

MyEclipse 6 Java EE 开发中文手册



2007 年 12 月

刘长炯 著

献给我最爱的父母！

愿上帝保佑苍生！

感谢

It's all of you, who makes me a super No. 1, thank you !

感谢 Michael Jackson 奉献好听的歌舞！

感谢所有鼓励我的好朋友们！

感谢 Weblogic 专家王超先生对本书的大力支持！

目 录

MyEclipse 6 Java EE 开发中文手册	1
目 录	4
介 绍	8
文档说明	8
适用的读者	8
关于作者	8
版权声明	9
第一章 安装配置开发环境	10
1.1 系统需求	10
1.2 JDK 的下载, 安装和配置 (可选)	10
1.2.1 下载JDK	10
1.2.2 安装JDK	13
1.2.3 配置环境变量 (可选)	14
1.2.4 JDK 6 中文文档下载地址(ZIP,HTML,CHM) (可选)	16
1.3 Tomcat服务器的下载, 安装和运行(可选)	17
1.4 JBoss 服务器的下载, 安装和运行 (可选)	19
1.5 MySQL 5 数据库服务器下载, 安装和运行 (可选)	22
1.5.1 MySQL 5 官方版本的下载和安装, 运行	22
1.5.2 MySQL 5 绿色版的下载安装和运行	23
1.5.2.1 下载	23
1.5.2.2 用法图解	24
1.6 Eclipse 3.3 的下载, 安装和运行	26
1.7 MyEclipse 6 的下载, 安装和运行	27
1.7.1 下载	28
1.7.2 安装	29
1.7.2.1 ALL in ONE 版本的安装	29
1.7.2.2 插件 (PLUG-IN) 版本的安装	30
1.7.3 运行	30
1.8 小结	31
第二章 开发第一个Java应用程序	32
2.1 介绍	32
2.2 手工编写, 编译并运行Java程序	32
2.3 使用Eclipse/MyEclipse来编写, 编译并运行Java程序	33
2.4 小结	36
第三章 Eclipse 的基础概念, 配置和使用	37
3.1 界面布局	37
3.1.1 菜单	37
3.1.2 工具栏	37
3.1.3 透视图 (Perspective) 切换器	38
3.1.4 视图 (View)	39
3.1.5 上下文菜单 (Context Menu)	41

3.1.6 状态栏 (Status Bar)	41
3.1.7 编辑器 (Editor)	41
3.2 常见概念和操作	42
3.2.1 项目 (Project)	42
3.2.2 工作区 (Workspace)	42
3.2.3 导入、导出Java项目	42
3.2.3.1 导入项目	42
3.2.3.2 导出项目	43
3.2.4 快速修正代码错误	43
3.2.5 优化导入列表	44
3.2.6 添加, 修改, 删除JRE	44
3.2.7 查看类定义, 层次和源码	44
3.2.8 查找类文件 (Open Type)	45
3.2.9 源码目录, 输出路径, Library和编译器版本设置	45
3.2.10 生成getter和setter 方法	46
3.2.11 格式化源代码	47
3.2.12 注释和取消注释	47
3.2.13 手工和自动编译	47
3.2.14 直接粘贴Java源码为类文件	47
3.2.15 复制项目中的文件	47
3.2.16 断点和调试器	48
3.2.17 快速加入、删除jar包到Build Path	49
3.2.18 查看当前类被哪些类引用	49
3.2.19 设置编辑器字体, 颜色和显示行号	49
3.2.20 Link文件	50
3.2.21 安装插件	51
3.2.22 获取帮助和阅读帮助文档	51
3.2.23 CVS团队源代码管理 (在线阅读)	51
3.3 小结	51
第四章 用MyEclipse Database Explorer管理数据库	52
4.1 功能一览	52
4.2 使用MyEclipse Database Explorer透视图	54
4.2.1 介绍	54
4.2.2 连接到MyEclipse Derby数据库	55
4.2.3 切换到MyEclipse Database Explorer透视图	55
4.2.4 打开数据库连接	56
4.2.5 关闭数据库连接	57
4.2.6 浏览数据库结构	57
4.2.7 编辑和执行SQL代码段	58
4.2.8 生成实体关系 (ER) 图	60
4.2.9 编辑表格数据	61
4.2.10 清空表格数据	62
4.2.11 创建和删除表格	62
4.2.12 创建和删除外键	63

4.2.13 创建和删除索引	64
4.2.14 生成SQL语句	65
4.2.15 建立到MySQL数据库的连接	66
4.3 小结	67
4.4 参考资料	67
第五章 开发JDBC应用	68
5.1 系统需求	68
5.2 创建数据库表格	68
5.3 创建Java项目	69
5.4 添加JDBC驱动到Build Path	69
5.5 编写JDBC访问类	70
5.6 小结	73
5.7 参考资料	74
第六章 管理应用服务器	75
6.1 简介	75
6.2 Servers 视图	75
6.3 浏览应用服务器连接器	76
6.4 配置连接器	77
6.4.1 第 1 步 配置服务器的安装信息	78
6.4.2 第 2 步 启用连接器	78
6.4.3 第 3 步 选择启动服务器时候所用的JDK	78
6.4.3.1 可选操作：添加 JVM	79
6.5 发布并运行Java EE项目	80
6.5.1 Java EE 项目的发布类型	80
6.5.1.1 散包发布	80
6.5.1.2 打包发布	80
6.5.2 向服务器发布应用	80
6.5.2.1 第 1 步 打开发布对话框	80
6.5.2.2 第 2 步 点击Add按钮启动新建发布对话框并完成发布	82
6.6 应用服务器的管理和调试	83
6.6.1 启动服务器	83
6.6.2 监控服务器启动过程	83
6.6.3 停止服务器	83
6.6.4 调试发布的企业应用	84
6.7 小结	84
第七章 开发Hibernate应用	85
7.1 介绍	85
7.2 Hibernate 一览	85
7.3 准备工作	86
7.4 创建 HibernateDemo 项目	86
7.4.1 创建表格	86
7.4.2 创建 HibernateDemo Java Project	87
7.4.3 添加 Hibernate Capabilities 到现有项目	88
7.4.4 使用Hibernate配置文件编辑器修改文件	92

7.4.5 使用反向工程快速生成Java POJO类，映射文件和DAO	94
7.4.6 调整生成的hbm文件	96
7.4.7 编写测试代码	97
7.5 MyEclipse Hibernate工具的高级部分	98
7.5.1 反向工程向导的完整说明	98
7.5.2 使用HQL编辑器	101
7.6 小结	103
7.7 参考资料	104
第八章 开发Web应用	105
8.1 介绍	105
8.2 Web项目和术语	105
8.2.1 Java EE 中的Web项目结构	105
8.2.2 MyEclipse Web 项目介绍	107
8.3 创建Web项目	107
8.4 创建HTML页面	109
8.5 创建JSP页面	110
8.6 创建Servlet	111
8.7 创建Filter(过滤器)	114
8.8 创建数据库访问层(DAO)	117
8.9 修改Servlet调用后台类	120
8.10 发布，重新发布，运行和测试应用	121
8.11 调试JSP应用	122
8.12 向现有Web项目添加Web开发能力	123
8.13 高级设置	123
8.13.1 修改Web项目的默认设置	123
8.13.2 给Web项目加入高级功能	124
8.14 小结	124
第九章 开发Struts 1.x应用	125
第十章 开发JSF应用	125
第十一章 开发XFire Web Service应用	125
第十二章 开发JPA应用	125
第十三章 开发Spring应用	125
第十四章 开发Spring+Struts+Hibernate应用	125
第十五章 MyEclipse UML 开发	125
第十六章 开发EJB 应用	125
附录	126

介 绍

Eclipse，日蚀也，日月无光是也！MyEclipse，吾之日月无光乎！这些都是望文生义。

MyEclipse 6.0 是现今国内企业流行的基于Eclipse的商业开发工具 MyEclipse的当前最新版本。Eclipse（官方网站：<http://www.eclipse.org>）是IBM公司主导下的一款开源免费的可以做基础Java项目开发的工具，然而大多数基于Eclipse二次开发的实用开发工具例如MyEclipse, IBM WSAD, BEA Workshop, Jbuilder 2007 等等都是商业产品，有别于Eclipse自身开放免费的大旗，这些软件不能免费使用，例如MyEclipse 6.0 只有 30 天的试用期，过期之后需要付费使用。因为Java开发工具领域的四分五裂，至今仍然没有一款IDE（Integrated Development Environment）可以真正媲美微软的Visual Studio 系列。

MyEclipse 6.0 集中了开源和商业软件的开发支持的大多数框架，方便易用，功能强大，获得了广大开发人员的喜爱。用它来开发比自己用 Eclipse 然后到处找插件安装要方便快捷的多。它支持开发基于 Spring, Hibernate, Struts, JSF, JPA, EJB, Web Service 等 Java EE 技术的项目。本书就如何使用 MyEclipse 开发 Java EE 应用进行简要的介绍，部分内容基于本人翻译的 MyEclipse 帮助文档。因为作者的水平有限，本书不可能涵盖 MyEclipse 或者 Eclipse 的方方面面，仅供初学者作为开发时的参考书来使用。

文档说明

版本	日期	作者	说明
1.0	2007 年 12 月	刘长炯	第一版

适用的读者

本书适用于希望了解如何使用 MyEclipse 6 进行 Java EE 开发的 Java 初学者。如果有一定 Java 语言基础或者 Eclipse 使用经验，对阅读本书有很大帮助。

衷心希望本书能对一些人有所帮助！

关于作者

刘长炯，暂住中国北京，毕业于中国西北某具有军事背景的院校。曾任Synnex China 公司系统架构师。从 2001 年起一直专著于Java方向的学习和开发。所维护的Java博客<http://www.blogjava.net/beansoft/> 曾获得 2007 年 12 月《程序员》杂志的编辑推荐。

联系方式：beansoft@126.com

版权声明

本文档版权归作者刘长炯所有，仅供个人研究和学习之用，不得用于任何商业目的。在免费、且无任何附加条件的前提下，可在网络媒体中自由传播。未经作者书面许可，不得以其他任何方式进行出版、篡改、编辑。

未经作者书面许可，任何商业培训机构不得使用本电子书作为培训教程,否则将依法追究其法律责任。

如需部分或者全文引用，请事先征求作者意见。

如果发现文中有错误的地方，欢迎将页码和出错的地方反馈给我；也期望您能对本书提出修改意见。

第一章 安装配置开发环境

本章内容供你来了解 Java 和数据库软件的常见下载地址，安装和运行，当然也可以直接跳到 MyEclipse ALL In ONE 一节进行学习，暂时先不用关心这些细节的内容。

如果是在 Windows 下安装 MyEclipse，可以不用单独下载和配置 JDK, Eclipse 3.3, 您可以直接参考 MyEclipse ALL In ONE 的安装说明。如果未加说明，本文的操作均在 Windows XP 操作系统下进行。

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 0: 下载 安装 运行 HelloWorld](#), [MyEclipse 6 实战开发讲解视频入门 1 安装运行 Mysql, MySQL-Front 管理, JDBCHelloWorld 开发](#)。

1.1 系统需求

如果只是安装 JDK 和 MySQL 等数据库，只需要 256MB 内存的电脑就可以了。反过来，如果要安装并使用 MyEclipse 6，那么建议电脑配置至少 512MB 内存，推荐 1G 或者更多内存，因为 MyEclipse 启动后经常会占用 200 多 MB 的常驻内存。安装后所占据的空间有 600 MB，因此建议磁盘上至少有 1G 空闲空间。换句话说，做 Java 开发需要大量的内存和磁盘空间。

1.2 JDK 的下载，安装和配置（可选）

注意：如果安装 MyEclipse ALL In ONE 版本，因为它自带了 JRE，不需要单独下载和安装 JDK，也可以进行开发；但是因为 JRE 不带 Java 类的源代码，因此不安装 JDK 将无法看到 JDK 类的源代码。

1.2.1 下载 JDK

JDK 的全称是 **Java(TM) SE Development Kit**，即 Java 标准版（**Standard Edition**）开发工具包。这是 Java 开发和运行的基本平台。换句话说所有用 Java 语言编写的程序要运行都离不开它，而用它就可以编译 Java 代码为类文件。

注意：不要下载 JRE（Java Runtime Environment, Java 运行时环境），因为 JRE 不包含 Java 编译器和 JDK 类的源码。

下载JDK可以访问官方网站 <http://java.sun.com/javase/downloads/index.jsp>，一般来说下载最新版本即可，目前的稳定版本是JDK 6。打开下载页后，首先点击页面中的 **Download** 按钮。如下所示：

图 1.1 JDK 下载页面并点击 **Download** 按钮

接着在可能弹出的浏览器安全连接警告中选择“**确定**”按钮继续，这时候来到下载页面。
要下载 JDK，必须接受下载协议：

Java(TM) SE Development Kit 6 Update 3

NOTE: These products are offered as either a single large file or to multiple platforms - please be sure to download the proper file(s) for your platform. We highly recommend using Sun Download Manager (SDM), as it provides a successful download experience. Just select the files you want to download and click the button to automatically install and start SDM. Alternately, click directly on the download links. For any download problems or questions, please see the Download Help page. How long will the download take?

Required: You must accept the license agreement to download the software.

☐ **Accept License Agreement** | [Review License Agreement](#)

☐ **Decline License Agreement**

图 1.2 点击 **Accept** 按钮接受下载协议

点击单选按钮 **Accept** 后就可以下载 JDK 的安装文件了。首先有必要了解一下供下载的文件列表：

Windows Platform – Java™ SE Development Kit 6 Update 3			
<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Download the full version as a single file .			
<input type="checkbox"/>	Windows Offline Installation, Multi-language 下载这个就可以了！	jdk-6u3-windows-i586-p.exe	65.64 MB

<input type="checkbox"/>	Windows Online Installation, Multi-language	jdk-6u3-windows-i586-p-iftw.exe	373.39 KB
--------------------------	---	---------------------------------	-----------

Linux Platform – Java™ SE Development Kit 6 Update 3

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
<input type="checkbox"/>	Linux RPM in self-extracting file	jdk-6u3-linux-i586-rpm.bin	61.64 MB
<input type="checkbox"/>	Linux self-extracting file	jdk-6u3-linux-i586.bin	65.40 MB

Solaris SPARC Platform – Java™ SE Development Kit 6 Update 3

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
<input type="checkbox"/>	Solaris SPARC 32-bit self-extracting file	jdk-6u3-solaris-sparc.sh	69.99 MB
<input type="checkbox"/>	Solaris SPARC 32-bit packages – tar.Z	jdk-6u3-solaris-sparc.tar.Z	116.45 MB
<input type="checkbox"/>	Solaris SPARC 64-bit self-extracting file	jdk-6u3-solaris-sparcv9.sh	10.69 MB
<input type="checkbox"/>	Solaris SPARC 64-bit packages – tar.Z	jdk-6u3-solaris-sparcv9.tar.Z	13.58 MB

Solaris x86 Platform – Java™ SE Development Kit 6 Update 3



<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
<input type="checkbox"/>	Solaris x86 self-extracting file	jdk-6u3-solaris-i586.sh	64.47 MB
<input type="checkbox"/>	Solaris x86 packages – tar.Z	jdk-6u3-solaris-i586.tar.Z	110.51 MB

Solaris x64 Platform – Java™ SE Development Kit 6 Update 3

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
<input type="checkbox"/>	Solaris x64 self-extracting file	jdk-6u3-solaris-amd64.sh	7.19 MB
<input type="checkbox"/>	Solaris x64 packages – tar.Z	jdk-6u3-solaris-amd64.tar.Z	10.21 MB

Linux x64 Platform – Java™ SE Development Kit 6 Update 3

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
-------------------------------------	-------------------------------------	--	--

	Linux x64 RPM in self-extracting file	jdk-6u3-linux-amd64-rpm.bin	56.16 MB
	Linux x64 self-extracting file	jdk-6u3-linux-amd64.bin	59.26 MB



Windows x64 Platform – Java™ SE Development Kit 6 Update 3			
			
	Windows x64 executable	jdk-6u3-windows-amd64.exe	38.63 MB

图 1.3 JDK 文件下载列表

首先我们看到 JDK 支持多个主流操作系统和硬件平台的安装，包括 Windows, Linux, Solaris 这些是操作系统软件的版本。而每个平台又区分了针对不同的硬件环境的（主要是 CPU 的），x86 就是一般的家用电脑的 32 位 CPU，例如 Intel 和 AMD 的；x64 则是 64 位 CPU，一般用在服务器上。因此，我们只要关注 **Windows x86** 版本的就可以了，如图 3 的第一个单元格所示：

Windows Platform – Java™ SE Development Kit 6 Update 3

在这个类别下又有两个版本的安装程序。第一个名为 **Windows Offline Installation, Multi-language** 的是 Windows 完整离线安装包，支持多国语言的版本，个头比较大，一般用户点击链接下载这个版本的就可以了。而下面的那个很小的 **Windows Online Installation, Multi-language**，则是需要在线安装的，装的时候电脑必须上网才可以，鉴于一般用户的电脑网速并不快，因此不推荐使用。点击下载链接后保存文件到硬盘上即可，例如这里我们可以下载 **jdk-6u3-windows-i586-p.exe**。

Linux 下面的文件版本呢，又分为 **executable**（可执行）和 **self-extracting**（自解压）两种安装类型，下载后不要忘了用 `chmod +x xxx.bin` 来给文件加执行权限后再安装。具体的细节限于篇幅暂时不讨论了。

1.2.2 安装 JDK

双击下载后的带有  图标的 JDK 安装程序 EXE 文件，接着就会使用 Windows Installer 开始安装过程，如下图所示：

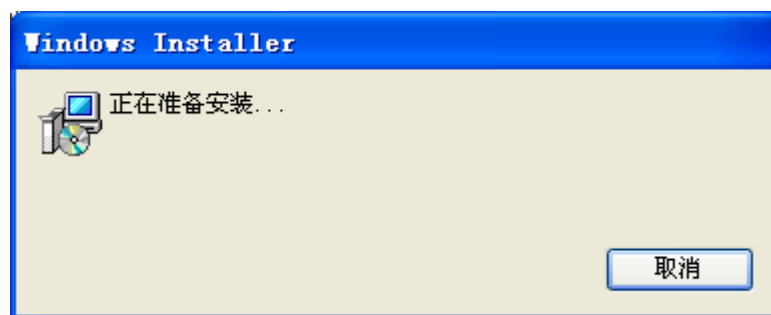


图 1.4 启动 JDK 安装程序

如果这一过程失败，请下载并安装最新版本的 Windows Installer 后再试，并检查电脑的 Windows Installer 服务已经启动(下载地址：[Windows Installer 3.1 Redistributable \(v2\)](#))。接

下来的安装界面是中文的，点击下一步或者继续按钮，当出现许可证协议对话框时点击**接受(A)>**按钮方可继续安装。这时候将会出现**自定义安装**对话框，如下图所示：



图 1.5 自定义安装对话框

在这个屏幕我们可以看到默认安装路径是到 `c:\Program Files\Java\jdk1.x.x_xx`，然而，这个安装路径需要进行修改，点击**更改(A)...**按钮来修改 JDK 的安装目录，点击后输入类似于 `C:\jdk1.6.0` 这样的不包含空格，也不包含中文路径的文件夹来安装。而这样的路径是不推荐的：`C:\Java 学习\JDK 1.6`。之所以这样做是因为路径带空格后有时候会出现不必要的问题，导致某些 Java 程序运行失败，也会在以后设置 PATH 和 CLASSPATH 时出现一些问题。现在你需要记下来安装的路径例如 `C:\jdk1.6.0`，然后接着点**下一步**按钮等待片刻就可以完成安装了。

1.2.3 配置环境变量（可选）

这一步呢，也不是必须的，如果打算使用 MyEclipse 来进行开发，而不是手工编译代码，可以完全忽略这一节的内容。

第一个需要配置的环境变量是 **JAVA_HOME**。在**我的电脑**上点击右键，选择**属性**，在弹出的对话框中选择**高级**页面，然后点击**环境变量**按钮，在出现的**环境变量**对话框的**系统变量(S)**栏目中点击**新建**按钮，出现新建系统环境变量的对话框，输入变量名为 **JAVA_HOME**，值为 JDK 安装目录，例如：`C:\JDK1.6.0`（例如 Tomcat 需要这个环境变量来查找 JDK）。如下图所示：

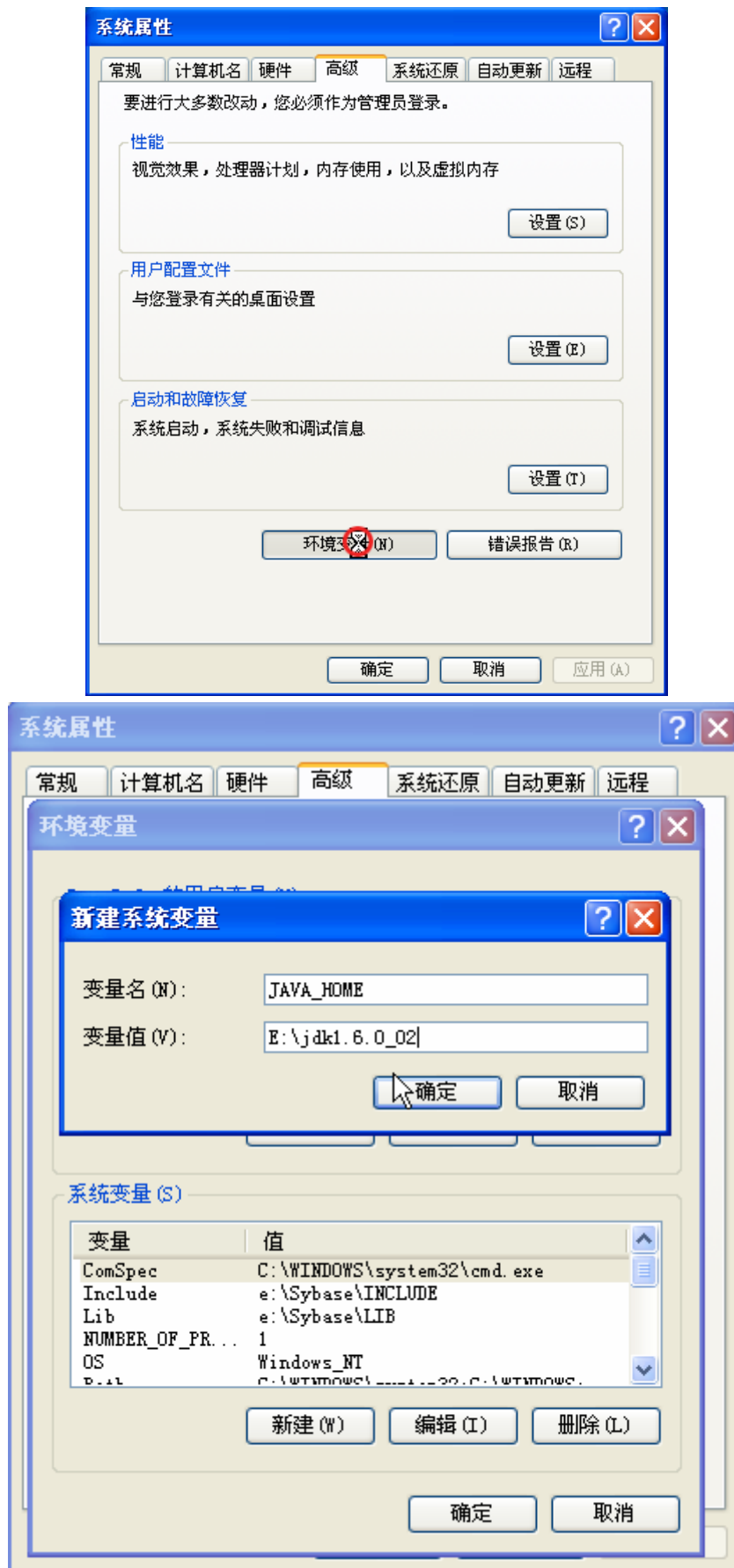


图 1.6 新建系统变量

接下来用类似的方法新建环境变量 **CLASSPATH**，取值为 `.`（**注意：**是英文的`.`），这个变量用来供 **Java** 虚拟机查找要加载的类。接下来需要把 **JDK** 的应用程序路径添加到系统的 **Path** 变量中，点击滚动条找到列表中名为 **Path** 的变量，点击“编辑(E)”按钮，即可修改 **PATH** 的变量值。一般来说我们只需要在开头加 `%JAVA_HOME%\bin;`（**注意**不要用中文全角的`;`），然后点击两次**确定**按钮即可。如下图所示：

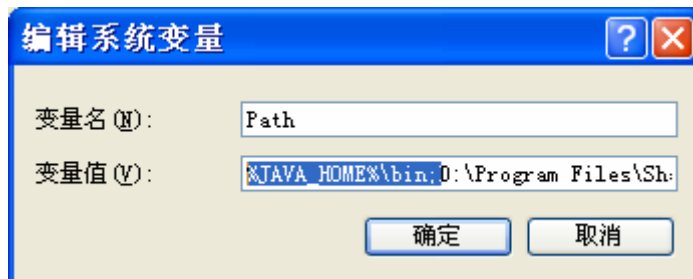


图 1.7 修改 Path 系统变量

注意：用户变量和系统变量的区别是用户变量只对 Windows 的当前登录用户可用，而系统变量则是对所有的用户都有影响。

当这些变量设置完毕后，就可以在命令行里面检查是否设置成功。点击 Windows 的**开始**按钮，选择**运行(R)...**，输入 `CMD` 后按下回车，这时候会出现命令行窗口。输入 `javac` 并按下回车（Enter）键，如果能看到如下的输出，则环境变量已经配置成功：

```
C:\Documents and Settings\BeanSoft>javac
用法: javac <选项> <源文件>
其中，可能的选项包括：
    -g                生成所有调试信息
    -g:none           不生成任何调试信息
    -g: {lines, vars, source} 只生成某些调试信息
    -nowarn           不生成任何警告
    -verbose          输出有关编译器正在执行的操作的消息
    -deprecation      输出使用已过时的 API 的源位置
    -classpath <路径> 指定查找用户类文件和注释处理程序的位置
    -cp <路径>       指定查找用户类文件和注释处理程序的位置
    ...
```

图 1.8 配置成功后 `javac` 命令的输出

反过来如果发现输出 `'javac'` 不是内部或外部命令，也不是可运行的程序或批处理文件。这样的输出信息，则说明环境变量配置失败了，请仔细检查设置的步骤。

这之后你就可以用记事本来编写 **Java** 文件然后用命令行的方式来编译和运行了。

1.2.4 JDK 6 中文文档下载地址(ZIP,HTML,CHM)（可选）

JDK 的中文 **API** 文档有助于理解和学习 **Java** 语言的基础，但是从长远看还是希望读者能逐渐熟悉阅读英文的 **Javadoc**。下载 **CHM** 格式就可以了，阅读起来比较方便，还可以搜索。

目前在 <http://developers.sun.com.cn> 已正式宣布发布 **Java SE 6 API** 中文版。

大家也可以从以下网址下载：

* HTML 格 式
(http://download.java.net/jdk/jdk-api-localizations/jdk-api-zh-cn/publish/1.6.0/html/zh_CN/)

[api/index.html](#))

* zip 格式
(http://download.java.net/jdk/jdk-api-localizations/jdk-api-zh-cn/publish/1.6.0/html_zh_CN.zip)

* CHM 格式
(http://download.java.net/jdk/jdk-api-localizations/jdk-api-zh-cn/publish/1.6.0/chm/JDK_API_1_6_zh_CN.CHM)

官方论坛地址 (Sun 中国社区):

<http://gceclub.sun.com.cn/NASApp/sme/jive/thread.jsp?forum=35&thread=44422>

1.3 Tomcat 服务器的下载，安装和运行(可选)

因为 MyEclipse 6 已经内置了一个简易的 Tomcat 6(MyEclipse Tomcat)，所以本节内容为可选操作，但是强烈推荐了解本节内容。

Tomcat是一款开源免费的JSP服务器，可以在 <http://tomcat.apache.org/> 下载并安装 Tomcat 5 或者 6。

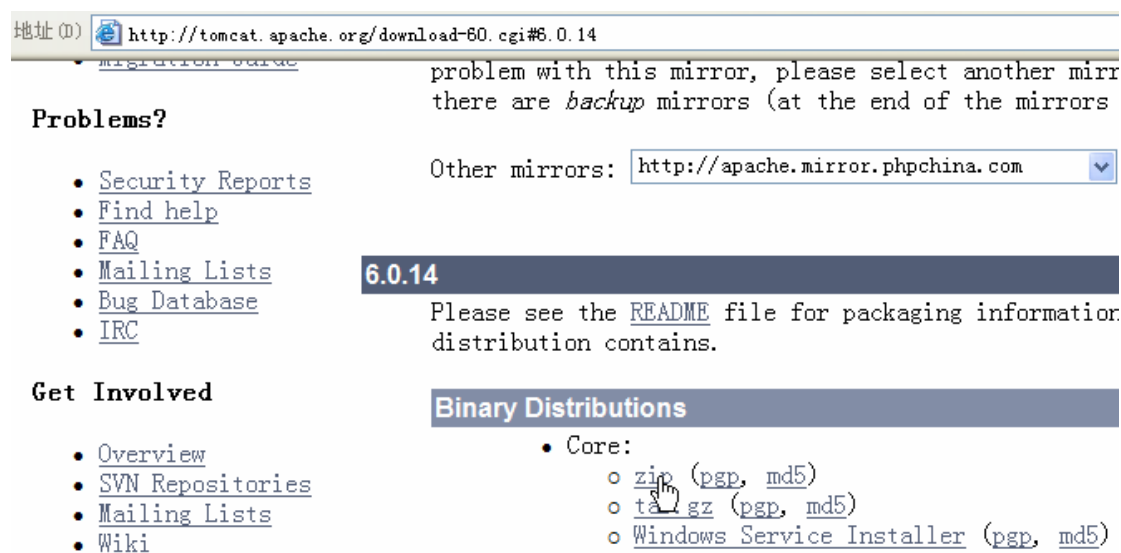


图 1.9 下载 Tomcat

建议下载压缩包版本（文件名是 **apache-tomcat-6.0.14.zip**），而不是 Windows Service Installer 的 EXE 安装文件。解压缩到磁盘目录，记下安装路径例如 **c:\apache-tomcat-6.0.14**，和 JDK 的安装一样，为了避免日后产生错误，解压缩的路径不要带有多余空格，如 *Program Files*。用解压缩工具来解压缩下载下来的 ZIP 格式的压缩包的时候（例如 WinRAR）千万不要解压缩成了 **c:\apache-tomcat-6.0.14\apache-tomcat-6.0.14** 这样的安装路径。

在 Windows 下面不需要设置 **CATALINA_HOME** 这个变量也可以运行 Tomcat，如果你配置了这个变量，那么你的电脑上将永远只能启动设置了 **CATALINA_HOME** 的那个 Tomcat，换句话说如果你想多个 Tomcat 版本并存，就不能设置 **CATALINA_HOME**。而使用 MyEclipse 进行开发的时候，也不需要这个变量。如果你想设置，就新建环境变量，名为 **CATALINA_HOME**，取值为 Tomcat 的安装目录例如 **c:\apache-tomcat-6.0.14**。

进入 Tomcat 安装目录下的 bin 子目录，可以看到 **startup.bat** 和 **shutdown.bat**。双击 **startup.bat** 启动 Tomcat 服务器，将产生如下的输出信息：

```

信息: The Apache Tomcat Native library which allows optimal performance in produ
ction environments was not found on the java.library.path: D:\Java\jdk1.7.0\bin;
.;C:\WINXP\Sun\Java\bin;C:\WINXP\system32;C:\WINXP;%JAVA_HOME%\bin;C:\oracle\ora
92\bin;C:\WINXP\system32;C:\WINXP;C:\WINXP\System32\Wbem;%JAVA_HOME%\bin;E:\_Por
tableJava\jdk1.6.0\bin;C:\oracle\ora92\bin;C:\WINXP\system32;C:\WINXP;C:\WINXP\S
ystem32\Wbem;E:\_PortableJava\jdk1.6.0\bin;C:\oracle\ora92\bin;C:\WINXP\system32
;C:\WINXP;C:\WINXP\System32\Wbem;E:\_PortableJava\apache-ant-1.6.2\bin;E:\_Porta
bleApps\SSH
2007-12-4 15:22:08 org.apache.coyote.http11.Http11Protocol init
信息: Initializing Coyote HTTP/1.1 on http-8080
2007-12-4 15:22:08 org.apache.catalina.startup.Catalina load
信息: Initialization processed in 2049 ms
2007-12-4 15:22:08 org.apache.catalina.core.StandardService start
信息: Starting service Catalina
2007-12-4 15:22:08 org.apache.catalina.core.StandardEngine start
信息: Starting Servlet Engine: Apache Tomcat/6.0.14
2007-12-4 15:22:13 org.apache.coyote.http11.Http11Protocol start
信息: Starting Coyote HTTP/1.1 on http-8080
2007-12-4 15:22:13 org.apache.jk.common.ChannelSocket init
信息: JK: ajp13 listening on /0.0.0.0:8009
2007-12-4 15:22:13 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/46  config=null
2007-12-4 15:22:13 org.apache.catalina.startup.Catalina start
信息: Server startup in 4859 ms

```

图 1.10 启动 Tomcat 服务器

当看到出现 *信息: Server startup in 4859 ms* 的输出后, Tomcat 就启动完毕了。反之则可能出现错误, 无法启动。要关闭 Tomcat 服务器, 可以关闭这个 CMD 窗口, 也可以运行 shutdown.bat。

接着在浏览器中键入 <http://localhost:8080/> 来测试是否运行成功。如下图所示:

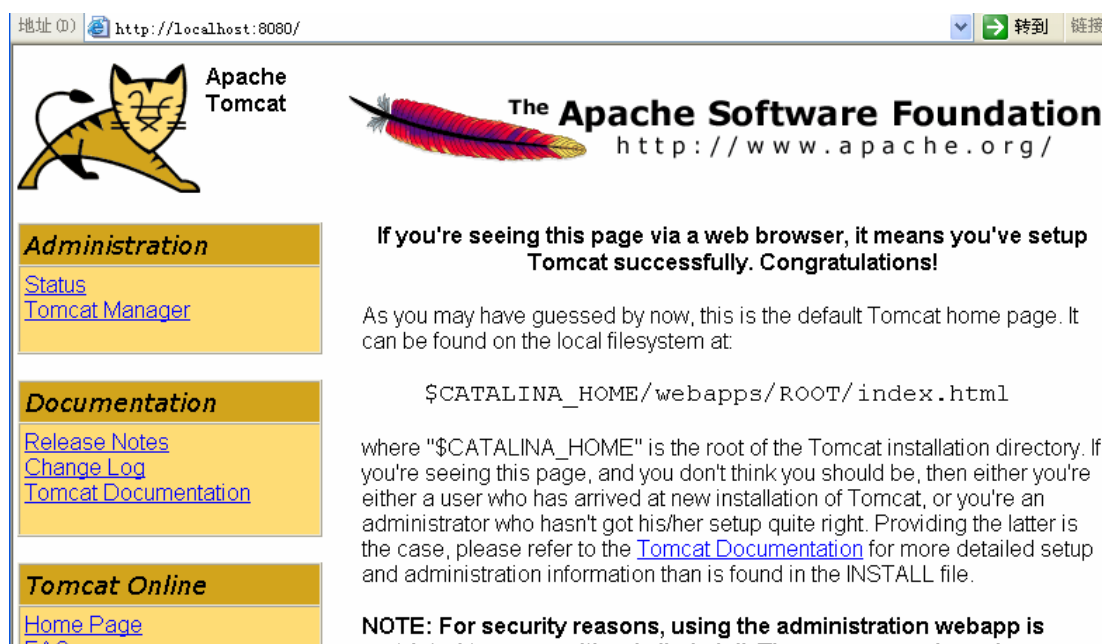


图 1.11 Tomcat 的欢迎页面

注意：有的时候您可能想修改 Tomcat 的默认监听端口，请用文本编辑器打开 *Tomcat* 安装目录/conf/server.xml，找到如下的定义：

<Connector port="8080" ...，替换 8080 为你想要的端口即可。假设改成 80，就可以省略端口这样访问：

<http://localhost/> 否则是 <http://localhost:新端口/> 或者 <http://127.0.0.1:新端口/>。Localhost或者 127.0.0.1 是个特殊的网络地址，它就代表你本机的地址。

1.4 JBoss 服务器的下载，安装和运行（可选）

注意：如果你不想开发 EJB 应用，本节内容不需要进行。

如果你没有安装 JDK，JBoss 将无法单独启动，但可以在 MyEclipse 里面正常启动。

JBoss 是一款开源免费的支持 EJB 3.0 和 JSP，JMS 等的应用服务器。<http://labs.jboss.com/jbossas/downloads> 任何一个 4.2 或者 5.0 版本均可。这里推荐您点击链接 <http://downloads.sourceforge.net/jboss/jboss-4.2.2.GA.zip> 直接下载。下载完毕后得到一个ZIP文件，例如jboss-4.2.2.GA.zip，解压缩到任何不含空格的目录即可，例如 c:\jboss-4.2.2.GA。在Windows下不需要像某些文章所说的需要配置JBOSS_HOME变量。要启动 JBoss 进入 JBoss 下面的 bin 子目录双击 run.bat 即可，例如 c:\jboss-4.2.2.GA\bin\run.bat，启动后将会产生下面的输出日志：

```
=====
JBoss Bootstrap Environment

JBOSS_HOME: C:\jboss-4.2.2.GA

JAVA: D:\Java\jdk1.7.0\bin\java

JAVA_OPTS: -Dprogram.name=run.bat -server -Xms128m -Xmx512m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000

CLASSPATH: D:\Java\jdk1.7.0\lib\tools.jar;C:\jboss-4.2.2.GA\bin\run.jar
```

```

=====
09:13:48,015 INFO [Server] Starting JBoss (MX MicroKernel)...
09:13:48,015 INFO [Server] Release ID: JBoss [Trinity] 4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)
09:13:48,015 INFO [Server] Home Dir: C:\jboss-4.2.2.GA
09:13:48,015 INFO [Server] Home URL: file:/C:/jboss-4.2.2.GA/
09:13:48,031 INFO [Server] Patch URL: null
09:13:48,031 INFO [Server] Server Name: default
09:13:48,031 INFO [Server] Server Home Dir: C:\jboss-4.2.2.GA\server\default
09:13:48,031 INFO [Server] Server Home URL: file:/C:/jboss-4.2.2.GA/server/default/
09:13:48,031 INFO [Server] Server Log Dir: C:\jboss-4.2.2.GA\server\default\log
09:13:48,031 INFO [Server] Server Temp Dir: C:\jboss-4.2.2.GA\server\default\temp
09:13:48,031 INFO [Server] Root Deployment Filename: jboss-service.xml
09:13:48,984 INFO [ServerInfo] Java version: 1.7.0-ea, Sun Microsystems Inc.
09:13:48,984 INFO [ServerInfo] Java VM: Java HotSpot(TM) Server VM 11.0-b08, Sun Microsystems Inc.
09:13:48,984 INFO [ServerInfo] OS-System: Windows XP 5.1,x86
09:13:49,703 INFO [Server] Core system initialized
09:13:52,359 INFO [WebService] Using RMI server codebase: http://127.0.0.1:8083/
09:13:52,359 INFO [Log4jService$URLWatchTimerTask] Configuring from URL: resource:jboss-log4j.xml
09:13:53,390 INFO [TransactionManagerService] JBossTS Transaction Service (JTA version) - JBoss Inc.
09:13:53,390 INFO [TransactionManagerService] Setting up property manager MBean and JMX layer
09:13:53,937 INFO [TransactionManagerService] Starting recovery manager
09:13:54,031 INFO [TransactionManagerService] Recovery manager started
09:13:54,031 INFO [TransactionManagerService] Binding TransactionManager JNDI Reference
09:13:57,281 INFO [EJB3Deployer] Starting java:comp multiplexer
09:13:57,593 INFO [STDOUT] no object for null
09:13:57,593 INFO [STDOUT] no object for null
09:13:57,609 INFO [STDOUT] no object for null
09:13:57,640 INFO [STDOUT] no object for {urn:jboss:bean-deployer}supplyType
09:13:57,640 INFO [STDOUT] no object for {urn:jboss:bean-deployer}dependsType
09:14:00,031 INFO [NativeServerConfig] JBoss Web Services - Native
09:14:00,031 INFO [NativeServerConfig] jbosswebs-native-2.0.1.SP2 (build=200710210837)
09:14:00,953 INFO [Embedded] Catalina naming disabled
09:14:01,156 INFO [AprLifecycleListener] The Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: D:\Java\jdk1.7.0\bin;.;C:\WINXP\Sun\Java\bin;C:\WINXP\system32;C:\WINXP;%JAVA_HOME%\bin;C:\oracle\ora92\bin;C:\WINXP\system32;C:\WINXP;C:\WINXP\system32\Wbem;%JAVA_HOME%\bin;E:\_PortableJava\jdk1.6.0\bin;C:\oracle\ora92\bin;C:\WINXP\system32;C:\WINXP;C:\WINXP\System32\Wbem;E:\_PortableJava\jdk1.6.0\bin;C:\oracle\ora92\bin;C:\WINXP\system32;C:\WINXP;C:\WINXP\System32\Wbem;E:\_PortableJava\apache-ant-1.6.2\bin;E:\_PortableApps\SSH
09:14:01,234 INFO [Http11Protocol] Initializing Coyote HTTP/1.1 on http-127.0.0.1-8080
09:14:01,234 INFO [AjpProtocol] Initializing Coyote AJP/1.3 on ajp-127.0.0.1-8080
09:14:01,250 INFO [Catalina] Initialization processed in 285 ms
09:14:01,250 INFO [StandardService] Starting service jboss.web
09:14:01,250 INFO [StandardEngine] Starting Servlet Engine: JBossWeb/2.0.1.GA
09:14:01,312 INFO [Catalina] Server startup in 70 ms
09:14:01,453 INFO [TomcatDeployer] deploy, ctxPath=/, warUrl=.../deploy/jboss-web-deployer/ROOT.war/
09:14:02,296 INFO [TomcatDeployer] deploy, ctxPath=/invoker, warUrl=.../deploy/http-invoker.sar/invoker.war/
09:14:02,421 INFO [TomcatDeployer] deploy, ctxPath=/jbosswebs, warUrl=.../deploy/jbosswebs.sar/jbosswebs-context.war/
09:14:02,546 INFO [TomcatDeployer] deploy, ctxPath=/jbossmq-httpil, warUrl=.../

```

```

deploy/jms/jbossmq-httpil.sar/jbossmq-httpil.war/
09:14:03,359 INFO [TomcatDeployer] deploy, ctxPath=/web-console, warUrl=.../dep
loy/management/console-mgr.sar/web-console.war/
09:14:03,781 INFO [MailService] Mail Service bound to java:/Mail
09:14:03,953 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/jboss-ha-local-jdbc.rar
09:14:04,000 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/jboss-ha-xa-jdbc.rar
09:14:04,062 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/jboss-local-jdbc.rar
09:14:04,109 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/jboss-xa-jdbc.rar
09:14:04,203 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/jms/jms-ra.rar
09:14:04,328 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/mail-ra.rar
09:14:04,390 INFO [RARDeployment] Required license terms exist, view META-INF/r
a.xml in .../deploy/quartz-ra.rar
09:14:04,390 INFO [QuartzResourceAdapter] start quartz!!!
09:14:04,484 INFO [SimpleThreadPool] Job execution threads will use class loade
r of thread: main
09:14:04,515 INFO [QuartzScheduler] Quartz Scheduler v.1.5.2 created.
09:14:04,515 INFO [RAMJobStore] RAMJobStore initialized.
09:14:04,515 INFO [StdSchedulerFactory] Quartz scheduler 'DefaultQuartzSchedule
r' initialized from default resource file in Quartz package: 'quartz.properties'

09:14:04,515 INFO [StdSchedulerFactory] Quartz scheduler version: 1.5.2
09:14:04,515 INFO [QuartzScheduler] Scheduler DefaultQuartzScheduler_$_NON_CLUS
TERED started.
09:14:05,968 INFO [ConnectionFactoryBindingService] Bound ConnectionManager 'jb
oss.jca:service=DataSourceBinding,name=DefaultDS' to JNDI name 'java:DefaultDS'
09:14:06,609 INFO [A] Bound to JNDI name: queue/A
09:14:06,625 INFO [B] Bound to JNDI name: queue/B
09:14:06,625 INFO [C] Bound to JNDI name: queue/C
09:14:06,625 INFO [D] Bound to JNDI name: queue/D
09:14:06,625 INFO [ex] Bound to JNDI name: queue/ex
09:14:06,656 INFO [testTopic] Bound to JNDI name: topic/testTopic
09:14:06,656 INFO [securedTopic] Bound to JNDI name: topic/securedTopic
09:14:06,656 INFO [testDurableTopic] Bound to JNDI name: topic/testDurableTopic

09:14:06,656 INFO [testQueue] Bound to JNDI name: queue/testQueue
09:14:06,703 INFO [UILServerILService] JBossMQ UIL service available at : /127.
0.0.1:8093
09:14:06,750 INFO [DLQ] Bound to JNDI name: queue/DLQ
09:14:06,843 INFO [ConnectionFactoryBindingService] Bound ConnectionManager 'jb
oss.jca:service=ConnectionFactoryBinding,name=JmsXA' to JNDI name 'java:JmsXA'
09:14:06,875 INFO [TomcatDeployer] deploy, ctxPath=/jmx-console, warUrl=.../dep
loy/jmx-console.war/
09:14:07,203 INFO [Http11Protocol] Starting Coyote HTTP/1.1 on http-127.0.0.1-8
080
09:14:07,375 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-127.0.0.1-8009
09:14:07,390 INFO [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBos
s_4_2_2_GA date=200710221139)] Started in 19s:344ms

