在 JDeveloper 中使用 Subversion

作者：不详 来源：网络

目录

简介

Subversion 是一个开放源代码的版本控制系统。它功能强大，可以替代常用的并发版本控制系统 (CVS)。它对 CVS 进行了多项改进，如版本控制目录和元数据，并且其体系结构可实现更简单、更灵活的网络访问。Oracle 为 JDeveloper IDE 提供了一个扩展功能，该扩展功能可将与 Subversion 交互所涉及的大部分手动工作自动化。有关 Subversion 的详细介绍，请查阅 http://svnbook.red-bean.com/ 上的“Version Control with Subversion”（用 Subversion 进行版本控制）。

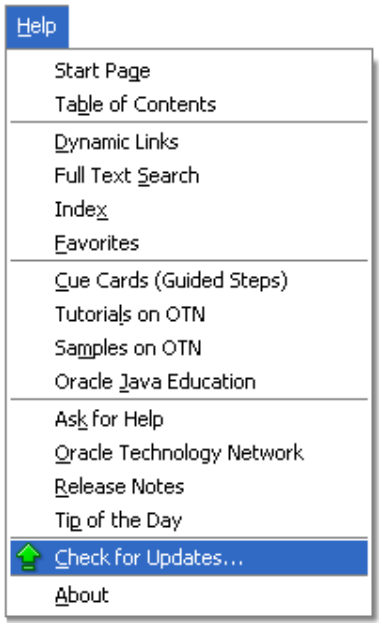
本指南将引导您完成安装 Subversion 并配置 JDeveloper 来使用它所必需的步骤。随后，还将介绍在 JDeveloper 中使用 Subversion 时常遇到的几种情形。

回页首

安装 JDeveloper Subversion VCS 扩展

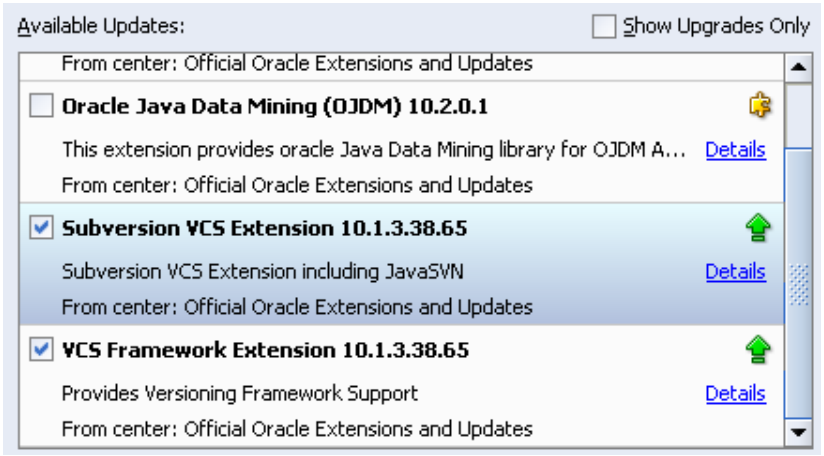
JDeveloper Subversion VCS 扩展基于 JDeveloper 生产版本 10.1.3.0.4。您将需要 J2EE 或 Studio 版；并将在未来版本中支持 Java 版。如果您尚未下载并安装 JDeveloper 10.1.3（J2EE 或 Studio），请从 http://otn.oracle.com/products/jdev 下载。

JDeveloper Subversion VCS 扩展可从 Oracle 官方更新中心获得。您可以使用 Help 菜单中的 Check for Updates 功能来安装更新：

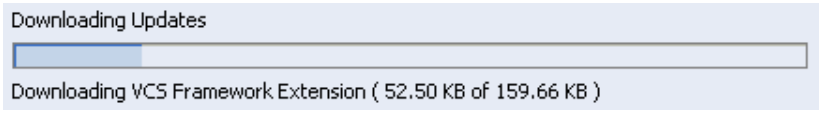


使用 Check for Updates 向导，搜索 Official Oracle Extensions and Updates 中心：

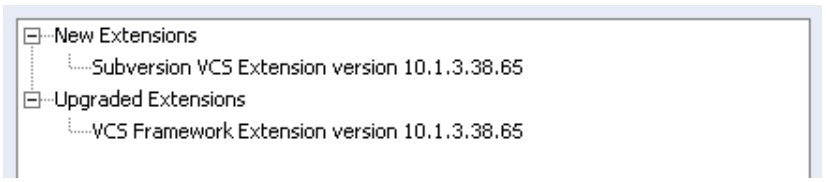
找到并选择 Subversion VCS Extension 项。选择该项还会自动选择 VCS Framework Extension 更新。



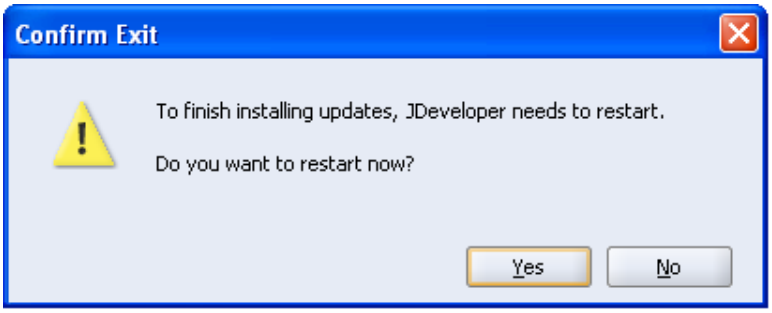
进入下载阶段：



下载更新后，您将看到一个摘要页面，其中有如下所示的信息：



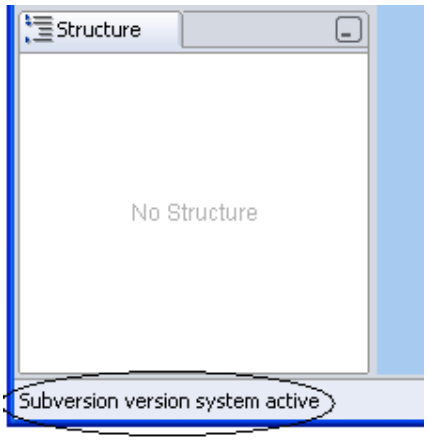
向导完成时，可能会询问您是否要自动重启 JDeveloper：



选择“Yes”。

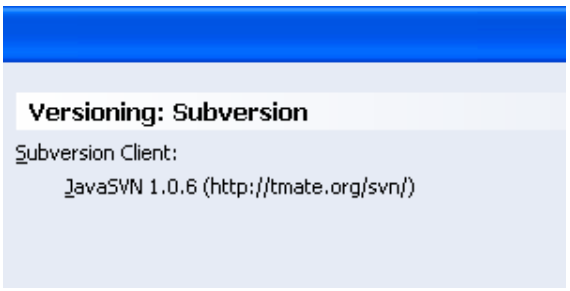
当 JDeveloper 重启后，打开 Versioning 菜单，选择 Select Version System，然后选择 Subversion。

