TortoiseSVN

针对 Windows 平台的 Subversion 客户端

Version 1.8

Stefan Küng
Lübbe Onken
Simon Large

目录

插图清单

表格清单

前言

TortoiseSVN

版本控制是管理信息修改的艺术，它一直是程序员最重要的工具，程序员经常会花时间作出小的修改，然后又在某一天取消了这些修改，想象一下一个开发者并行工作的团队 - 或许是同时工作在同一个文件！- 你就会明白为什么一个好的系统需要管理潜在的混乱。

1.#什么是 TortoiseSVN?

TortoiseSVN 是一个 Windows 下的版本控制系统 Apache™ Subversion® 的客户端工具。就是说，TortoiseSVN 常年管理文件和目录。文件存储于一个中央版本库中。版本库就像一个常见的文件服务器，除了它保存你对文件和目录所有的改变。这一特性使得你可以恢复文件的旧版本并查看历史-谁在什么时间如何进行的修改。这就是为什么很多人认为 Subversion 和版本控制系统是一种“时间机器”。

某些版本控制系统也是软件配置管理(SCM)系统，这种系统经过精巧的设计，专门用来管理源代码树，并且具备许多与软件开发有关的特性 - 比如，对编程语言的支持，或者提供程序构建工具。不过 Subversion 并不是这样的系统；它是一个通用系统，可以管理任何类型的文件集，包括源代码。

2.#TortoiseSVN 的特性

是什么让 TortoiseSVN 成为一个好的 Subversion 客户端？下面是一个简短的特性列表。

外壳集成

TortoiseSVN 无缝地整合进 Windows 的外壳(例如资源管理器)。这意味着你可以继续使用已经熟悉的工具。而且当需要版本控制功能时你不用切换到不同的应用程序。

而且你并没有被限制在 Windows 资源管理器中; TortoiseSVN 的右键菜单可以在很多其它文件管理器中以及标准 Windows 程序的 文件/打开 对话框中被调出。不过，你应该记住 TortoiseSVN 是专门作为 Windows 资源管理器的扩展进行开发的。因此，有可能在其它程序中整合的不那么完整，例如重载图标可能不显示。

重载图标
每个版本控制的文件和目录的状态使用小的重载图标表示，可以让你立刻看出工作副本的状态。

图形用户界面
当你列出文件或文件夹的更改时，你可以点击任意版本查看提交注释。也可以看到更改过的文件列表 - 只要双击文件就可以查看更改内容。

提交对话框列出了本次提交将要包括的条目，每一个条目有一个复选框，所以你可以选择包括哪些条目。未版本控制的文件也会被列出，以防你忘记添加新文件。

Subversion 命令的简便访问
所有的 Subversion 命令存在于资源管理器的右键菜单，TortoiseSVN 在那里添加子菜单。

因为 TortoiseSVN 是一个 Subversion 客户端，我们也很愿意为你展示一些 Subversion 本身的特性:

目录版本控制
CVS 只能追踪单个文件的历史，但是 Subversion 实现了一个“虚拟”文件系统，可以追踪整个目录树的修改，文件和目录都是版本控制的，结果就是可以在客户端对文件和目录执行移动和复制命令。

原子提交
提交要么完全进入版本库，要么一点都没有，这允许开发者以一个逻辑块提交修改。

版本控制的元数据
每个文件和目录都有一组附加的“属性”，你可以发明和保存任意的键/值对，属性是版本控制的，就像文件内容。

可选的网络层
Subversion 在版本库访问方面有一个抽象概念，利于人们去实现新的网络机制，Subversion 的“高级”服务器是 Apache 网络服务器的一个模块，使用 HTTP 的变种协议 WebDAV/DeltaV 通讯，这给了 Subversion 在稳定性和交互性方面很大的好处，可以直接使用服务器的特性，例如认证、授权、传输压缩和版本库浏览等等。也有一个轻型的，单独运行的 Subversion 服务器，这个服务器使用自己的协议，可以轻松的用 SSH 封装。

一致的数据处理
Subversion 使用二进制文件差异算法展现文件的区别，对于文本(人类可读)和二进制(人类不可读)文件具备一致的操作方式，两种类型的文件都压缩存放在版本库中，差异在网络上双向传递。

高效的分支和标签
分支与标签的代价不与工程的大小成比例，Subversion 建立分支与标签时只是复制项目，使用了一种类似于硬链接的机制，因而这类操作通常只会花费很少并且相对固定的时间，以及很小的版本库空间。

3. #许可协议

TortoiseSVN 是一个基于 GNU 通用公共许可协议 (GPL) 开发的开源软件。它可以免费下载和使用，无论是个人或是商业目的，并且没有安装数量的限制。

尽管大多数人只下载安装文件，但你可以取得它的全部源代码。你可以访问以下链接得到源代码: http://code.google.com/p/tortoisesvn/source/browse/。我们当前正在开发的最新的版本放在 /trunk/ 目录下，而已发布版本放在 /tags/ 下。

4. #开发

TortoiseSVN 和 Subversion 均由参与项目工作的社区人员开发。他们来自于遍布全球的各个国家，一起致力于创造伟大的软件。

4.1. #TortoiseSVN 的历史

2002年，Tim Kemp 发现 Subversion 是一个非常好的版本管理系统，但是缺乏一个好的图形界面客户端程序。做一个与 Windows 外壳整合的 Subversion 客户端程序的想法是受一个叫 TortoiseCVS 的 CVS 客户端程序所启发的。Tim 研究了 TortoiseCVS 的源码并以此为 TortoiseSVN 的基础。他开始运作这个项目，注册了域名 tortoisesvn.org 并且将源码放在了网上。

就在同时， Stefan Küng 正在寻找一个好用的并且免费的版本控制系统。他找到了 Subversion 和 TortoiseSVN 的源码。因为 TortoiseSVN 还不能使用，他加入了项目并开始编码。很快，他就重写了现有的大部分代码并开始添加命令和功能，到了某个时段，最初的代码已经都被改写了。

由于 Subversion 变得越来越稳定，它吸引了越来越多用户，他们同时也开始使用 TortoiseSVN 作为 Subversion 的客户端程序。用户数量快速增长(并且每天还在增长)。这时候，Lübbe Onken 提出帮助项目提供精美的图标和 TortoiseSVN 的标志。现在他负责照看网站和管理多语言翻译。

4.2. #致谢

Tim Kemp
启动 TortoiseSVN 项目

Stefan Küng
辛苦工作使 TortoiseSVN 达到现在的样子，并领导整个项目。

Lübbe Onken
制作了漂亮的图标，标志，跟踪错误，翻译并且维护翻译结果

Simon Large
: 维护文档

Stefan Fuhrmann
: 日志缓存和版本图

Subversion 手册
: 为了对 Subversion 大量介绍，我们复制了其第二章

Tigris 样式项目
: 我们在本文重用了一些样式

我们的贡献者
: 提供了补丁、错误汇报及新的想法，并且在邮件列表上回答了其他人的问题

我们的捐赠者
: 他们发送给我们的那些音乐带来了快乐

5. #阅读指南

本手册是为那些想使用 Subversion 来管理数据，并且喜欢使用图形界面客户端程序替代命令行程序的电脑用户编写的。TortoiseSVN 是 Windows 外壳扩展，并且假设用户熟悉和使用 Windows 资源管理器。

在 前言 一章里解释了什么是 TortoiseSVN，一些关于 TortoiseSVN 项目和开发人员社区的消息，以及使用和分发它的许可条件。

在 第 1 章开始 一章里解释了如何在电脑中安装 TortoiseSVN，以及如何立刻开始使用。

在第 2 章 基本版本控制概念一章里简短地介绍了 Subversion 版本控制系统，Subversion 是 TortoiseSVN 的基础。这一章借用了 Subversion 项目的文档，介绍了各种版本控制模式，以及 Subversion 的工作原理。

第 3 章版本库这一章解释了如何设置一个本地版本库，本地版本库对于在一台 PC 上测试 Subversion 和 TortoiseSVN 非常有用，这一章也介绍了一点版本库管理，也就是如何管理服务器上的版本库。如果你需要一台服务器，这里还有一节介绍如何搭建服务器

第 4 章日常使用指南是最重要的章节，介绍了 TortoiseSVN 最主要特性的使用。它以教程的形式，从检出一个工作副本开始，然后修改，提交你的修改，之后进入高级主题。

第 5 章ubWCRev 程序 是 TortoiseSVN 的一个独立程序，可以从工作副本抽取信息并记录到一个文件，可以用来在项目中包含构建信息。

附录 B,如何实现 …这一节回答了一些操作方面的常见问题。这些常见问题在其他章节没有被明确的提到过。

附录 D,TortoiseSVN 操作 这一节展示了如何使用命令行调用 TortoiseSVN 的 GUI 对话框，当你在使用脚本时仍希望用户交互时非常有用。

附录 E,命令行交叉索引给出了 TortoiseSVN 命令与其对应的 Subversion 命令行工具 svn.exe 命令之间的关系。

6. #本文使用的术语

为了使文档更加易读，所有 TortoiseSVN 的窗口名和菜单名使用不同的字体，例如日志对话框。

菜单选择使用箭头显示。TortoiseSVN → 显示日志的含义是：从 TortoiseSVN 右键菜单选择显示日志。

在 TortoiseSVN 对话框中出现的右键菜单，可能是这个样子：右键菜单 → 另存为 ...

用户界面按钮的显示形式：点击OK以继续。

用户动作使用粗体字表示。Alt+A：表示按住键盘上的 Alt 键同时按下 A 键。右键拖拽：表示按住鼠标右键同时拖拽条目到新位置。

系统输出和键盘输入也使用不同的字体显示。

使用图标标记的重要提示。

技巧让你的生活更加简单。

操作时需要小心的地方。

需要非常小心操作的地方。如果忽略警告内容将会导致数据损坏或其它严重后果。

第#1#章#开始

这一节面向那些想要了解 TortoiseSVN 的用途并且想试用一下它的人。介绍了如何安装 TortoiseSVN 及设置本地版本库，还将展示最常用的操作。

1.1.#安装 TortoiseSVN

1.1.1.#系统要求

TortoiseSVN 运行于 Windows XP SP3 或更高版本，并支持32位和64位系统。64位的安装程序也包含了32位的扩展部件。这意味着你不需要额外安装32位版就可以在32位程序中使用 TortoiseSVN 右键菜单和图标重载。

如果你使用 Windows XP，你必须安装 SP3 或以上版本的补丁。如果没有安装该 SP 补丁会导致 TortoiseSVN 无法运行。

自 1.2.0 版起放弃对 Windows 98， Windows ME 和 Windows NT 的支持。自 1.7.0 版起放弃对 Windows 2000 和 XP SP2 的支持。如果需要，仍然可以下载、安装旧版本。

1.1.2.#安装

TortoiseSVN 以简单易用的安装包的形式发布。双击安装文件并按照提示操作。安装文件会照顾其余的事情。安装结束后不要忘记重启电脑。

你需要管理员权限来安装 TortoiseSVN。

TortoiseSVN 提供语言补丁(Language pack)，它可以将用户界面翻译成多种文字。请查看 附录 G,语言包和拼写检查器 获得更多关于安装语言补丁的信息。

如果在安装 TortoiseSVN 过程中或结束后遇到任何问题，情参考我们的在线常见问题: http://tortoisesvn.net/faq.html

1.2.#基本概念

Before we get stuck into working with some real files, it is important to get an overview of how Subversion works and the terms that are used.

版本库
: Subversion 使用集中的数据库，它包含了所有的版本控制文件及其完整历史。这个数据库就是版本库。版本库通常位于运行 Subversion 服务器的文件服务器上，向 Subversion 客户端(例如 TortoiseSVN)提供需要的数据。如果只备份一个东西，请备份版本库，因为它是你数据的主副本。

工作副本
: 这是实际工作的地方。每一个开发者在自己的电脑上都有属于自己的工作副本，有时可以将其理解为沙箱。你可以将最新的版本从版本库上取下来，在本地的副本上工作而不影响其他人，如果对更改满意就可以将其提交到版本库中。

  Subversion 工作副本不包含项目的历史， 但是它保存了你修改前的本件的副本，就像这些文件在版本库中的状态一样。这意味着你可以轻而易举的准确检查出都做了哪些改动。

你还要知道从哪里开始运行 TortoiseSVN，因为在开始菜单中看不到。这是因为 TortoiseSVN 是一个外壳扩展，所以第一步，打开 Windows 资源管理器。在资源管理器中用右键单击一个文件夹，然后就会发现在右键菜单中出现一些新的条目，就像这样:

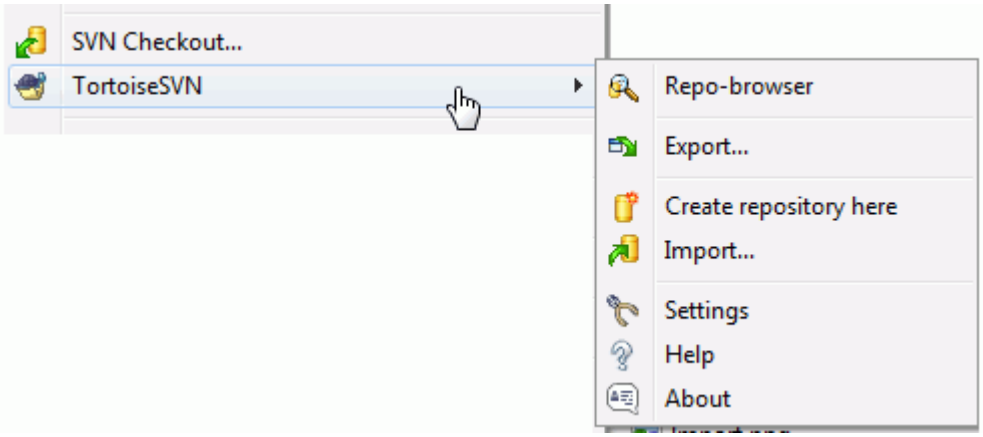


图 1.1. 未版本控制文件夹的 TortoiseSVN 菜单

1.3. #开始试用

这一节通过一个小的实验版本库向你展示如何开始一些最普通的用途。实际上这里没有介绍全部功能 - 这只是一个快速开始向导而已。一旦你开始使用，你应该花点时间读一下本手册的其它部分，在那里将会详细地介绍本软件的方方面面。也会介绍更多关于设置一个全功能 Subversion 服务器的内容。

1.3.1. #创建版本库

对于一个实际的项目，需要在一个安全的地方创建版本库并设置 Subversion 服务器来控制它。而对于本教程而言，我们将会使用 Subversion 的本机版本库功能，该特性使得用户可以直接访问本机硬盘上的版本库而不需要服务器。

First create a new empty directory on your PC. It can go anywhere, but in this tutorial we are going to call it C:\svn_repos. Now right click on the new folder and from the context menu choose TortoiseSVN → Create Repository here.... The repository is then created inside the folder, ready for you to use. We will also create the default internal folder structure by clicking the Create folder structure button.

对于测试和评估用途来说，本机版本库功能非常有用，但除非你是一个只使用自己电脑进行独自工作的开发者，否则你应该使用全功能的 Subversion 服务器。有些小公司为了避免设置服务器的工作，就通过网络共享来存取版本库。不要这样做。这样会丢失数据。阅读 第 3.1.4 节 “访问网络共享磁盘上的版本库”了解为什么这是一个坏主意，以及如何设置服务器。

1.3.2. #导入项目

Now we have a repository, but it is completely empty at the moment. Let's assume I have a set of files in C:\Projects\Widget1 that I would like to add. Navigate to the Widget1 folder in Explorer and right click on it. Now select TortoiseSVN → Import... which brings up a dialog

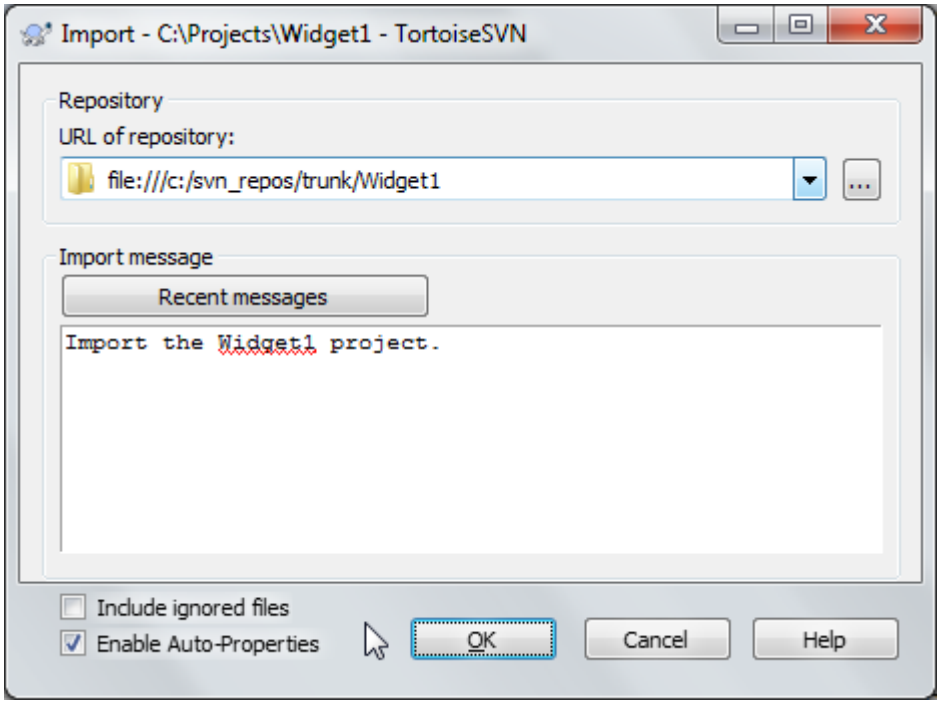


图 1.2. 导入对话框

A Subversion repository is referred to by URL, which allows us to specify a repository anywhere on the Internet. In this case we need to point to our own local repository which has a URL of file:///c:/svn_repos/trunk, and to which we add our own project name Widget1. Note that there are 3 slashes after file: and that forward slashes are used throughout.

这个对话框中另一个重要的功能就是导入信息文本框，你可以输入一段信息来描述你的操作。当你查看项目的历史时，这些提交信息是记录了你修改什么和为什么修改的宝贵资料。在这个例子中，我们可以简单的写一下，例如“导入 Widget1 项目”。点击确定，文件夹就加入到版本库中了。

1.3.3. #检出工作副本

Now that we have a project in our repository, we need to create a working copy to use for day-to-day work. Note that the act of importing a folder does not automatically turn that folder into a working copy. The Subversion term for creating a fresh working copy is Checkout. We are going to checkout the Widget1 folder of our repository into a development folder on the PC called C:\Projects\Widget1-Dev. Create that folder, then right click on it and select TortoiseSVN → Checkout.... Enter the URL to checkout, in this case file:///c:/svn_repos/trunk/Widget1 and click on OK. Our development folder is then populated with files from the repository.

你会注意到，这个文件夹看起来与我们原来的文件夹不一样。每一个文件的左下角都有一个绿色的对钩。它们就是只出现在工作副本中的 TortoiseSVN 状态图标。绿色的图标表示文件未被修改，和版本库中的文件版本一致。

1.3.4. #进行修改

工作时间到了。 在 Widget-Dev 中，我们开始编辑文件 - 假设我们对 Widget1.c 和 ReadMe.txt 进行了修改。注意，这些文件的重载图标变成了红色，这说明本机文件被修改了。

但是我们做了哪些更改？右键单击任意一个修改过的文件然后选择 TortoiseSVN → 比较差异。启动 TortoiseSVN 的文件比较工具，准确地显示哪些行被修改了。

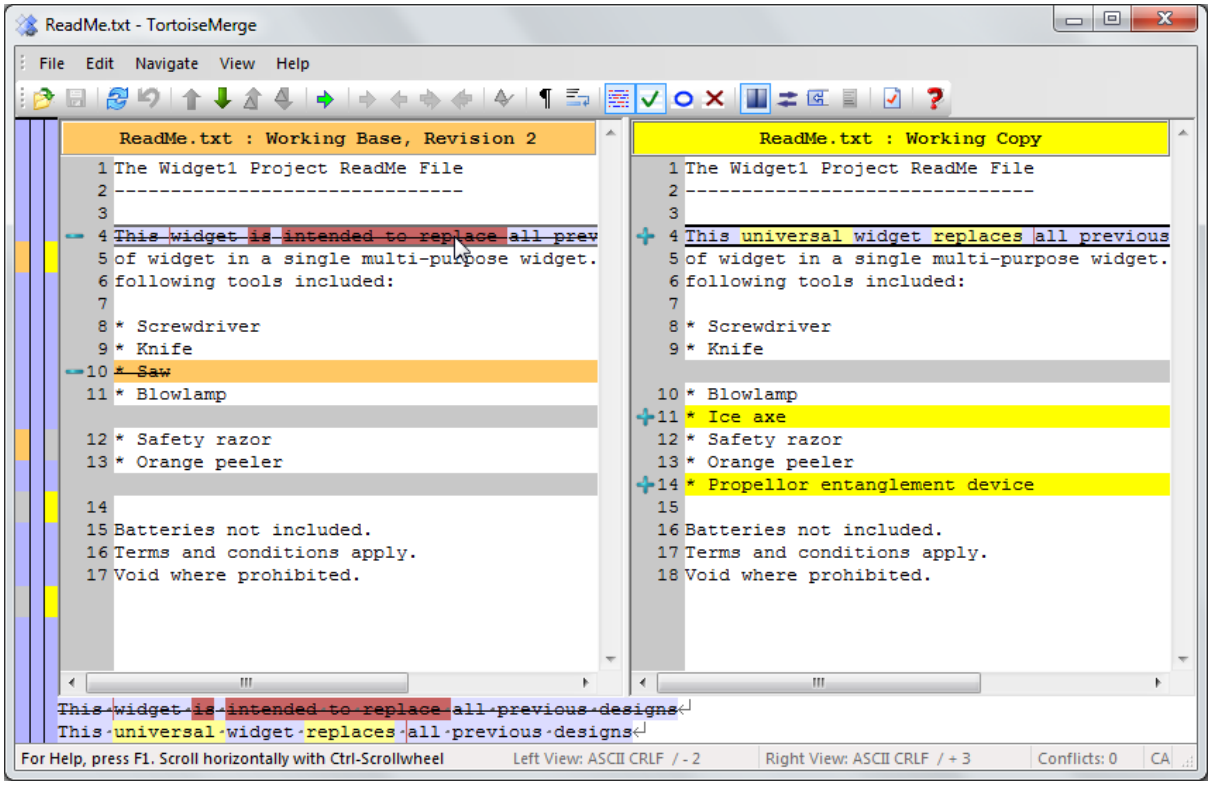


图 1.3. 文件差异查看器

好的，我们对这些更改很满意，让我们更新版本库。这个动作叫 提交 更改。右键单击文件夹 Widget1-Dev 然后选择 TortoiseSVN → 提交。提交对话框列出了修改过的文件，每一个都有一个复选框。你可以选中列表中的部分文件，但在这个例子中我们将要提交全部修改过文件。输入一段信息来描述做了什么修改然后单击 确定。进度对话框显示被上传到版本库中的文件并且完成提交。

1.3.5. #添加更多的文件

随着项目的发展，需要添加新文件 - 例如说你要添加新功能，有了新文件 Extras.c 还要在现有的文件 Makefile 中加入起对该文件的引用。右键单击文件夹然后选择 TortoiseSVN → 增加。 加入对话框显示了所有未被版本控制的文件，你可以选择哪些文件要被添加。另一个增加文件的方法是右键单击文件自身然后选择 TortoiseSVN → 加入。

现在当你提交文件夹时，新文件会显示为增加，原有的文件显示为修改。注意你可以双击修改的文件查看做了哪些改动。

1.3.6. #查看项目历史

TortoiseSVN 最有用的功能之一便是日志对话框。它显示针对文件或文件夹所有的提交记录列表，并显示提交者输入的那些详细的提交信息。 ;-)

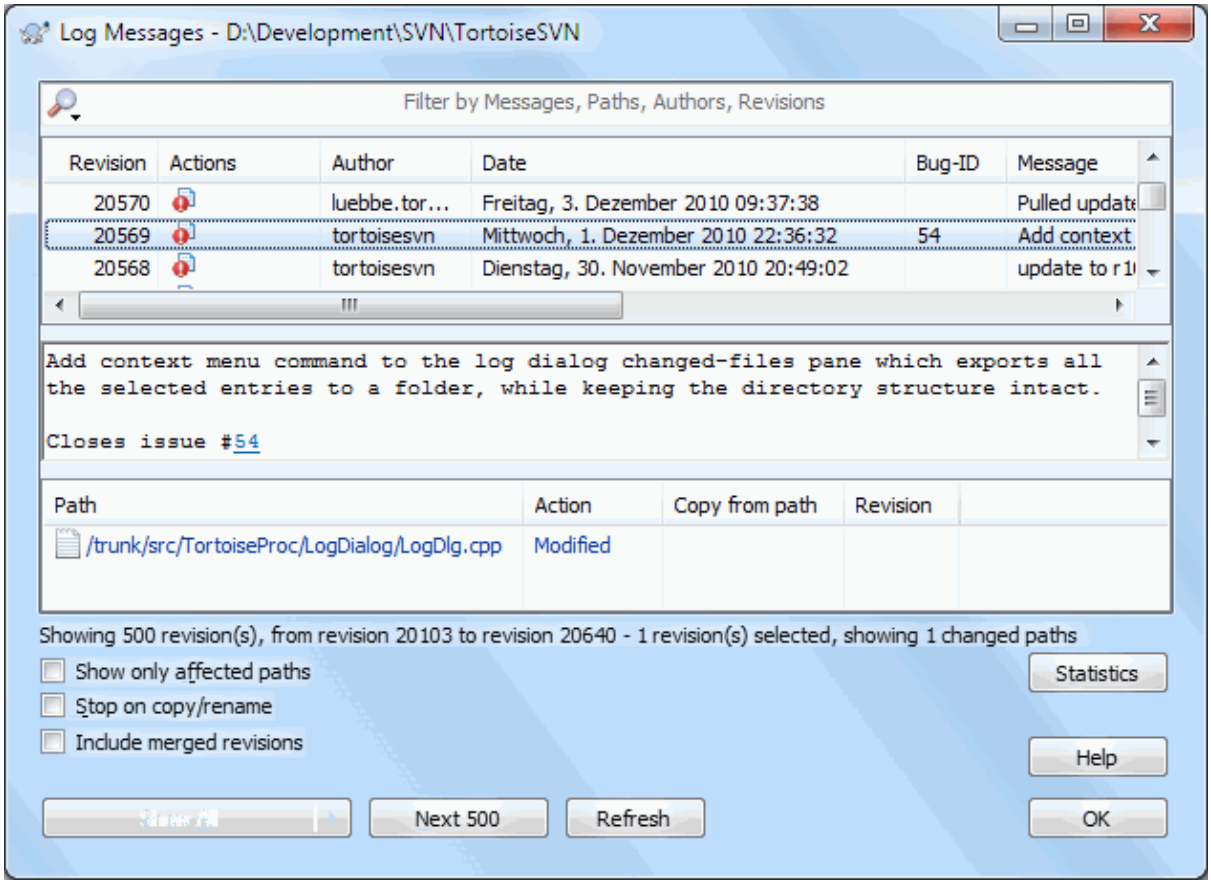


图 1.4. 日志对话框

好吧，我在这里小小的作弊一下，使用一张显示 TortoiseSVN 版本库的屏幕截图。

对话框的上部显示提交的版本列表以及提交信息的开头。如果选中版本列表中的任意一个，对话框的中部显示该版本完整的日志消息，对话框的底部显示更改的文件和文件夹列表。

对话框的每一部分都有右键菜单提供很多使用这些信息的方法。在底部可以 双击 文件来完整的查看该版本所做的修改。参阅 第 4.9 节 “版本日志对话框”获得完整的内容。

1.3.7. #撤消更改

所有的版本控制系统都有的功能就是让你可以撤销之前做的更改。如你预料的一样，TortoiseSVN 非常容易做到这一点。

如果你想抛弃还没有提交的更改并将文件复原到修改之前的状态，TortoiseSVN → SVN 还原 就是你的好伙伴。它抛弃了所做的更改(扔到回收站里)并复原到修改之前的版本。如果你想抛弃更改的一部分，可以使用 TortoiseMerge 来查看区别并有选择的复原被修改的文件行。

如果你要撤销某一个特定版本的影响，启动日志对话框，找到不想要的版本。选择 右键菜单 → 复原此版本作出的修改 然后这些更改就会被撤销。

1.4. #继续前进 . . .

