第一部分

Hibernate快速入门

我们的第一个目标就是尽快地了解Hibernate的最新进展。这一部分的大多数章是基于《Hibernate: A Developer's Notebook》（O'Reilly）的内容并进行了更新，反映了Hibernate 3带来的主要变化。示例代码现在使用的都是一些开发工具的最新版本，我们通过它们来提供一个方便而且实用的Hibernate开发环境。还有一章专门介绍Java 5的标注功能，使用标注，而不是用XML映射文件，也可以配置Hibernate映射。

注意：当然，和其他任何关于活跃的开源项目的图书一样，图书介绍的内容总赶不上开源项目的发展！附录E列出了本书讨论的工具的特定版本以及应对变化的指导思想。

因为我们采用Maven来帮助下载许多工具和库，所以使用这本书来着手学习和实践书中的代码示例会更加容易。因为我们希望你可以明白，没有什么理由不去亲自实践一下这些代码示例。

在熟悉了Hibernate的基础之后，第二部分将演示如何将Hibernate绑定到其他组件环境，让各种组件互相配合，以发挥更大的作用。

万变不离其宗，现在就开始学习吧。

第1章 安装和设置

我一直很惊讶，竟然会有这么多免费而又好用的开源Java工具。多年前，我开发一个JSP的电子商务项目时，需要一个轻量级对象/关系数据库映射服务，那时还没有Hibernate这样的工具，只能自己构建了一个这样的组件。这个组件经过几年的发展，开发出一些很酷、很独特的功能。但是在我发现了Hibernate以后，我想在下一个项目中，就不会再继续使用自己熟悉的那个系统了（我当然对自己的系统抱有偏爱），而是会使用Hibernate。用过之后，你一定会知道Hibernate有多棒！

正在读这本书的你，一定急于想知道这种功能强大而且使用方便的技术，是如何架起连接Java对象和关系数据库这两个世界之间的桥梁的！Hibernate很好地充当了这个角色，它并不很复杂，所以学习起来也不困难。为了展示这一点，本章将要指导你理解Hibernate的用法，让你看看为什么Hibernate会这么令人激动。

之后的章节将介绍在更复杂环境（例如Spring和Stripes）下，把Hibernate作为它们的组成部分的应用，以及它和其他数据库的配合使用。第1章的目标是要向你展示，使用Hibernate构建一个基本的、自我包含的环境，并且用它完成真正的操作是多么容易的。

获得Ant发布版本

可能有些令人意外，在运行Hibernate之前需要做几件与Hibernate本身无关的事。首先，你必须搭建一个开发环境，以供示例代码可以运行。也可以为你可能构建的任何实际项目奠定坚实的基础，这将是令人高兴的意外收获。

如果在你的Java项目中，还没有使用Ant去管理构建（build）、测试（test）、运行（run）以及打包（package）的工作，现在就是开始使用Ant的好时机。本书的示例都是Ant驱动的，所以，你得安装一个能用的Ant才能运行示例代码，并验证在系统中对代码做出的修改，这才是最佳的学习方式。

首先，获得Ant的二进制发布版本，并安装它。

为何在意

我们选择使用Apache Ant来处理示例有几个原因。Ant很方便，而且功能强大，它已经成为基于Java开发的标准构建工具，而且是免费、跨平台的工具。如果使用Ant，我们的示例将可以在任何Java环境中一样地正常运行；也就是说，本书的任何读者都不会因为运行示例而遇到麻烦。这也意味着，我们可以少花一点力气就能够做很多很酷的事情，尤其是几个Hibernate工具特意支持Ant之后，更是如此。我们会教你如何利用这些工具（值得注意的是，最近更复杂的Java项目经常使用的是Maven（注1），它增加了很多其他的项目管理功能。所以我必须从二者中挑选一个，本着尽可能简单和实用的原则，我就决定继续使用Ant来管理这些示例）。

如果你目前正在使用Maven作为代码构建工具，你会注意到我们使用Maven的Ant任务（Task）来管理Ant构建的依赖关系。虽然Maven的发展势头强劲，但Ant仍然是目前Java开发中使用最广泛的构建工具。每一章的示例代码文件夹中也包含一个Maven的pom.xml文件，可以用Maven进行编译。在许多情况下，使用Maven Hibernate3插件，Maven构建文件提供的功能与Ant的build.xml文件一样。第12章介绍了如何用完整的Maven来构建和部署Hibernate应用程序的方法，但本书大部分示例仍旧使用Ant作为构建工具，同时使用Maven Ant Task来查找和下载需要的各种库文件，包括库文件之间互相信赖的文件。

为了能够使用这些功能，需要做的第一件事就是先安装Ant，让它可以正常运行起来。

注意：我以前觉得奇怪，可以用Make，为什么还要用Ant？现在，我已经明白用Ant来管理Java的代码构建有多么美妙，没有Ant还真的不行。

应该怎么做

Ant的二进制发布包可以在http://ant.apache.org/bindownload.cgi下载。滚动网页，找到Ant的当前最新版本，然后下载适合的压缩文件格式。选择一个适合存放的位置保存文件，然后解压。压缩文件展开的目录就是ANT_HOME。假如你把压缩文件解压到目录/usr/local/apache-ant-1.7.0，你可能会想创建一个符号链接（symbolic link）以方便使用，同时当你升级到新版本时，也可以免去更新环境配置的麻烦：

```
/usr/local % ln -s apache-ant-1.7.0 ant
```

安装好Ant之后，需要做一些设置才能让它正常工作。你得将Ant的bin目录（在这个例子中，就是/usr/local/ant/bin）添加到命令路径中。还需要设置环境变量ANT_HOME，将其设定为安装Ant的最顶级目录（在这个例子中，就是/usr/local/ant）。至于如何在不同的操作系统中执行以上这些处理步骤，如果需要的话，可以参阅Ant的手册（http://ant.apache.org/manual/）。

注1： http://maven.apache.org/.

