第 27 章　Web项目开发

前面已经比较全面地向读者介绍了网站开发过程中所涉及到的技术。本章将重点向读者介绍软件开发的基本过程、以及在实际开发过程中的 CVS 版本控制工具的使用，另外还对数据库连接池技术作了总结，并介绍 PoolMan 连接池配置方法。从而为下面两个章节介绍的综合 Web 项目作准备。

本章要点包括以下内容：

- 对 Web 应用开发方式作总结
- 软件开发的基本过程
- CVS 版本控制工具的安装和使用
- PoolMan 数据库连接池的配置

27.1　Web开发方式总结

前面章节以实例形式介绍了多种 Web 开发方式，下面具体总结这些方式的优缺点，以及适用的范围。

27.1.1　纯JSP页面Web开发

在第 3 章和第 12 章列举的 JSP 应用实例就是使用纯 JSP 页面实现的。纯 JSP 页面开发方式，就是将所有的功能代码都在 JSP 页面中实现，其中逻辑功能是嵌入在<%和%>之间的 Java 脚本语言实现的，这包括必要的数据库连接和操作语句。这样的一种 Web 开发方式，可以使得代码相对集中，从而便于管理。但是这种方式的缺点也相当明显，由于各种功能的代码相对太集中，显得非常混乱，特别是当代码量非常大时。另外，大量代码集合在一个 JSP 页面上，也使得耦合性非常差，从而导致代码的重用性也比较低。

总结起来，这样的开发方式只适合超小型项目，便于 JSP 初学者学习。因为这样的开发方式不需要考虑太多的层次结构。

27.1.2　JSP＋JavaBean的Web开发

在本书的第 14 章，引入了 JavaBean 的概念，并使用 JSP＋JavaBean 重新实现了在第 12 章列举的用户注册系统。使用 JavaBean 类来实现逻辑操作，可以更好的实现代码重用，因为 JSP 页面可以随意调用所需要的 JavaBean 类，并进行相应操作。

这样的开发方式可以适用在中型 Web 项目中，层次结构不太复杂，易于读者的了解。在接下来介绍的综合实例使用了 JSP＋JavaBean 方式。

27.1.3 JSP＋Struts的Web开发

不管是纯 JSP 开发方式，还是 JSP＋JavaBean 开发方式，都属于 Model1 模式。MVC 框架可以使得 Web 项目层次结构清晰，JSP 页面使用来进行 View 显示功能，而使用 Servlet 来实现控制层，JavaBean 类实现相应的 Model 数据模型层。Struts 是实现 MVC 的最为流行的一种解决方案，它可以使得 Web 项目的后期程序代码的维护和扩展更加容易，并且使用 Struts 划分项目层次，可以更好地实现职责划分。

使用 Struts 开发可以更好的实现大型项目。但是对初学者来说，不易于理解，因为开发步骤更加繁琐和复杂。

27.1.4 JSP＋Struts＋Hibernate的Web开发

在 Web 应用系统开发过程中，使用 MVC（Model 模型－Veiw 视图－Controller 控制）模式作为系统的构架模式。MVC 模式实现了架构上表现层（View）和数据处理层（即 Model）分离的紧耦合。但是由于 Model 模型层中包含了复杂的业务逻辑和数据逻辑，以及数据存储机制（例如 JDBC 连接、SQL 语句生成、statement 创建以及 ResultSet 结果集的读取等操作）等。为了将系统的紧耦合关系转化为松耦合关系（即解耦合），急迫需要将这些复杂的业务逻辑和数据逻辑进行分离，这就是持久层技术需要做的工作。

Hibernate 数据库持久层技术改变了传统的面向过程的数据设计，实现了与系统功能的面向对象设计相统一。Struts＋Hibernate 是当今最为流行的 Web 开发组合。

27.1.5 自定义标签库、标准标签库的Web开发－JSP2.0 范畴

自定义标签开发以及标准标签的使用在本书的第十六、十七章作了重点介绍。使用标签库来实现 Web 动态性，可以使得 JSP 页面风格更加统一，但是这种 Web 开发方式现在还不是很流行，可以是由于自定义标签的开发过程比较繁琐。读者对这方面知识只要了解就行。

27.2 软件开发的基本过程

任何一个项目开始之前，进行充分的需求分析是非常重要的，而且需要分析是一门很需要技术和技巧的工作，必须工作人员有丰富的工作经验和耐心。做需求分析是进行软件开发的第一步，一旦需求分析错误，后期所有的工作将是白白浪费。进行需求分析就是一个交流、沟通以及对数据、业务流程进行调查的过程，是一个反复多次的过程。需求分析非常重要，虽然很对开发人员和项目管理人员都明白这个道理，但是在现实当中，忽视或者不重视需求分析的情况随时可见，最终导致客户对产品的不满。