这个向导提供一个非常短小的例子来展示 TortoiseSVN 的最重要和最有用的功能，当然，这里很有很多内容没有被覆盖。我们强烈推荐你花点时间阅读这本手册剩余的部分，尤其是 第 4 章日常使用指南，这一章展示了很多日常工作的详细内容。

我们已经不辞劳苦的确保本手册有丰富的信息并容易阅读，但我们也承认在使用过程中还有很多的困难！花点时间，独自用测试版本库练练手。最好的方法就是在实践中学习。

第#2#章#基本版本控制概念

本章修改自《使用 Subversion 进行版本管理》的相同章节，它的在线版本位于：http://svnbook.red-bean.com/。

这一章是对 Subversion 一个简短随意的介绍，如果你对版本控制很陌生，这一章节完全是为你准备的，我们从讨论基本概念开始，深入理解 Subversion 的思想，然后展示许多简单的实例。

尽管我们的例子展示了人们如何分享程序源代码，仍然要记住 Subversion 可以控制所有类型的文件－它并没有限制只为程序员工作。

2.1. #版本库

Subversion 是一种集中的分享信息的系统，它的核心是版本库，储存所有的数据，版本库按照文件树形式储存数据－包括文件和目录，任意数量的客户端可以连接到版本库，读写这些文件。通过写数据，别人可以看到这些信息；通过读数据，可以看到别人的修改。

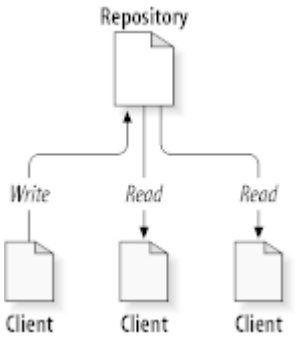


图 2.1. 一个典型的客户/服务器系统

所以为什么这很有趣呢？讲了这么多，让人感觉这是一种普通的文件服务器，但实际上，版本库是另一种文件服务器，而不是你常见的那一种。最特别的是 Subversion 会记录每一次的更改，不仅针对文件也包括目录本身，包括增加、删除和重新组织文件和目录。

当一个客户端从版本库读取数据时，通常只会看到文件系统的最新版本，但是客户端也可以去看以前的任何一个版本。举个例子，一个客户端可以问这样的“历史问题”——“上个星期三的目录是怎样的？”，或者“是谁最后一个修改了这个文件，改动了什么？”这些问题就是所有版本控制系统——用来记录和跟踪数据随时间修改的系统——的核心问题。

2.2. #版本模型

所有的版本控制系统都需要解决这样一个基础问题：怎样让系统允许用户共享信息，而不会让他们因意外而互相干扰？版本库里意外覆盖别人的更改非常的容易。

2.2.1. #文件共享的问题

考虑这个情景，我们有两个共同工作者，Harry 和 Sally，他们想同时编辑版本库里的同一个文件，如果首先 Harry 保存它的修改，过了一会，Sally 可能凑巧用自己的版本覆盖了这些文件，Harry 的更改不会永远消失(因为系统记录了每次修改)，Harry 所有的修改不会出现在 Sally 的文件中，所以 Harry 的工作还是丢失了—至少是从最新的版本中丢失了—而且是意外的，这就是我们要明确避免的情况！

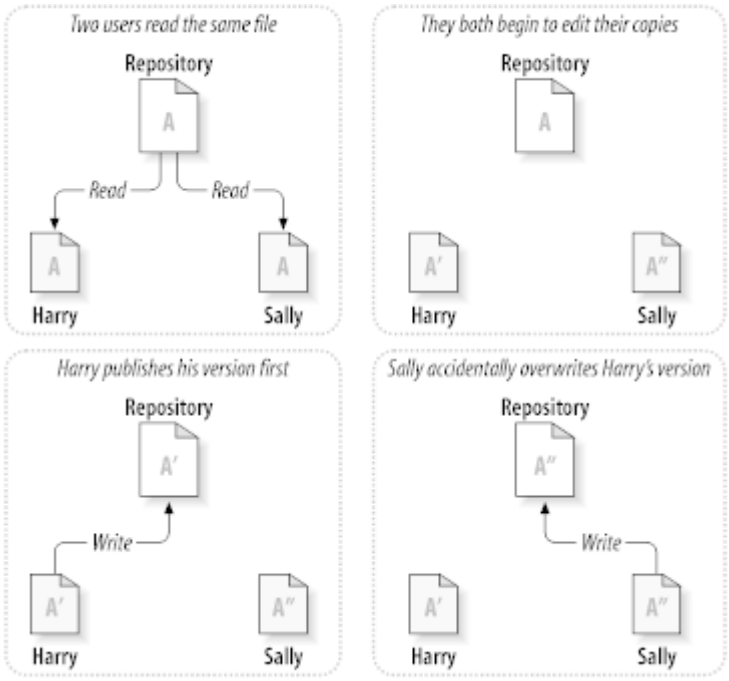


图 2.2. 需要避免的问题

2.2.2. #锁定-修改-解锁 方案

许多版本控系统使用 锁定-修改-解锁 模型来解决这个问题，这是一个简单的解决方案。在这种系统中，在同一时间版本库只允许一个用户修改一个文件。首先，Barry 必须在修改前 锁定 该文件。锁定文件有点像从图书馆结束; 如果 Harry 锁定了一个文件，那么 Sally 就修改该文件。如果她试图锁定该文件，版本库会拒绝这个请求。她只能读取这个文件，并等待 Harry 结束修改并释放文件锁。在 Harry 解锁文件后，他的任务就完成了，现在 Sally 可以接手工作 - 锁定并编辑文件。

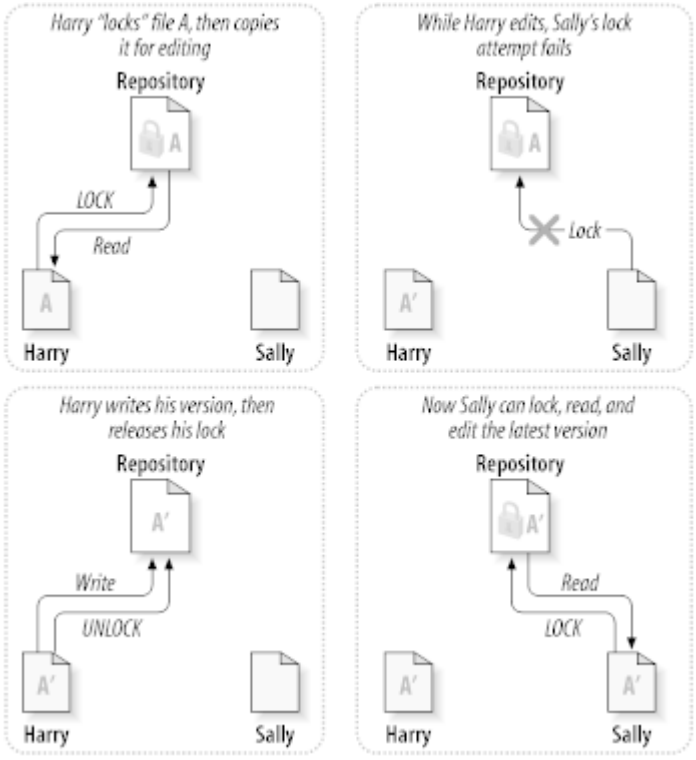


图 2.3. 锁定-修改-解锁 方案

锁定-修改-解锁模型有一点问题就是限制太多，经常会成为用户的障碍:

- 锁定可能导致管理问题。有时候 Harry 会锁住文件然后忘了此事，这就是说 Sally 一直等待解锁来编辑这些文件，她在这里僵住了。然后 Harry 去旅行了，现在 Sally 只好去找管理员放开锁，这种情况会导致不必要的耽搁和时间浪费。

- 锁定可能导致不必要的线性化开发。如果 Harry 编辑一个文件的开始，Sally 想编辑同一个文件的结尾，这种修改不会冲突，设想修改可以正确的合并到一起，他们可以轻松的并行工作而没有太多的坏处，没有必要让他们轮流工作。

- 锁定可能导致错误的安全状态。假设 Harry 锁定和编辑一个文件 A，同时 Sally 锁定并编辑文件 B，如果 A 和 B 互相依赖，这种变化是必须同时作的，这样 A 和 B 不能正确的工作了，锁定机制对防止此类问题将无能为力—从而产生了一种处于安全状态的假相。很容易想象 Harry 和 Sally 都以为自己锁住了文件，而且从一个安全，孤立的情况开始工作，因而没有尽早发现他们不匹配的修改。

2.2.3. #复制-修改-合并 方案

Subversion，CVS 和一些版本控制系统使用 复制-修改-合并 模型，在这种模型里，每一个客户读取项目版本库建立一个私有工作副本 - 版本库中文件和目录的本地映射。用户并行工作，修改各自的工作副本，最终，各个私有的复制合并在一起，成为最终的版本，这种系统通常可以辅助合并操作，但是最终要靠人工去确定正误。

这儿有一个例子。比如说 Harry 和 Sally 参加同一个项目每人都有各自的工作副本，从同一个版本库复制出来的。他们同时工作，在自己的副本中修改同一个文件 A。Sally 先将她的更改保存到版本库中。 稍后，当 Harry 尝试提交他的更改时，版本库提示他的文件 A 已经过时。换句话说，自从他上次复制文件后，无论如何版本库中的文件 A 已经被修改了。所以 Harry 要用客户端程序将版本库中文件 A 的新更改 合并 到他的工作副本中。碰巧的是 Sally 的更改和他的不重合; 所以一旦他整合了两人的更改，他就可以把他的工作副本复制回版本库。

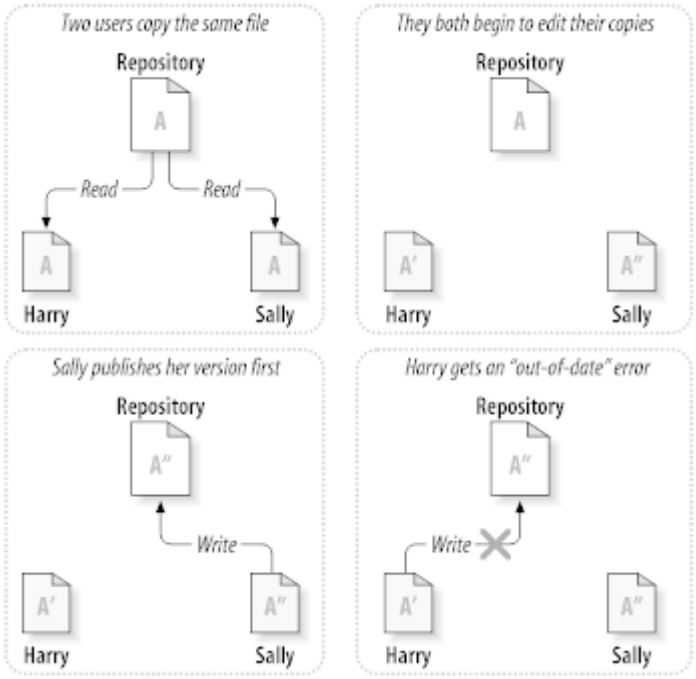


图 2.4. 复制-修改-合并 方案

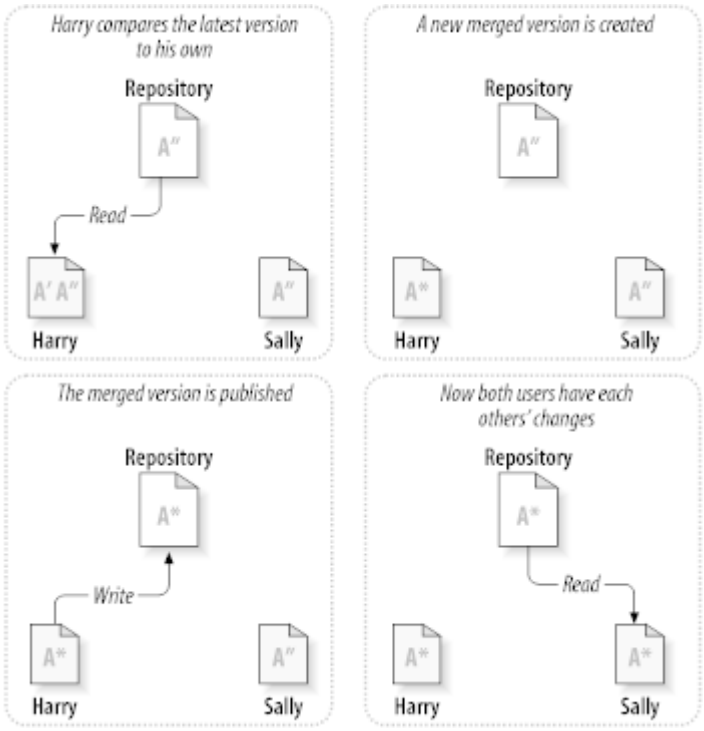


图 2.5. 复制-修改-合并 方案(续)

但是如果 Sally 和 Harry 的修改重叠了该怎么办？这种情况叫做冲突，这通常不是个大问题，当 Harry 告诉他的客户端去合并版本库的最新修改到自己的工作副本时，他的文件 A 就会处于冲突状态：他可以看到一对冲突的修改集，并手工的选择保留一组修改。需要注意的是软件不能自动的解决冲突，只有人可以理解并作出智能的选择，一旦 Harry 手工的解决了冲突(也许需要与 Sally 讨论)，他就可以安全的把合并的文件保存到版本库。

复制-修改-合并模型感觉是有一点混乱，但在实践中，通常运行的很平稳，用户可以并行的工作，不必等待别人，当工作在同一个文件上时，也很少会有重叠发生，冲突并不频繁，处理冲突的时间远比等待解锁花费的时间少。

最后，一切都要归结到一条重要的因素：用户交流。当用户交流贫乏，语法和语义的冲突就会增加，没有系统可以强制用户完美的交流，没有系统可以检测语义上的冲突，所以没有任何证据能够承诺锁定系统可以防止冲突，实践中，锁定除了约束了生产力，并没有做什么事。

有一种情况下锁定-修改-解锁模型会更好，也就是你有不可合并的文件，例如你的版本库包含了图片，两个人同时编辑这个文件，没有办法将这两个修改合并，Harry 或 Sally 会丢失他们的修改。

2.2.4.#Subversion 怎么做？

Subversion 缺省使用复制-修改-合并模型，大多数情况下可以满足你的需求。然而，Subversion 1.2 后还是支持锁定，如果你有不可合并的文件，或者你只是想实行强制管理策略，Subversion 仍然会提供你需要的特性。

2.3.#Subversion 实战

2.3.1.#工作副本

你已经阅读过了关于工作副本的内容，现在我们要讲一讲客户端怎样建立和使用它。

一个 Subversion 工作副本是你本地机器一个普通的目录，保存着一些文件，你可以任意的编辑文件，而且如果是源代码文件，你可以像平常一样编译，你的工作副本是你的私有工作区，在你明确的做了特定操作之前，Subversion 不会把你的修改与其他人的合并，也不会把你的修改展示给别人。

当对工作副本中的文件做了一些更改并确认他们能够正常工作后，Subversion 提供将这些更改公布给同项目的其他人员的命令(通过写入版本库)。如果其他人公布他们的更改，Subversion 提供将这些更改合并到工作副本的命令(通过读取本版本库)。

A working copy also contains some extra files, created and maintained by Subversion, to help it carry out these commands. In particular, your working copy contains a subdirectory named .svn, also known as the working copy administrative directory. The files in this administrative directory help Subversion recognize which files contain unpublished changes, and which files are out-of-date with respect to others' work. Prior to 1.7 Subversion maintained .svn administrative subdirectories in every versioned directory of your working copy. Subversion 1.7 takes a completely different approach and each working copy now has only one administrative subdirectory which is an immediate child of the root of that working copy.

一个典型的 Subversion 的版本库经常包含许多项目的文件(或者说源代码)，通常每一个项目都是版本库的子目录，在这种安排下，一个用户的工作副本往往对应版本库的的一个子目录。

举一个例子，你的版本库包含两个软件项目。

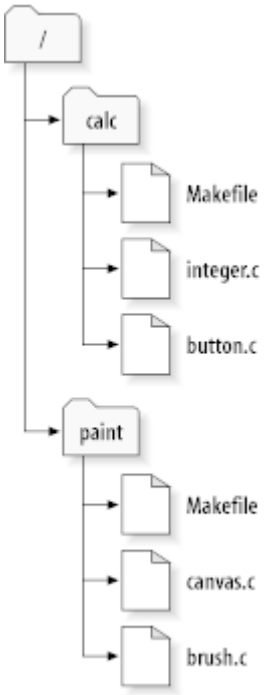


图 2.6. 版本库的文件系统

换句话说，版本库的根目录包含两个子目录: paint 和 calc。

要获得工作副本，必须通过 检出 版本库中的某个子树。(术语 检出 听起来好像会锁定和保留资源，但实际上不是这样; 它只是给项目创建了一个私有的副本。)

假定你修改了 button.c，因为 .svn 目录记录着文件的修改日期和原始内容，Subversion 可以告诉你已经修改了文件，然而，在你明确告诉它之前，Subversion 不会将你的改变公开。将改变公开的操作被叫做提交(或者是检入)，它提交修改到版本库中。

发布你的修改给别人，可以使用 Subversion 的提交命令。

这时你对 button.c 的修改已经提交到了版本库，如果其他人取出了 /calc 的一个工作副本，他们会看到这个文件最新的版本。

设你有个合作者，Sally，她和你同时取出了 /calc 的一个工作副本，你提交了你对 button.c 的修改，Sally 的工作副本并没有改变，Subversion 只在用户要求的时候才改变工作副本。

要使项目最新，Sally 可以要求 Subversion 更新她的工作副本，通过使用更新命令，可以将你和所有其他人在她上次更新之后的修改合并到她的工作副本。

注意，Sally 不必指定要更新的文件，Subversion 利用 .svn 以及版本库的进一步信息决定哪些文件需要更新。

2.3.2. #版本库的 URL

Subversion 可以通过多种方式访问－本地磁盘访问，或各种各样不同的网络协议，但一个版本库地址永远都是一个 URL，URL 方案反映了访问方法。

方案	访问方法
file://	直接版本库访问(本地磁盘或者网络磁盘)。
http://	通过 WebDAV 协议访问支持 Subversion 的 Apache 服务器。
https://	与 http:// 相似，但是用 SSL 加密。
svn://	通过未认证的 TCP/IP 自定义协议访问 svnserve 服务器。
svn+ssh://	通过认证并加密的 TCP/IP 自定义协议访问 svnserve 服务器。

表 2.1. 版本库访问 URL

对于大多数情况，Subversion 的 URL 使用标准格式，允许服务器名称和端口作为 URL 的一部分明确的指出来。files:// 访问方式一般用于本地访问，尽管它可以使用 UNC 路径来引用网络主机。因此 URL 使用这种格式 file://hostname/path/to/repos。对于本机而言，URL 中的 hostname 部分必须省略或者使用 localhost。就是因为这个原因，本机路径通常含有 3 个斜线，file:///path/to/repos。

同样，在 Windows 平台上使用 fiel:// 方案的用户需要使用一种非官方的 “标准” 格式来访问哪些位于本机但是和客户端当前运行的硬盘分区不同的分区中的版本库。

```
file:///X:/path/to/repos
...
file:///X|/path/to/repos
...
```

注意 URL 使用普通的斜杠，而不是 Windows 本地(非 URL)形式的路径。

你可以安全的访问网络共享的 FSFS 版本库，但是你不能以这种方式访问 BDB 版本库。

不要创建和访问网络共享上的 Berkeley DB 版本库，它不能存在于一个远程的文件系统，即使是映射到盘符的共享。如果你希望在网络共享使用 Berkeley DB，结果难以预料－你可能会立刻看到奇怪的错误，也有可能几个月之后才发现数据库已经损坏了。

2.3.3. #修订版本

svn commit 操作可以作为一个原子事务操作发布任意数量文件和目录的修改。在你的工作副本中，你可以改变文件内容，创建、删除、改名和复制文件和目录，然后作为一个整体提交。

在版本库中，每次提交被当作一次原子事务操作: 要么所有的改变发生，要么都不发生，Subversion 努力保持原子性以应对程序错误、系统错误、网络问题和其他用户行为。

每当版本库接受了一个提交，文件系统进入了一个新的状态，叫做版本，每个版本被赋予一个独一无二的自然数，一个比一个大，初始修订号是 0，只创建了一个空目录，没有任何内容。

可以形象的把版本库看作一系列树，想象有一组版本号，从 0 开始，从左到右，每一个修订号有一个目录树挂在它下面，每一个树好像是一次提交后的版本库“快照”。

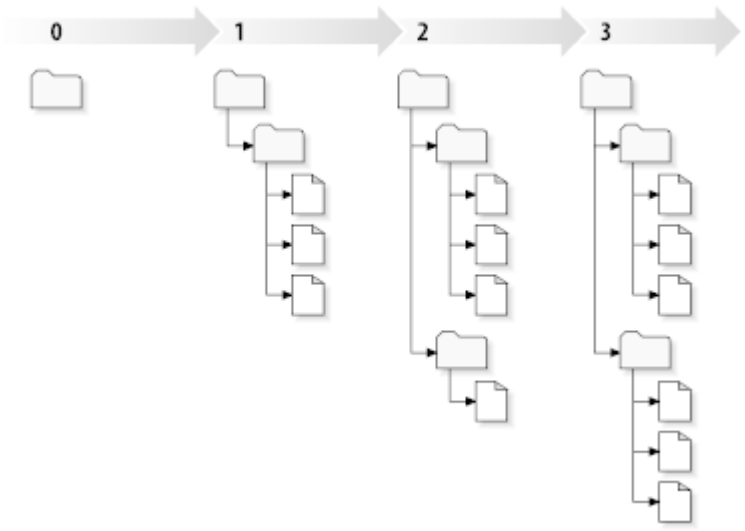


图 2.7. 版本库

全局版本号

不像其它版本控制系统，Subversion 的版本号是针对整个目录树的，而不是单个文件。每一个版本号代表了一次提交后版本库整个目录树的特定状态，另一种理解是版本 N 代表版本库已经经过了 N 次提交。当 Subversion 用户讨论“foo.c的版本 5”时，他们的实际意思是“在版本 5 时的foo.c”。需要注意的是，一个文件的版本 N 和 M 并不表示它必定不同。

需要特别注意的是，工作副本并不一定对应版本库中的单一版本，他们可能包含多个版本的文件。举个例子，你从版本库检出一个工作副本，最新的版本是 4:

```
calc/Makefile:4
     integer.c:4
     button.c:4
```

此刻，工作目录与版本库的版本 4 完全对应，然而，你修改了 button.c 并且提交之后，假设没有别的提交出现，你的提交会在版本库建立版本 5，你的工作副本会是这个样子的:

```
calc/Makefile:4
     integer.c:4
     button.c:5
```

假设此刻，Sally 提交了对 integer.c 的修改，建立修订版本 6，如果你使用 svn update 来更新你的工作副本，你会看到:

```
calc/Makefile:6
     integer.c:6
     button.c:6
```

Sally 对 integer.c 的改变会出现在你的工作副本，你对 button.c 的改变还在，在这个例子里，Makefile 在 4、5、6 版本都是一样的，但是 Subversion 会把 Makefile 的版本设为 6 来表明它是最新的，所以你在工作副本顶级目录作一次干净的更新，会使所有内容对应版本库的同一修订版本。

2.3.4. #工作副本怎样跟踪版本库

对于工作副本的每一个文件，Subversion 在管理目录 .svn/ 记录两项关键的信息:

- 工作文件的基准版本(叫做文件的工作版本)和
- 一个本地副本最后更新的时间戳。

给定这些信息，通过与版本库通讯，Subversion 可以告诉我们工作文件是处与如下四种状态的那一种:

未修改且是当前的
: 文件在工作目录里没有修改，在工作版本之后没有修改提交到版本库。svn commit 操作不做任何事情，svn update 不做任何事情。

本地已修改且是当前的
: 工作副本已经修改，从基准版本之后没有修改提交到版本库。本地修改没有提交，因此 commit 会成功的提交，update 不做任何事情。

本地未修改且过时
: 这个文件在工作副本没有修改，但在版本库中已经修改了。这个文件应当更新到最新公共版本。commit 不做任何事情，update 将会更新工作副本到最新的版本。

本地已修改且过时
: 这个文件在工作副本和版本库中都被修改了。提交 该文件将会因为 过时 而失败。该文件应该先更新; 更新 命令将会尝试合并公共更改和本机更改。如果 Subversion 不能顺利的自动完成合并，则需要用户解决冲突。

2.4. #摘要

我们在这一章里学习了许多 Subversion 基本概念:

- 我们介绍了中央版本库、客户工作副本和版本库中版本树队列的概念。
- 我们介绍了两个协作者如何使用使用“复制-修改-合并”模型，用 Subversion 发布和获得对方的修改。
- 我们讨论了一些 Subversion 跟踪和管理工作副本信息的方式。

第#3#章#版本库

无论你用什么协议访问你的版本库，都至少需要创建一个版本库，这可以使用Subversion命令行客户端或TortoiseSVN完成。

如果你还没有创建Subversion版本库，是时间开始了。

3.1.#创建版本库

You can create a repository with the FSFS backend or with the older Berkeley Database (BDB) format. The FSFS format is generally faster and easier to administer, and it works on network shares and Windows 98 without problems. The BDB format was once considered more stable simply because it has been in use for longer, but since FSFS has now been in use in the field for several years, that argument is now rather weak. Read Choosing a Data Store [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.planning.html#svn.reposadmin.basics.backends] in the Subversion book for more information.