```

图 1.12 启动 JBoss 服务器

当看到 **Started in 33s:188ms** 之类的信息后，服务器即启动完毕了，否则就是出错或者启动失败了。

接着在浏览器中键入 <http://localhost:8080/> 来测试是否运行成功。如下图所示：



图 1.13 JBoss 4.2 的欢迎页面

注意：JBoss 中也已经包含了 JSP 服务器功能，而且它监听的端口也是 8080，所以 Tomcat 和 JBoss 是不能同时在一台电脑启动的。默认情况下 JBoss 只监听 localhost 的请求，如果要想让局域网的电脑访问 JBoss 服务，请在命令行下用下面的参数来运行：

```
run.bat -b 0.0.0.0
```

1.5 MySQL 5 数据库服务器下载，安装和运行（可选）

因为 MyEclipse 6 自带了一款嵌入式 Java 数据库 Derby(MyEclipse Derby)，足够开发使用，因此本节内容也是可选的。

MySQL 是一款用的比较广泛的轻量级的免费数据库服务器。

1.5.1 MySQL 5 官方版本的下载和安装，运行

可以访问 MySQL 官方网站下载原版安装程序和 JDBC 驱动，请访问：
<http://dev.mysql.com/downloads/mysql/5.0.html#win32>，如下所示：

Windows Essentials (x86)	5.0.45	22.9M	Pick a mirror
MD5: 9efd5d841174b1476a317e94becf8786			
Windows ZIP/Setup.EXE (x86)	5.0.45	42.4M	Pick a mirror
MD5: 1566ff960b22cda4903e03d4f6cfa205 Signature			
Without installer (unzip in C:\)	5.0.45	50.0M	Pick a mirror
MD5: c40ba57fe2ecb965f9ca88897b6e7d8b Signature			

再这三个版本中选择第一个或者第二个，点击 **Pick a mirror** 后即可下载，接着双击点击 **Next** 或者 **Yes** 按钮安装。安装完毕后服务器即会自动启动，无须每次手工启动。

MySQL的JDBC 驱动下载地址位于 <http://dev.mysql.com/downloads/connector/j/5.1.html>:

Source and Binaries (tar.gz)	5.1.5	7.8M	Pick a mirror
MD5: 85289f74093a2b165d42f5ac38850d18 Signature			
Source and Binaries (zip)	5.1.5	8.0M	Pick a mirror
MD5: b207959597d8974545ef99d5a2cee662 Signature			

任选一个进行下载，解压缩后得到 **mysql-connector-xxx.jar** 就是驱动程序类库了。

1.5.2 MySQL 5 绿色版的下载安装和运行

BeanSoft MySQL Java 开发套装包含 MySQL 5.0 服务器,管理工具,JDBC 驱动和 Java 访问数据库的示例代码。

1.5.2.1 下载

http://gro.clinux.org/frs/download.php/2106/portable_mysql.exe 4.02MB (自解压包)

用法: 下载后解压缩到硬盘的任意位置, 然后双击 **PStart.exe** 开始, 先启动 MySQL 服务器, 然后即可编译运行 JDBC 测试代码。

注意: 这个版本的 MySQL 绿色版默认采用的字符集是 GBK, 如果你修改成了别的字符集, MySQL-Front 将显示为乱码。

1.5.2.2 用法图解

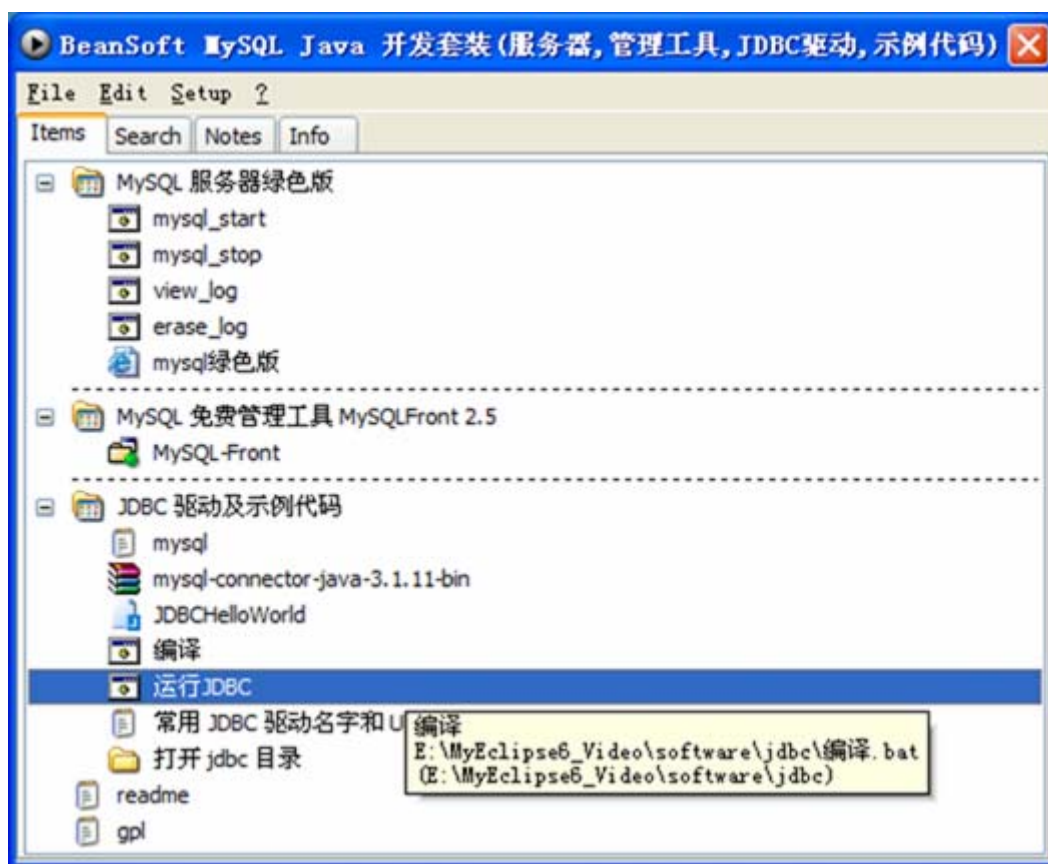


图 1.14 MySQL 绿色版主界面

主界面，双击 **mysql_start** 启动 MySQL 服务器，双击 **mysql_stop** 停止 MySQL 服务器。详细用法双击 网页 *mysql 绿色版* 来了解。

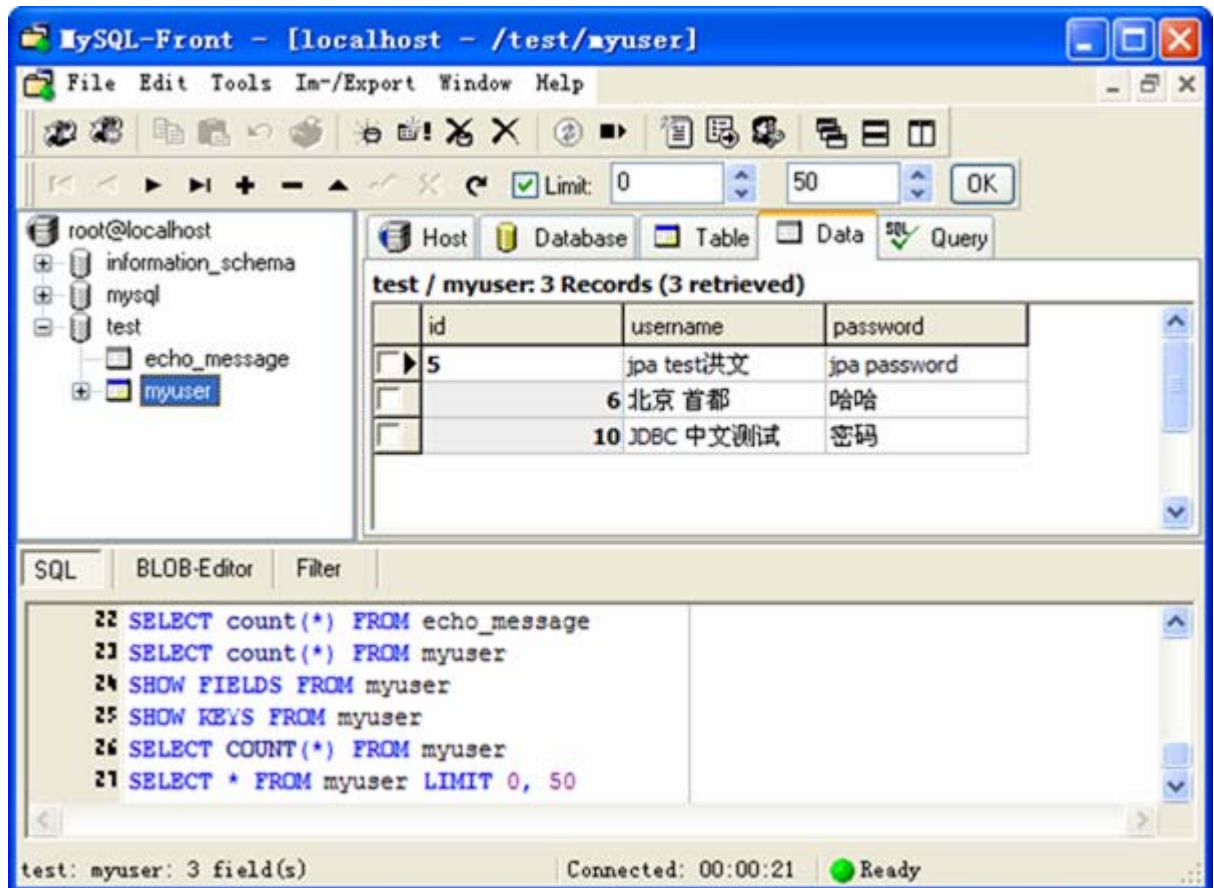


图 1.15 双击主界面的 MySQL-Front 启动管理工具

```

E:\MyEclipse6_Video\software\jdbc>java -cp .;mysql-connector-java-3.1.11-bin.jar
JDBCHelloWorld
5
jpa test 洪文
jpa password

6
北京 首都
哈哈

9
user
password

11
JDBC 中文测试
密码

E:\MyEclipse6_Video\software\jdbc>pause
请按任意键继续...

```

图 1.16 双击主界面的运行 JDBC 进行插入和读取数据测试

1.6 Eclipse 3.3 的下载，安装和运行

Eclipse 是一款基础的，开源免费的Java开发工具，目前比较流行。Eclipse 3.3 可以在 <http://www.eclipse.org/> 下载，进入首页后点击黄色的Download按钮，如下图所示：



图 1.17 Eclipse 首页面

点击后可以看到下载页面中的内容：

Eclipse Europa Fall Maintenance Packages - Windows (compare packages)		
	Eclipse IDE for Java Developers - Windows (78 MB) The essential tools for any Java developer, including a Java IDE, a CVS client, XML Editor and Mylyn. Find out more...	Windows Linux Mac OS X
	Eclipse IDE for Java EE Developers - Windows (126 MB) Tools for Java developers creating JEE and Web applications, including a Java IDE, tools for JEE and JSF, Mylyn and others. Java 5 (or higher) required. Find out more...	Windows Linux Mac OS X
	Eclipse IDE for C/C++ Developers - Windows (63 MB) An IDE for C/C++ developers. Find out more...	Windows Linux Mac OS X
	Eclipse for RCP/Plug-in Developers - Windows (153 MB) A complete set of tools for developers who want to create Eclipse plug-ins or Rich Client Applications. It includes a complete SDK, developer tools and source code. Find out more...	Windows Linux Mac OS X
	Eclipse Classic 3.3.1.1 - Windows (140 MB)  The classic Eclipse download: the Eclipse Platform, Java Development Tools, and Plug-in Development Environment, including source and both user and programmer documentation. Find out more...	Windows Linux Mac OS X All Versions

图 1.18 下载 Eclipse

Eclipse 3.3 分出了几个类型的下载包，第一个是普通的Java开发包，我们下载它就可以了，点击 [Eclipse IDE for Java Developers](#) 就可以下载了。第二个是提供有限的Java EE开发支持的，包括EJB, JSP, JSF的开发；第三个是C/C++的开发包；第四个是专门做插件和RCP（Rich Client Platform，富客户端平台，IBM主推的基于Eclipse的桌面应用开发平台，提供有限的系统底层调用和仿Eclipse外观的界面）开发的；第五个是传统的Eclipse下载包，包括Eclipse平台，Java开发工具和插件开发。

下载后得到一个压缩包 **eclipse-java-europa-fall2-win32.zip**，解压缩到 c:\后会自动得到 c:\eclipse 这个目录，这样就算安装完毕了。

要运行，进入目录 c:\eclipse，双击 **eclipse.exe**，就可以启动并运行 Eclipse 了。启

动过程中会提示你选择 workspace，点击 **OK** 按钮就可以继续启动，如下图所示：

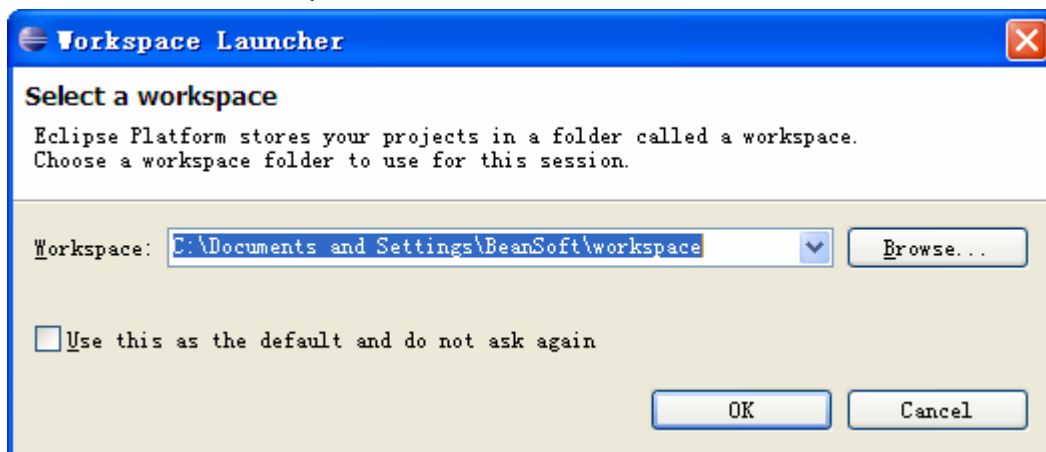



图 1.19 启动时选择 workspace

注意：如果你不希望以后看到这个提示，选中复选框 *Use this as the default and do not ask again* 即可。

第一次启动后主界面还显示一个欢迎页面 (Welcome)，点击上面的  图标关闭欢迎页面，之后可以做一些基础的 Java 应用开发。这时界面如下所示：

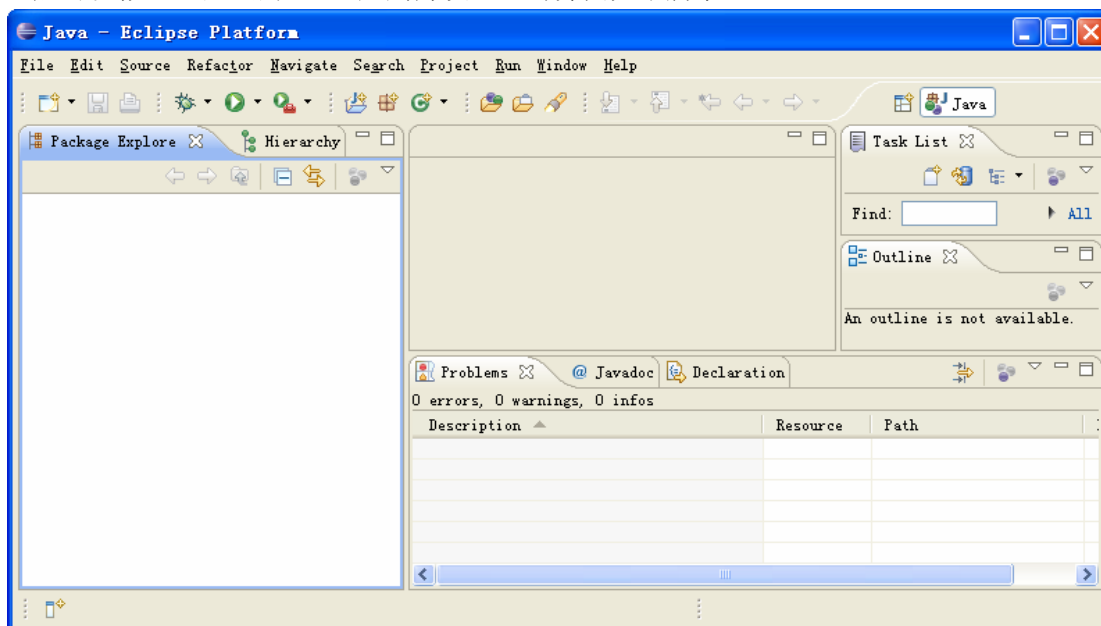


图 1.20 Eclipse 主界面


至此 Eclipse 就算安装完毕了。

1.7 MyEclipse 6 的下载，安装和运行

MyEclipse 6 是一款商业的基于Eclipse的Java EE集成开发工具，换句话说不是免费产品。官方站点是 <http://www.myeclipseide.com/>。

MyEclipse 的安装分为插件版本和 ALL in ONE 版本，其中 ALL in ONE 版本无需自己另外下载安装和配置 JDK，Eclipse 3.3，因此如果你打算已最快的速度装好 MyEclipse，请选择 ALL in ONE 版本。

1.7.1 下载

打开首页后点击页面中的下载按钮：，之后来到MyEclipse 6 的下载页面，需要接受协议然后才能进行下载：

MyEclipse challenges the misconception that good development tools have to be expensive by delivering the most cost-effective and full featured Eclipse-based J2EE IDE on the market today.

You can try MyEclipse free for a 30-day trial membership to test drive the full package and see if it is right for you. Want to learn more? [Register](#) for a FREE MyEclipse Webinar!



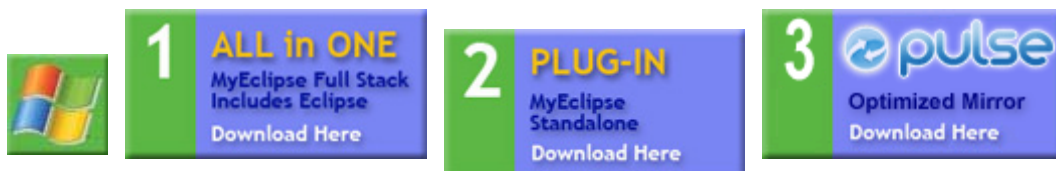
图 1.21 接受 MyEclipse 6 的下载协议

。点击 **DOWNLOAD** 按钮后来到真正的下载页面：

Downloads: Eclipse 3.3 Downloads

1. MyEclipse Enterprise Workbench 6.0.1 GA for Windows 98/2000/NT/XP/Vista (10/16/2007) ★

Description:



MyEclipse Enterprise Workbench 6.0.1 - Windows Edition is now available for download. Make sure to review the [release notes](#). Also make sure Eclipse 3.3.x and JDK 1.5.0_08 or a later release are installed if downloading other than the All-in-one installer. The All-in-One installer is bundled with Eclipse and JRE, and also supplies users with MyEclipse [SNAPs](#). You can download Eclipse Directly from the [Genuitec Mirror site](#) or at Eclipse.org [here](#).

MyEclipse is also available for download through [Pulse](#) - the new and easy way to download and manage all of your Eclipse Installs.

Version: 6.0.1 GA | File size: 176.33 MB

MD5 : For All-in-One : 1eba3b2521e66870c07b9db3d62addc2 | For Windows Plugin: 504fc0aaa1e9b1252773f816e443f9d8

Added on: 16-Oct-2007

图 1.22 下载不同版本的 MyEclipse 6

那么编号为 1 的就是最容易安装的 ALL in ONE 版本；编号为 2 的就是插件（PLUG-IN）版本，这个版本的安装需要你按照前文的叙述分别下载和安装 JDK 以及 Eclipse 3.3；编号为 3 的是 MyEclipse 新推出的基于点对点的自动下载和安装工具。对于初学者来说，我们推荐下载 ALL in ONE，基本上不会出什么问题。分别点击您需要的版本（二者只选其一即可）后即可开始下载过程，因为文件比较大，大约有 200 MB，所以需要耐心等待。而 ALL in ONE 版本则个头更大。

1.7.2 安装

1.7.2.1 ALL in ONE 版本的安装

ALL in ONE 直接双击文件就可以运行，无需选择更多选项(这个下载的文件名可能是 **MyEclipse_6.0.1GA_E3.3.1_FullStackInstaller.exe**)。首先第一个屏幕是欢迎页面，点击 **Next** 按钮继续，这一页显示的是许可协议，点击  **I accept the terms of the License Agreement**，然后点击 **Next** 按钮继续安装，接下来显示的是安装路径，默认是安装到 **C:\Program Files\MyEclipse 6.0**，因为前面已经讲过 Java 程序在这种路径下可能会出现不必要的问题，因此推荐在安装的时候选择一个不带空格的安装路径，如下图所示：

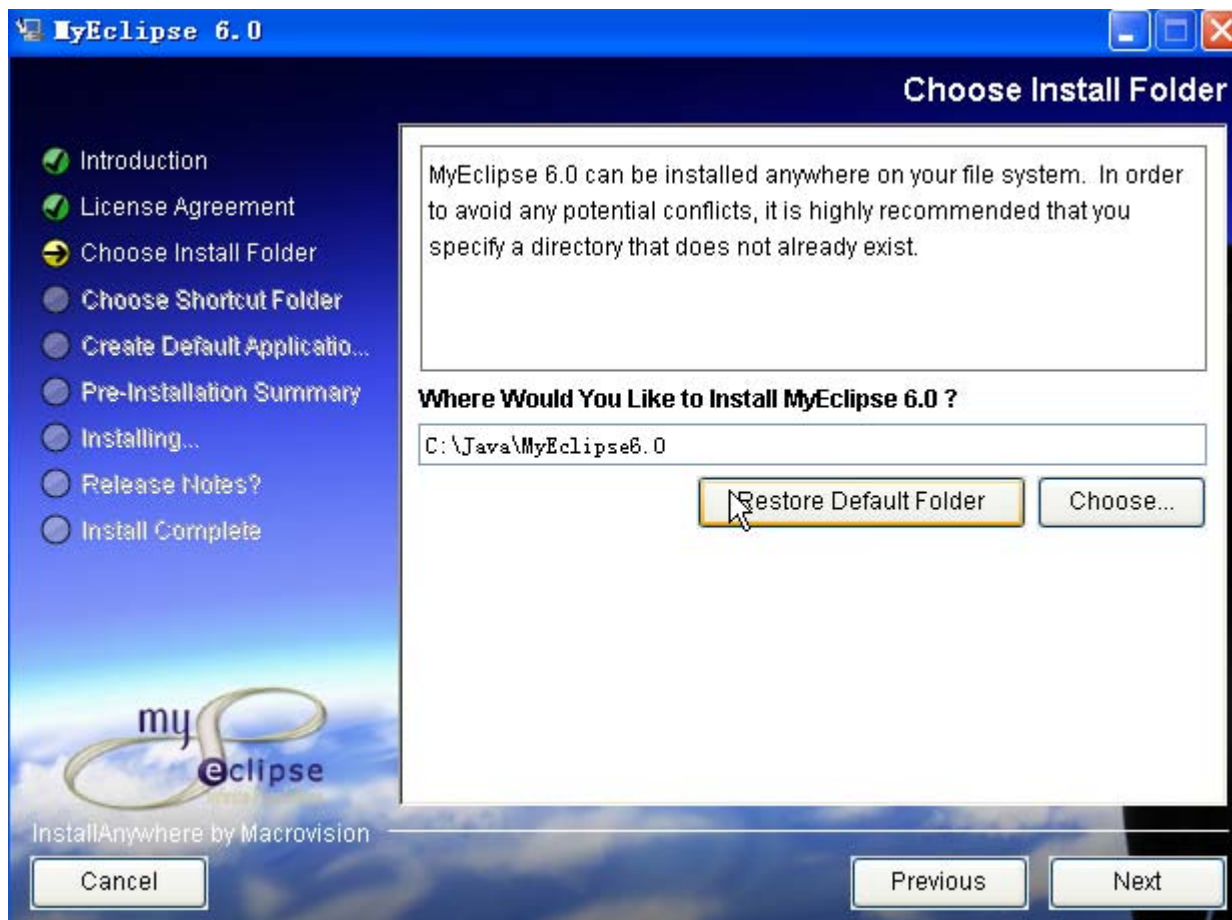


图 1.23 修改 MyEclipse 的安装目录为不带空格的路径

，接着一路点击 **Next** 按钮等待直到最后完成安装即可。

1.7.2.2 插件（PLUG-IN） 版本的安装

插件版本的安装基本上和上述一致，所不同的是在接受协议后将会出现一个选择现有 Eclipse 3.3 安装目录的对话框，如下图所示：

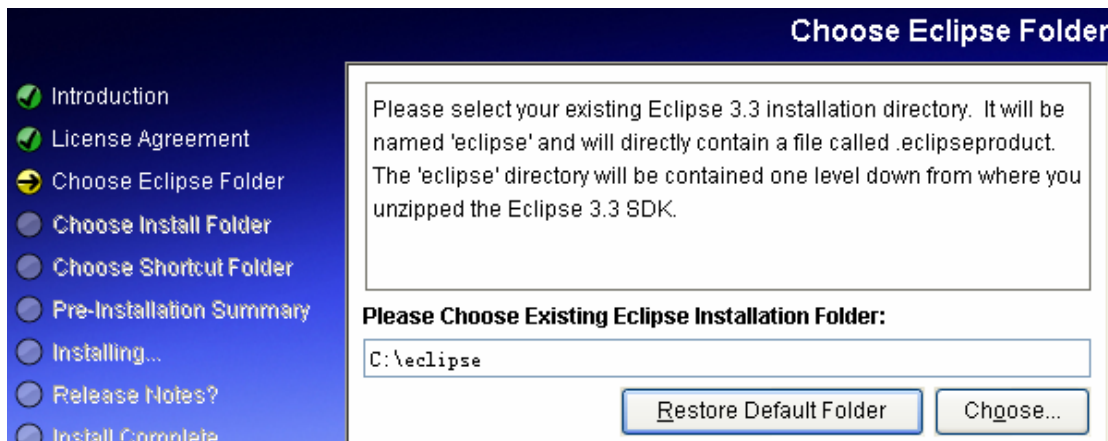


图 1.24 选中现有 Eclipse 3.3 的安装目录

点击 **Choose...**按钮后选中安装好的 Eclipse 3.3 所在目录例如 `c:\eclipse` 然后一路点击 **Next** 按钮即可。


注意：如果你这里选择了错误的 Eclipse 版本例如 3.2 等等，安装能够继续，但是安装完毕后 MyEclipse 将无法正常启动和使用。

1.7.3 运行

点击 Windows 系统的开始菜单后选择**所有程序**，然后选择 **MyEclipse 6.0** 的快捷方式组里面的 **MyEclipse 6.0.1** 即可运行，如下图所示：



图 1.25 通过快捷方式运行 MyEclipse 6

启动过程中会提示你选择 **workspace**，点击 **OK** 按钮就可以继续启动，如前图 2.14 所示。第一次启动后主界面还显示一个欢迎页面（Welcome），点击上面的  图标关闭欢迎页面，之后就可以进行开发了。这时界面如下所示：

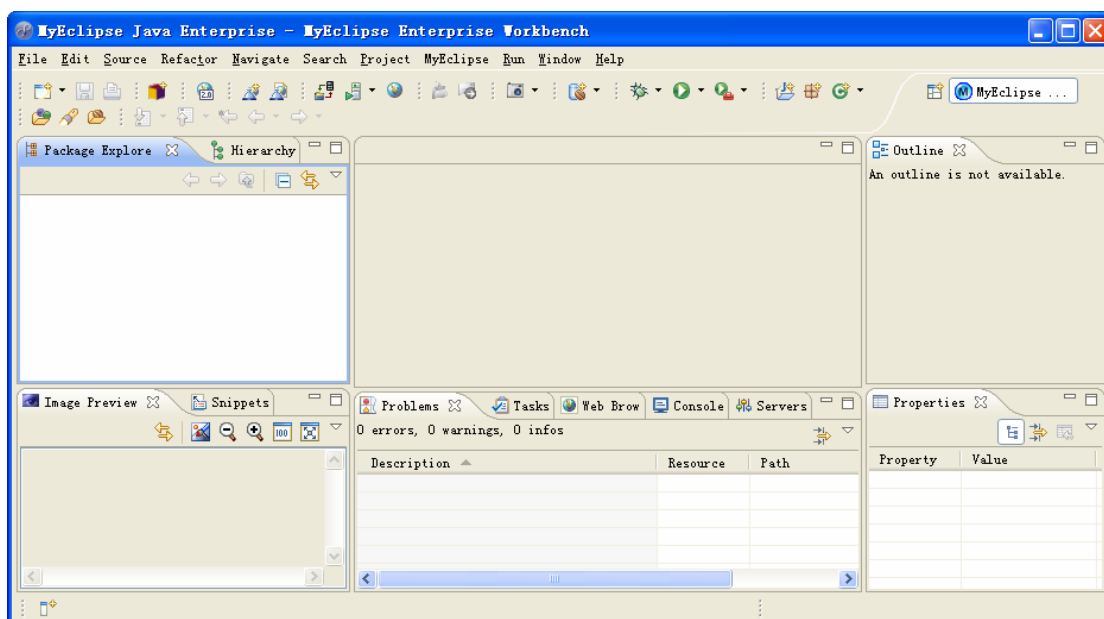


图 1.26 MyEclipse 6 的工作界面

如果你购买或者获得了 MyEclipse 的注册码，可以选择菜单 **MyEclipse - Subscription Information...**来输入，这样你就可以没有时间和功能限制的使用 MyEclipse 的所有功能了。

如果要卸载 MyEclipse 则点击 **Uninstall MyEclipse 6.0.1** 后安装提示一步步点击 **Next** 按钮即可。

1.8 小结

通过本章的介绍，您应该大致了解了本书所使用的 Java，数据库以及服务器软件的安装，设置和运行方式。再次强调如果使用 MyEclipse 6.0 All in ONE 版本，将会最大限度的减少安装难度。

第二章 开发第一个 Java 应用程序

2.1 介绍

本章通过对比手工开发和使用 MyEclipse 开发第一个 HelloWorld 的 Java 程序，让读者对手工写代码和使用开发工具有一定的比较。

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 0: 下载 安装 运行 HelloWorld](#)。

2.2 手工编写，编译并运行 Java 程序

要进行本节所介绍的操作，请务必首先按照第二章的 [JDK 的下载，安装和配置](#) 一节配置好JDK开发环境。

双击桌面上的我的电脑图标，接着双击 C 盘，再选择菜单工具(T)...→文件夹选项(O)...，按照下图设置显示已知文件类型的扩展名：

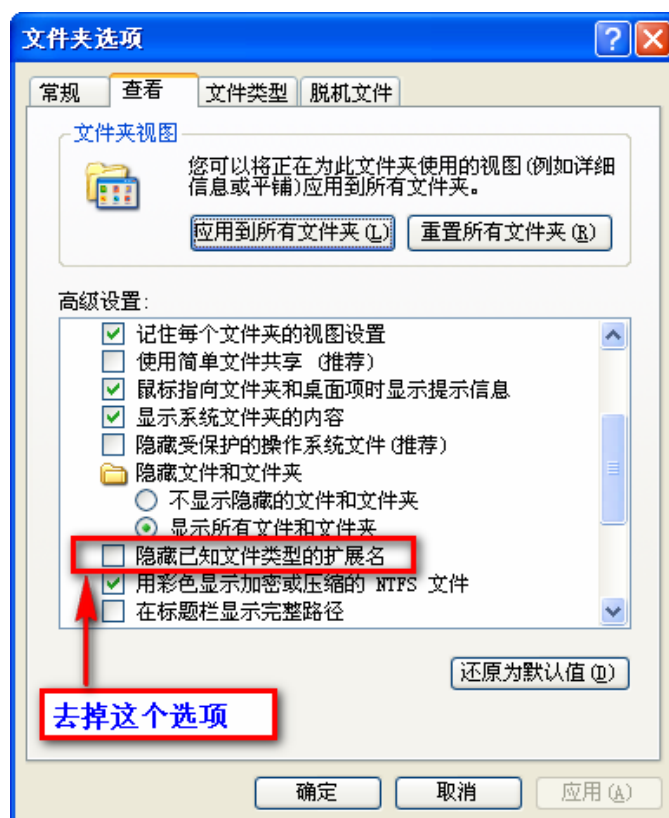


图 2.1 设置 Windows 显示所有文件的扩展名

。这样做的目的是为了以后能方便的区分出来 Hello.txt, Hello.java, Hello.class 这些不同的文件类型，也为了使记事本软件不会自作聪明的把我们要写的程序最后给保存成 HelloWorld.java.txt。

然后选择 Windows 的开始菜单，依次选择**所有程序→附件→记事本**来启动文本编辑器记事本，当然你也可以用自己喜欢的写文本的软件例如 EditPlus, UltraEdit, Notepad2, Notepad++等，但是不要使用 Word 这样的软件来写代码。启动后输入下面的代码：

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("你好 世界!");  
    }  
}
```

接着点击记事本的菜单**文件→保存**，在弹出的文件保存对话框中输入路径：**C:\HelloWorld.java**，然后点击对话框的**保存**按钮即可。

注意：文件名后缀以及大小写均不能写错，使用输入法的时候不要使用中文标点或者全角字母来输入代码，否则会出现无法识别的情况。

接着我们需要编译和运行这个程序。点击**开始菜单**，选择**运行**，输入 *cmd* 后点击确定按钮启动命令解释器。接着输入下列命令：

```
cd C:\  
javac HelloWorld.java  
java HelloWorld
```

如果没有出现任何错误的话，将会打印出：

```
你好 世界!
```

2.3 使用 Eclipse/MyEclipse 来编写，编译并运行 Java 程序

从菜单栏选择 **File > New > Java Project**，接着会打开 **New Java Project** 向导：

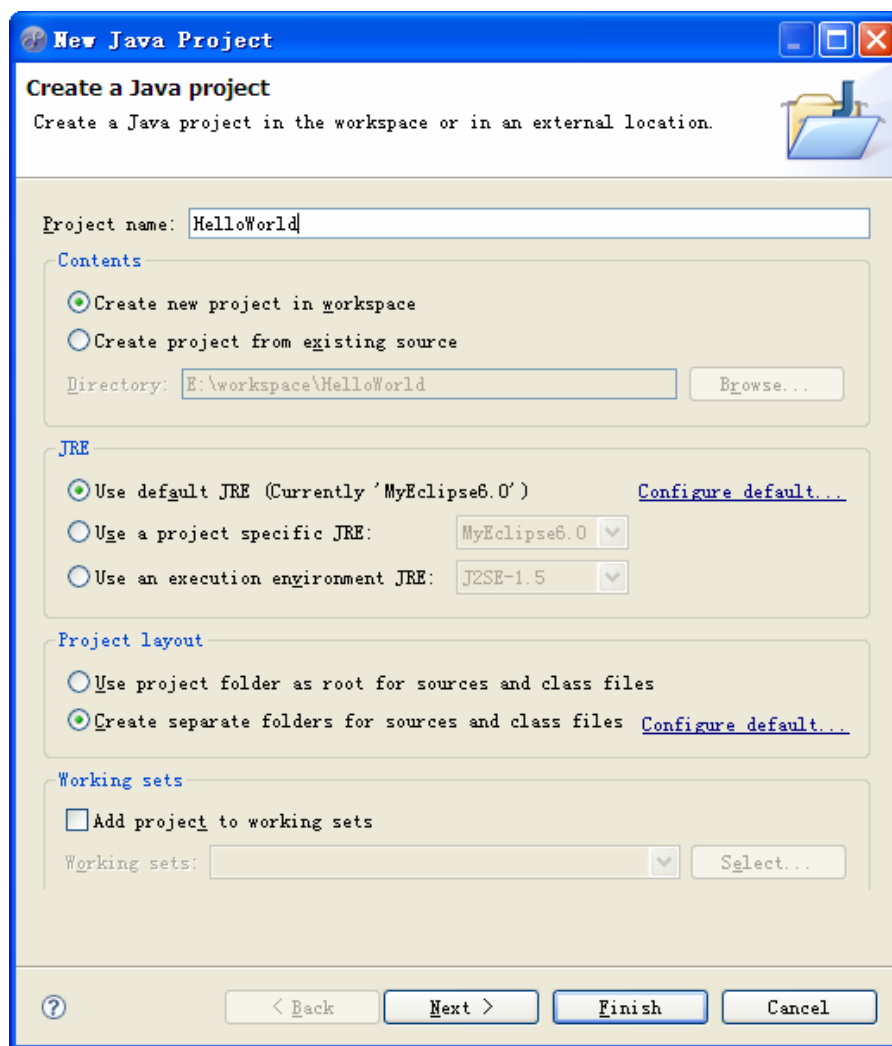


图 2.2 新建 Java 项目

在 **Project name** 中输入 *HelloWorld*，点击 **Finish** 按钮关闭对话框，这样一个 Java 项目就建立完毕了。稍等片刻会弹出一个切换透视图的对话框：

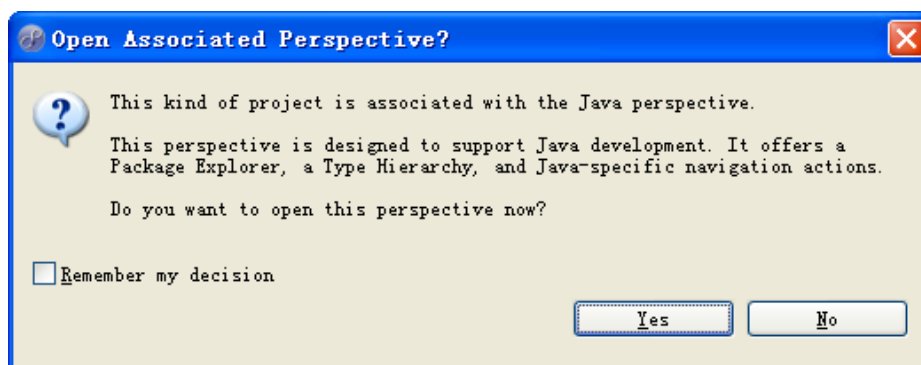


图 2.3 切换到响应的透视图的对话框

为了避免造成更多的麻烦，我们一般选择 **No** 按钮就可以了。接着选择菜单 **File > New > Class**，然后新建类的对话框就出现了：

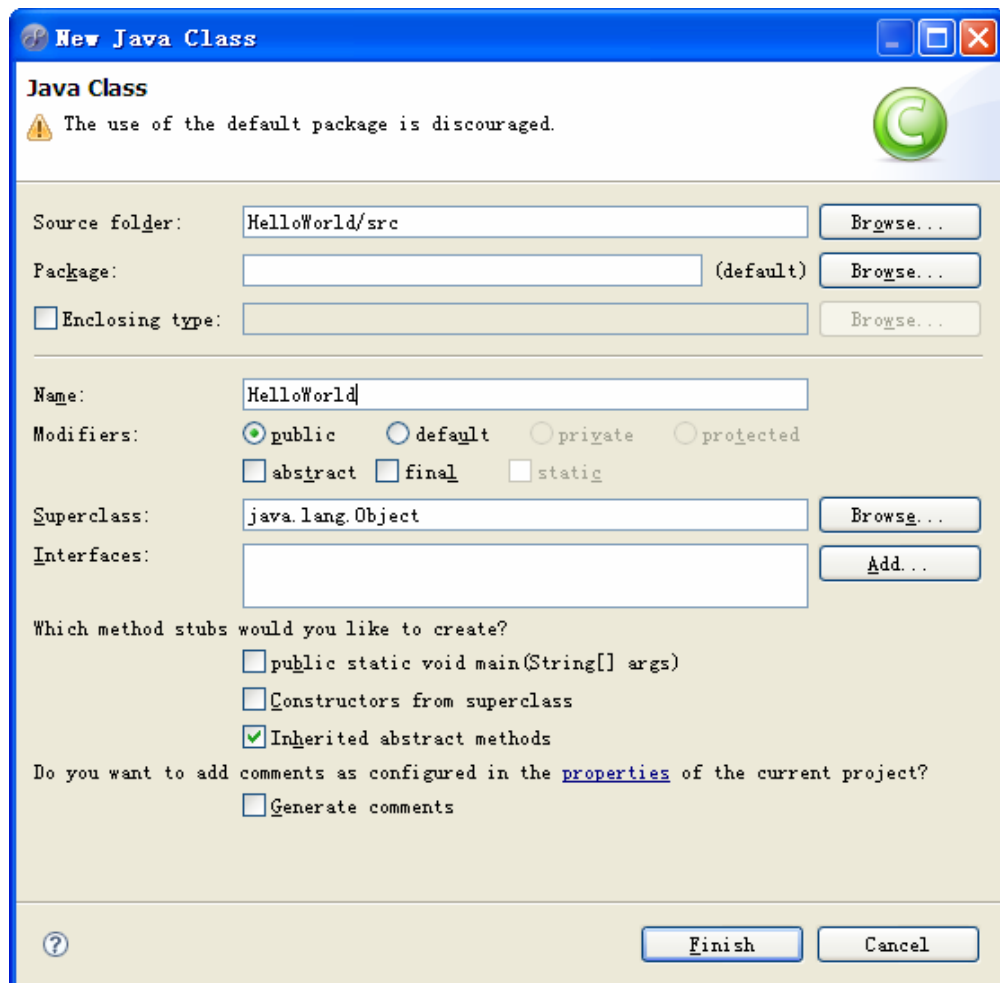


图 2.4 新建类的对话框

在 **Name** 输入框中输入 *HelloWorld*，点击完成。接着将编辑器里面的代码修改成如下所示：

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("你好 世界!");
    }
}
```

。当你的代码编写完毕后，MyEclipse 会自动将代码编译成类文件。

接下来就可以运行写好的类了，选择菜单 **Run** → **Run** 或者按下快捷键 **Ctrl+F11**，就可以看到 Eclipse 会自动调用 Java 解释器，然后在 **Console** 视图中输出“你好，世界”，如下图所示：

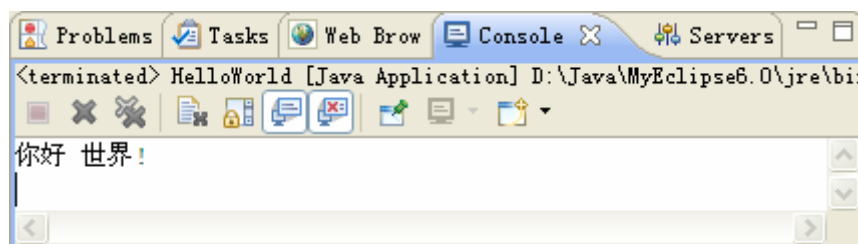


图 2.5 运行并查看程序输入

2.4 小结

通过比较，我们可以看到除了新建 **Java Project** 外，使用 **Eclipse** 来开发 **Java** 类显然要方便快捷的多。而且 **Eclipse** 提供语法高亮显示和错误纠正功能，这在做大型项目时尤其有用。然而，**Eclipse** 的开发模式也和手工来进行开发一样，需要经过创建 **.java** 文件，编译（**Eclipse** 使用的是内置自动编译器）以及运行的过程。

第三章 Eclipse 的基础概念，配置和使用

本章内容让您对 Eclipse 有一个快速的了解，便于以后介绍 MyEclipse 下的开发。如果您已经很熟悉 Eclipse 工具，可以放心的跳过这一章的内容。

3.1 界面布局

和常见的带界面的程序一样，Eclipse 也支持标准的界面和一些自定义的概念。完整的界面布局请参考图 2.19。

3.1.1 菜单

界面最上面的是菜单条，菜单条中包含菜单（如 **File**）和菜单项（如 **File → New**），菜单项下面则可能显示子菜单（如 **Window → Show View → Console**）。如下图所示：