那什么是软件开发成功的标准了，就是能够按期完成开发进度并且最终产品能够获得客户的满意和好评。下面介绍一个比较基本的软件开发过程，详细和周密的开发计划和控制是软件开发成功的关键：

27.2.1 需求分析阶段

在软件产品/服务开发之前，客户方和开发方都必须成立专门的需求分析小组或者指派专门的负责人进行双方的交流与沟通，而且这一过程必须是持续的、多次反复的。调查的对象包括企业或者单位的业

务流程以及数据流向等，必要时，还需要和客户方的高层领导和使用人员进行深刻的交流和沟通，因为开发人员需要了解到这些使用者希望最终的产品或者系统能做什么。

开发方的需求调查人员在对客户方做出相关调查和分析之后，需要使用 Word、Visio 或者别的工具绘出软件的界面、业务和数据流程图等（这些都是开发方根据自己调查之后的理解编辑的文档），并附上详细的文字说明，然后交付给用户负责人员进行详细的审核。用户通过这些文档和图形进行初步的认识和理解，在这过程中，需要开发人员和客户方人员进行讨论，尽量让用户理解文档和图形的真正含义。在做需求分析时，最容易出现的问题就是和客户进行交流时，双方出现理解上的偏差，即你认为用户是这样理解的，其实不然，或者客户认为你已经理解他的意思，但是你没有，这是因为语言存在模糊性的原因。所以开发人员在编写需求文档时，要尽量地使用可视化的图形，因为图形更能表示出流程的逻辑性关系。大部分情况下，用户根本不懂技术，所以图形的使用也可以避免大量技术概念的出现。所有这些也是产生 UML（统一建模语言）的原因。

27.2.2　系统设计阶段

经过以上多次的文档和口头交流和讨论之后确定出最终的需求。开发人员根据该需求进行功能设计和数据库设计，然后组织开发小组进行开发。

27.2.3　开发和测试阶段

开发过程可能会根据应用系统的功能分成多个模块，进行模块的分组开发，最后再把多个模块整合在一起（注意各模块之间的接口设计）。当软件产品完成 70%之后，就可以进行初步的产品安装、配置和运行，并由专门的测试人员进行测试工作。

使用一个专门的 BUG 管理软件来管理所有的 BUG，测试人员把检测出来的 BUG 加入到管理系统中，这样开发人员可以随时查看到自己的新 BUG 以及已经修改的 BUG 是否已经通过了测试。这样对 BUG 进行了文档性管理，所有的 BUG 记录在案，方便测试人员的回归测试。

另外，除了使用专门的 BUG 管理软件，开发小组还需要使用 CVS 软件做版本管理，并要求开发人员随时都需要上传完整的修改文件，所以可以保证 CVS 上传的软件版本都是可以运行的，开发人员也可以使用 CVS 随时更新本地上的文件，而不用担心更新之后的本地软件系统不能运行。关于 CVS 软件的使用，接下来会进行详细的讲解。

CVS 不仅仅可以作为软件的版本管理仓库，还可以作为文档的管理仓库，需求文档和管理文档都放置在 CVS 上，这样可以非常方便地跟踪各个版本地文档。

在整个的软件开发过程中，仍然会出现部分的需求变更，出现这样的情况有两个原因：第一，出现部分的需求变更并不表示之前的需求分析错误，而可能是因为用户对系统的需求突然发生了变化，这也是常常发生的事情；第二，任何事情都不能是完全完美和正确的，所以在需求分析过程难免会出现小的错误，需求分析做的比较好，只是为了使需求变更会变得更少一点。

在完成一个比较稳定的软件版本，并经过严格的测试之后，就可以交付给客户进行初次预览和检查。然后用户对该版本的软件产品提出适当的改进要求（也可以在开发过程中就让用户参与进来），对于小的需求修改就在本版本中完成，大的需求改动则可以放在下一版本中进行。

27.2.4　交付使用和维护阶段

完成了 Beta 版之后，测试人员会对 Beta 版本进行最后全面的测试。这时的 BUG 已经非常少了，测试完成之后，就可以把它交付给用户正是使用了。到此，整个开发过程就可以算完成了。

进入维护阶段之后，大部分开发人员就可以撤出开发小组，而只保留 1 个或者两个人员进行维护性的工作，针对用户需求，每隔一段时候（一般为一个月）递交一次更新版本。

在开发过程中，不是所有的需求变更都是能够顺利地进行修改，有些需求变化会打乱项目地进度计划。如果软件的设计不合理，导致软件的可扩展性很差，从而有可能会导致无法针对用户的需求进行相应的变更。

