提到 Web 项目开发的分层思想，通常将整个项目分为表现层、控制层、业务逻辑层和数据访问层。每一层完成特定的工作，在开发中，可能一位开发人员完成其中一层的开发，从而就需要每一位开发人员创建一个项目 。具体项目开发环境的搭建，这里就不再重复，可以前面知识点的讲解。全部开发完成后将它们整合在一起，从而完成项目的开发。我们首先在本节中开发数据访问层，创建一个项目名称为“NEWS_dao”的项目。

15.4.1　生成数据访问层程序

使用 MyEclipse 开发工具中集成的 Hibernate 框架技术能够非常方便的开发数据访问层程序。并且在本章中采用 Spring+Hibernate 框架整合的技术开发，所以应该选择“Spring DAO”选项。当然

在 MyEclipse 的数据库视图中，选中 “MySQL”|“Connection to MySQL”|“db_myeclipse_news”|“TABLE”|“user”试题数据表节点，单击鼠标右键，在弹出的菜单中选择“Hibernate Reverse Engineering”命令，将弹出创建 Hibernate 程序的界面，在其中选择创建实体类、映射文件和 DAO 程序，如图 15-9 所示。

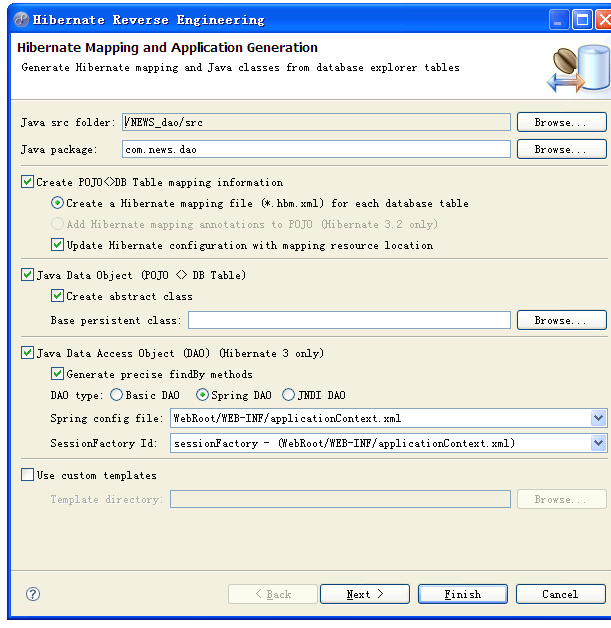


图 15-9　创建用户数据访问层

单击“Finish”按钮，将完成用户相关的实体类、映射文件和 DAO 实体类程序的创建。创建 DAO 程序后，要进行抽取接口的操作。在包资源管理器中，选中“UserDAO”类节点，单击鼠标右键，在弹出的菜单中，选择“Refactor”|“Extract Interface”命令，将弹出抽取接口界面，在其中输入接口名为“IUserDAO”，然后将其中的所有方法选中，界面如图 15-10 所示。

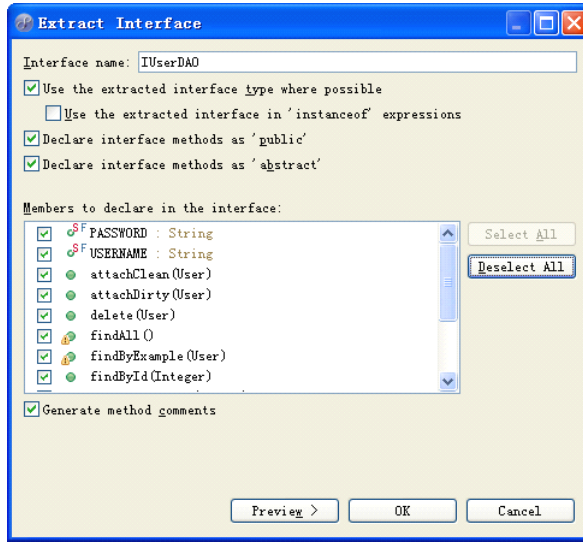


图 15-10　抽取接口

单击“OK”按钮，将完成用户 DAO 接口的抽取操作。使用同样的操作步骤，依次完成版面 DAO 接口和新闻 DAO 接口的抽取操作。

15.4.2 提交到版本控制服务器

开发完程序后，通常需要程序员自己完成单元测试，但是由于数据访问层程序完全是由 MyEclipse 自动生成的，从而可以省略该操作。为了让团队中其他程序员使用开发的数据访问层，需要将这些程序提交到 CVS 版本控制服务器中。

提交程序到版本控制服务器中，首先第一步就需要和服务器建立连接。将 MyEclipse 的透视图切换到 CVS Repository Exploring”CVS 服务器视图中，然后在“CVS Repositories”视图界面的空白处，单击鼠标右键，在弹出的菜单中选择“New”|“Repository Location”命令，将弹出添加 CVS 资源库的界面，在其中输入相关信息后，界面如图 15-11 所示。

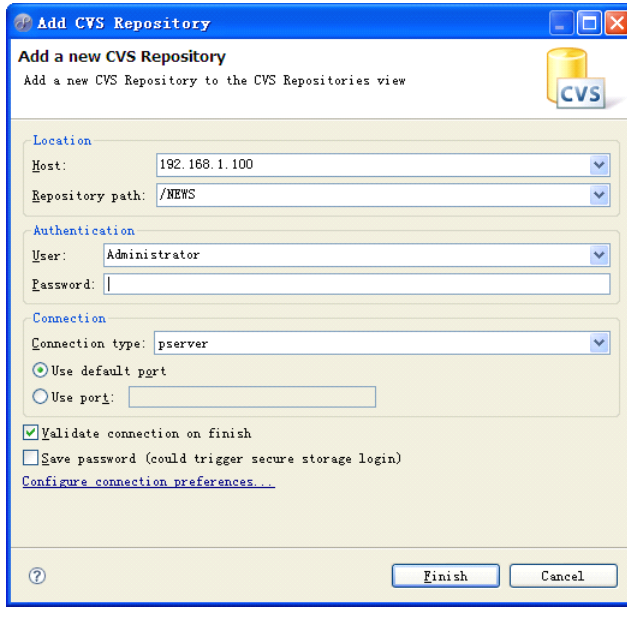


图 15-11　建立服务器连接

其中“Host”选项就是填写的公司主机的 IP 地址，“Repository path”表示资源库的路径，也就是我们在配置 CVS 服务器时配置的，这里填写“/NEWS”。单击“Finish”按钮，将完成 CVS 服务器的连接，从而在“CVS Repositories”视图界面多出一个资源库。