如果配置成功，您将在结构窗格下面看到状态消息：



假定配置成功，请查看 Subversion 的首选项，检查 JavaSVN 客户端库是否按预期安装：打开 Preferences 对话框 (Tools > Preferences)，然后在左窗格中选择 Versioning | Subversion。

正确安装扩展后，您将看到与以下类似的信息：



回页首

连接 Subversion 信息库

Subversion 信息库保存所有版本控制数据。

如果您先前使用 Subversion 作为版本控制系统，则可以在 JDeveloper 中将连接该信息库。在 Subversion Navigator (View > Subversion Navigator) 中，右键单击 Subversion 节点，然后选择 New Repository Connection。在 Create Subversion Connection 对话框中，使用适当的访问方法协议 (http://、https://、svn://、svn+ssh://) 输入信息库位置。您还可以使用 file:/// 协议，但这样做需要安装其他软件，如附录二中所述。

> 提示：如果您要使用本开发人员指南中的示例，则需要安装一个名为 c:/svn-repos 的本地 Subversion 信息库，并使用 svn:// 协议与之连接。如果您还安装有单独的 Subversion 客户端软件，则 JDeveloper Subversion VCS 扩展将允许您安装一个本地信息库。请参阅创建本地 Subversion 信息库。

如果您要保存 Subversion 信息库连接的详细信息，以便能够在以后使用或在其他机器上使用，则可以将这些信息导出到一个文本文件中。以后，您可以通过从文件导入这些详细信息来重建连接。要执行导出操作，请在 Subversion Navigator 中右键单击 Subversion 节点，然后选择 Export Connections。完成 Export Subversion Connections 对话框，然后单击 OK。要从文件中重新导入连接详细信息，请右键单击 Subversion 节点，然后选择 Import Connections。通过 Import Subversion Connections 对话框，浏览到包含您希

望导入的连接详细信息的文件，然后单击 OK。

回页首

将 JDeveloper 项目导入 Subversion

要提供用于本指南的示例，请创建一个新的 Java 应用程序和项目，该项目具有一个 Java 程序包和两个简单的 Java 类。指定 c:\jdev1013\jdev\mywork\test 作为项目位置。

您在使用 Subversion 控件之前于 JDeveloper 中创建的文件（如上述文件）必须导入 Subversion 信息库，然后再从其中签出。这些签出的文件就成为您的“工作副本”。

您将使用 JDeveloper 的 Import to Subversion 向导来导入现有 JDeveloper 项目。

在开始之前，您需要知道将存储 JDeveloper 项目的 Subversion 信息库的位置。对于本指南中使用的示例，使用了一个 URL，如下所示：

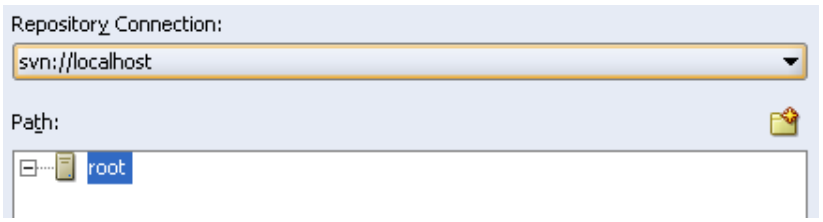
```
svn://localhost/
```

另一个有用的信息是您的 JDeveloper 应用程序所在目录的位置，但这将根据调用向导的上下文设置。就本指南而言，我们使用的示例是：

```
c:\jdev1013\jdev\mywork\test
```

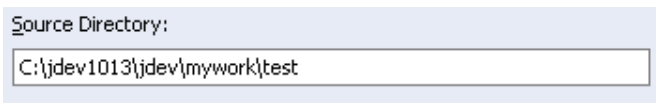
通过右键单击您要导入的 JDeveloper 应用程序或项目（在本例中是 TestApplication）启动向导，然后选择 Versioning > Import Files。这将开启 Import to Subversion 向导。

在 Destination 页面上，确保 Repository Connection 框包含您先前创建的信息库的 URL（或连接名称，如果您已指定）：



通常，您应该在信息库中选择要导入到的位置。对于本指南，您已经创建了一个新的空白库，因此，您现在可以保留 Path 选择为“root”。

在 Source 页面上，确保 Source Directory 框包含您的 JDeveloper 应用程序所在目录的位置。对于本指南中的示例，该目录应该是：

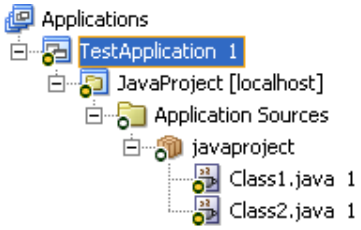


您不必在 Filters 页面上进行任何更改。

在 Options 页面上，确保已选中 Perform Checkout 复选框：

单击 Finish 完成导入操作。

展开 TestApplication 节点及其下面的节点，可以看到以下视图。



由于您允许向导签出导入的文件，因此这些文件显示在导航树中，且旁边有一个版本号。

在 Subversion 中，版本号指的是整个信息库而不是其中的单个文件的版本号。因此，文件旁边的版本号表示该文件的最新修订。在导入新的空白信息库时，所有文件的版本号都设为“1”。如果导入非空信息库，则版本号将根据信息库的现有内容指定，但所有新导入文件的版本号都是相同的。

在 JDeveloper 中，通过在 Application Navigator 中右键单击文件并选择 Versioning > Properties，您可以看到该文件的版本信息。

如果您设置首选项以显示导航器状态重叠图标（参见 Tools > Preferences | Versioning | Subversion | General），则还可以看到指示文件版本状态的小符号。例如，桔黄色圆圈表示文件自上次在信息库中更新后未曾修改过。

现在可以在 Application Navigator 中看到的文件是 Subversion “工作副本”文件，您随时可以使用它们。

回页首

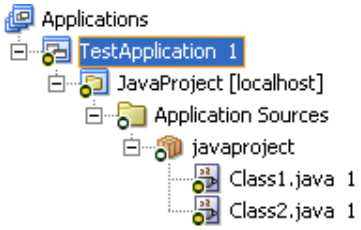
签出文件

“签出”是当您希望创建 Subversion 信息库中存储的文件和文件夹的初始本地副本时所执行的任务。您处理的正是这些存储在 Subversion 信息库外的本地副本。

如果您使用 Import to Subversion 向导时选中了 Perform Checkout 向导，则已经签出导入的文件。

例如，要独立于导入操作签出文件，请在 Subversion Navigator 中，右键单击连接节点（在示例中为 svn://localhost），然后选择 Check Out。确认您要从根节点签出。系统会为签出文件建议一个位置。为了避免与作为导入操作的一部分而签出的文件相混淆，将目标更改为 c:/work，然后单击 OK。