所以在软件开发过程中，要尽量考虑到各种可能情况的发生。在和用户商定最终交货时期时应该尽量争取到更多的开发时间。而在内部指定开发计划时，确应该考虑到提前完成，这样做到外松内紧，以防止不可预测的突发事件，最终能够保证按时完成既定的项目。

关于软件工程方面的知识已经超出了本书的内容，有兴趣的读者可以参考相关的书籍。总之，在整个开发过程中，要注意团体合作和交流的作用性，加上规范的需求分析和测试，做到文档管理，这些都是成熟软件开发所必不可少的。

整个软件开发过程可以由图 27.1 所示。

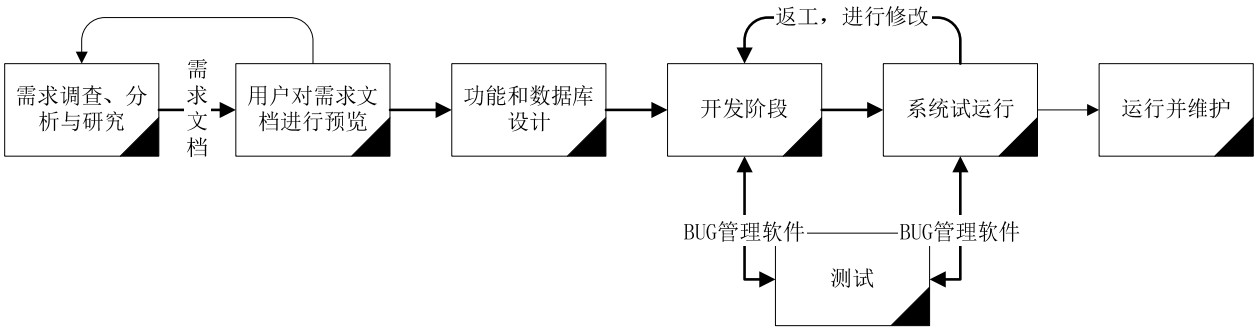

图 27.1　软件开发过程

27.3　CVS安装和配置

CVS 的全称为 Concurrent Version System（并发版本系统），它是一个开源的项目产品，是当前最为流行的版本控制系统，并且在 Eclipse 中已经集成了 CVS 客户端。

27.3.1　CVS服务器端的安装和配置

使用 CVS 之前，需要安装和配置 CVS 服务器，具体过程如下：

（1）在 cvsnt 主页 http://www.cvsnt.com/downloads 上下载 CVS 服务器安装程序，下载之后的文件名为 cvsnt-2.5.03.2151.msi，可以安装在 Windows NT/2000/XP/2003 上。

（2）直接双击下载的安装文件，在安装过程选择默认设置即可，cvsnt 默认的安装路径为“C:\Program Files\CVSNT”。

（3）选择“开始”|“所有程序”|“CVSNT”|“Service control panel”命令，弹出 cvsnt 的设置面板，如图 27.2 所示，可以单击“Start”或者“Stop”按钮进行相应的服务启动或者关闭的操作。

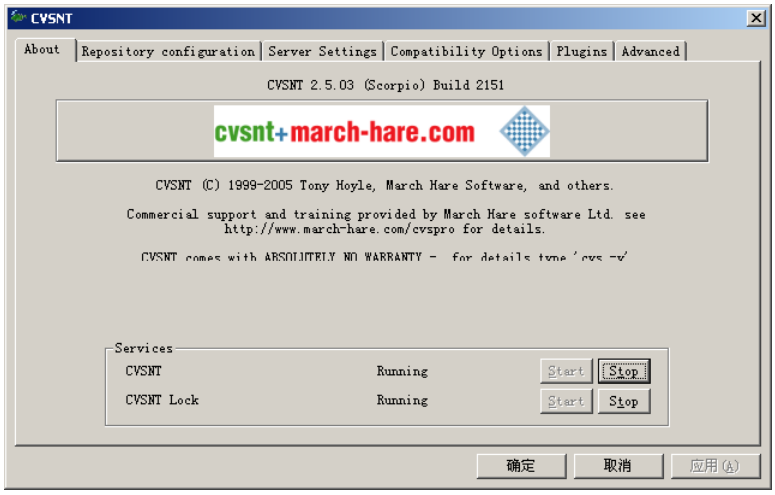

图 27.2　设置面板

（4）选择“Repositories”选项卡，单击“Add”按钮，创建一个存放版本文件的目录，本书选择的目录为“c:/cvsfile”。然后单击“OK”按钮，即完成创建，如图 27.3 所示。