检查Java版本

当然，我们假定你已经安装好了Java software development kit (SDK)。目前你应该使用Java 5或更新的版本，因为新版本的SDK会提供一些有用的新功能。尽可能使用最新稳定版本的SDK，Java 5或Java 6都可以支持本书的所有示例。用Java 1.3也可以使用Hibernate2的大部分功能，但你得用1.3版本的Java编译器重新构建Hibernate JAR文件。发布版本越新，对Java版本的要求就越高；而且Java 5已经发布很长时间了，它本身就提供了很多优点，所以我们没有必要为兼容早期的JDK而花费时间。我们的示例都假定你用的至少是Java 5，如果使用更低版本的JDK，那么就得做大量修改调整。运行以下命令可以查看JDK版本：

```
% java -version
java version "1.6.0_02"
Java(TM) SE Runtime Environment (build 1.6.0_02-b06)
Java HotSpot(TM) Client VM (build 1.6.0_02-b06, mixed mode, sharing)
```

你也应该使用官方发布的Java版本（例如Sun或Apple发布的版本）。在编写本书时，我们的技术审阅者发现GNU公共授权的“功能类似”的Java实现并不能正确运行这些工具和示例代码。许多Linux发布版本安装的默认Java环境就是GNU的。如果你正在使用Linux，可能需要自己下载Sun的JDK，并确保使用的是正确的版本（通过运行java -version命令）。既然Sun已经开放了Java的源代码，希望将来这种情况会得到改善，到时候可能在任何自由软件版本中都会自带Sun JRE和JDK。不过，在那一天实现以前，你必须自己下载。

在编写本书时，基于Debian的发布版本可以用它们的安装管理工具来安装Sun JDK(Ubuntu的“Feisty Fawn”和“Gutsy Gibbon”发行版本就自带了JDK 5和6)。Red Hat系列的发布版本仍然需要直接从Sun Microsystems的网站下载Java。具体情况具体分析吧。

安装好以后，就应该能够启动Ant进行测试，来确认一切都没有问题：

```
% ant -version
Apache Ant version 1.7.0 compiled on December 13 2006
```

发生了什么事

嗯，目前还不多，不过现在已经可以尝试我们稍后将要提供的示例了，再以这些示例作为起点，去做实际的Hibernate项目。

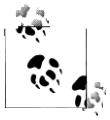
如果你是Ant新手，最好先简单阅读一下它的手册来了解Ant的工作原理和功能。这样可以让你了解示例中用到的build.xml文件是怎么回事。如果你开始或已经喜欢上了Ant，想深入研究，那么你可以仔细阅读它的手册（注2），或阅读O’Reilly的《Ant: The Definitive Guide》

注2： http://ant.apache.org/manual/.

（当然，应该先把这本书看完）。

其他

Eclipse（注3）、JBuilder（注4）、NetBeans（注5），还是其他Java IDE？嗯，你当然可以使用这些IDE，但是怎么把Ant整合到IDE的构建过程中，就是你自己的事了（有好几种IDE已经支持Ant，所以你可能已经走在前面了；对于其他IDE，你可能还需要跨越学习的障碍）。如果都行不通，你还可以使用IDE开发自己的程序代码，然后使用我们提供的一个build脚本，从命令行来调用Ant。



如果你使用的是Maven，则可以通过在任意一章的示例目录或最顶级的examples目录中执行mvn eclipse:eclipse，来生成Eclipse IDE项目文件。如果在examples目录中运行mvn eclipse:eclipse，Maven将为每一章的示例生成一个Eclipse项目。第12章将详细介绍如何用Maven来构建示例代码，第11章将详细介绍Hibernate的Eclipse工具的用法。

获得Maven Tasks for Ant

稍等一下，难道我还没有讲完用Ant构建本书示例项目的用法？确实还没有说完，不过，这也并不是全部。虽然本书以Ant作为示例构建的基础，我们决定本书应该通过Maven Tasks for Ant来演示一下Maven优秀的依赖（dependency）管理功能。本书的最初版本在这一部分只用了几页的篇幅来介绍如何下载和管理一系列第三方库：包括从Jakarta Commons Lang到CGLIB。如果这些工作由你亲自来做，最少也得花费你宝贵的几分钟的时间，而且说明如何操作和操作本身费事又繁琐。本书中，我们在build.xml文件中声明了项目需要依赖的库文件，并由Maven负责下载和管理这些依赖文件。这样就节省了许多步骤和时间。好，现在就开始安装Maven Tasks for Ant。

应该怎么做

有两种方法来整合Maven Tasks for Ant：第一种方法是将需要的JAR文件放在Ant的lib目录，第二种方法就是在Ant的构建（build）文件中用typedef声明来包括antlib定义。我们打算使用前一种方法，将maven-ant-tasks-2.0.8.jar文件添加到Ant的lib目录中，因为这样对示例的build.xml文件修改量最少。首先，从Maven的网站（注6）下载需要的JAR文件，在它的首页上应该可以看到用于下载Maven Tasks for Ant的链接（如图1-1所示）。

注3： http://www.eclipse.org/.
注4： http://www.borland.com/jbuilder/.
注5： http://www.netbeans.org/.
注6： http://maven.apache.org/.

图1-1：Maven网站上Maven Tasks for Ant的下载链接

在编写本书时，Maven Tasks for Ant的版本是2.0.8。点击Maven Tasks for Ant 2.0.8的链接，选择一个镜像，就可以下载到一个名为maven-ant-tasks-2.0.8.jar的文件。将该文件保存到本地目录。

安装Maven Tasks for Ant

接下来，将下载好的maven-ant-tasks-2.0.8.jar文件复制到ANT_HOME/lib目录中。如果你从头至尾按本章的说明来操作，现在应该已经下载并安装好了Ant。还应该设置一个名为ANT_HOME的环境变量，当然，你也要了解Ant的安装目录是什么。在将maven-ant-tasks-2.0.8.jar复制到ANT_HOME/lib 目录以后，任何build.xml 文件都可以包含相应的命名空间（namespace），以使用Maven Tasks for Ant。

如果你运行示例使用的计算机上有多个用户，而且你没有将JAR文件放到ANT_HOME/lib目录的管理权限，不必担心，你也可以将maven-ant-tasks-2.0.8.jar文件放在~/.ant/lib这个目录中。Ant也会自动在这个目录中查找任何JAR文件。

在将maven-ant-tasks-2.0.8.jar文件复制到ANT_HOME/lib目录以后，你应该能够运行以下命令，以检查当前UNIX的类路径（class path）中是否包含了maven-ant-tasks-2.0.8.jar：

```
% ant -diagnostics | grep maven | grep bytes
maven-ant-tasks-2.0.8.jar (960232 bytes)
```

在Windows中，运行ant -diagnostics，并检查输出的类路径包括的类库列表中是否包含了maven-ant-tasks-2.0.8.jar。

使用HSQLDB数据库引擎

Hibernate支持非常多的关系数据库，可能对于下一个项目中你打算使用的关系数据库，Hibernate就已经提供了支持。我们需要为示例挑选一个数据库，幸运的是，眼前就有一个好的选择。这是一个免费的、纯Java的开源自由软件项目，它功能强大，足以担当我们的商业软件项目中的数据存储支持系统。令人惊讶的是，HSQLDB也相当完善而且安装简单（简单到我们可以让Maven在新版中负责它的安装），所以在此讨论它很恰当（你是否听说过Hypersonic-SQL，现在叫做HSQLDB。Hibernate的很多说明文档还沿用旧的名称）。