现在，您从 Subversion 信息库签出的文件将出现在 Application Navigator 中，它们旁边有一个版本号。

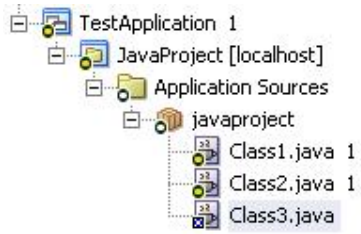


回页首

添加和提交文件

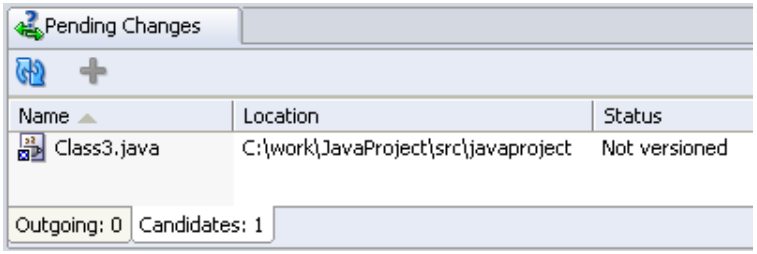
当您在 JDeveloper 中创建新文件时，如果要对其使用版本控制功能，并希望其他开发人员能够使用它们，则必须将它们添加到 Subversion 控件。为此，您需要使用 JDeveloper 的 Add 命令，然后使用 Commit 命令。

在 Application Navigator 中，创建一个新的 Java 文件 (Class3.java) 作为版本控制应用程序的一部分。



您将看到，新文件的导航图标是具有蓝色背景的白色斜交叉线。这表示，目前版本控制系统不知道该文件，您应该添加它。

在添加之前，应查看 JDeveloper 工具，该工具将帮助您记录应对信息库进行的所有更改。如果您尚未开启 Pending Changes 窗口，请通过选择 Versioning > Pending Changes 来开启。该窗口的 Candidates 选项卡将是桔黄色的，这表示它包含您还未看过的更改。单击选项卡，您将注意到，其中列出了您创建的文件 (Class3.java)，其状态为“Not versioned”。

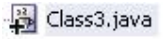


您可以在此处将 Class3.java 添加到 Subversion 控件。在未来版本中，Pending Changes 窗口还将告诉您是否已经对信息库中的文件进行了更改，并允许您更新本地副本（如果您打算这么做）。

要将 Class3.java 添加到 Subversion 控件，请执行下列操作之一：

- 在 Application Navigator 中选择该文件，然后选择 Versioning > Add。
- 在 Pending Changes 窗口中右键单击该文件，然后选择 Add。
- 在 Pending Changes 窗口中选择该文件，然后单击 Add 图标 。

上述任何一种方法都会开启一个对话框，您可以通过这个对话框确认文件添加。一旦添加，导航图标就会变为小型黑色十字叉：



这表示，虽然 Subversion 知道了有该文件，但该文件还不是信息库的一部分。要使该文件成为信息库的一部分，您可以使用 JDeveloper 的 Commit 命令：在 Application Navigator 中选择 Class3.java，然后选择 Versioning > Commit。这将开启 Commit Resources 对话框。您可以使用该对话框中的 Comments 框来描述您对该文件执行的操作。在本例中，您可以添加注释“Initial commit of Class3.java”。当您查看文件的版本历史时，可以看到这些注释，并且它们将帮助您和项目的其他开发人员识别各个版本。

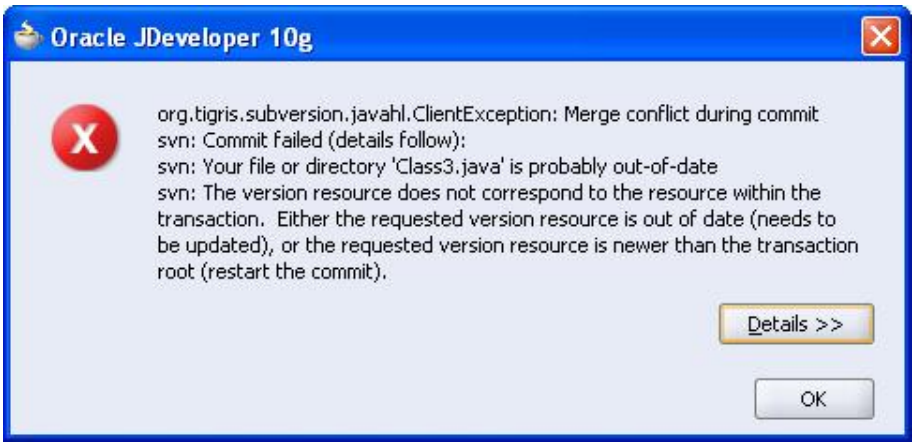
提示：在现场开发环境中，为了避免不必要的冲突以及带来的解决工作，您应该在提交文件之前更新这些文件。

回页首

更新文件

“更新”是将本地文件与 Subversion 信息库中的文件同步的过程。因为其他开发人员将更改提交给了信息库，所以应该执行该操作。您可以使用 JDeveloper 的 Update 命令来更新文件。

当您尝试提交文件的本地版本时，可以知道信息库中是否有较新版本的文件。因为系统会显示一个消息，使您无法继续执行提交过程。



您可以更新整个工作副本、单个目录或程序包，或者单个文件。

要更新文件，请在 Application Navigator 中选择这些文件或其父目录，然后选择 Versioning > Update。

- 如果可以更新文件且不会导致冲突，则导航图标将变为桔黄色圆圈 ，而文件内容将更新为信息库版本中的内容。
- 如果在更新文件时有冲突，则导航图标将变为惊叹号 ，并且文件内容将发生更改，以显示原始内容和信息库中的内容。冲突内容将由冲突标记分隔开（请参阅解决冲突）。

回页首

编辑文件

您可以直接在 JDeveloper 中或外部编辑工具中编辑工作副本中的文件。二进制文件和文本文件均可以由版本控制系统处理，因此您可以使用文本编辑器、文字处理程序、图形程序或您常用的任何其他工具。

当您希望编辑文件时不需要进行注册（换句话说，您不必像在其他版本控制系统中那样“获得编辑权”）。当您进行更改时，Subversion 会自动检测到文件已经更改，并且将在 JDeveloper 中更新文件的版本控制状态以反映这一点。