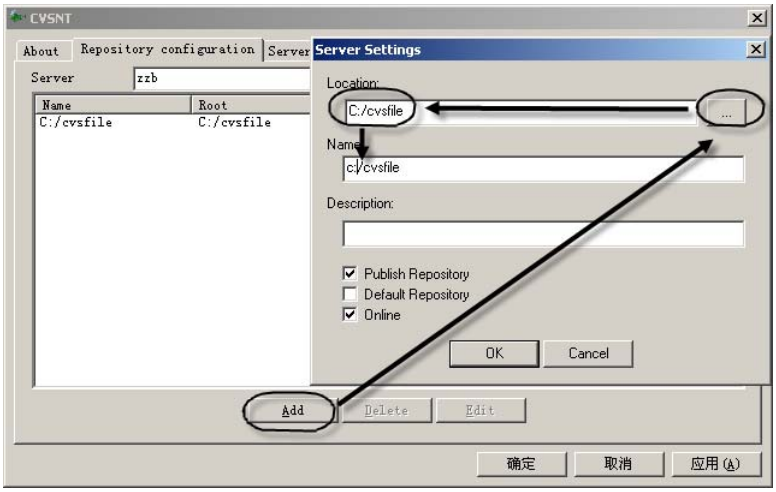

图 27.3　添加存放版本文件的目录

注意：把“Name”输入框中自动生成的“/cvsfile”修改成绝对路径“c:/cvsfile”，保持和 Location 名称一致，一是为了好记忆，第二有可能会导致 cvsnt 安装之后不能正常使用。

（5）选择“Compatibility Options”选项卡，按照图 27.4 所示，把标注的选项全部选上。

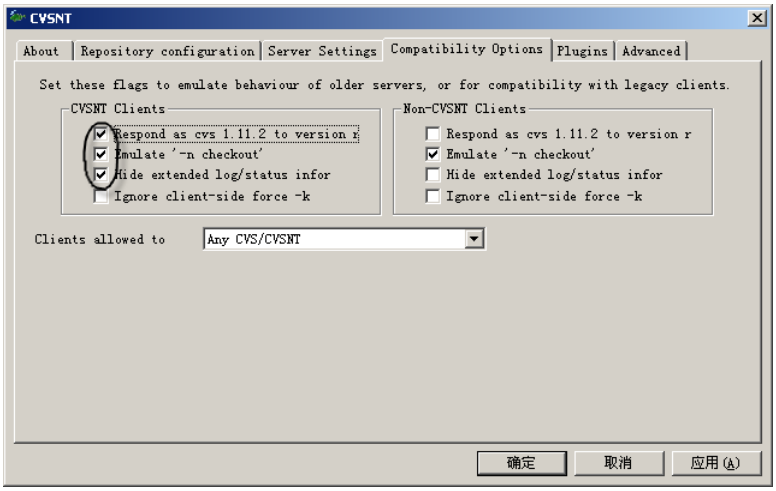

图 27.4　“Compatibility Options”选项卡

注意：如果你使用的 Eclipse 版本是 3.0.1 以下，则这一步骤非常重要，否则将无法在 Eclipse 中使用 CVS。

（6）在客户端连接 CVS 服务端之前，还需要为客户端在服务器端上分配一个用户名，并且用户名必须是客户端 Windows 的登录用户名。本书把服务器端和客户端放在了同一个机器上，当然 CVS 服务器端可以安装在一个专门的机器上。本书登录系统的用户名为 zzb，密码为空。选择“开始”|“运行…”命令，然后输入 cmd 进入 Windows 的命令行窗口，再转到 C:\Program Files\CVSNT 目录下，输入命令 cvs –d c:\cvsfile passwd –a zzb，然后回车，在输入用户名为 zzb 的登录密码（此处的密码可以和系统登录密码不相同），这里设置为 123456。

至此，CVS 服务器部分的安装和配置就已经完成了，这时客户端就可以使用 zzb 用户名来登录到 CVS 上。

27.3.2　CVS客户端配置

由于本书介绍的是使用 Eclipse＋Lomboz 来构架 Web 应用，这里介绍怎样在 Eclipse 中配置和使用 CVS 客户端。

27.3.2.1　配置 Eclipse 客户端来连接 CVS 服务器

选择 Eclipse 的“Window”|“Open Perspective”|“other”菜单选项，在弹出的对话框中选择“CVS Repository Exploring”透视图。