和服务器建立连接后，就可以将开发的程序提交到服务器中。第一次提交是比较复杂的，需要向进行共享操作。在 MyEclipse 中，切换到普通透视图后，在包资源管理器中选中要提交的项目。单击鼠标右键，在弹出菜单中选择“Team”|“Share Project”命令，将弹出选择资源库的界面，如图 15-12 所示。

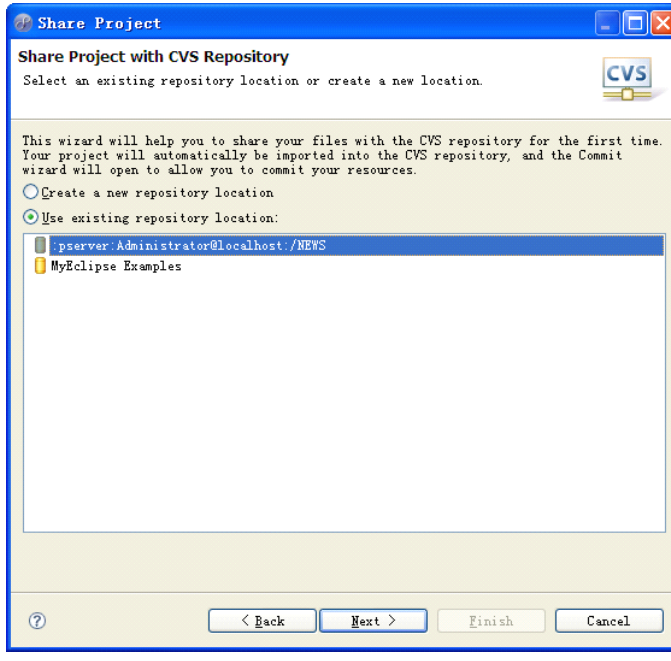


图 15-12　选择资源库

在其中选择“/NEWS”资源库单击“Next”按钮，将弹出设置提交后名称的界面，如图 15-13 所示。

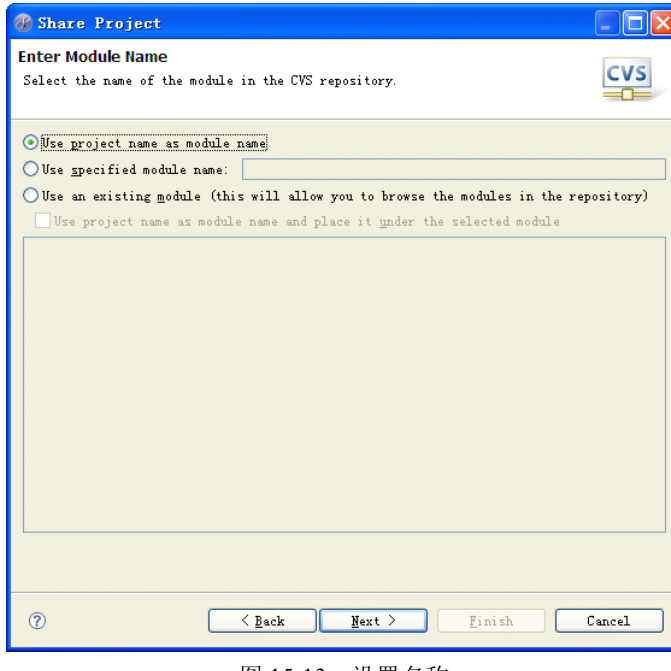


图 15-13　设置名称

在其中选择“Use project name module name”选项，它表示使用项目的名称做为提交后名称。单击“Next”按钮，将弹出共享项目资源界面，在其中选中了“Launch the Commit wizard”复选框后单击“Finish”按钮，将完成共享操作，并弹出提交项目界面。

在该界面中输入输入注释和描述后，界面如图 15-14 所示。

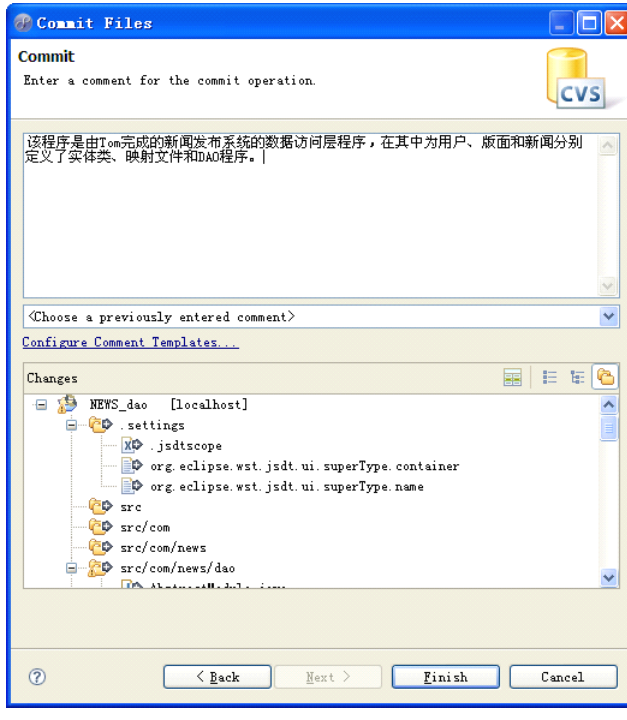


图 15-14　添加注释和描述

单击“Finish”按钮，将完成项目的第一次提交。当程序发生改变后，再次提交将变的容易。在包资源管理器中，选中被修改的程序节点或者选中项目节点，单击鼠标右键，在弹出的菜单中选择“Team”|“Commit”命令，将弹出提交文件界面，在其中再次输入提交描述后，单击“Finish”按钮，将完成提交文件的操作。

15.5　开发业务逻辑层

当不是团队开发时，业务逻辑层通常放在数据访问层后开发。但是在团队开发中，这两层可能同时进行，也可能是按照前面的顺序。业务逻辑层开发人员，从版本控制服务器中检出数据访问层程序后，进行开发。不过业务逻辑层开发人员也可以自己先开发一个空的数据访问层程序，等到数据访问层开发完成后，直接进行替换。

15.5.1　检出数据访问层程序

同时开发两层并没有什么难点的，只要两组人员间将程序和方法的名称统一好就可以。在本章中我们重点来看一下相继开发的方式，该方式是建立在检出版本控制服务器中的程序为基础的。

在 MyEclipse 中，切换到 CVS 操作透视图中后，在“CVS Repositories”视图界面中选中要检出的项目，例如这里选择前面操作上传的“NEWS_dao”项目。单击鼠标右

键，在弹出的菜单中选择“Check Out As”命令，将弹出修改检出项目名称的界面，在其中修改项目的名称为“NEWS_service”。其他检出操作见前面知识点章节，这里就不再重复给出。