如果你偶然访问http://hsql.sourceforge.net/，发现这个项目好像已经停止了，不要惊慌。这是错误的网址，它是HSQLDB项目的前身。图1-2演示了该项目当前的主页，它还相当活跃。

为何在意

本书示例是基于数据库的，每个人都可以下载这些示例，方便地在此基础上做试验，其间不需要转换任何SQL方言（dialect）或操作系统命令，就可以和你的数据库一起使用（也许这意味着你可以节省下一两天的时间，不用去研究怎么下载、安装和配置某种常用数据库环境）。最后，如果你是HSQLDB新手，那么在用过之后感到印象深刻和好奇心十足的概率肯定很高，最终在你自己的项目中会选择使用这种数据库，正如其项目主页上所说的：

> HSQLDB是用Java编写的领先的SQL关系数据库引擎。它提供了一个JDBC驱动程序，支持ANSI-92 SQL的一个丰富子集（BNF 树格式），以及SQL 99和2003增强（enhancement）。它提供了一个小巧（Applet版本的体积小于100kB）而快速的数据库引擎，以及基于内存和磁盘的两种数据表，支持嵌入式和服务器模式。此外，还包括了一些工具，例如微型Web服务器、内存查询（in-memory query）和管理工具（能够作为Applet运行），以及很多演示例子。

应该怎么做

当构建本书的示例时，Maven Ant Tasks将自动从位于http://repo1.maven.org/maven2/的Maven仓库（repository）下载HSQLDB JAR(以及其他需要的JAR)。所以，如果你想马上体验一下，可以直接转到1.7节。否则，如果你想下载HSQLDB供自己使用，或者想查阅它的文档、在线论坛或邮件列表，就可以访问它的项目主页http://hsqldb.org/。点击下载当前最

新稳定版本（latest stable version）的链接（在编写本书时，最新的版本是1.8.0.7，如图1-2所示）。这将打开一个典型的SourceForge下载页面，上面列出了当前选中的版本可以提供的下载链接。选择一个合适的镜像站点，就可以开始下载ZIP文件。

注意：去吧，下载HSQLDB。哎呀，它们的个头都很小！

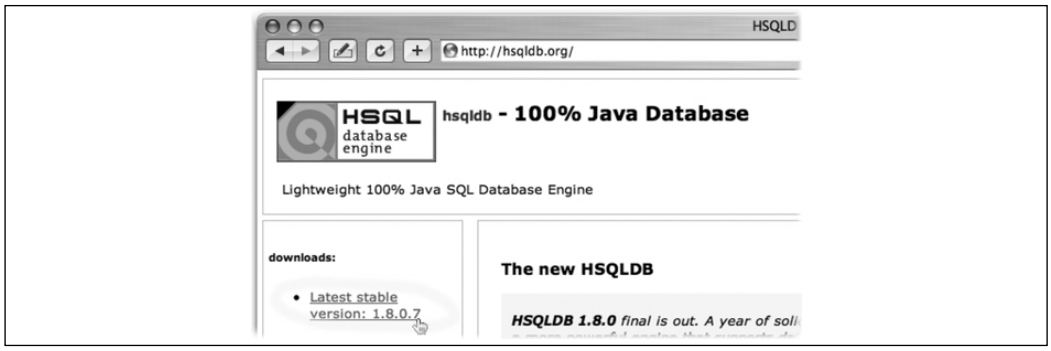


图1-2：HSQLDB主页上最新的稳定版本的链接

其他

Hibernate对其他数据库的支持怎么样呢？不用担心，Hibernate现在可以支持MySQL、PostgreSQL、Oracle、DB2、Sybase、Informix、Apache Derby等各种数据库（本书将在第10章和附录C中教你如何为不同的数据库指定各自的“方言”（dialects））。不过，如果你真的需要，可以试着从一开始就去搞清楚怎么和你最喜欢的数据库打交道。这也意味着你在跟着本书示例走时多花费一些额外工夫，同时也会错过发现HSQLDB美妙之处的大好机会。

获得Hibernate Core

不需要再讲什么激励的话吧！你选择这本书的目的就是想学习如何使用Hibernate。或许并不太令人感到意外，Hibernate中为应用程序提供对象/关系映射服务的核心部分称为Hibernate Core。当构建本书的示例时，Maven会自动为你下载Hibernate和它的所有相关的依赖文件。即便新版的随书代码示例可以通过Maven Ant Tasks来获取Hibernate，你也可以亲自下载最新的Hibernate发布版本，浏览它的源代码，或只是查看在线文档、论坛或其他支持资源。如果你已经准备好了Hibernate，则可以跳过这一节，直接学习1.7节的内容。

应该怎么做

先访问Hibernate的主页http://hibernate.org/，要获得其完整的发布版本，需要找到

“ Download ”链接，如图1-3的左边所示。

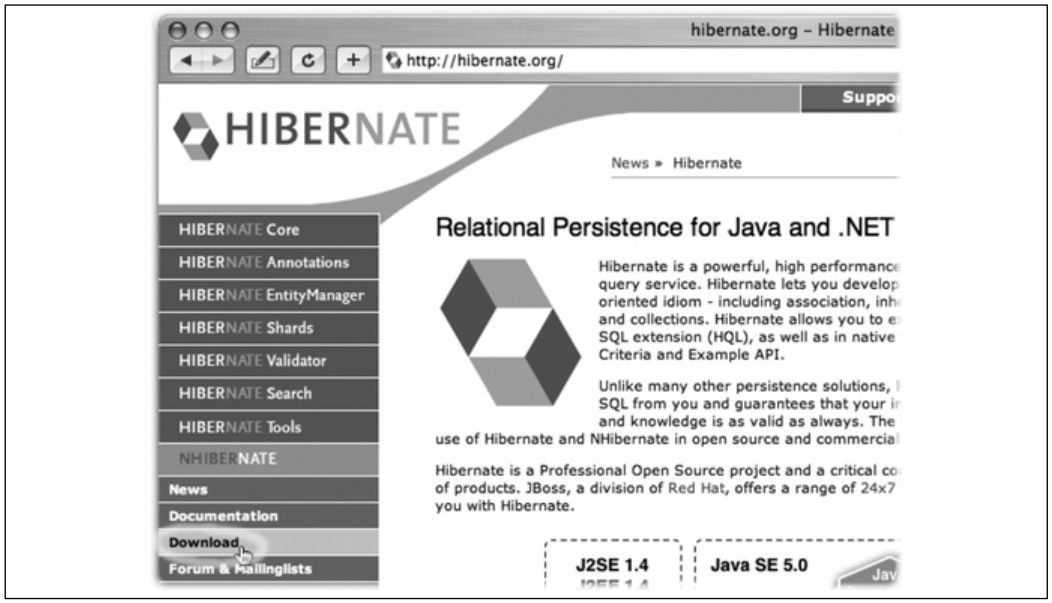


图1-3：Hibernate主页上的Download链接

页面上的“ Binary Releases ”部分将列出Hibernate Core的推荐下载的版本（如果你有足够的勇气，可以尝试下载“ 开发 ”（Development）发布版本，但是最安全的还是支持下载最新的“ 产品 ”（Production）发布版本）。选择好以后，点击表格相应行上的“ Download ”链接（如图1-4所示）。