右击“CVS Repositories”视图，选择“New”|“Ropository Location…”命令或者直接选择“CVS Repositories”视图上方的“Add CVS Repository”选项，在弹出的对话框中输入 CVS 的相关信息，如图 27.5 所示，然后单击“Finish”按钮。

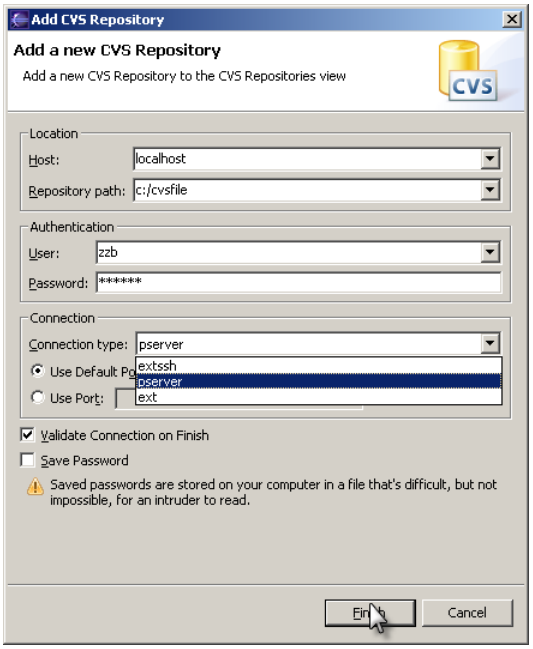

图 27.5　添加 CVS 资料库

注意：**图 27.5 中的**“Repository path”**输入框中内容应该对应着图 27.3 的**“Name”**中内容。**

27.3.2.2　使用 CVS 提交项目

使用 CVS 客户端把开发的项目提交到 CVS 服务器的仓库中的步骤如下：

（1）右击项目名（这里以 MyWebProject 项目为例，这个项目将在下面创建），然后选择“Team”|“Share Project…”命令，弹出如图 27.6 所示的对话框。