15.5.2 开发用户业务逻辑层

在新闻发布系统中，用户的主要作用是发布新闻，而这部分用户不是通过注册定义了，这是由后台管理人员进行分配。所以在后台管理中，要有对系统用户的管理，这也是本节将要讲解的系统用户管理模块。

在新闻发布系统中，系统用户管理模块中包括新增用户、浏览用户、修改用户和删除用户等功能，这些功能也就是需要在用户业务逻辑层中开发的方法。因为在本章的新闻发布系统中将使用到 Spring 框架，所以我们将在业务逻辑层中将数据访问层对象注入，其具体代码如下所示：

```
01  package com.news.service;
02  import com.news.dao.IUserDAO;
03  import com.news.dao.User;
04  import com.news.util.Page;
05  import com.news.util.PageUtil;
06  import com.news.util.Result;
07  public class UserServiceImpl{
08      private IUserDAO userDAO;
09      public void setIUserDAO(IUserDAO userDAO) {
10          this.userDAO = userDAO;
11      }
12      public boolean addUser(User user) {
13          String username = user.getUsername();//获得用户名
14          if(userDAO.findByUsername(username) == null) {//用户名没有被占用
15              userDAO.save(user);//保存用户
16              return true;        //保存成功
17          }else{
18              return false;               //保存失败
19          }
20      }
21      public Result findAllUser(Page page) {
22          page = PageUtil.createPage(page, userDAO.findAll().size());
23          Result result = new Result();
24          result.setList(userDAO.findAll());
25          result.setPage(page);
26          return result;
27      }
28      public boolean deleteUser(String username) {
29          User user = (User)userDAO.findByUsername(username).get(0);
30          if(user == null) {//不存在该用户
31              return false;               //删除失败
```

```
32                  }else{
33                      userDAO.delete(user);
34                      return true;        //删除成功
35                  }
36          }
37          public boolean updateUser(User user) {
38                  User queryUser
39                      =(User)userDAO.findByUsername(user.getUsername()).get(0);
40                  if(queryUser == null) {//不存在该用户
41                      return false;               //更新失败
42                  }else{
43                      user.setUserId(queryUser.getUserId());//设置用户 ID
44                      userDAO.attachDirty(user);
45                      return true;        //更新成功
46                  }
47          }
48  }
```

上述代码中第 8 行定义了用户数据访问层接口引用，并为它定义了 Setter 方法，通过这种方式将用户数据访问层实现类对象注入。在后面的代码中依次定义了新增用户、浏览用户、删除用户和更新用户的业务方法。