File Edit Source Refactor Navigate Search Project MyEclipse Run Window Help

图 3.1 菜单条

菜单条包含了大部分的功能，然而和常见的 Windows 软件不同，MyEclipse 的命令不能全部通过菜单来完成。

3.1.2 工具栏

位于菜单条下面的是工具栏。如下图所示：



图 3.2 工具栏

工具栏包含了最常用的功能。拖动工具栏上的虚线可以更改按钮显示的位置。下表列出了常见的 MyEclipse 工具栏按钮及其对应的功能：

	新建文件或项目		保存
	打印		新建 UML 模块文件
	启动 AJAX 网络浏览器		新建 Web Service
	启动 Web Service 浏览器		发布 Java EE 项目到服务器
	启动/停止/重启服务器		启动 MyEclipse 网络浏览器
	打开项目或者文件所在目录		启动电子邮件软件发送文件
	截屏		新建 EJB 3 Bean
	调试程序		运行程序
	运行外部工具		新建 Java 项目
	新建包		新建类
	打开类型		搜索

	打开 Spring bean 定义		切换单词匹配
	下一标注		前一标注
	最后一次修改的位置		文件的后一位置
	文件的前一位置		

3.1.3 透视图（Perspective）切换器

位于工具栏最右侧的是 Eclipse 特有的工具栏：透视图切换器。如下图所示：

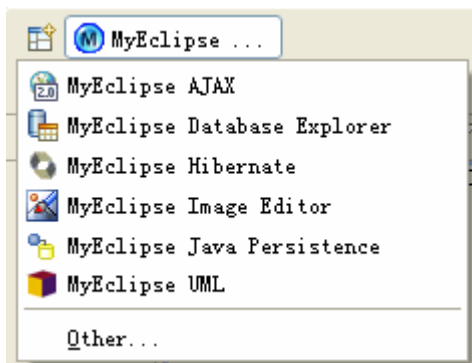


图 3.3 透视图切换器

点击 按钮可以显示多个透视图供切换。什么是透视图？当前的界面布局就是一个透视图，通过给不同的布局起名字，便于用户在多种常用的功能模块下工作，总体来说，一个透视图相当于一个自定义界面。一个透视图保存了当前的菜单栏，工具栏按钮以及视图的大小，位置，显示与否的所有状态，可以在下次切换回来是恢复原来的布局。

要关闭透视图，在 按钮上点击右键，在弹出的菜单中选择 **Close** 即可。要再次显示，如上段所示点击对应的视图名称即可。如果发现在上面的列表找不到，请点击图 4.3 中的 **Other...** 菜单项，在下面所示的打开透视图对话框中选中对应的透视图，然后点击 **OK** 即可。

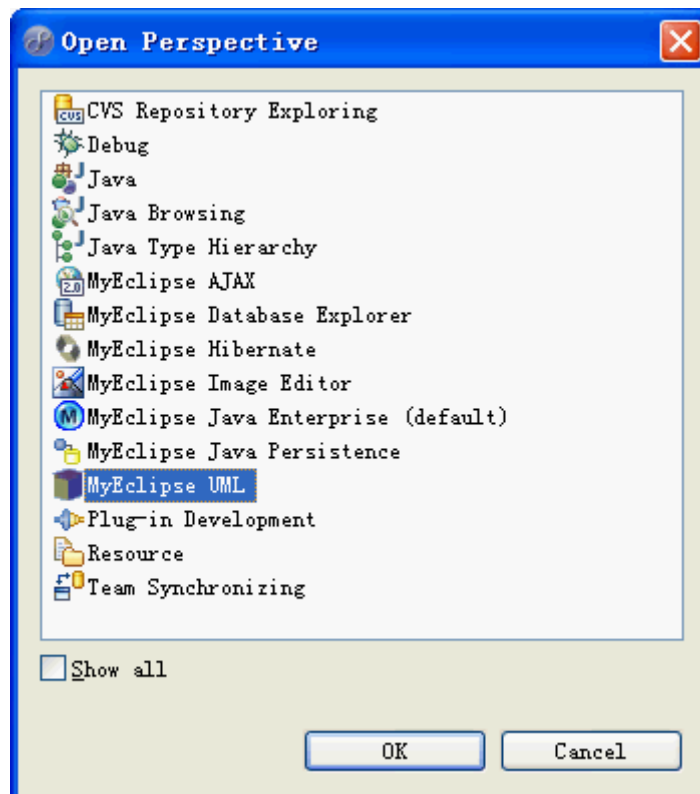


图 3.4 切换所有透视图

在这个图中有个复选框 **Show all**，在这里可以看到和 MyEclipse 冲突的一些透视图，例如 **WTP Java EE**，因此一般来说不要使用。MyEclipse 的默认透视图是 **MyEclipse Java Enterprise**。

3.1.4 视图 (View)

视图是显示在主界面中的一个小窗口，可以单独最大化，最小化显示，调整显示大小和位置，以及关闭。除了工具栏，菜单栏和状态栏之外，Eclipse 的界面就是由这样一个的小窗口组合起来，像拼图一样构成了 Eclipse 界面的主要部分。下面的是大纲视图：

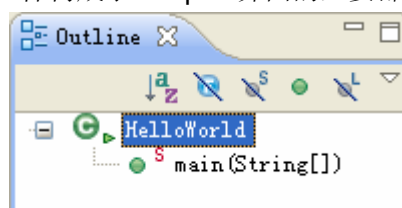


图 3.5 视图







每个视图包括关闭按钮，最大化和最小化按钮，视图工具栏以及视图主体和边框组成。视图最顶部显示的是标题栏，拖动这个标题栏可以在主界面中移动视图的位置，单击标题栏则会切换显示对应视图的内容；双击标题栏或者点击最大化按钮  可以让当前视图占据整个窗口。点击  将会关闭当前的视图。点击  按钮最小化当前视图，这时候当前视图会缩小为一个图标并显示在界面的上侧栏，或者右侧栏和状态栏上，如下图所示：



图 3.6 最小化显示的视图

拖动工具栏上的虚线可以更改最小化视图显示的位置。点击按钮可以恢复最小化之前的视图位置和大小，点击最小化后的图标可以暂时显示（术语叫快速切换 **Fast View**）视图的完整内容。鼠标放在边框上并拖动可以调节视图的显示大小。点击视图上的工具栏按钮可以执行对应的功能，而点击按钮则可以显示和当前视图相关联的功能菜单。

当视图不小心关闭后，可以通过下列菜单再次打开：**Window** → **Show View**，如下图所示，可以选择要显示的视图：

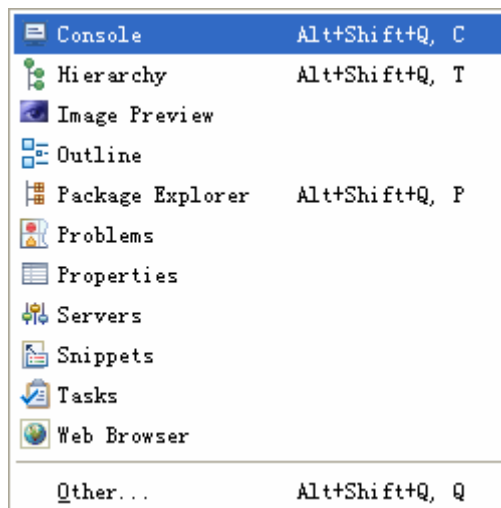


图 3.7 视图列表子菜单

如果在上面没有发现要显示的视图，可以点击 **Other...** 菜单，然后在所有的视图列表中选择所显示的视图，如下图所示：

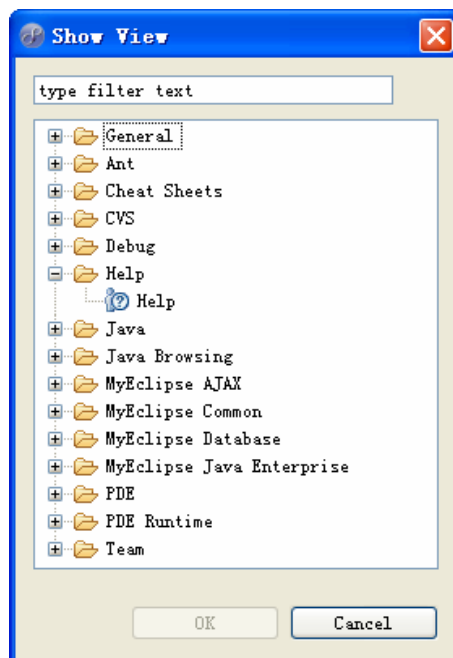


图 3.8 显示视图对话框

可以在 **type filter text** 这个框中输入视图的部分名字来模糊查找，这样能够快速定位到要显示的视图。

常见的视图及其功能：

Package Explorer	Java 包结构	Hierarchy	类层次（继承关系）
Outline	大纲，显示成员等	Problems	错误，警告等信息

Tasks	任务如 TODO 标记	Web Browser	网络浏览器
Console	控制台，程序输出	Servers	服务器列表和管理
Properties	属性	Image Preview	图片预览
Snippets	代码片段		

3.1.5 上下文菜单（Context Menu）

在界面的任何地方点击鼠标右键所弹出的菜单叫上下文菜单，它能快捷的显示和鼠标所在位置的组件或者元素动态关联的功能。

3.1.6 状态栏（Status Bar）

在界面的最底部显示的是状态栏，相对来说，Eclipse 的状态栏功能大大弱化，它的主要功能都集中在视图中。如下图所示：

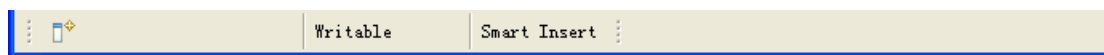


图 3.8 状态栏

3.1.7 编辑器（Editor）

在界面的最中央会显示代码或者文件编辑器。这个编辑器和视图非常相似，也能最大化和最小化，所不同的是会显示多个标签页，也没有工具栏按钮，而且有一个很特殊的组件叫隔条，如图中的代码最左侧的蓝色竖条所示，隔条上会显示行号，警告，错误，断点等提示信息。编辑器里面的内容没有被修改时，会在标签页上显示 * 号。如下图所示：

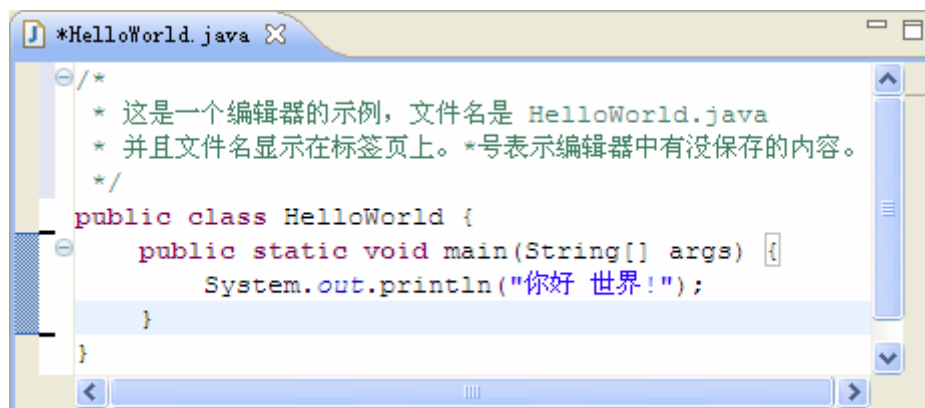


图 3.9 编辑器

当同时打开多个文件时，会显示标签页下拉框例如>>4，数字表示了有多少个已打开的文件未在编辑器区域中显示。单击>>可以显示文件列表选择框，可以在输入框中输入部分字母来模糊匹配已打开的文件。如下图所示：

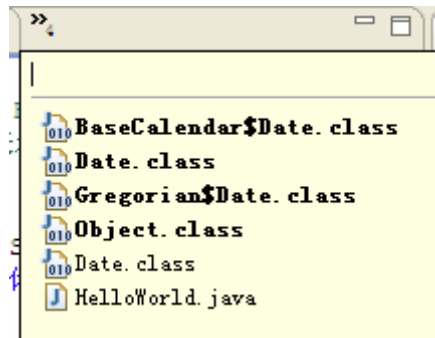


图 3.10 文件列表选择框

3.2 常见概念和操作

3.2.1 项目(Project)

Eclipse 中所有的可以编译运行的资源必须放在项目中, 单独打开文件很多功能不可用。项目表示了一系列相关的文件和设置 (例如类路径, 编译器级别, 发布路径等等的设置)。一般来说目录下的 **.project** 和 **.classpath** 这两个文件描述了当前项目的信息。打开项目可以先选中单个或者多个项目, 然后选择菜单 **Project → Open Project**, 或者点右键选择菜单 **Open Project**。关闭项目可以先选中要关闭的单个或者多个项目, 然后选择菜单 **Project → Close Project**, 或者点右键选择菜单 **Close Project**。

3.2.2 工作区(Workspace)

一个 Eclipse 可以有多个工作区, 每个工作区包含了多个项目, 以及所有其余的设置信息例如界面布局, 文字大小, 服务器定义等等。但是一个工作区只能被单个 Eclipse 进程使用。另外同一个项目也会加入到不同的工作区中。**注意:** 删除工作区目录的时候很可能误删位于工作区中的项目文件。工作区目录会有一个名为 **.metadata** 的目录来保存所有设置信息。在 Eclipse 启动的时候会让你选择要使用的工作区。如果输入的工作区目录不存在, Eclipse 会自动创建它。

3.2.3 导入、导出 Java 项目

3.2.3.1 导入项目

当下载了包含 Eclipse 项目的源代码文件后, 我们可以把它导入到当前的 Eclipse 工作区然后编辑和查看。点击菜单 **File→Import**, 然后在弹出的 **Import** 对话框中展开 **General** 目录, 选择 **Existing Projects into Workspace**, 接着点击 **Next** 按钮。当选中单选钮 **Select root directory:** 时可以点击 **Browse...** 按钮选中包含项目的文件夹, 如果包含项目的话就可以在中间的 **Projects** 列表框中显示; 而当选中单选钮 **Select archive file:** 时可以点击 **Browse...** 按钮选中包含项目的 ZIP 压缩包, 如果包含项目的话就可以在中间的 **Projects** 列表框中显示。最后点击 **Finish** 按钮就可以导入项目并打开了。如下图所示:

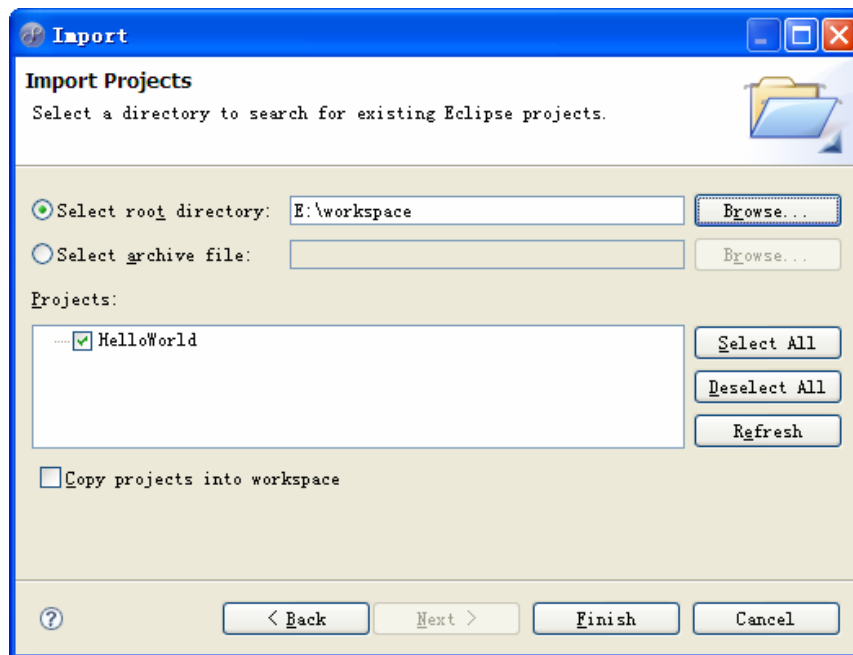


图 3.11 导入项目

3.2.3.2 导出项目

点击菜单 **File→Export**，然后在弹出的 **Export** 对话框中展开 **General** 目录，选择 **Archive File**，接着点击 **Next** 按钮。然后在 **To archive file:**输出框中选中要保存的文件名，一般写成 *项目名.zip*，然后点击 **Finish** 按钮即可导出当前项目。还有一种方式是手工打包，用 WinRAR 或者 WinZIP 等工具都可以，先点击工具栏上的 打开项目所在目录，接着就可以用你喜欢的工具来打包代码目录了。

3.2.4 快速修正代码错误

在 Eclipse 的编辑器中编写代码以及编译后会显示检查出来的错误或者警告并在出问题的代码行首的隔条上显示红色的灯泡。左键点击灯泡或者按下快捷键 **Ctrl+1** (或者菜单 **Edit > Quick Fix**)可以显示修正意见，并在修正前显示预览。如下图所示：

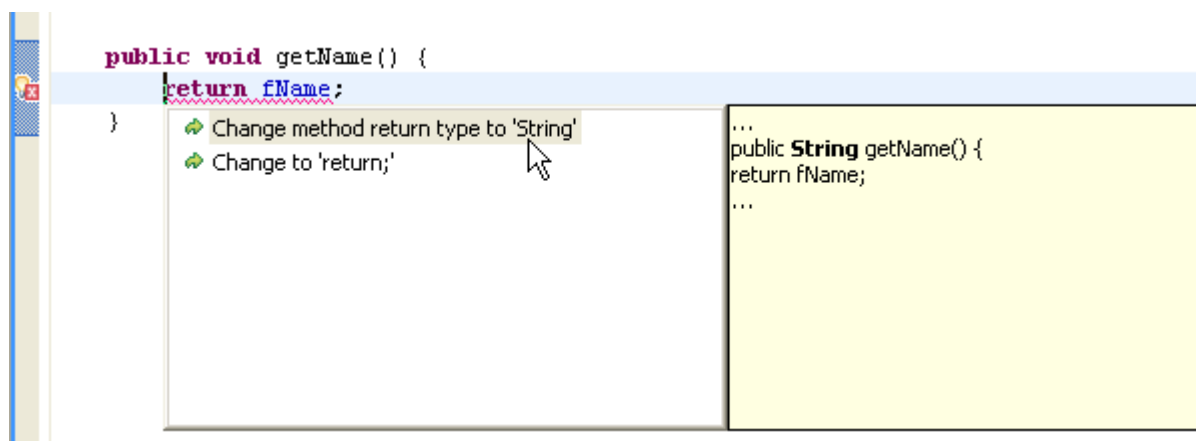


图 3.12 快速修正代码错误

3.2.5 优化导入列表

代码中经常会导入无用的包和类，通过菜单 **Source** → **Organize Imports** 或者在编辑器的上下文菜单中选择菜单项 **Source** → **Organize Imports**，或者按下快捷键 **Ctrl+Shift+O** 也可以来重新组织并去掉无用的类和包。

3.2.6 添加，修改，删除 JRE

通过菜单 **Window** → **Preferences**，然后选择 **Java** > **Installed JREs**，可以打开供在 Eclipse 编写程序所使用的 JRE 列表。复选框选中的 JRE 是默认的 JRE，它被项目里面所有的项目来作为编译和启动的 JRE（除非在项目的 Build Path 中指定了其它的 JRE）。可以通过 **Add...** 按钮来添加新的 JRE 定义（在弹出的对话框中选择 **Browse...** 按钮然后选中 JDK 的安装目录，之后点击 **OK** 即可），**Edit...** 按钮来修改 JRE 定义，**Remove** 按钮来删除 JRE 定义，选中不同的 JRE 前面的复选框来把它作为默认 JRE。虽然 MyEclipse 能够自动找到并显示一个 JRE，但是强烈建议大家添加一个 JDK 来进行开发，便于查看 JDK 类源码和编码时能够显示提示信息。如下图所示：

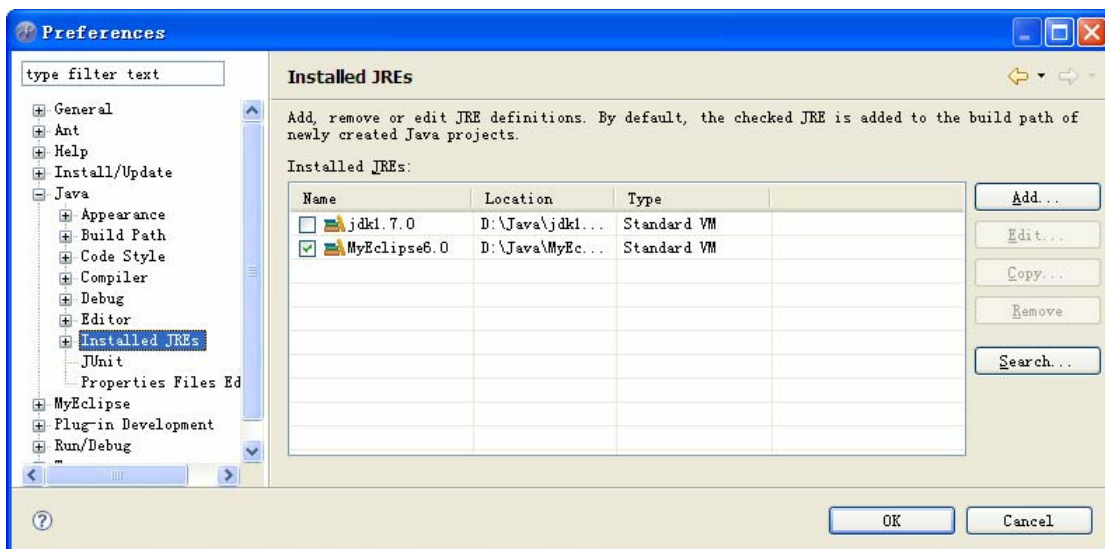



图 3.13 配置安装的 JRE

3.2.7 查看类定义，层次和源码

查看类定义或者其源码，可以在编辑器的上下文菜单中选择 **Open Declaration**，或者选择菜单 **Navigate** → **Open Declaration**，或者按下 **F3** 键。如果这个类关联了源码（例如 JDK 里面的类），就可以看到源代码，否则只能看到类的方法和成员信息。

查看类的继承层次，可以在编辑器的上下文菜单中选择 **Open Type Hierarchy**，或者选择菜单 **Navigate** → **Open Type Hierarchy**，或者按下 **F4** 键，或者将类或者包拖放到 **Hierarchy** 视图，就可以在 **Hierarchy** 视图看到类的继承层次，之后就可以点击对应的类看到定义了。

3.2.8 查找类文件（Open Type）

要快速找到某个类型的定义，选择菜单 **Navigate → Open Type**，或者按下 **Ctrl+Shift+T** 键，或者按下工具栏按钮。这时候可以出现 Open Type 对话框，在 **Enter type name prefix or pattern** 输入框中键入类的头几个字母，或者也可以使用?和*这样的通配符来模糊查找，对话框下面的列表中将会显示匹配类文件，选中列表中显示的单个或者多个类定义来打开它。如果这个类关联了源码（例如 JDK 里面的类），就可以看到源代码，否则只能看到类的方法和成员信息。如下图所示：

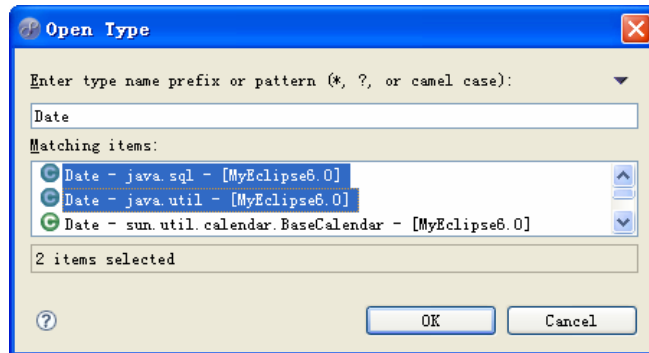


图 3.14 Open Type 对话框

3.2.9 源码目录，输出路径，Library 和编译器版本设置

点击菜单 **Project → Properties** 或者在 **Package Explorer** 项目节点上右键点击选择上下文菜单中的 **Properties**，或者用快捷键 **Alt+Enter**，可以打开项目属性对话框。选择左侧的 **Java Build Path**，可以在右侧显示项目的类路径有关的设置标签页。**Source** 页显示了源代码目录（可以使用一个或者多个，里面的源文件将会被编译）以及 **Java** 源代码编译后产生的类文件所存放的目录。这些参数都可以修改，源代码目录可以添加或者删除。

Package Explorer 视图默认是不显示类文件的输出目录的。如下图所示：

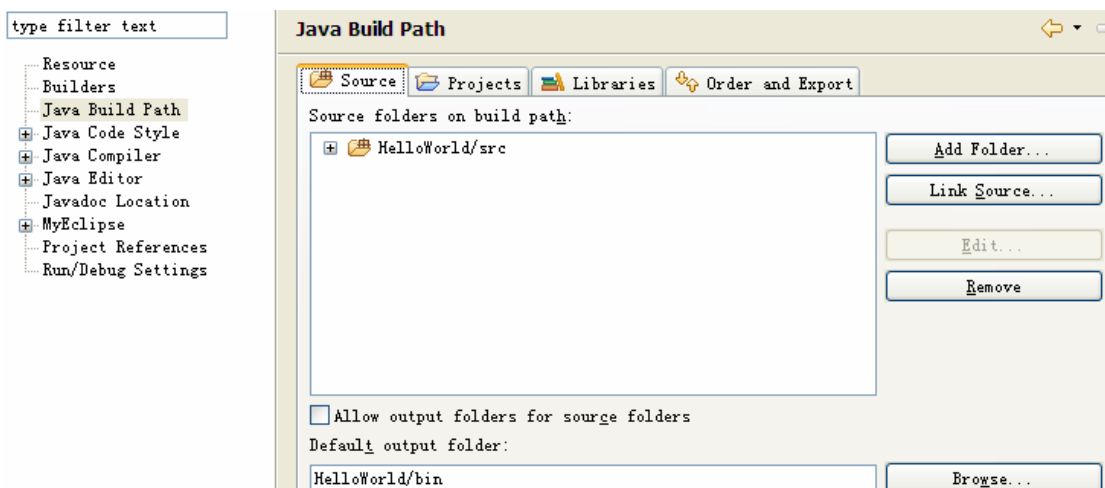


图 3.15 源码目录和输出路径

点击 **Libraries** 页面则可以设置当前项目的类路径，这些类库在编译源文件时使用。如下图所示：

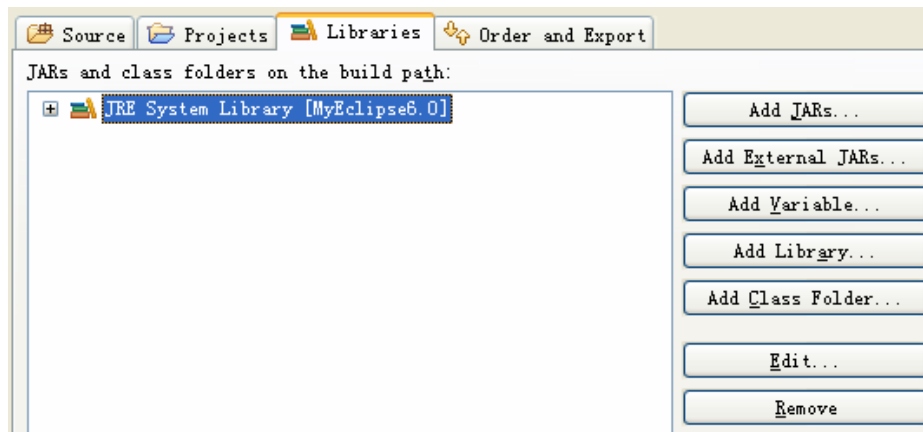


图 3.16 类库

Add JARs 按钮可以将当前项目中的 jar 文件加入到类路径，**Add External JARs** 则将添加项目外的 jar 文件到类路径，**Add Variable** 添加变量，**Add Library** 可以添加类库（一个或者多个 jar 文件的集合，由开发工具定义和管理），**Add Class Folder** 则添加目录中的类文件，**Edit** 可以修改所选类库的设置，**Remove** 则从类路径中删除选中的类库。

在开发中不可避免的需要设置源代码的编译级别，例如使用 JDK1.6 来开发将来运行于 JDK1.4 上的项目，那么这时候需要设置编译器的等级，否则将来的类文件会因为版本过高而不能被目标 JDK 识别。点击项目属性对话框中的 **Java Compiler** 可以设置代码的编译器级别。如下图所示：

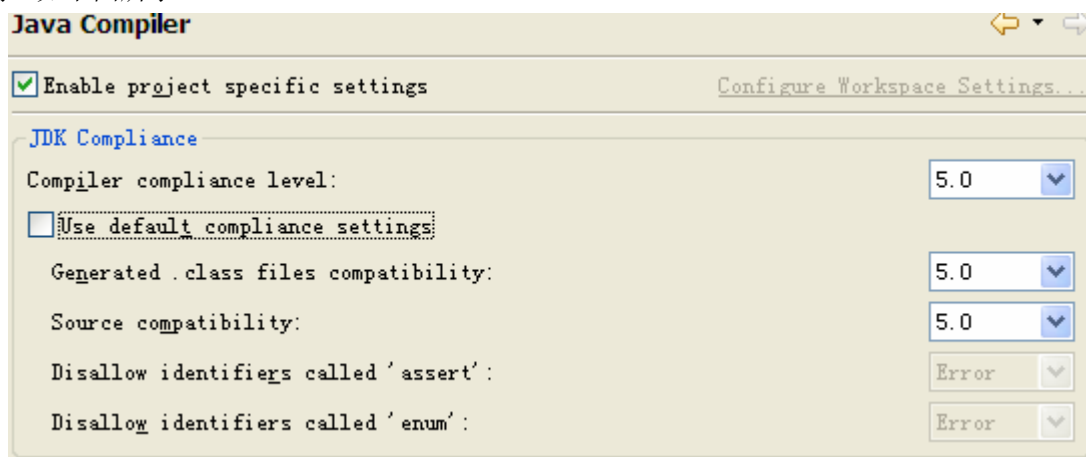


图 3.17 编译器级别

如果只是修改当前项目的编译器级别，可以选中复选框 **Enable project specific settings**，然后在 **Compiler compliance level** 右侧的下拉框中选择目标的编译级别，例如 5.0, 1.4 等等。还可以去掉 **Use default compliance settings** 复选框的选中状态，来进一步设置。这些设置将会影响到源代码中的语法错误检查，例如要在 1.4 级别的项目中用 5.0 的语法写代码，肯定是会报错的。

如果要修改所有项目的默认编译级别，点击 **Configure Workspace Settings...** 来打开全局设置对话框，这两处的设置几乎是一样的，就不再赘述了。

3.2.10 生成 getter 和 setter 方法

在写 **JavaBean** 的时候常常要写一些模式化的 **getXXX()** 和 **setXXX()** 这样的方法，我们可以用 **Eclipse** 来自动生成这些模版化的方法。先写好 `private String name;` 这样的变量定

义,然后选择菜单 **Source** → **Generate Getters and Setters...** 或者在编辑器中点击右键选择菜单 **Source** → **Generate Getters and Setters...**就可以打开 **Generate Getters and Setters** 对话框,在对话框中选择要生成的方法,然后点击 **OK** 按钮即可。

3.2.11 格式化源代码

有时候代码手写的很乱,这时候可以先选中要格式化的代码(不选择是格式化当前文件的所有代码),通过选择菜单 **Source** → **Format** 或者在编辑器中点击右键选择菜单 **Source** → **Format** 或者通过快捷键 **Ctrl+Shift+F** 来快速的将代码格式化成便于阅读的格式。这个操作在 MyEclipse 中也可以格式化 XML, JSP, HTML 等源文件。

3.2.12 注释和取消注释

使用快捷键 **Ctrl + /** 可以将选中的代码快速的添加或者去掉两个斜线(//)风格的注释。

3.2.13 手工和自动编译

如果是特别大的项目,例如几千个源代码,使用 Eclipse 来自动编译将会是一场噩梦。每键入一行代码都会自动启动编译器检查进程,严重时候屏幕甚至会卡着不动(这也是 Eclipse 的一个优点之中的缺点)。这时候可以切换 Eclipse 的自动编译为手工编译。去掉菜单 **Project** → **Build Automatically** 的选中状态后,项目就变成了手工编译状态;再次点击菜单可以重新切换会自动编译状态。这时候再键入代码就不会自动检查编译错误了,也不会生成编译后的类文件,这样有助于快速的写代码。此时要进行编译可以选择菜单 **Project** → **Build Project** 来编译当前项目或者 **Project** → **Build All** 来编译所有项目。

3.2.14 直接粘贴 Java 源码为类文件

Eclipse 3.3 支持一个功能就是如果剪贴板上放的是 Java 源程序,例如如下所示的代码复制到剪贴板上:

```
public class YetAnother {
}
```


那么点击菜单 **Edit** → **Paste** 或者在 **Package Explorer** 视图的项目节点的上下文菜单中选择 **Paste**,或者按下快捷键 **Ctrl + V**,那么 Eclipse 会根据这段代码自动生成一个新的.java 文件并把它加入到当前项目的源代码目录中。

3.2.15 复制项目中的文件

首先选中 **Package Explorer** 视图的文件节点(Java 类或者普通文件都可以),那么点击菜单 **Edit** → **Copy** 或者在 **Package Explorer** 视图的项目节点的上下文菜单中选择 **Copy**,或者按下快捷键 **Ctrl + C**,之后再选择粘贴的话,会在要粘贴的位置创建原始文件的副本,如果是类的话会自动修改其包定义或者提示你输入类的新名称。如果你在 Windows

的文件浏览器中选中的一个文件或者文件夹复制，之后再在 **Eclipse** 中粘贴，那么这个文件或者文件夹会立即复制并加入到当前项目中，这样可以快速的导入一些单独的源代码。

3.2.16 断点和调试器

在源代码的隔条上双击鼠标可以切换是否在当前行设置断点（break point），断点以  的形式显示，如下图所示：

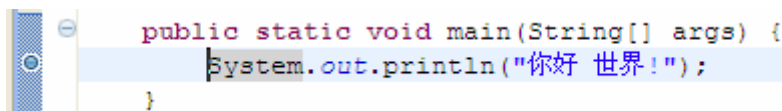



图 3.18 断点

之后我们可以通过菜单 **Run → Debug**，或者 **Run → Debug As → 1 Java Application**，或者通过工具栏按钮 ，或者快捷键 **F11**，或者在编辑器的上下文菜单中选择 **Debug As → 1 Java Application** 来启动调试器。当调试器遇到断点时就会挂起当前线程并切换到调试透视图。调试透视图将会显示 **Debug** 视图，**Variables** 视图，**Breakpoints** 视图和 **Expressions** 视图。例如我们的程序调试时如下所示：

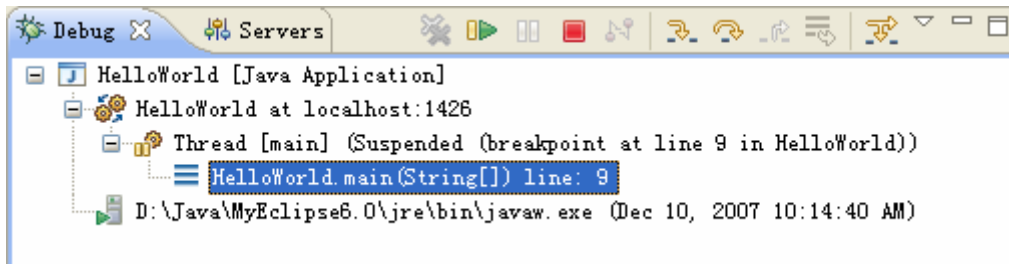


图 3.19 Debug 视图

Debug 视图中显示了当前所有运行中的线程以及所执行的代码所在的位置。这时候编辑器中将会以绿色高亮行背景指示执行代码的位置，如下图所示：

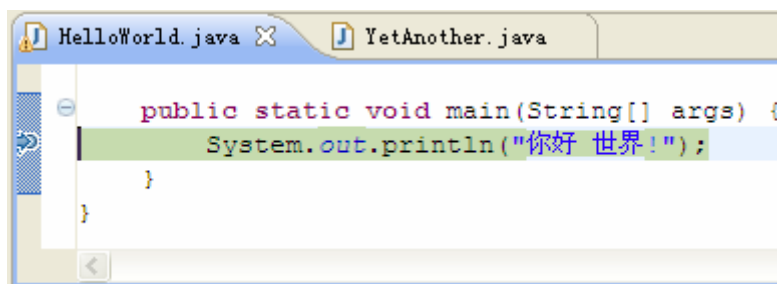


图 3.20 调试时候的代码指示器

而 **Variables** 视图则显示当前方法或者类中的局部，全局等变量的值。

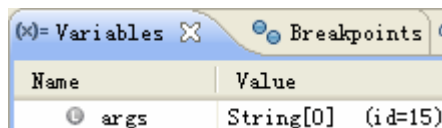


图 3.21 Variables 视图

这时候线程已经挂起，点击 **Debug** 视图的  **Resume** 按钮来继续往下执行，要重新挂起可以选择某个线程，然后点击  **Suspend** 按钮。要一行行的调试代码，可以点击  **Step Over** 按钮来往下执行，或者按下 **F6** 键。要终止调试，可以点击  按钮。

注意：Eclipse 中的调试器功能很完善，但是使用也非常复杂，更详细的资料可以参考 IBM 的开发人员站点或者 Eclipse 的帮助文档。限于篇幅这里就不再多介绍了。

3.2.17 快速加入、删除 jar 包到 Build Path

首先将jar文件复制到项目中(参考[复制项目中的文件](#)一节)，然后在**Package Explorer**视图的jar文件上单击右键，选择菜单**Build Path** → **Add to Build Path** 就可以将这个jar文件加入Build Path；要从项目的Build Path中去掉这个jar文件，可以选择上下文菜单中的**Build Path** → **Remove from Build Path**。

如果是 MyEclipse 的 Web 项目的话，当你将 jar 文件添加到 **WebRoot/WEB-INF/lib** 下后，MyEclipse 会自动把它加入到当前项目。如果发现新加入的文件没有显示在 Eclipse 中，可以在 **Package Explorer** 视图中选择上下文菜单中的 **Refresh** 或者按下快捷键 **F5** 就可以看到了。

3.2.18 查看当前类被哪些类引用

在项目中如果能看到类或者变量，方法被哪些其它的类所引用，将会大大的加快调试或者理解程序结构的进度。可以在编辑器的上下文菜单中选择 **References** → **Project** 来显示当前项目哪些类引用到了它，或者 **References** → **Workspace** 来看整个工作区里面哪些类引用到了它。查找结果显示在 **Search** 视图中。

3.2.19 设置编辑器字体，颜色和显示行号

默认情况下 Eclipse 的代码编辑器是不显示行号的，要显示它可以通过菜单 **Window** → **Preferences...** 来打开 **Preferences** 设置对话框，几乎所有 Eclipse 的设置选项都可以在这里找到。要显示行号，可以展开节点 **General** → **Editors** → **Text Editors**，在右侧的设置中选复选框 **Show line numbers** 即可。如图所示：

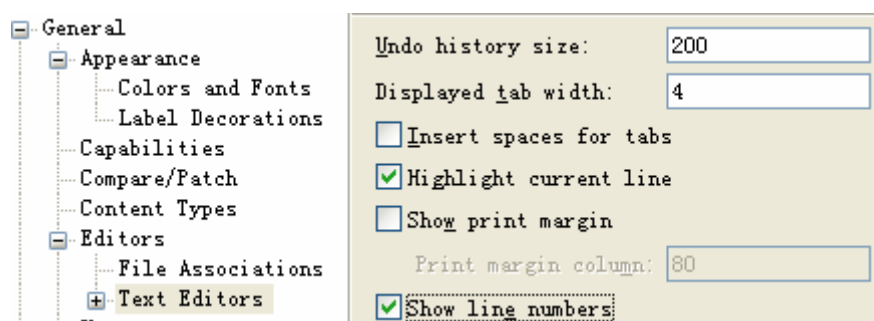


图 3.22 显示行号

显示了行号的编辑器如下所示：

```

5 public class HelloWorld {
6     private String name;
7
8     public static void main(String[] args) {
9         System.out.println("你好 世界!");
10    }
11 }

```

图 3.23 显示了行号的编辑器

要修改编辑器的字体，可以选择 **Preferences** 对话框的 **General** → **Appearance** → **Colors and Fonts**，之后就可以在右侧修改字体了。

注意：编辑器的字体是设置 **Basic**→**Text Font**，之后点击 **Change...**按钮即可。如下图所示：

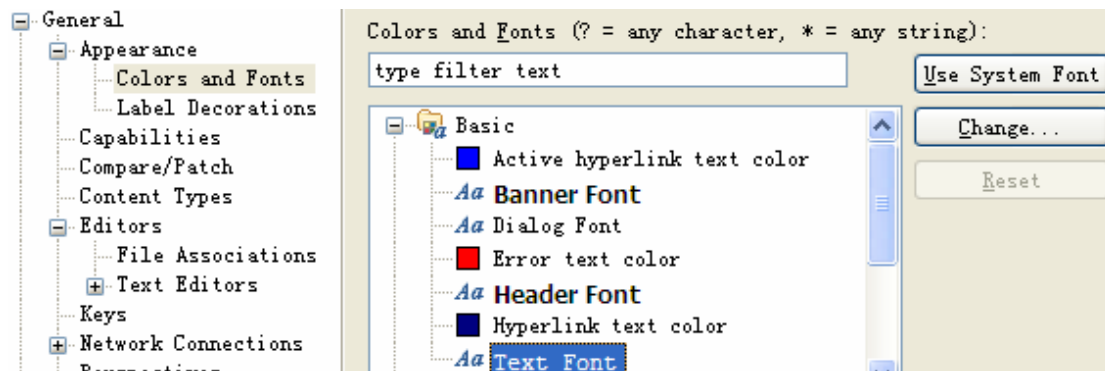


图 3.24 修改编辑器字体

3.2.20 Link 文件

Eclipse 支持一种特殊的概念叫 Link 文件，其实和 Windows 的快捷方式这个概念是非常像的。选择菜单 **File** → **New** → **File** 或者 **File** → **New** → **Folder**，可以打开新建文件或者目录的对话框，如下图所示：

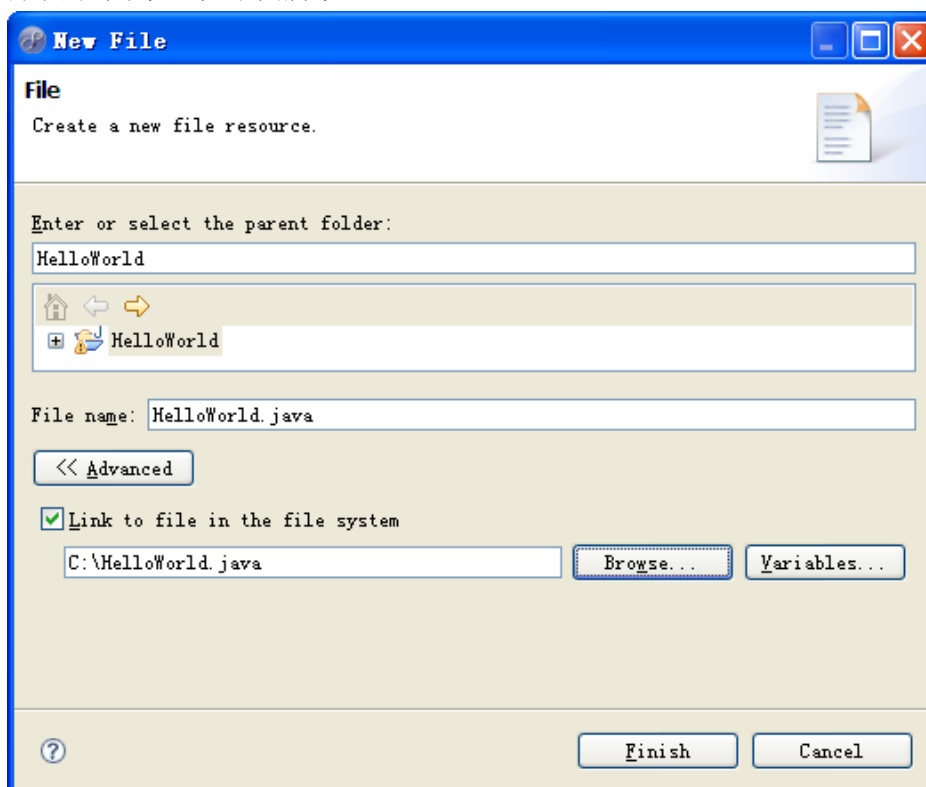




图 3.25 Link 方式创建文件

这时候如果点击 **Advanced** 按钮，然后选中复选框 **Link to file in the file system**，之后就可以点击 **Browse...**按钮来选中项目之外的其它文件。此时创建的文件就叫一个 Link 文件，相当于快捷方式，真正的内容是存储在 `c:\HelloWorld.java` 中，但是对项目中的这个

文件的修改会自动的同步到 `c:\HelloWorld.java` 中去,就好像这个文件是在当前项目中一样。创建完毕的文件图标上会显示一个箭头来说明这个文件是个 **Link** 文件,看起来像这样; 目录的图标显示起来像这样:。Link 目录中的 Java 源代码也可以加入到源代码目录中去进行编译。