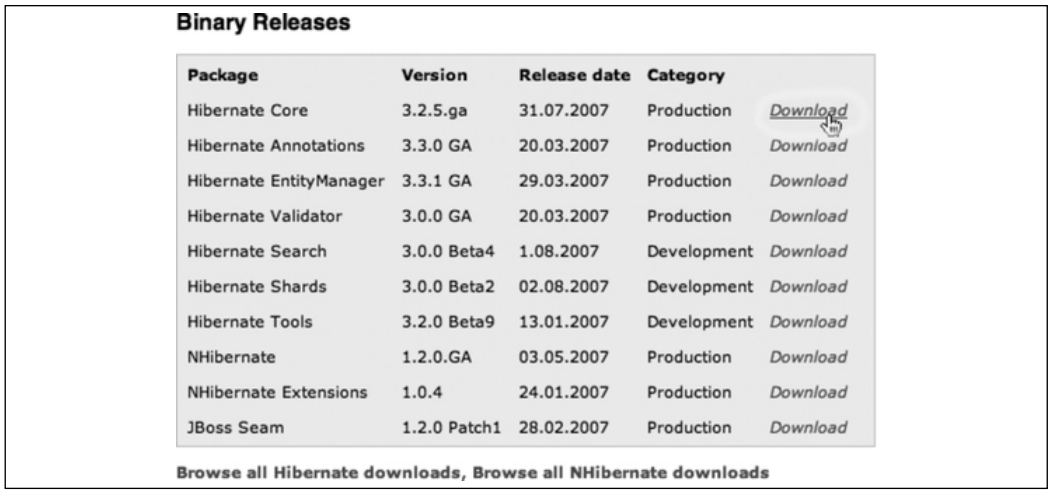


图1-4：Hibernate二进制发布版本

之后会打开一个SourceForge的下载页面，包含了你刚才选择下载的版本，以及可供选择的文档格式。选择对你最方便的压缩文件格式进行下载。下载的文件名看起来就像hibernate-3.x.y.tar.gz 或 hibernate-3.x.y.zip这样（在编写本书时，文件名开始以hibernate-3.2.5.ga命名，因为Hibernate 3.2.5的第一个稳定版本就是当前的产品发布版本）。

选择一个适合的地方将文件解压。

在Hibernate的下载页面，也可以看到“Hibernate Tools”部分（Download链接将打开一个名为“JBoss Tools”的页面，不过仍然可以在上面找到Hibernate Tools）。它们提供了几个有用的功能，这些功能虽然不是使用Hibernate的应用程序所必须的，但是对开发人员创建某些应用程序来说非常有帮助。我们稍后首次对Hibernate进行试验时，将使用其中的一个工具来生成Java代码。这个工具的文件名看起来应该类似于hibernatetools-3.x.y.zip的样子（不一定和Hibernate本身的版本号一样，通常可以使用的只有beta版本；在Hibernate的下载页面中，位于“Binary Releases”下面的“Compatibility Matrix”（兼容性矩阵）表格显示了Hibernate各组件之间的相互兼容关系）。

同样，下载这个文件，将其解压到存放Hibernate的相关目录中。

如果你下载链接时遇到麻烦，可能是因为网站正在维护，处于不稳定的状态，所以就看不到你要的文件。如果真遇到这样的情况，你可以点击“Binary Releases”方框下面的“Browse all Hibernate downloads”链接，滚动页面，查找需要下载的内容。Hibernate项目非常活跃，所以发生这种情况比你想象的要更频繁。

建立项目层次结构

虽然起步只是一小步，但是，当我们开始设计数据结构，并创建用于表示它们的Java类和数据库表（database table），再配合所有配置文件和控制文件，将全部内容整合起来以发挥作用时，我们最后还是会得到一大堆文件。所以，起步时我们得先有个良好的组织体系。从前面下载的工具和相关的支持库之间就可以看出，有许多文件得好好加以组织。幸运的是，Maven Ant Tasks可以帮我们下载和管理所有的外部依赖文件。

为何在意

如果你通过扩展本书的示例而做出了一些很酷的东西，想将它们转换成实用的应用程序，那么你的代码从一开始就得有良好的组织。更重要的是，如果你按照这里所说的方式组织，我们在示例中给你的命令和指令才会有意义，才能起实际作用。书中有很多示例也彼此相关，因此一开始走对路是很重要的。

如果你想跳过这一段去看后面的示例，或者不想输入一些很长的示例程序代码和配置文件，则可以从本书网站下载各章示例的“最终”版本。这些下载到的文件的组织方式的确如这里所述。我们强烈建议你下载示例代码，并将它们作为你阅读本书时的参考。

应该怎么做

以下介绍如何建立一个空的项目层次结构，如果你还没有下载“最终的”示例：

1. 在硬盘目录中选择一个位置，作为演练这些示例的目录，然后创建一个新的目录，从现在开始，我们就称这个目录为你的项目目录。
2. 进入该目录，创建名为src及data的子目录。Java源代码和相关资源的层次结构会放在src目录中。我们的构建过程在编译代码时，会将结果放在编译时创建的classes目录下，然后把运行时需要的任何资源都复制到这个目录中。data目录用于存放HSQLDB数据库相关的文件。
3. 我们打算创建的示例类都将放在com.oreilly.hh（harnessing Hibernate）包中，而将Hibernate生成的数据bean都放在com.oreilly.hh.data包中，以便将它们与我们手工创建的类分离开来，所以我们在src目录下创建这些目录。在Linux 和 Mac OS X上，可以使用以下命令来创建目录：

```
mkdir -p src/com/oreilly/hh/data
```