3.1.1.#使用命令行工具创建版本库

1. 创建一个名为SVN(例如D:\SVN\)的空文件夹，作为你的所有版本库的根。
2. 在D:\SVN\里创建另一个目录MyNewRepository。
3. 打开命令行窗口(或DOS窗口)，进入D:\SVN\目录，输入

```
svnadmin create --fs-type bdb MyNewRepository
```

或

```
svnadmin create --fs-type fsfs MyNewRepository
```

现在你在D:\SVN\MyNewRepository创建了一个新的版本库。

3.1.2.#使用 TortoiseSVN 创建版本库

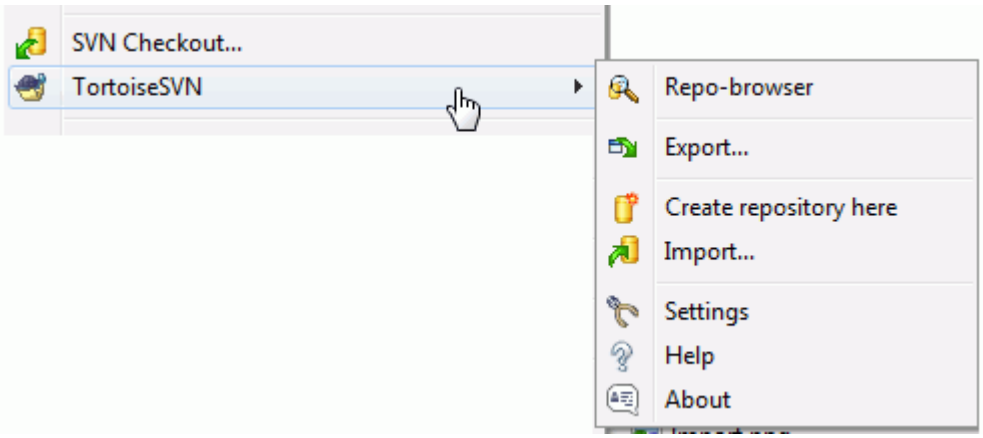


图 3.1. 未版本控制文件夹的 TortoiseSVN 菜单

1. 打开资源管理器
2. 创建一个新的文件夹，命名为SVNRepository

3. 右键单击 新建的文件夹并选择 TortoiseSVN → 在此创建版本库...。

   然后就会在新文件夹创建一个版本库，不要手工编辑这些文件！！！如果你得到什么警告，一定要先确定目录非空并且没有写保护。

   你会被询问是否要在版本库中创建目录结构。要获得关于目录结构的选项情参阅 第 3.1.5 节 “版本库布局”。

   TortoiseSVN 将会在创建版本库时为其设置一个特定的文件夹图标，便于辨别本地版本库。如果使用官方的命令行客户端创建版本库则不会设置文件夹图标。

TortoiseSVN 不再给你创建 BDB 版本库的选择，尽管你仍旧可以使用命令行工具创建。FSFS 版本库通常很容易维护，也让我们维护 TortoiseSVN 变得更容易，因为我们不再需要处理不同 BDB 版本之间的兼容性问题。

因为兼容性问题，TortoiseSVN 不支持使用 file:// 方式访问 BDB 版本库。不过通过 svn:// , http:// 或 https:// 协议访问位于服务器端的 BDB 格式版本库肯定是长期支持的。

当然，我们推荐你根本就不要使用 field:// 访问，除非是基于本机测试的目的。对于除独立开发者以外的人而言，使用服务器更安全更可靠。

3.1.3. #本地访问版本库

为了访问本地版本库，你需要这个文件夹的路径，只要记住Subversion期望所有的版本库路径使用的形式为file:///C:/SVNRepository/，请注意全部使用的是斜杠。

为了访问网络共享中的版本库，你可以使用驱动器影射或使用UNC路径，对于UNC路径，形式为file://ServerName/path/to/repos/，请注意这里前面只有两个斜杠。

在SVN 1.2之前，UNC路径曾经是一种非常晦涩的格式file:///\ServerName/path/to/repos，这种格式依然支持，但不推荐。

不要创建和访问网络共享上的伯克利数据库 (BDB) 版本库。它不能存在于远程文件系统。即使是映射到盘符的共享。如果你尝试在网络共享使用伯克利数据库 (BDB)，结果难以预料 — 你可能会立刻看到奇怪的错误，也有可能几个月之后才发现数据库已经损坏了。

3.1.4. #访问网络共享磁盘上的版本库

尽管从理论上说，将一个 FSFS 格式的版本库放在网络中共享，并且多用户通过 file:// 协议访问是可以的。但这样做是非常不妥当的。事实上我们强烈反对这样做，并且不支持这样的用法。

首先，这样赋予所有用户对版本库的写权限，所以任何一个用户都可能意外的删除整个版本库，或者因为别的问题导致版本库不可用。

其次，不是所有的网络文件共享协议都支持 Subversion 需要的文件锁定，所以你会发现你的版本库被毁了。它也许不会马上发生，但是总有一天会有 2 个用户同时访问版本库。

第三，文件的权限必需设置得井井有条。也许 Windows 的共享可以避开这个问题，但是在 SAMBA 中却是相当困难的。

file:// 访问是为本机工作而准备的，只能单用户访问，特别是测试和调试。当你打算共享版本库的时候，你真的需要设置一个适当的服务器，而且它并不像你想象的那样困难。阅读第 3.5 节 “访问版本库”获得选择指南，并配置服务器。

3.1.5. #版本库布局

在将你的数据导入到版本库之前，首先你得考虑如何组织你的数据。如果你使用一种推荐的布局，你在后面的操作将会更容易许多。

有一些标准的，推荐使用的组织版本库结构的方法。大多数人创建一个 trunk 目录掌管开发的 “主干”，一个 branches 目录存放分支副本，以及一个 tags 目录存放标记副本。如果一个版本库只掌管一个项目，那么人们通常创建这些顶级目录：

```
/trunk
/branches
/tags
```

因为这个布局非常通用，所以当使用 TortoiseSVN 创建版本库时，它会提出帮你创建这个目录结构。

如果一个版本库包含多个项目，人们通常按分支来安排布局:

```
/trunk/paint
/trunk/calc
/branches/paint
/branches/calc
/tags/paint
/tags/calc
```

……或者按项目:

```
/paint/trunk
/paint/branches
/paint/tags
/calc/trunk
/calc/branches
/calc/tags
```

如果项目不是密切相关，而且每一个是单独被检出，那么按项目布局是合理的。对于那些你想一次检出所有项目，或需要将它们打成一个分发包的相关项目，按分支来布局通常比较好。这种方式你只要检出一个分支，而且子项目之间的关系也比较清楚。

如果你采用顶层/trunk /tags /branches这种方式，并不意味着你必须复制整个主线为分支或标签，而且某些情况下这种结构更具灵活性。

对于不相关的项目，你可能更愿意使用不同的版本库。当你提交时，改变的是整个版本库的修订号，而不是项目的。让两个不相关的项目共用一个版本库，会导致修订号出现较大的跳跃。Subversion和TortoiseSVN项目看起来是在同一个主机地址，但是它们是在完全独立的版本库中开发着，并且版本号也不相干。

当然，你完全可以不理会上面提及的通用布局。你可以自由改变，来满足你和你团队的需要。请记住，不管你选择哪种布局，它都不是永久的。你可以在随时重新组织你的版本库。因为分支和标签是普通的目录，只要你愿意，TortoiseSVN 可以将它们移动或重命名。

从一种布局转换到另一种布局仅仅是在服务器端移动一些文件或目录；如果你不喜欢版本库的组织形式，仅管大胆地修改那些目录。

因此，如果你还没有在版本库中创建基本的文件夹结构，你应该立刻创建。创建文件夹有 2 种方法。如果你只想创建一个 /trunk /tags /branches 结构，你可以使用版本库浏览器创建这 3 个文件夹(独立的 3 次提交)。如果你想创建一个层次更深的结构，那么更简单的做法是先在硬盘中创建好文件夹结构，然后将其导入(只有 1 次提交)，就像这样：

1. 在你的硬盘上创建一个空的文件夹
2. 在那个文件夹下创建你想要的顶级目录——千万不要放任何文件进去！
3. 将这个结构导入版本库中，只需 右键单击 包含这个结构的文件夹并选择 TortoiseSVN → 导入...。在导入对话框中输入版本库的 URL 并单击确定。这样就会将临时文件夹导入版本库中创建基本布局。

注意，你所导入的那个文件夹的名字并不存在于版本库中，仅仅是它所包含的内容。比如，创建如下结构的文件夹

```
C:\Temp\New\trunk
C:\Temp\New\branches
C:\Temp\New\tags
```

导入C:\Temp\New到版本库的根目录，版本库中将会是这样:

```
/trunk
/branches
/tags
```

3.2. #版本库备份

无论你使用何种版本库，定期维护和验证版本库备份非常重要，或许你可以访问最近版本的文件，但是如果没有版本库，所有的历史将会丢失。

最简单(但不推荐)的方法是复制整个版本库目录到备份介质，然而你必须绝对确定没有访问数据的进程，在这里“访问”的意思是任何访问，一个BDB版本库即使在访问看起来只需要读时也会有写操作，如果在复制时版本库被访问了(web浏览器，WebSVN等等)，备份将毫无价值。

推荐的方法是运行

```
svnadmin hotcopy path/to/repository path/to/backup --clean-logs
```

，用一种安全的方式创建版本库的备份，备份是一个副本，--clean-logs选项并不必须，但是通过删除BDB版本库中多余的日志文件可以节省一些空间。

The svnadmin tool is installed automatically when you install the Subversion command line client. The easiest way to get this is to check the option to include the command line tools when installing TortoiseSVN, but if you prefer you can download the latest version of command line tools directly from the Subversion [http://subversion.apache.org/packages.html#windows] website.

3.3. #服务器端钩子脚本

A hook script is a program triggered by some repository event, such as the creation of a new revision or the modification of an unversioned property. Each hook is handed enough information to tell what that event is, what target(s) it's operating on, and the username of the person who triggered the event. Depending on the hook's output or return status, the hook program may continue the action, stop it, or suspend it in some way. Please refer to the chapter on Hook Scripts [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.create.html#svn.reposadmin.create.hooks] in the Subversion Book for full details about the hooks which are implemented.

这些钩子脚本被版本库所在的服务器执行。TortoiseSVN 也允许你配置由确定事件触发，在本地执行的客户端脚本。请参看 第 4.30.8 节 “客户端钩子脚本”以获得更多信息。

钩子脚本的例子位于版本库的 hooks 目录下。这些示例版本适用于 Unix/Linux 服务器，需要修改后才能用于基于 Windows 的服务器。钩子可以是批处理文件或者可执行文件。下面是一个可以用作版本属性更改之前 (pre-revprop-change) 钩子的批处理文件。

```
rem Only allow log messages to be changed.
if "%4" == "svn:log" exit 0
echo Property '%4' cannot be changed >&2
```

```
exit 1
```

注意，任何送到标准输出 (stdout) 的内容都会被丢弃。如果想要在拒绝提交对话框中显示信息，必须将该信息送到标准错误 (stderr)。在批处理文件中，通过使用 >&2 来实现。

跨越钩子

如果一个钩子脚本拒绝你的提交，那这就是它的最终决定。但是你可以在脚本中使用 魔咒 来构建一种跨越机制。如果脚本要拒接操作，它首先在日志信息中扫描某个特定的通关短语，可以是一个固定的短语或者带有某个前缀的文件名。如果它找到了魔咒则允许提交继续进行。如果没有找到则阻挡提交并返回一条消息，例如 “你没有念魔咒”。 :-)

3.4. #检出链接

如果你希望你的 Subversion 版本库对于别人可用，你可以在你的站点包含一个链接。为了让其更加容易访问，你可以为其它 TortoiseSVN 用户包含一个检出链接。

当你安装了 TortoiseSVN，它会注册一个 tsvn: 协议，当 TortoiseSVN 用户点击这样一个链接，检出窗口会自动弹出，且版本库 URL 已经填入。

想要在你个人的 html 页面中加入这样的链结，只需要添加像这样的代码即可:

```
<a href="tsvn:http://project.domain.org/svn/trunk">
</a>
```

当然，如果能插入一张合适的图片会看起来更好。你可以使用 TortoiseSVN 标志 [http://tortoisesvn.net/images/TortoiseCheckout.png] 或者你自己的图片。

```
<a href="tsvn:http://project.domain.org/svn/trunk">
<img src=TortoiseCheckout.png></a>
```

你同样可以使链接指向一个特定的版本，例如

```
<a href="tsvn:http://project.domain.org/svn/trunk?100">
</a>
```

3.5. #访问版本库

要使用 TortoiseSVN (或其它 Subversion 客户端程序)，需要定位版本库的所在之处。版本库可以位于本机使用 file:// 协议访问，或者位于服务器端使用 http:// 或 svn:// 协议访问。这两个服务器协议都可以加密。那就是 https:// 或 svn+ssh://， 抑或带 SASL 加密的 svn://。

如果你使用公共的主机服务，例如 Google Code [http://code.google.com/hosting/]，或者已经有人为你架设好了服务器，那么在这里你不用做什么。前进到 第 4 章日常使用指南 。

如果你没有服务器并且独自工作，或者只是想在独立的环境下评估 Subversion 和 TortoiseSVN，那么本地版本库可能是你最好的选择。只要按照 第 3 章版本库 中先前描述的那样在你自己的电脑中创建一个版本库就好了。你可以跳过本章剩余的内容，进入 第 4 章日常使用指南 了解如何开始用它。

如果你打算在网络共享中设者一个多用户的版本库，请重新考虑。阅读第 3.1.4 节 “访问网络共享磁盘上的版本库”以了解为什么我们认为这是一个坏主意。设置一个服务器并不像听上去那样难。并且还会为你提供更好的性能甚至更快的速度。

More detailed information on the Subversion server options, and how to choose the best architecture for your situation, can be found in the Subversion book under Server Configuration [http://svnbook.red-bean.com/en/1.8/svn.serverconfig.html].

In the early days of Subversion, setting up a server required a good understanding of server configuration and in previous versions of this manual we included detailed descriptions of how to set up a server. Since then things have become easier as there are now several pre-packaged server installers available which guide you through the setup and configuration process. These links are for some of the installers we know about:

- VisualSVN [http://www.visualsvn.com/server/]
- CollabNet [http://www.open.collab.net/products/subversion/whatsnew.html]
- UberSVN [http://www.ubersvn.com/]

You can always find the latest links on the Subversion [http://subversion.apache.org/packages.html] website.

You can find further How To guides on the TortoiseSVN [http://tortoisesvn.net/usefultips.html] website.

第#4#章#日常使用指南

本文目的在与描述TortoiseSVN客户端的日常使用。不是一个版本控制系统指南，也不是Subversion (SVN)的指南。本文档的价值在于，当你知道大概要做什么，却又记不起应该怎么做的时候，可以有个参考的地方。

如果你需要了解使用Subversion进行版本控制的指南，我们建立你阅读以下这本梦幻之书: 《使用 Subversion 进行版本管理》 [http://svnbook.red-bean.com/].

本文档与TortoiseSVN和Subversion一样，也是处于“正在开发”的状态。如果你找到了错误之处，请向邮件列表报告，这样我们就可以更新它。日常使用指南(DUG)中的一些屏幕截图也许不符合当前软件中的情况。请您原谅我们。毕竟我们只是用业余的时间在制作TortoiseSVN 。

为了获得比每日用户指南更多的信息:

- 你应该已经安装了TortoiseSVN。
- 你应该熟悉版本控制系统。
- 你应该知道Subversion的基础。
- 你应该已经建立了一个服务器并且可以访问Subversion库。

4.1. #基本特性

This section describes some of the features of TortoiseSVN which apply to just about everything in the manual. Note that many of these features will only show up within a Subversion working copy.

4.1.1. #图标重载

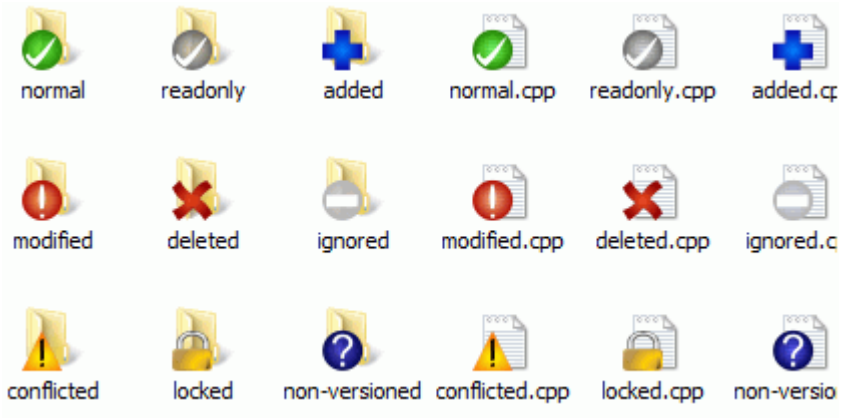


图 4.1. 显示重载图标的资源管理器

TortoiseSVN 最明显的特性之一就是图标重载，重载的图标显示在你的工作副本文件上。你一眼就可以看到文件被修改过了。参考 第 4.7.1 节 “图标重载”查阅不同的重载图标含义。

4.1.2. #右键菜单

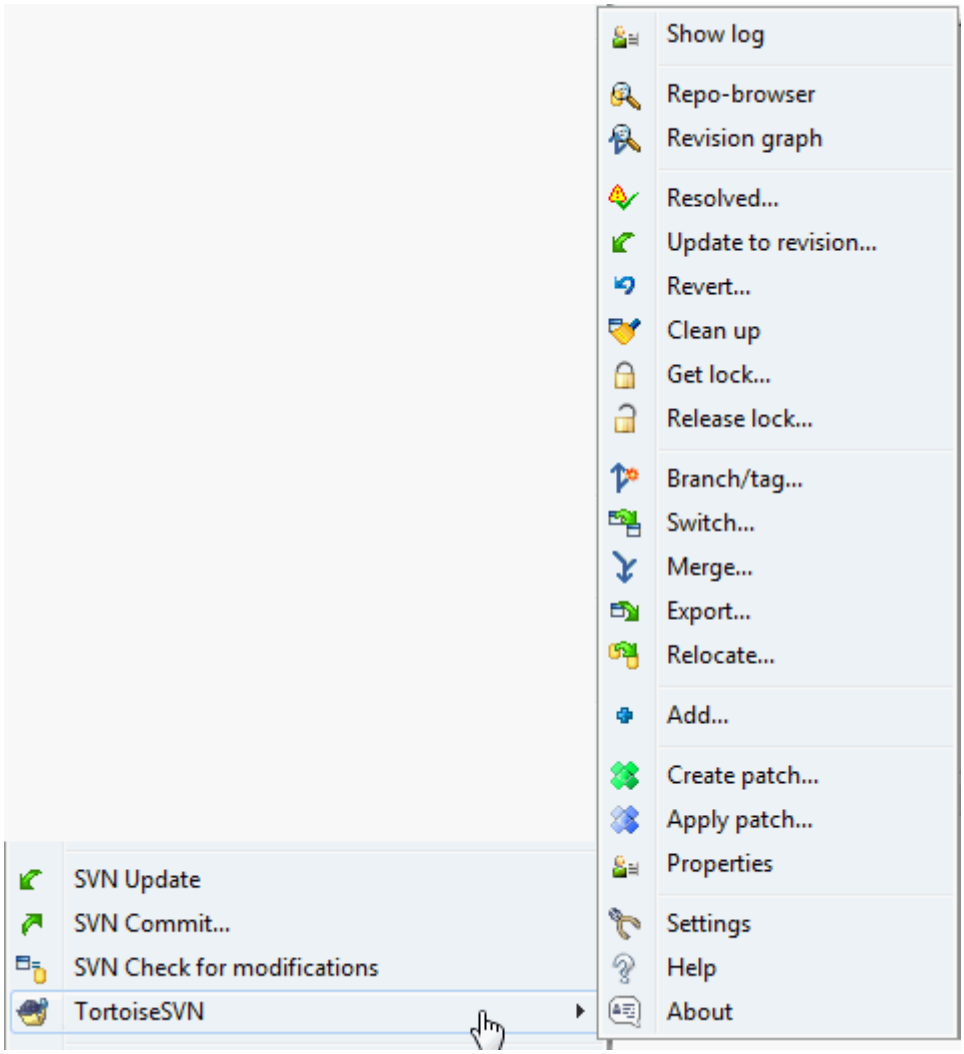


图 4.2. 版本控制下一个目录的右键菜单

所有的TortoiseSVN命令都是通过windows资源管理器的右键菜单执行。右键点击一个文件或者文件夹，大多数菜单项都能够直接显示。一个命令是否显示取决于这个文件或文件夹或者它们的父文件夹是否受版本控制，你也可以将TortoiseSVN的菜单作为资源管理器菜单的一部分。

某些很少被用到的命令只出现在扩展右键菜单中。要想打开扩展右键菜单，需要在 右键单击 时按住 Shift 键。

在某些情况下，你可能看到多个TortoiseSVN条目。这不是BUG!