（2）选择默认设置，直接单击“Next”按钮，弹出如图 27.7 所示的对话框。

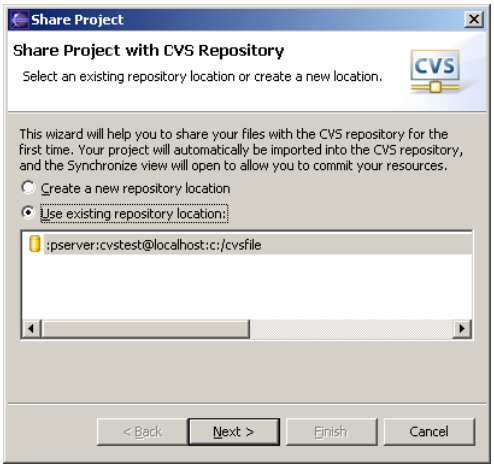

图 27.6　“Share Project”对话框

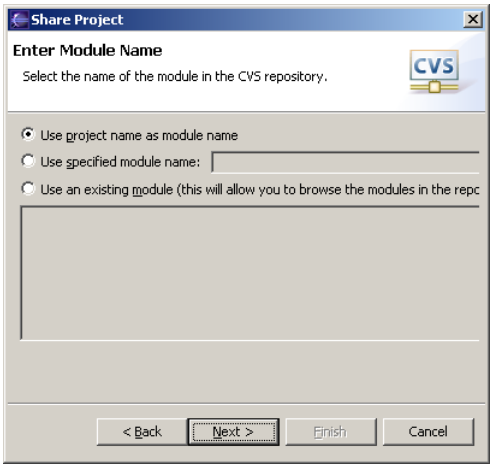

图 27.7　输入模块名称

（3）提示输入模块名称，这里也选择默认选项，单击“Next”按钮，会弹出如图 27.8 所示的对话框。

（4）输入上面设置的密码 123456，直接单击“ok”按钮，弹出如图 27.9 所示的对话框。

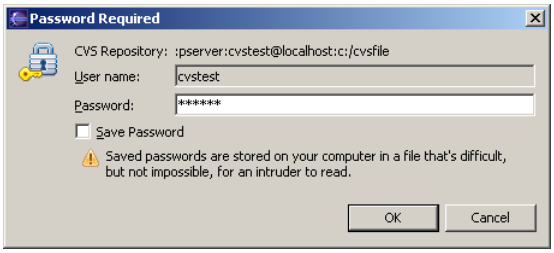

图 27.8　密码输入框

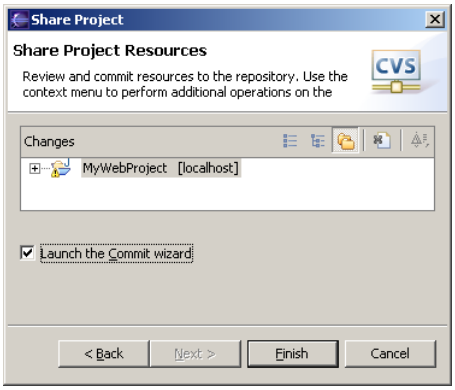

图 27.9　选择需要提交的项目资源

（5）在窗口中选择需要提交到 CVS 上的项目资源文件，这里不必作任何设置，直接单击“Finish”按钮执行资源的提交。

（6）然后执行过程中，可能会连续弹出两个对话框，直接单击“Yes”按钮即可。这样就完成了项目文件的提交。

27.3.2.3　把 CVS 服务器上的项目导入到 Eclipse 中

如果项目小组有新成员加入，这时候就需要将需要把项目资源从 CVS 服务器上导入到自己的 Eclipse 中。导入的步骤如下：

（1）在 Eclipse 主菜单中选择“File”|“Import”命令，弹出如图 27.10 所示的对话框。

（2）选择“Checkout Projects from CVS”选项，单击“Next”按钮，弹出一个设定资源库位置的对话框，直接选择默认设置即可，单击“Next”按钮，弹出如图 27.11 所示的“Select Module”对话框。

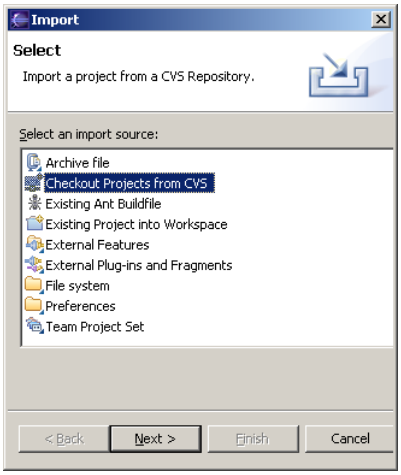

图 27.10　导入选择窗口

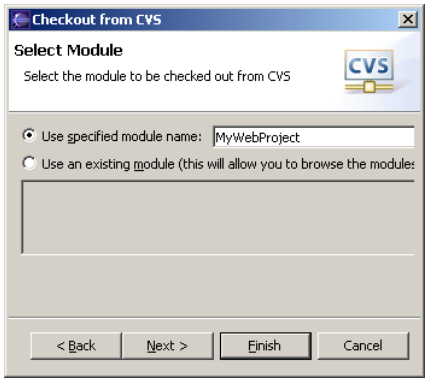

图 27.11　填写 CVS 上的模块名

（3）在输入框中，填上 MyWebProject 模块名（请注意这里是 CVS 服务器上的模块名，因为前面设置了 CVS 服务器上的这个模块名和项目名相同），然后单击“Next”按钮，弹出如图 27.12 所示的对话框。

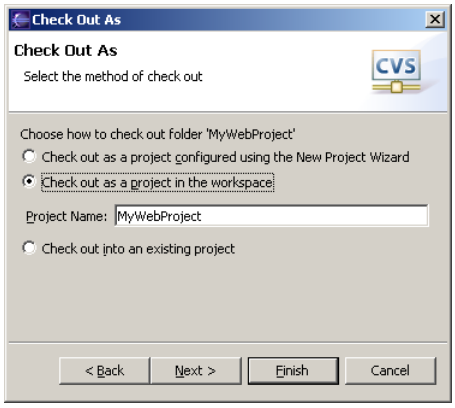

图 27.12　“Check Out As”对话框

（4）选择“Check Out as a Project in the workspace”选项，填上在本地 Eclipse 生成的项目名称，并把 CVS 项目资源文件导入到这个项目中，这里取的项目名称也是 MyWebProject（当然可以是任意取名）。然后单击“Finish”按钮完成导入工作。

当然关于 CVS 的运用还有很多方面，这里不一一具体介绍了，请读者自己查阅相关的书籍。

27.4　数据库连接

这里将要向读者介绍另一种数据库连接池的配置方法，在介绍之前，先总结一下数据库连接有哪几种方式。

27.4.1　数据库连接方式的总结

之前已经介绍了多种数据库的连接方法，第一种就是不使用数据库连接池；第二种使用数据库连接池。而实现方式又分很多种。在第 12 章中就是直接在 JSP 中实现数据库连接。在第 14 章中，使用了 JavaBean 来实现数据库的连接操作。对于数据库连接池的使用也有很多种实现方法：第一可以使用 Tomcat 中的连接池技术，见第七章的配置和实现方法；如果使用 Struts 构架 Web 应用，Struts 也支持了数据库连接池配置；如果使用了 Hibernate 数据库持久层机制，可以在 Hibernate 中实现数据库连接池。