在项目目录中执行这个命令，一步就可以完成目录的创建。

这时，你的项目目录结构应该如图1-5所示。

注意：与本书第1版相比，这一版的目录结构要简单得多，似乎不值得一提！

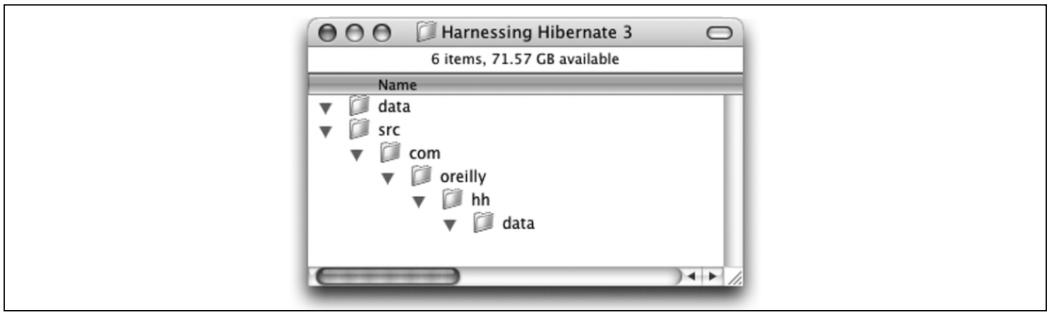


图1-5：最初的项目目录结构

快速测试

在我们真正使用Hibernate来做些有用的工作以前，还应该检查一下其他支持库是否存在和

可以使用。我们先从整个项目都会用到的Ant配置文件开始，告诉Ant我们把要用的文件放在哪里，同时让Ant启动HSQLDB数据库图形界面。这可以证明Maven Ant Tasks能够找到并下载示例依赖的库，可以访问数据库图形界面，通过这个界面我们可以查看Hibernate为我们创建的真实数据。此时只是作为一个基本的完整性检查，以确认没有少什么东西，保证我们为继续学习做好准备。

打开你最喜欢的文本编辑器，在项目目录最顶层创建一个名为build.xml的文件。将例1-1的内容输入到这个文件中。

例1-1：Ant构建（build）文件

```
<?xml version="1.0"?> ❶
<project name="Harnessing Hibernate 3 (Developer's Notebook Second Edition)"
         default="db" basedir="."
         xmlns:artifact="antlib:org.apache.maven.artifact.ant"> ❷

  <!-- Set up properties containing important project directories --> ❸
  <property name="source.root" value="src"/>
  <property name="class.root" value="classes"/>
  <property name="data.dir" value="data"/>

  <artifact:dependencies pathId="dependency.classpath"> ❹
    <dependency groupId="hsqldb" artifactId="hsqldb" version="1.8.0.7"/>
    <dependency groupId="org.hibernate" artifactId="hibernate"
                version="3.2.5.ga">
      <exclusion groupId="javax.transaction" artifactId="jta"/>
    </dependency>
    <dependency groupId="org.hibernate" artifactId="hibernate-tools"
                version="3.2.0.beta9a"/>
    <dependency groupId="org.apache.geronimo.specs"
                artifactId="geronimo-jta_1.1_spec" version="1.1"/>
    <dependency groupId="log4j" artifactId="log4j" version="1.2.14"/>
  </artifact:dependencies>

  <!-- Set up the class path for compilation and execution -->
  <path id="project.class.path"> ❺
      <!-- Include our own classes, of course -->
      <pathelement location="${class.root}" /> ❻
      <!-- Add the dependencies classpath -->
      <path refid="dependency.classpath"/> ❼
  </path>

  <target name="db" description="Runs HSQLDB database management UI
against the database file--use when application is not running"> ❽
       <java classname="org.hsqldb.util.DatabaseManager"
             fork="yes">
          <classpath refid="project.class.path"/>
          <arg value="-driver"/>
          <arg value="org.hsqldb.jdbcDriver"/>
          <arg value="-url"/>
          <arg value="jdbc:hsqldb:${data.dir}/music"/>
```

```
            <arg value="-user"/>
            <arg value="sa"/>
        </java>
    </target>
</project>
```

输入时要小心那些标点符号，对于自结束的（self-closing）XML标签更要小心（是“ />”，而不是“ >”）。如果输入错误，代价就是运行Ant时得到解析错误的信息。如果你不想手工输入这些文件的内容，可以从本书网站下载这些文件。如果你正在阅读的是PDF电子文档，也可以剪切和粘贴代码，但需要去掉一些无关的排版字符。

如果你以前没有看过Ant构建文件，以下简单介绍可以帮助你熟悉它。（如果你想查看更多的细节，可以访问文档http://ant.apache.org/manual/index.html。）

❶ 第1行只是声明这个文件的类型是XML。如果你以前在其他情况下用过XML，就应该很习惯这种文件格式；如果没有，那就再看一次（Ant目前不需要这一行，但大多数XML解析器都需要，因此养成这种习惯是一件好事）。

❷ Ant的构建文件总包含一个project定义。default属性用于告诉Ant，如果在命令行没有指定要构建任何目标（target），Ant要默认构建哪个目标（后面会定义）。basedir属性用于指定所有路径计算都对应于哪个目录。我们可以不指定这个属性，因为在默认情况下总是将build.xml所在的目录作为相对目录的基本目录，但是明确指定类似的基本设置是一种好习惯。在这个project元素中需要注意的一个重要事情是，xmlns:artifact是专门为Maven Ant Tasks提供的命名空间定义。该命名空间定义使用了artifact前缀，这样就可以在构建文件中使用Maven Ant Tasks了（后面有具体的使用方法）。

❸ 接下来的几行定义了3个属性，我们可以在构建文件的其他部分中，通过名称来引用相应的属性。基本上，我们在此是为项目中各个部分用到的重要目录定义其符号名称。这样做并非必要（尤其是目录名称非常简单时），不过这也是一种好习惯。至少，如果你得修改这些目录中的某个目录时，只需要修改构建文件中的一个地方，而不用进行繁琐的搜索和替换操作。

❹ artifact:dependencies元素来源于Maven Ant Tasks，你（以及Ant）可以通过artifact:前缀判断出这一点。在这个元素中，我们定义了一套依赖，项目需要它们才可以编译和执行。这些依赖对应于Maven 2 Repository（位于http://repol.maven.org/maven2）中央仓库中的JAR文件（或其他生成文件）。每个artifact（工件）由一个groupId、artifactId以及version号进行惟一标识。在这个项目中，我们需要依靠Hibernate、HSQLDB、Log4J以及JTA API。当Maven Ant Tasks遇到这些依赖声明时，就会从Maven 2中央仓库中按照需要将每个artifact下载到你的本地Maven 2仓库中（在~/.m2/repository目录下）。如果你对这一区段的配置内容还没有什么感觉，大可不必担心，在稍后几页的内容中，我们将深入探索相关的细节。