开发完用户业务逻辑实现类后，要进行抽取接口操作，抽取出一个用户业务逻辑接口。抽取操作界面如图 15-15 所示。

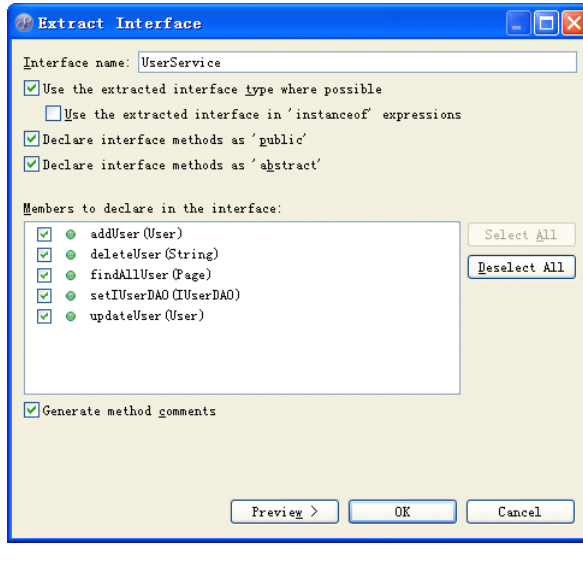


图 15-15　抽取用户业务逻辑接口

单击“OK”按钮，将完成用户业务逻辑接口的抽取。开发完用户业务逻辑实现类后，要在 Spring 配置文件中进行配置，配置界面如图 15-16 所示。

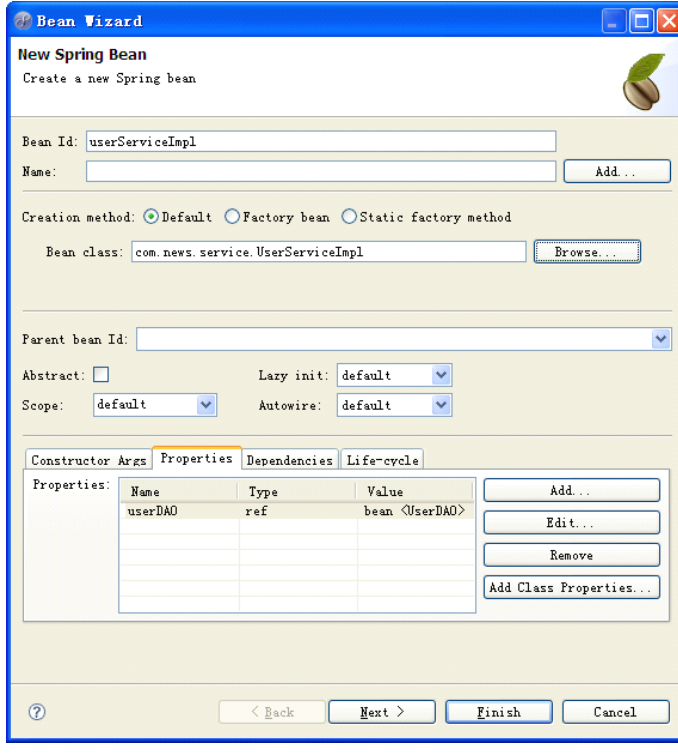


图 15-16　配置用户业务逻辑类

单击“Finish”按钮，将完成用户业务逻辑实现类的 Spring 配置，将用户数据访问层实现类对象注入到其中。

15.5.3　对用户业务逻辑类进行单元测试

开发完用户业务逻辑层程序后，是不能直接运行项目的，但是为了保证控制层开发的正确性，业务逻辑层程序开发人员需要完成对程序的单元测试，从而保证程序的正确性。

首先我们需要将 JUnit 单元测试所需要的 JAR 包导入到项目中，然后创建一个用户业务逻辑层的测试程序，创建 JUnit 程序的界面如图 15-17 所示。

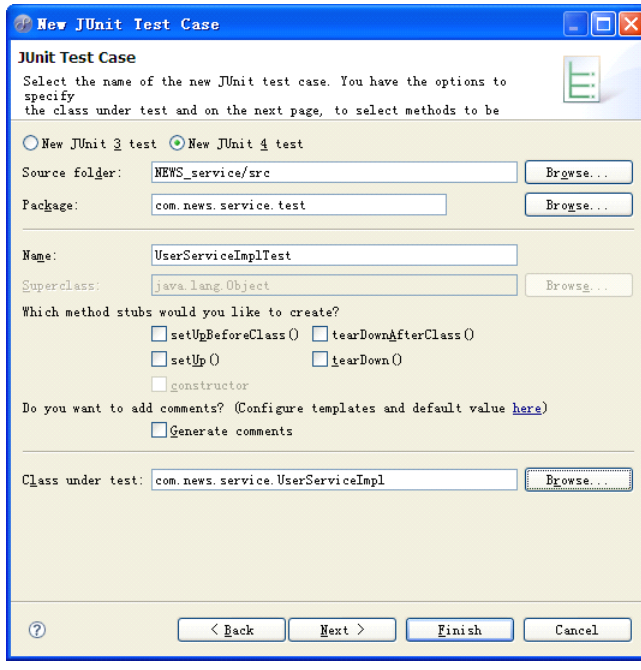


图 15-17　创建 JUnit 程序

单击“Next”按钮，将进入选择测试方法的界面，在其中选择用户业务逻辑类中的业务方法，如图 15-18 所示。

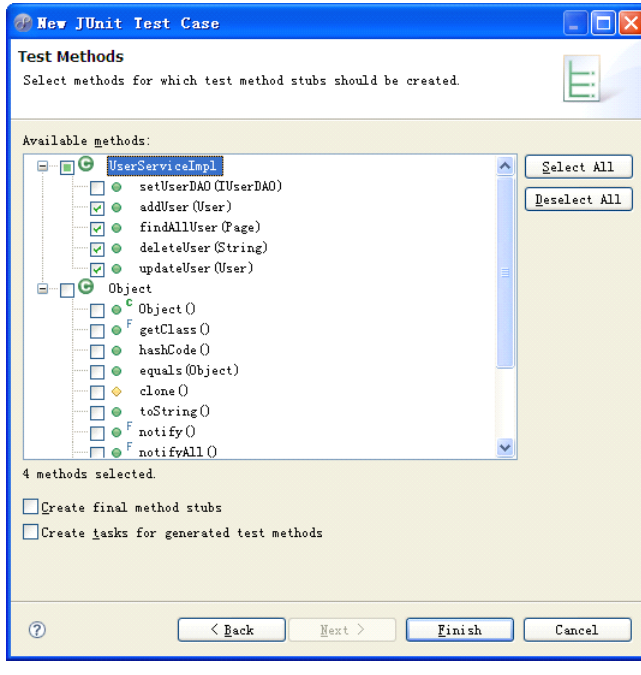


图 15-18　选择业务方法

单击“Finish”按钮，将自动生成代码。这是在提前导入 JUnit 单元测试 JAR 包的

前提下，如果没有导入，将进入添加 JAR 包的界面。在生成的代码中加入测试的程序，代码如下所示：

```
01  package com.news.service.test;
02  import static org.junit.Assert.*;
03  import org.junit.Test;
04  import org.springframework.beans.factory.BeanFactory;
05  import org.springframework.context.support.ClassPathXmlApplicationContext;
06  import com.news.dao.User;
07  import com.news.service.UserServiceImpl;
08  public class UserServiceImplTest {
09      @Test
10      public void testAddUser() {
11          User user=new User();
12          user.setUserId(1);
13          user.setUsername("admin");
14          user.setPassword("admin");
15          BeanFactory factory =
16              new ClassPathXmlApplicationContext("applicationContext.xml");
17          UserServiceImpl userServiceImpl=(UserServiceImpl)
18              factory.getBean("userServiceImpl");    //获取数据访问接口
19          boolean b=userServiceImpl.addUser(user);
20          assertTrue(b);
21      }
22      @Test
23      public void testFindAllUser() {
24      }
25      @Test
26      public void testDeleteUser() {
27          User user=new User();
28          user.setUserId(1);
29          user.setUsername("admin");
30          user.setPassword("admin");
31          BeanFactory factory =
32              new ClassPathXmlApplicationContext("applicationContext.xml");
33          UserServiceImpl userServiceImpl=(UserServiceImpl)
34              factory.getBean("userServiceImpl");    //获取数据访问接口
35          boolean b=userServiceImpl.deleteUser(user.getUsername());
36          assertTrue(b);
37      }
38      @Test
39      public void testUpdateUser() {
40          User user=new User();
41          user.setUserId(1);
42          user.setUsername("admin");
43          user.setPassword("admin");
```

```
44              BeanFactory factory =
45                      new ClassPathXmlApplicationContext("applicationContext.xml");
46              UserServiceImpl userServiceImpl=(UserServiceImpl)
47                      factory.getBean("userServiceImpl");    //获取数据访问接口
48              boolean b=userServiceImpl.updateUser(user);
49              assertTrue(b);
50          }
51      }
```

上述代码中分别对新增用户、删除用户和更新用户进行了测试，对于查询所有用户来说是没有测试的必要的。在测试方法中是不能够直接通过 new 来创建业务逻辑对象的，这是因为在其中使用 Spring 框架将数据访问对象注入，需要通过实例化 Bean 容器创建对象。

注意：在实例化 Bean 容器时，需要使用到 Spring 的配置文件，也就是“applicationContext.xml”。采用这种方式实例，必须将它拷贝到 src 目录下一份，不然是不能够实例成功的。

15.5.4 开发版面业务逻辑层

在新闻发布系统中，很重要的一个组成元素就是版面，通过使用版面可以将新闻进行分类。例如娱乐新闻、体育新闻等。在版面管理中，包括创建新版面、浏览目前所有版面、更新版面和删除版面等操作，所以在版面业务逻辑层中需要创建对应的业务方法，其程序代码如下所示：

```
01  package com.news.service;
02  import java.util.List;
03  import com.news.dao.IModuleDAO;
04  import com.news.dao.Module;
05  import com.news.util.Page;
06  import com.news.util.PageUtil;
07  import com.news.util.Result;
08  public class ModuleServiceImpl{
09          private IModuleDAO moduleDAO;
10          public void setModuleDAO(IModuleDAO moduleDAO) {
11                  this.moduleDAO = moduleDAO;
12          }
13          public boolean addModule(Module module) {
14                  String moduleName = module.getModuleName();//获得版面名称
15                  if(moduleDAO.findByModuleName(moduleName).size() == 0) {//没有被占用
16                          moduleDAO.save(module);//保存版面
17                          return true;        //保存成功
18                  }else{
19                          return false;               //保存失败
20                  }
21          }
```

```
22          public boolean deleteItemByModuleID(int moduleID) {
23              Module module = moduleDAO.findById(moduleID);
24              if(module == null) {//不存在该版面
25                  return false;              //删除失败
26              }else{
27                  moduleDAO.delete(module);
28                  return true;        //删除成功
29              }
30          }
31          public Result findAllModule(Page page) {
32              page = PageUtil.createPage(page, moduleDAO.findAll().size());
33              Result result = new Result();
34              result.setList(moduleDAO.findAll());
35              result.setPage(page);
36              return result;
37          }
38          public boolean updateModule(Module module) {
39              Module queryModule = moduleDAO.findById(module.getModuleId());
40              if(queryModule == null) {//不存在该版面
41                  return false;              //更新失败
42              }else{
43                  moduleDAO.attachDirty(module);
44                  return true;        //更新成功
45              }
46          }
47          public Module findModule(int moduleID) {
48              return moduleDAO.findById(moduleID);
49          }
50          public List<Module> findAllModule() {
51              return moduleDAO.findAll();
52          }
53      }
```