图 4.3. 在一个版本控制的文件夹下资源管理器文件菜单中的快捷方式。

本示例是在一个受控文件夹下的某个未受控的快捷方式，在资源管理器的文件菜单下有三个TortoiseSVN条目。一个是受控文件夹本身的，一个是快捷方式本身的，第三个是快捷方式所指向的对象。为了帮助你区分它们，菜单条目的图标的右下角有标志，表明是文件、快捷方式、文件夹或是选中了多项。

Windows 2000 的用户将会发现右键菜单仅显示文字，没有上图所示的菜单图标。我们知道这是因为在旧版下使用的缘故，由于微软改变了 Vista 中图标句柄工作方式，我们只好使用不同的方式，很遗憾，这种方式不能在 Windows 2000 下工作。

4.1.3. #拖放

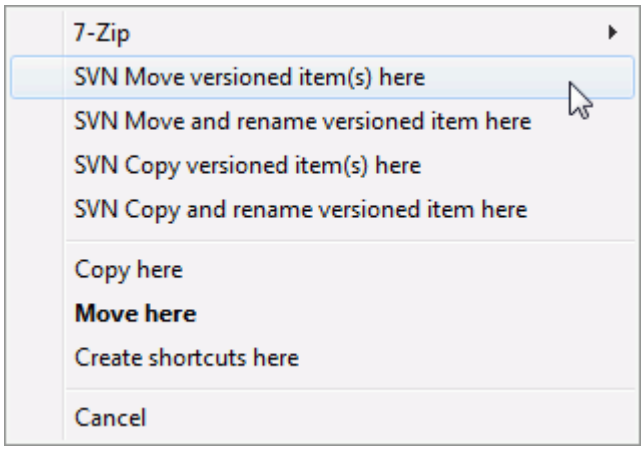


图 4.4. 版本控制下的一个目录的右键拖拽菜单

在工作副本里右键拖拽文件或目录到新的位置，或者右键拖拽一个非版本控制的文件或文件夹到一个版本控制目录下的时候，右键菜单还能够出现其他的命令。

4.1.4. #常用快捷方式

一些常见的操作与 Windows 的快捷键是一样的，但没有出现在按钮或是菜单中。如果你找不到一些显而易见的操作，比如刷新视图，请参考以下内容。

F1
: 当然是帮助。

F5
: 刷新当前视图。这也许是单键命令中唯一一个最常用的了。比如... 在资源浏览器中，这个键可以刷新工作副本中的图标重载。在提交对话框中，它可以重新扫描查找哪些是需要提交的。在版本日志对话框中，可以重新联系版本库以检查更多的最近修改情况。

Ctrl-A
: 全选。可用于在得到一个错误消息并想要复制粘贴到电子邮件时。使用Ctrl-A to选择错误错误，然后...

Ctrl-C
: ... 复制选中的文本。

4.1.5. #认证

如果连接的版本库需要密码，就会显示认证对话框。

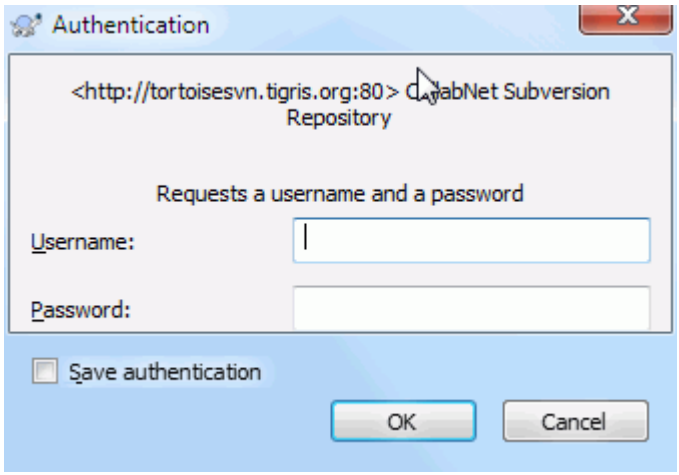


图 4.5. 认证对话框

输入你的用户名和密码。复选框能让 TortoiseSVN 在 Subversion 的缺省目录: %APPDATA%\Subversion\auth 的三个子目录内保存认证信息:

- svn.simple 里包含了基本认证方式所需要的认证信息(用户名/密码)。注意，保存的密码是通过 WinCrypt API 加密的，不是文本形式。
- svn.ssl.server 里包含了SSL服务器证书。
- svn.username 里包含了用户名认证的认证信息(不需要提供密码)。

如果想要清除所有服务器的认证缓存，可以通过 TortoiseSVN 设置对话框的已保存数据页来实现。那个按钮能够清除 Subversion 的 auth 目录中缓存的所有认证数据，以及老版本的 TortoiseSVN 存储在注册表里的认证数据。请参考 第 4.30.6 节 “已保存数据的设置”。

如果你想清除某一个范围的认证信息，那么你需要打开那些目录，找到包含你要清除的信息的文件并删除文件。

有些人希望在注销 Windows 或者关机时删除认证数据。完成这个愿望的方法是使用关机脚本来删除 %APPDATA%\Subversion\auth 目录，例如:

```
@echo off
rmdir /s /q "%APPDATA%\Subversion\auth"
```

关于如何安装这种脚本的详细说明请参阅: http://www.windows-help-central.com/windows-shutdown-script.html。

更多的关于如何设置服务器进行认证和授权控制的信息，请参考: 第 3.5 节 “访问版本库”

4.1.6. #最大化窗口

大多数 TortoiseSVN 的对话框显示很多信息，但是经常只有最大化高度或者宽度有用，而不是全部最大化，覆盖整个屏幕。为了方便，在 最大化 按钮有快捷方式做这些工作。使用鼠标中键最大化高度，右键最大化宽度。

4.2. #导入数据到版本库

4.2.1. #导入

如果将项目导入一个已经含有其它项目的版本库中，那么版本库的结构已经确定了。如果要导入一个新的版本库中，那么最好花点时间来想一下如何设置版本库的结构。阅读 第 3.1.5 节 “版本库布局” 获得更多建议。

This section describes the Subversion import command, which was designed for importing a directory hierarchy into the repository in one shot. Although it does the job, it has several shortcomings:

- 不能选择包括哪些文件或文件夹，除非使用全局忽略设置。
- 导入的文件夹不能变成工作副本。你必须通过签出操作从服务器拿回文件。
- 很容易导入到版本库中错误的文件夹层次。

For these reasons we recommend that you do not use the import command at all but rather follow the two-step method described in 第 4.2.2 节 “导入适当的位置,” unless you are performing the simple step of creating an initial /trunk /tags /branches structure in your repository. Since you are here, this is how the basic import works ...

在将你的项目导入到版本库之前，你应该:

1. 删除所有构建工程不需要的文件(临时文件，编译器产生的文件，例如 *.obj，生成的二进制文件，...)
2. 组织目录和子目录内的文件。尽管以后可以改名/删除文件，我们还是建议你在导入之前使你的项目结构组织良好!

现在进入资源管理器，选择你的项目的顶层目录，右击打开上下文菜单。选择命令TortoiseSVN → 导入 ...，它会弹出一个对话框:

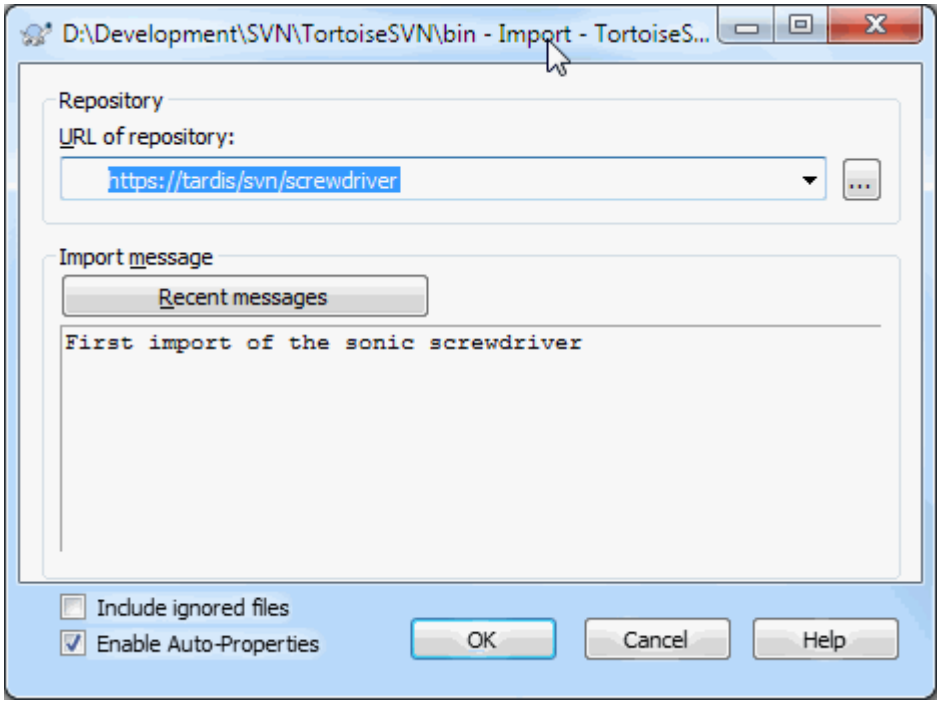


图 4.6. 导入对话框

在这个对话框中，需要输入版本库所在的 URL，你的项目将会导入到这里。非常重要的事项，你必须了解：你要导入的本地文件夹自身不会出现在版本库中，版本库中只有文件夹中的内容。例如，你有这样的文件夹结构：

```
C:\Projects\Widget\source
C:\Projects\Widget\doc
C:\Projects\Widget\images
```

你将 C:\Projects\Widget 导入到 http://mydomain.com/svn/trunk，然后你会惊奇的发现：你的子目录径直地进入 trunk 中，而不是在 Widget 子目录中。你需要将子目录作为 URL 的一部分明确的指出来，http://mydomain.com/svn/trunk/Widget-X。注意，如果版本库中不存在指定的子目录，导入命令将会自动创建它们。

这个输入信息将用作提交日志。

默认情况下，匹配全局忽略模式的文件和文件夹不会被导入。你可以使用包含忽略文件检验栏来禁止此行为。参考第 4.30.1 节 “常规设置”以获得关于全局忽略模式的更多信息。

当你点击确认时，TortoiseSVN 会导入包含所有文件的完整目录树到版本库。现在这个工程就存贮在版本库，被版本控制。请注意，你导入的文件夹没有被版本控制！你需要检出刚才导入的版本，以便获得受版本控制的工作副本。或者继续阅读，找到如何导入文件夹到合适的位置。

4.2.2. #导入适当的位置

假定你已经有个版本库，你想给它增加一个新目录结构，只需以下步骤：

1. Use the repository browser to create a new project folder directly in the repository. If you are using one of the standard layouts you will probably want to create this as a sub-folder of trunk rather than in the repository root. The repository browser shows the repository structure just like Windows explorer, so you can see how things are organised.

2. Checkout the new folder over the top of the folder you want to import. You will get a warning that the local folder is not empty. Ignore the warning. Now you have a versioned top level folder with unversioned content.

3. 在此受版本控制的文件夹上使用TortoiseSVN → 增加...增加部分或全部内容。你可以增加或删除文件，在文件夹上设置svn:ignore属性，或者你需要的其它修改。

4. 提交顶级目录，你有一个新的版本树，一份从你已有目录创建的本地工作副本。

4.2.3. #专用文件

有时候你需要版本控制一个包含用户专用的数据。它意味着你有一个文件，每个开发者/用户都需要修改，一边满足他/她的本地配置。但是版本控制这样的文件是困难的，因为每个用户可能都要提交他/她的修改。

在这种情况下，我们建议使用模版文件。创建一个包含所有开发者需要的数据的文件，增加到版本库中，让开发者检出。然后，每个开发者创建一个副本，改名此文件。于是，修改这个文件不再是问题。

作为例子，你可以看看TortoiseSVN的构建脚本。它调用一个TortoiseVars.bat文件，它并不在版本库中。 只有TortoiseVars.tmpl在版本库中。TortoiseVars.tmpl是一个模版文件，每个开发者都需要创建一个副本，改名为TortoiseVars.bat。在这个文件中，我们增加了注释，所以用户知道他们需要编辑那些行，以便适应他们的本地配置，使其能工作。

于是为了不干扰用户，我们也将TortoiseVars.bat增加到它的父目录的忽略列表，也就是，我们设置了Subversion属性svn:ignore包含这个文件名称。这样，每次提交时它都不会作为没有版本控制的文件出现。

4.3. #检出工作副本

为了得到一个工作副本，需要进行从版本库检出的操作。

在Windows资源管理器里选择一个存放工作副本的目录。右键点击弹出右键菜单，选择TortoiseSVN → 检出…命令。然后就会看到下面的对话框:

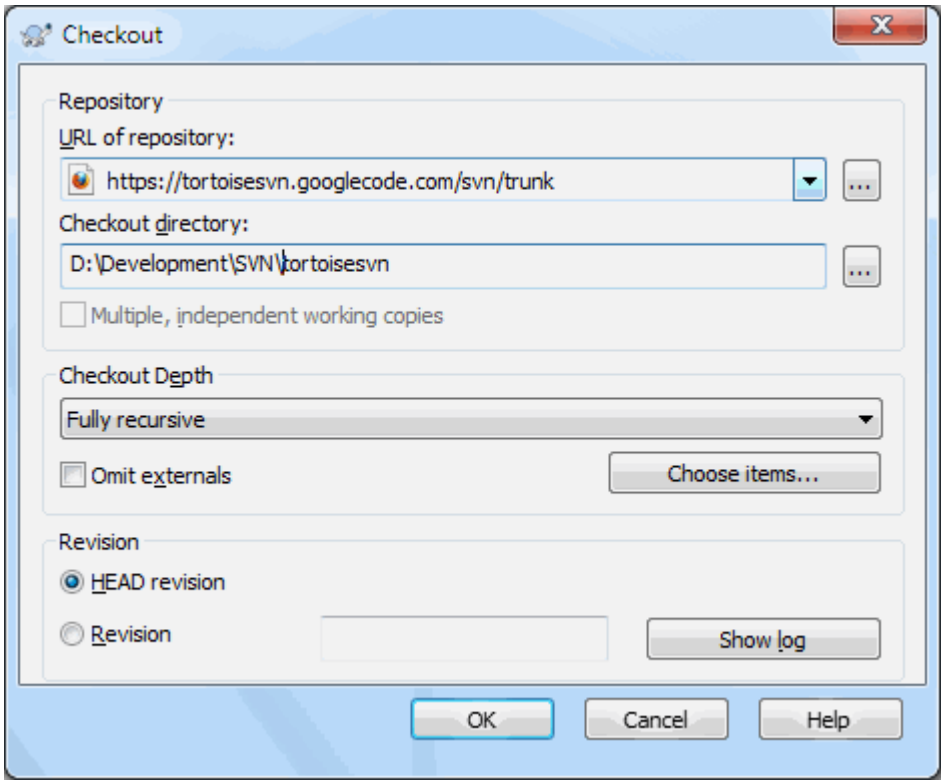


图 4.7. 检出对话框

如果输入一个并不存在的目录名，那么这个名字的目录就会被创建出来。

4.3.1.#检出深度

你可以选择要检出的深度，它允许你指定子目录递归的深度。如果你只需要大目录中的几个子条目，你可以只检出最高层目录，然后递归的更新选择的目录。

全递归
: 检出完整的目录树，包含所有的文件或子目录。

直接子节点，包含文件夹
: 检出目录，包含其中的文件或子文件夹，但是不递归展开子文件夹。

仅文件子节点
: 检出指定目录，包含所有文件，但是不检出任何子文件夹。

仅此项
: 只检出目录。不包含其中的文件或子文件夹。

工作副本
: 保持工作副本指定的深度。此选项不用于检出对话框，但它是其它所有含有深度配置对话框的默认配置。

排除
: 对于已经创建好的工作副本，可以使用此选项来缩减文件夹的深度。这个选项只在更新至版本对话框中可用。

To easily select only the items you want for the checkout and force the resulting working copy to keep only those items, click the Choose items... button. This opens a new dialog where you can check all items you want in your working copy and uncheck all the items you don't want. The resulting working copy is then known as a sparse checkout. An update of such a working copy will not fetch the missing files and folders but only update what you already have in your working copy.

If you check out a sparse working copy (i.e., by choosing something other than fully recursive for the checkout depth), you can easily add or remove sub-folders later using one of the following methods.

4.3.1.1.#Sparse Update using Update to Revision

Right click on the checked out folder, then use TortoiseSVN → Update to Revision and select Choose items.... This opens the same dialog that was available in the original checkout and allows you to select or deselect items to include in the checkout. This method is very flexible but can be slow as every item in the folder is updated individually.

4.3.1.2.#Sparse Update using Repo Browser

Right click on the checked out folder, then use TortoiseSVN → Repo-Browser to bring up the repository browser. Find the sub-folder you would like to add to your working copy, then use Context Menu → Update item to revision....

4.3.1.3.#Sparse Update using Check for Modifications

In the check for modifications dialog, first shift click on the button Check repository. The dialog will show all the files and folders which are in the repository but which you have not checked out as remotely added. Right click on the folder(s) you would like to add to your working copy, then use Context menu → Update.

当你想要检出一个很大的文件树的某些部分而且想要方便的只更新单一的工作副本时，该功能非常有用。假设有一个很大的文件树，其中包含 99 个子文件夹从 Project01 到 Project99，你只想检出 Project03，Project25 和 Project76/SubProj。按下列步骤操作：

1. 检出父文件夹时检出深度使用“仅此项”。现在，你获得一个空的顶级文件夹。

2. 选中新文件夹，使用 TortoiseSVN → 版本库浏览器 来显示版本库的内容。

3. 右键单击 Project03 然后选择右键菜单 → 更新项目至版本...。保持默认设置并单击 确定。现在这个文件夹就位于你的工作副本中了。

   为 Project25 重复相同的操作。

4. 定位至 Project76/SubProj 并且进行相同的操作。这次需要注意，Project76 文件夹中除了新增的 SubProj 没有其它内容。Subversion 创建了相关的文件夹并没有拿出其全部内容。

改变工作副本深度

一旦以某个深度检出了工作副本后，以后还可以修改这个深度来获得更多或更少的内容，使用 右键菜单 → 更新该项至版本。在弹出的对话框中，确认选中了 粘滞深度 复选框。

使用旧版本服务器

1.5 版之前的服务器不支持工作副本深度请求。所以不能有效的处理这种请求。不过该命令仍然可以使用，但是旧版本的服务器会发送所有的数据，让客户端来过滤掉哪些是不需要的，这就意味着大量的网络传输。如果可能，应该将服务器升级到 1.5 版以上。

如果项目含有外部项目的引用，而这些引用你不希望同时检出，请选中忽略外部项目复选框。

如果选中了 忽略外部项目，或者想要增加深度值，你应该使用 TortoiseSVN → 更新至版本... 替代 TortoiseSVN → 更新 来对工作副本进行更新。标准的更新操作会包含外部项目并保持深度。

强烈建议你只检出 trunk 或更低层的目录树。如果你在 URL 中指定了根路径，你的硬盘有可能被塞满，因为你将会得到整个版本库树的副本，包括项目所有的分支和标签(tag)！

关于导出

有时你可能想要建立一个没有.svn目录的本地的副本，比如建立一个源代码压缩包。要达到这个目的，请参考第 4.26 节 “导出一个Subversion工作副本”。

4.4. #将你的修改提交到版本库

将你对工作副本的修改发送给版本库，称为提交修改。但在你提交之前要确保你的工作副本是最新的。你可以直接使用TortoiseSVN → 更新，或者，你可以先使用TortoiseSVN → 检查修改看看哪些文件在本地或是服务器上已经有了改动。

4.4.1. #提交对话框

如果你的工作副本是最新的，并且没有冲突，你就已经为提交做好准备了，选择你要提交的文件和/或文件夹，然后TortoiseSVN → 提交...

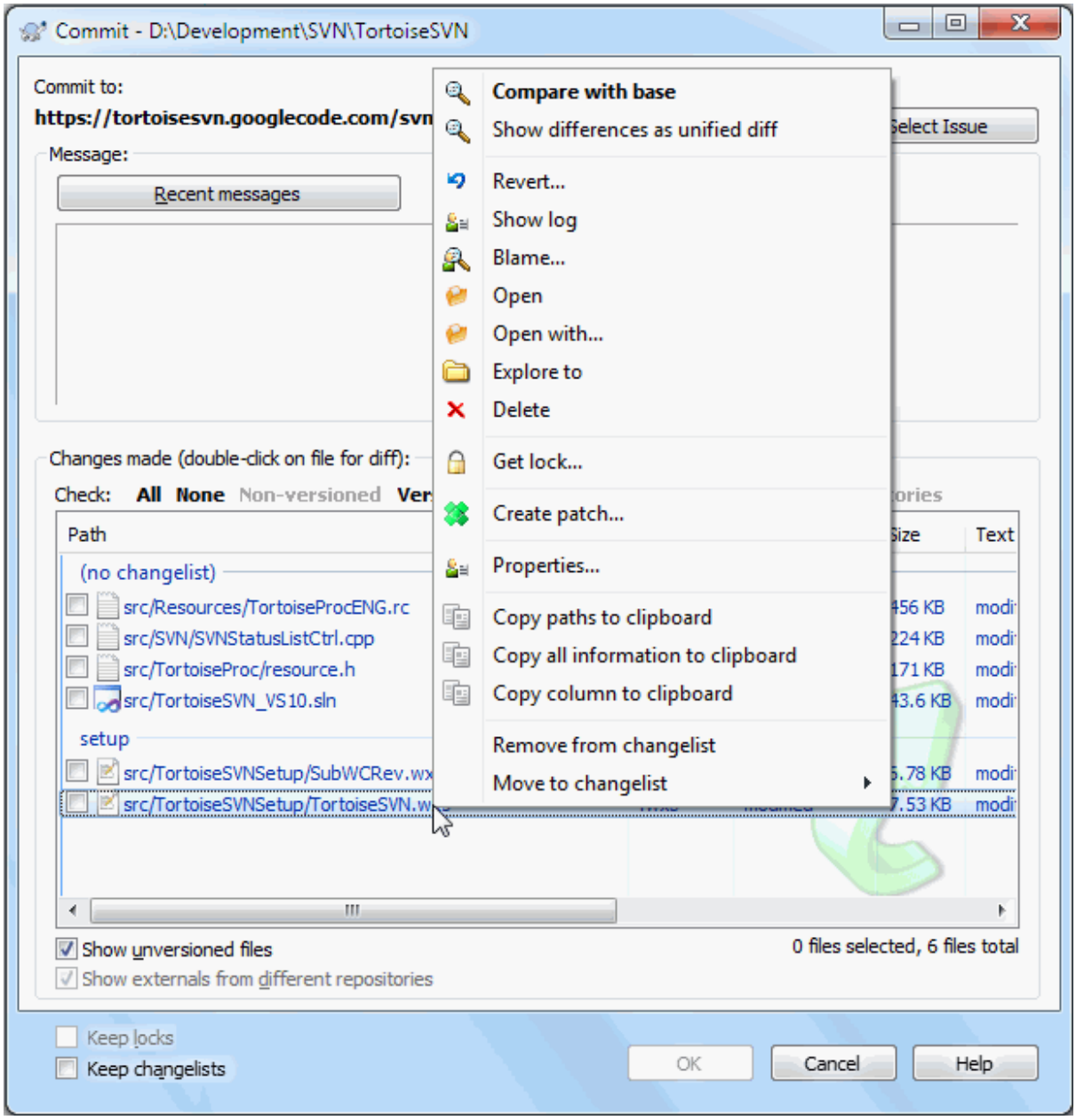


图 4.8. 提交对话框

提交对话框将显示每个被改动过的文件，包括新增的、删除的和未受控的文件。如果你不想改动被提交，只要将该文件的复选框的勾去掉就可以了。如果你要加入未受控的文件，只要勾选该文件把它加入提交列表就可以了。

For information on the coloring and overlays of the items according to their status, please see 第 4.7.4 节 “本地与远程状态”

那些被切换(switched)到不同版本库路径的项也用(s)标记来表示。当工作在分支上的时候你可能切换到某处，然后忘记切换回主干。这是你的警告信号！

提交文件还是文件夹？

当你提交文件时，提交对话框只显示你所提中的文件。当你提交文件夹中，提交对话框将自动选择有改动的文件。如果你忘记了你建立的一个新文件，提交文件夹将使你可以找到它。提交一个文件夹并不意味着每个文件都被标识为修改过的，它仅仅是通过帮你多做些事从而让你的生活更滋润一点。

在提交对话框中有很多未受控的文件

如果你认为提交对话框显示了太多的未受版本控制的文件(如编译器产生的文件或是编辑器的备份文件)，有几种方法可以处理这种情况。你可以:

- 将文件(或是通配符扩展)加入到设置页的排除列表中。这对每个工作副本都起作用。
- 使用TortoiseSVN → 加入忽略列表，将文件加入svn:ignore列表。 这只对你设置了svn:ignore属性的路径有效。使用SVN属性对话框，你可以改变一个目录的svn:ignore属性。
- add the file to the svn:global-ignores list using TortoiseSVN → Add to ignore list (recursively) This will affect the directory on which you set the svn:global-ignores property and all subfolders as well.