如果需要，你可以对这一区段的配置作出修改（只要修改version值），以便使用这些依赖的程序包的更新版本（因为在本书交付印刷之后，可能会有新版本推出）。不过你可以放心，本书印刷出版后，书中的例子将一定能够正常运行，因为不论你什么时候研究这些例子，Maven仓库可以确保我们测试过的版本总是可用的。我们之所以在本书中采用Maven Ant Tasks，这也是很大的一部分原因。

❺ class-path区段的用途很明确。这个功能正是我每次做Java项目时，几乎总要为其配置一个简单的Ant构建文件的原因。无论做什么规模的项目，总是需要将很多第三方程序库放到类路径中，而且还得确保编译时和运行时的配置都完全一样。Ant使得这一工作变得很简单。我们定义了一个path，有点像一个属性，但是它知道应该如何解析、收集文件和目录信息。我们的类路径包含classes目录，经过编译的Java文件就放在这个目录中（这个目录现在还不存在；下一章会在构建处理中多增加一个步骤以创建它），此外，也包含与artifact:dependencies元素中列出的依赖相对应的所有JAR文件。这正是编译和运行项目所需要的一切。

对于Java路径和类层次结构的理解和处理来说，Ant是一个很棒的工具，值得我们深入学习。

❻ 这一行的标点符号有点令人迷惑，但实际上可以划分成具有意义的几个部分。Ant可以使用置换（substitution）机制，将变量值插入到规则中。当你看到像“${class.root}”这样的语句时，这表示“寻找名为class.root的属性值，用这个值来置换这里的字符串”。所以，根据前面对class.root的定义，置换的结果就好像是在这里输入了：<pathelement location="classes"/>。那么，为什么要这么做呢？这是为了可以在整个构建文件中共享某个属性值，这样，如果需要修改相关的配置，则只需要关注一个地方就可以了。在大型的复杂项目中，这种组织和管理是很重要的。

❼ 我们在前面看到的artifact:dependencies元素使用它的pathId属性，将所有声明的依赖组装到一个名为dependency.classpath的路径中。这里，我们将dependency.classpath的内容附加到project.class.path中，以便在编译和运行时可以找到我们用Maven取回的依赖包。

❽ 最后，这些前期工作做好以后，我们就能够定义第一个构建目标了。一个目标其实就是一系列任务，为了完成项目的目标，必须按顺序执行任务。典型的构建目标就是做些类似于编译代码、运行测试、为发布而打包文件等各种事情。可以从Ant内建的一组丰富的功能中选择任务，而像Hibernate这样的第三方工具则可以扩展Ant，以便提供它们自己的任务，这在下一章就会看到。我们的第一个构建目标是db，它会运行HSQLDB的图形界面，让我们查看示例数据库。我们可以用Ant内建的java任务来完成这项工作，它会为我们运行Java虚拟机，并配置好我们需要的启动类、参数（argument）以及属性。

就此例而言，我们想要调用的类是org.hsqldb.util.DatabaseManager，在HSQLDB JAR中可以找到这个类（Maven Ant Tasks将为我们管理这个依赖）。将fork属性设置为“yes”，这是告诉Ant使用另一个单独的虚拟机，而不是默认的虚拟机，因为默认虚拟机得花费比较长的时间，而且通常没有这个必要。在这个例子中，这一点很重要，因为我们想让数据库管理器GUI一直保持运行，直到我们关掉它，但是，如果在Ant自己的VM中运行，就达不到这样的效果。

如果你的数据库GUI弹出后，就马上消失，请再次检查你的java任务的fork属性。

你可以看到我们先是如何告诉java任务，有关之前已经配置好的类路径信息的（这是我们所有的构建目标都得使用的配置属性）。然后，我们为数据库管理器提供了很多参数，告诉它使用标准的HSQLDB JDBC驱动程序，到哪儿去找数据库，以及使用的用户名是什么。我们在data目录中指定了一个名为music的数据库。这个目录当前还是空的，所以HSQLDB会在我们第一次使用时创建这个数据库。新数据库默认的“系统管理员”用户名是sa，最初的配置是一开始不需要密码。显然，如果你计划让数据库在网络上也可以使用（HSQLDB能够做得到），就需要设置密码。我们不需要做这些花哨的事，所以现在先不用理会其他配置。