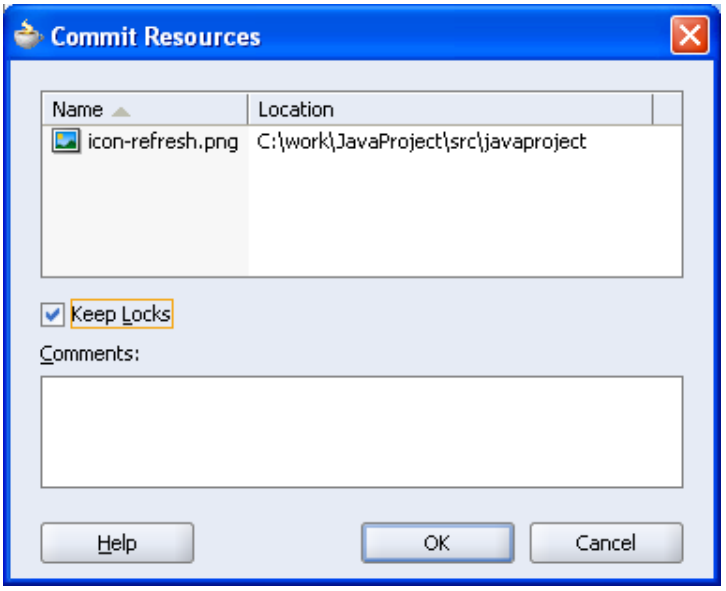
> 提示：要确保您在 JDeveloper 中看到文件的最新状态，请单击 Application Navigator 工具栏上的 Refresh 按钮 。

如果多个开发人员要处理同一个文件，则必须有一个机制可以防止一个开发人员的工作无意地覆盖另一个开发人员的工作。Subversion 针对这一点的首选解决方案就是复制-修改-合并模型。在该模型中，使用文件的每个开发人员都会复制该文件，修改该文件，然后将其与其他开发人员的副本合并在一起。看起来，这似乎会在内容上导致频繁的冲突，但实际上很少出现这种情况。万一发生冲突，我们还有一个有效的解决方法（请参阅解决冲突）。

上述机制的一个替代机制是锁定-修改-解锁模型。在该模型中，开发人员会锁定文件，修改文件，将文件添加回信息库，然后进行解锁。锁定文件不会防止其他开发人员处理文件的副本：它只会防止该副本的内容返回信息库，直到锁定解除。因此，该模型仍然会产生不必要的工作。

锁定-修改-解锁模型更适合处理二进制格式的文件。在二进制格式中，不可能合并冲突的更改。它可确保开发人员严格按次序将更改提交给文件。

在该版本的 JDeveloper 中，锁定支持仅限于在您要提交的文件上的锁定。您在使用 Subversion lock 命令前必须锁定文件。在未来版本中，将全面支持锁定和解锁功能。