参考 第 4.13 节 “忽略文件和目录”获得更多的信息。

在提交对话框中双击任何修改过的文件，将运行外部 diff 工具显示你做的改动。上下文菜单将给你更多的选项，请看屏幕截图。你可以从这里将文件拖动到另一个应用程序中，如文本编辑器或 IDE。

可以通过单击条目左侧的复选框来选中或不选该条目。对于目录，可以按下 Shift 键再 选择 就可以递归该动作。

在底部面板中显示的列是可定制的。如果你右击任何一列的头部，你就会看到一个上下文菜单，允许你选择哪一列要显示。还可以在鼠标移动到列边界时通过拖动手把来改变列的宽度。这些定制的内容都会被保留下来，下一次你会见到相同的列。

缺省情况下，当你成功提交修改后，你在这些文件上持有的锁会被自动释放。如果你需要保留锁，请确认选中检查框保留锁。此检查框的缺省状态从 Subversion 配置文件的 no_unlock 选项获取。参考 第 4.30.1 节 “常规设置”以获得更多关于编辑 Subversion 配置文件的信息。

拖放

你可以将文件从别的地方拖动到提交对话框，只要工作副本是由同一版本库中检出就可以了。比如，你有一个很大的工作副本，要开好几个资源管理器窗口来查看层次中不同的文件夹。如果你要避免从顶级文件夹提交(冗长而缓慢的文件夹改动检查)，你可以打开一个文件夹的提交对话框，然后将别的窗口中的项拖进去，可样就可以一次提交它们了。

你可以将未版本控制的文件拖到工作副本提交对话框中，它们就会被自动增加。

从提交对话框底部的列表中将文件拖拽到日志消息编辑框中，就能以文本格式将文件的路径插入编辑框中。当你想将本次提交相关的文件路径插入日志消息中时该功能非常有用。

修复外部改名

有时候文件不是用 Subversion 改名，于是它们在文件列表中作为丢失和未版本控制的文件出现。为了避免丢失历史，你需要通知Subversion。简单的选择老名称(丢失)和新名称(未版本控制)，然后使用右键菜单 → 修复移动来指明这两个文件是改名关系。

修复外部复制

如果复制了一个文件，但不是通过 Subversion 的命令来做的，你可以修复此次复制来确保新文件不会丢失历史。只要简单的选择旧文件(正常或已修改)和新文件(无版本控制)，然后使用 右键菜单 → 修复复制来修复两个文件的复制关系。

4.4.2. #修改列表

提交对话框支持 Subversion 的更改列表功能来分组相关的文件。关于这个功能，请查看第 4.8 节“修改列表”。

4.4.3. #Commit only parts of files

Sometimes you want to only commit parts of the changes you made to a file. Such a situation usually happens when you're working on something but then an urgent fix needs to be committed, and that fix happens to be in the same file you're working on.

right click on the file and use Context Menu → Restore after commit. This will create a copy of the file as it is. Then you can edit the file, e.g. in TortoiseMerge and undo all the changes you don't want to commit. After saving those changes you can commit the file.

After the commit is done, the copy of the file is restored automatically, and you have the file with all your modifications that were not committed back.

4.4.4. #从提交列表中排除项目

有时，你经常更改一些版本控制的文件但你却不打算提交它们。这有可能说明你的构建过程中存在瑕疵 - 那些文件为什么是版本控制的？应该使用模版文件吗？但可能这是无法避免的。一个经典的原因是当你每次构建的时候，集成开发环境(IDE)更改了项目文件的时间戳。项目文件是版本控制的，因为它包含全部的构建设置。但是，仅仅因为时间戳更改了的情况下，你并不需要提交它。

为了解决这样一个棘手的问题，我们准备了一个名叫 ignore-on-commit 的更改列表。任何一个被添加到这个列表的文件在提交对话框中将不会自动选中。你仍然可以提交此文件的更改，不过你需要在提交对话框中手动选中它。

4.4.5. #提交日志信息

确保输入描述你所提交的修改内容的日志信息。这可以帮你回顾做了什么，什么时候做的。信息的内容可长可短，许多项目规定了要包含的内容、使用的语言甚至是严格的格式。

你可以使用与电子邮件相似的约定，简单格式化日志消息。如果对文本采用这些样式，使用*文本*表示粗体，_文本_表示下划线，^文本^表示斜体。

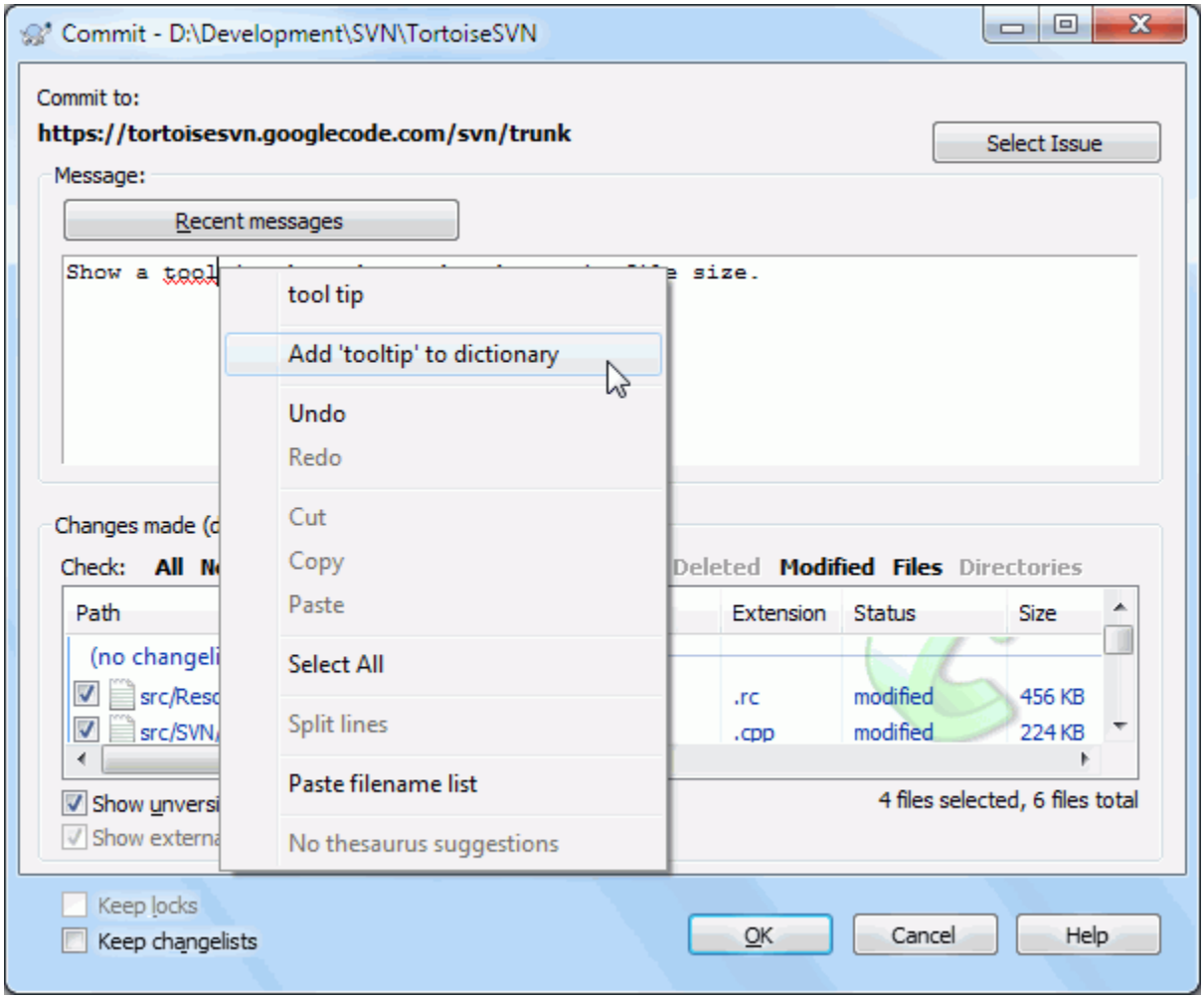


图 4.9. 提交对话框的拼写检查器

TortoiseSVN包含了一个拼写检查器帮助你正确地书写日志信息。对任何错误拼写的词都高亮显示。使用右键菜单可以获得修改建议。当然它不会知道所有的技术术语，所以有时一些拼写正确的词会被当作错误。但不用担心，你可以使用右键菜单将它们加入你的个人字典中。

日志消息窗口同时也含有文件名和函数自动完成的功能。它使用正则表达式从本次提交的(文本)文件中抽取类和函数的名称，以及被提交文件的文件名。如果你输入的单词与列表中的内容匹配(至少键入 3 个字符，或者按 Crtl+空格键)，就会显示一个下拉列表让你选择完整的名字。TortoiseSVN 支持的正则表达式位于 TortoiseSVN 安装文件夹下的 bin 文件夹中。你可以自定义匹配规则然后将其保存在文件 %APPDATA%\TortoiseSVN\autolist.txt 中。当然，在升级新版本的 TortoiseSVN 时你个人的自动匹配规则列表不会被覆盖。如果你不熟悉正则表达式，可以看一下介绍：http://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F，以及在线文档和教程：http://www.regular-expressions.info/。

写出正确的匹配规则有点棘手，所以为了帮你写出合适的表达式，我们提供了一个对话框，你可以输入表达式，然后输入文件名来做个测试。要启动该对话框，在命令提示符中输入这个命名：TortoiseProc.exe /command:autotexttest。

你可以重复使用先前键入的日志信息。只需要点击最近信息即可查看你为此工作副本键入的最近几条信息。

你可以从 TortoiseSVN 的设置窗口的已保存数据页中清除所有的保存消息，或者你可以在最近信息对话框中使用 Delete 键单独删除某条消息。

如果想要在日志信息中加入选中的文件路径，可以在编辑框中使用 右键菜单 → 粘贴文件名列表。

另一个向日志消息中插入路径的方法是：从文件列表中将文件拖拽到文本框中。

指定文件夹属性

有几个特殊的文件夹属性可用于帮助我们得到更多的对提交日志信息的格式以及拼写检查模块的控制。参考第 4.17 节 “项目设置”以了解详情。

与缺陷跟踪工具集成

如果你激活了一个bug跟踪系统，你可以在Bug-ID / Issue-Nr:文本框中设置一个或多个问题。多个问题应该用逗号分割。或者，如果你使用基于正则表达式的bug跟踪支持，只要将你的问题引用作为日志信息的一部分加入就可以了。详情请见第 4.28 节 “与 BUG 跟踪系统/问题跟踪集成”。

4.4.6. #提交进程

在按下OK之后，会出现一个对话框显示提交的进度。

图 4.10. 显示提交进度的进度对话框

进度对话框使用颜色代码来高亮显示不同的提交行为。

蓝色
: 提交一个修改。

紫色
: 提交一个新增项。

深红
: 提交一个删除或是替换。

黑色
: 所有其他项。

这是默认的配色方案，但你可以通过设置对话框来定制这些颜色。参考第 4.30.1.5 节 “TortoiseSVN 颜色设置”获得详情。

4.5. #用来自别人的修改更新你的工作副本

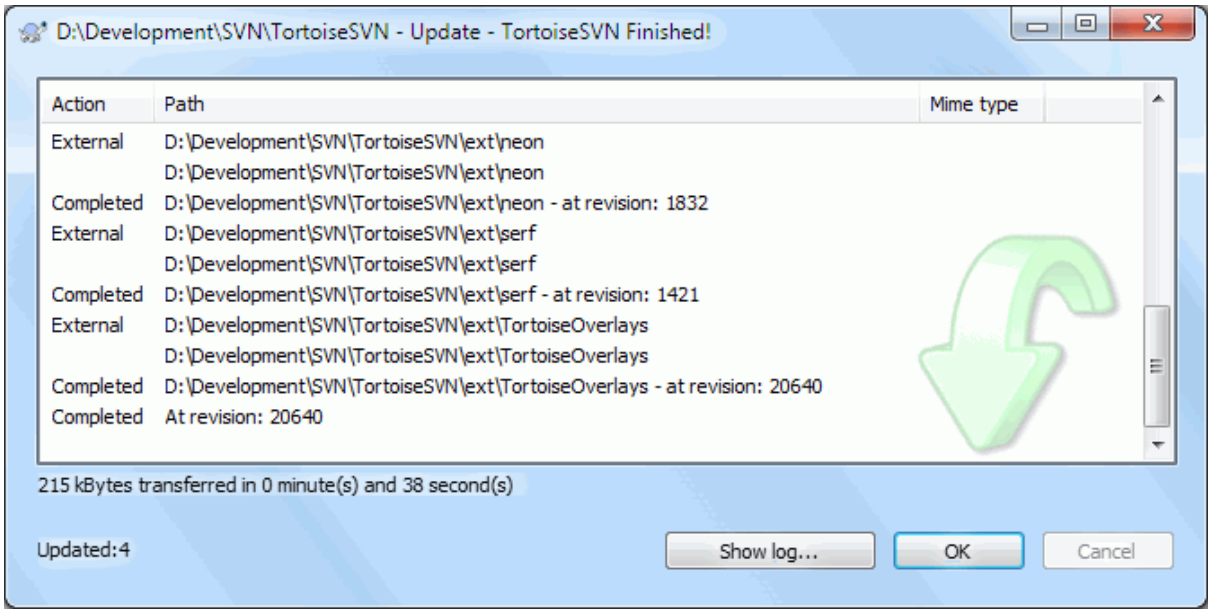


图 4.11. 已经完成更新的进度对话框

你需要定期地确保将别人所做的修改整合到你本地的工作副本中。从服务器获取更改到本地副本的过程就叫做 更新。更新的对象可以是一个文件，选中的多个文件或者对整个目录结构进行递归。要进行更新，选中要更新的文件或目录，右键单击然后选中右键菜单中的 TortoiseSVN → 更新。将会弹出一个窗口，随着更新的进行显示进度。其他人所做的更改将会合并到你的文件中，并保留同一个文件中你所做的更改。版本库 不会 受更新影响。

进度对话框使用颜色代码来高亮不同的更新行为

紫色
: 新项已经增加到你的工作副本中。

深红
: 你的工作副本中删除了多余项，或是你的工作副本中丢失的项被替换。

绿色
: 版本库中的修改与你的本地修改成功合并。

亮红
: 来自版本库的修改在与本地修改合并时出现了冲突，需要你解决。

黑色
: 你的工作副本中的没有改动的项被来自版本库中新版本所更新。

这是默认的配色方案，但你可以通过设置对话框来定制这些颜色。参考第 4.30.1.5 节 “TortoiseSVN 颜色设置”获得详情。

如果在更新过程中发生了 冲突 (有可能是因为你和他人修改了同一个文件的同一行并且这些更改不匹配)那么对话框会用红色显示冲突。可以通过 双击 那些行来启动外部合并工具解决冲突。

等更新结束后，对话框会在文件列表的底部显示一段总结，包括已更新、已添加、已删除、冲突等情况的条目数量。可以使用 Ctrl+C 将这段信息复制到剪贴板。

The standard Update command has no options and just updates your working copy to the HEAD revision of the repository, which is the most common use case. If you want more control over the update process, you should use TortoiseSVN → Update to Revision... instead. This allows you to update your working copy to a specific revision, not only to the most recent one.

Suppose your working copy is at revision 100, but you want it to reflect the state which it had in revision 50 - then simply update to revision 50.

In the same dialog you can also choose the depth at which to update the current folder. The terms used are described in 第 4.3.1 节 “检出深度”. The default depth is Working copy, which preserves the existing depth setting. You can also set the depth sticky which means subsequent updates will use that new depth, i.e. that depth is then used as the default depth.

To make it easier to include or exclude specific items from the checkout click the Choose items... button. This opens a new dialog where you can check all items you want in your working copy and uncheck all the items you don't want.

You can also choose whether to ignore any external projects in the update (i.e. projects referenced using svn:externals).

如果你将文件或文件夹更新至某个特定版本，就不应该修改这些文件。当提交它们时会得到错误消息 “过期”！如果你想撤销某个文件的更改然后从先前的版本开始重新开始，可以从版本日志对话框中回滚至先前的版本。查看 第 B.4 节 “Roll back (Undo) revisions in the repository” 获得更多的指示以及其它可选的操作方式。

当要查看你的项目在过去的某个时间是什么样子时， 更新至版本 功能有时会很有用。但一般来说，更新单个文件到一个较早的版本不是一个好主意，因为这样就使工作副本处于一种不一致的状态。如果更新的文件改变了文件名，你甚至会发现该文件从你的工作副本中消失了，因为在较早的版本中没有叫这个名字的文件。也还会发现该条目显示正常的绿色图标重载，所以很难察觉到存在过期的文件。

如果你只是想要某文件早先版本的本地副本，从该文件的日志对话框中使用 右键菜单 → 保存版本至... 更好。

多文件/文件夹

如果你在资源管理器中选择了多文件和文件夹，然后选择更新，这些文件/文件夹一个接一个的被更新。TortoiseSVN确保所有的来自同一版本库的文件/文件夹被更新到同一个版本！即使在更新过程中发生了另一个提交。

4.6. #解决冲突

偶尔，当你从版本库更新、合并文件时，或者切换工作副本至一个不同的 URL 时你会遇到冲突。有两种冲突:

文件冲突
: 当两名(或更多)开发人员修改了同一个文件中相邻或相同的行时就会发生文件冲突。

树冲突
: 当一名开发人员移动、重命名、删除一个文件或文件夹，而另一名开发人员也对它们进行了移动、重命名、删除或者仅仅是修改时就会发生树冲突。

4.6.1. #文件冲突

A file conflict occurs when two or more developers have changed the same few lines of a file. As Subversion knows nothing of your project, it leaves resolving the conflicts to the developers. The conflicting area in a text file is marked like this:

```
<<<<<<< filename
    your changes
=======
    code merged from repository
>>>>>>> revision
```

Also, for every conflicted file Subversion places three additional files in your directory:

文件名.扩展名.mine
: 这是你的文件，在你更新你的工作副本之前存在于你的的工作副本中——也就是说，没有冲突标志。这个文件除了你的最新修改外没有别的东西。

文件名.扩展名.r旧版本
: 这是在你更新你的工作副本之前的基础版本(BASE revision)文件。也就是说，它是在你做最后修改之前所检出的文件。

文件名.扩展名.r新版本
: 这个文件是当你更新你的工作副本时，你的 Subversion 客户端从服务器接收到的。这个文件对应于版本库中的最新版本。

You can either launch an external merge tool / conflict editor with TortoiseSVN → Edit Conflicts or you can use any text editor to resolve the conflict manually. You should decide what the code should look like, do the necessary changes and save the file. Using a merge tool such as TortoiseMerge or one of the other popular tools is generally the easier option as they generally present the files involved in a 3-pane view and you don't have to worry about the conflict markers. If you do use a text editor then you should search for lines starting with the string <<<<<<<.

然后，执行命令 TortoiseSVN → 已解决 并提交人的修改到版本库。需要注意的是已解决命令并不是真正的解决了冲突，它只是删除了文件 文件名.扩展名.mine 和 文件名.扩展名.r*，允许你提交修改。

如果你的二进制文件有冲突，Subversion不会试图合并文件。本地文件保持不变(完全是你最后修改时的样子)，但你会看到文件名.扩展名.r*文件。如果你要撤消你的修改，保留版本库中的版本，请使用 SVN 还原(Revert)命令。如果你要保持你的版本覆盖版本库中的版本，使用已解决命令，然后提交你的版本。

你可以右击父文件夹，选择TortoiseSVN → 已解决...，使用“已解决”命令来解决多个文件。这个操作会出现一个对话框，列出文件夹下所有有冲突的文件，你可以选择将哪些标记成已解决。

4.6.2. #属性冲突

当两名或更多的开发人员修改了某个文件的属性时就会发生属性冲突。属性作为文件的一部分，解决属性冲突只能由开发人员来完成。

如果一个更改必须被另一个覆盖，那么就在 使用本地属性解决 和 使用远程属性解决 中选择一个。如果更改想要被合并，那就选择 手工编辑属性，选出所要的属性值然后标记为已解决。

4.6.3. #树冲突

当一名开发人员移动、重命名、删除一个文件或文件夹，而另一名开发人员也对它们进行了移动、重命名、删除或者仅仅是修改时就会发生树冲突。有很多种不同的情形可以导致树冲突，而且不同的情形需要不同的步骤来解决冲突。

当一个文件通过 Subversion 在本机删除后，文件也从本机文件系统中删除。因此即使它是树冲突的一部分，却既不能显示冲突的叠加图标也不能通过右键单击来解决冲突。使用检查修改对话框来获得编辑冲突选项。

TortoiseSVN 能够协助找到合并更改的正确位置，但是需要作一些额外的工作来整理冲突。请牢记：当进行一次更新操作后，工作副本的基础文件将会包括每一个项目在执行更新操作时版本库中的版本。如果你在进行更新后再撤销更改，工作副本将返回到版本库的状态，而不是你开始进行更改前的状态。

4.6.3.1. #本地删除，当更新时有更改进入

1. 开发人员 A 修改 Foo.c 并将其提交至版本库中。

2. 开发人员 B 同时在他的工作副本中将文件 Foo.c 改名为 Bar.c，或者仅仅是删除了 Foo.c 或它的父文件夹。

更新开发人员 B 的工作副本会导致树冲突:

- 在工作副本中，Foo.c 被删除了，但是被标记为树冲突。
- 如果冲突是由于更改文件名引起的而不是删除文件引起的，那么 Bar.c 被标记为添加，但是其中却不包括开发人员 A 修改的内容。

开发人员 B 现在必须做出选择是否保留开发人员 A 的更改。在更改文件名的案例中，他可以将 Foo.c 的更改合并到改名后的文件 Bar.c 中去。对于删除文件或文件夹的案例中，他可以选择保留包含开发人员 A 更改内容的项目并放弃删除操作。或什么也不做而直接将冲突标记为已解决，那样他实际上丢弃了开发人员 A 的更改。

如果 TortoiseSVN 能够找到被改名为 Bar.c 的原始文件，冲突编辑对话框将可以合并更改。这取决于在什么地方调用更新操作，它也许不能找到原始文件。

4.6.3.2. #本地更改，当更新时有删除进入

1. 开发人员 A 将文件 Foo.c 改名为 Bar.c 并将其提交至版本库中。

2. 开发人员 B 在他的工作副本中修改文件 Foo.c。

或者在一个文件夹改名的案例中...

1. 开发人员 A 将父文件夹 FooFolder 改名为 BarFolder 并将其提交至版本库中。

2. 开发人员 B 在他的工作副本中修改文件 Foo.c。

更新开发人员 B 的工作副本会导致树冲突。对于一个简单的文件冲突:

- Bar.c 被当作一个正常文件添加到工作副本中。
- Foo.c 被标记为添加(包括其历史记录)并且产生树冲突。

对于一个文件夹冲突:

- BarFolder 被当作一个正常文件夹添加到工作副本中。
- FooFolder 被标记为添加(包括其历史记录)并且产生树冲突。

  Foo.c 被标记为已修改。

开发人员 B 现在需要做出决定是否接受开发人员 A 作出的结构改变并且合并她的更改到新结构下适当的文件中，或者直接放弃开发人员 A 的更改并保留本地文件。

要合并她的本机更改到新布局中，开发人员 B 必须先找出冲突的文件 Foo.c 经过改名/移动后在版本库中的新文件名是什么。可以使用日志对话框来完成这个任务。更改必须要手工合并，因为没有办法自动的或者简单的完成此操作。一旦更改移植完毕，冲突的路径就是多余的并且可以删除。在此案例中，使用冲突编辑对话框中的删除按钮进行清理并将冲突标记为已解决。

如果开发人员 B 认为 A 的更改是错误的，那么在冲突编辑对话框中她必须选择保留按钮。这样就会标记冲突的文件/文件夹为已解决，但是需要手工删除开发人员 A 的更改。又是通过日志对话框帮助追踪哪些文件移动了。

4.6.3.3. #本地删除，当更新时有删除进入

1. 开发人员 A 将文件 Foo.c 改名为 Bar.c 并将其提交至版本库中。

2. 开发人员 B 将 Foo.c 改名为 Bix.c。

更新开发人员 B 的工作副本会导致树冲突:

- Bix.c 被标记为添加(包括其历史记录)。
- Bar.c 被添加到工作副本中，其状态为‘正常’。
- Foo.c 被标记为删除并且产生一个树冲突。

要解决这个冲突，开发人员 B 必须找出冲突的文件 Foo.c 经过改名/移动后在版本库中的新文件名是什么。可以使用日志对话框来完成这个任务。

然后，开发人员 B 需要决定 Foo.c 的新文件名中的哪一个需要保留 - 开发人员 A 改的那个还是他自己改的那个。

在开发人员 B 手工解决冲突后，使用冲突编辑对话框中的按钮将树冲突标记为已解决。

4.6.3.4. #本地缺少，当合并时有更改进入

1. 开发人员 A 在主干上工作，修改 Foo.c 并将其提交至版本库中

2. 开发人员 B 在分支上工作，将 Foo.c 改名为 Bar.c 并将其提交至版本库中

合并开发人员 A 的主干更改到开发人员 B 的分支工作副本会导致树冲突:

- Bar.c 已经存在于工作副本中，其状态为‘正常’。
- Foo.c 被标记为缺少并产生树冲突。

要解决这个冲突，开发人员 B 要在冲突编辑对话框中标记文件为已解决，这样就会将其从冲突列表中删除。她接下来需要决定是否将缺少的文件 Foo.c 从版本库中复制到工作副本中，是否将开发人员 A 的对 Foo.c 的更改和合并到改名后的 Bar.c 或者是否通过标记冲突为已解决来忽略更改什么事也不做。

注意，如果你将缺少的文件从版本库中复制到工作副本中然后再标记为已解决，你复制下来的文件将被再次删除。你必须先解决冲突。

4.6.3.5. #本地更改，当合并时有删除进入

1. 开发人员 A 在主干上工作，将 Foo.c 改名为 Bar.c 并将其提交至版本库中。

2. 开发人员 B 在分支上工作，修改 Foo.c 并将其提交至版本库中

当文件夹改名时有类似的案例，但是在 Subversion 1.6 中还未被识别...