27.4.2　PoolMan第三方数据库连接池

这里将要使用的是第三方提供的连接池包 PoolMan，使用的是 2.0.4 版本，下载地址为 http://telia.dl.sourceforge.net/sourceforge/poolman/。把下载的压缩文件解压，并把解压后 lib 目录下的所有 jar 包文件复制到 MyWebProject\webshop\WEB-INF\lib 目录下（可能只有部分 jar 包使用到，但是为方便复制了所有包），然后把它们引入到 Eclipse 库中。

由于这里使用的是 MySQL 数据库，所以还需要把数据库连接的 mysql-connector-java-3.1.12-bin.jar 包复制到 MyWebProject\webshop\WEB-INF\lib 目录下，同样需要手动将这个包引入到 Eclipse 库中。

使用第三方 poolman 连接池还需要在 WEB-INF\class 目录下配置一个 poolman.xml 文件，该文件设置了连接数据库的各参数。

27.4.1　配置poolman.xml文件

poolman.xml 详细的配置内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<poolman>
  <management-mode>local</management-mode>
  <!-- ========================================================= -->
  <!-- These entries are an example of JDBC Connection pooling.     -->
  <!-- Many of the parameters are optional. Consult the             -->
  <!-- UsersGuide.html doument and the poolman.xml.template file    -->
  <!-- for guidance and element definitions.                        -->
  <!-- ========================================================= -->
  <datasource>
    <!-- ============================== -->
    <!-- Physical Connection Attributes -->
    <!-- ============================== -->
    <!-- Standard JDBC Driver info -->
    <!-- don't do any change with dbname and jndiName -->
    <dbname>qqNews</dbname>
    <jndiName>jndi-qqNews</jndiName>
    <!-- driver is your database driver. Default:com.mysql.jdbc.Driver -->
    <driver>com.mysql.jdbc.Driver</driver>
    <!--url is the database connection URL used by lybbs. You can change IP and database name to fit your
config-->
    <url>jdbc:mysql://localhost/webshop</url>
    <!-- username is your username connected to database. Default:lybbs -->
    <username>root</username>

    <!-- password is your password connected to database. Default:lybbs -->
    <password>12345</password>

    <!-- emailIfConnectionFailed is the email viewed on web for contact when Database connection failed. -->
    <emailIfConnectionFailed>FrankJohnson@163.com</emailIfConnectionFailed>

    <minimumSize>5</minimumSize>
    <maximumSize>20</maximumSize>
    <connectionTimeout>600</connectionTimeout>
    <userTimeout>12</userTimeout>
    <shrinkBy>10</shrinkBy>

    <logFile>D:\eclipse\EclipseSource\MyWebProject\webshop\WEB-INF\classes</logFile>
    <debugging>true</debugging>

    <!-- Query Cache Attributes-->
    <cacheEnabled>true</cacheEnabled>
    <cacheSize>20</cacheSize>
    <cacheRefreshInterval>120</cacheRefreshInterval>
  </datasource>
</poolman>
```

代码说明：

- <driver>元素中设置了连接 MySQL 数据库的驱动。
- <url>元素设置了连接数据库的地址。
- <username>和<password>分别设置连接数据库的用户名和相应密码。
- <minimumSize>、<maximumSize>、<connectionTimeout>、<userTimeout>与<shrinkBy>分别设置最小连接数、最大连接数、最大空闲连接数、最大等待连接限制以及扩展的连接数。开发者可根据系统的需求进行配置。

27.4.2　创建数据库连接和操作类DBConnect.java

用于数据库连接和操作的 DBConnect 类源代码如下：

```
package cn.com.zzb.sql;            //类所在的包
import com.codestudio.util.*;
import java.sql.*;
public class DBConnect {
    private Connection    conn;
    private Statement    stmt;
    private ResultSet    rst;
    private static SQLManager sqlman = SQLManager.getInstance();
                                        //创建数据库的一个 Connection 连接
    public void init(){
        try{
            conn = sqlman.requestConnection("qqNews");        //为 poolman.xml 配置文件中的 JNDI 名称
        }
       catch(Exception e){
            e.printStackTrace();
        }
    }
                                        //构造函数
    public DBConnect(){
        try{
            if(sqlman == null) {
                sqlman = SQLManager.getInstance();
            }
            init();                     //获得一个数据库连接
            stmt = conn.createStatement();
        }
        catch(Exception e)   {e.printStackTrace();}
    }
                                        //执行数据库查询语句，s 为 sql 语句，把查询结果赋给 ResultSet
    public void excuteQuery(String s) {
        try{
            if(stmt != null) {
                rst = stmt.executeQuery(s);
            }
        }
        catch(Exception e) {
            e.printStackTrace();
```