保留锁定意味着，其他开发人员仍然无法提交他们对其文件副本所作的更改。

要使用 Subversion lock 命令锁定文件，请在命令提示符下输入以下命令：

```
svn lock [文件名] -m "[消息]"
```

要解除文件锁定，请使用以下命令：

```
svn unlock [文件名]
```

回页首

比较与合并文件修订

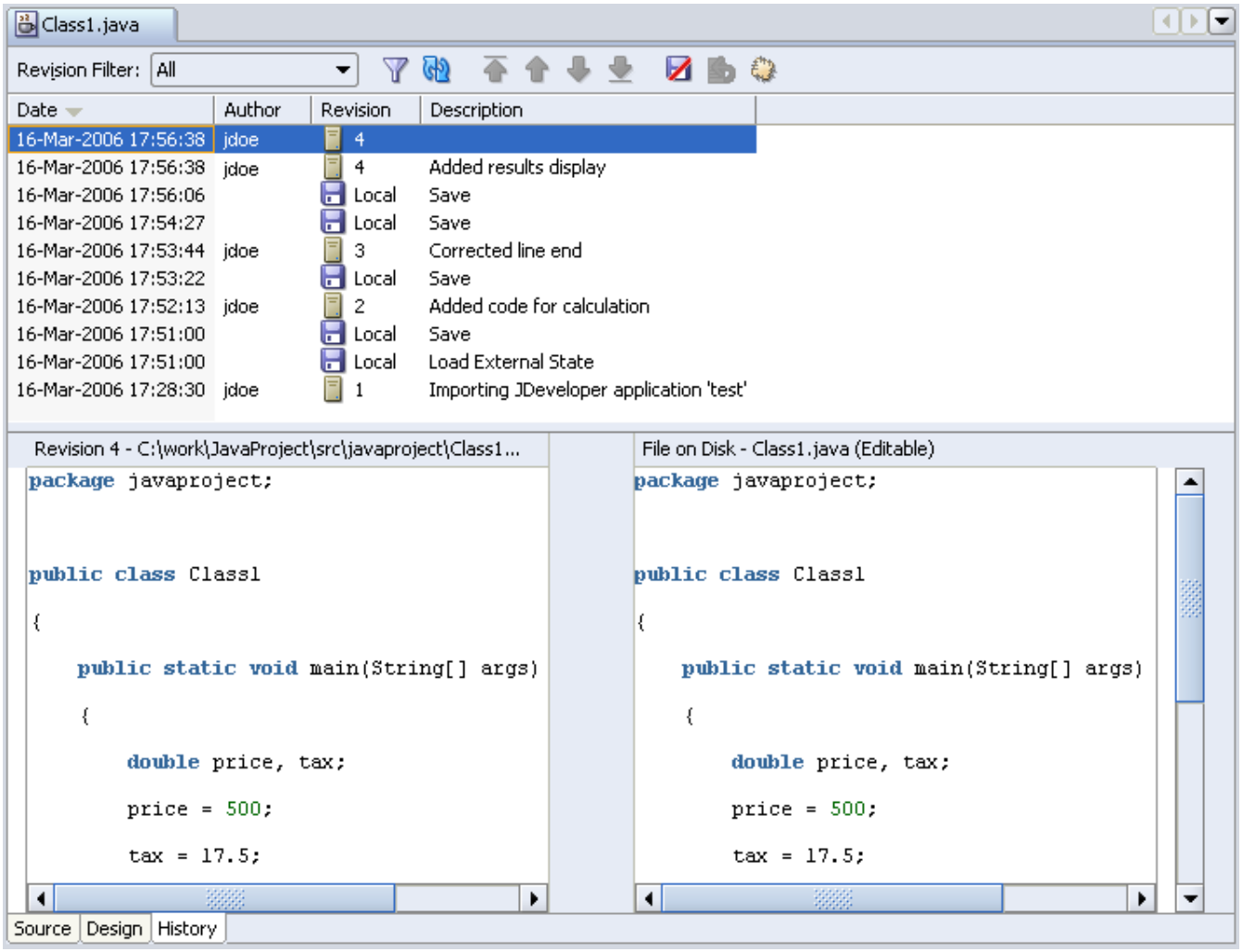
JDeveloper 提供了两个比较文件的工具，即本地历史工具和版本历史工具。

使用本地历史工具

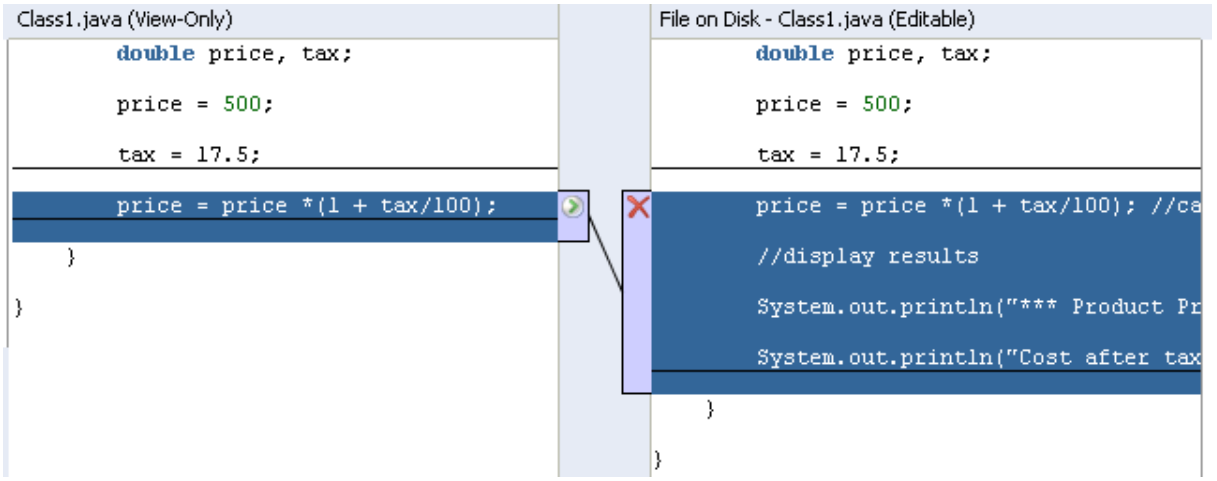
使用本地历史工具，您可以：

- 查看文件早期版本的内容。
- 将早期版本的内容合并到最新版本中。
- 解决早期版本和最新版本之间的冲突。

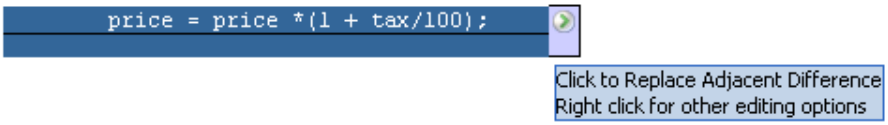
要查看特定文件的历史，请双击打开该文件，然后单击 History 选项卡。



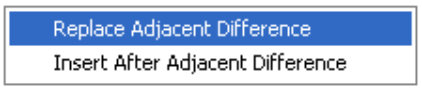
左面板包含当前选择的修订内容。您无法编辑该内容。右面板包含最新修订的内容（与您在 Source 选项卡上看到的一样）。您可以编辑右面板中的内容。如果您进行了编辑，则您的更改会立即反映到 Source 选项卡上。对于最新修订，左面板和右面板中的内容没有差异。对于早期修订，差异通过描述冲突行的链接框显示。



两个面板间的空白中的符号给出了解决冲突的操作建议。将鼠标指针悬停在符号上可以显示说明操作建议的工具提示。



如果有任何其他操作，您可以从符号的上下文菜单（右键单击符号）中实现。



使用版本历史工具

使用版本历史工具，可以将文件的任何版本与该文件的任何其他版本相比较。这对于查看其他开发人员对文件所作的更改十分有用。所显示的文件版本是只读的：您可以比较文件两个版本的内容，但无法编辑任一版本。因此，版本历史工具无法用于合并文件版本。

要查看和比较文件的多个版本，请在 Applications Navigator 中选择该文件，然后选择 Versioning > Version History。

版本历史工具与前面描述的本地历史工具类似，但具有以下区别：

- 它有两个版本选择面板而非一个，因此您可以选择要比较的任意版本组合。（但是，请注意，只有左面板具有差异导航控件。）
- 虽然版本间的差异可以通过亮显和链接框显示出来，但没有与之关联的操作图标或上下文菜单。

回页首

解决冲突

当两个文件的相同行位置上包含不同内容时，就会发生冲突。当您尝试提交文件时，将遇到文件的本地副本与信息库中的版本相冲突的情况。在这种情况下，系统会禁止提交，并且当您从信息库更新本地文件之后，文件的导航图标中会显示一个惊叹号  。

如果在您的某一个文件中发生冲突，您必须先解决这些冲突，然后才能将文件提交到信息库。JDeveloper 中提供了一个合并工具。通过这个工具，您可以解决基于文本的文件中的冲突。解决二进制文件中的冲突没有简单的方法，因此 JDeveloper 可让您指示 Subversion 即使没有进行更改也可以解决冲突。

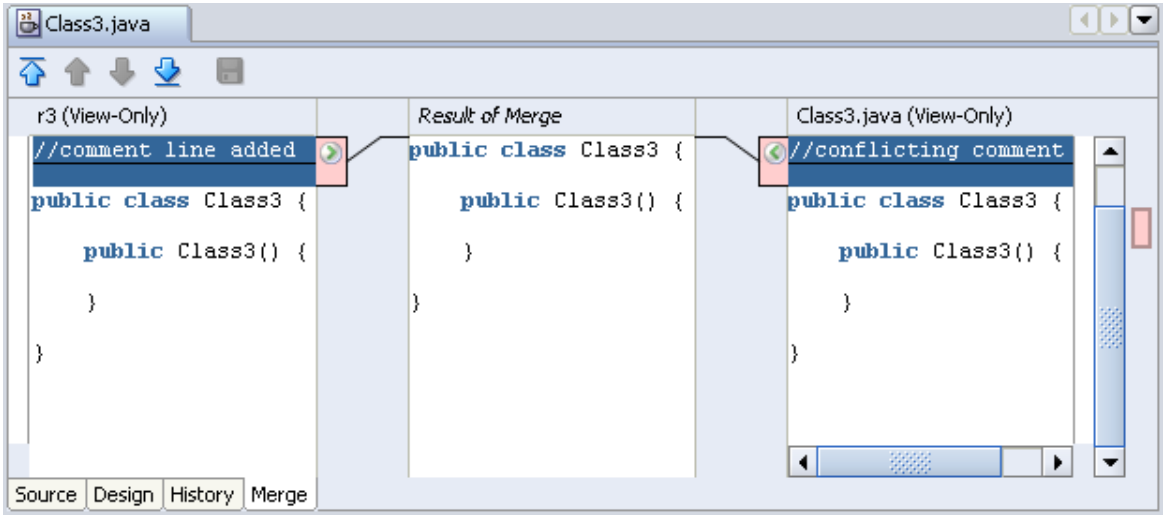
基于文本的文件中的冲突

在产生冲突的基于文本的文件中，将更改文件内容，显示原始内容和信息库中文件的内容。冲突内容将由冲突标记分隔开。下面的简单示例显示了一个由于在 Java 文件的同一行添加不同注释而引发的冲突。

```
<<<<<<< .mine
//conflicting comment line
=======
//comment line added
>>>>>>> .r321
```