上述代码中第 9 行定义了版面数据访问接口引用，并为它定义了 Setter 方法，从而将版面数据访问类对象注入。在后面的程序中，依次定义了新增版面、删除版面、查询所有版面、更新版面和查找指定版面的业务方法。

开发完版面业务逻辑实现类后，要进行抽取接口操作，抽取出一个版面业务逻辑接口。抽取操作界面如图 15-19 所示。

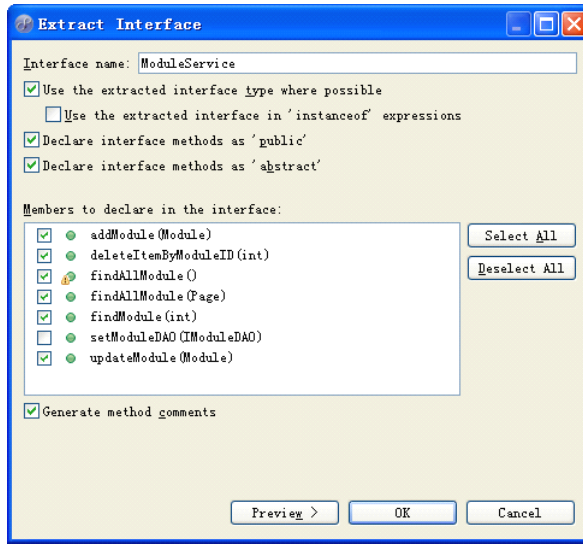


图 15-19　抽取版面业务逻辑接口

单击“OK”按钮，将完成版面业务逻辑接口的抽取。开发完版面业务逻辑实现类后，要在 Spring 配置文件中进行配置，配置界面如图 15-20 所示。

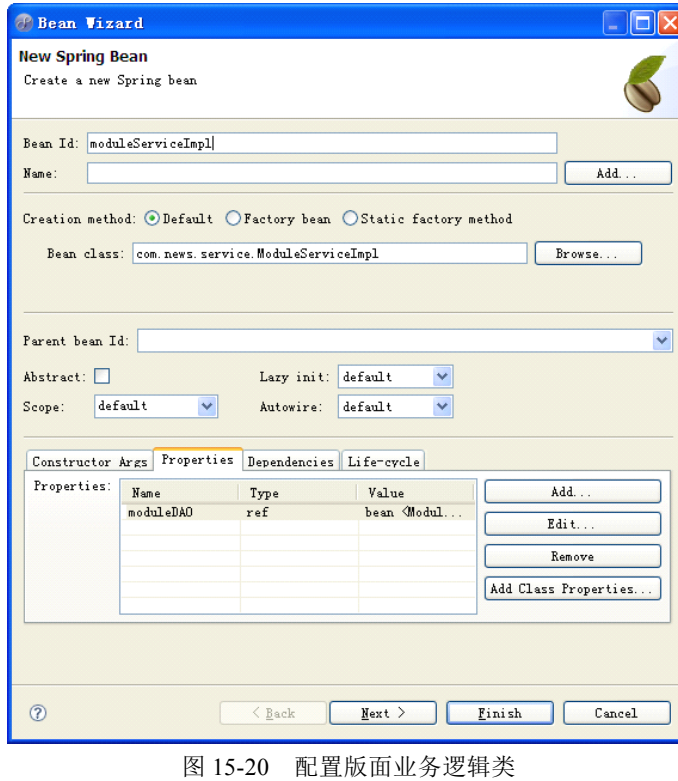


图 15-20　配置版面业务逻辑类

单击“Finish”按钮，将完成版面业务逻辑实现类的 Spring 配置，将版面数据访问层实现类对象注入到其中。

15.5.5 对版面业务逻辑类进行单元测试

开发完版面业务逻辑层程序后，同样也需要对它进行测试。创建测试程序的步骤这里就不再给出，它的具体代码如下所示：

```
01  package com.news.service.test;
02  import static org.junit.Assert.*;
03  import org.junit.BeforeClass;
04  import org.junit.Test;
05  import org.springframework.beans.factory.BeanFactory;
06  import org.springframework.context.support.ClassPathXmlApplicationContext;
07  import com.news.dao.Module;
08  import com.news.service.ModuleServiceImpl;
09  public class ModuleServiceImplTest {
10      static ModuleServiceImpl moduleServiceImpl=null;
11      @BeforeClass
12      public static void beforeClass() {
13          BeanFactory factory =
14                  new ClassPathXmlApplicationContext("applicationContext.xml");
15          moduleServiceImpl=(ModuleServiceImpl)
16                  factory.getBean("moduleServiceImpl");
17      }
18      @Test
19      public void testAddModule() {
20          Module module=new Module();
21          module.setModuleId(1);
22          module.setModuleName("国内新闻");
23          module.setModuleDescription("该栏目刊登国内新闻");
24          module.setAuditing("1");
25          boolean b=moduleServiceImpl.addModule(module);
26          assertTrue(b);
27      }
28      @Test
29      public void testDeleteItemByModuleID() {
30          boolean b=moduleServiceImpl.deleteItemByModuleID(1);
31          assertTrue(b);
32      }
33      @Test
34      public void testUpdateModule() {
35          Module module=new Module();
36          module.setModuleId(1);
37          module.setModuleName("国内新闻");
38          module.setModuleDescription("该栏目刊登国内新闻");
39          module.setAuditing("0");
40          boolean b=moduleServiceImpl.updateModule(module);
41          assertTrue(b);
```

```
42          }
43          @Test
44          public void testFindModule() {
45                  Module m=moduleServiceImpl.findModule(1);
46                  assertNotNull(m);
47          }
48  }
```