注意: 因为 Link 方式的文件依赖于文件系统的绝对路径,因此不建议使用这种方式来把你的项目打包发给别人来使用。

3.2.21 安装插件

我们假定的是 `C:\JavaMyEclipse6.0` 为你的 MyEclipse 的安装目录,一般的 Eclipse 插件只需要复制到 `C:\JavaMyEclipse6.0\eclipse\plugins` 下面就可以安装完毕,这样的插件一般是单独的 jar 文件。如果发现下载的插件是个 ZIP 格式而且发现解压缩后带有 eclipse 子目录,那需要把它直接复制到 `C:\JavaMyEclipse6.0` 覆盖 eclipse 目录即可完成安装(**注意:** 不要删除老的 eclipse 目录)。

3.2.22 获取帮助和阅读帮助文档

在任何位置按下 **F1** 键, Eclipse 会显示相关的帮助文档;完整的帮助文档可以通过菜单 **Help → Help Contents** 来阅读。绝大多数的 MyEclipse 和 Eclipse 的操作说明,相关的一些教程,都可以在帮助文档中找到,虽然内容是英文的,但是内容是非常全面,图文并茂的。**MyEclipse Learning Center** 里的内容是所有 MyEclipse 自带的操作和教程文档。

3.2.23 CVS 团队源代码管理（在线阅读）

请参考 IBM 开发人员社区的一篇文章来学习:

使用 Eclipse 平台共享代码——Eclipse 如何使用源代码版本控制

<http://www.ibm.com/developerworks/cn/linux/opensource/os-ecshare/index.html>。

3.3 小结

在本章中介绍了常见的一些 Eclipse 的使用和概念,因为 MyEclipse 基于 Eclipse,所以这些内容也适用于 MyEclipse,在学习过程中反复练习可以明显提高开发的效率。

第四章 用 MyEclipse Database Explorer 管理数据库

本章的内容将会介绍 **MyEclipse Database Explorer**，不了解本章的内容您将可能在后续的自动生成 **Hibernate**，**JPA**，**EJB 3** 类文件开发时遇到困难。本章将会介绍如何使用 **MyEclipse Database Explorer** 来管理 **MySQL** 数据库和 **MyEclipse Derby** 数据库。


Derby 数据库现在称为 **Java DB**，已经成为了 **JDK 1.6** 的一部分，它也是一款开源免费的数据库，所不同的是它是基于纯 **Java** 进行开发的。

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 2 用 MyEclipse Database Explorer 管理 MySQL 数据库](#)。

4.1 功能一览

MyEclipse Database Explorer 支持连接到任何支持 **JDBC** 驱动的数据库，可以浏览数据库和表结构，浏览和修改表格数据，生成并执行 **SQL** 脚本，创建表格，修改索引等等。另外它还对 **Oracle**，**SQL Server**，**MySQL** 等数据库提供了额外的支持功能。

总的来说它有下面的一些功能：

- **Database Browser** 可以浏览数据库的结构
 - 可以树状浏览 **schema**，表，视图，**sequence** 等，...
- **ER Designer** 可以提供数据库结构的图形化表示
 - 自动布局表格及其关系
 - 对图形元素进行拖放操作
 - 修改表格的大小和关系之间的连接
 - 可以配置颜色和字体
 - 导出为 **JPG** 格式
- **Database Explorer** 透视图 （本章的重点内容，见图 4.1）
- 使用向导创建表格，外键和索引
- **SQL 编辑器**（见图 4.2）
 - 语法高亮
 - 表和列名高亮显示
 - 表和列名自动完成提示
 - 将编辑器和数据库连接相关联并执行 **SQL** 代码片段
- 支持管理多个数据库连接
- **SQL 生成工具**（见图 4.3）
- 支持多种数据库

Axion	Mimer SQL
Hypersonic DB	MySQL
InstantDB	Oracle

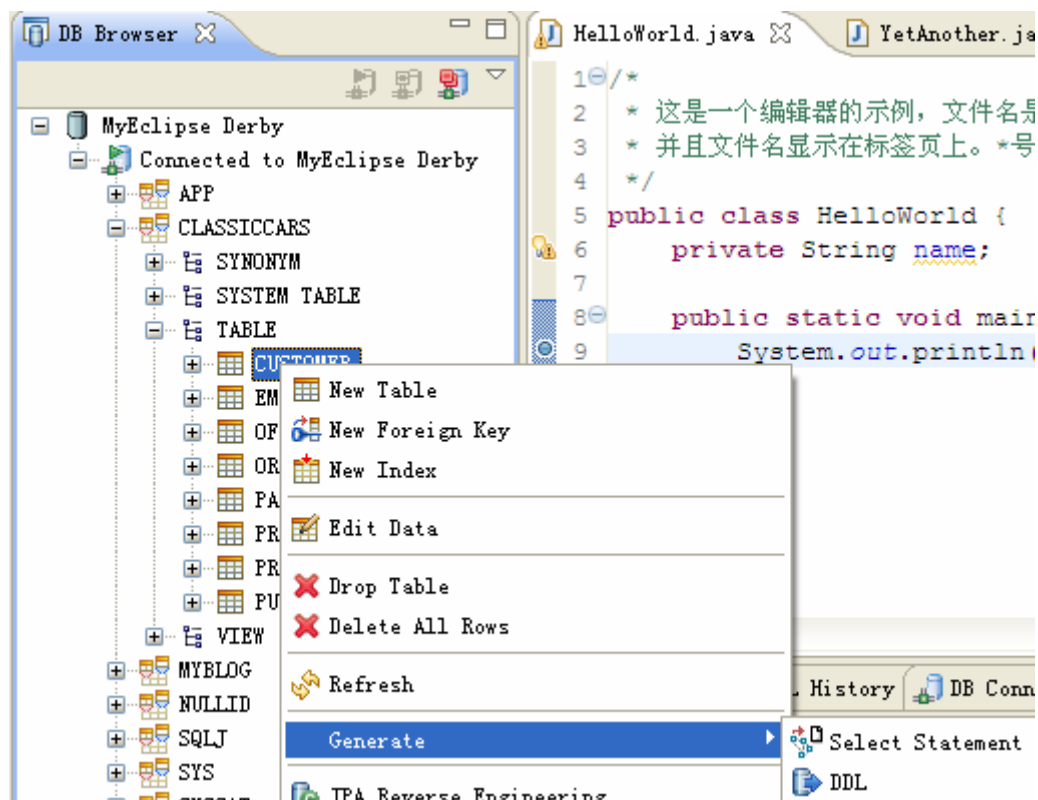


图 4.3 SQL 生成工具

4.2 使用 MyEclipse Database Explorer 透视图

4.2.1 介绍

Java 的企业应用开发离不开关系数据库。MyEclipse 提供了 **Database Explorer** (以下简称 **DE**, 数据库浏览器) 来支持数据库的开发, **DE** 提供了一系列的工具来支持数据库的开发。它包含下列一些功能:

- 支持 25 种预定义的 JDBC 驱动模版
- 创建多个数据库连接来连接到同一个或者多个数据库
- 数据库视图:
 - 显示数据库结构, 例如表格, 列, 视图等
 - 表格数据编辑器
 - 详细表格属性查看器
 - SQL 执行历史
 - 数据库连接属性查看器
- 支持代码完成的 SQL 编辑器
- 等等其它功能

透视图如下所示:

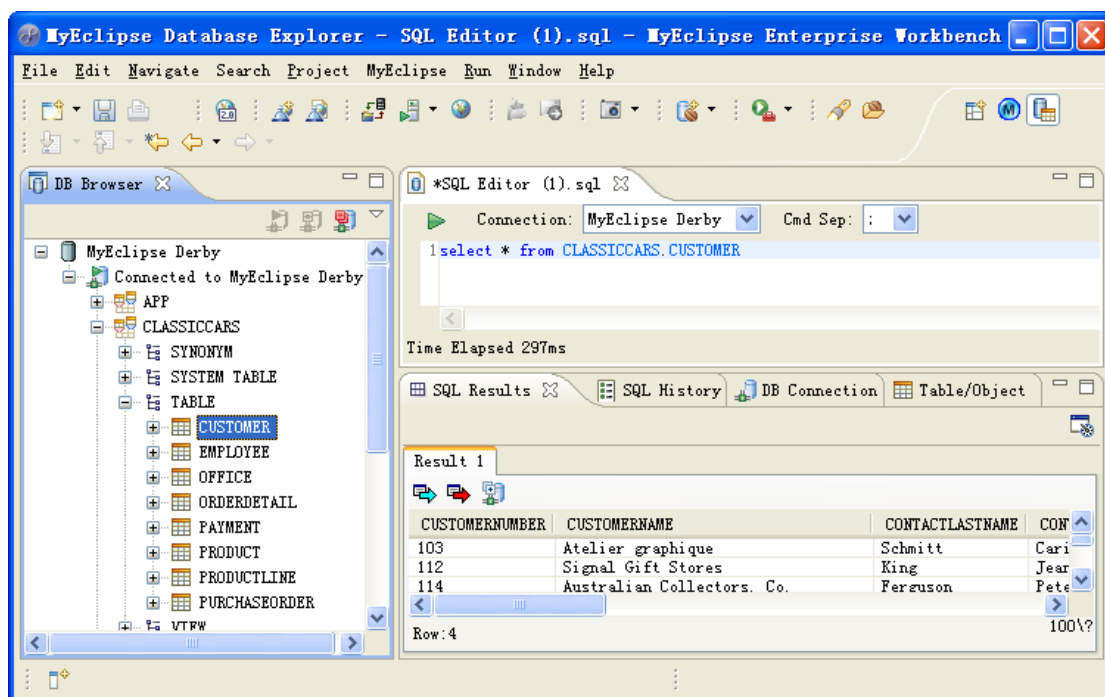


图 4.4 MyEclipse Database Explorer 透视图

我们这个教程将会首先连接到不需要配置的 MyEclipse Derby 数据库,之后再介绍如何连接到 MySQL 数据库。这样先易后难,便于了解。

4.2.2 连接到 MyEclipse Derby 数据库

首先需要启动数据库服务器,在 **Servers** 视图中选中 **MyEclipse Derby**,然后点击工具栏上的  按钮启动数据库服务器。

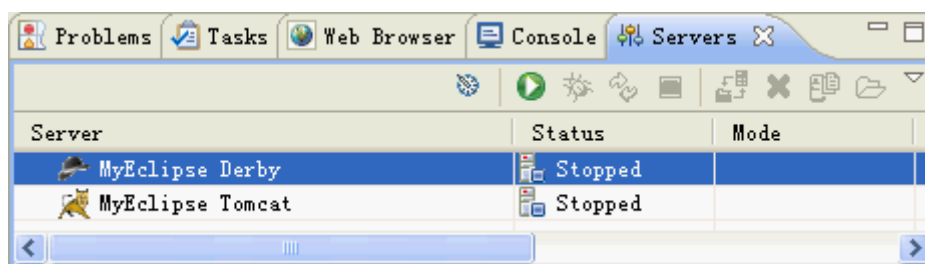



图 4.5 启动 MyEclipse Derby 数据库

之后在 **Console** 视图中如果打印出:

Apache Derby Network Server - 10.2.2.0 - (485682) 已启动并且已准备好
2007-12-10 09:42:37.312 GMT 时在端口 1527 上接受连接
那么数据库服务器就启动起来了。

4.2.3 切换到 MyEclipse Database Explorer 透视图

切换透视图有两种办法,如何切换请参考 [透视图 \(Perspective\) 切换器](#)。一种比较快办法是如那一节介绍的,点击工具栏上的  按钮可以显示多个透视图供切换,如图 3.3

所示，然后单击其中的**MyEclipse Database Explorer** 即可切换到此透视图；另一种办法是选择菜单 **Window > Open Perspective > Other > MyEclipse Database Explorer**来显示打开透视图对话框，然后点击**OK**按钮，如下图所示：

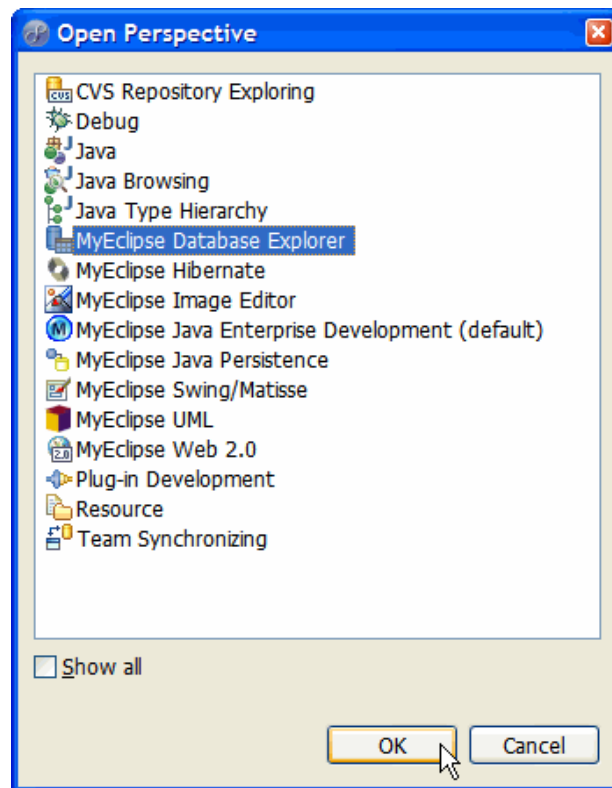



图 4.6 打开 MyEclipse Database Explorer 透视图

4.2.4 打开数据库连接

默认情况下 **DB Browser** 视图只有一个建好的数据库连接：**MyEclipse Derby**，这个连接的 URL 是 `jdbc:derby://localhost:1527/myeclipse`，用户名和密码都是 `classiccars`。可以从 **DB Browser** 视图打开现有的数据库连接，可以通过点击工具栏上的  按钮来打开数据库连接，或者点击右键从上下文菜单中选择 **Open connection...**也可以。之后就可以对数据库进行操作了。如下图所示：

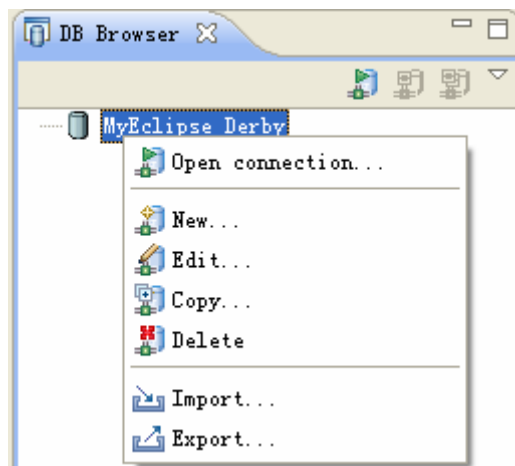



图 4.7 在 **DB Browser** 视图打开数据库连接

这时候如果要再打开一个到当前数据库的新连接，可以通过点击工具栏上的  按钮来打开数据库连接，或者点击右键从上下文菜单中选择 **Open another connection...** 也可以。

打开了数据库连接之后就可以操作数据库了。我们可以浏览数据库结构，执行 SQL，编辑数据，查看表结构等等，都可以。

打开其它数据库连接的方式和这里是一样的。

4.2.5 关闭数据库连接

可以通过 **DB Browser** 视图工具栏上的两个按钮来关闭数据库连接： 。第一个用来关闭单个连接（这个按钮只有在选中 **Connected to MyEclipse Derby** 节点后才可以），而第二个用来关闭所有数据库连接。

4.2.6 浏览数据库结构

打开了数据库之后就可以浏览数据库的结构。如下图所示首先可以看到的当前连接中的数据库 (schema) 列表：

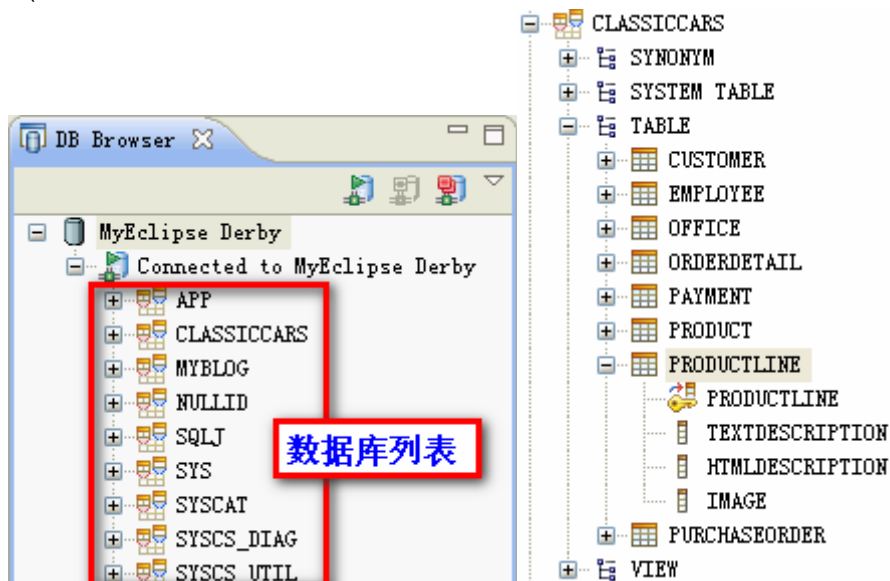



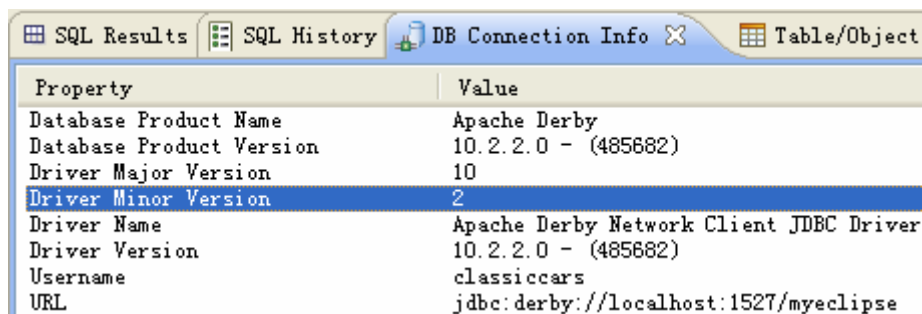


图 4.8 浏览数据库列表

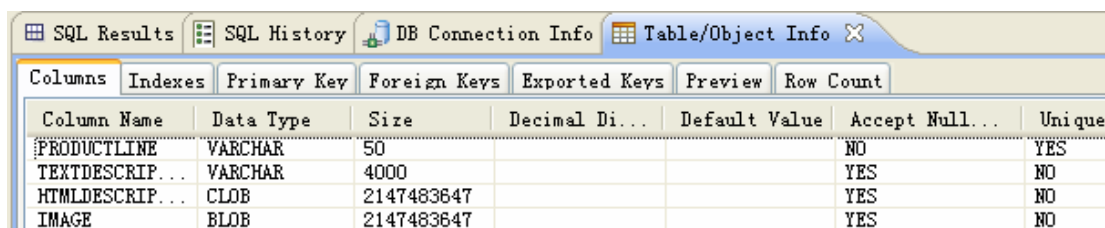
再展开 **classiccars** 这个目录可以显示数据库中的表，系统表，视图等信息，如图 4.7 所示。 图标之后显示的是表，展开表节点之后会显示当前表中的列信息（ 图标）和主键（ 图标）等信息。

另外这时候还有两个视图可以供你来了解数据库的更多信息，这两个视图分别是 **DB Connection Info**（数据库连接信息）和 **Table/Object Info**（表格/对象信息），如下面的两个图所示：



Property	Value
Database Product Name	Apache Derby
Database Product Version	10.2.2.0 - (485682)
Driver Major Version	10
Driver Minor Version	2
Driver Name	Apache Derby Network Client JDBC Driver
Driver Version	10.2.2.0 - (485682)
Username	classiccars
URL	jdbc:derby://localhost:1527/myeclipse

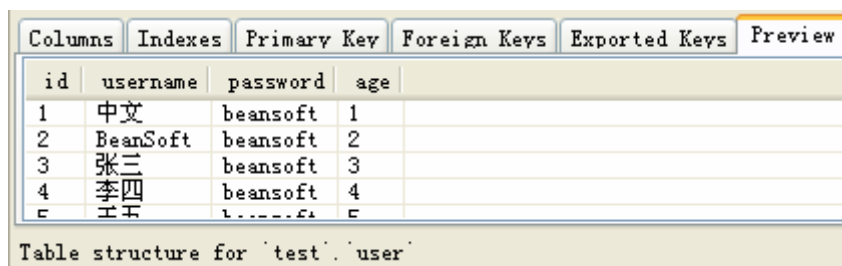
图 4.9 DB Connection Info 视图



Column Name	Data Type	Size	Decimal Di...	Default Value	Accept Null...	Unique
PRODUCTLINE	VARCHAR	50			NO	YES
TEXTDESCRIP...	VARCHAR	4000			YES	NO
HTMLDESCRIP...	CLOB	2147483647			YES	NO
IMAGE	BLOB	2147483647			YES	NO

图 4.10 Table/Object Info 视图

在 **Table/Object Info** 视图中有 7 个标签页，可以看到所选中的表格的详细信息。**Columns** 显示了表的列信息，**Indexes** 显示了索引信息，**Primary Key** 显示了主键，**Foreign Keys** 显示了外键，**Exported Keys** 显示了导出的主键，**Preview** 则显示了表格里面的数据，**Row Count** 显示了表格里数据的行数。其中 **Preview** 标签大概是最有用的标签吧，如下所示：



id	username	password	age
1	中文	beansoft	1
2	BeanSoft	beansoft	2
3	张三	beansoft	3
4	李四	beansoft	4
5	王五	beansoft	5

Table structure for 'test'. 'user'

图 4.11 预览表格数据

4.2.7 编辑和执行 SQL 代码段

现在可以启动 **SQL 编辑器**来编辑 **SQL**，有两种办法可以，第一种是在数据库连接节点上（**注意**：只能在这个节点上看到）点击右键，然后选择 **New SQL Editor**，即可打开编辑器，如下图所示：

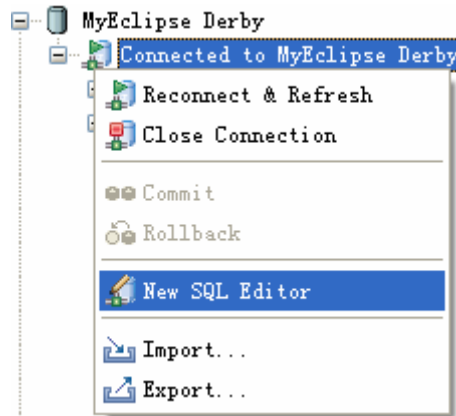


图 4.12 打开 SQL 编辑器

另一种办法是选择菜单 **File > New > SQL File**，然后打开新建 SQL 文件向导，之后就可以打开 SQL 编辑器了，如下图所示：

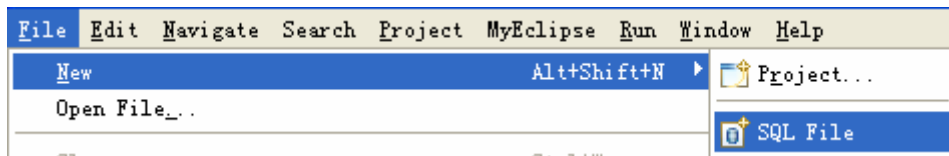


图 4.13 新建 SQL 文件

接着就可以看到 SQL 编辑器了，如下图所示：

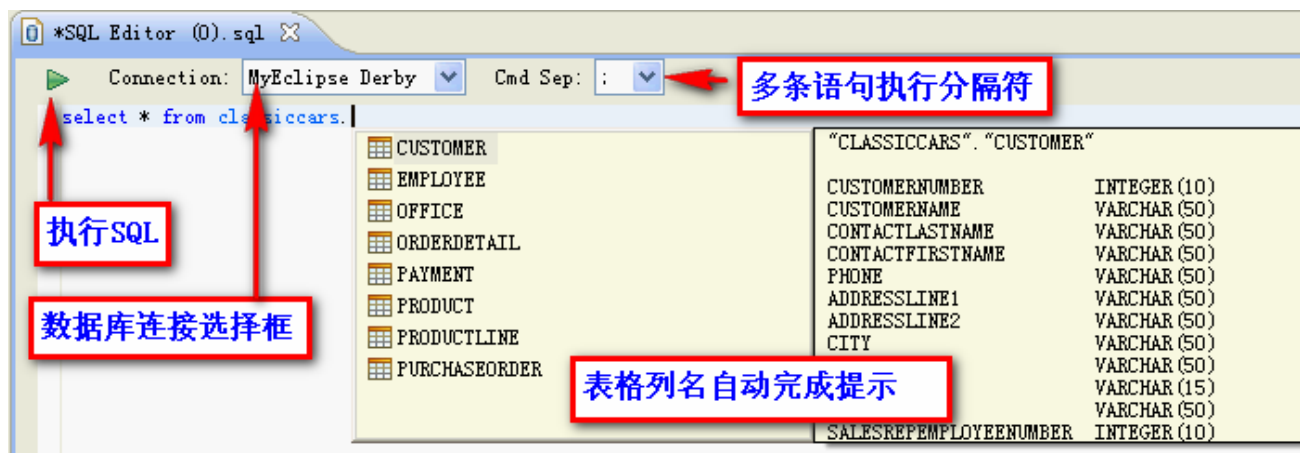


图 4.14 SQL 编辑器

这个编辑器对键入的 SQL 可以加颜色显示，并会显示自动提示功能，还可以执行键入的 SQL，点击 图标或者按下快捷键 **Ctrl + F9** 可以执行正在编辑的 SQL，执行的结果将会显示在 **Results** 视图中，如下图所示：

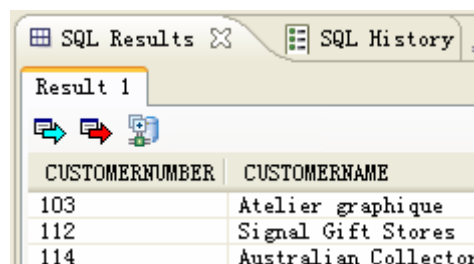


图 4.15 SQL 执行结果显示在 Results 视图

同时刚刚执行过的 SQL 会保存到 **SQL History** 视图，可以供你以后再次使用它，点击

鼠标右键可以看到操作菜单,可以打开(**Open in editor**),从历史中删除(**Remove from history**)或者复制到剪贴板(**Copy to Clipboard**)。如下图所示:

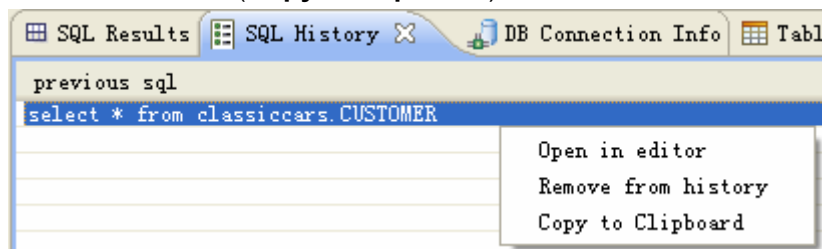


图 4.16 SQL 执行历史视图

4.2.8 生成实体关系 (ER) 图

数据库之间的表关系图可以生成出来,在数据库列表节点上点击右键,然后选择菜单**New ER Diagram**,就可以为当前数据库生成一个ER (Entity Relation, 实体关系)图,如图 4.17 所示。这之后会显示**Create New ER Diagram**对话框来选择保存要生成的ER图的位置,如图 4.18 所示,点击**OK**按钮,会显示选择要生成关系的表的对话框**Create ER Diagram**,如图 4.19 所示。在图中点击**Add-->** 和 **Add All-->**按钮可以选择要包含哪些表,之后点击**Finish**按钮即可生成关系图然后打开。最终生成的ER图如图 4.20 所示。因为这张图可能会非常大,建议打开视图**ER Diagram Overview**和**Outline**来帮助浏览。关于如何打开视图请参考 [视图](#)一节的内容。

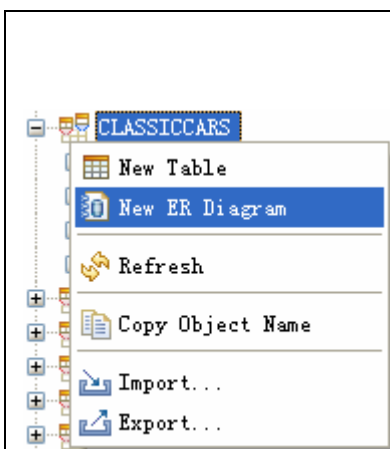


图 4.17 生成 ER 图的菜单

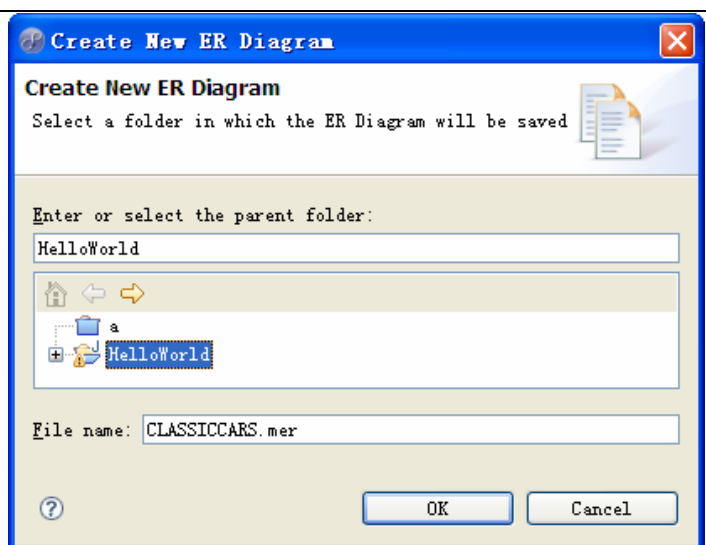


图 4.18 新建 ER 图对话框

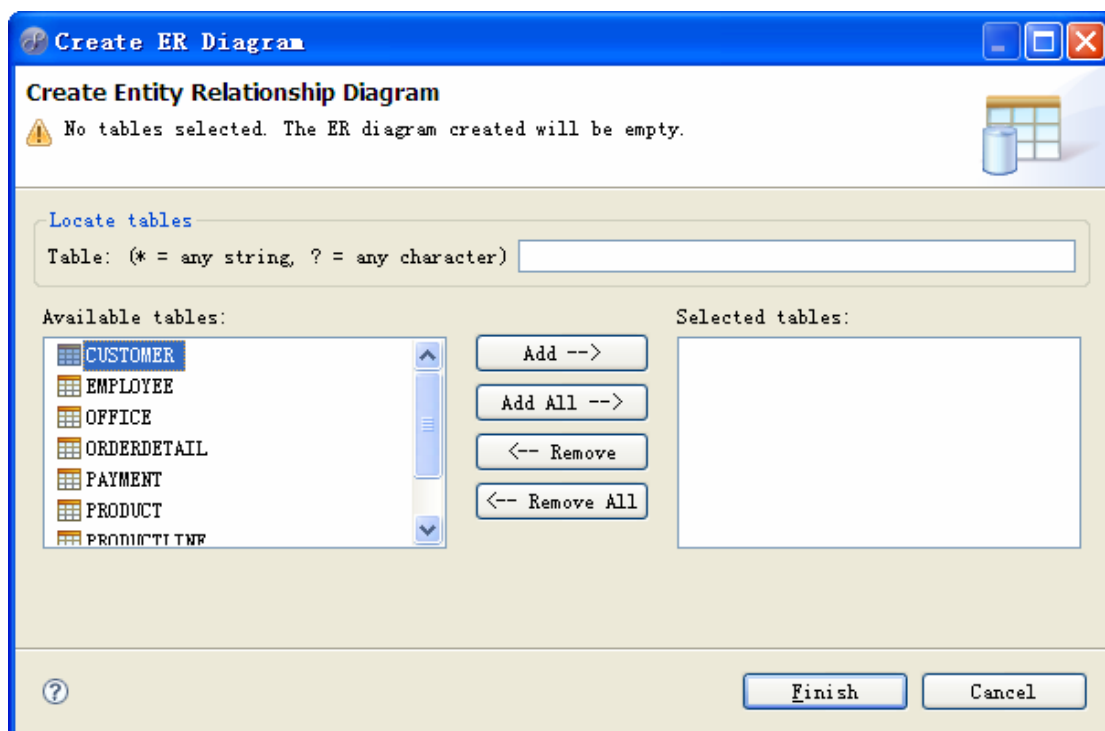


图 4.19 选择 ER 图要包含的表格

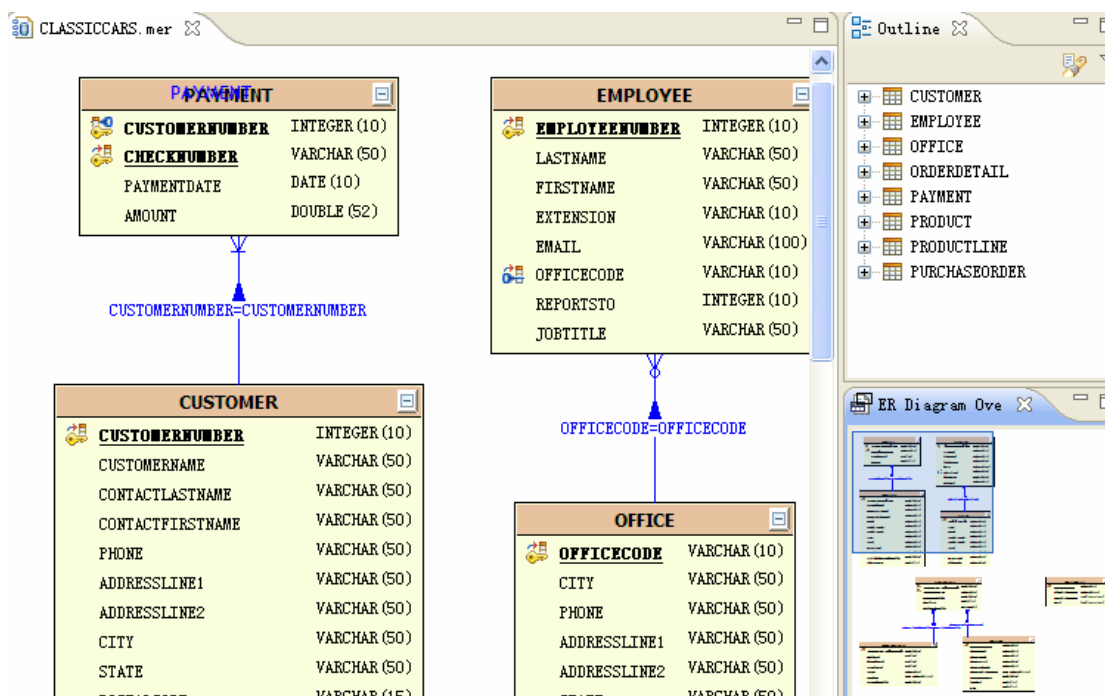


图 4.20 生成的 ER 图

在这个图上我们可以通过拖放来调节表的位置和大小。在图上点击右键可以选择菜单 **Export As JPEG...** 来将这个图导出为图片文件便于以后交流用。

4.2.9 编辑表格数据

数据库浏览器提供了编辑数据的功能，在 **DB Browser** 视图中选中表，然后在上下文菜单中选中 **Edit Data**，如下图所示：

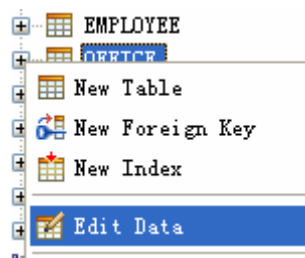


图 4.21 在上下文菜单选中 Edit Data

接着就可以在新显示的 **Edit table "CLASSICCARS"."OFFICE"** 视图中修改表格数据了，如下图所示：

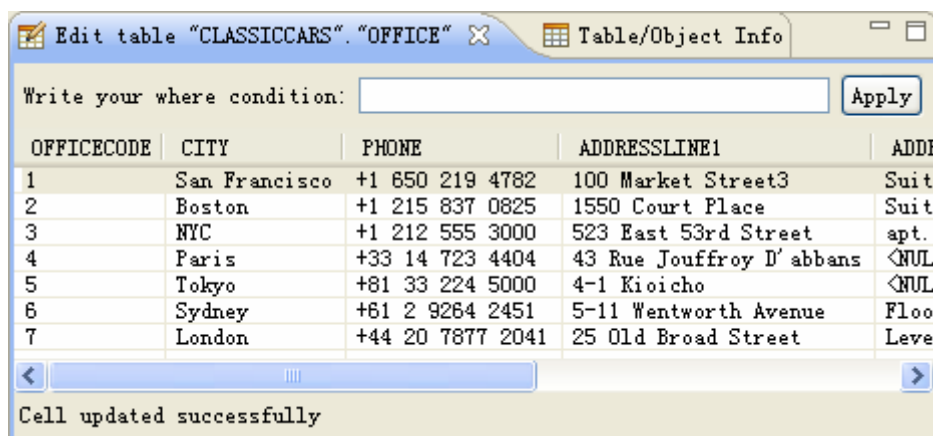


图 4.22 编辑表格数据

4.2.10 清空表格数据

如图 4.21 所示，在 **DB Browser** 视图中选中表，然后在上下文菜单中选中 **Delete All Rows**，就可以删除表里面的所有数据，而表本身不会被删除。

4.2.11 创建和删除表格

要创建表格，可以在 SQL 编辑器中输入 **create** 语句，然后执行，也可以在 **DB Browser** 视图上下文菜单中选中 **New Table**，然后使用 MyEclipse 提供的新建表格设计器来建表。如图 4.23 所示。在向导中点击 **Add...** 按钮可以添加新的列定义，点击 **Edit...** 可以修改选中的列定义。

要删除表格，可以在表上右键点击选中 **Drop Table** 即可。

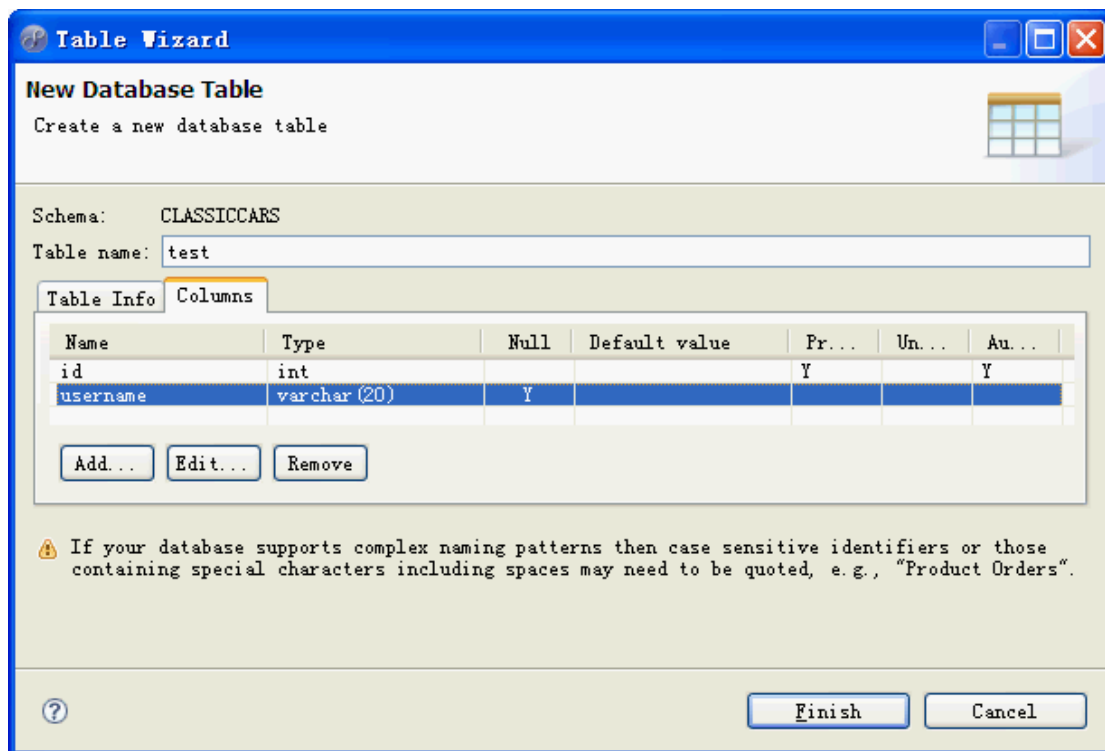


图 4.23 新建表格向导

4.2.12 创建和删除外键

如图 4.21 所示，在 **DB Browser** 视图中选中表，然后在上下文菜单中选中 **New Foreign Key**，之后就可以启动创建外键的向导，参考图 4.25。当然用 SQL 语句来创建也许会更快，弹出的对话框的详细意义请参考 SQL 语法。

要删除外键则是在 **Table/Object Info** 视图的 **Foreign Keys** 标签中的外键列表上点击右键，然后选择 **Drop Foreign Key** 即可。如下图所示：

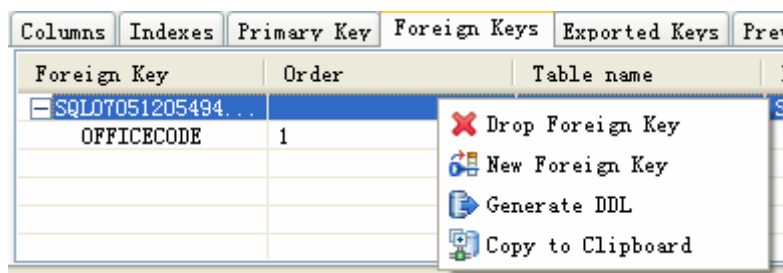


图 4.24 删除外键

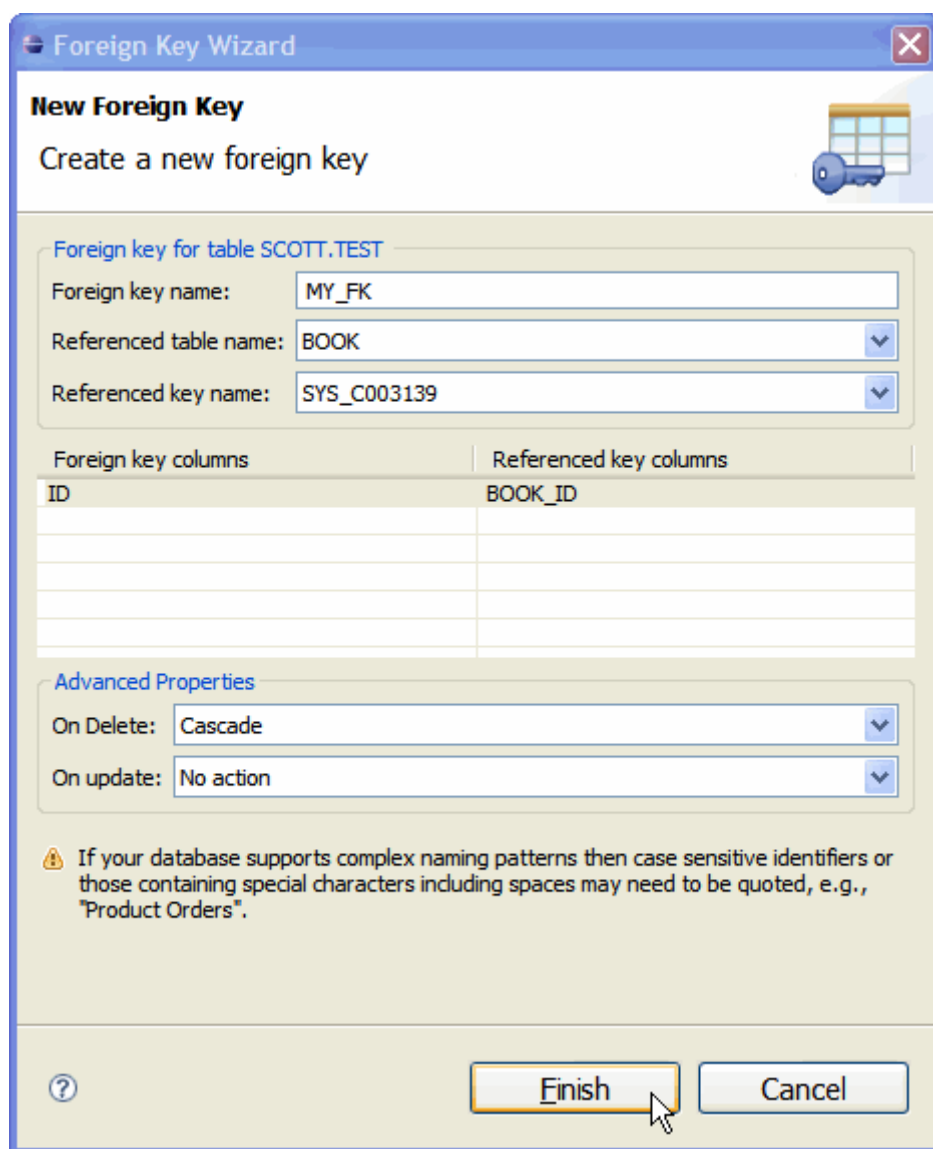


图 4.25 创建外键向导

4.2.13 创建和删除索引

如图 4.21 所示,在 **DB Browser** 视图中选中表,然后在上下文菜单中选中 **New Index**,之后就可以启动创建外键的向导,见图 4.26。当然用 **SQL** 语句来创建也许会更快,弹出的对话框的详细意义请参考 **SQL** 语法。