```
        }
    }
                                        //对数据库进行 update 操作
    public int excuteUpdate(String s) {
        int status = 0;
        try{
            if(stmt != null)
                status = stmt.executeUpdate(s);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
        return status;
    }
                                        //以下为赋值方法
    public void setString(int i,String s) {
       try { pstmt.setString(i,s); }
       catch(Exception e) { e.printStackTrace(); }
    }
   public void setBoolean(int i, boolean flag) {
        try{ pstmt.setBoolean(i, flag);}
        catch(Exception e) { e.printStackTrace();}
   }
   public void setDate(int i, Date date) {
         try{pstmt.setDate(i, date);}
         catch(Exception e) { e.printStackTrace();}
   }
   public void setTime(int i, Time time) {
         try{pstmt.setTime(i,time);}
         catch(Exception e) { e.printStackTrace();}
   }
   public void setShort(int i, short word0) {
         try{pstmt.setShort(i,word0);}
         catch(Exception e) {e.printStackTrace();}
   }
   public void setInt(int i, int j) {
         try{pstmt.setInt(i,j);}
         catch(Exception e) { e.printStackTrace();}
   }
   public void setLong(int i, long l) {
         try{pstmt.setLong(i, l);}
         catch(Exception e) { e.printStackTrace();}
   }
   public void setFloat(int i, float f) {
         try{pstmt.setFloat(i, f);}
         catch(Exception e) { e.printStackTrace();}
   }
   public void setDouble(int i, double d) {
         try{pstmt.setDouble(i, d);}
         catch(Exception e) {e.printStackTrace();}
```

```
}
                                        //以下为取值方法
public boolean getBoolean(int i)    throws Exception{
    return rst.getBoolean(i);
}
public boolean getBoolean(String s) throws Exception{
    return rst.getBoolean(s);
}
public Date getDate(int i) throws Exception{
    return rst.getDate(i);
}
public Date getDate(String s) throws Exception{
    return rst.getDate(s);
}
public Time getTime(int i) throws Exception{
    return rst.getTime(i);
}
public Time getTime(String s) throws Exception{
    return rst.getTime(s);
}
public double getDouble(int i) throws Exception{
    return rst.getDouble(i);
}
public double getDouble(String s) throws Exception{
    return rst.getDouble(s);
}
public float getFloat(int i) throws Exception{
    return rst.getFloat(i);
}
public float getFloat(String s) throws Exception{
    return rst.getFloat(s);
}
public int getInt(int i) throws Exception{
    return rst.getInt(i);
}
public int getInt(String s) throws Exception{
    return rst.getInt(s);
}
public long getLong(int i) throws Exception{
    return rst.getLong(i);
}
public long getLong(String s) throws Exception{
    return rst.getLong(s);
}
public short getShort(int i) throws Exception{
    return rst.getShort(i);
}
public short getShort(String s) throws Exception{
    return rst.getShort(s);
}
```

```
    public String getString(int i)   throws Exception{
        return rst.getString(i);
    }
    public String getString(String s)   throws Exception {
        return rst.getString(s);
    }
                                        //指针下移一位
     public boolean next(){
            Try {return rst.next();}
            catch(Exception e) {
                  e.printStackTrace();
                  return false;
            }
     }
                                        //释放内容
     public void close(){
            try{
                  if(conn !=null)   conn.close();
                  if(stmt !=null)   stmt.close();
                  if(rst != null)   rst.close();
            }
            catch(Exception e)   {e.printStackTrace();}
     }
}
```

代码说明：

（1）首先初始化 SQLManager 类，并调用其中的 requestConnection(String JDNIName)方法，来读取到 poolman.xml 配置文件中的参数设置，然后创建一个数据库连接池。

（2）本书推荐使用第三方连接池或者在 Hibernate 中设置连接池，如果直接使用容器提供的连接池技术，则每次 Web 应用移植都需要重新配置一次容器的连接池。

27.5　本章小结

本章对 Web 开发的基本形式和方式、数据库连接方式进行了总结，并介绍了软件开发的基本过程。另外还介绍了 CVS 版本控制工具的使用、PoolMan 第三方数据库连接池配置方法。在下面章节介绍的综合实例中将使用 PoolMan 配置的数据库连接池。