上述对版面业务逻辑层进行测试的程序和前面对用户业务逻辑类进行测试的程序有所不同，在其中第 12 行中定义了类初始化方法，在该方法中定义了实例化 Spring 的 Bean 容器的代码，从而完成创建业务逻辑类对象的工作。

在后面的具体测试方法中，将不用在进行实例化 Bean 容器的工作，只需要写具体的测试方法，分别定义了新增版面、删除版面、更新版面和查找版面方法的测试方法。

15.5.6 开发新闻业务逻辑层

新闻业务可以说是新闻发布系统的核心功能，在新闻业务逻辑层程序中定义了完成发布新闻、浏览所有新闻和删除新闻等功能，所以要在新闻业务逻辑类中定义相对应的方法，该程序的具体代码如下所示：

```
01  package com.news.service;
02  import com.news.dao.IModuleDAO;
03  import com.news.dao.INewsDAO;
04  import com.news.dao.Module;
05  import com.news.dao.News;
06  import com.news.util.Page;
07  import com.news.util.PageUtil;
08  import com.news.util.Result;
09  public class NewsServiceImpl{
10          private INewsDAO newsDAO;
11          private IModuleDAO moduleDAO;
12          public void setNewsDAO(INewsDAO newsDAO) {
13                  this.newsDAO = newsDAO;
14          }
15          public void setModuleDAO(IModuleDAO moduleDAO) {
16                  this.moduleDAO = moduleDAO;
17          }
18          public boolean addNews(News news) {
19                  String newsTitle = news.getNewsTitle();//获得新闻标题
20                  if(newsDAO.findByNewsTitle(newsTitle).size() == 0) {//没有被占用
21                          Module module = moduleDAO.findById(news.getModule());
22                          news.setModule(module.getModuleId());
23                          newsDAO.save(news);        //保存新闻
24                          return true;        //保存成功
25                  }else{
26                          return false;                //保存失败
```

```
27              }
28          }
29          public boolean deleteItemByNewsID(int newsID) {
30              News news = newsDAO.findById(newsID);
31              if(news == null) {//不存在该版面
32                  return false;              //删除失败
33              }else{
34                  newsDAO.delete(news);
35                  return true;        //删除成功
36              }
37          }
38          public Result findAllNews(Page page) {
39              page = PageUtil.createPage(page, newsDAO.findAll().size());
40              Result result = new Result();
41              result.setList(newsDAO.findAll());
42              result.setPage(page);
43              return result;
44          }
45          public News findNews(int newsID) {
46              return newsDAO.findById(newsID);
47          }
48          public boolean updateNews(News news) {
49              News queryNews = newsDAO.findById(news.getNewsId());
50              if(queryNews == null) {//不存在该版面
51                  return false;              //更新失败
52              }else{
53                  newsDAO.attachDirty(news);
54                  return true;        //更新成功
55              }
56          }
57      }
```

上述代码中第 10 行和第 11 行分别定义了新闻数据访问接口引用和版面数据访问接口引用，并为它们定义了 Setter 方法，从而将对应的数据访问对象注入。然后依次定义了新增新闻、删除新闻、查询所有新闻、查询单个新闻和更新新闻的业务方法，从而完成对应的功能。