1. 开发人员 A 在主干上工作，将父文件夹 FooFolder 改名为 BarFolder 并将其提交至版本库中。

2. 开发人员 B 在分支上工作，在她的工作副本中修改 Foo.c 。

合并开发人员 A 的主干更改到开发人员 B 的分支工作副本会导致树冲突:

- Bar.c 被标记为添加。
- Foo.c 被标记为修改并产生树冲突。

开发人员 B 现在需要做出决定是否接受开发人员 A 作出的结构改变并且合并她的更改到新结构下适当的文件中，或者直接放弃开发人员 A 的更改并保留本地文件。

要合并她的本机更改到新布局中，开发人员 B 必须先找出冲突的文件 Foo.c 经过改名/移动后在版本库中的新文件名是什么。可以通过适用于合并源码的日志对话框来完成这个任务。冲突编辑器仅显示工作副本的日志因为它不知道将哪个路径的更改合并进来，所以你需要自己找到它。更改必须要手工合并，

因为没有办法自动的或者简单的完成此操作。一旦更改移植完毕，冲突的路径就是多余的并且可以删除。在此案例中，使用冲突编辑对话框中的删除按钮进行清理并将冲突标记为已解决。

如果开发人员 B 认为 A 的更改是错误的，那么在冲突编辑对话框中她必须选择保留按钮。这样就会标记冲突的文件/文件夹为已解决，但是需要手工删除开发人员 A 的更改。又是通过日志对话框帮助追踪哪些文件移动了。

4.6.3.6. #本地删除，当合并时有删除进入

1. 开发人员 A 在主干上工作，将 Foo.c 改名为 Bar.c 并将其提交至版本库中。

2. 开发人员 B 在分支上工作，将 Foo.c 改名为 Bix.c 并将其提交至版本库中

合并开发人员 A 的主干更改到开发人员 B 的分支工作副本会导致树冲突:

- Bix.c 被标记为正常(未修改)状态。
- Bar.c 被标记为添加(包括其历史记录)。
- Foo.c 被标记为缺少并且产生树冲突。

要解决这个冲突，开发人员 B 必须先找出冲突的文件 Foo.c 经过改名/移动后在版本库中的新文件名是什么。可以通过适用于合并源码的日志对话框来完成这个任务。冲突编辑器仅显示工作副本的日志因为它不知道将哪个路径的更改合并进来，所以你需要自己找到它。

然后，开发人员 B 需要决定 Foo.c 的新文件名中的哪一个需要保留 - 开发人员 A 改的那个还是他自己改的那个。

在开发人员 B 手工解决冲突后，使用冲突编辑对话框中的按钮将树冲突标记为已解决。

4.6.3.7. #其它树冲突

有一些其它情况被标记为树冲突，因为冲突中卷入了文件夹而不是文件。例如你在主干和分支中添加了同名的文件夹然后在合并时就会发生树冲突。如果你想要保留合并目标中的文件夹，只要将冲突标记为已解决即可。如果你想要保留合并源中的文件夹，那么就要先使用 SVN 删除目标中的文件夹然后再合并。如果你需要处理更复杂的情况那就需要手工解决了。

4.7. #获得状态信息

当你在你的工作副本上工作时，你时常需要知道哪些文件你已经修改/增加/删除或改名了，或者甚至是哪个文件已经被其他人修改并提交了。

4.7.1. #图标重载

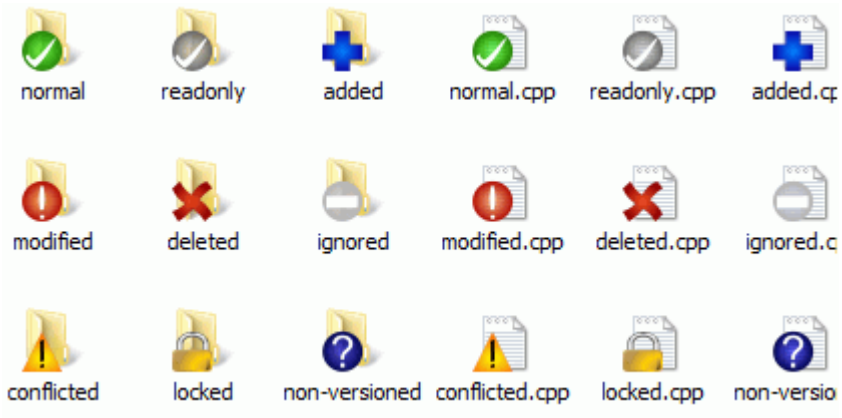


图 4.12. 显示重载图标的资源管理器

现在你已经从 Subversion 版本库中检出了一份工作副本，你可以在资源管理器中看一下这些文件的图标有什么变化。这也正是 TortoiseSVN 这么流行的原因之一。TortoiseSVN 加入了被称为重载图标的功能重载了原始的文件图标。根据文件的 Subversion 状态的不同，重载的图标也不同。


一个新检出的工作副本使用绿色的对勾做重载。表示 Subversion 状态正常.


在你开始编辑一个文件后，状态就变成了已修改，而图标重载变成了红色感叹号。通过这种方式，你可以很容易地看出哪些文件从你上次更新工作副本后被修改过，需要被提交。


如果在更新的过程中出现了冲突，图标会变成黄色感叹号。


如果你给一个文件设置了svn:needs-lock属性，Subversion 会让此文件只读，直到你获得文件锁。具有这个重载图标的文件来表示你必须在编辑之前先得到锁。


如果你拥有了一个文件的锁，并且 Subversion 状态是正常，这个重载图标就提醒你如果不使用该文件的话应该释放锁，允许别人提交对该文件的修改。


这个图标表示当前文件夹下的某些文件或文件夹已经被调度从版本控制中删除，或是该文件夹下某个受版本控制的文件丢失了。


加号告诉你有一个文件或目录已经被调度加入版本控制。


横条告诉你有一个文件或目录被版本控制系统所忽略。这个图标重载是可选的。


这个图标说明文件和目录未被版本控制，但是也没有被忽略。这个图标重载是可选的。

事实上，你会发现并不是所有的图标被你的系统使用。这是由于 Windows 允许的重载图标数量很有限，如果你同时使用旧版的 TortoiseCVS，就没有足够的重载可用。TortoiseSVN 试图成为一个“良好市民(TM)”，限制自身使用重载图标，为别的程序留下机会。

Now that there are more Tortoise clients around (TortoiseCVS, TortoiseHg, ...) the icon limit becomes a real problem. To work around this, the TortoiseSVN project introduced a common shared icon set, loaded as a DLL, which can be used by all Tortoise clients. Check with your client provider to see if this has been integrated yet :-)

要获得图表重载与 Subversion 状态的对应关系，或者其它技术细节，请阅读第 F.1 节 “图标重载”。

4.7.2. #详细状态

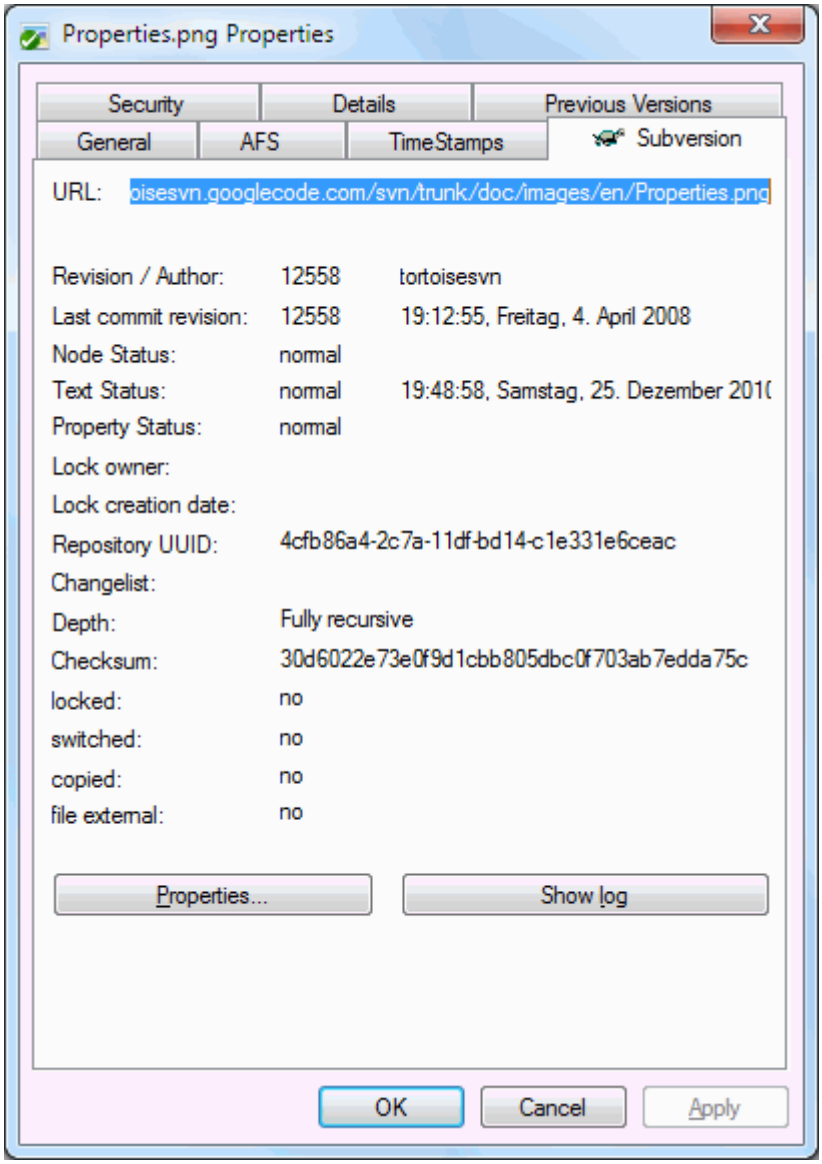


图 4.13. 资源管理器属性页，Subversion 页面

有时你可能想得到关于一个文件/目录的更多的细节信息而不仅是一个重载的标志。 你能得到Subversion的属性对话框中浏览到的所有信息。只需选择指定文件或目录，然后在文件菜单中选择Windows 菜单 → 属性(注意：这是资源管理器提供的标准属性菜单，而不是TortoiseSVN 子菜单的其中之一)。在TortoiseSVN 属性对话框中已经为在Subversion控制下的文件/目录增加新的属性页。在这里你能看到所有的关于选择文件/目录的相关信息。

4.7.3.#在 Windows 资源管理器中的 TortoiseSVN 列

在Windows资源管理器的详细信息视图中，附加列中可以显示与图标重载所表达相同的信息(还可以显示更多其他信息)。

右键点击列头，从出现的右键菜单中选择其他...。出现一个对话框，你可以指定在“详细信息”视图中要显示的列及其顺序。滚动对话框中的条目直到SVN开头的条目出现。在你要显示的条目上打勾，然后点击确定按扭关闭对话框。你选择的列就会出现在当前显示的列的右边。你可以通过拖放它们来重新排序或是修改列宽度，使它们适合你的需求。

Windows 资源管理器中的附加列在 Vista 系统中不可用，因为微软决定除了特定的文件类型外，不再允许为所有类型的文件显示这样的列。

如果你想要当前的布局对你所有的工作副本都有效，你可以考虑把它设成默认视图。

4.7.4. #本地与远程状态

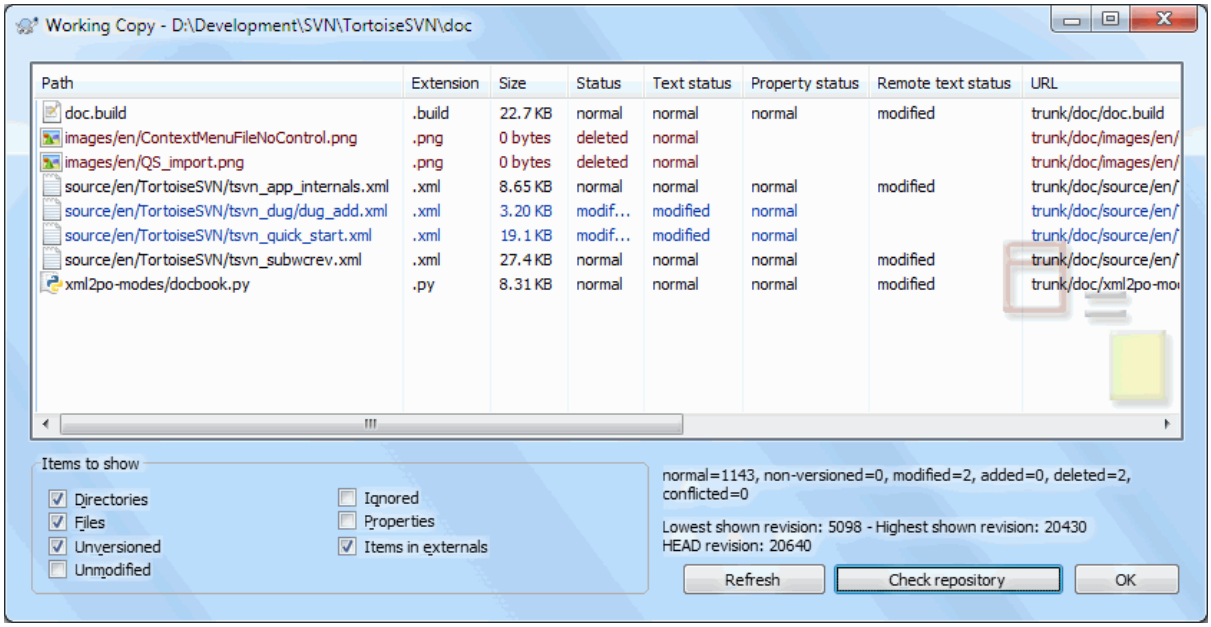


图 4.14. 检查修改

通常知道你修改了哪些文件以及哪些文件已经由另人修改并提交了是很有用的。这就是命令 TortoiseSVN → 检查修改 的用武之地了。这个对话框显示了所有你的工作副本中进行了任何形式的修改的的文件，也包括了当前存在的未受控的文件。

如果单击 检查版本库 就会同时查看版本库中的更改。通过这种方法可以在更新之前检查是否可能发生冲突。你也可以从版本库中只更新选中的文件而无需更新整个文件夹。在默认情况下，检查版本库 按钮只获取位于工作副本深度内的条目的远程状态。如果要查看版本库中所有的文件和文件夹，即使有些文件并没有被检出，那么就应该在单击 检查版本库 按钮时按住 Shift 键。

对话框使用颜色代码来高亮显示状态。

蓝色
: 本地被修改过的项

紫色
: 增加的项。那些被加入的项在文本状态列中有个+号表示，工具提示(tooltip)显示了该项是从哪里复制来的。

深红
: 删除的或是丢失的项。

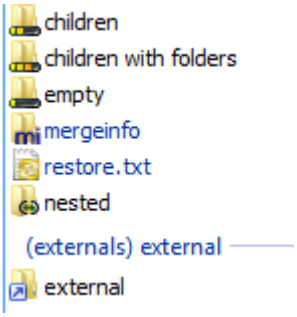
绿色
: 在本地和版本库中都有被修改过的项。改动将在更新的时候被合并。这种情况很可能在更新的时候产生冲突。

亮红
: 在本地被修改过但在版本库中已经被删除的项，或者在版本库中被修改但在本地被删除的项。这种情况必将在更新时产生冲突。

黑色
    未修改和未受控的项。

这是默认的配色方案，但你可以通过设置对话框来定制这些颜色。参考第 4.30.1.5 节 “TortoiseSVN 颜色设置”获得详情。

Overlay icons are used to indicate other states as well. The screenshot below shows all the possible overlays that are shown if necessary.



Overlays are shown for the following states:

- Checkout depth empty, meaning only the item itself.
- Checkout depth files, meaning only the item itself and all file children without child folders.
- Checkout depth immediates, meaning only the item itself and all file and folder children, but without children of the child folders.
- Nested items, i.e., working copies inside the working copy.
- External items, i.e., all items that are added via an svn:externals property.
- Items that are restored after a commit. See 第 4.4.3 节 “Commit only parts of files” for details.
- Items that have property modifications, but only to the svn:mergeinfo property. If any other property is modified, the overlay is not used.

Items which have been switched to a different repository path are also indicated using an (s) marker. You may have switched something while working on a branch and forgotten to switch back to trunk. This is your warning sign! The context menu allows you to switch them back to the normal path again.

在对话框的上下文菜单中你可以显示改变的差异。使用 上下文菜单 → 与基础版本比较检查你所作的本地修改。使用上下文菜单 → 使用标准差异格式显示差异检查版本库中别人作的修改。

你也可以还原单个文件的修改。如果不小心删除了某个文件，就会显示为 缺少 而且可以使用 SVN 还原 来恢复它。

可以使用右键菜单 → 删除将未版本控制的或忽略的文件丢到垃圾箱。如果你想彻底删除(不使用垃圾箱)，在点击删除时，请按着Shift键。

If you want to examine a file in detail, you can drag it from here into another application such as a text editor or IDE, or you can save a copy simply by dragging it into a folder in explorer.

这些列是可定制的。如果你右击任何一列的头部，你就会看到一个上下文菜单，允许你选择哪一列要显示。还可以在鼠标移动到列边界时通过拖动把手来改变列的宽度。这些定制的内容都会被保留下来，下一次你会见到相同的头部。

如果你同时做几个不相关的任务，也可以在修改列表中分组文件。阅读第 4.4.2 节 “修改列表”以获得更多信息。

在对话框的底部可以看到在你的工作副本中所使用的版本库版本号范围的总结。它们是 提交 的版本号，不是 更新 的版本号；它们表示这些文件上一次提交的版本号范围，不是它们被更新的版本号。注意，这个版本号范围仅仅对应于显示的条目，而不是整个工作副本。如果要查看整个工作副本的信息，必须要选中 未修改的 复选框。

如果你需要工作目录的全面视图，也就是所有文件和文件夹都同时显示，以便方便的使用检查修改。只要选择现实未修改文件检查栏，显示工作目录中的所有文件即可。

修复外部改名

有时候文件不是用 Subversion 改名，于是它们在文件列表中作为丢失和未版本控制的文件出现。为了避免丢失历史，你需要通知Subversion。简单的选择老名称(丢失)和新名称(未版本控制)，然后使用右键菜单 → 修复移动来指明这两个文件是改名关系。

修复外部复制

如果复制了一个文件，但不是通过 Subversion 的命令来做的，你可以修复此次复制来确保新文件不会丢失历史。只要简单的选择旧文件(正常或已修改)和新文件(无版本控制)，然后使用 右键菜单 → 修复复制来修复两个文件的复制关系。

4.7.5. #查看差别

通常你想要深入文件中了解你修改了什么。要达到这个目的，你可以选中这个文件，然后在TortoiseSVN的右键菜单中选择比较。这个操作会启动一个外部的差别检查程序，由它来比较当前文件与上一次检出或更新后的原始的副本(基础版本)。

即使你不是在一个工作副本中工作或者你有多个版本的文件，你都可以按以下方法来进行比较:

选择你要比较的两个文件(比如，你可以使用Ctrl 键加鼠标)，然后从TortoiseSVN的右键菜单中选择比较。最后一个被鼠标点中的文件(具有焦点的文件，比如有虚线框的文件具有焦点)，将作为被比较文件的后一个。

4.8. #修改列表

理想情况下，你任何时候都只做一件事，你的工作副本只包含一个逻辑修改集合。很好，回到现实。你经常会同时做几件不相关的事，当你察看提交对话框时，所有修改混到一起。修改列表特性帮助你分组，让你容易看到正在做什么。当然它只能在修改不重合的时候工作。如果两个不同的任务影响到同一个文件，没有办法隔离修改。

在很多地方可以看到修改列表，但是最常见的是提交对话框和检查修改对话框。让我们从检查修改对话框开始--在你完成了多个特性和很多文件后的检查修改对话框。当你第一次打开对话框，所有的修改过的文件被列在一起。假设你现在想要组织任务并且按照特性将这些文件分组。

选择一个或多个文件并且使用右键菜单 → 移动到修改列表可以增加一个项目到修改列表。最初没有修改列表，所以你第一次做的时候，会创建一个新的修改列表。给出一个能描述它的作用的名称，然后点击确定。提交对话框会改变为显示项目分组。

当你创建好修改列表后，你就可以将文件拖放进去，既可以从其它修改表中拖过去，也可以从 Windows 资源管理中拖过去。将一个未被修改的文件加入修改列表时，从 Windows 资源管理器拖拽就很有用。你可以在检查修改对话框中这样做，但要显示未修改的文件。

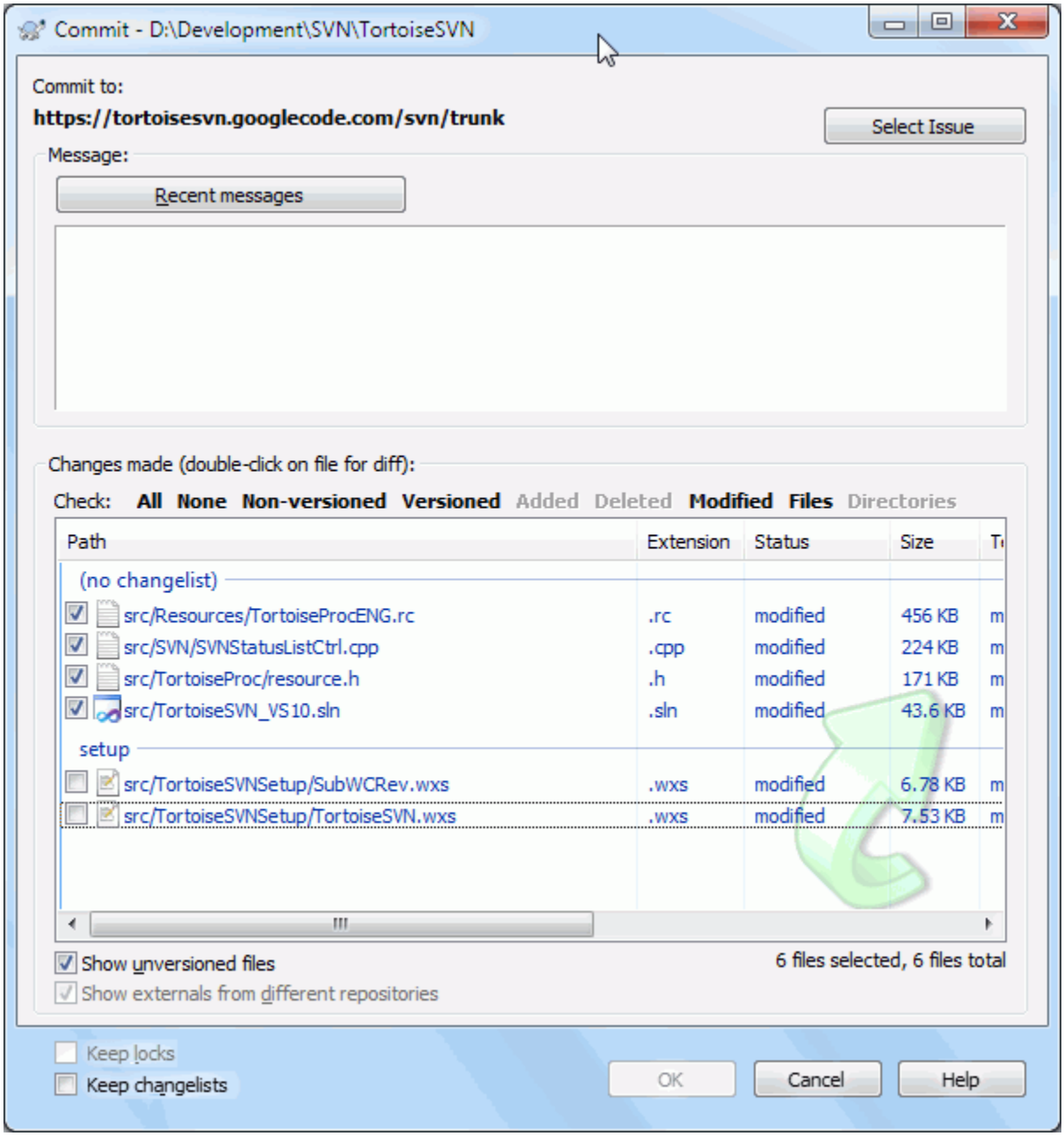


图 4.15. 带有修改列表的提交对话框

在提交对话框中可以看到被修改列表分组的那文件。除了分组可以直接指示之外，你也可以使用组头选择提交哪些文件。

在 XP 中，当你右键点击组头的时候，会有一个右键菜单让你选定或取消选定全组的条目。但在 Wista 中右键菜单不是必需的。点击组头可以选择全部条目，然后点击被选中的任一个条目就可以选中全组。

TortoiseSVN 保留了一个它自己使用的修改列表名称--ignore-on-commit。这个列表用于标记某些版本控制的文件，它们在本地被修改你却不打算提交他们。这个特性在 第 4.4.4 节 “从提交列表中排除项目” 中有描述。

当提交属于修改列表的文件后，通常情况下用户不再需要修改列表的成员关系。所以在默认情况下，当提交时文件会从修改列表中除去。如果你希望文件被保留在更改列表中，选中提交对话框底部的 保持修改列表。