冲突内容的开头由起始尖括号表示，结尾由结束尖括号表示，后面还有一个修订编号。您的内容在“等号”符号行前面显示。信息库中的内容在“等号”符号行后面显示。

要启动合并工具来解决冲突，请在 Application Navigator 中选择文件，然后选择 Versioning > Resolve Conflicts。



左面板包含信息库中版本的内容。面板标题会显示修订编号。您无法编辑该内容。右面板包含最新本地版本的内容。该内容也无法编辑。中心面板包含合并结果。当您解决单个冲突后，内容将会更新。您还可以直接编辑内容。三个面板间的空白中的符号指出了解决每个冲突的操作建议。将鼠标指针悬停在符号上可以显示说明操作建议的工具提示。如果有任何其他操作，您可以从符号的上下文菜单中实现。接受最初的操作建议可能会导致出现其他操作建议。

要完成合并，您必须使用 Save 按钮保存所作的更改。

二进制文件中的冲突

由于解决二进制文件（如图形文件）中的冲突并不容易，您可以向 Subversion 表明冲突文件就是那个应该被视为正确的文件，即使您没有对其进行更改。为此，请在 Application Navigator 中选择文件，然后选择 Versioning > Mark Resolved。文件的导航图标从惊叹号 变为星号 以表示经过修改的文件。当您提交该文件 (Versioning > Commit) 时，Subversion 信息库应该能接受它，而不会显示冲突消息。

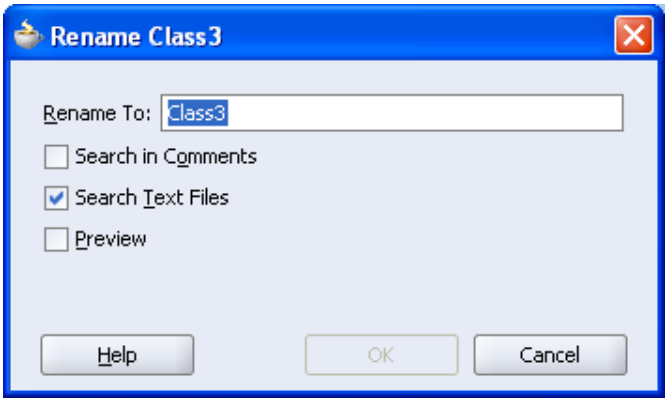
回页首

重命名文件

Subversion 允许重命名受版本控制的源文件，并且 JDeveloper 支持这项功能。

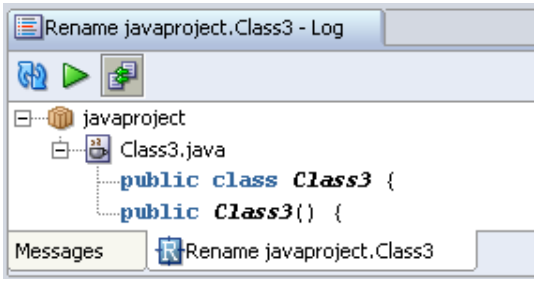
当您在 JDeveloper 中重命名源文件时，将重命名该文件以及其他源文件中的该名称。

要将 Java 源文件 Class3.java 重命名为 Class4.java，请在 Application Navigator 中选择 Class3.java，然后选择 Refactor > Rename。这将开启 Rename 对话框。



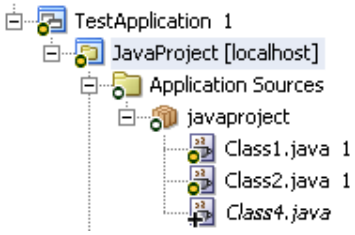
用新名称 (Class4) 覆盖现有名称 (Class3)。您不需要指定文件扩展名。如果您要在项目的文本文件中更改名称实例，请选中 Search Text Files 复选框。

> 提示：如果您要在进行更改前查看名称使用列表，请选中 Preview 复选框。当您单击 OK 后，预览将显示在 Log 窗口中。日志将显示一个包含程序包和 Java 文件的可折叠树。每个文件下都会列出一个或多个名称使用实例。双击名称使用实例可以在 Edit 窗口中进行查看。在完成重命名操作之前，可以使用预览来检查并修改或排除个别使用实例。



如果您选择了 Preview 选项，可以通过在 Log 窗口中单击 Do Refactoring 按钮 来完成重命名。

如果您没有选择 Preview 选项，则在单击 OK 时进行文件名更改。Application Navigator 现在显示，需要保存（其名称为斜体）并提交（导航图标包含一个小型黑色十字叉）新命名的文件。



在 Pending Changes 窗口中，具有旧文件名的文件已列入删除计划，而具有新文件名的文件已列入添加计划。

当您尝试提交文件时，将告诉您已经修改了 Class4.java，并提示您保存文件（您应该保存文件）。

如果您查看 Class4.java 的本地历史（在 JDeveloper 中打开该文件，并单击 History 选项卡），您将看到该文件的历史详细信息仍然是 Class3.java 的历史详细信息。

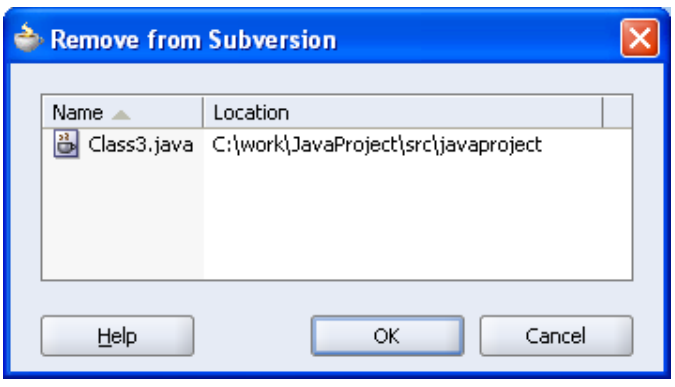
> 提示：如果您在提交前改变想法，不打算重命名文件，则可以使用 Revert 命令。首先，保存重命名文件 — 如果没有保存，您将无法继续。然后，还原重命名的文件 (Versioning > Revert)。这将使用用于重命名文件的名称创建一个新文件（在上述示例中是 Class4.java）。您可以删除这个新文件（如果愿意）。现在，转到 Pending Changes 窗口，找到具有原始名称的文件，然后进行还原。这样将把原始文件恢复到 Application Navigator，并显示为最新。但是，请注意，这也会重命名其他文件中对该文件的引用，而且这些引用将无法改回原始名称。您必须搜索这些引用，并对其进行手动更改。