开发完新闻业务逻辑实现类后，要进行抽取接口操作，抽取出一个新闻业务逻辑接口。抽取操作界面如图 15-20 所示。

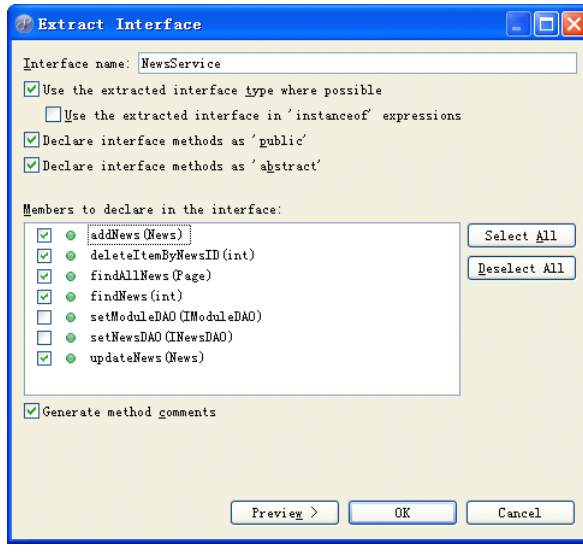


图 15-20　抽取新闻业务逻辑接口

单击“OK”按钮，将完成新闻业务逻辑接口的抽取。开发完新闻业务逻辑实现类后，要在 Spring 配置文件中进行配置，配置界面如图 15-21 所示。

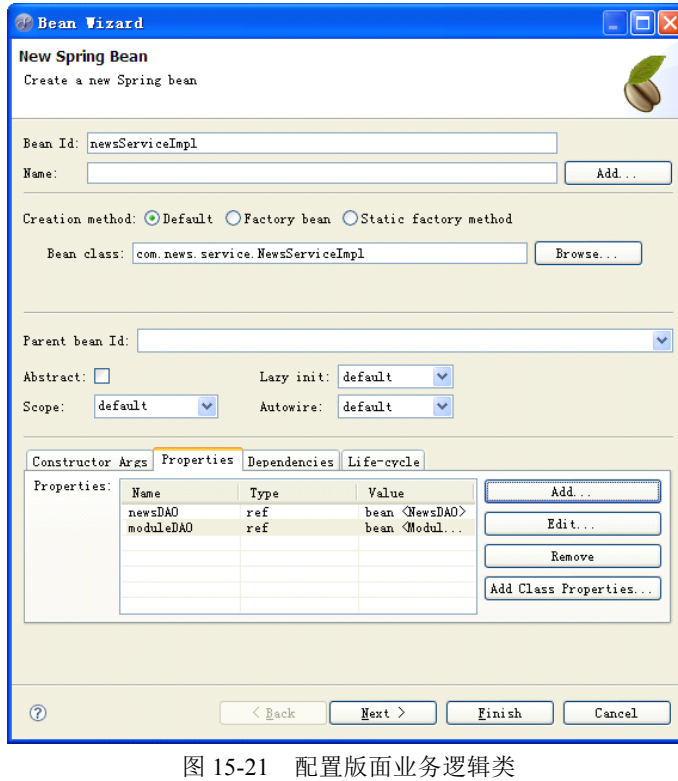


图 15-21　配置版面业务逻辑类

单击“Finish”按钮，将完成新闻业务逻辑实现类的 Spring 配置，将新闻数据访问层实现类对象注入到其中。

15.5.7 对新闻业务罗逻辑类进行单元测试

和前面的两个业务逻辑类一样，开发完程序后都需要编写对应的单元测试程序，通过该程序完成对程序功能的测试。具体测试程序的开发步骤同样还是不再给出，主要来看它的具体程序，代码如下所示：

```
01  package com.news.service.test;
02  import static org.junit.Assert.*;
03  import java.sql.Timestamp;
04  import org.junit.BeforeClass;
05  import org.junit.Test;
06  import org.springframework.beans.factory.BeanFactory;
07  import org.springframework.context.support.ClassPathXmlApplicationContext;
08  import com.news.dao.News;
09  import com.news.service.NewsServiceImpl;
10  public class NewsServiceImplTest {
11      static NewsServiceImpl newsServiceImpl=null;
12      @BeforeClass
13      public static void beforeClass() {
14          BeanFactory factory =
15                  new ClassPathXmlApplicationContext("applicationContext.xml");
16          newsServiceImpl=(NewsServiceImpl)factory.getBean("newsServiceImpl");
17      }
18      @Test
19      public void testAddNews() {
20          News news=new News();
21          news.setNewsId(1);
22          news.setNewsTitle("一年一度的两会召开");
23          news.setNewsContent("一年一度的两会在北京召开");
24          news.setNewsAuthor("李磊");
25          news.setNewsKey("两会");
26          news.setOrigin("");
27          news.setPublishdDate(new Timestamp(55555));
28          boolean b=newsServiceImpl.addNews(news);
29          assertTrue(b);
30      }
31      @Test
32      public void testDeleteItemByNewsID() {
33          boolean b=newsServiceImpl.deleteItemByNewsID(1);
34          assertTrue(b);
35      }
36      @Test
37      public void testFindNews() {
38          News news=newsServiceImpl.findNews(1);
39          assertNotNull(news);
40      }
```

```
41          @Test
42          public void testUpdateNews() {
43                  News news=new News();
44                  news.setNewsId(1);
45                  news.setNewsTitle("一年一度的两会召开");
46                  news.setNewsContent("一年一度的两会在北京召开");
47                  news.setNewsAuthor("李磊");
48                  news.setNewsKey("两会");
49                  news.setOrigin("");
50                  news.setPublishdDate(new Timestamp(55555));
51                  boolean b=newsServiceImpl.updateNews(news);
52                  assertTrue(b);
53          }
54  }
```

上述代码中同样在第 13 行定义了类初始化方法，在该方法中定义了实例化 Spring 的 Bean 容器的代码，从而完成创建业务逻辑类对象的工作。在后面的具体测试方法中，将不用在进行实例化 Bean 容器的工作，只需要写具体的测试方法，分别定义了新增新闻、删除新闻、更新新闻和查找新闻方法的测试方法。

15.6 开发控制层

开发完业务逻辑层后，同样也需要将程序提交到版本控制服务器中，从而共享给其他开发团队，主要是提供给控制层开发团队。在控制层开发团队开发对应程序时，如果业务逻辑层程序已经开发完成，将从版本控制服务器中检出相应的程序。这些操作读者可以参考前面的讲解学习，这里就不再给出。

15.6.1 开发发布新闻 Action

由于本章开发的新闻发布系统中的功能相对是比较多的，所以对应处理这里功能的 Action 程序也就比较多。这里就采用其中最核心的发布新闻功能来讲解 Action 程序的开发。在 MyEclipse 的原始安装中，并没有集成 Struts2 框架，从而也就不存在直接创建 Action 的功能。在前面的学习中，已经讲解了如何通过创建类的方式来创建 Action 程序，这里就不再讲解，直接来看它的程序，其代码如下所示：

```
01  package com.news.action;
02  import java.sql.Timestamp;
03  import com.news.dao.Module;
04  import com.news.dao.News;
05  import com.news.service.NewsService;
06  import com.opensymphony.xwork2.ActionSupport;
07  public class NewsAddAction extends ActionSupport {
08          private NewsService newsService;
09          private String newsTitle;           //新闻标题
10          private String newsContent;         //新闻内容
```

```
11          private String newsAuthor;          //新闻作者
12          private String newsKey;                 //关键字
13          private String origin;              //新闻来源
14          private int module;                     //所属板块
15          public void setNewsService(NewsService newsService) {
16              this.newsService = newsService;
17          }
18          public String getNewsTitle() {
19              return newsTitle;
20          }
21          public void setNewsTitle(String newsTitle) {
22              this.newsTitle = newsTitle;
23          }
24          public String getNewsContent() {
25              return newsContent;
26          }
27          public void setNewsContent(String newsContent) {
28              this.newsContent = newsContent;
29          }
30          public String getNewsAuthor() {
31              return newsAuthor;
32          }
33          public void setNewsAuthor(String newsAuthor) {
34              this.newsAuthor = newsAuthor;
35          }
36          public String getNewsKey() {
37              return newsKey;
38          }
39          public void setNewsKey(String newsKey) {
40              this.newsKey = newsKey;
41          }
42          public String getOrigin() {
43              return origin;
44          }
45          public void setOrigin(String origin) {
46              this.origin = origin;
47          }
48          public int getModule() {
49              return module;
50          }
51          public void setModule(int module) {
52              this.module = module;
53          }
54          public String execute() throws Exception {
55              //将所有的参数组合成一个 News 对象
```

```
56                  News news = new News();
57                  news.setNewsAuthor(newsAuthor);
58                  news.setNewsContent(newsContent);
59                  news.setNewsKey(newsKey);
60                  news.setNewsTitle(newsTitle);
61                  news.setOrigin(origin);
62                  news.setPublishdDate(new Timestamp(5555));
63                  Module newModule = new Module();
64                  newModule.setModuleId(module);
65                  news.setModule(module);
66                  if(newsService.addNews(news)){//保存成功
67                      this.addActionMessage("添加新闻成功!");
68                  }else{
69                      this.addActionMessage("新闻标题已经被占用，请重新输入");
70                  }
71                  return SUCCESS;
72              }
73      }
```

上述代码中第 8 行定义了新闻业务逻辑接口引用，并为它在第 15 行定义了 Setter 方法，从而将新闻业务逻辑对象注入。从第 9 行到第 14 行依次定义了新闻的标题、内容、作者、关键字、发表时间和所属版面等属性，并为这些属性定义了 Setter 和 Getter 方法，从而能够获取发布新闻网页中的提交信息。

在 Action 程序中的处理方法中，首先创建了新闻对象，然后将获取到的属性值设置到该对象中，在第 66 行中调用新闻业务接口中的添加新闻的业务方法，从而将保存在新闻对象中的信息提交给后台程序。