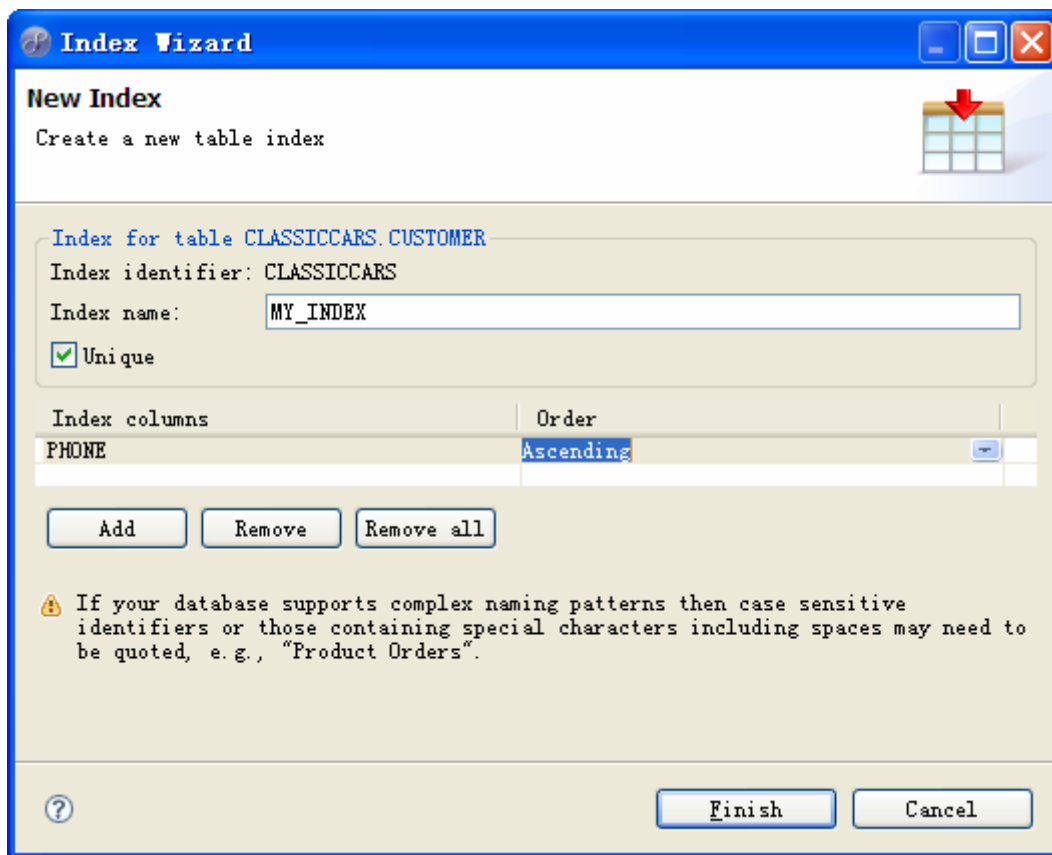


图 4.26 创建索引向导

要删除索引则是和上面删除索引的操作十分相似，在 **Table/Object Info** 视图的 **Indexes** 标签中的索引列表上点击右键，然后选择 **Drop Index** 即可。

4.2.14 生成 SQL 语句

在 **DB Browser** 视图中选中表，然后在上下文菜单中选中 **Generate**，在子菜单中则可以生成 **SELECT** 语句或者 **DDL**（建表语句），同样在 **Table/Object Info** 视图的 **Indexes**，**Foreign Keys** 等标签中也有类似的菜单（参考图 4.24），点击右键即可看到对应的生成 SQL 语句的菜单。生成后的 SQL 将会自动放在新的 SQL 编辑器中。

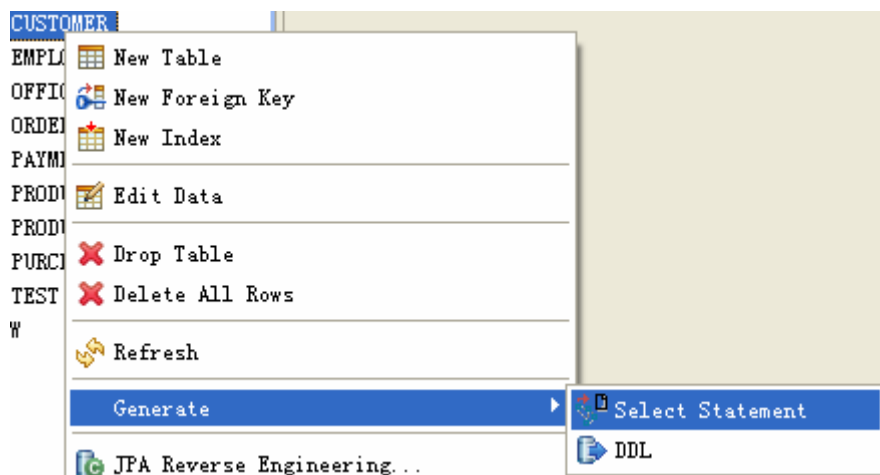


图 4.27 生成 SQL 语句

4.2.15 建立到 MySQL 数据库的连接

在上面我们介绍的内容大多是连接到 MyEclipse 自带的 MyEclipse Derby 连接中完成的，那么在实际开发中一般是连到自己公司所使用的数据库，在本书中我们将会用 MySQL 来完成大部分的工作，因此本节介绍如何连接到 MySQL。请多花点时间放到本节的内容上。

首先确保MySQL服务器已经启动并且已经下载了相应的JDBC驱动，详情请参考[MySQL 5 数据库服务器下载，安装和运行（可选）](#)一节的内容。

要新建连接，如下图所示在 **DB Browser** 视图的上下文菜单中选择 **New...**来启动新建驱动向导，向导如图 4.29 所示。

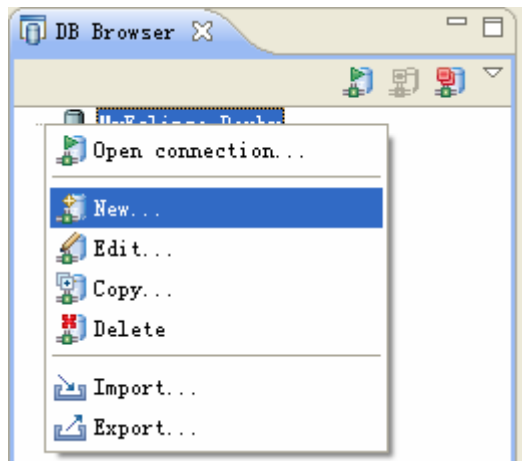


图 4.28 启动新建驱动向导

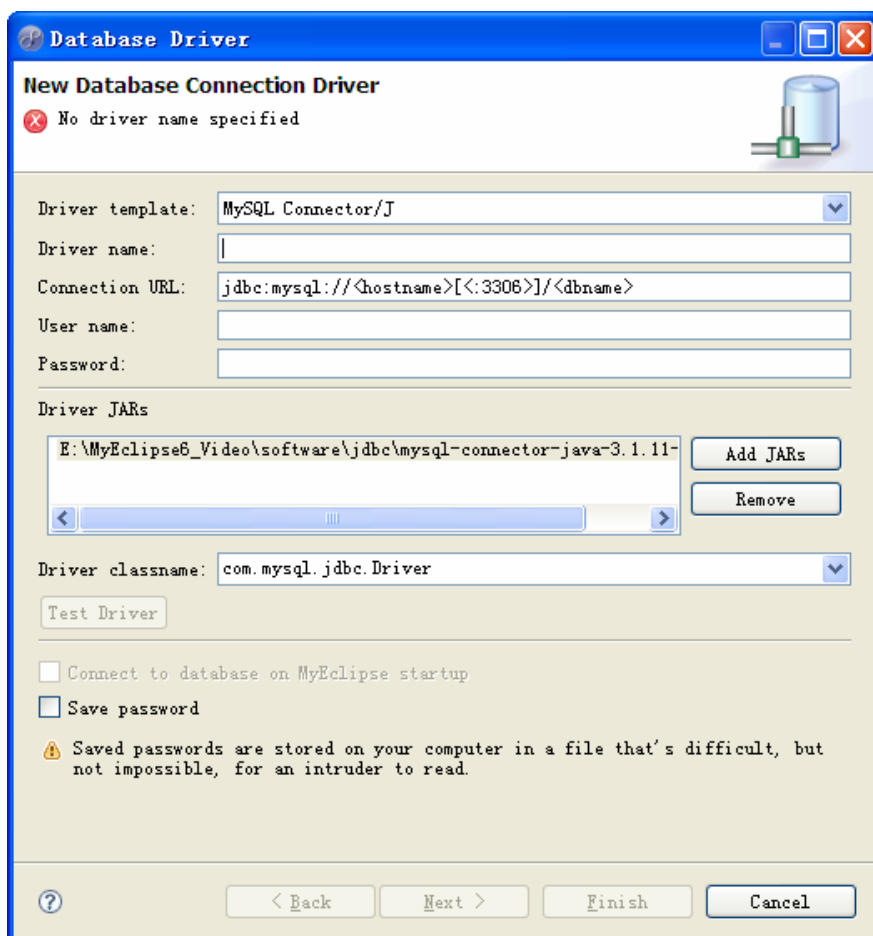


图 4.29 数据库驱动向导

那么这个向导对话框中有很多输入框，这些输入框的意义如下表所示：

输入框	必填	说明
Driver name	是	数据库连接的名称，将会显示 Database Browser 视图和 SQL 编辑器中。
Connection URL	是	数据库连接字符串，每个数据库都有自己的连接字符串格式，例如 jdbc:mysql://<主机名>[:3306]/<数据库名>。
User Name	否	数据库连接账户的登录用户名。
Password	否	数据库连接账户的登录密码。
Driver JARs	是	用户自己提供的 JAR 文件列表，这些文件加入驱动管理器的类路径。如果在默认的 Java 类路径中没有合适的数据库驱动类，那么对应的 JAR 文件必须被加到这里。点击 Add JARs 按钮浏览并选择对应的 JAR 加入到这里来。
Driver classname	是	JDBC 驱动类的完整类名。这个类必须能够在 Java 类路径（Eclipse 启动时候的那个）和 Driver JAR 列表中找到。
Save Password	否	选中这个选项的时候，密码会保存起来。否则每次打开数据库连接的时候都会提示你输入密码。
Open on Eclipse Startup	否	选中这个选项的话每次 Eclipse 启动的时候都会自动连接到这个数据库。

表 4.1 驱动向导的对话框输入值

那么要连接到MySQL数据库，我们这里需要在**Driver name**中输入mysql5（这个名字是任意的），而在**Driver Template**下拉框中选中MySQL Connector/J，**Connection URL**中输入 jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=GBK，**User Name**中输入 root，**Password**输入留空，**Driver JARs**则点击Add JARs按钮从电脑上找到下载的mysql-connector-java-xxx-bin.jar然后点击文件对话框的打开按钮将它添加进来，**Driver classname**则点击下拉列表选择com.mysql.jdbc.Driver，然后点击Finish按钮即可完成创建过程。至此，就成功连接到了MySQL数据库了，以后就可以参考 [连接到MyEclipse Derby数据库](#)一节的内容来连接到MySQL数据库了，之后的其它操作都是相同的。

如果要修改连接的信息，可以在 **DB Browser** 视图的上下文菜单中选择 **Edit...** 来启动驱动修改对话框，这一步操作如图 4.28 所示。

4.3 小结

本章介绍了MyEclipse 数据库浏览器的使用方法，这么多内容可以只关注 [切换到MyEclipse Database Explorer透视图](#)，[打开数据库连接](#)，[浏览数据库结构](#)，[建立到MySQL数据库的连接](#)，这些节的内容就可以。

4.4 参考资料

在 [MyEclipse 教程库（英文）](#) 可以看到很多官方的MyEclipse教程。

有关JDBC 的内容可以从Sun Microsystems官方网站的这个 [链接](#) 来学习。

当然更多的信息可以用搜索引擎例如 www.google.com或者 www.baidu.com 来搜索。

第五章 开发 JDBC 应用

企业的开发离不开数据库的操作，用Java如何访问数据库这也是一个难点，当时我读书的时候初学Java，试了好多次才成功的向数据库中插入了数据。那么本节内容就讲解如何在普通项目中用JDBC访问数据库，使用的数据库是MySQL 5.0，用别的数据库或者Derby也可以完成这个练习，只是建表语句略微不同，可以参考第四章的内容用 [新建表格向导](#)来完成。

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 1 安装运行 Mysql, MySQL-Front 管理, JDBC HelloWorld 开发](#)。

5.1 系统需求

本节内容需要安装MySQL数据库，请参考 [MySQL 5 数据库服务器下载, 安装和运行\(可选\)](#) 一节内容来安装并启动MySQL数据库以及获得对应的JDBC驱动jar文件。


也可以直接用 MyEclipse Derby 数据库完成这个练习。

5.2 创建数据库表格

对于多数的项目来说，基本上前期的工作就是进行需求分析，分析的结果一是功能模块，二可能就是实体或者对象，实体最终就会映射到数据库设计中去。所以这里先从创建数据库开始。

首先需要创建一个学生表，建表的 SQL 如下所示：

```
CREATE TABLE Student (  
  id int NOT NULL auto_increment,  
  username varchar(200) NOT NULL,  
  password varchar(20) NOT NULL,  
  age int,  
  PRIMARY KEY (id)  
) ENGINE=MyISAM DEFAULT CHARSET=GBK
```

注意：这个是MySQL数据库的建表语句。这个表有一个自增的ID列作为主键，还有用户名，密码和年龄三个列，最后一句`ENGINE=MyISAM DEFAULT CHARSET=GBK`指定了表的默认字符集是GBK中文字符，这一句是MySQL特有的语法。请参考第四章的内容在 **MyEclipse Database Explorer** 透视图的 **DB Browser** 视图中打开 `mysql5` 这个连接，展开并选中 `test` 数据库，然后参考 [编辑和执行SQL代码段](#) 一节的内容打开SQL编辑器，将上述代码粘贴进去，然后点击运行按钮  来创建这个表。

如果要用 Derby 数据库做这个练习，对应的建表语句是：

```
CREATE TABLE Student (  
  id int NOT NULL generated always as identity,  
  username varchar(200) NOT NULL,  
  password varchar(20) NOT NULL,  
  age int,
```

```
PRIMARY KEY (id)
)
```

需要指出的是，并非所有数据库都支持自增类型的主键，而且不同的数据库实现主键生成的关键字也是没有统一规定的，例如 Oracle 使用一个叫 `sequence` 的概念来生成主键。那么通用的不带主键生成器的建表语句如下所示：

```
CREATE TABLE Student (
  id int NOT NULL,
  username varchar(200) NOT NULL,
  password varchar(20) NOT NULL,
  age int,
  PRIMARY KEY (id)
)
```

用 MyEclipse 建表的操作过程如下图所示：

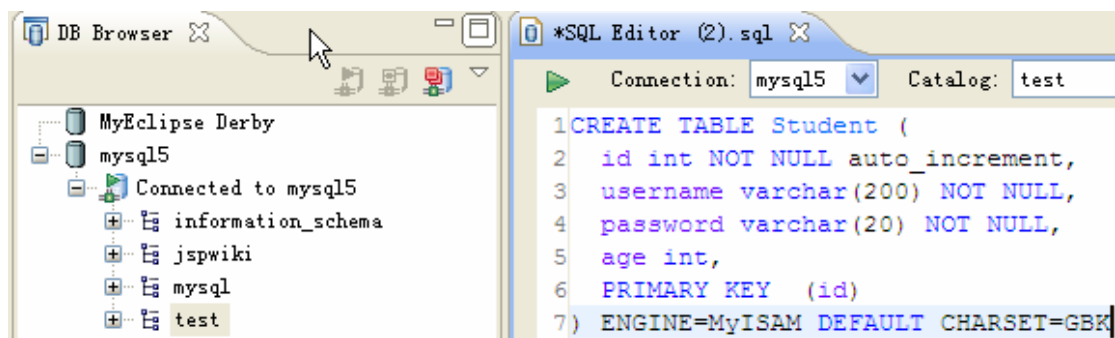


图 5.1 用 MyEclipse Database Browser 创建表格

当然对于 MySQL 数据库来说，也可以用 MySQL-Front, Navicat 等工具来方便的建表，所以本节的内容也是可能的操作方式之一。

5.3 创建 Java 项目

请参考 [使用 Eclipse/MyEclipse 来编写，编译并运行 Java 程序](#) 一节来创建一个名为 JDBCHelloWorld 的 Java 项目。

首先确保已经打开了 MyEclipse Java Enterprise 透视图，然后从菜单栏选择 **File > New > Java Project**，接着会打开 **New Java Project** 向导对话框，在 **Project name** 中输入 *JDBCHelloWorld*，点击 **Finish** 按钮关闭对话框，这样一个 Java 项目就建立完毕了。稍等片刻会弹出一个切换透视图的对话框，为了避免造成更多的麻烦，我们一般选择 **No** 按钮就可以了。

5.4 添加 JDBC 驱动到 Build Path

要连接数据库必须要将 JDBC 驱动类库加入到项目的 **Build Path** 中，详情可以 [参考复制项目中的文件](#) 和 [快速加入、删除 jar 包到 Build Path](#) 一节的内容。不同的数据库的 JDBC 驱动类库文件是不一样的。首先需要找到对应的驱动，MySQL 的可以从官方网站下载，而

MyEclipse Derby的则位于Windows当前用户的配置文件目录下面,例如C:\Documents and Settings\BeanSoft\myeclipse\libs\derby_10.2.2.0\derbyclient.jar。在Windows的文件浏览器中选中文件并复制到剪贴板(选择菜单**编辑**→**复制**,或者在文件上点击右键选择菜单**复制**,或者按下组合键**Ctrl + C**),然后点击任务栏切换到MyEclipse的窗口,在**Package Explorer**视图中选中刚刚创建的**JDBCHelloWorld**项目,接着可以进行粘贴操作,点击菜单**Edit** → **Paste** 或者在**JDBCHelloWorld**项目节点的上下文菜单中选择**Paste**,或者按下快捷键**Ctrl + V**,这时候驱动程序的JAR文件例如mysql-connector-java-3.1.11-bin.jar就复制并添加到了当前项目中。点击一下来选中这个jar文件,然后单击鼠标右键,选择菜单**Build Path** → **Add to Build Path** 就可以将这个jar文件加入Build Path中,如下图所示:

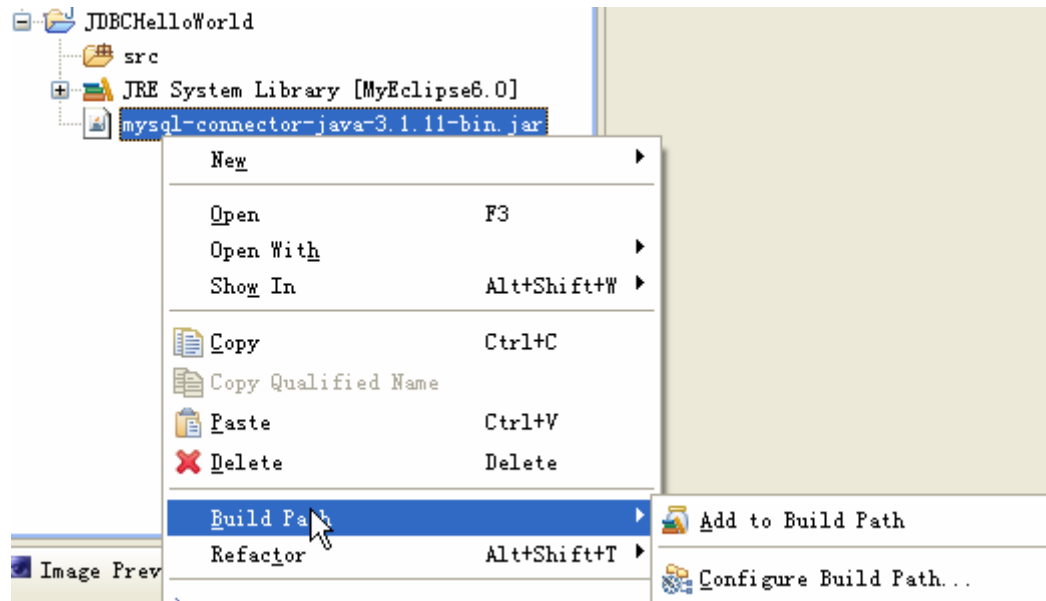


图 5.2 将驱动 jar 文件加入 Build Path

5.5 编写 JDBC 访问类

至此开发环境已经准备好了,所缺的就是写 Java 程序来访问数据库了。

接着选择菜单 **File > New > Class**, 然后(New Java Class)新建类的对话框就出现了,参考图 2.4,接着在 **Name** 输入框中输入 **JDBCHelloWorld**, 点击完成。接着将编辑器里面的代码修改成如下所示:

```
/*
 * JDBCHelloWorld.java
 * 版权所有 2007 刘长炯(BeanSoft@126.com)
 * Blog: http://www.blogjava.net/beansoft/
 * 本代码协议: GPL
 */
import java.sql.SQLException;
/**
 * 第一个 JDBC 的 HelloWorld 程序, 数据库访问 MySQL.
 * @author BeanSoft@126.com
 * @version 0.3 2007-12-12
 */
```

```

public class JDBCHelloWorld {

    public static void main(String[] args) {
        // 1. 注册驱动
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } // Mysql 的驱动

        //先定义变量，后使用和关闭
        java.sql.Connection conn = null; //数据库连接
        java.sql.Statement stmt = null; //数据库表达式
        java.sql.ResultSet rs = null; //结果集

        try {
            // 2. 获取数据库的连接
            conn = java.sql.DriverManager.getConnection(

                "jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=GBK", "root", null); // root是用户名，密码为空

            // 3. 获取表达式
            stmt = conn.createStatement();

            // 执行插入数据的 SQL
            stmt.executeUpdate("insert into Student(username, password, age) values('张三', '1234', 20)");

            // 4. 执行 SQL
            rs = stmt.executeQuery("select * from Student");

            // 5. 显示结果集里面的数据
            while(rs.next()) {
                System.out.println("编号=" + rs.getInt(1));
                System.out.println("学生姓名=" + rs.getString("username"));
                System.out.println("密码=" + rs.getString("password"));
                System.out.println("年龄=" + rs.getString("age"));
            }
        }
    }
}

```

```

        // 执行删除数据的 SQL
        stmt.executeUpdate("delete from Student");

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // 6. 释放资源，建议放在finally语句中确保都被关闭掉了
        try {
            rs.close();
        } catch (SQLException e) {
        }
        try {
            stmt.close();
        } catch (SQLException e) {
        }
        try {
            conn.close();
        } catch (SQLException e) {
        }
    }
}
}

```

jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=GBK 这个 是连接到 MySQL 数据库的 JDBC URL，关于这段代码中的?useUnicode=true&characterEncoding=GBK 是说以支持国际字符集（Unicode）的方式并以 GBK 中文字符集连接到数据库，如果安装时候选择的数据库的默认字符集是 UTF-8，那么这个地方把 GBK 换称 UTF-8 就可以了。当然更快的办法是你将这段代码直接复制，然后在 MyEclipse 中点击菜单 **Edit → Paste** 就可以生成这个类文件了。当你的代码编写完毕后，MyEclipse 会自动将代码编译成类文件。

接下来就可以运行写好的类了，选择菜单 **Run → Run** 或者按下快捷键 **Ctrl+F11**，就可以看到 Eclipse 会自动调用 Java 解释器，然后在 **Console** 视图中输出：

```

编号=1
学生姓名=张三
密码=1234
年龄=20

```

这个程序就执行成功了。再执行一次将会输出两行数据：

```

编号=1
学生姓名=张三
密码=1234
年龄=20
编号=2
学生姓名=张三

```



```
密码=1234
```

```
年龄=20
```

。可以看到 ID 是自动生成的。

如果要连接到 Derby 数据库，这段程序稍作修改即可，参考红色字体：

```
/*
 * JDBCHelloWorld.java
 * 版权所有 2007 刘长炯(BeanSoft@126.com)
 * Blog: http://www.blogjava.net/beansoft/
 * 本代码协议: GPL
 */
import java.sql.SQLException;
/**
 * 第一个 JDBC 的 HelloWorld 程序，数据库访问 MySQL.
 * @author BeanSoft@126.com
 * @version 0.3 2007-12-12
 */
public class JDBCHelloWorld {

    public static void main(String[] args) {
        // 1. 注册驱动
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        .....

        try {
            // 2. 获取数据库的连接
            conn = java.sql.DriverManager.getConnection(
                "jdbc:derby://localhost:1527/myeclipse",
                "classiccars", "classiccars");
            .....
        }
    }
}
```

5.6 小结

本节内容讨论了如何使用 MyEclipse 开发用 JDBC 访问数据库的 Java 类，基本上包括安装数据库，建表，创建项目，加入驱动类，创建类，运行代码（相当于测试）这几个步骤。

因为实际工作中很多公司因为项目周期长的原因，因此纯 JDBC 方式的数据库访问代码可能会经常看到。

注意：本节内容并不能代替您来学习 JDBC，请阅读书籍等来获取更多信息。关于 JDBC 分页，PreparedStatement 等高级内容可以从 Google 搜索。

5.7 参考资料

[基于 JDBC 2.0 驱动的分页代码实现](#)

[学习 SQL 语法的好资料: Transact-SQL 参考](#)

这个资料是我上大学时候一直看到现在的，既可以作为学习资料，更可以作为开发时候的速查手册。具体来说就是安装了 SQL Server 2000 后里面自带的帮助文档的一部分。

第六章 管理应用服务器

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 5 MyEclipse 6 + Tomcat 6 Servlet 入门开发](#)。

6.1 简介

MyEclipse 支持对多达 20 种（Application Server）应用服务器的启动，停止，发布，重新发布，测试，调试，查看服务器输出信息等等。这些服务器包括：Glassfish, JBoss, Jetty, Jonas, JRun, Oracle, Orion, Resin, Sun App Server, Tomcat, BEA WebLogic Server, IBM WebSphere 等等。MyEclipse 通过对每个服务器配置连接器（Connector）来管理这些服务器。

不过，MyEclipse 只支持在本机安装的服务器的管理，不能对远程服务器进行管理。另外，这些服务器最好通过 JDK 来启动，**尽量不要使用 JRE**。不过，在实践中发现 Tomcat 5, Tomcat 6 和 JBoss 4.2 是可以通过 JRE 正常启动和运行的。

只有 MyEclipse Java EE 类型的项目（Enterprise, EJB 和 WEB）才能用 MyEclipse 来进行发布。

MyEclipse 6 自带了一个 Tomcat 服务器，因此除了开发 EJB 项目外，可以不用单独下载和安装 Tomcat 等服务器。

6.2 Servers 视图

Servers 是一个特殊的 MyEclipse 视图，可以查看全面所有配置的应用服务器连接器的状态。这个视图是 **MyEclipse Java Enterprise** 透视图的一个标准部分（参考图 6.1）。

从菜单栏选择 **Windows>Show View>Other>MyEclipse>Servers**，就可以打开 **Servers** 视图。这个视图的工具栏按钮分成两大部分（见途中红线框中的部分），左边的一侧可以配置服务器，启动和关闭，右边的一侧则管理发布到选中的服务器上的 J2EE 项目。表 6.1 列出了每个工具栏按钮功能的简要描述。

注意：这个图中的 **MyEclipse Derby** 是内置的数据库服务器，而 **MyEclipse Tomcat** 则是内置的 JSP 服务器。

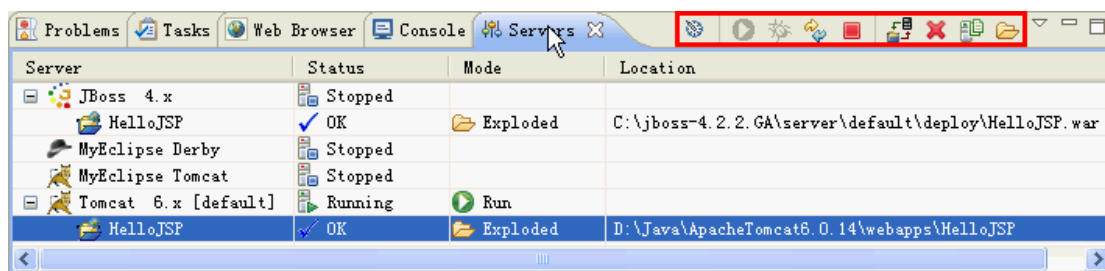


图 6.1 Servers 视图以及工具栏按钮

	以运行模式启动所选应用服务器
	以调试模式启动所选应用服务器，支持热替换调试
	重启服务器（先停止，再启动）
	停止所选应用服务器
	配置所选应用服务器的连接器
	打开 Deployment Manager（发布管理器）
	删除已经发布的 J2EE 项目
	重新发布选中的已发布过的 J2EE 项目（做开发是比较有用）
	打开文件浏览器来浏览服务器的自动发布目录以及发布后的项目文件所在的位置

表 6.1 服务器管理工具栏功能描述

6.3 浏览应用服务器连接器

想连接到自己用的应用服务器的话，可以点击 Servers 视图的工具栏上的  按钮来打开服务器配置对话框，或者从主菜单中选择 **Window > Preferences**，这样就可以打开配置对话框。

当配置对话框打开后，可以展开左侧的树，选择 **MyEclipse > Servers**，然后就可以看到可用的应用服务器连接器了。如下图所示：

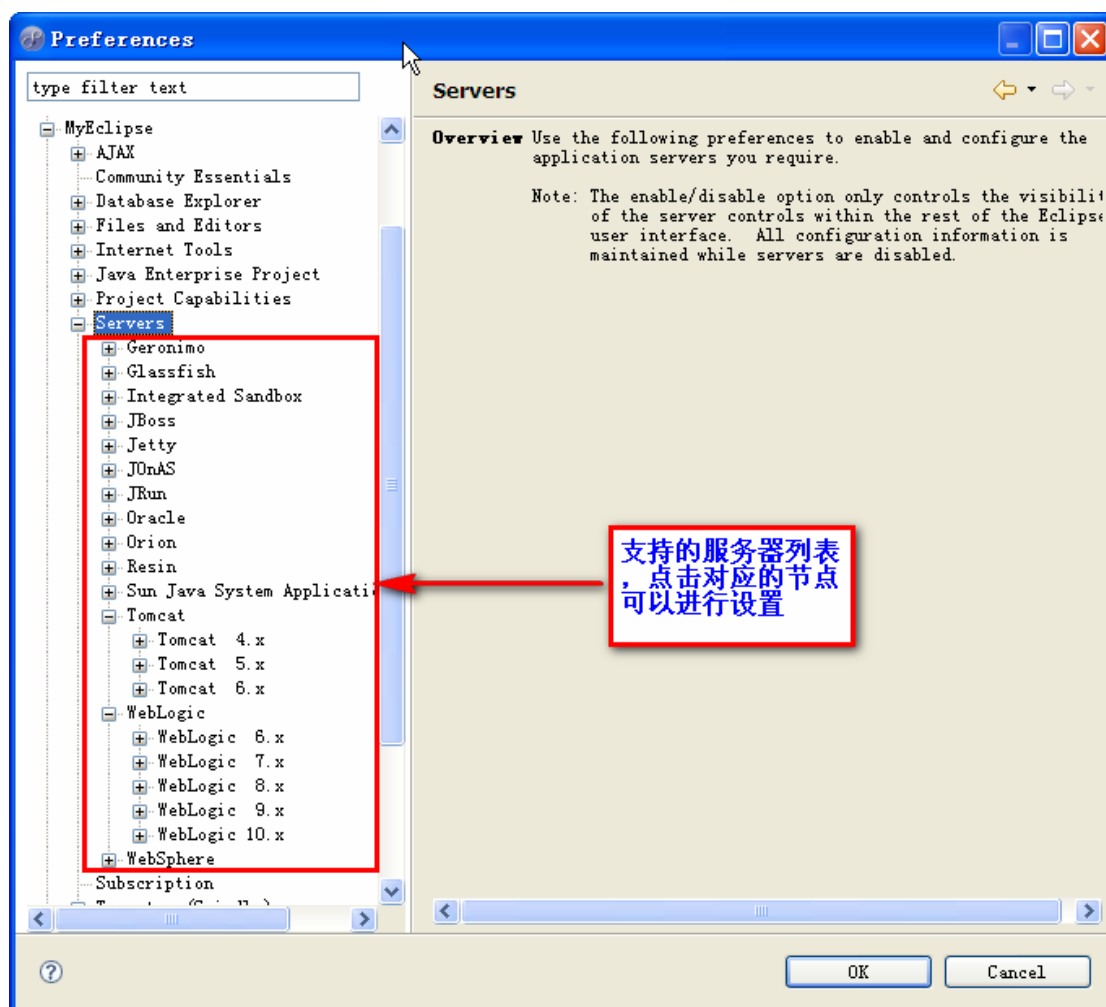


图 6.2 查看服务器连接器列表

下图则展示了一个配置好的 JBoss 服务器的例子：

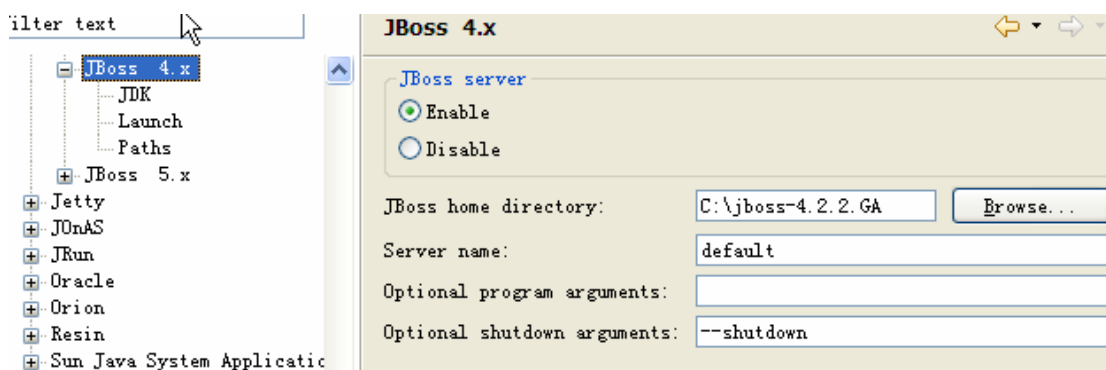


图 6.3 JBoss 4 的配置

除了服务器的安装目录外，还可以设置一些额外的信息和启动参数等等。

6.4 配置连接器

配置一个服务器基本上来说需要 2 到 3 步，包括：

1. 配置服务器的安装信息；
2. 启用连接器；

3. 指定启动时需要的 JDK 或者 JRE（可选）。

注意：一些特殊的服务器需要额外的配置信息，这时候你可以去查看对应服务器的使用说明书和帮助文档以及从技术支持人员那里获得帮助（可能会付费）。

下面就以配置常用的 Tomcat 5 为例来介绍如何配置连接器。

6.4.1 第 1 步 配置服务器的安装信息

首先我们选中服务器节点 **Tomcat 5.x**，然后点击 **Browse...** 按钮来选择 Tomcat 的安装根目录，如图 6.4 所示。接下来 MyEclipse 会尝试根据服务器的默认设置填入其它的设置信息，如图 6.5 所示。在这个例子中选择的安装目录是 *E:\apache-tomcat-5.5.20-cn*。

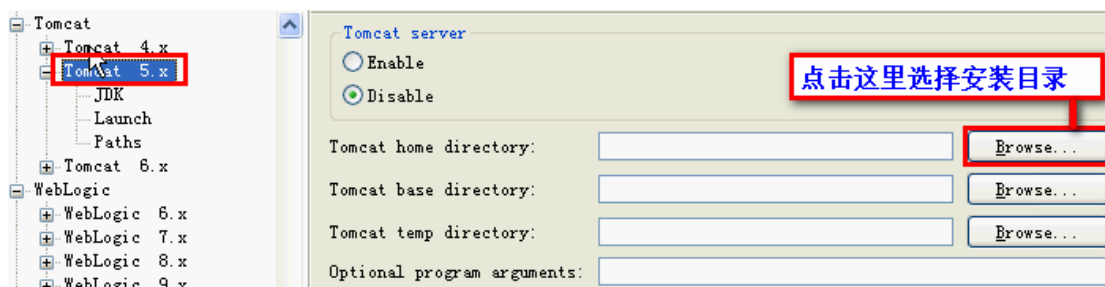


图 6.4 选择 Tomcat 的安装目录

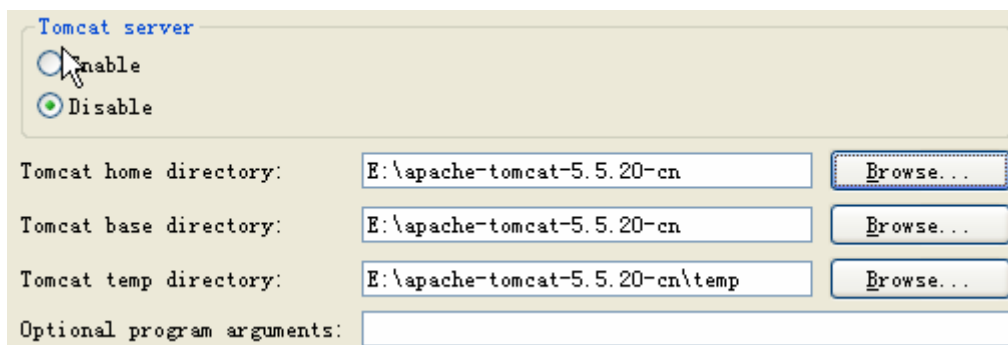


图 6.5 默认的 Tomcat 设置信息

6.4.2 第 2 步 启用连接器

要想使用这个服务器，必须启用连接器，之后你才能在 **Servers** 视图中看到它并对它进行管理。如图 6.6 所示。到这一步可以点击 **OK** 按钮就可以完成配置了。

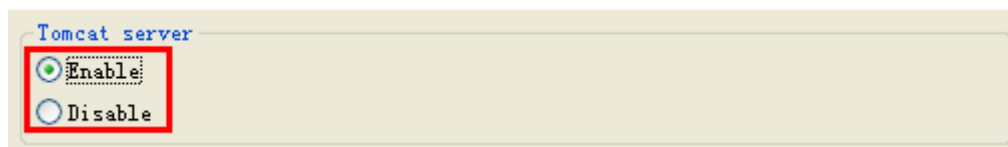


图 6.6 启用 Tomcat 连接器

6.4.3 第 3 步 选择启动服务器时候所用的 JDK

最后一步可选的操作是为服务器指定合适版本的 JDK，对于 Tomcat 5，Tomcat 6 和

JBoss 4 来说，默认的 JRE（一般是 1.5 版本或者更高，例如 MyEclipse6.0）就可以了，不需要 JDK。然而对于一些低版本的服务器，例如 Tomcat 4，则只能选择 JDK 1.4（不能用 JRE 1.4），这时候就必须指定 JDK。而 Tomcat 5 则必须是 JDK 5 或者更高版本。点击左侧的 **JDK** 节点，然后选择右侧的 **Tomcat JDK name** 下方的 JDK 列表下拉框，可以选中对应版本的 JDK。如下图所示：

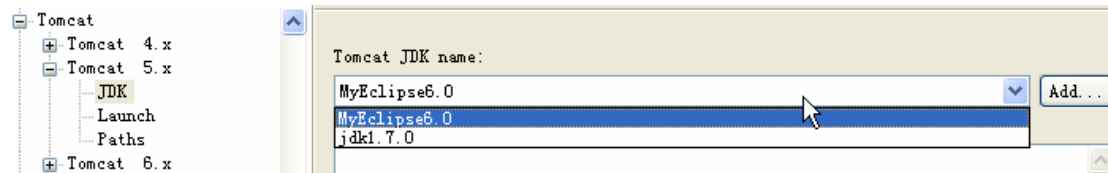


图 6.7 Tomcat JDK 设置页面

最后点击 **OK** 按钮就可以完成配置了。

6.4.3.1 可选操作：添加 JVM

如果您要使用的 JDK 版本在列表里没有出现，请点击 **Add...**按钮来启动 **Add JVM** 对话框，如图 6.8 所示：

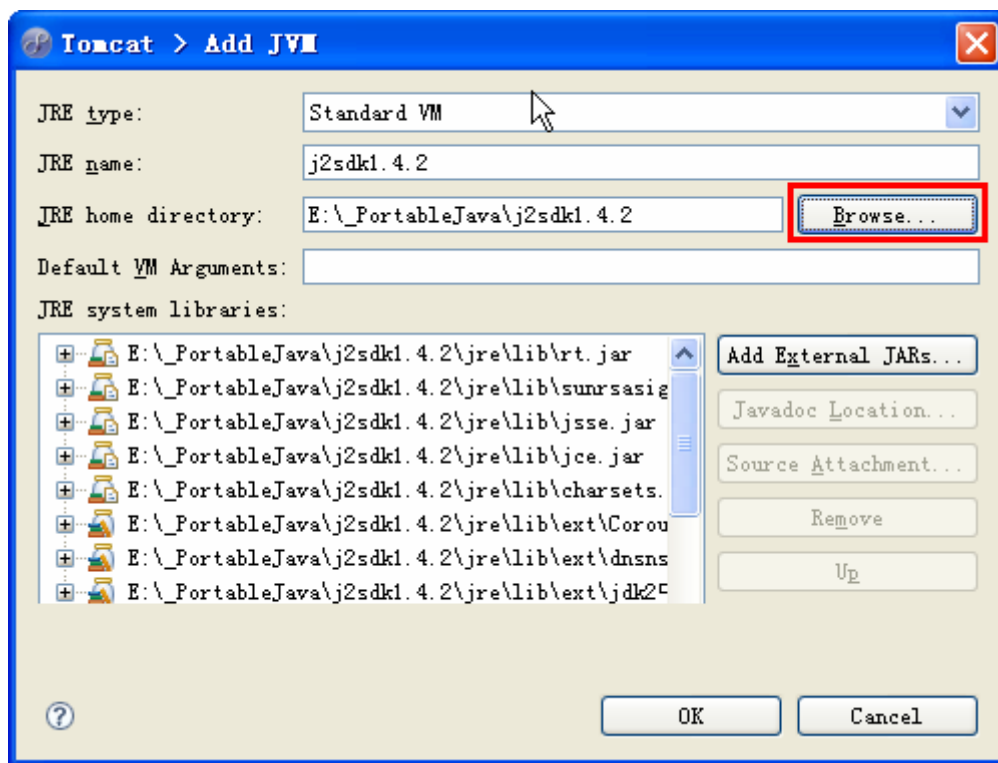


图 6.8 Add JVM 对话框

在这个对话框里面点击 **Browse...**按钮，然后在目录选择对话框中选中 **JDK** 安装目录的根目录，例如 **c:\jdk1.5**，而不是选中其中的 **JRE** 子目录，例如 **c:\jdk1.5\jre**，之后点击目录选择对话框的**确定**按钮返回 **Add JVM** 对话框，这时候 **JRE name** 输入框会自动输入一个名称，你也可以手工输入一个新名称。最后点击 **OK** 按钮就完成了 JVM 的添加工作，接下来你就可以在 **Tomcat JDK** 配置页面选中这个新加入的 JDK 了。

6.5 发布并运行 Java EE 项目

6.5.1 Java EE 项目的发布类型

MyEclipse 支持发布 Web, EJB 和 Enterprise Application 项目到任何 MyEclipse 支持的服务器上。它支持散包和打包发布。请找服务器的顾问或者文档来了解是否支持散包发布。目前来说 Tomcat 和 JBoss 都是支持散包发布的。

6.5.1.1 散包发布

散包发布一般是开发时候来使用, MyEclipse 会把所有的文件按照 Java EE 规定的目录结构放在服务器的发布目录下。在这种情况下, MyEclipse 还会自动把修改过的文件, 例如 JSP 文件, 类文件等等复制过去, 实现自动同步功能, 这时修改了 JSP 页面不需要重新发布就能在浏览器里刷新后看到新的结果。这样对开发来说是非常方便的。但是需要指出的是: 并非所有服务器都支持散包发布, 散包发布也不是 Java EE 规范强制规定的内容。

6.5.1.2 打包发布


这种模式一般是用在生产机上的, 也就是打算正式上线并把应用产品化的时候选择的。它会把所有的文件按 Java EE 规范打包成单个的 ZIP 文件(后缀可能是.EAR, .JAR, .WAR 等等), 然后放到服务器的发布目录下完成发布过程。这种模式下的缺点就是 MyEclipse 不会自动更新 ZIP 文件里面的内容, 也无法自动重新发布。