回页首

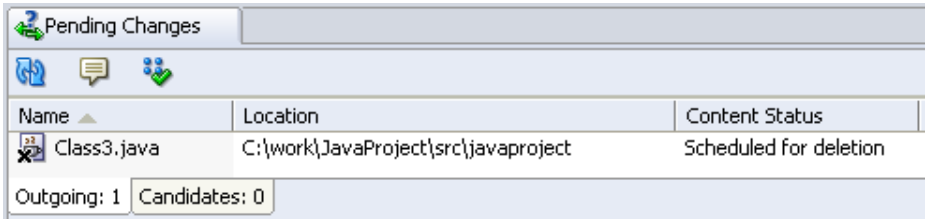
从 Subversion 控件中移除文件

> 警告：从 Subversion 控件中移除文件时会从磁盘中清除文件。

要从 Subversion 控件中移除文件，请在导航器中选择文件，然后选择 Versioning > Remove。



如果您单击 OK 按钮，Pending Changes 窗口的 Outgoing 选项卡会将该文件显示为计划删除：



提示：如果您此时改变想法，不打算移除文件，请在 Pending Changes 窗口中选择文件，然后选择 Versioning > Revert。

要从 Subversion 控件中移除文件，请在 Pending Changes 窗口中选择文件，然后选择 Versioning > Commit。将注释添加到 Commit Resources 对话框，然后单击 OK。该文件将从 Subversion 信息库视图、本地工作副本以及 JDeveloper 的所有视图中移除。

回页首

附录一：安装 Subversion 客户端软件

除了 JDeveloper Subversion VCS 扩展以外，您不必安装任何 Subversion 软件，但以下情况除外：

- 您希望使用 JDeveloper Subversion VCS 扩展创建本地 Subversion 信息库。
- 您希望使用 Java 绑定（帮助类库）而不是该扩展提供的 JavaSVN。
- 您希望通过代理服务器连接到 Subversion 信息库（请参阅通过代理服务器连接到 Subversion 信息库）。

在上述所有情况下，您需要安装单独的 Subversion 客户端软件。如果您希望使用其他 Java 绑定，还需要安装绑定软件（请参阅安装 JavaHL 库）。

如果您需要安装 Subversion 客户端软件，可以从 http://subversion.tigris.org 获得必要的 Subversion 源代码以及特定平台的二进制文件。您应该安装 Subversion 1.3.2。如果您使用的操作系统带有程序包管理系统（例如，GNU/Linux 的某些发布版本或其他常用的 Unix 系统），则可以咨询您的 OS 供应商，或根据源代码编译客户端软件。

对于 Microsoft Windows，请从 http://subversion.tigris.org 下载 Subversion 安装程序 svn-1.3.2-setup.exe。然后，运行该安装程序，并将 Subversion 客户端放在一个方便的位置，如 c:\subversion。要检查安装，请打开一个命令提示符，然后键入 `svn help`。如果您没有看到子命令列表，请检查系统路径是否包含了 Subversion 的安装位置（例如，`echo %PATH%` 应该包含 c:\subversion\bin）。

回页首

附录二：安装其他 Java 帮助类库

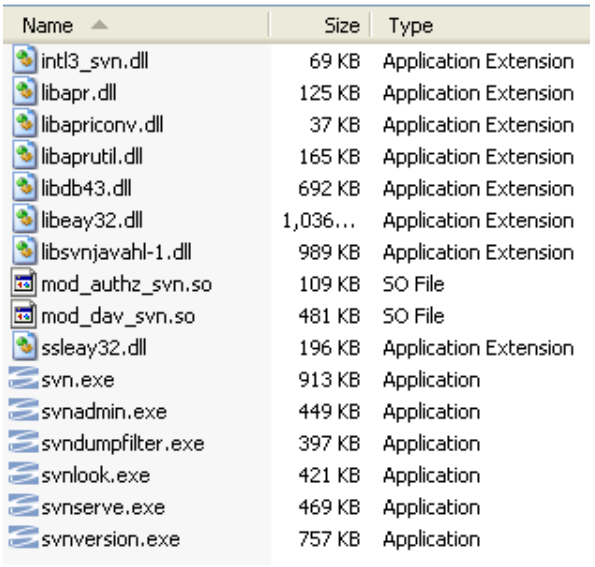
JDeveloper 需要一个帮助类库，才能与 Subversion 信息库和工作副本进行通信。帮助类库 JavaSVN（请参见 http://tmate.org）将自动与 JDeveloper Subversion VCS 扩展一同安装。您还可以安装 Subversion 自己的库，即 JavaHL。JavaHL 绑定的优势是可以由 Subversion 的创建者开发和维护，从而允许通过各种协议（`http://`、`https://`、`file:///`、`svn://`、`svn+ssh://`）访问信息库。如果您安装了 JavaHL，JDeveloper 将让您选择是使用 JavaHL 还是使用 JavaSVN。

安装 JavaHL 库

JDeveloper 可以与 JavaHL 库的 1.3.2 版配合使用。

对于 Microsoft Windows：

1. 首先，确保客户端安装的 bin 目录（请参阅安装 Subversion 客户端软件）位于系统路径上。
2. 从 http://subversion.tigris.org 下载 JavaHL 库 (svn-win32-1.3.2_javahl.zip)。
3. 使用工具（如 WinZip）将其解压到临时位置（如 c:\temp）。
4. 在 Windows 资源管理器中找到解压缩目录，如 c:\temp\svn-win32-1.3.2。
5. 将 libsvnjavahl-1.dll 文件复制到 Subversion 客户端安装的 bin 目录下，如 c:\subversion\bin。
   客户端安装的 bin 目录应该包含以下文件：



Name	Size	Type
intl3_svn.dll	69 KB	Application Extension
libapr.dll	125 KB	Application Extension
libapriconv.dll	37 KB	Application Extension
libaprutil.dll	165 KB	Application Extension
libdb43.dll	692 KB	Application Extension
libeay32.dll	1,036...	Application Extension
libsvnjavahl-1.dll	989 KB	Application Extension
mod_authz_svn.so	109 KB	SO File
mod_dav_svn.so	481 KB	SO File
ssleay32.dll	196 KB	Application Extension
svn.exe	913 KB	Application
svnadmin.exe	449 KB	Application
svndumpfilter.exe	397 KB	Application
svnlook.exe	421 KB	Application
svnserve.exe	469 KB	Application
svnversion.exe	757 KB	Application

提示：查看您的计算机上是否有上述文件的其他版本。如果有，请确保系统路径中指向它们的任何项要晚于指向 Subversion 文件的项。如果它们在路径中出现的较早，Subversion 将尝试使用它们，但如果它们不兼容，将发生错误。例如，如果使用了文件 ssleay32.dll 的不兼容版本，您将看到消息“SSL negotiation failed: SSL disabled due to library version mismatch”（SSL 通信失败：SSL 因库版本不符而禁用）。