15.6.2 配置 Action 程序

当使用 Struts 2+Spring 框架开发时，配置 Action 程序是比较繁琐的，不但要在 Spring 的配置文件中进行配置，还要在 Struts 2 的配置文件中进行配置。通常首先在 Spring 的配置文件中进行配置，它的配置界面如图 15-22 所示。

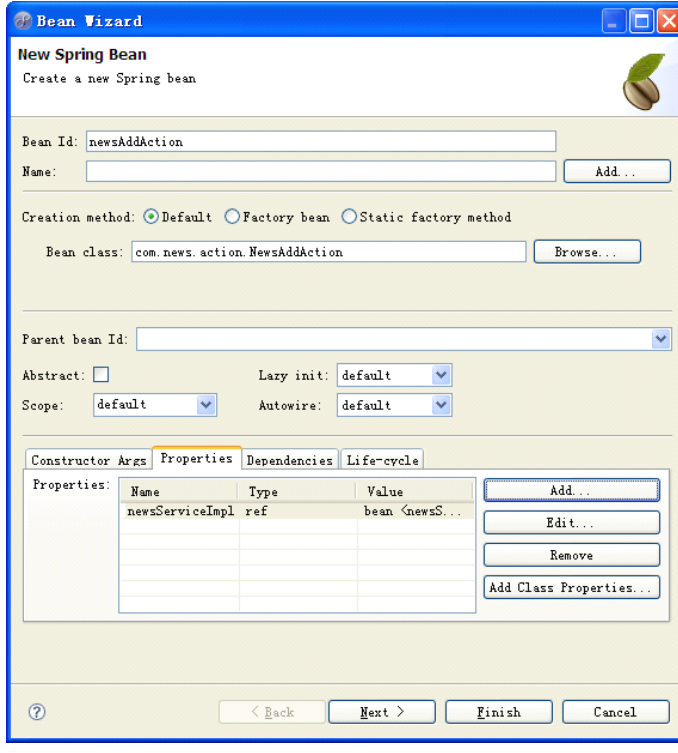


图 15-22　在 Spring 中配置 Action

单击“Finish”按钮，将发布新闻 Action 的配置，并将新闻业务逻辑实现类对象注入进去。完成 Spring 配置后，要在 Struts 2 配置文件中进行配置，它的配置代码为：

```
01        <action name="newsAdd" class="newsAddAction">
02            <result name="success" type="chain">publishNewsBefore</result>
03        </action>
```

其中第 1 行的 class 属性指定 Spring 配置文件中的 id 属性。第 2 行配置了当发布新闻后跳转到显示所有新闻的 Action 程序，该 Action 程序以及其他 Action 读者可以在光盘源代码中看到，这里就不再一一讲解。

15.7　开发表现层并运行系统

15.7.1　开发前台表现层

除了前面开发的数据访问层、业务逻辑层和控制层，一个项目中还应该具有表现层，也就是前台显示程序。这部分程序通常是由专门的前台美工来完成，在 MyEclipse 中虽然也集成了前台程序的开发功能，但是并不优秀的，所以这部分程序我们在光盘中直接给出，这里就不再详细讲解。

15.7.2　运行系统

给出表现层程序后，我们就可以访问我们开发的整个系统了。将项目部署到 Tomcat

服务器中后，启动服务器。如果启动时发生错误，将项目中的“asm-x.jar”JAR 包删除，因为其中该包重复。启动成功后，打开浏览器，在其中输入如下地址：

http://localhost:8080/NEWS/goIndex.action

运行结果如图 15-23 所示。

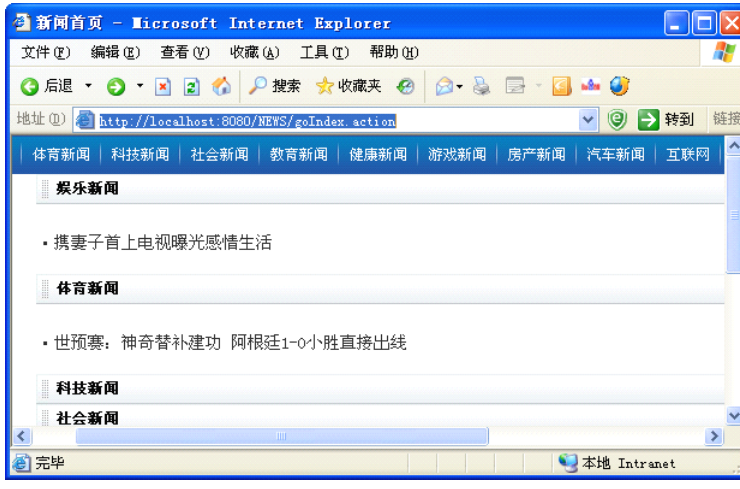


图 15-23　新闻系统前台首页

在该页面中，单击版面超链接，将查看该版面中的所有新闻。单击新闻标题超链接，可以查看该新闻的详细信息。再次在地址栏中输入如下地址：

http://localhost:8080/NEWS/admin/index.jsp

运行结果如图 15-24 所示。

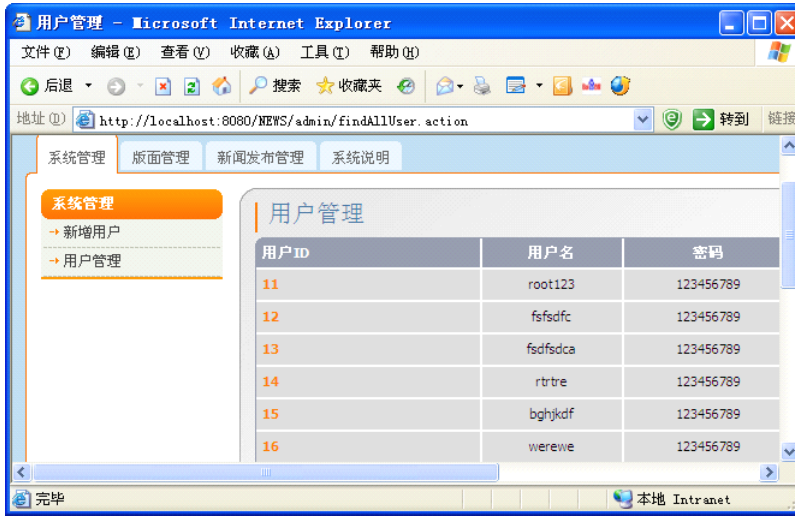


图 15-24　新闻系统后台首页

该页面是对新闻系统进行管理的页面，其中包括用户管理、版面管理和新闻管理，通过它们可以完成发布新闻等日常操作。发布新闻后，在前台页面中就会看到发布的内容。这些操作读者可以自己运行进行演示。