修改列表是一个作用范围仅限于本地客户端的特性。创建和删除修改列表不会影响版本库或其他人的工作副本。这只是一个让你能够整理文件的便利工具。

4.9. #版本日志对话框

对于每次进行修改和提交，你应该有针对性地留下日志信息。这样，你就可以在以后方便地看到你都做了什么，为什么这么做。当然这么做还是你拥有了开发过程的详细日志。

版本日志对话框可以获取所有的日志信息，并将其显示出来。对话框的视图分成3个面板。

- 最上方的面板显示了版本的列表。这其中包含了日期和时间，以及提交的用户和日志信息开头的部分内容。

  以蓝色显示的行表示某些内容被复制到该开发版本中(可能是从一个分支中复制而来)。

- 中间的面板显示了被选中的版本的完整的日志信息。
- 最下面的面板显示了被选中版本中都对哪里文件和文件夹进行了修改。

当然，对话框的作用不止于此——它提供了右键菜单，通过它可以获取更多的项目历史信息。

4.9.1. #调用版本日志对话框

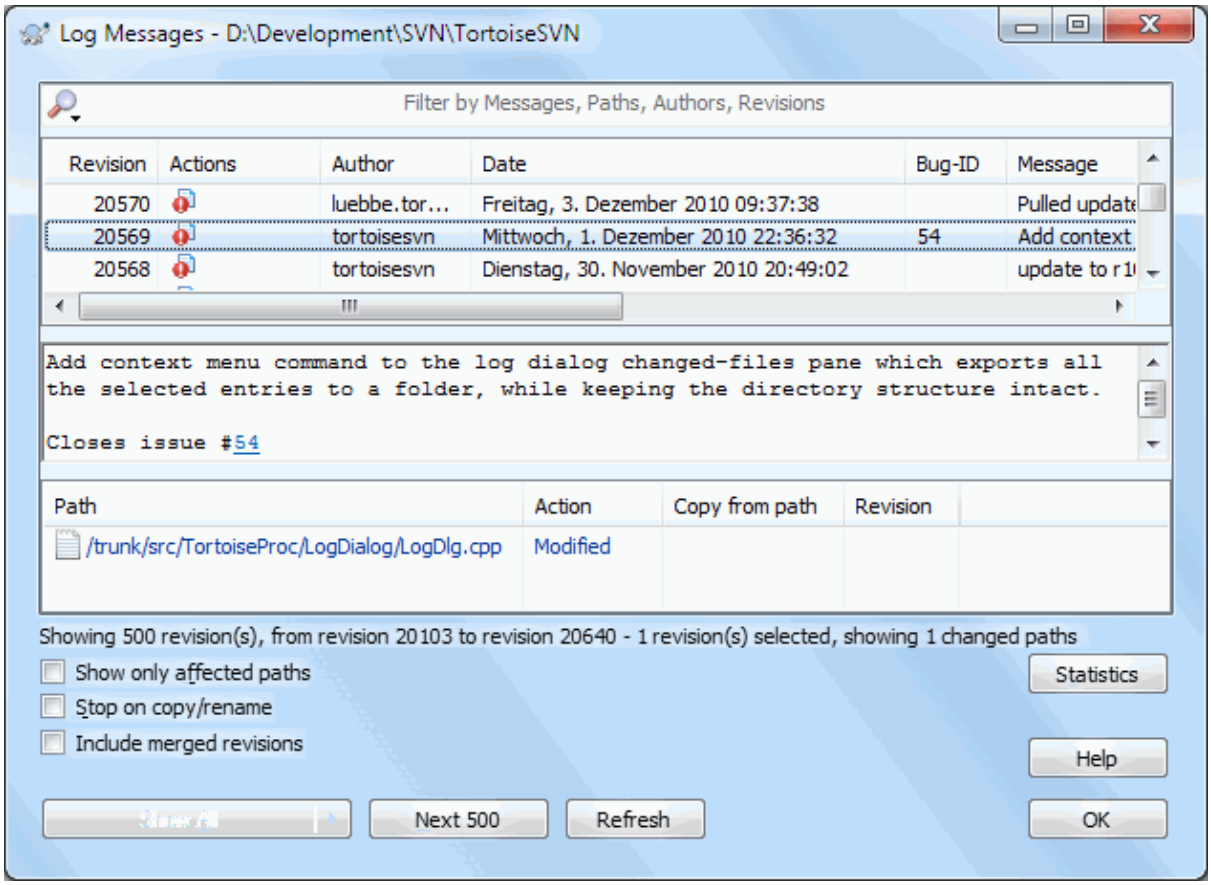


图 4.16. 版本日志对话框

有几种途径可以调出日志对话框:

- 从右键菜单的TortoiseSVN子菜单中调用
- 从属性页中调用

• 在更新结束后，从进度对话框中调用。在这里，日志对话框只显示你上一次更新以来的版本变化。

如果版本库不可用，你将会看到 要离线? 对话框，在 第 4.9.10 节 “离线方式”中有详细描述。

4.9.2. #版本日志动作

顶部面板有个动作列，包含了此版本的动作概要图标。有四个不同的图标，每个都在自己的列显示。

如果某个版本修改了文件或目录，已修改 图表就会在首列显示。

如果某个版本增加了文件或目录，已增加 图表就会在第二列显示。

如果某个版本删除了文件或目录，已删除 图表就会在第三列显示。

如果某个版本替换了文件或目录，已替换 图标就会在第四列显示。

4.9.3. #获得更多信息

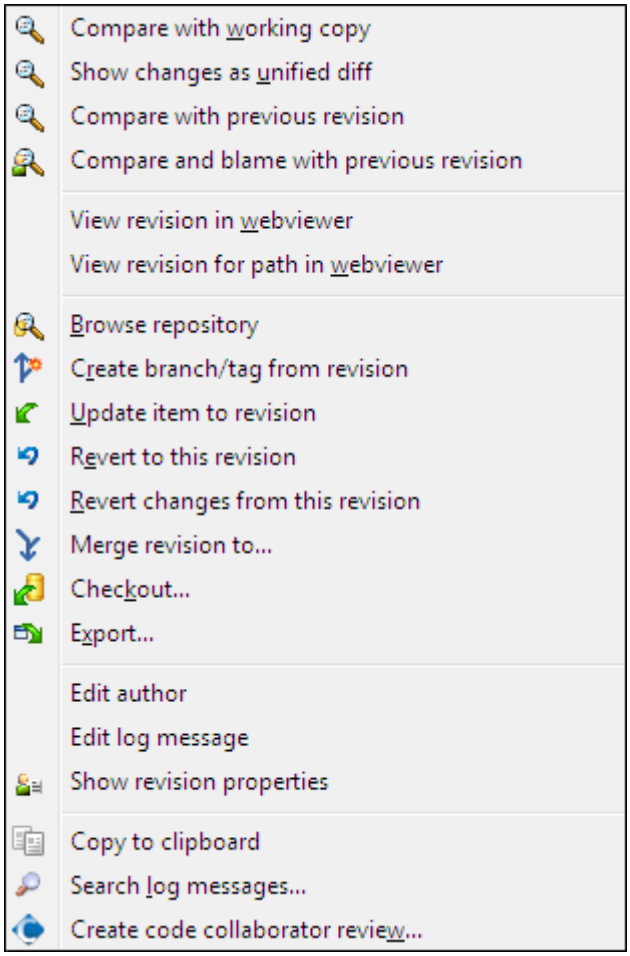


图 4.17. 版本日志对话框的顶部面板的右键菜单

The top pane of the Log dialog has a context menu that allows you to access much more information. Some of these menu entries appear only when the log is shown for a file, and some only when the log is shown for a folder.

与工作副本比较
将你的工作版本与选中的版本进行比较。默认的比较工具是与 TortoiseSNV 一同发布的 TortoiseMerge，如果日志对话框是针对文件夹的，那么就会出现一个被修改的文件的列表，你可以单独地查看每个文件所做的修改。

与工作基础版本比较并追溯
追溯文件的选中版本与你工作的 BASE 版本，使用可视化差异工具显示差异。详情请参阅第 4.23.2 节 “追溯不同点”。(仅适用于文件)

以标准差异文件显示改变
将选中的版本作为单一差异文件(GNU补丁格式)查看。相对于可视化的文件比较器，它更难阅读，但它将所有的变化显示在一个格式更为紧凑的文件中。

If you hold down the Shift key when clicking on the menu item, a dialog shows up first where you can set options for the unified diff. These options include the ability to ignore changes in line endings and whitespaces.

与前一版本比较
比较选中的版本和以前版本。它与比较工作副本类似。对于文件夹，这个选项首先会显示已修改的文件对话框让你选择要比较的文件。

与前一版本比较并追溯
显示修改的文件对话框让你选择文件。追溯选中的版本和前一版本，并使用可视化比较工具显示差异。(仅适用于文件夹)

保存版本至...
将选中的版本保存成文件，你可以得到一份该文件的旧版本。(仅适用于文件)

打开 / 打开方式...
用默认查看器或你指定的程序打开选中文件的选中版本。(仅适用于文件)

追溯...
追溯文件直到选中的版本。(仅适用于文件)

浏览版本库
打开版本库浏览器，基于选中的版本，在版本库中检查选中的文件或目录。

从版本创建分支/标记
从选中的版本建立一个分支/标记。这个选项很有用。比如: 如果你忘记建立标记，并且提交了某些你不想使其进入发行版的修改。

更新项目至版本
将你的工作副本更新到选中的版本。如果你想要你的工作副本折返到过去的某个时间，或者在版本库中有一系列提交而你想每次只更新工作副本一小步，那这个功能就很好用。你最好是更新工作副本的整个目录而不是单一某个文件，因为如果只更新某个文件，否则你的工作副本就可能不一致。

如果你想要永久撤销先前的更改，使用 复原到此版本。

复原到此版本
恢复到某个以前的版本。如果你做了多处修改，然后决定要返回到版本 N，你就可以使用这个命令。恢复的修改位于你的工作副本，在你提交之前，并不会影响版本库。注意，这将会丢弃从那个版本以来的所有修改，使用选中的版本来替换文件/文件夹。

如果你的工作副本处于未修改的状态，在执行此操作后，工作副本将会显示为已修改。如果你已经进行了本地修改，这个命令将会把撤销的改变合并至你的工作副本中。

内部的动作是 Subversion 对选中版本之后的修改内容执行了反向合并，撤销这些先前提交产生的影响。

如果在执行这个动作后你察觉到你需要撤销这次撤销并且让工作副本返回到先前没有修改的状态，你应该在 Windows 资源管理器中使用 TortoiseSVN → SVN 还原， 这个命令将会丢弃本次撤销动作带来的修改。

如果你只是想看看某个文件或者文件夹在先前的版本是什么样子，使用 更新至版本 或 保存版本为... 功能替代此操作。

复原此版本作出的修改
还原选中版本所做的修改。还原的内容只在你的工作副本中，所以此操作完全不会影响版本库！要注意的是，这个操作仅仅还原该版本中的修改。不是将整个文件替换成选中的那个版本。它对于已经做过其它无关修改的还原早期修改非常有用。

如果你的工作副本处于未修改的状态，在执行此操作后，工作副本将会显示为已修改。如果你已经进行了本地修改，这个命令将会把撤销的改变合并至你的工作副本中。

内部的动作是 Subversion 对这个版本的修改内容执行了反向合并，撤销先前提交产生的影响。

你可以使用上文复原到此版本中描述的撤销这次撤销。

合并版本到...
合并选中的版本到不同的工作副本。可以通过文件夹选择对话框来确定合并到哪一个工作副本中，但是此操作没有冲突对话框，也没有尝试测试合并的机会。合并到未修改的工作副本是一个好主意，这样当合并不成功时你可以还原工作副本。当你想要将某个分支上选中的版本合并至其他分支时，这个功能很有用。

检出...
检出你选择的目录的选中版本，创建一个全新副本。它弹出对话框，让你确认URL和版本，并且选择保存的位置。

导出...
导出选择的文件/目录的选中版本。它弹出对话框，让你确认URL和版本，选择导出位置。

编辑作者 / 日志信息
编辑之前提交时的日志信息或是作者。请阅读第 4.9.7 节 “修改日志消息和作者”了解其工作原理。

显示版本属性
查看和编辑任何版本属性，不仅仅是日志信息和作者。参看 第 4.9.7 节 “修改日志消息和作者”。

复制到剪贴板
将选中版本的详细日志信息复制到剪贴板。它会复制版本号，作者，日期，日志信息，以及每个版本的改变项目列表。

查找日志信息...
在日志信息中搜索你输入的的文字。这个操作搜索日志信息，也搜索由Subversion建立的提交行为总结(最底部的面板中的内容)。搜索大小写无关。

Create code collaborator review...
This menu is shown only if the SmartBear code collaborator tool is installed. When invoked for the first time, a dialog is shown prompting the user to enter user credentials for both code collaborator and SVN. Once the settings are stored, the settings dialog is no longer shown when the menu is invoked, unless the user holds Ctrl while executing the menu item. The configuration and the chosen revision(s) are used to invoke the code collaborator graphical user interface client, which creates a new review with the selected revisions.

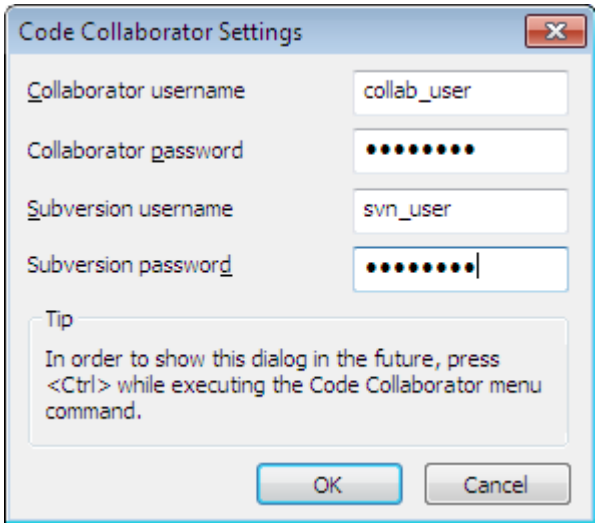


图 4.18. The Code Collaborator Settings Dialog

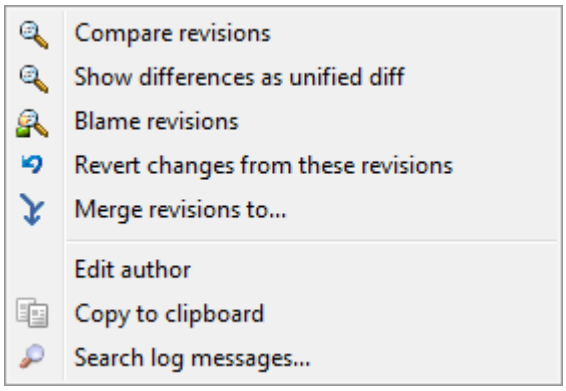


图 4.19. 选中两个版本的顶部面板的右键菜单

如果你使用Ctrl组合键一次选中了两个版本，右键菜单有所改变:

比较版本差异
: 使用可视化差异比较工作比较两个选中的版本。默认的比较工作是与TortoiseSVN一起提供的TortoiseMerge。

  如果你是针对文件夹选中这个选项，则会弹出一个对话框列出修改过的文件，提供了更多的差异比较选项。请参考比较版本对话框获得详情: 第 4.10.3 节 “比较文件夹”

追溯版本
: 追溯两个版本，并使用可视化差异工具显示差异。详情请参考第 4.23.2 节 “追溯不同点”。

以标准差异文件显示修改
: 使用单一差异文件显示差异。这对文件和文件夹都有效。

复制到剪贴板
: 如前所述将日志消息复制到剪贴板。

查找日志信息...
: 如前所述可以搜索日志消息。

如果你用 Ctrl 或 Shift 组合键选择了两个或多个版本，右键菜单将有一个选项，可以让你还原这些选中的版本中的修改。这是一次性还原一组版本中修改的最简方法。

你也可以合并选中的版本到别的工作副本，就像上面描述的那样。

如果所有选中的版本作者是相同的，你可以同时修改这些版本的作者。

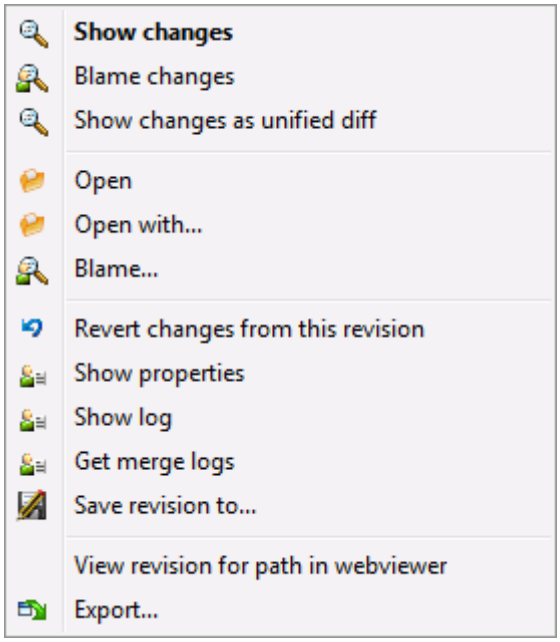


图 4.20. 日志对话框的底部面板的右键菜单

The bottom pane of the Log dialog also has a context menu that allows you to

显示改变
显示选中的文件在选中的版本所做的更改。

追溯改变
追溯选中文件的选中版本与前一个版本，使用可视化差异工具显示差异。详情请参阅第 4.23.2 节 “追溯不同点” 。

以标准差异格式显示改变
以标准差异格式显示改变。这个菜单条目只对显示为已修改的文件有效。

打开 / 打开方式...
用默认查看器或你指定的程序打开选中文件的选中版本。

追溯...
打开追溯对话框，你可以追溯到选中的版本。

复原此版本作出的修改
还原选中文件的选中版本所作的变更。

显示属性
查看选中项的Subversion属性。

显示日志
显示选中的单个文件的版本日志。

取得合并日志
显示被选中的单个文件的版本日志，包括合并修改。在 第 4.9.6 节 “合并跟踪特性”中查看更多信息。

保存版本至...
将选中的版本保存成文件，你可以得到一份该文件的旧版本。

导出...
Export the selected items in this revision to a folder, preserving the file hierarchy.

When multiple files are selected in the bottom pane of the Log dialog, the context menu changes to the following:

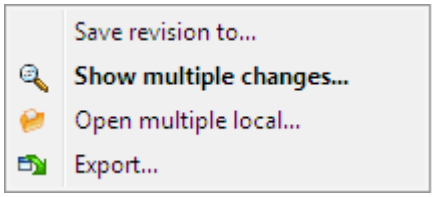


图 4.21. The Log Dialog Bottom Pane with Context Menu When Multiple Files Selected.

保存版本至...
将选中的版本保存成文件，你可以得到一份该文件的旧版本。

Show multiple changes...
Show changes made in the selected revision for the selected files. Note that the show changes functionality is invoked multiple times, which may bring up multiple copies of your selected diff tool, or just add a new comparison tab in your diff tool. If you have selected more than 15 files, you will be prompted to confirm the action.

打开多个本地项目...
This will open local working copy files that correspond to your selected files using the application that is registered for the extension. [The behavior is the one you would get double-clicking the working-copy file(s) in Windows explorer]. Depending on how your file extension is associated to an application and the capabilities of the application, this may be a slow operation. In the worst case, new instances of the application may be launched by Windows for each file that was selected.

If you hold Ctrl while invoking this command, the working copy files are always loaded into Visual Studio. This only works when the following conditions are met: Visual Studio must be running in the same user context while having the same process integrity level [running as admin or not] as TortoiseProc.exe. It may be desirable to have the solution containing the changed files loaded, although this is not strictly necessary. Only files that exist on disk with extensions [.cpp, .h, .cs, .rc, .resx, .xaml, .js, .html, .htm, .asp, .aspx, .php, .css and .xml] will be loaded. A maximum of 100 files can be loaded into Visual Studio at one time, and the files are always loaded as new tabs into the currently open instance of Visual Studio. The benefit of reviewing code changes in Visual Studio lies in the fact that you can then use the built-in code navigation, reference finding, static code analysis and other tools built into Visual Studio.

导出...
Export the selected files/folder at the selected revision. This brings up a dialog for you to confirm the URL and revision, and select a location for the export.

你可能会注意到，我们有时候说改变，有时候说差异。它们的区别在哪儿？

Subversion 使用版本号代表 2 种不同的东西。版本通常表示版本库在某一个时间点的状态，但它也可以表示创建该版本时的更改集合，例如 “在 r1234 完成” 表示在 r1234 提交的更改实现了 X 功能。为了避免混淆，我们使用两个不同的术语。

如果你选择了两个版本 N 和 M，上下文菜单会显示这两个版本的差异。用 Subversion 术语说，就是diff -r M:N。

如果你选择了一个版本 N，上下文菜单会显示这个版本的改变。用 Subversion 术语说，就是diff -r N-1:N 或 diff -c N。

底部面板显示在所有选中版本中被修改的文件，所以右键菜单通常会提供显示改变。

4.9.4. #获取更多的日志信息

日志对话框并不总是显示所有曾经的修改，日志不显示的可能原因如下:

- 对于一个大的库，可能存在几百上千个改动，全部得到它们可能要花上很久的时间。通常你只关心最近的修改。默认情况下，日志消息限制只获取100条，但你可以在TortoiseSVN → 设置中修改这个值(第 4.30.1.2 节 “TSVN对话框设置一”)，
- 当复制/重命名时停止复选框被选中时， 如果选中的文件或文件夹是从版本库中的其他地方复制而来的，显示日志将停止在该点。这对于查看分支(或标记)时很有用，因为它会停在分支的根节点上，可以快速查看该分支的修改。

  一般情况下你可以不要勾选它。 TortoiseSVN会记住它的状态，以改进性能。

  如果你在从合并对话框中调用的显示日志对话框，那么这个复选框默认将总是选中的。这是由于合并通常都是查看分支中的修改， 获取分支的根之前的日志在这种情况下通常没有什么意义。

  注意，Subversion当前是用复制/删除来实现重命名的，所以重命名一个文件或文件夹也会造成日志显示停止(如果选择了复制/重命名时停止)在该点。

如果你要查看更多的日志信息，点击下100个，以获取下100个日志信息。如果有需要你可以多次重复这个操作。

这个按钮旁边的是一个多功能按钮，它可以记住上一次你要它进行的操作。点击它上面的箭头，可以看到更多的选项。

如果你要查询指定范围的版本，使用显示范围 ...。这会出现一个对话框，要求输入开始和结束的版本。

如果你要查询从最新版本直到版本1的所有的日志消息，使用显示所有。

当日志对话框显示后有其他人进行了提交的情况下，可以单击 F5 键来刷新最新的版本。

单击 Ctrl-F5 键来刷新日志缓存。

4.9.5. #当前工作副本的版本

因为日志对话框从最新版本开始显示日志，而不是从当前工作副本的版本开始。还未被更新至工作副本中的版本的日志消息也经常会显示出来。为了使其更清楚，符合当前工作副本版本的提交信息使用粗体显示。

当你显示文件夹的日志时，高亮的版本是此文件夹中的最高版本，这就需要遍历工作副本。遍历操作在单独的进程中进行，因此不会使显示日志有延迟，但是这样的结果就是高亮也许不会立刻显示出来。

4.9.6. #合并跟踪特性

Subversion 1.5 及以后的版本使用属性保留合并记录。关于已合并的修改，我们可以获得更详细的历史。例如，你在分支中开发了一个新特性并且将此分支合并到主干，此特性开发将会以一次合并提交的形式显示在主干的日志中，即使在分支开发中可能有 1000 次提交。

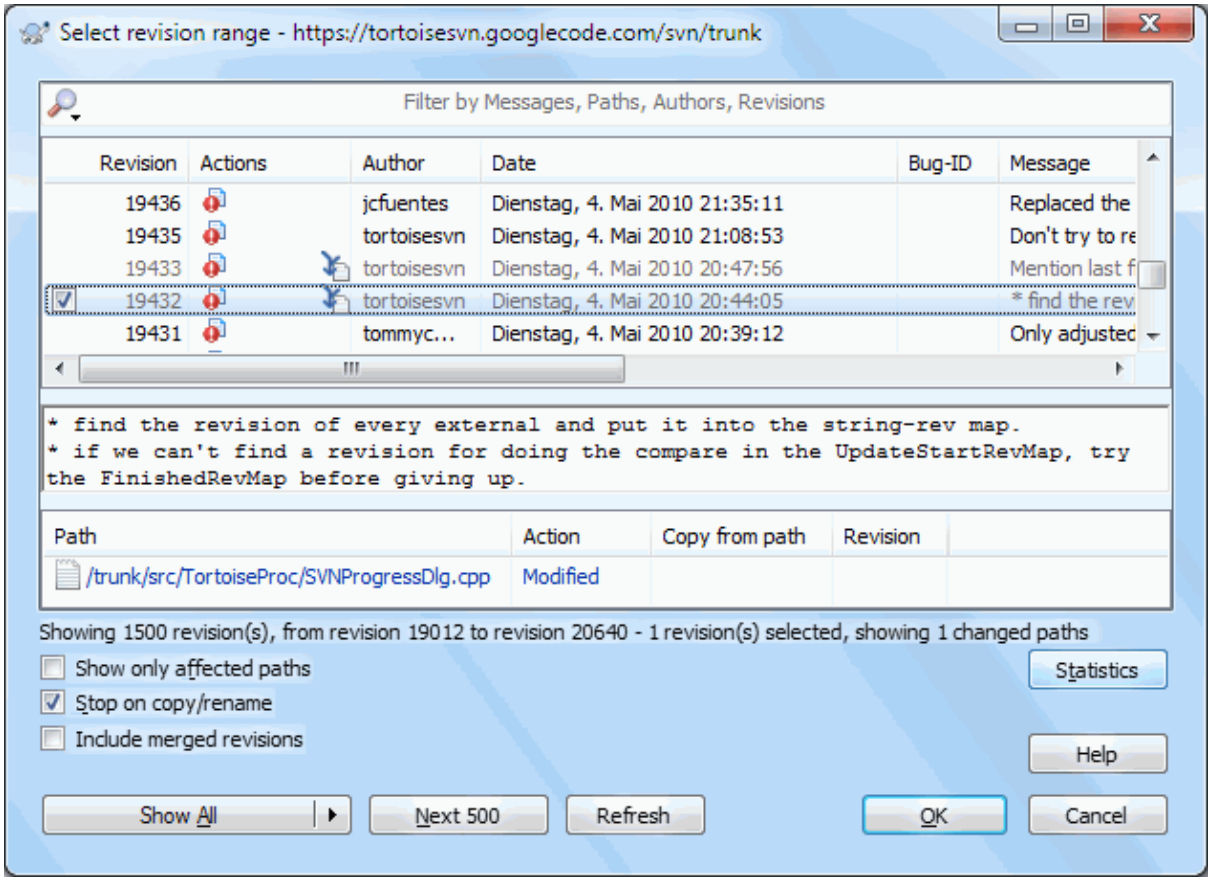


图 4.22. 日志对话框显示合并跟踪版本

如果你想查看此次提交中有哪些版本被合并的详细信息，选中 包含合并版本。这样将会再次获取日志信息，同时也会插入被合并的版本的日志信息。被合并的版本使用灰色显示，因为它们代表在版本库中不同部分的修改。

当然，合并绝非简单！在分支上进行特性开发过程中，也许不时的会将主干的内容合并至分支使分支保持同步。所以分支的合并历史将会包含其它层次的合并历史。这些不同层次的信息在日志对话框中使用不同的缩进级别显示。

4.9.7. #修改日志消息和作者

版本属性完全不同于其它的 Subversion 属性。版本属性是关联于版本库中的特定版本号的描述项目，例如日志消息，提交日期和提交者名称(作者)。

有时你可能想要修改你曾经输入的日志消息，也许是因为有拼写错误或是你想改进消息内容，或是其他别的原因。偶尔你还想修改提交者，可能是你忘了设置认证等原因。

Subversion lets you change revision properties any time you want. But since such changes can't be undone (those changes are not versioned) this feature is disabled by default. To make this work, you must set up a pre-revprop-change hook. Please refer to the chapter on Hook Scripts [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.create.html#svn.reposadmin.create.hooks] in the Subversion Book for details about how to do that. Read 第 3.3 节 “服务器端钩子脚本”to find some further notes on implementing hooks on a Windows machine.