6.5.2 向服务器发布应用

在 MyEclipse 6 中, 向服务器发布应用的最快最方便的办法是选择菜单 **Run > Run As > 3 MyEclipse Server Application**, 之后 MyEclipse 可能会显示一个可用的服务器列表, 选中其中的服务器之一并点击 **OK** 按钮后, 就会自动发布或者重新发布应用然后启动服务器。如果选中的是 **MyEclipse Tomcat** 这个服务器, 甚至可以自动打开一个 **MyEclipse Web Browser** 视图, 并在这个内置浏览器中打开 Web 项目的首页面。这个方法是比较方便快捷的过程。

传统的发布方式, 步骤比较多, 可以参考下面的说明来进行。

6.5.2.1 第 1 步 打开发布对话框

点击主界面工具栏上的按钮, 就可以打开 **Project Deployments** 对话框, 如下图所示:

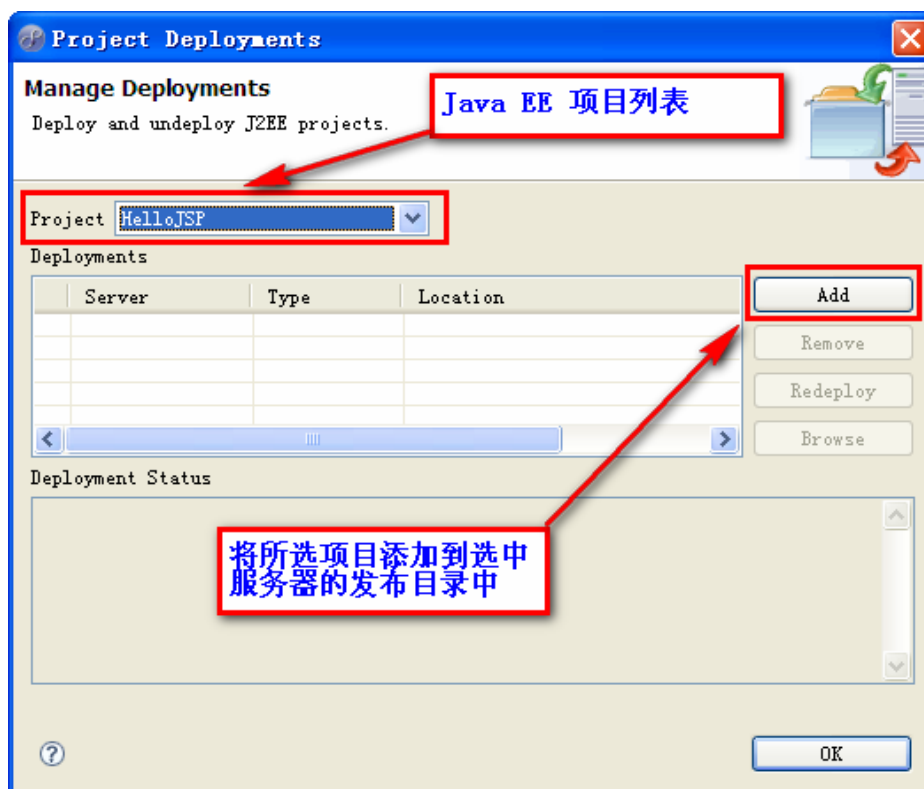



图 6.9 项目发布向导对话框

如果点击的是或者 Servers 视图工具栏的按钮，则会打开 Servers Deployments 对话框，显示的内容略有不同而已，如下图所示：

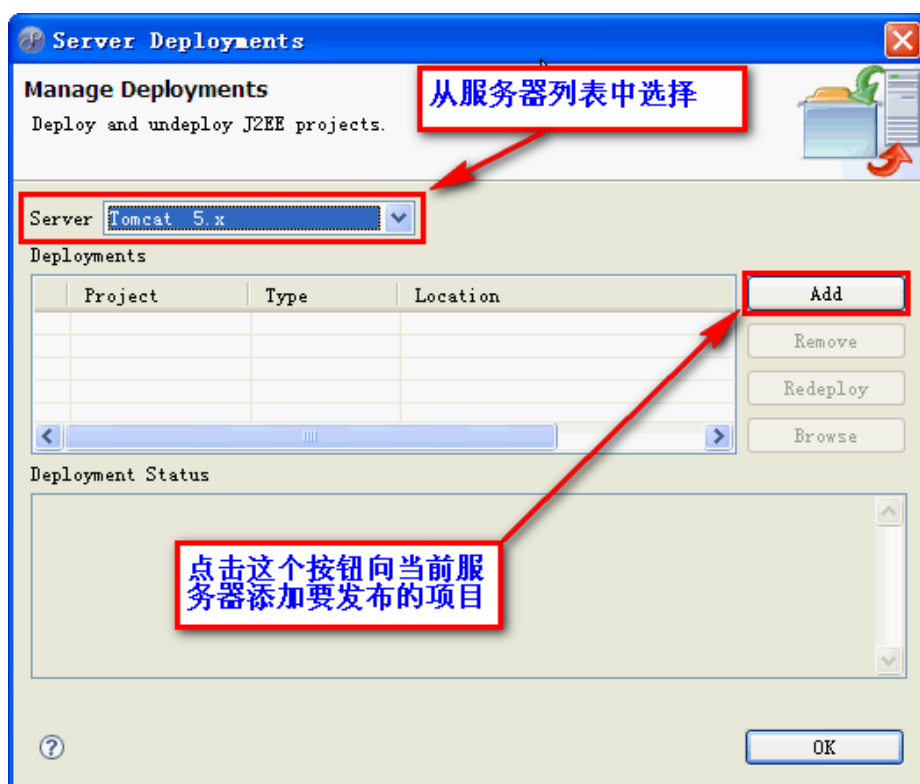


图 6.10 服务器发布向导对话框

6.5.2.2 第 2 步 点击 Add 按钮启动新建发布对话框并完成发布

当点击图 6.9 中的 **Add** 按钮后，可以启动 **New Deployment** 向导，如下图所示：

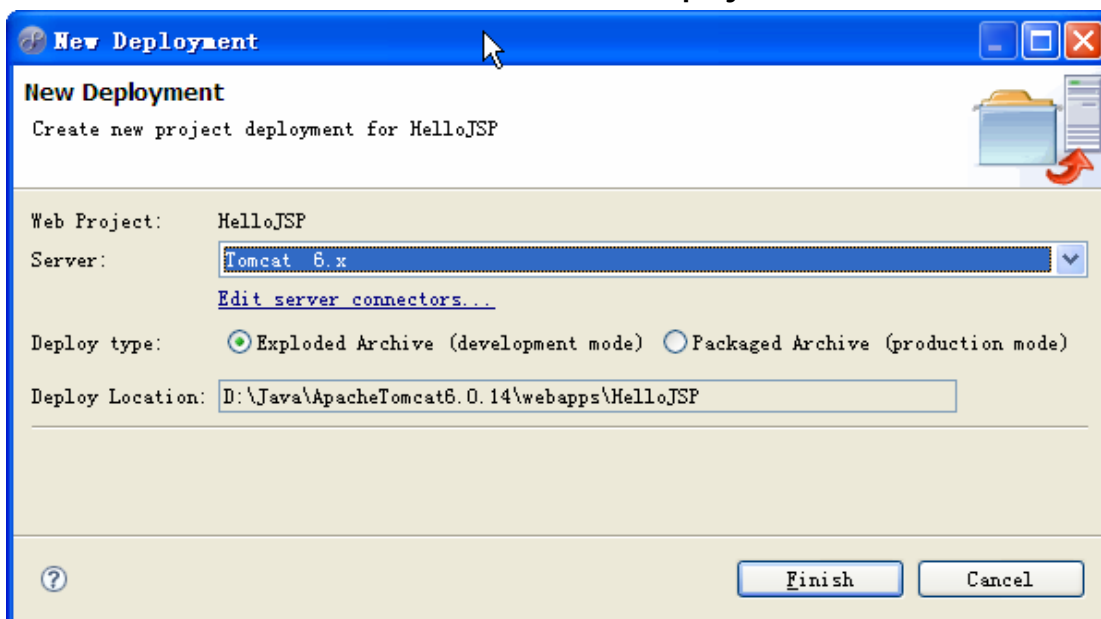


图 6.11 新建发布对话框

在这一步我们可以指定将项目发布到的服务器，以及发布的类型。点击 **Server** 右侧的下拉框选择对应的服务器定义，选择 **Deploy type** 中的两个单选钮之一来指定发布类型（左侧的为散包发布，开发模式；右侧的为打包发布，生产机模式），而 **Deploy Location** 则显示了最终项目文件被发布到的目标目录。点击 **Finish** 按钮就可以显示发布的进程并等待最终完成发布过程。发布结束后会在 **Project Deployments** 对话框里面显示此次发布的结果和状态，如下图所示：

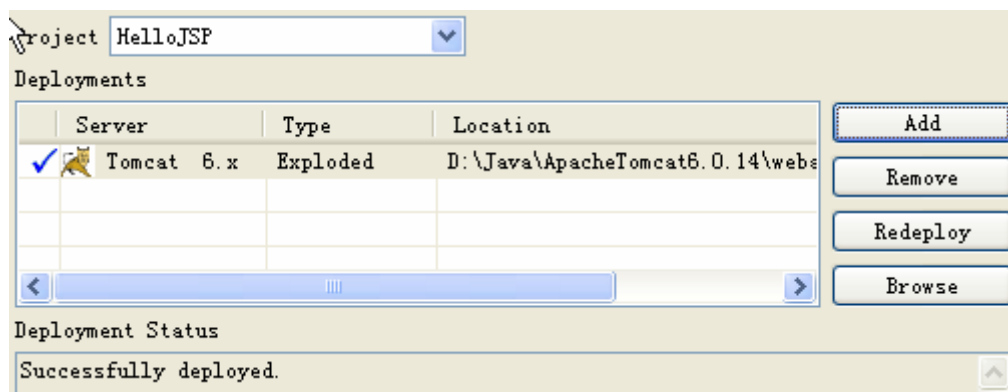




图 6.12 发布结果对话框

如果是通过图 6.10 开始的发布过程，点击 **Add** 按钮后只是对话框中选项的位置略有不同，其它概念和操作都是相似的。点击 **Remove** 按钮会删除这个发布，点击 **Redeploy** 按钮则会重新发布这个应用，点击 **Browse** 按钮则会在系统的文件浏览器中打开发布后的应用所在的目录。

6.6 应用服务器的管理和调试

6.6.1 启动服务器

要运行服务器有两种办法，一种是在 **Servers** 视图中选中服务器，之后点击视图工具栏上的  按钮以运行模式启动服务器，或者点击  按钮以调试模式启动服务器。或者点击主界面工具栏上的 **Run/Stop/Restart MyEclipse Servers** 按钮来启动服务器，如下图所示：

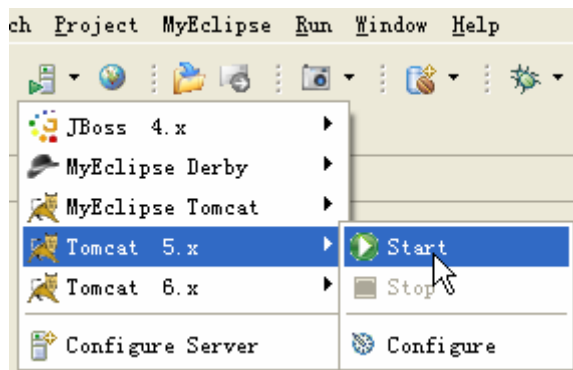


图 6.13 工具栏上的启动服务器按钮

。这样就可以启动所选择的服务器了。

6.6.2 监控服务器启动过程

服务器启动之后，输出的日志就会显示在 **Console** 视图中，便于我们浏览和跟踪查看日志来判断服务器是否正常启动完毕。例如下图显示了正常的 **Tomcat** 启动完毕后的输出日志：

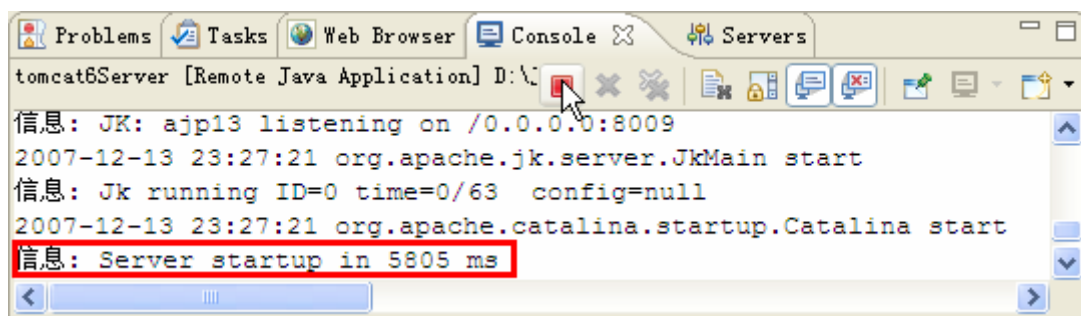





图 6.14 Tomcat 服务器启动成功的日志输出

6.6.3 停止服务器

可以点击 **Console** 视图工具栏上的  来强制终止服务器进程，或者可以通过 **Servers** 视图上的  按钮来正常停止服务器，也可以通过图 6.13 上的  Stop 菜单项来停止服务器。服务器关闭过程中的日志也会显示在 **Console** 视图中。

6.6.4 调试发布的企业应用

MyEclipse扩展了Eclipse的调试器，这样它可以在JSP中设置断点和进行调试，也可以对EJB进行调试。可以像在[断点和调试器](#)一节所介绍的那样来对JSP或者Servlet设置调试信息。**注意：**服务器一定要以调试模式启动，参考[启动服务器](#)一节的内容。其它的操作和普通的调试都是一样的，可以进行单步执行等操作。这样对开发时解决问题来说是非常的方便快捷的，当然另一种更好的办法实在代码里面用System.out.println(“调试信息”)来输出调试代码。

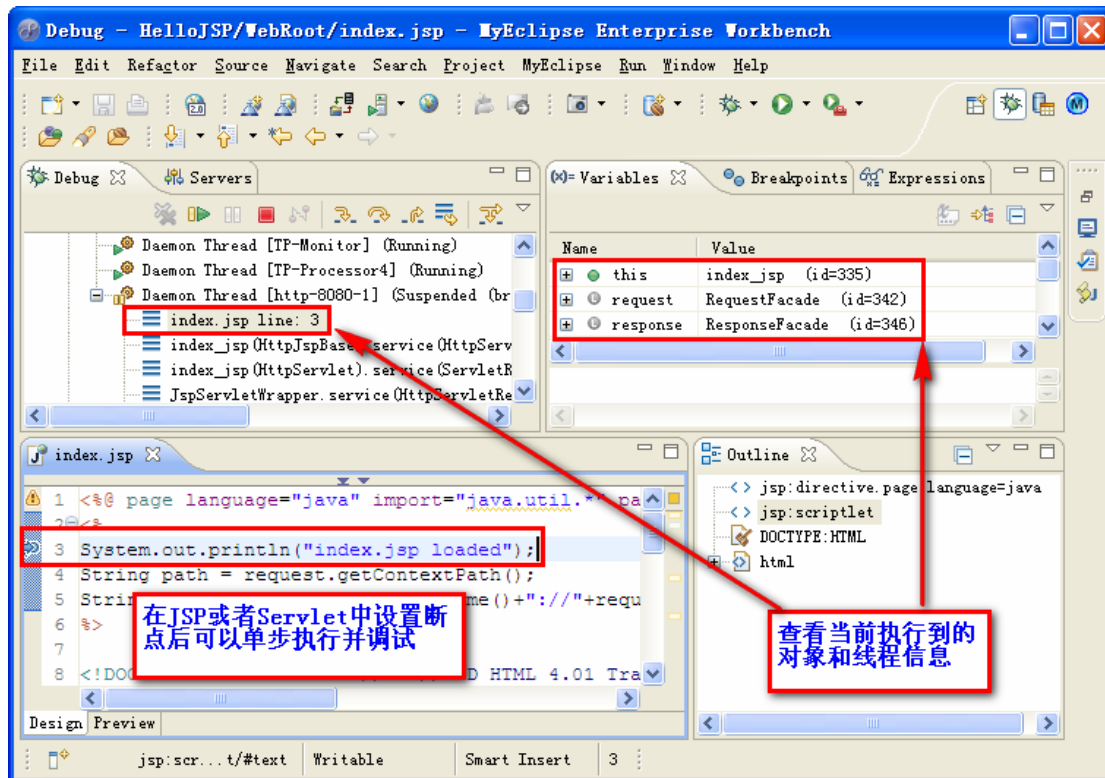


图 6.15 对 JSP 进行调试

6.7 小结

在本章中我们对如何使用 MyEclipse 进行服务器管理，调试，配置等进行了详细的讨论，本节内容将会对以后理解 Web 项目的开发有很大的帮助。

第七章 开发 Hibernate 应用

本章内容参考视频：[MyEclipse 6 实战开发讲解视频入门 3 MyEclipse Hibernate 快速入门开发](#)。

7.1 介绍

在 [第五章 开发JDBC应用](#)中，我们介绍了如何使用JDBC来开发访问数据库的Java程序。可以看到使用JDBC写程序是相对比较麻烦的（当然也许前面的例子比较简单，体会不到难度），因此本节我们介绍如何用MyEclipse进行Hibernate的开发。通过使用MyEclipse提供的Hibernate反向工程技术，可以在 2 分钟内完成所有文件的生成工作。

本章介绍了使用 **MyEclipse Enterprise Workbench** 开发 **Hibernate** 的基本功能，概念和技术。我们将全程带领你来开发一个非常简单的 **Java Hibernate** 应用。对于没有涉及到的问题和概念，推荐参考 [资源](#) 部分列出的 **Hibernate** 资源。

本章介绍了如何进行下列工作：

- 为 **Java** 项目添加 **MyEclipse Hibernate** 支持
- 在项目中创建 **Hibernate** 配置文件
- 如何使用自定义的 **Session Factory**
- 从 **Database Explorer** 的表定义中生成 **Java** 类和 **Hibernate** 数据库映射文件 (.hbm.xml)
- 使用 **HQL** 编辑器
- 创建使用 **Hibernate** 的小测试应用

7.2 Hibernate 一览

Hibernate (www.hibernate.org) 是一个非常流行的开源的易于配置和运行的基于 **Java** 的对象-关系映射(JORM) 引擎。它提供了很丰富的功能，包括单不局限于下列功能：

- 多种映射策略
- 可迁移的持久化
- 单个对象映射到多个表
- 支持集合
- 多态关联
- 可自定义的 **SQL** 查询

Hibernate 使用 **Java** 编写，是一个高度可配置的软件包，可以通过两种配置文件格式来进行配置。第一种配置文件名字为 **hibernate.cfg.xml**。在启动时，**Hibernate** 查询这个 **XML** 里面的属性来进行操作，例如数据库连接字符串和密码，数据库方言 (database dialect)，以及映射文件位置等。**Hibernate** 在类路径中查找这个文件。第二种配置文件是映射描述文件(文件扩展名为 ***.hbm.xml**)，它来指示 **Hibernate** 如何来将特定的 **Java** 类和一个或者多个数据库表格中的数据进行映射。**MyEclipse** 提供了工具来处

理这两种配置文件，并且可以将它们和你对数据库和 Hibernate 映射的 Java 类的修改进行同步。

Hibernate 可以用在任何需要将 Java 对象和数据库表格中的数据进行移动的 Java 应用中。因此，它在开发两层和三层的 J2EE 应用中很有用。向你的应用中集成 Hibernate 包括：

- 向你的项目中安装 Hibernate 核心类和依赖的 JAR 类库
- 创建 hibernate.cfg.xml 文件来描述如何访问你的数据库
- 为每个持久化 Java 类创建单独的映射描述文件

更多关于 Hibernate 的基本和高级特性，或者如何使用 Hibernate 进行开发的信息，请查看本章的 资源 部分。

7.3 准备工作

本节内容需要安装MySQL数据库，请参考 [MySQL 5 数据库服务器下载, 安装和运行\(可选\)](#) 一节内容来安装并启动MySQL数据库以及获得对应的JDBC驱动jar文件。

也可以直接用 MyEclipse Derby 数据库完成这个练习，其它种类的常见数据库也能够完成这个练习。

进行之前请参考 [建立到MySQL数据库的连接](#) 一节的内容建立好一个mysql5 的连接。当然如果是别的数据库也可以，只要建立好对应的连接就可以了。

注意：必须阅读 [用MyEclipse Database Explorer管理数据库](#) 这一章的内容，否则本节的快速开发功能将无法进行。

7.4 创建 HibernateDemo 项目

这一部分描述了创建名为 **HibernateDemo** 的简单的 Java 项目的过程，这个项目使用 Hibernate 来保存文本消息到一个单独的数据库表格中。因为多数企业的网络应用都是和企业关系数据库中的数据进行交互，我们将集中精力到编写 Java 数据对象和映射文件来操作现有的数据库。

7.4.1 创建表格

请参考第 5 章 [创建数据库表格](#) 的内容来建立项目所需要的表格（包括用于MySQL数据库和Derby数据库等等的SQL语句），这里仅仅列出来MySQL建表的SQL语句：

```
CREATE TABLE Student (
  id int NOT NULL auto_increment,
  username varchar(200) NOT NULL,
  password varchar(20) NOT NULL,
  age int,
  PRIMARY KEY (id)
) ENGINE=MyISAM DEFAULT CHARSET=GBK
```

7.4.2 创建 HibernateDemo Java Project

我们先来创建一个普通的名为 *HibernateDemo* 的 Java 项目，这个项目读取写入数据到 *Student* 数据库表。

1. 从 MyEclipse 菜单栏选择 **File > New > Project > Java Project**，接着会打开 **New Java Project** 向导。
2. 输入 *HibernateDemo* 到 **Project name**。
3. 在 **Project Layout** 下选中 **Create separate source and output folders** 单选钮。
4. 选择 **Finish** 来完成这个页面，如下图所示：

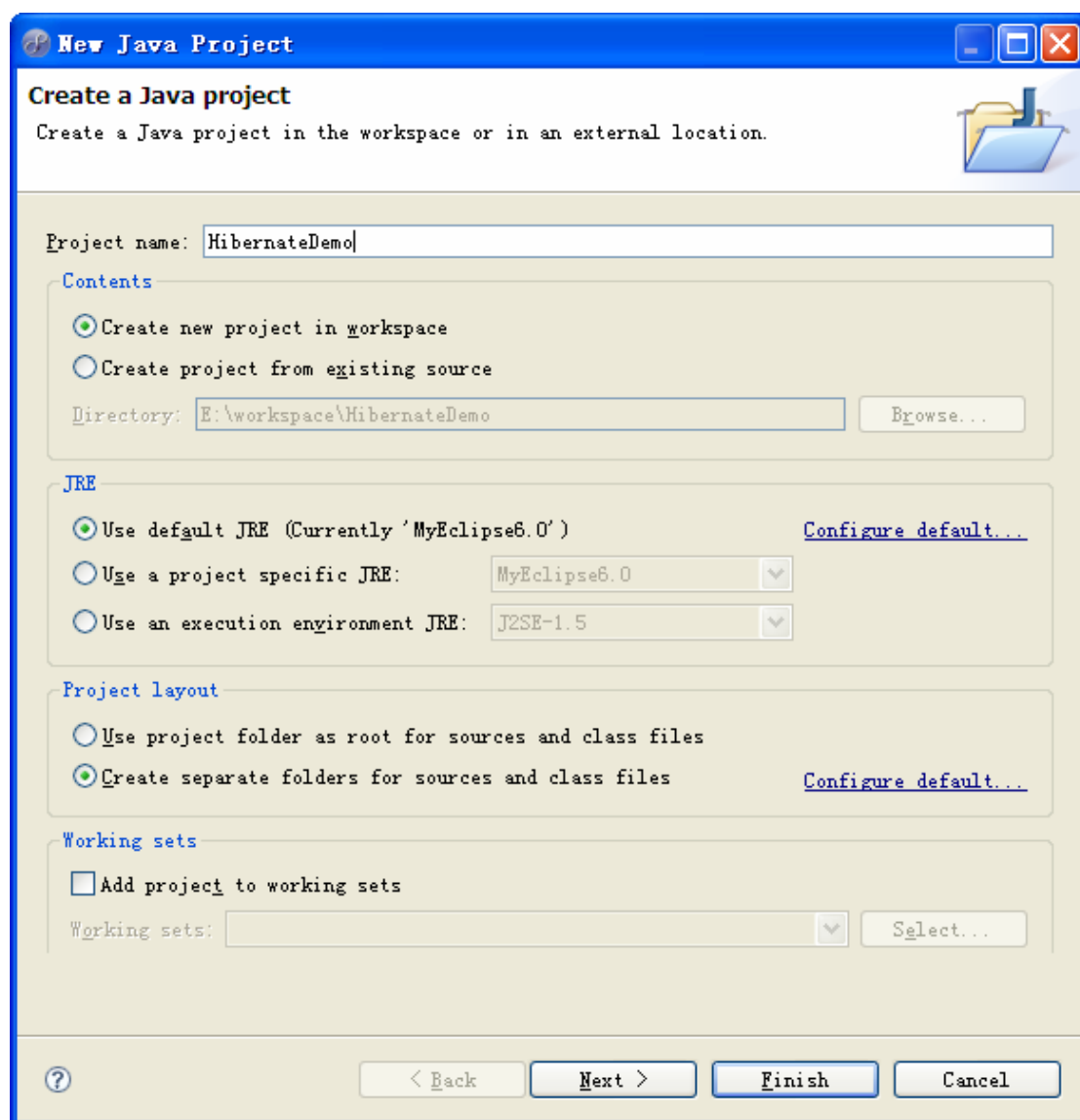


图 7.1 新建 HibernateDemo 项目

注意：也可以创建 Web Project 等，Hibernate 可以添加到几乎所有类型的 Java 项目中。

7.4.3 添加 Hibernate Capabilities 到现有项目

现在 *HibernateDemo* 项目已经创建，然后就可以添加 MyEclipse Hibernate 功能到这个项目。整个处理过程将执行如下操作：

- 添加 Hibernate 类库 (JARs) 到项目的类路径，
- 在项目中创建并配置 `hibernate.cfg.xml` ，
- 在项目中创建自定义的 Session Factory 类来简化 Hibernate 会话处理。

要给项目添加 Hibernate 功能，请按照下面的步骤进行：

1. 在 **Package Explorer** 中选择 *HibernateDemo* 项目
2. 接下来，从 MyEclipse 菜单栏选择 **MyEclipse > Project Capabilities > Add Hibernate Capabilities ...** 来启动 **Add Hibernate Capabilities** 向导，如下图示：

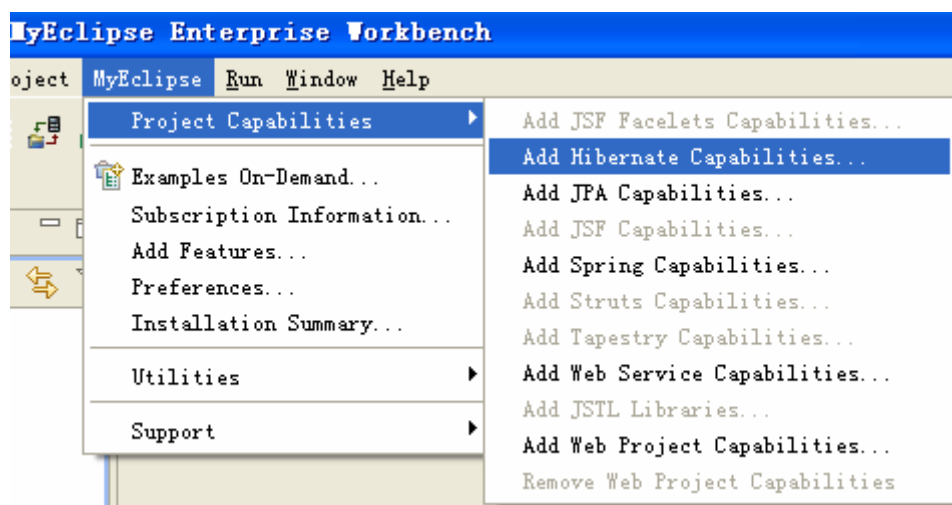


图 7.2 启动添加 Hibernate Capabilities 向导

接着将会弹出 **Add Hibernate Capabilities** 对话框，如下图所示：

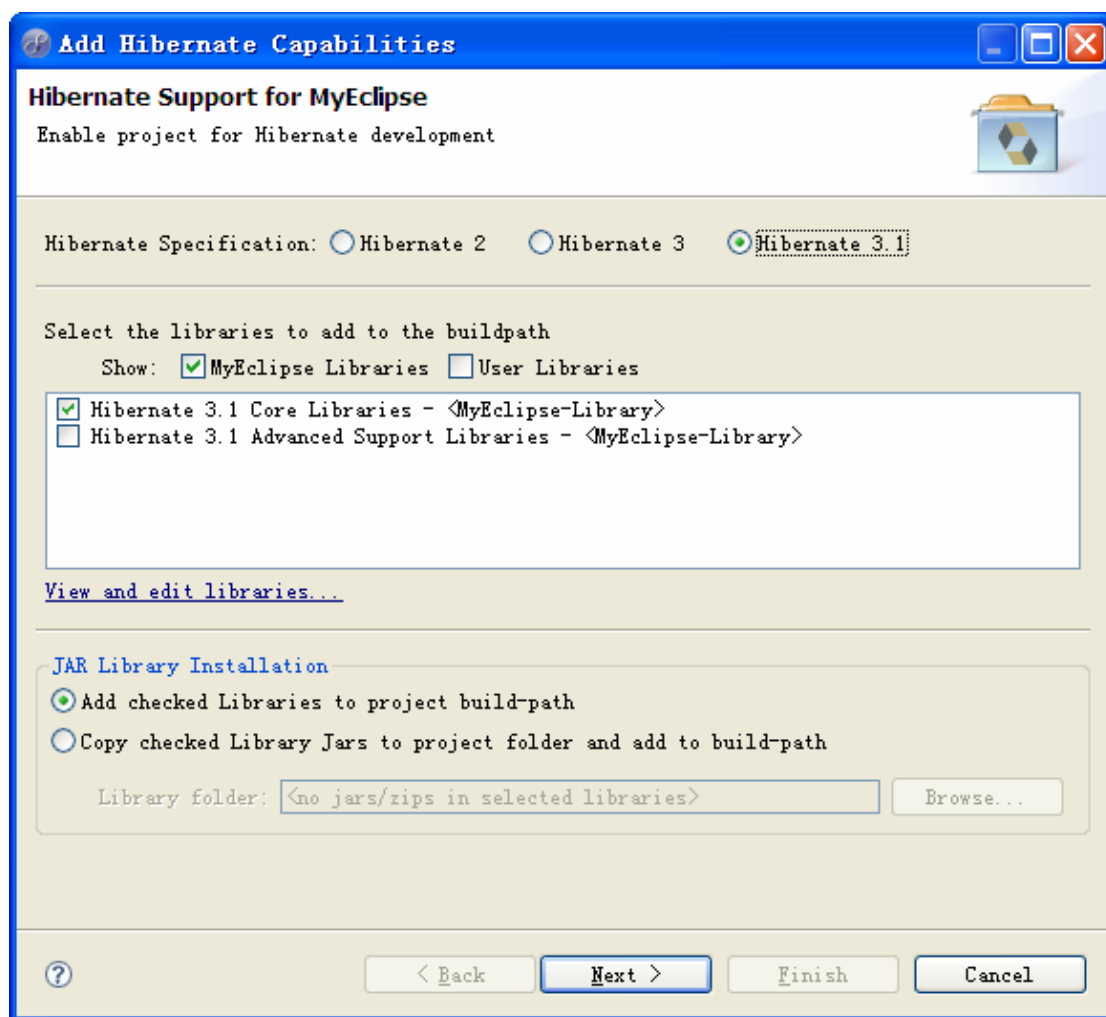


图 7.3 Add Hibernate Capabilities 向导对话框

3. 保持 **Hibernate Specification** 右侧的单选钮 **Hibernate 3.1** 选中不变，这一步是选择 Hibernate 类库的版本。
4. 选择你需要的类库集合，在这个示例中 **Core** 类库足够了。
5. 保持 **Add checked Libraries to project build-path** 选中。

在 Add Hibernate Capabilities 对话框中，有不少选项，那么各个选项的说明如下表所示：

选项	描述
Hibernate Specification	要添加到项目中的 Hibernate 版本支持功能。为了最大限度的利用 MyEclipse Hibernate 工具，推荐 Hibernate 3.1 。
MyEclipse/User Libraries	可以添加到你的项目的构造路径的类库集合
Add checked Libraries to project build-path	选中的类库将会添加到你的项目的构造路径中，但是相应的 JAR 文件将不会复制到你的项目中。这些 JAR 文件将会在发布程序时复制，这是推荐的设置方式。
Copy checked Library Jars to project folder and add to build-path	选中的类库 JAR 文件将会被复制到你的项目并添加到构造路径中去（这个方式在开发不依赖于 MyEclipse 的项目的时候，或者解决 JAR 包冲突的时候很有用）
Library Folder	仅在上一行的选项选中时可用 相对于现在项目的路径，可以新建或者使用现有目录， Hibernate 类库将会被向导复制到这里。

表 7.1 Add Hibernate Capabilities 向导 - 第 1 页选项描述

6. 选择 **Next** 按钮前进到下一页，这一页将显示 **Create Hibernate XML configuration file** 这个向导，也就是创建 Hibernate XML 配置文件。如下图所示：

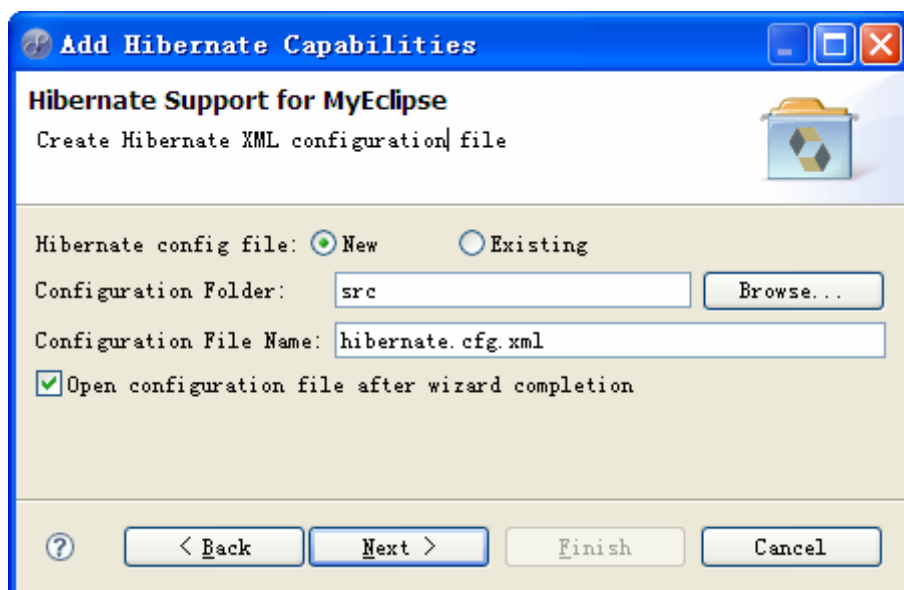


图 7.4 创建 Hibernate XML 配置文件

如果是新项目保持默认设置然后点击 **Next** 按钮就可以了。反过来如果以前有存在的 Hibernate 配置文件的话，可以点击选中 **Existing** 单选钮后选择现有 Hibernate 配置文件的路径即可，然后点击 **Next** 按钮。

7. 接下来会显示选择 Hibernate 所使用的数据库连接的对话框，如下图所示：

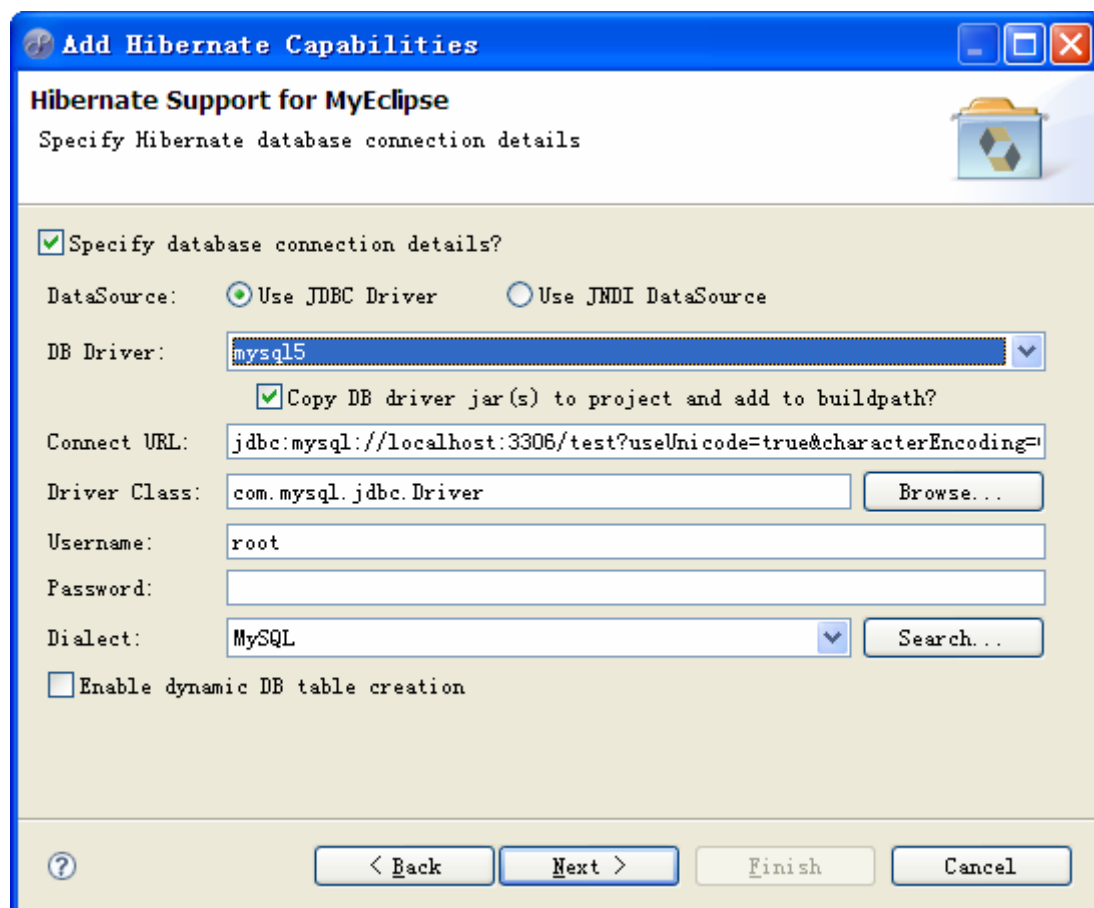


图 7.5 选择数据库连接的对话框

点击 **DB Driver** 右侧的现有数据库连接列表，选择以前创建好的数据库连接例如 *mysql5*，这时候相关的连接信息将会自动填入到对话框中，对应的 **Hibernate** 方言也会选择好（**Dialect** 右侧的下拉列表可以选择一种 **Hibernate** 所支持的方言）。复选框 **Copy DB driver jar(s) to project and add to buildpath?** 选中后则会自动加入相应的数据库驱动类库 jar 文件到项目的类路径中。之后点击 **Next** 按钮即可。

注意：如果你不想现在就设置数据库连接属性，去掉 **Specify database connection details?** 前面的复选框即可跳过。

8. 这时候来到向导的最后一页，创建一个 **SessionFactory**。在这一页点击 **Java package** 输入框右侧最右侧的 **New...** 按钮来创建一个包，我们这里输入 *dao*，然后点击 **Finish** 按钮即可。

注意：这一页的设置也是可选的，如果你不想现在就设置数据库连接属性，去掉 **Create SessionFactory class?** 前面的复选框即可跳过设置并直接点击 **Finish** 按钮完成向导。

界面截屏如下所示：

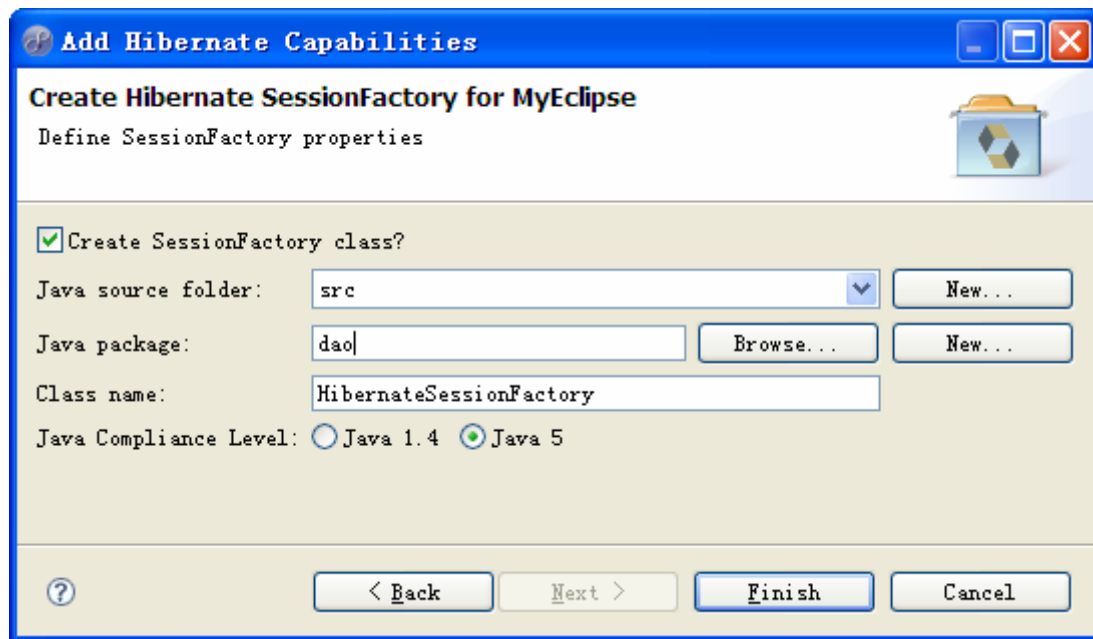


图 7.6 创建 SessionFactory 对话框

这个向导结束后将进行下列操作：

- 如果在第一页选择了复制类库到你的项目，将复制 Hibernate 类库 (JARs) 到项目中
- 更新项目的构造路径来包含已安装的 Hibernate 类库
- 给项目创建并配置 hibernate.cfg.xml 文件
- 在项目中创建一个自定义的 SessionFactory 类 (例如 HibernateSessionFactory) 来简化 Hibernate 的会话处理。

7.4.4 使用 Hibernate 配置文件编辑器修改文件

最后，MyEclipse 会自动打开 hibernate.cfg.xml 编辑器，如下图所示：

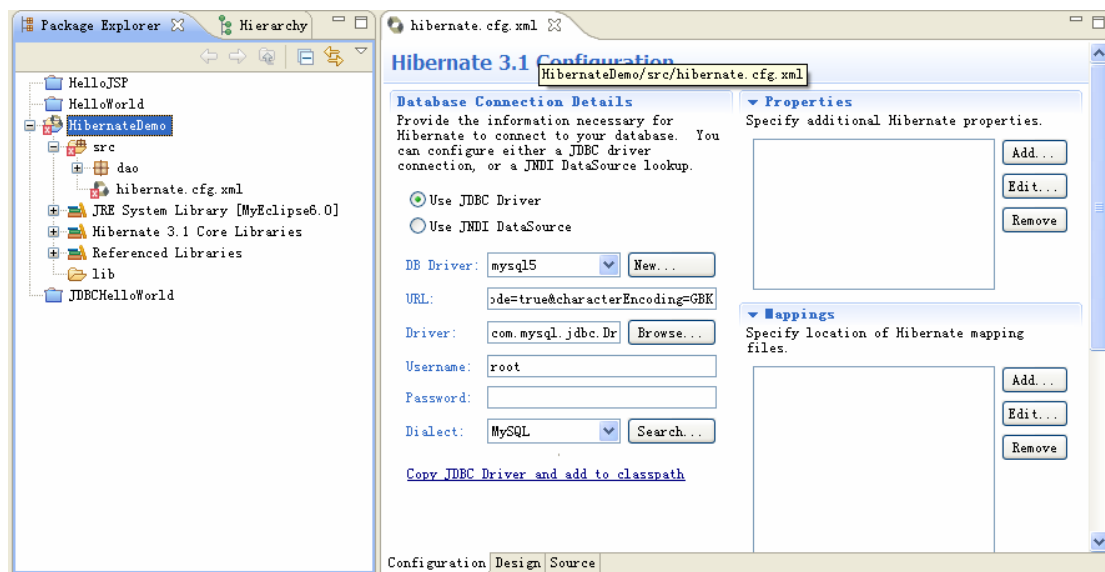


图 7.7 向导结束后的界面和 hibernate.cfg.xml 编辑器

可以注意到文件 `hibernate.cfg.xml` 前面出现了一个小红叉，表明这个文件出现了错误，这是因为用的 MySQL 数据库 JDBC 的 URL 地址里面包含了一个 `&` 字符，而这个字符在 XML 中应该用 `&` 的转义字符的方式书写。因此我们要在 URL 右侧的输入框内输入下面的内容：

`jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=GBK`

，之后点击保存按钮这个错误即可消失。

注意：并非所有数据库都会有这个问题，只要是 XML 里出现特殊字符就会出错。

Hibernate 配置文件编辑器包括三个标签页：**Configuration**、**Design** 和 **Source**。在 **Configuration** 标签中显示的是 Hibernate 的配置信息。在这个标签点击 **Properties** 组里面的 **Add...** 按钮可以添加新的属性信息，而 **Edit...** 按钮可以修改现有的属性信息，而 **Remove** 按钮则可以删除选中的属性信息；而 **Mappings** 组里面则可以对 Hibernate 的实体映射文件的位置进行修改；点击 **DB Driver** 右侧的连接列表可以重新设置要连接到哪个数据库；而 **Driver** 右侧的输入框则可以更改 JDBC 驱动程序类名，**URL** 则可以更改连接地址，**Username** 则可以更改数据库连接账户用户名，**Password** 修改密码，**Dialect** 右侧的选择框可以更改方言，点击 **Copy JDBC Driver and add to classpath** 连接则可以将驱动类加入到项目类路径。**Design** 和 **Source** 则是显示的配置文件的源码。可在 **Source** 标签下看到如下的内容，当然也可用手工对此文件进行编辑，下面是一份稍作修改过的配置文件内容（粗斜线内容）：

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!-- Generated by MyEclipse Hibernate Tools. -->
<hibernate-configuration>

    <session-factory>
        <!-- 显示后台的 SQL, 便于调试 -->
        <property name="show_sql">true</property>
        <property name="connection.username">root</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/test?useUnicode=tru
e&amp;amp;characterEncoding=GBK</property>
        <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="myeclipse.connection.profile">mysql5</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>


    </session-factory>
```

```
</hibernate-configuration>
```

7.4.5 使用反向工程快速生成 Java POJO 类，映射文件和 DAO

术语解释：DAO 是 **Data Access Object**，数据访问对象的缩写。POJO 是 **Plain and Old Java Object**，普通和旧式的 Java 对象的缩写，也就是普通 Java 类的意思。进行 Java 开发时经常会看到各式各样的缩写词。

为了让大家看这章内容的时候不至于太累，我们已经暂时跳过了很多内容。接下来就进入最激动人心的部分：不用写代码就能得到和数据库相对应的 Java 类，Hibernate 映射文件，甚至还有 DAO 类。

好了，首先打开**MyEclipse Database Explorer**透视图。切换透视图有两种办法，如何切换请参考[透视图（Perspective）切换器](#)。一种比较快办法是如那一节介绍的，点击工具栏上的点击按钮可以显示多个透视图供切换，如图 3.3 所示，然后单击其中的**MyEclipse Database Explorer** 即可切换到此透视图；另一种办法是选择菜单 **Window > Open Perspective > Other > MyEclipse Database Explorer**来显示打开透视图对话框，然后点击**OK**按钮。

接着选中 **DB Browser** 视图中在上一节所创建的 **Hibernate** 配置文件使用的那个数据库连接，点击并展开数据库里面的树状表结构，直到看到你希望处理的数据库表为止，单击选中表。**注意：**你可以选中或者一个多个要处理的表。在这个例子中要找的表是 **Student**。接着点击右键在上下文菜单中选择 **Hibernate Reverse Engineering...**，这将启动 **Hibernate Reverse Engineering** 向导。如下图所示：

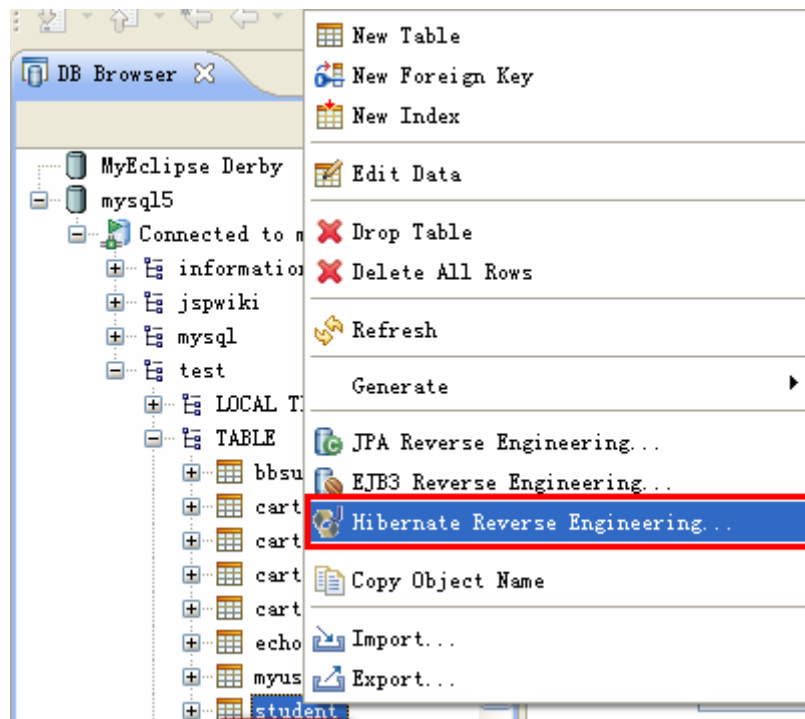


图 7.8 在 DB Browser 视图中启动反向工程向导之后启动的 **Hibernate Reverse Engineering** 向导则如下图所示：

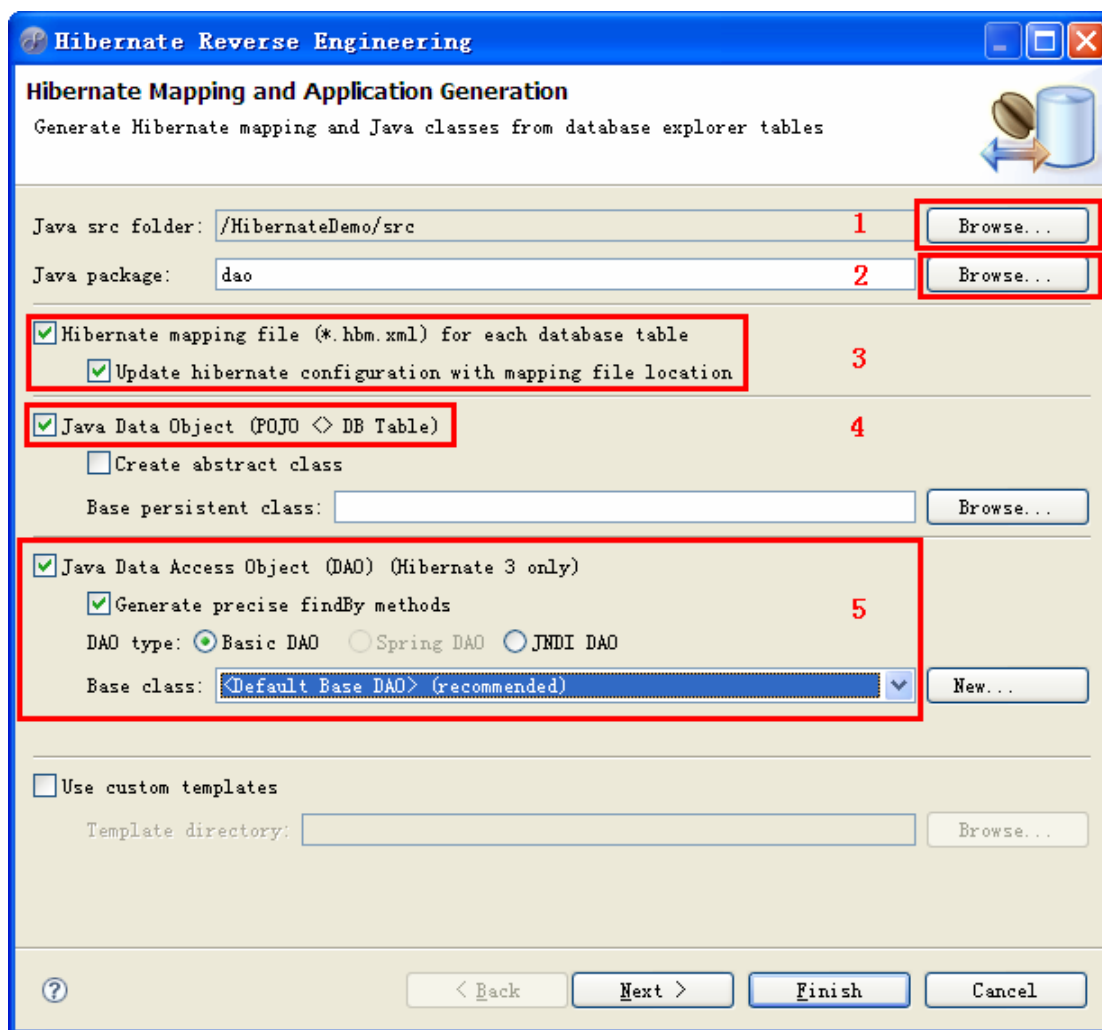


图 7.9 Hibernate Reverse Engineering 向导 第 1 页

点击 1 处的 **Java src folder** 最右侧的 **Browse** 按钮，查看可用的 **Hibernate** 项目以及源码目录，这些目录将用来存放最终生成的文件。这里选中 **HibernateDemo** 项目中的 **src** 文件夹。

点击 2 处的 **Java package** 输入框右侧的 **Browse** 按钮，选中 **dao** 包，或者新建一个其它的包来存放生成的代码所在的包。

将 3 中的两个复选框选中，这样将为每个数据库表生成 **Hibernate** 映射文件 (*.hbm.xml)，并在 **hibernate.cfg.xml** 中将新生成的映射文件加入。

在 4 中选中复选框 **Java Data Object (POJO <> DB Table)**，这样为映射文件和表格生成对应的数据对象 (POJO)。

按照图示选中 5 处的复选框，这样将能生成普通的 DAO 类。

注意：只有使用 **Hibernate 3** 才能生成便于访问映射后的类的数据访问对象。

注意：**Spring DAO** 这个单选按钮是灰色，如果可用则能生成 **Spring+Hibernate** 的 DAO (**HibernateTemplate**)，如何生成这样的 DAO 在以后会做介绍。

最后，为了简化，直接点击 **Finish** 按钮就可以结束代码的生成。只需轻轻点击几下鼠标，就生成了 **Hibernate** 实体类，映射文件，是不是太方便了啊？下面是切换到 **MyEclipse Java Enterprise** 透视图后得到的结果：

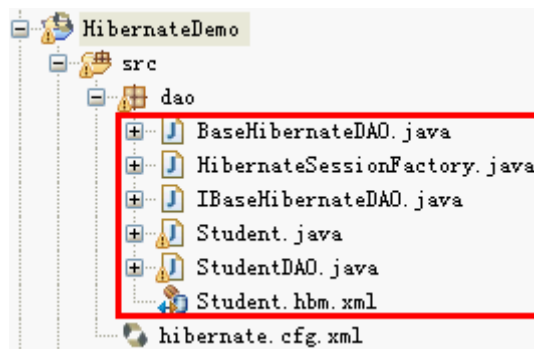


图 7.10 向导自动生成的 Hibernate 类和映射文件

那么我们需要关心的代码包括 StudentDAO, Student.hbm.xml, 和 Student。

7.4.6 调整生成的 hbm 文件

如果我们打开 Student.hbm.xml, 就可以启动 MyEclipse 的 HBM 文件编辑器, 点击 Source 标签后可以看到文件的源代码, 如下面清单所示:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse Persistence Tools
-->
<hibernate-mapping>
  <class name="dao.Student" table="student" catalog="test">
    <id name="id" type="java.lang.Integer">
      <column name="id" />
      <generator class="assigned" />
    </id>
    <property name="username" type="java.lang.String">
      <column name="username" length="200" not-null="true" />
    </property>
    <property name="password" type="java.lang.String">
      <column name="password" length="20" not-null="true" />
    </property>
    <property name="age" type="java.lang.Integer">
      <column name="age" />
    </property>
  </class>
</hibernate-mapping>
```

那么这段代码需要改动一个地方, 就是主键生成器, 默认的是赋值 (assigned), 现在打算用开发时候比较方便进行测试的 increment (自增), 将清单中的加框粗斜体的内容改成如下所示:

<generator class="increment" />

这样所有的 Hibernate 代码都算正式完成了。

7.4.7 编写测试代码

最后,新建一个普通的Java类(通过菜单**File > New > Class**来新建一个类,参考 [2.3 使用Eclipse/MyEclipse来编写,编译并运行Java程序](#))来测试上面生成的代码,代码清单如下所示:

```
import org.hibernate.Transaction;
import dao.*;

/**
 * Hibernate DAO 的测试类.
 * @author BeanSoft
 */
public class HibernateDAOTest {


    public static void main(String[] args) {

        // 实例化 DAO
        StudentDAO dao = new StudentDAO();
        // 打开 transaction
        Transaction tran = dao.getSession().beginTransaction();
        // 生成普通 Java 类
        Student bean = new Student();
        // 设置属性
        bean.setUsername("张三");
        bean.setPassword("1234");
        bean.setAge(100);
        // 插入数据
        dao.save(bean);
        // 提交事务
        tran.commit();

        // 读取数据
        java.util.List<Student> results = dao.findAll();

        // 列出列表中的所有数据
        for(Student o : results) {
            System.out.println("编号:" + o.getId());
            System.out.println("姓名:" + o.getUsername());
        }

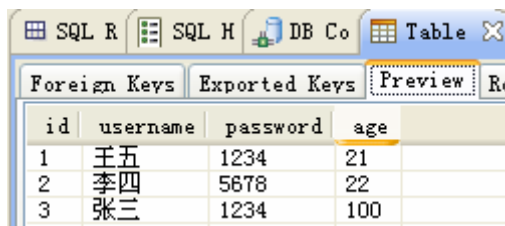
        dao.getSession().close();
    }
}
```

然后选择菜单 **Run > Run** 或者点击工具栏上的  来运行这个类,将会在 **Console** 视图看

到如下输出：

```
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.Environment).
log4j:WARN Please initialize the log4j system properly.
Hibernate: select max(id) from student
Hibernate: insert into test.student (username, password, age, id) values (?, ?, ?, ?)
Hibernate: select student0_.id as id0_, student0_.username as username0_,
student0_.password as password0_, student0_.age as age0_ from test.student student0_
编号:1
姓名:王五
编号:2
姓名:李四
编号:3
姓名:张三
```

。之后还可以在 **MyEclipse Database Explorer** 透视图的 **Table/Object Info** 视图中选中 **Preview** 标签看到新插入的数据，如下图所示：



id	username	password	age
1	王五	1234	21
2	李四	5678	22
3	张三	1234	100

图 7.11 检查数据库中的记录

7.5 MyEclipse Hibernate 工具的高级部分

在上一节中已经尽可能简单的给大家介绍了如何快速完成 **Hibernate** 应用的开发。然而在实际情况中经常会遇到复杂的表关系，例如一对多等等，**MyEclipse** 也支持这种情况的代码生成。因此本节将会介绍上面遗漏掉的内容，包括定制映射关系和使用 **HSQL** 编辑器。

7.5.1 反向工程向导的完整说明

首先介绍的是图 7.9 **Hibernate Reverse Engineering** 向导 第 1 页各个输入项的说明：

选项	描述
Java src folder	选中映射文件, POJO 和 DAO 生成后所在的项目和源码文件夹.
Java package	映射文件, POJO 和 DAO 生成后所在的包.
Hibernate mapping file	从选中的表格生成映射文件.
Update hibernate configuration	将生成后的映射文件添加到 Hibernate 配置文件中.
Java Data Object	为映射文件和表格生成对应的数据对象 (POJO).
Create abstract class	为每个数据对象生成一个抽象的父类. 这个抽象类将在以后的重新生成过程中覆盖掉, 但是对应的子类将不会被覆盖掉.
Base persistence	如果需要的话, 输入生成的 POJO 所要集成的父类的完整名称.

class	
Java Data Access Object	生成便于访问映射后的类和表格的数据访问对象. 用户可以在 Basic , Spring 和 JNDI DAO 中选择一种.
Generate precise findBy methods	为映射类中的每个属性生成一个 "findBy" 方法. 例如 <code>findByFirstName("name");</code>
Use custom templates	覆盖 MyEclipse 的内部 velocity 模版为你自己的版本. 参考 使用模版来更好的调节生成的代码 来获取更多信息.
Template directory	包含了自定义模版的目录树的根节点.

表 7-2 Hibernate Reverse Engineering 向导 第 1 页 参数说明

在前文的介绍过程中直接点击 **Finish** 按钮就完成了向导, 实际上点击 **Next** 按钮还可以进到第 2 页的设置, 如下图所示:

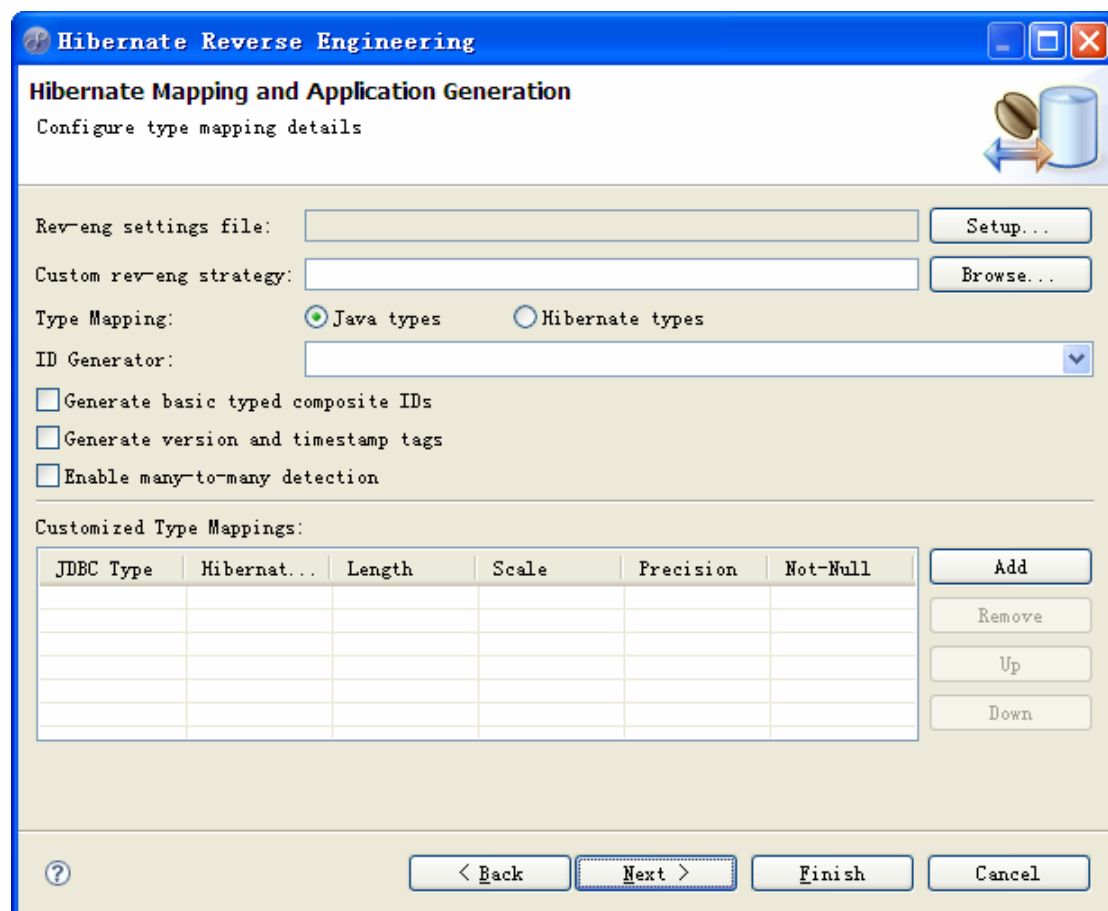


图 7.12 Hibernate Reverse Engineering 向导 第 2 页

这一页主要用来配置类型映射的细节内容, 包括映射类, 主键生成器, 多对多探测等等, 一般来说保持默认就可以了, 详细说明如下表所示:

选项	描述
Rev-eng settings file	这个文件包含了反向工程的配置和选项以供以后使用. 点击 Setup... 按钮来选择现有的文件或者创建一个新的文件. 如果找不到一个这样的配置文件的话向导将会自动创建此文件.
Custom rev-eng strategy	允许你指定一个自定义的反向工程策略类. 这个类允许你用编程的方式来定义反向工程处理过程的各个方面. 参考 使用自定义反向工程策略 来获取

	详细信息.
Type Mapping	决定是否在类型映射属性中使用 Java 或者 Hibernate 类型, 例如 <code>java.lang.String</code> 对应 <code>string</code> . 这个设置只能在向导第 3 页的 Customized Type Mappings 列表中没有指定更多信息时才能使用.
ID Generator	ID Generator 是 Hibernate 映射文件必须有的内容. 它定义了持久类实例的唯一主键生成器 <code>Jaav</code> 类. 参考 资源 部分里面的 Hibernate 文档链接, 里面描述了每个 ID 生成器的详细信息. 如果留空或者更详细的配置在这个向导的第 3 页没有配置, Hibernate 映射引擎将自动为你选择一个 ID 生成器.
Generate basic typed composite IDs	如果数据库表格包含有多个列的主键, 将总是使用 <复合主键> 映射. 如果这个选项启用并且有对应的多个外键, 每个主键列将依然会被作为'简单的'标量 (<code>string</code> , <code>long</code> , 等), 而不是引用到一个实体. 将会创建 <many-to-one> 元素, 但是它们将会标记为非可更新和非可插入的字段. 如果你禁用这个选项(默认推荐用这种方式), 将会创建 <key-many-to-one> 元素来代替上面的生成内容.
Generate version and timestamp tags	如果启用, 名为 "version" 和 "timestamp" 的列将会在生成的映射文件中作为 <version> 和 <timestamp> 标记出现.
Customized Type Mappings	允许你来指定一个自定义的 JDBC 类型到 Hibernate 类型的转换, 使用 Length, Scale, Precision 和 Nullability 作为精度控制对应原来的 JDBC 类型.

表 7-3 Hibernate Reverse Engineering 向导 第 2 页 参数说明

。在这一页进行设置后点击 **Next** 按钮可以进到最后一页, 设置反向工程的详细信息。如图 7.13 所示。

图中红色的 **1** 所框中的内容列出了当前需要反向工程的表。

2 框中的两个复选框则指示是否包含引用到这个表和这个表引用的其它表, 如果数据库支持外键的话, 相关的表格会自动出现在 **1** 框中。遗憾的是, MySQL 还不支持外键, 图中使用的是支持外键的 MyEclipse Derby, 选中两个复选框后, 相关联的表也就自动出现在 **1** 框中了, 这样就可以生成一对多的映射代码了。

3 框中的则是生成关联到当前表格的关联表的尚未反向工程过的代码。

4 框中则显示了自定义反向工程过程的一些细节内容。

关于这一页内容的配置信息可以参考表 7-4。

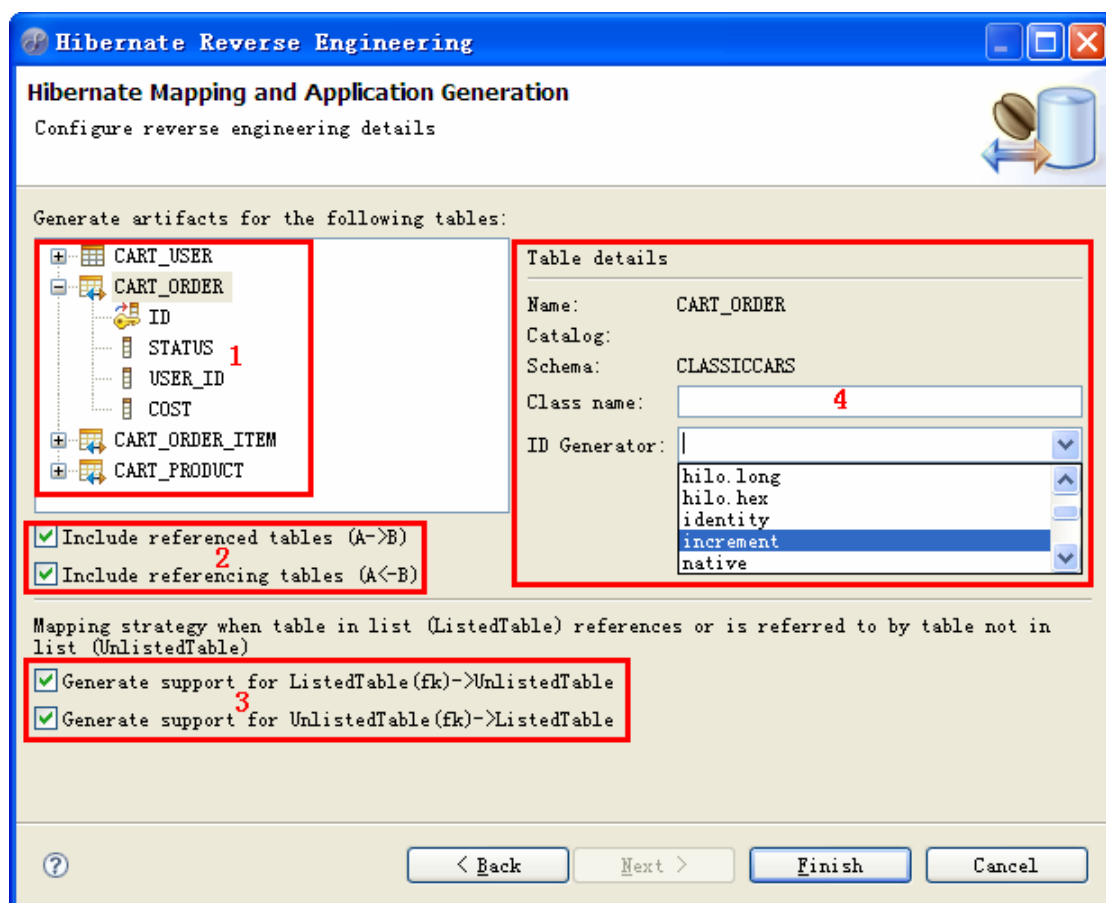


图 7.13 Hibernate Reverse Engineering 向导 第 3 页

选项	描述
Class name	对应当前数据库表格的数据对象类的完整名称。
ID Generator	想要对当前表所使用的 ID 生成器。
JDBC type	对当前列所使用的 JDBC 类型覆盖。
Property name	对应当前列所生成的属性名。
Hibernate type	对应当前列的 Hibernate 类型。
Include referenced / referencing tables	包含反向工程时当前数据库表引用的表格以及其它引用到当前表的数据库表。
Generate support for ListedTable(fk)->UnlistedTable and UnlistedTable(fk)->ListedTable	生成关联到当前表格的关联表的尚未反向工程过的代码，这些表在当前配置页面尚未被显示。

表 7-4 Hibernate Reverse Engineering 向导 第 3 页 参数说明

有了本节介绍的内容，你就可以进行一对多和多对多的映射文件的生成了。

注意：一定要数据库支持外键才可以进行这样的映射文件的生成。

7.5.2 使用 HQL 编辑器

MyEclipse 包含了一个 Hibernate 查询语言编辑器以及几个视图，允许你根据当前的 Hibernate 配置来执行 HQL 查询语句。

功能包括：

- 内容自动完成提示.
- **Hibernate Dynamic Query Translator** 在敲入 HSQL 查询语句时查看翻译后的 SQL 语句.
- **Hibernate Query Results** 视图可以查看多个查询结果集; 结果的属性显示在 **Properties** 视图.
- **Query Parameters** 视图可以很方便的执行带有参数的查询.
- 项目选择器允许你随时切换不同的 **Hibernate** 项目中的 **Hibernate** 配置

好了, 首先需要打开**MyEclipse Hibernate**透视图。切换透视图有两种办法, 如何切换请参考 [透视图 \(Perspective\) 切换器](#)。

接下来右键点击 **Package Explorer** 中的 **HibernateDemo** 项目, 在上下文菜单中选中 **MyEclipse > Open HQL Editor...**, 即可打开 HSQL 编辑器。如果之前尚未切换到 **MyEclipse Hibernate** 透视图, 那么它会提示切换透视图。

注意: HQL 编辑器也可在双击后缀为 hql 的文件时打开。

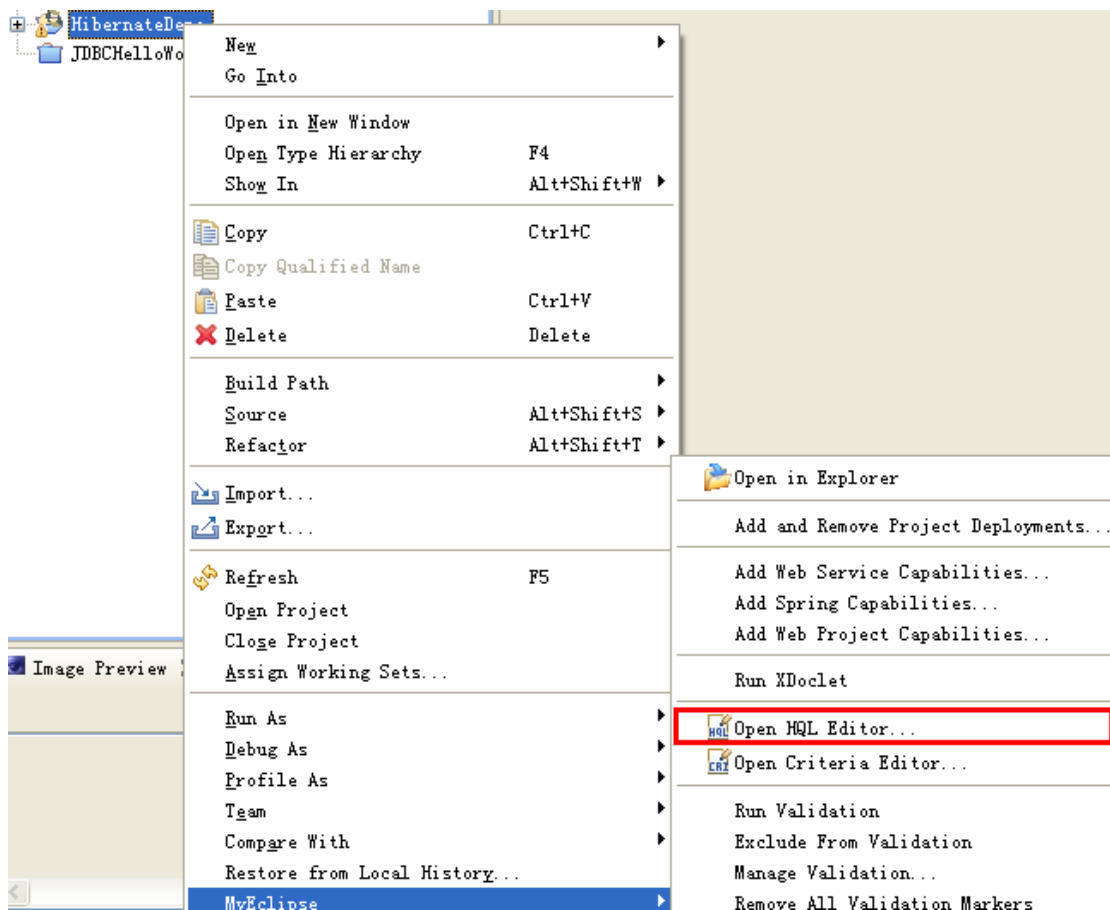


图 7.14 打开 HQL 编辑器

打开编辑器后, 即可输入 HQL 语句进行查询, 如下图所示:

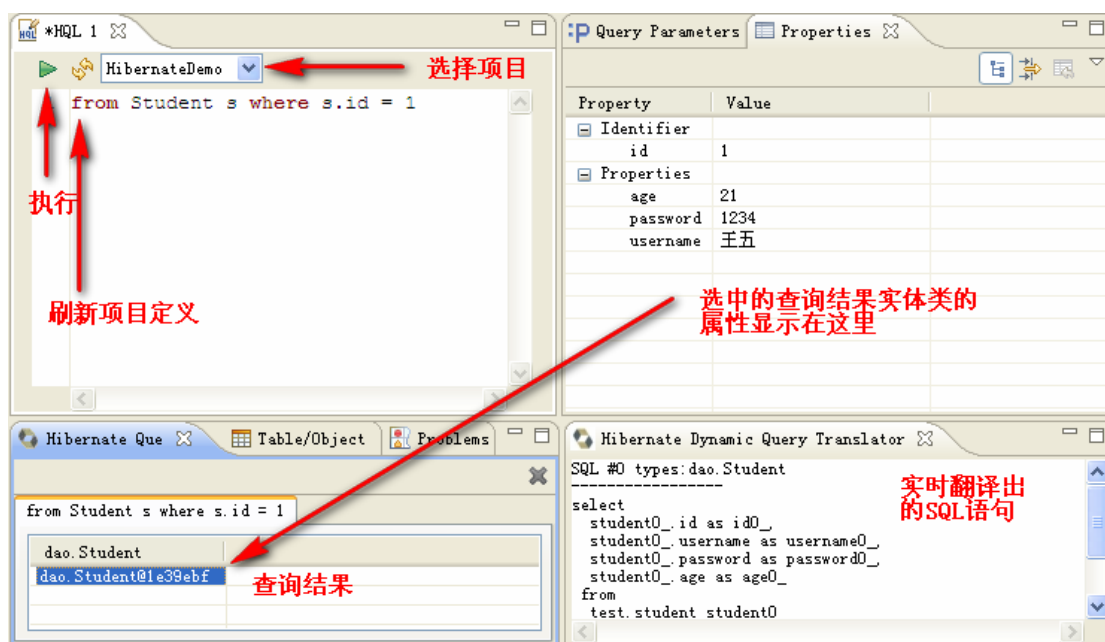


图 7.15 HQL 编辑器界面

要执行 HQL 语句，可以点击工具栏上的执行按钮。另外编辑器还能提供代码自动完成功能，如下图所示：

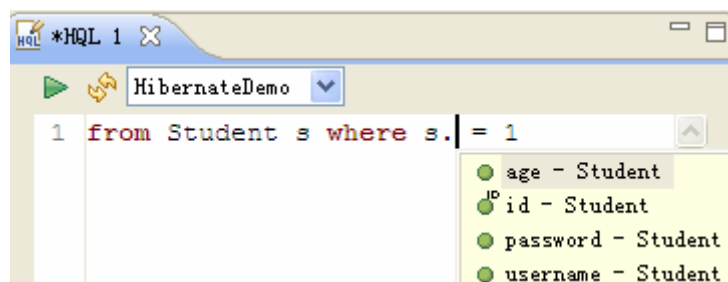


图 7.16 HQL 代码自动提示

注意：如果你在 HSQL 编辑器所选中的项目初始化后修改了配置，映射文件或者数据类，一定要记得点击嵌入的工具栏上的 **Refresh** 按钮来让编辑器使用最新的配置信息。

另外 **Query Parameters** 视图可以执行带参数的 HQL 语句，如下图所示：

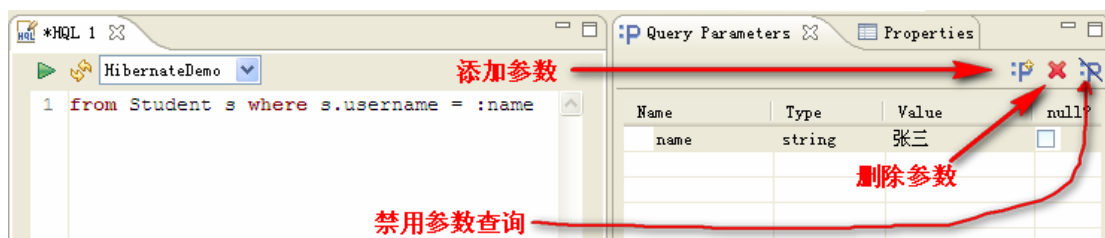


图 7.17 带参数的 HQL 查询

这样可以看到 HQL 编辑器的功能的的确是非常强大的，利用它可以方便的学习，开发和调试 HQL。

7.6 小结

本章简要讨论了 MyEclipse 中如何开发 Hibernate 应用，并从一个例子开始，由简单到复杂的展示了使用 MyEclipse 开发 Hibernate 应用的过程。读者需要事先对 Hibernate 有所

了解，方能更好的进行后续的开发。开发工具不能代替人的作用的，但是可以它能大大提高开发的效率。

7.7 参考资料

夏昕 http://www.xiaxin.net/Spring_Dev_Guide.rar Hibernate开发指南 PDF 格式

Hibernate 中文参考手册	Cao Xiaogang(曹晓钢)	3.2	238 页, 1.42M (PDF) HTML HTML Single
------------------	-------------------	-----	--

第八章 开发 Web 应用

8.1 介绍

本章将介绍如何使用 MyEclipse 来开发 Web 项目（包括 HTML，JSP，Servlet，Filter 和后台 Java 类），并进行发布，运行，测试和调试。本章将通过开发一个使用 JDBC 进行登录验证的简单例子来给大家展示相关的操作过程。

那么哪些应用算是 Web 应用呢？简单说通过网络浏览器，例如 IE，Firefox 等等上网看到的绝大多数网页，都属于 Web 应用的范围，所以它的应用是非常的广的。要想做好一个 Web 应用，只掌握 Java 是远远不够的，您还得深入了解 HTML，CSS，JavaScript 甚至 AJAX，Flash，ActiveX 等技术。俗话说的好：三分外貌七分妆。用户第一印象看到的只能是看到的网页的样子和友好度，他是完全不懂所谓的.NET，PHP，JSP，ASP 还有什么 ROR 的，所以提示初学者多花些时间在 Web 层的技术上。

8.2 Web 项目和术语

8.2.1 Java EE 中的 Web 项目结构

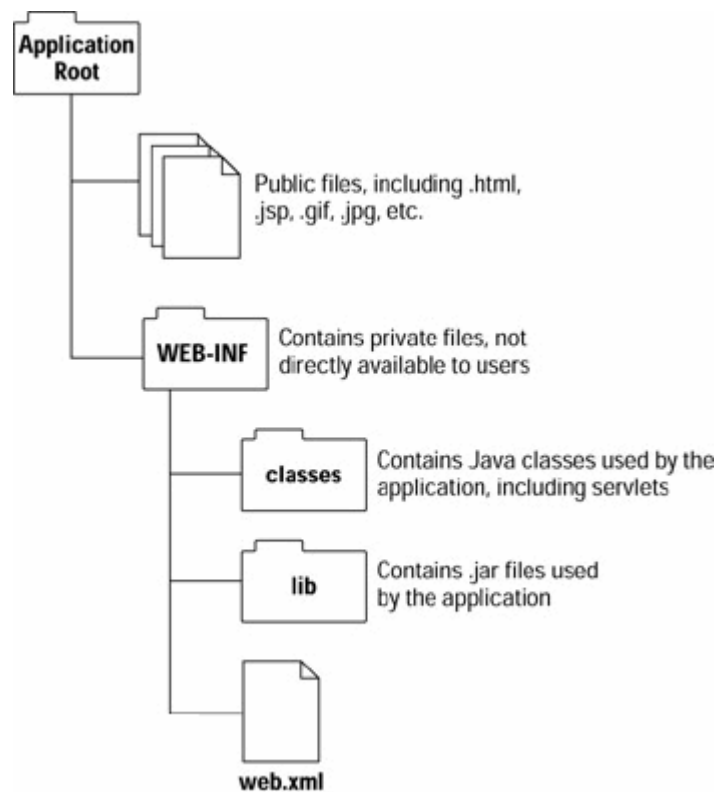


图 8.1 Web 应用结构示意图

按照 Java EE 规范的规定，一个典型的 Web 应用程序有四个部分：

1. 公开目录
2. WEB-INF/web.xml 文件，发布描述符（必选）
3. WEB-INF/classes 目录，编译后的 Java 类文件（可选）
4. WEB-INF/lib 目录，Java 类库文件 (*.jar)（可选）

公开目录存放所有可以被用户的访问的资源，包括 .html, .jsp, .gif, .jpg, .css, .js, .swf 等等。

WEB-INF 目录是一个专用区域，容器不能把此目录中的内容提供给用户。这个目录下的文件只供容器使用，里面包含不应该由客户直接下载的资源，例如：Servlet(这些组件包括应用程序逻辑以及对其他资源如数据库的可能访问)，Web 应用程序中 servlet 可直接访问的其他任何文件，在服务器方运行或者使用的资源(如 Java 类文件和供 servlet 使用的 JAR 文件)，由您的应用程序生成的临时文件，发布描述符以及其它任何配置文件。这些资源是专用的，因此只能由它们自己的 Web 应用程序及容器访问。特别地，JSP/Servlet 程序文件也能通过 `ServletContext` 访问到这个目录下的文件，例如 JSP 中可以通过 `application.getRealPath("/WEB-INF/web.xml")` 访问到发布描述符文件的路径。Web 容器要求在你的应用程序中必须由 **WEB-INF** 目录。

注意： 如果你的 Web 应用程序中没有包含这个目录，它可能将无法工作（这是因为不同的服务器对此情况的处理不甚一致，所以有时候也能工作）。

WEB-INF 中包含着发布描述符，一个 **classes** 目录和一个 **lib** 目录，以及其它内容。

发布描述符(deployment descriptors)是 J2EE Web 应用程序不可分割的一部分(也就是说是它的最小部分，必不可缺的一部分)。它们在应用程序发布之后帮助管理 Web 应用程序的配置。对于 Web 容器而言，发布描述符是一个名为 **web.xml** 的 XML 文件，存储在 Web 应用程序的 /WEB-INF 目录下。

发布描述符有多种用途：

- **为 Servlet 和 Web 应用程序提供初始化参数** 这使我们的 Web 应用程序中的硬性编写的代码的初始化值更少。例如常见的 `<param-name>`, `<param-value>` 标记，就可以为 Servlet 提供参数，这个参数可以在 `init()` 方法中加载。Struts 的 `ActionServlet` 也是通过这种方式来找到它们需要的配置文件 `struts-config.xml` 的位置，从而加载并分析它，来初始化 Struts 框架用到的各种 `FromBean`, `Action`, `Forward` 等。
- **Servlet/JSP 定义** 可以为 Web 应用程序中的每个 Servlet 或者预编译的 JSP 网页提供定义。包括 Servlet/JSP 的名字，Servlet/JSP 的类以及一个可选的描述。
- **Servlet/JSP 映射** Web 容器使用这些信息把进入请求映射到 servlet 和 JSP 网页。
- **MIME 类型** 由于每个 Web 应用程序可以包含多种内容类型，因此我们可以在发布描述符中为每一种类型指定 MIME 类型。
- **安全性** 我们可以使用发布描述符来管理应用程序的访问控制。例如，可以指定我们的 Web 应用程序是否需要登录，如果需要的话，应该使用什么登录页面，以及用户会作为何种角色。

发布描述符还可以用来自定义其他元素，包括欢迎网页，出错网页，会话配置。

classes 目录用于存储编译过的 servlet 及其它程序类，例如 `JavaBean`。如果一个程序有打包的 JAR 文件(例如一个第三方 API 打包成了一个 JAR 文件，如 Struts 框架的类库 `struts.jar`，MySQL 的数据库 JDBC 驱动程序文件 `mysql-connector-java-3.1.11-bin.jar` 等)，

那么它们可以被复制到 **lib** 目录中(如果解压缩这些压缩包的话, 请将它们复制到 **classes** 目录中)。Web 容器使用这两个目录来查找 **servlet** 及其他相关类, 也就是说, 容器的类装入器会自动查看 **classes** 目录, 以及 **lib** 目录下的 JAR 文件。这就意味着你不需要明确的把这些类和 JAR 文件添加到 **CLASSPATH** 中。Web 容器自动将这两个目录中的文件加入 Web 应用的类路径中。

8.2.2 MyEclipse Web 项目介绍

MyEclipse Web Project 完全支持上一节所提到的 Web 应用的目录结构。如下图所示:

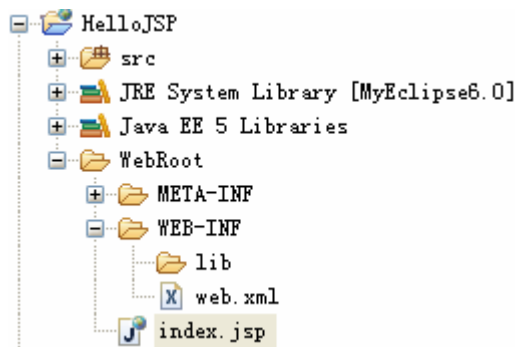



图 8.2 MyEclipse Web Project 结构

Web 项目的图标以  的格式显示。**src** 目录下面的 Java 源代码编译后的类文件将会输出到 **WebRoot/WEB-INF/classes** 下面。**WebRoot** 目录则包含了发布后的 Web 项目的目录结构, 例如图中显示的 **index.jsp**, 发布后的路径是 **d:\tomcat6\webapps\HelloJSP\index.jsp**, 这就是这个目录的特殊之处。而 **WEB-INF**, **web.xml**, **lib**, **classes** 目录的作用则请参考上一节的内容。