OK，让我们试一下吧！保存好这个文件，在你的顶级项目目录（build.xml所在的目录）的命令提示行中输入以下命令：

```
ant db
```

（或者，因为我们将db设置为默认的构建目标，你也可以只输入ant）在Ant开始运行以后，如果一切顺利，你会看到像下面这样的输出结果：

```
Buildfile: build.xml
Downloading: hsqldb/hsqldb/1.8.0.7/hsqldb-1.8.0.7.pom
Transferring 0K
Downloading: org/hibernate/hibernate/3.2.5.ga/hibernate-3.2.5.ga.pom
Transferring 3K
Downloading: net/sf/ehcache/ehcache/1.2.3/ehcache-1.2.3.pom
Transferring 19K
Downloading: commons-logging/commons-logging/1.0.4/commons-logging-1.0.4.pom
Transferring 5K
Downloading: commons-collections/commons-collections/2.1/commons-collections-
2.1.pom
Transferring 3K
Downloading: asm/asm-attrs/1.5.3/asm-attrs-1.5.3.pom
Transferring 0K
Downloading: dom4j/dom4j/1.6.1/dom4j-1.6.1.pom
Transferring 6K
```

```
Downloading: antlr/antlr/2.7.6/antlr-2.7.6.pom
Transferring 0K
Downloading: cglib/cglib/2.1_3/cglib-2.1_3.pom
Transferring 0K
Downloading: asm/asm/1.5.3/asm-1.5.3.pom
Transferring 0K
Downloading: commons-collections/commons-collections/2.1.1/commons-collections-
2.1.1.pom
Transferring 0K
Downloading: org/hibernate/hibernate-tools/3.2.0.beta9a/hibernate-tools-
3.2.0.beta9a.pom
Transferring 1K
Downloading: org/hibernate/hibernate/3.2.0.cr5/hibernate-3.2.0.cr5.pom
Transferring 3K
Downloading: freemarker/freemarker/2.3.4/freemarker-2.3.4.pom
Transferring 0K
Downloading: org/hibernate/jtidy/r8-20060801/jtidy-r8-20060801.pom
Transferring 0K
Downloading: org/apache/geronimo/specs/geronimo-jta_1.1_spec/1.1/geronimo-
jta_1.1_spec-1.1.pom
Transferring 1K
Downloading: org/apache/geronimo/specs/specs/1.2/specs-1.2.pom
Transferring 2K
Downloading: org/apache/geronimo/genesis/config/project-config/1.1/project-
config-1.1.pom
Transferring 14K
Downloading: org/apache/geronimo/genesis/config/config/1.1/config-1.1.pom
Downloading: org/apache/geronimo/genesis/config/config/1.1/config-1.1.pom
Downloading: org/apache/geronimo/genesis/config/config/1.1/config-1.1.pom
Transferring 0K
Downloading: org/apache/geronimo/genesis/genesis/1.1/genesis-1.1.pom
Downloading: org/apache/geronimo/genesis/genesis/1.1/genesis-1.1.pom
Downloading: org/apache/geronimo/genesis/genesis/1.1/genesis-1.1.pom
Transferring 6K
Downloading: org/apache/apache/3/apache-3.pom
Downloading: org/apache/apache/3/apache-3.pom
Downloading: org/apache/apache/3/apache-3.pom
Transferring 3K
Downloading: log4j/log4j/1.2.14/log4j-1.2.14.pom
Transferring 2K
Downloading: org/hibernate/hibernate-tools/3.2.0.beta9a/hibernate-tools-
3.2.0.beta9a.jar
Transferring 352K
Downloading: org/hibernate/jtidy/r8-20060801/jtidy-r8-20060801.jar
Transferring 243K
Downloading: commons-collections/commons-collections/2.1.1/commons-collections-
2.1.1.jar
Transferring 171K
Downloading: commons-logging/commons-logging/1.0.4/commons-logging-1.0.4.jar
Transferring 37K
Downloading: antlr/antlr/2.7.6/antlr-2.7.6.jar
Transferring 433K
Downloading: org/apache/geronimo/specs/geronimo-jta_1.1_spec/1.1/geronimo-
jta_1.1_spec-1.1.jar
```

```
Transferring 15K
Downloading: net/sf/ehcache/ehcache/1.2.3/ehcache-1.2.3.jar
Transferring 203K
Downloading: asm/asm/1.5.3/asm-1.5.3.jar
Transferring 25K
Downloading: freemarker/freemarker/2.3.4/freemarker-2.3.4.jar
Transferring 770K
Downloading: dom4j/dom4j/1.6.1/dom4j-1.6.1.jar
Transferring 306K
Downloading: asm/asm-attrs/1.5.3/asm-attrs-1.5.3.jar
Transferring 16K
Downloading: cglib/cglib/2.1_3/cglib-2.1_3.jar
Transferring 275K
Downloading: hsqldb/hsqldb/1.8.0.7/hsqldb-1.8.0.7.jar
Transferring 628K
Downloading: log4j/log4j/1.2.14/log4j-1.2.14.jar
Transferring 358K
Downloading: org/hibernate/hibernate/3.2.5.ga/hibernate-3.2.5.ga.jar
Transferring 2202K

db:
```

一大堆下载文件信息表明Maven Ant Tasks完成了它的任务，为我们下载了指定的文件（包括HSQLDB和Hibernate）以及所有依赖的库文件。这一过程可能需要花费一段时间才能完成（取决于网络连接的速度和服务器的情况），但是它只进行一次。下一次再运行Ant时，Maven Ant Tasks就会注意到你的本地库中已经包含了所有这些文件，就会默认继续执行其他需要完成的任务了。

在所有下载完成以后，Ant会打印输出“ db: ”，表明它正开始执行你请求的目标。一会儿，你应该会看到HSQLDB的图形界面，如图1-6所示。我们的数据库现在还没有什么东西，所以，除了命令能运行以外，没有其他内容。窗口左上部的树形视图是数据库中可以浏览的各种数据表和字段。就目前而言，只要确认最上层是否为“ jdbc:hsqldb:data/music ”就可以了。

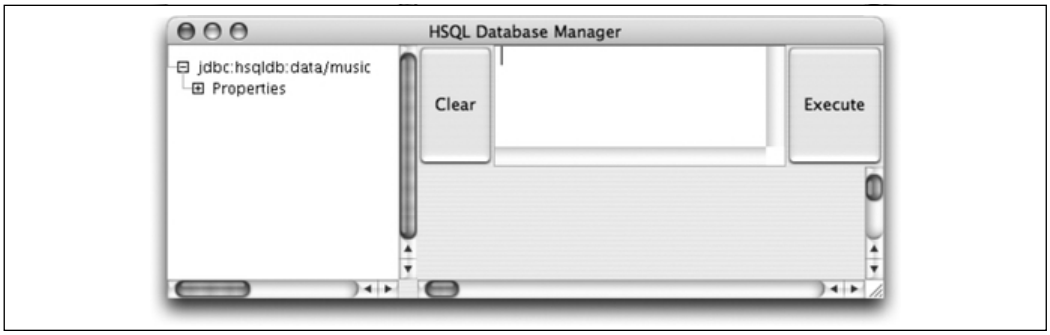


图1-6：HSQLDB数据库管理器界面

如果你喜欢，可以浏览一下菜单，但不要对数据库进行任何修改。在完成以后，选择“ File ”

→“Exit”，窗口就会关闭，同时Ant将报告以下信息：

```
BUILD SUCCESSFUL
Total time: 56 seconds
```

“Total time”是你运行这个数据库管理器程序所用的时间，所以其值是变化的（刚才我们在Ant的构建文件中为java任务设置了fork属性，所以直到关闭数据库以前，Ant会一直等待）。此时，如果你查看data目录，你会发现HSQLDB 已经创建了一些文件来保存数据库信息：

```
% ls data
music.log        music.properties music.script
```

你甚至可以查看这些文件的内容。和大多数数据库系统不同，HSQLDB是以人类可读的格式来存储数据的。.properties文件中包含一些基本设置，而.script文件中的数据则以SQL语句的形式保存。.log文件用于当应用程序崩溃或没有正常关闭数据库而强行退出时，可以重新构建一致的数据库状态。现在，你在这些文件中看到的都是基本定义，都是默认值；以后我们开始创建数据表并添加数据时，就可以看到这些文件会有所变化。这也可以作为一种有用的调试手段来进行基本的完整性检查，甚至比启动图形界面执行查询还要快。

注意：能够直接读取HSQLDB数据库文件的内容，是一件诡异但有趣的事。

发生了什么事

目前我们已经成功运行了第一个示例，建立了我们的项目构建文件，现在解释Ant如何取回这个项目必需的依赖文件。我们再查看一下示例的build.xml文件中的artifact:dependencies元素（如例1-2所示）。