对于 GNU/Linux 和其他基于 UNIX 的操作系统：

- 如果您的 OS 版本使用程序包管理系统，请查阅该系统的文档，确保安装了 JavaHL 库对象。
- 如果您根据源代码编译了 Subversion 客户端，则可能需要执行额外的步骤来确保构建 JavaHL 库对象。通常，这需要使用以下命令：`make javahl && make install-javahl`

针对 JavaHL 库配置 JDeveloper

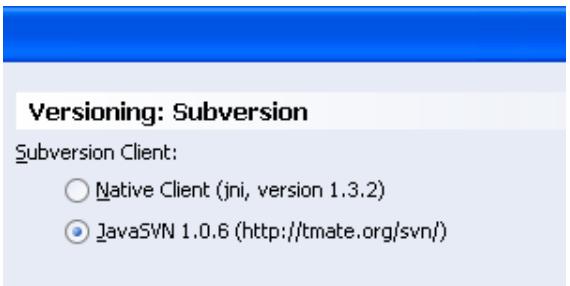
如果您安装了 JavaHL 库，请仔细检查 OS 原生对象是否在系统路径上：

- 对于 Microsoft Windows，如果您遵循本文档前面描述的步骤，则 DLL 应该已经位于系统路径上。要验证这一点，请打开一个命令提示符，回显路径变量（`echo %PATH`），并检验它是否包含 DLL 的路径（尤其是 libsvnjavahl-1.dll）。
- 对于基于 GNU/Linux 的系统，您可以使用 root 身份运行“ldconfig -v | grep svn”。如果您没有看到相关项，请查阅供应商的文档，了解如何纠正该情形。

启动或重新启动 JDeveloper。

打开 Preferences 对话框（Tools > Preferences），然后在左窗格中选择 Versioning | Subversion。

如果已正确安装 JavaHL 库，您将看到类似下面的信息：



要使用 JavaHL 库，请选择 Native Client 选项。

请注意，如果您以后从 Oracle 更新中心更新 JDeveloper Subversion VCS 扩展，将把上述首选项重设为 JavaSVN。

回页首

附录三：通过代理服务器连接 Subversion 信息库

如果您希望通过代理服务器连接 Subversion 信息库，则必须先安装单独的 Subversion 客户端软件。请参阅安装 Subversion 客户端软件。

一旦您安装了 Subversion 客户端软件，在 Windows Application Data 目录下就会有一个 Subversion 子目录。要查找 Application Data 目录，请在 c:/ 提示符下键入 cd %APPDATA%。然后，打开 Subversion 子目录。（在 Linux 上，等价的子目录将位于 ~/.subversion，其中 ~ 是主目录。）

在 Subversion 中，子目录将是一个名为 servers 的文件。使用文本编辑器打开该文件，然后找到 [global] 区段。从 http-proxy-host 行移除注释标记（#），然后用您使用的代理服务器的详细信息覆盖占位符代理信息。从 http-proxy-port 行移除注释标记（#），然后用代理服务器的端口号覆盖占位符端口信息。如果您希望某些 URL 不使用代理服务器，请从 http-proxy-exceptions 行移除注释标记（#），然后用这些 URL 覆盖占位符 URL。

用您使用的任何其他代理服务器的详细信息添加额外的 http-proxy-host 和 http-proxy-port 行。

重要的是，代理服务器支持 Subversion 使用的所有 http 方法。默认情况下，某些代理服务器不支持以下方法：PROPFIND、REPORT、MERGE、MKACTIVITY 和 CHECKOUT。如果您在使用代理服务器访问 Subversion 信息库时遇到问题，可以要求服务器的系统管理员更改配置，以支持这些 http 方法。

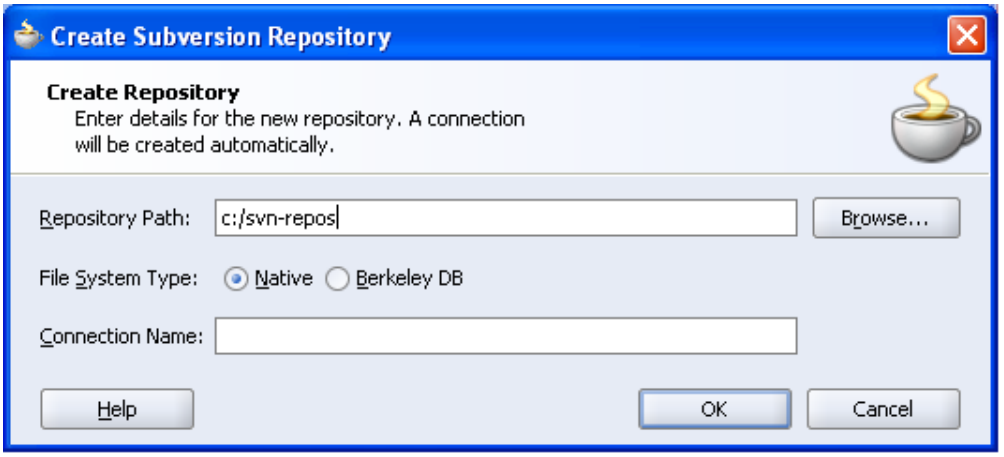
回页首

附录四：创建本地 Subversion 信息库

JDeveloper Subversion VCS 扩展包含一个创建本地 Subversion 信息库的功能。由于该扩展目前提供的支持软件的限制，您必须先安装单独的 Subversion 客户端软件，才能使用 JDeveloper 创建本地信息库。安装 Subversion 客户端软件的说明请见安装 Subversion 客户端软件。

要创建本地 Subversion 信息库，请选择 Versioning > Create Repository。

在 Create Subversion Repository 对话框中，会为您的新信息库推荐一个位置和文件系统类型。为了提供在本指南中使用的示例，我们将信息库路径改写为 c:/svn-repos。您可以将文件系统类型保留为默认设置 Native，不必提供连接名称。



当信息库创建之后，还将自动创建指向它的连接。您可以在 Subversion Navigator 中看到这个连接。

如果您使用 JavaSVN，则需要将访问方法协议从 file:/// 更改为 http://、https://、svn:// 或 svn+ssh://。这些协议将在 Subversion 文档中描述。

> 提示：对于不需要着重考虑安全性的小组或单个开发人员来说，svn:// 协议特别有用，并且易于设置。在命令提示符下输入：
> svnserve -d -r [信息库位置]
> 保持命令窗口打开。当您创建信息库连接时，请使用：
> svn://[计算机名]/

作为信息库位置。

对于本指南中使用的示例，在命令提示符下输入 svnserve -d -r c:/svn-repos/，并保持命令窗口打开。在 Subversion Navigator 中，双击连接名称 file:///c:/svn-repos。在 Edit Subversion Connection 对话框中，用 svn://localhost/ 改写 Repository URL。测试连接，然后单击 OK 关闭对话框。