MyEclipse Web 项目可以通过新建或者向现有项目添加 Web 开发能力来创建。

注意: 只有一个项目是 MyEclipse Web 项目时才可以被发布到服务器上运行。

8.3 创建 Web 项目

本节内容将介绍如何创建一个 *JSPHelloWorld* 的 Web 项目。选择菜单 **File > New > Web Project**, 可以启动创建 Web 项目的向导, 如图 8.3 所示。

在这个图的 **Project Name** 中输入 *JSPHelloWorld*, 然后选中 **J2EE Specification Level** 下面的 **Java EE 5.0** 单选钮, 最后点击 **Finish** 按钮就可以创建 Web 项目了。创建完成后的 Web 项目和图 8.2 所示相似。

注意: 选择哪个版本的 J2EE Specification Level 取决于你使用的服务器, 例如 Tomcat 4, Weblogic 9 以下版本请选择 J2EE 1.4, 而 Tomcat 5, JBoss 4, 或者 GlassFish 这样的服务器可以选择 Java EE 5.0。Java EE 5.0 可以直接使用 EL 表达式和 JSTL。

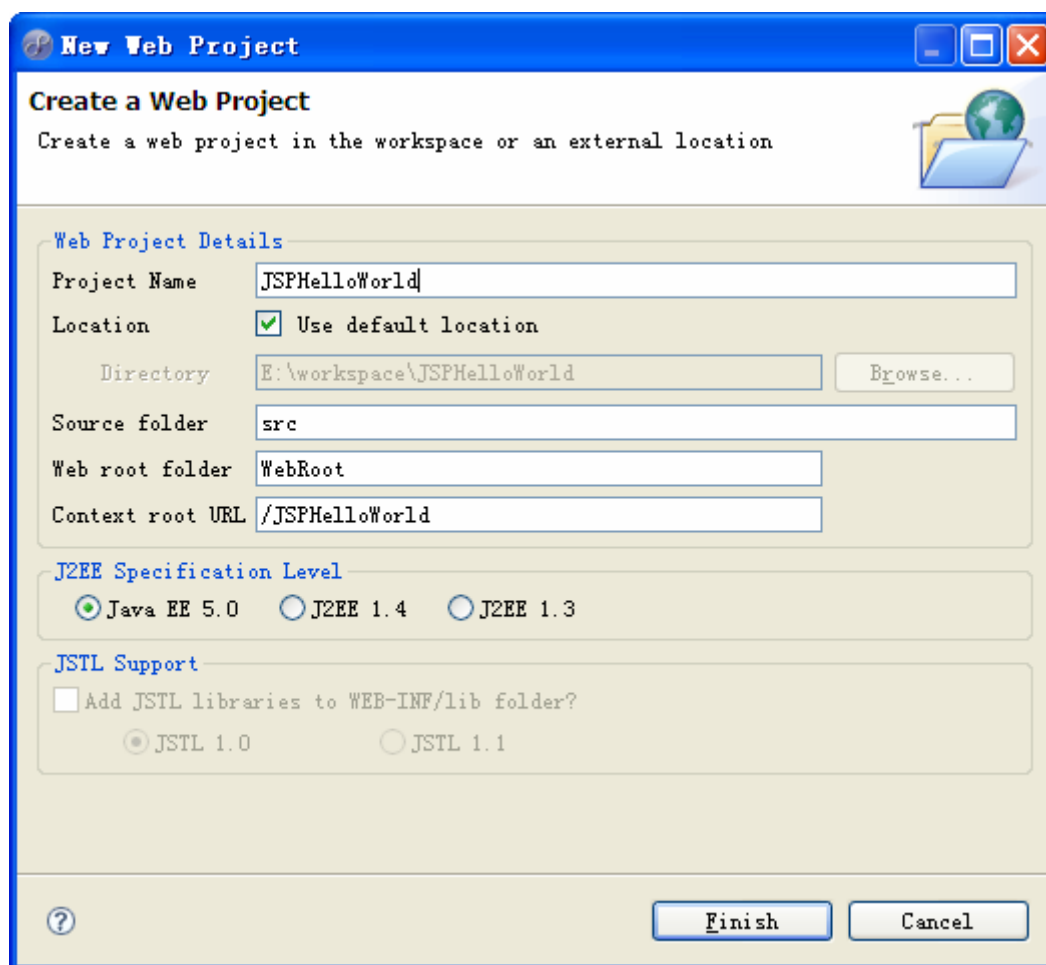


图 8.3 新建 Web Project 对话框

关于输入框的详细意义请参考下表：

选项	描述
Project name	项目的名称。必须是有效的 Eclipse Java 项目名。。
Location	选中这个复选框来选择新项目的文件将存放到电脑上的其它位置
Directory	项目的默认存放位置是放在 MyEclipse 启动时候工作区目录下的。当然可以选择位于工作区目录外的其它路径。 注意： 不能选择位于工作区目录下的另一子目录，因为 Eclipse 禁止这种做法！
Source folder	Java 源代码目录—将包含 Java 包和.java 文件。这些内容会被加入到项目的 Java 构造路径中。
Web root folder	这个目录将包含 web 应用的内容，WEB-INF 目录以及对应的子目录。如果这个输入框内容为空，那么项目的根目录 ("/") 将会成为 web 根目录。
Context root URL	MyEclipse的发布工具会发布新Web项目时候所使用这个路径。默认使用的值是项目的名字。什么是上下文根目录？它是访问发布后的应用时所用的根路径，例如输入myapp后，将用地址 http://localhost:8080/myapp 来访问这个项目。你可以把这个输入框中的内容修改成全是小写字母的内容。
J2EE specification	指定 J2EE 规范的版本。需要检查服务器的文档来了解其所支持的版本。

level	
Add JSTL 1.0 libraries	启用此选项来添加 Java Standard Template Library (Java 标准模版库 1.0 或者 1.1 版本)的 JAR 文件到新项目的<web-root>/WEB-INF/lib 目录下。

表 8.1 新建 Web 项目的选项说明

8.4 创建 HTML 页面

注意：在实际的开发中，一般使用的都是像 Frontpage 或者 DreamWeaver 这样的工具来创建，可视化的（所见即所得，WYSWYG）的来修改静态网页（HTML）。因为 MyEclipse 的可视化网页编辑器功能是比较弱的。因此本节内容仅供参考。

启动创建 HTML 页面的对话框有多种方式，这里只介绍两种：1. 选择菜单 **File > New > Html(Advanced Template)**；2. 选中 **Package Explorer** 视图的 **WebRoot** 目录，点击右键选择上下文菜单中的 **New > Html(Advanced Template)**。这时候将会弹出创建 HTML 页面的对话框，如下图所示：

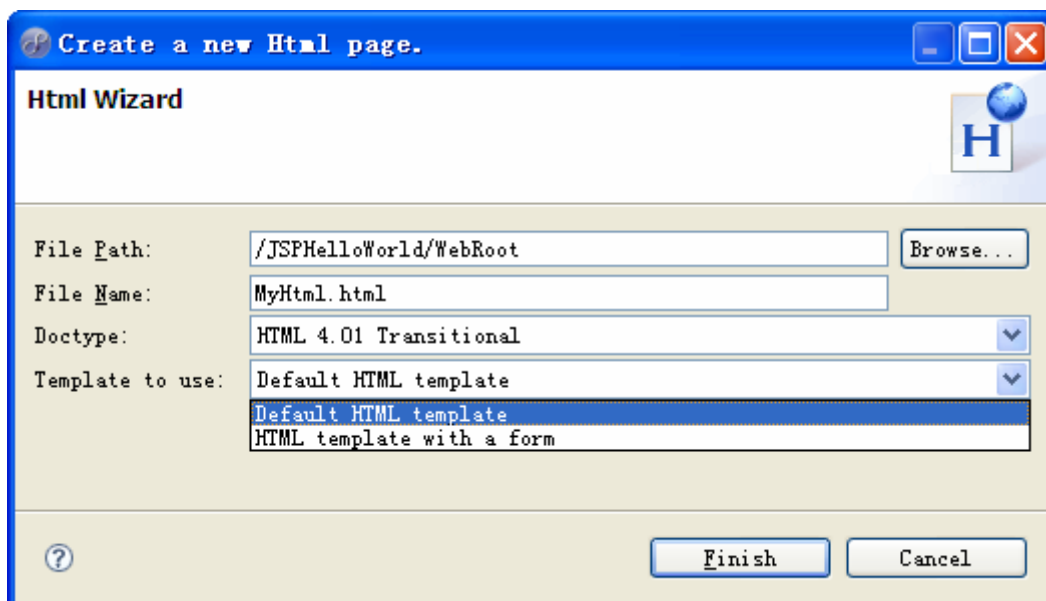


图 8.4 新建 HTML 页面向导

在这个对话框中的 **File Name**（文件名）框中输入 *login.html*，**Template to use:**（要使用的模版）右侧的下拉框选中 *Default HTML template*（默认 HTML 模版，另一个是带表单的模版）。最后点击 **Finish** 按钮完成向导。稍后 MyEclipse 会用 HTML 编辑器来打开刚创建的文件，如图 8.5 所示。在这里可以在页面设计器中可视化的修改网页内容，也可以点击格式工具栏上的按钮来可视化的修改网页的格式，还可以在源代码面板中直接修改 HTML 源码。点击 **Preview** 标签可以同时 IE 和 Mozilla 浏览器中查看页面的显示效果。

点击 **Palette**（调色板，确切说应该是叫组件面板）可以选择对应的一些常用的代码片段插入到当前页面中，例如超链接，图片，表单和表单元素等等。

现在我们把它的内容修改成一个包含登录表单的页面，然后点击工具栏上的保存按钮。代码内容请参考清单 8.1。

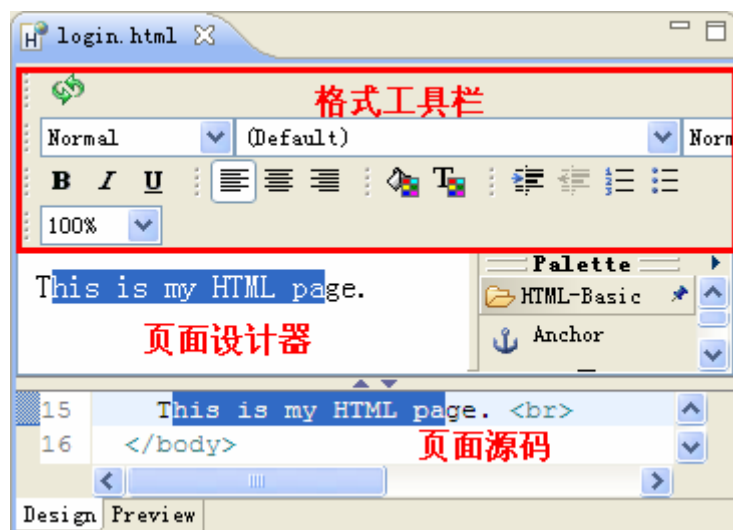


图 8.5 HTML 编辑器

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>登录</title>
    <meta http-equiv="content-type" content="text/html; charset=GBK">
  </head>
  <body>
    请登录 <br>
    <form action="login.aspx" method="post">
      用户名: <input type="text" name="username"><br>
      密码: <input type="password" name="password"><br>
      <input type="submit" value="提交" name="submit1">
      <input type="reset" value="重置" name="reset1">
    </form>
  </body>
</html>
```

清单 8.1 login.html 源码

把代码保存完毕后，页面看起来将如下图所示：

请登录
 用户名:
 密码:

清单 8.2 登录页面

至此，创建静态页面的过程就简要介绍完毕了。在此先虚晃一枪，为什么表单的提交页面是 login.aspx 呢？难道 Java 也支持 ASP.NET 嘛？

8.5 创建 JSP 页面

本节内容将会讲解创建 JSP 页面。实际上 JSP 编辑器许多地方都是和上文介绍的 HTML

编辑器非常相似的。因此本节内容将会简要介绍其过程。

启动创建 HTML 页面的对话框有多种方式，这里只介绍两种：1. 选择菜单 **File > New > JSP(Advanced Template)**；2. 选中 **Package Explorer** 视图的 **WebRoot** 目录，点击右键选择上下文菜单中的 **New > JSP(Advanced Template)**。这时候将会弹出创建 JSP 页面的对话框，和图 8.4 非常相似。只需要在这个对话框中的 **File Name**（文件名）框中输入 *result.jsp*，然后点击 **Finish** 按钮即可创建这个 JSP 页面。

注意：Template to use 右侧的模版下拉框中有很多 JSP 模版可以使用，例如支持 JSF，Struts 等等的模版，这样可以加快开发的速度。

稍后 MyEclipse 会用 HTML 编辑器来打开刚创建的文件，界面已经操作方法和图 8.5 非常类似，不同的是 **Palette** 里面多了很多 JSP 特有的内容，而编辑器的代码视图呢，也支持自动查错（但是不支持自动修正错误）和代码编写提示功能，如下图所示：

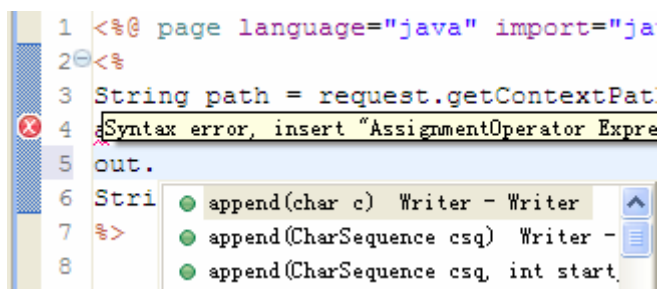


图 8.6 JSP 编辑器的查错和编码提示

，当你在变量后按下.之后，会弹出代码完成提示，另外还支持断点的设置等等。因此使用 MyEclipse 的 JSP 编辑器可以大大减少开发人员出错的机会（在出现能查错的 JSP 编辑器之前这是个大问题）。

好了，现在我们将把这个页面的代码改写成如下清单所示内容：

```
<%@ page language="java" pageEncoding="GBK"%>
<html>
<head>
<title>登录结果</title>
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
</head>
<body>
    您登录${message}了！
</body>
</html>
```

清单 8.3 登录结果页面 result.jsp

8.6 创建 Servlet

启动创建 Servlet 的对话框有多种方式，这里只介绍两种：1. 选择菜单 **File > New > Servlet**；2. 选中 **Package Explorer** 视图的项目，点击右键选择上下文菜单中的 **New > Servlet**。这时候将会弹出新建 Servlet 类的对话框，如图 8.7 所示。在这个对话框中的 **Package**（包）框中输入 *servlets*，**Name**（类名）输入 *LoginServlet*，然后点击 **Next** 按钮可以进一步设置映射文件。也可以点击 **Finish** 按钮直接完成这个创建向导，不过此处将选择 **Next**。

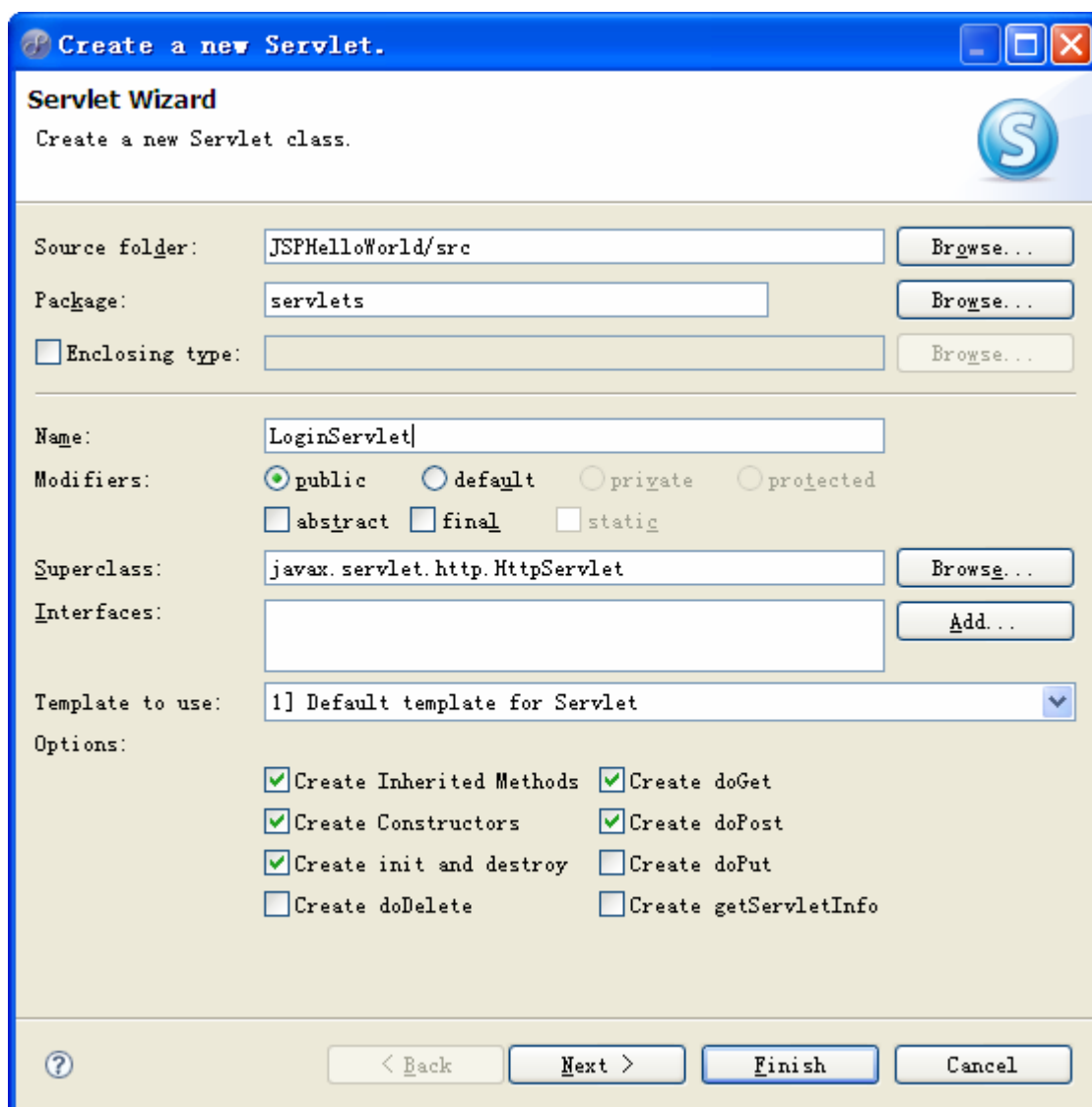


图 8.7 新建 Servlet 类对话框

在这一页中可以设置 Servlet 的父类（**Superclass**），以及修饰符（**Modifiers**），添加接口（**Interfaces**），选择模版（**Template to use**）以及一些选项（**Options**）。Options 中可以选择是否创建继承的方法（**Inherited Methods**），创建构造器（**Constructors**），初始化和销毁方法（**init and destroy**）以及 doGet, doPost, doPut, doDelete, getServletInfo 等方法。详细意义请参考 Servlet 开发的书籍。

当点击 **Next** 按钮后，将会进入修改，设置 web.xml 的向导页面，如图 8.8 所示。注意图中的红框，我们在这里在 **Servlet/JSP Mapping URL** 右侧输入框中输入 `/login.aspx`。这个路径可以帮你理解一个概念，那就是其实 Servlet 的后缀可以是任何形式的字符串，例如 .do, .php 等等。最后点击 **Finish** 按钮来完成创建 Servlet 的过程。

注意：Servlet 的映射路径一定要以 `/` 开始，或者以 `*.do` 的方式出现，而且不能输入 `/*.do`。

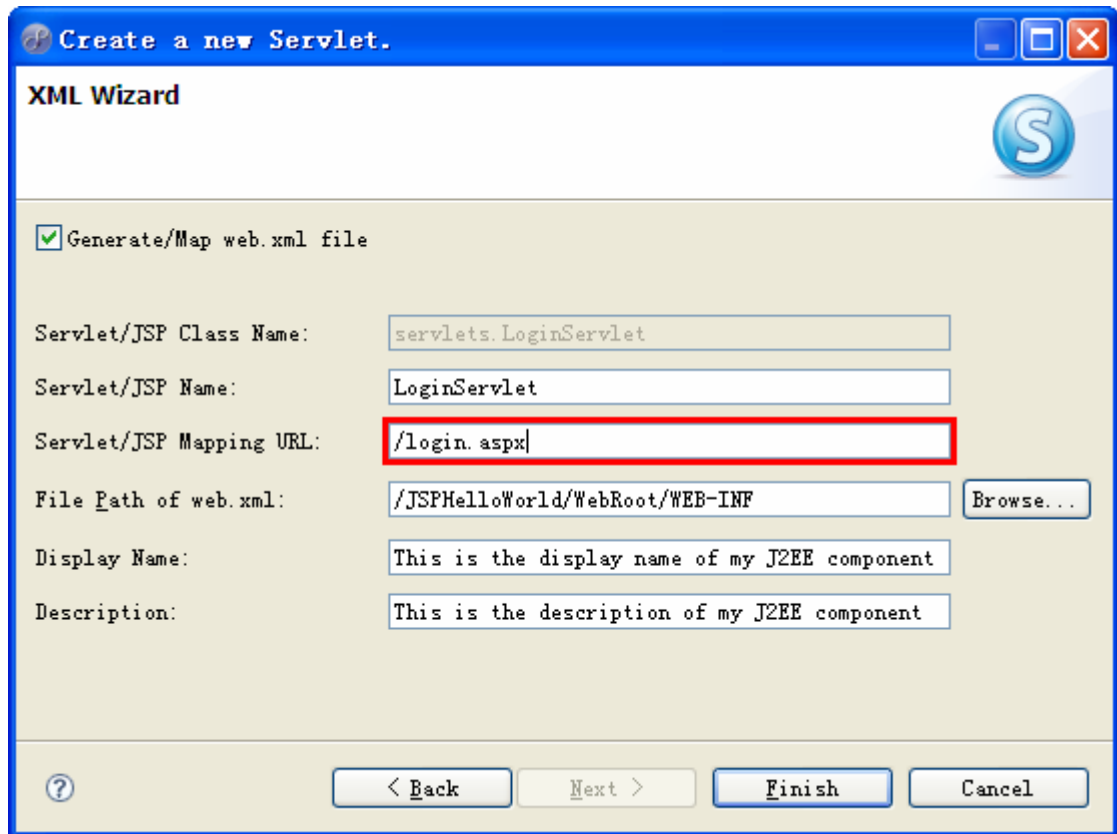


图 8.8 修改，设置 web.xml 中的映射信息

对话框关闭后，稍等片刻 web.xml 和新创建的 LoginServlet.java，可以看到 web.xml 的内容已经被自动加入了新的 Servlet 定义，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <description>This is the description of my J2EE
component</description>
    <display-name>This is the display name of my J2EE
component</display-name>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>servlets.LoginServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login.aspx</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
```

```
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

清单 8.4 加入了 Servlet 定义的 web.xml

至此Servlet就创建出来了，你可以接着修改Servlet的源码来加入更多功能。这个Servlet的最终访问路径是：<http://localhost:8080/JSPHelloWorld/login.aspx>，是不是看起来非常像.NET应用呢？不过这是个假的而已。

8.7 创建 Filter(过滤器)

实际开发中都需要开发一些很有用的过滤器，来解决中文表单提交问题啊，给请求和响应加入 GZIP 压缩功能啊，用户权限控制啊，等等，然而遗憾的 MyEclipse 不支持直接创建过滤器。在这里只好手工创建一个解决 Tomcat 表单提交中文问题的过滤器。

选择菜单 **File > New > Class**，来创建一个名为 **TomcatFormFilter** 的类，包名为 **filters**。然后把类的代码修改为如下所示：

```
package filters;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

public class TomcatFormFilter implements Filter {
    /**
     * Request.java
     * 对 HttpServletRequestWrapper 进行扩充，不影响原来的功能并能提供所
     * 有的 HttpServletRequest
     * 接口中的功能。它可以统一的对 Tomcat 默认设置下的中文问题进行解决而只
     * 需要用新的 Request 对象替换页面中的
     * request 对象即可。
     */

    class Request extends HttpServletRequestWrapper
    {

        public Request(HttpServletRequest request) {
            super(request);
        }
    }
}
```

```
/**
 * 转换由表单读取的数据的内码。
 * 从 ISO 字符转到 GBK。
 */
public String toChi(String input) {
    try {
        byte[] bytes = input.getBytes("ISO8859-1");
        return new String(bytes, "GBK");
    }
    catch (Exception ex) {
    }
    return null;
}

/**
 * Return the HttpServletRequest holded by this object.
 */
private HttpServletRequest getHttpServletRequest()
{
    return (HttpServletRequest)super.getRequest();
}

/**
 * 读取参数 -- 修正了中文问题。
 */
public String getParameter(String name)
{
    return
toChi(getHttpServletRequest().getParameter(name));
}

/**
 * 读取参数列表 - 修正了中文问题。
 */
public String[] getParameterValues(String name)
{
    String values[] =
getHttpServletRequest().getParameterValues(name);
    if (values != null) {
        for (int i = 0; i < values.length; i++) {
            values[i] = toChi(values[i]);
        }
    }
}
```

```

        return values;
    }
}

public void destroy() {

}

public void doFilter(ServletRequest request, ServletResponse
response,
    FilterChain chain) throws IOException, ServletException {
    HttpServletRequest httpreq = (HttpServletRequest)request;
    if(httpreq.getMethod().equals("POST")) {
        request.setCharacterEncoding("GBK");
    } else {
        request = new Request(httpreq);
    }

    chain.doFilter(request, response);
}

public void init(FilterConfig filterConfig) throws
ServletException {
}
}

```

清单 8.5 过滤器代码

然后修改 web.xml 加入 Servlet 定义，修改后的代码清单如下所示：

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <description>
            This is the description of my J2EE component
        </description>
        <display-name>
            This is the display name of my J2EE component
        </display-name>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>servlets.LoginServlet</servlet-class>
    </servlet>

    <filter>

```

```

        <filter-name>TomcatFormFilter</filter-name>
        <filter-class>filters.TomcatFormFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>TomcatFormFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login.aspx</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>

```

清单 8.6 加入了过滤器的 web.xml 内容

清单中的粗斜体部分就是新加入的过滤器的映射信息。

8.8 创建数据库访问层(DAO)

做 Web 应用一般来说不访问数据库是不太可能的，因此本节就介绍给应用加入数据库访问功能。

首先第一步是创建数据库表，参考 [5.2 创建数据库表格](#) 节内容进行操作。

第二步是要加入JDBC驱动类库，详情请参考 [5.4 添加JDBC驱动到Build Path](#) 一节。这里打算使用MySQL数据库，对于Web项目来说加入类库文件非常容易，只要把 *mysql-connector-java-3.1.11-bin.jar* 这个文件复制到 **WebRoot/WEB-INF/lib** 下，MyEclipse会自动把文件加入到项目的类路径中。

第三步需要创建一个实体类，来代表数据库中的 Student 对象，这个类用来保存和传递来自数据库的数据信息。代码清单如下：

```

package entity;
/** 学生实体类 */
public class Student {
    private int id;
    private int age;
    private String username;
    private String password;
    public int getId() {
        return id;
    }
    public void setId(int id) {

```

```

        this.id = id;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

清单 8.7 Student 实体类

最后是创建一个数据库访问对象，请参考 [5.5 编写JDBC访问类](#) 一节的内容。所不同的是这个类将会放入dao包里面，类名叫StudentDAO。其源码如下所示：

```

package dao;
import java.sql.SQLException;
import entity.Student;
/**
 * 学生数据访问类
 * @author BeanSoft@126.com
 * @version 0.1 2007-12-21
 */
public class StudentDAO {
    /**
     * 根据用户名和密码找到用户对象。
     * @param username 用户名
     * @param password 密码
     * @return 找到的用户对象，找不到返回null
     */
    public Student findStudent(String username, String password) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {

```

```

        e.printStackTrace();
    }

    java.sql.Connection conn = null; //数据库连接
    java.sql.PreparedStatement pstmt = null; //数据库表达式
    java.sql.ResultSet rs = null; //结果集
    String sql = "select * from Student where username = ? and password
= ?"; //SQL

    try {
        conn = java.sql.DriverManager.getConnection(

            "jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=GBK", "root", null);

        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, username);
        pstmt.setString(2, password);

        rs = pstmt.executeQuery();

        if(rs !=null && rs.next()) {
            // 读到数据，生成实体类
            Student student = new Student();

            student.setId(rs.getInt(1));
            student.setUsername(rs.getString("username"));
            student.setPassword(rs.getString("password"));
            student.setAge(rs.getInt("age"));

            return student;
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // 6. 释放资源，建议放在finally语句中确保都被关闭掉了
        try {
            rs.close();
        } catch (SQLException e) {}
        try {
            pstmt.close();
        } catch (SQLException e) {}
        try {

```

```

        conn.close();
    } catch (SQLException e) {}
}

return null;
}
}

```

清单 8.8 StudentDAO 数据访问类

至此数据访问层的开发已经完成,为什么要多写这么多代码呢?这是因为分层的设计能够便于多人合作开发,也便于单独测试每一层的功能。当然,项目规模小的话把所有的代码都放到那个 Servlet 里面就行了。

8.9 修改 Servlet 调用后台类

现在我们可以修改 Servlet 来加入调用 DAO 层代码然后判断登录的功能了,设置完登录状态后,会转向到/result.jsp。修改后的 Servlet 代码如下所示:

```

package servlets;

import java.io.IOException;

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

import dao.StudentDAO;
import entity.Student;

public class LoginServlet extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        Student student = new StudentDAO().findStudent(

request.getParameter("username"),request.getParameter("password")
);

        if(student != null &&
student.getUsername().equals(request.getParameter("username")) &&

student.getPassword().equals(request.getParameter("password"))) {
            request.setAttribute("message", "成功");
            //保存登录用户到session中

```



```



        request.getSession().setAttribute("student", student);
    } else {
        request.setAttribute("message", "失败");
    }
    //转向登录结果页面
    request.getRequestDispatcher("/result.jsp").forward(request,
response);
    }
}

```

清单 8.9 Servlet 加入登录判断功能

至此，这个简单的登录小项目就开发完毕了。

8.10 发布，重新发布，运行和测试应用

如何发布，运行Web项目请参考 [6.5.2 向服务器发布应用](#)，[6.6.1 启动服务器](#)一节的内容，完整的服务器管理过程可以参考 [第六章 管理应用服务器](#)。点击主界面工具栏上的按钮发布完毕后就可以启动服务器来进行测试了。在**Servers**视图中选中服务器，之后点击视图工具栏上的按钮以运行模式启动服务器。

服务器启动完毕后，点击工具栏上的按钮来打开浏览器视图，然后键入地址：<http://localhost:8080/JSPHelloWorld/login.html> 来打开登录页面，如下图所示：

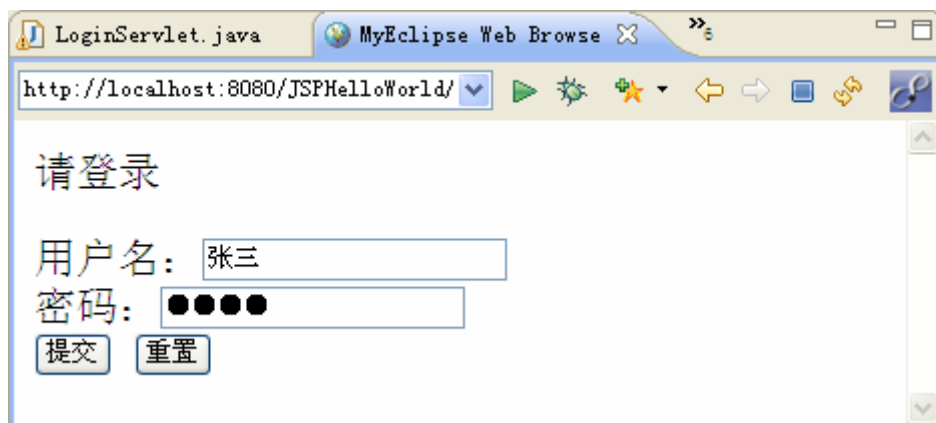


图 8.9 在 MyEclipse 中打开浏览器测试应用

在表单的**用户名**处输入数据库里存在的学生的名字，例如张三，再输入密码，然后点击提交，就可以显示登录的结果了。

如果只是修改了 JSP 页面，那么 MyEclipse 会自动把 JSP 更新到服务器上，但是如果是改了类文件或者一些配置文件，那么需要手工**重新发布**这个项目。如何重新发布这个项目呢？我们可以在 **Servers** 视图上选中所发布的项目，然后点击视图工具栏上的来重新发布，或者在项目上点击右键选择菜单 **Redeploy**，如图 8.10 所示。之后稍等片刻就可以完成重新发布的过程。

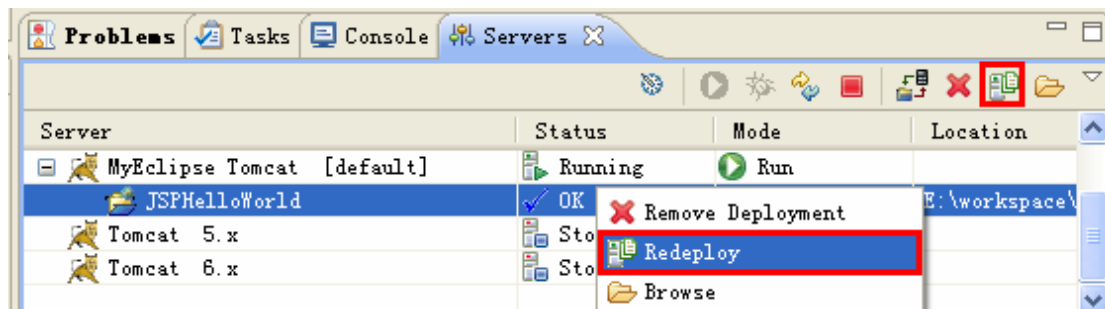


图 8.10 重新发布项目

8.11 调试 JSP 应用

如 [6.6.4 调试发布的企业应用](#) 一节介绍，MyEclipse 可以对 Web 应用里面的类或者 JSP 页面，Servlet 进行调试。例如可以双击 JSP 编辑器的行首的隔条，来设置断点，然后以调试模式启动服务器。例如下面是访问 index.jsp 时的调试透视图界面，我们可以修改变量值 b 为 4（被修改过的值以黄色高亮显示），然后点击 Debug 视图的 Resume 按钮来继续往下执行，就可以看到最终的执行结果是 5。如下图所示：

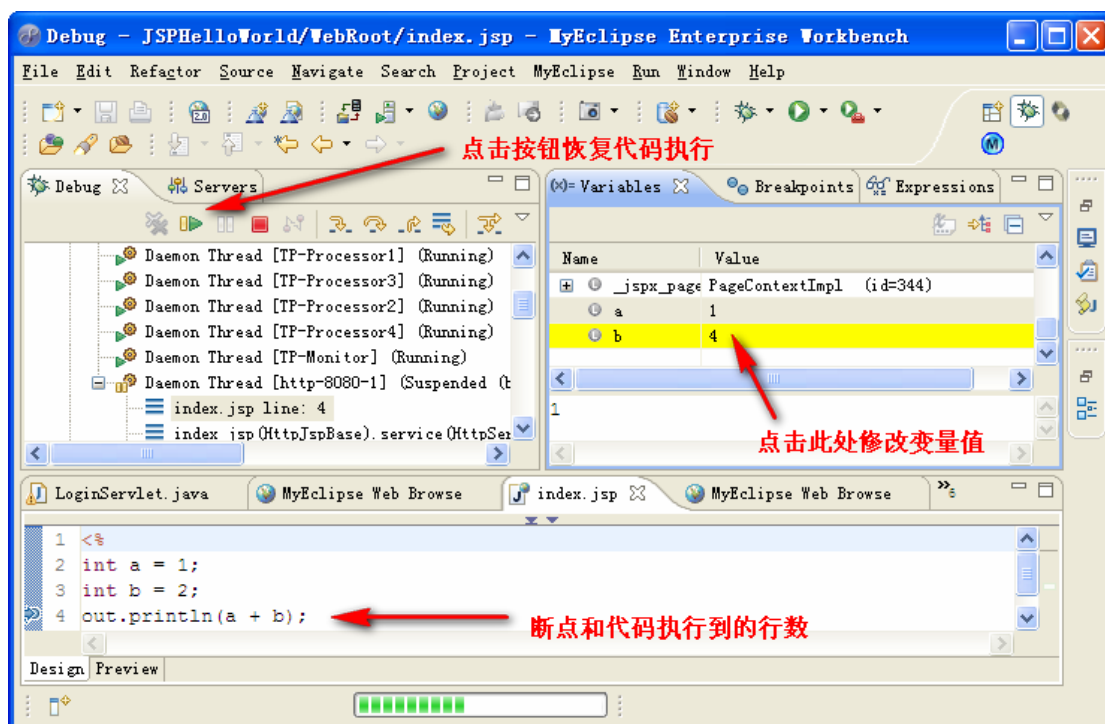


图 8.11 修改变量跟踪调试 JSP 页面

这里用的 index.jsp 页面的源码清单如下所示：

```

<%
int a = 1;
int b = 2;
out.println(a + b);
%>
  
```

清单 8.10 用来调试的 JSP 页面源码

使用调试器可以很方便的快速定位出现问题的地方，加快修改代码的速度。

8.12 向现有 Web 项目添加 Web 开发能力

如果拿到了一个其它开发工具例如 WTP, Netbeans, JBuilder 等所制作的 Web 项目, 而不是通过 MyEclipse 所创建的 Web 项目, 那么 MyEclipse 将会拒绝对它进行发布, 调试等操作。这种情况下可以从 MyEclipse 菜单栏选择 **MyEclipse > Project Capabilities > Add Web Project Capabilities ...** 来启动 **MyEclipse Web Capabilities** 向导, 如下图示:

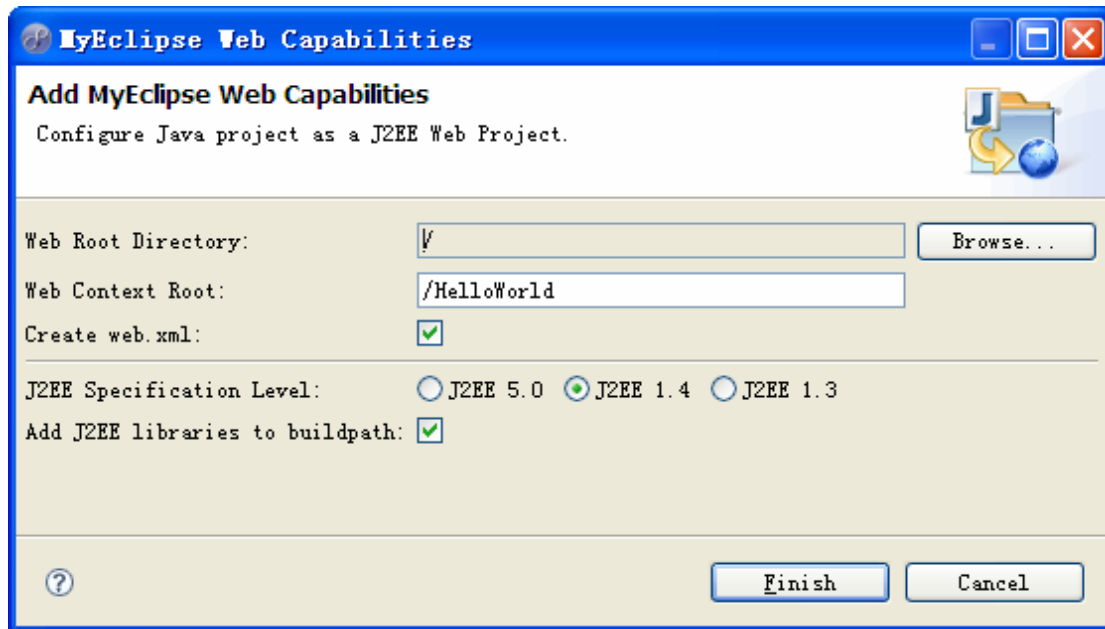


图 8.12 MyEclipse Web 项目向导

在这个对话框中选中 Web 项目的根目录, 点击 **Web Root Directory** 最右侧的 **Browse...** 按钮来选中, 之后设置上下文访问路径 (**Web Context Root**), 并选择合适的 Java EE 版本 (**J2EE Specification Level**) 后, 点击 **Finish** 按钮后就给当前项目加入了 Web 开发能力了, 之后就可以方便的对它进行发布等操作。

8.13 高级设置

本节内容仅供了解, 大部分情况下都不需要对这些内容进行修改。

8.13.1 修改 Web 项目的默认设置

选择菜单 **Window > Preferences**, 打开 Preferences 对话框, 在选项树上选择 **MyEclipse > Java Enterprise Project > Web Project**, 来进一步设置 Web 项目的默认设置, 如下图所示:

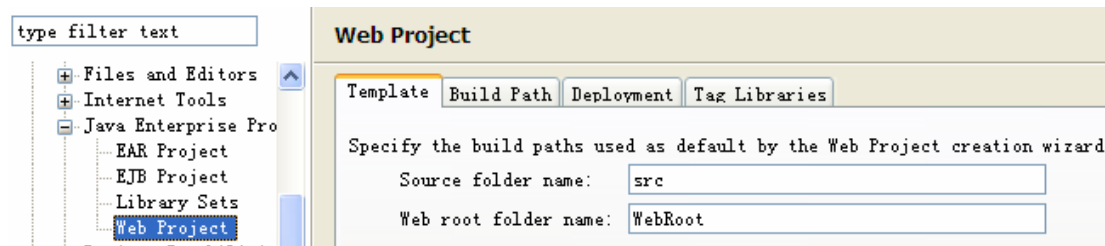


图 8.13 修改 Web 项目的默认设置

在 **Template** 标签页可以设置 Web 项目的模版, 包括默认的源代码目录名以及 Web 应用根目录的名称。而在 **Build Path** 标签页, 则可以设置是否自动将 WEB-INF/lib 下面的 jar 或者 zip 文件发布到服务器上。**Deployment** 标签则设置了当项目存在依赖的时候如何进行发布。**Tag Libraries** 标签则可以修改一些自定义的标签库的快速代码段。

除此之外, 还可以对单个项目的 Web 功能进行设置。点击菜单 **Project > Properties** 可以打开项目的属性对话框, 这时候可以点击 **MyEclipse > Web** 节点进行一些必要的设置, 如下图所示:

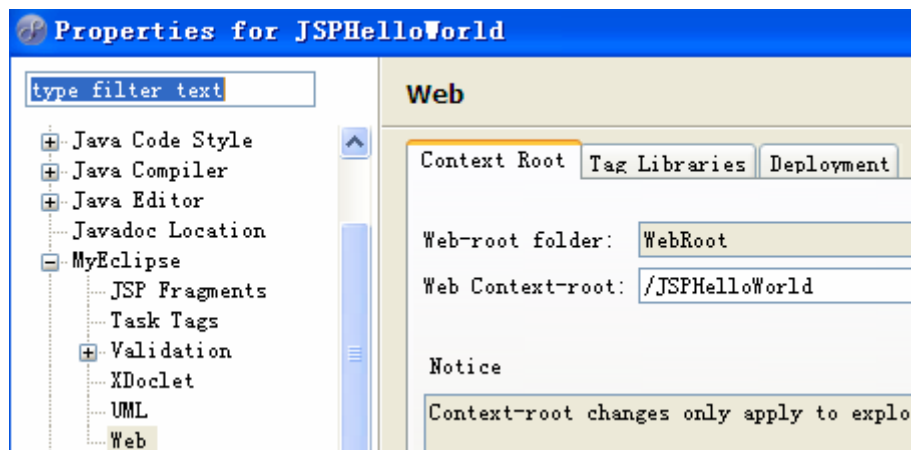


图 8.14 修改单个项目的设置

8.13.2 给 Web 项目加入高级功能

在实际开发中不可避免的要在 Web 项目中使用 Hibernate, Struts, Spring 等技术, 那么这些都可以点击菜单 **MyEclipse > Project Capabilities** 然后选择需要使用的技术, 就可以将对应的类库和配置文件加入到当前项目中。例如要开发 Struts 和 Hibernate 应用, 分别点击两次子菜单 **Add Struts Capabilities** 和 **Add Hibernate Capabilities** 就可以了。

8.14 小结

在本章我们介绍了如何开发, 发布, 运行, 测试, 调试 Web 应用, 这些概念适用于以后所介绍的其它基于 Web 的项目例如 Struts, JSF 等等。通过本章, 你将对如何使用 MyEclipse 如何开发 Web 项目有一个大致的了解, 并为后续 Web 框架开发打好基础。

第九章 开发 **Struts 1.x** 应用

第十章 开发 **JSF** 应用

第十一章 开发 **XFire Web Service** 应用

第十二章 开发 **JPA** 应用

第十三章 开发 **Spring** 应用

第十四章 开发 **Spring+Struts+Hibernate** 应用

第十五章 **MyEclipse UML** 开发

第十六章 开发 **EJB** 应用

附录