例1-2：artifact:dependencies元素

```
<artifact:dependencies pathId="dependency.classpath">
  <dependency groupId="hsqldb" artifactId="hsqldb" version="1.8.0.7"/>
  <dependency groupId="org.hibernate" artifactId="hibernate"
              version="3.2.5.ga">
    <exclusion groupId="javax.transaction" artifactId="jta"/>
  </dependency>
  <dependency groupId="org.hibernate" artifactId="hibernate-tools"
              version="3.2.0.beta9a"/>
  <dependency groupId="org.apache.geronimo.specs"
              artifactId="geronimo-jta_1.1_spec" version="1.1"/>
  <dependency groupId="log4j" artifactId="log4j" version="1.2.14"/>
</artifact:dependencies>
```

如果你以前从没有用过Maven，这个例子看起来非常令人费解。我们先定义一些术语。首先是artifact。一个artifact就是项目生成的一个文件，它可以是任意类型的文件——Web应用程序的WAR文件、企业应用程序的EAR文件或是简单的JAR文件。对于我们的用途来说，我们使用的是JAR artifact，如果你不在dependency元素中指定它，其默认的类型就是jar，非常方便。

artifact有4个属性：groupId、artifactId、version以及type。例如，我们需要org.hibernate组内的hibernate artifact，其版本是3.2.5.ga，文件类型是jar。Maven将使用这些标识符在位于http://repo1.maven.org/maven2/的中央Maven 2库中来查找定位适当的依赖文件。使用这些值，Maven将通过以下模式来尝试定位Hibernate的JAR：<repositoryUrl>/<groupId>/<artifactId>/<version>/<artifactId>-<version>.<type>，其中，groupId中的点号“.”将转换成URL的路径分隔符。使用这样的模式，就可以定位Hibernate和HSQLDB依赖的JAR文件了，如例1-3所示。

例1-3：项目依赖文件的URL

```
http://repo1.maven.org/org/hibernate/hibernate/3.2.5.ga/hibernate-3.2.5.ga.jar
http://repo1.maven.org/hsqldb/hsqldb/1.8.0.7/hsqldb-1.8.0.7.jar
```

在build.xml文件中，我们没有在Hibernate依赖声明中包含JTA依赖。这是因为Hibernate库使用的是一个非自由（nonfree）的JAR文件，所以Maven 2公共库中没有这个文件。这个项目没有使用Sun提供的标准JTA API JAR，而是使用了Apache Geronimo项目创建的JTA API。geronimo-jta_1.1_spec是Java Transaction API.的一个自由的开源实现版本。这个替换是通过在Hibernate 3.2.5.ga的依赖声明中使用exclusion元素来实现的，并随后明确声明了需要使用Geronimo JTA spec 1.1。

好了，我们已经介绍了Maven Ant Tasks如何从Maven库中取回依赖文件，但是，如果已经下载好了这些依赖文件，又会怎么样呢？Maven Ant Tasks将所有的依赖文件都下载到一个本地的Maven库中，由Maven负责维护这个本地库，使其结构与远程Maven库的结构保持一致。当需要检查某个artifact时，Maven先检查本地库，如果本地库没有这个artifact时，再从远程库中请求该artifact。这意味着，如果同时有20个项目都引用了同一版本的Hibernate，从Maven远程库中只下载一次Hibernate的依赖artifact，所有20个项目都会引用保存在本地Maven库中的同一个副本。那么，这个不可思议的本地Maven库是放在哪里呢？最简单的回答就是，在build.xml文件中添加一个目标。在原来的Ant build.xml的末尾添加以下目标，如例1-4所示。

例1-4：打印输出依赖类的路径

```
<target name="print-classpath" description="Show the dependency class path">
  <property name="class.path" refid="dependency.classpath"/>
  <echo>${class.path}</echo>
</target>
```

运行这个目标，将生成以下输出结果：

```
% ant print-classpath
Buildfile: build.xml

print-classpath:
     [echo] ~\.m2\repository\commons-logging\commons-logging\1.0.4\
commons-logging-1.0.4.jar;\
~\.m2\repository\dom4j\dom4j\1.6.1\dom4j-1.6.1.jar;\
~\.m2\repository\cglib\cglib\2.1_3\cglib-2.1_3.jar;\...
```

运行该目标，输出结果会随你正在使用的操作系统的不同而有所变化，不过，你可以看到，依赖类的路径都引用了本地Maven库。在运行Windows XP的计算机上，这一路径可能位于C:\Documents and Settings\Username\.m2\repository；在运行Windows Vista的计算机上，这一路径则可能位于C:\Users\Username\.m2\repository；而在Unix和Macintosh上，该路径将会是~/.m2/repository目录。

你也可能会注意到，在类路径中列出的依赖文件的数目要比我们在build.xml的artifact:dependency元素中声明的多。这些额外的依赖就是所谓的可传递的依赖（transitive dependency），它们是你明确声明的依赖文件所需要的依赖。例如，Hibernate需要依赖CGLib、EHCache、Commons Collections以及其他程序库。对Maven的详细介绍超出了本书讨论的范围，我在这里只对Maven Ant Tasks如何为你的项目构造整套依赖结构提供一些提示。如果在构建好一个示例后，你查看一下本地Maven库，你会注意到在每个JAR artifact旁边会有一个扩展名为.pom的项目对象模型（POM）文件。POM是Maven构建系统和仓库的基础，每一个POM描述了一个artifact和它的依赖。Maven Ant Tasks使用这种元数据（metadata）来构造一个描述依赖传递关系的树状结构。换句话说，Maven Ant Tasks并不只是负责下载Hibernate，它们还负责下载Hibernate依赖的所有文件。

你实际需要知道的是它的工作原理就是这样的。

接下来做什么

感谢Maven Ant Tasks，有了它，就不用像本书的最初版本那样亲自查找、下载、解压软件，并组织好它们。现在你可以从一个更高的起点来开始使用Hibernate，在下一章中你可以看到，我们的进展会非常快速。你将看到Hibernate已经为你写好了Java代码！数据库模式（database schema）也好像是从天而降一样（或者，至少是来自于产生Java程序代码的同一个XML映射表）！真正的数据表和数据也会出现在HSQLDB管理器界面中（或者，至少是示范数据）！

听起来令人兴奋吧？嗯，和你以前的开发方法相比如何？我们继续研究下去，把Hibernate的强大功能唤醒吧。

为何无法工作

如果你没有看到数据库管理器窗口，而是出现了错误信息，那么应该尝试弄清楚是否是文件构建出了问题；是否是安装设置Ant或项目层次结构出错；是否是因为访问Internet困难，而不能从Maven库下载到依赖文件；或是其他什么原因。再次确认所有东西都安排妥当，并且按照前面介绍的方式正确地安装。如果是自己输入的文件有问题，可以考虑下载示例程序。