一旦你按需要为服务器设置了钩子，你就可以使用日志对话框顶部面板的右键菜单来修改任意版本的作者和日志信息(或其它版本属性)了。你也可以使用中间面板的右键菜单编辑日志信息。

由于 Subversion 的版本属性不受版本控制，对于这种属性(如 svn:log提交信息属性)作出的修改将永久覆盖该属性之前的值。

因为 TortoiseSVN 保存了对所有日志信息的缓存，对作者和日志信息的修改仅会在本地显示出来。其他 TortoiseSVN 用户还将会看到缓存的(旧的)作者和日志消息直到他们刷新了日志缓存。

4.9.8. #过滤日志信息

如果你只想要显示上千条日志中你所感兴趣的日志，你可以使用日志对话框顶部的过滤器控件。开始和结束日期控件允许你查看指定日期范围内的输出。查找框帮你查出含有指定内容的信息。

单击查找图标可以选择在哪些信息中查找，并选择 正则表达式 模式。正常情况下你可能只需要进行简单的子字符串查找，但如果你需要更有弹性的查找条件，就可以使用正则表达式。如果将鼠标在文本框上停留一会，就会出现一个工具提示条显示一些如何使用正则表达式功能和子字符串功能的使用帮助。过滤器检查字符串是否与日志内容匹配，并且只有 匹配 过滤字符串的条目才会显示。

Simple sub-string search works in a manner similar to a search engine. Strings to search for are separated by spaces, and all strings must match. You can use a leading - to specify that a particular sub-string is not found (invert matching for that term), and you can use ! at the start of the expression to invert matching for the entire expression. You can use a leading + to specify that a sub-string should be included, even if previously excluded with a -. Note that the order of inclusion/exclusion is significant here. You can use quote marks to surround a string which must contain spaces, and if you want to search for a literal quotation mark you can use two quotation marks together as a self-escaping sequence. Note that the backslash character is not used as an escape character and has no special significance in simple sub-string searches. Examples will make this easier:

Alice Bob -Eve

searches for strings containing both Alice and Bob but not Eve

Alice -Bob +Eve

searches for strings containing both Alice but not Bob, or strings which contain Eve.

-Case +SpecialCase

searches for strings which do not contain Case, but still include strings which contain SpecialCase.

!Alice Bob

searches for strings which do not contain both Alice and Bob

!-Alice -Bob

do you remember De Morgan's theorem? NOT(NOT Alice AND NOT Bob) reduces to (Alice OR Bob).

"Alice and Bob"

searches for the literal expression “Alice and Bob”

""

searches for a double-quote anywhere in the text

"Alice says ""hi"" to Bob"

searches for the literal expression “Alice says "hi" to Bob”.

如何使用正则表达式进行查找超出了本手册的范围，你可以查看在线文档和教程 http://www.regular-expressions.info/。

要注意的是，这些过滤器只对已经获取的信息有效。它们并不从版本库中下载信息。

在底部的面板选中 仅显示影响的路径 复选框可以过滤路径名称。影响的路径就是那些用来显示日志的路径。如果是获取文件夹的日志，就是指文件夹中的或者以下的任何内容。对于文件，就是指该文件。正常情况下，路径列表会显示受本次提交影响的其它路径，但是是灰色的。如果选中该复选框，这些路径就会被隐藏。

有时，工作规范要求日志消息符合一个特定的格式，这就意味着描述修改的文本不能在顶部面板中以简短的摘要形式显示。属性 tsvn:logsummary 可以用于提取日志消息的一部分显示在顶部面板中。参阅第 4.17.2 节 “TortoiseSVN 项目属性”了解如何使用这个属性。

在版本库浏览器中没有日志格式化

Because the formatting depends upon accessing Subversion properties, you will only see the results when using a checked out working copy. Fetching properties remotely is a slow operation, so you will not see this feature in action from the repo browser.

4.9.9.#统计信息

统计按扭，可以显示一些你感兴趣的关于日志对话框中版本的信息。可以显示已经有几个作者做了工作，他们各提交了几次，按周的统计，等等。现在，你可以发现一个大概情况: 谁最勤快，谁偷懒。;-)

4.9.9.1.#统计页

此页可以提供所有你可以想到的数据，特别是周期和包括的版本数，还有一些最大/最小/平均值。

4.9.9.2. #作者提交次数统计页

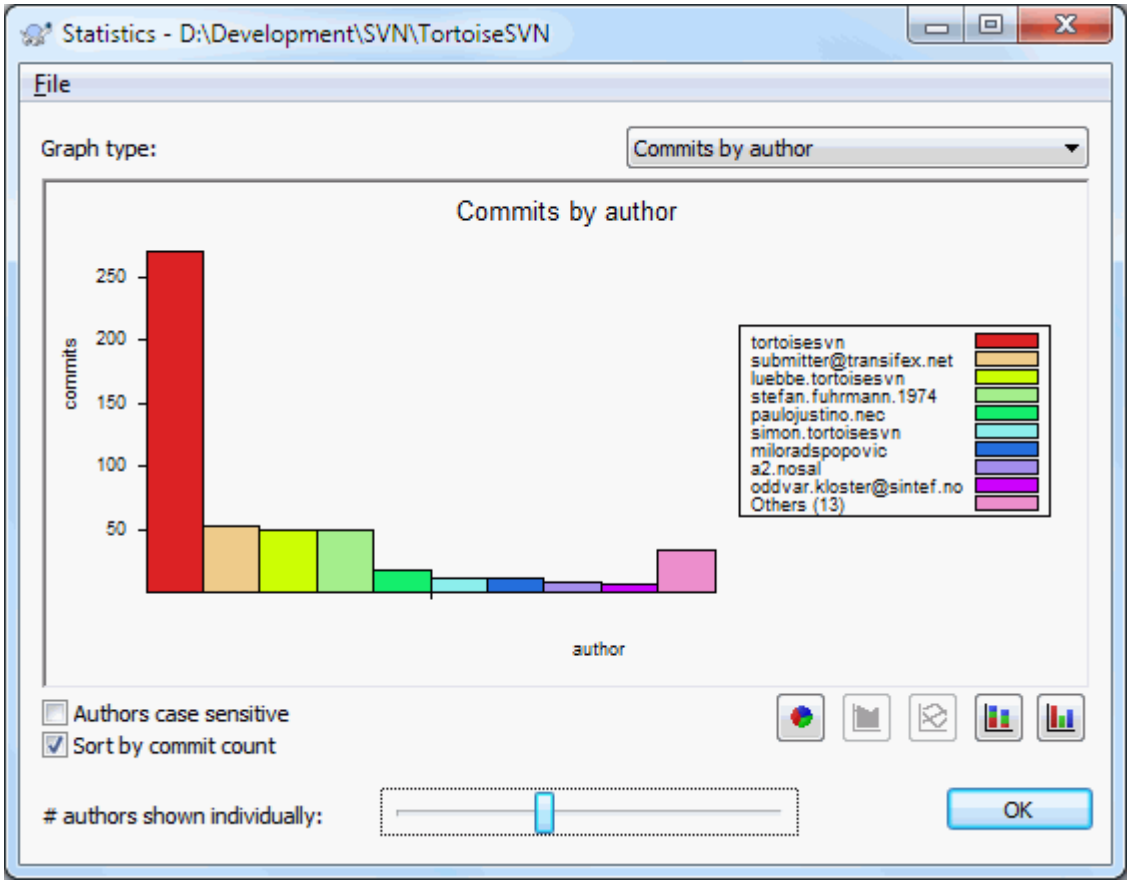


图 4.23. 作者提交次数统计柱状图

此图用简单柱状图、叠加柱状图或饼图显示了哪些作者已经在项目中活跃了。

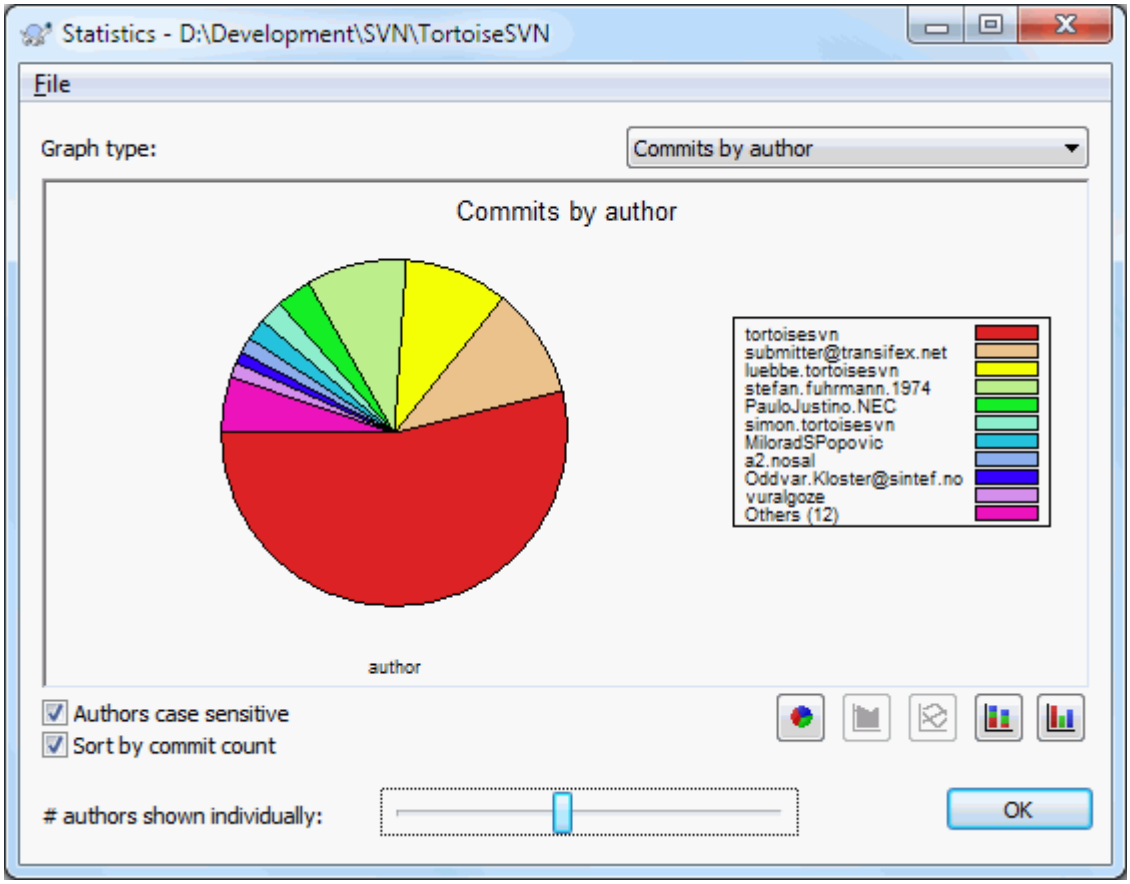


图 4.24. 作者提交次数统计饼图

其中有几个主要作者和许多辅助的贡献者。由于太小的部分会导致图形难于阅读，所以在底部有个滑动条，可以设置一个范围(占所有提交的百分比)，在这个范围下的所有行为都整合成其他类。

4.9.9.3. #按日期提交统计页

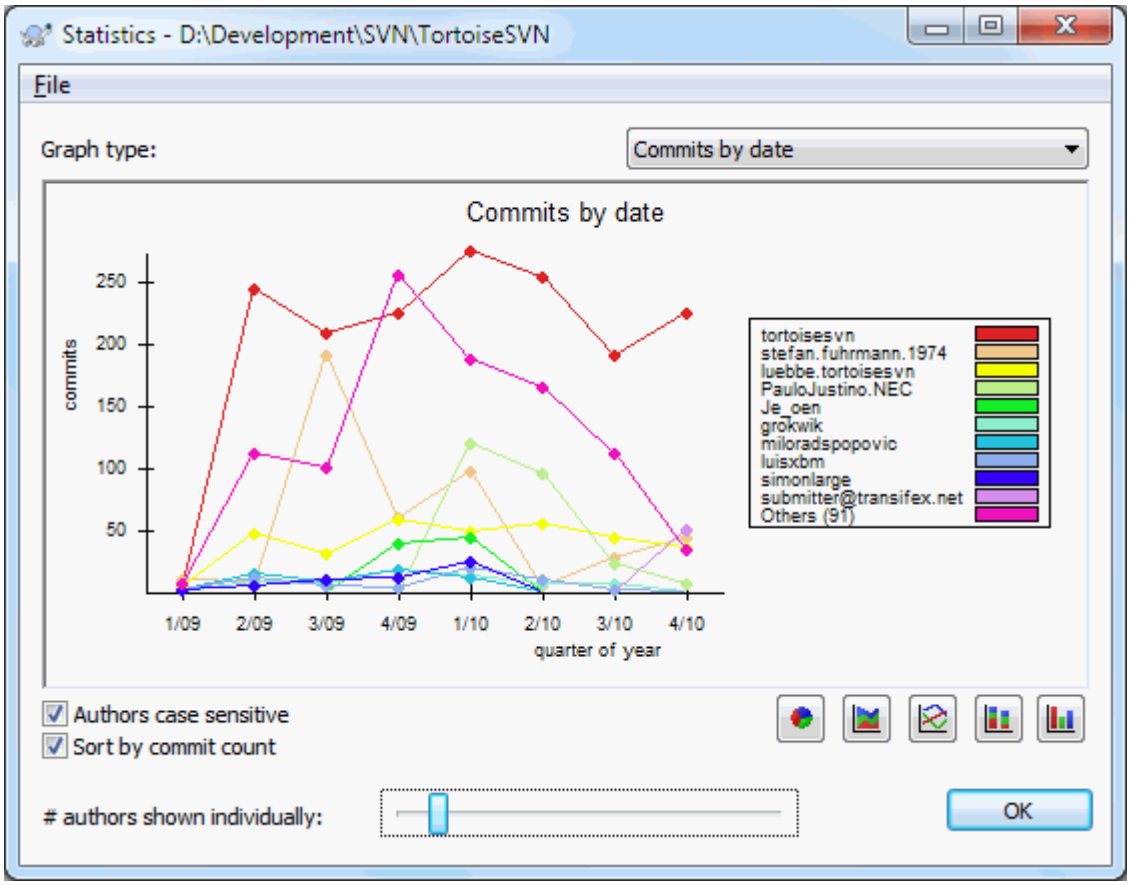


图 4.25. 按日期提交统计图

本页图示了以提交次数和作者作为条件的项目行为统计。这里可以看出项目什么时候有人在工作，以及什么人在什么时候进行了工作。

如果有多个作者，你就会在图中看到多行。有两种视图可用正常，在这里，每个作者的行为都相对于基线；叠加，在这里每个作者的行为是相对于他的下面那条线。后一种视图避免了线的交叉，对于图来说更明了，但对查看一个作者的输出比较不直观。

默认统计是区别大小写的，也就是说用户 PeterEgan 与 PeteRegan 被认为是两个不同的作者。但在多数时候用户名并不区别大小写，有时会存在不一致，所以你可能希望 DavidMorgan 和 davidmorgan 能被当成是同一个作者。使用作者区分大小写复选框来控制。

注意，统计只包括了日志对话框中的那段时期。如果日志对话框中只显示一个版本，那么统计就没有什么意义了。

4.9.10. #离线方式

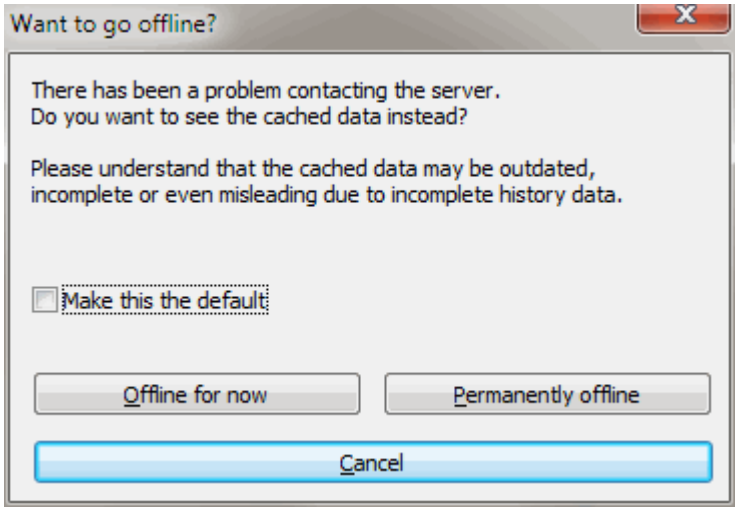


图 4.26. 要离线对话框

如果服务器不可用，并且你已经启用了日志缓存，那么你可以在离线方式下使用日志对话框和版本图。它使用缓存中的数据，使得你可以继续工作，尽管信息可能不是最新的甚至不完整。

这里你有三个选择:

立即离线
: 使用离线方式完成当前操作，但当下次需要日志数据时重新尝试链接版本库。

永久离线
: 保持离线方式，直到特别要求进行版本库检查。参见 第 4.9.11 节 “刷新视图”

取消
: 如果你不想使用失效的数据来继续操作，就取消吧。

选中设为默认值复选框使用你本次的选项作为以后的选择，避免再次显示此对话框。你可以通过 TortoiseSVN → 设置修改(或删除)此默认值。

4.9.11. #刷新视图

如果你想再次检查服务器是否有新的日志消息，可以使用 F5 键来刷新视图。如果使用日志缓存(默认启用)，这将会检查版本库是否有新的消息并且只获取新的消息。如果日志缓存处于离线模式，这将会尝试恢复在线模式。

如果使用日志缓存并且你认为消息内容或者作者可能被更改，你可以使用 Shift-F5 或 Ctrl-F5 来从服务器重新获取显示的消息并更新日志缓存。注意: 此动作只影响当前显示的消息并不会影响此版本库的全部缓存。

4.10. #查看差异

在项目开发中，有一个很常用的要求就是查看更改。可能是你要求查看同一文件的两个版本之间的差异，或者是查看两个独立的文件的差异。TortoiseSVN 自带了一个工具叫 TortoiseMerge 用来查看文本文件的差异。也有一个叫 TortoiseIDiff 的工具来查看图像文件的差异。当然，你可以根据你自己的喜好来选择比较差异的工具。

4.10.1. #文件差异

本地修改
: 如果你想看到你的本地副本有哪些更加，只用在资源管理器中右键菜单下选TortoiseSVN → 比较差异。

与另外一个分支/标签之间的差异
: 如果你想查看主干程序(假如你在分支上开发)有哪些修改或者是某一分支(假如你在主干上开发)有哪些修改，你可以使用右键菜单。在你点击文件的同时按住Shift键，然后选择TortoiseSVN → URL比较。在弹出的对话框中，将特别显示将与你本地版本做比较的版本的URL地址。

  你还可以使用版本库浏览器，选择两个目录树比较，也许是两个标记，或者是分支/标记和最新版本。右键菜单允许你使用比较版本来比较它们。阅读第 4.10.3 节 “比较文件夹”以便获得更多信息。

与历史版本的比较差异
: 如果你想查看某一特定版本与本地副本之间的差异，使用显示日志对话框，选择要比较的版本，然后选择在右键菜单中选与本地副本比较差异

  如果你想查看上一次提交的版本和你的工作副本之间的差异，假设你的工作副本没有被修改，只要用右键单击文件。然后选择 TortoiseSVN → 与前一版本比较。这样将会对上一次提交时间(记录在你的工作副本中)之前的版本和工作基础版本进行差异比较。它将会显示工作副本中此文件变成当前状态的那一次修改。它不会显示比工作副本更新的修改。

两个历史版本的比较
: 如果你要查看任意已提交的两个历史版本之间的差异，在版本日志对话框中选择你要比较的两个版本(一般使用 Ctrl-更改)，然后在右键菜单中选比较版本差异

  如果你在文件夹的版本日志中这样做，就会出现一个比较版本对话框，显示此文件夹的文件修改列表。阅读第 4.10.3 节 “比较文件夹”以便获得更多信息。

提交所有修改
: 如果你要在一个视窗中查看某一版本的所有更改，你可以使用统一显示所有比较 (GNU 片段整理)。它将显示所有修改中的部分内容。它很难显示一个全面清晰的比较，但是会将所有更改都集中显示出来。在版本日志对话框中选择某一版本，然后在右键菜单中选择统一显示所有比较。

文件差异
: 如果你要查看两个不同文件之间的差异，你可以直接在资源管理器中选择这两个文件(一般使用 Ctrl-modifier)，然后右键菜单中选TortoiseSVN → 比较差异。

WC文件/文件夹与URL之间的比较差异
: 如果你要查看你本地副本中的某个文件与任意 Subversion 版本库中的某个文件之间的差异，你可以之间在资源管理器中操作，选中文件然后按下 Shift 键，同时右键单击显示右键菜单。选中 TortoiseSVN → 与 URL 比较。你可以对工作副本文件夹做相同的操作。TortoiseMerge 通过与显示补丁文件相同的方法显示文件夹差异 - 你可以通过一个已修改文件列表分别查看某个文件。

追溯信息之间的比较差异
: 如果你要查看的不仅是比较差异而且包括修改该版本的作者，版本号和日期，你可以在版本日志对话框中综合比较差异和追溯信息。这里有更多详细介绍第 4.23.2 节 “追溯不同点”。

比较文件夹差异
: TortoiseSVN 自带的内置工具不支持查看多级目录之间的差异，但你可以使用支持该功能的外置工具来替代。在 第 4.10.6 节 “其他的比较/合并工具”介绍了一些我们使用过的工具。

如果你已经配置了第三方的比较工具，你可以在选择比较差异命令的时候按下 Shift 键来选择替代工具。关于配置其它比较工具，请参阅 第 4.30.5 节 “外部程序设置”。

4.10.2. #行结束符和空白选项

在项目的生命周期中，有时可能会将行结束符由 CRLF 改为 LF，或者修改一段代码的缩进。不幸的是这样将会使大量的代码行被标记为已修改，尽管代码本身并没有被修改。这里列出的选项将会在比较差异和应用补丁时帮助你应对这些修改。你将会在合并和追溯对话框中看到这些设置，它们同样也出现在 TortoiseMerge 的设置中。

忽略行结束符 排除仅行结束符的差异。

比较空白 将所有缩进和行内空白差异视为增加/删除的行。

忽略空白修改 排除那些完全是针对空白数量或类型的修改，例如，修改缩进或者将 tab 改为空格。在原来没有空白的地方增加空白或删除全部空白仍然会显示为修改。

忽略所有空白 排除仅空白改变的差异。

自然的，任何已经修改内容的行永远都包含在差异中。

4.10.3. #比较文件夹

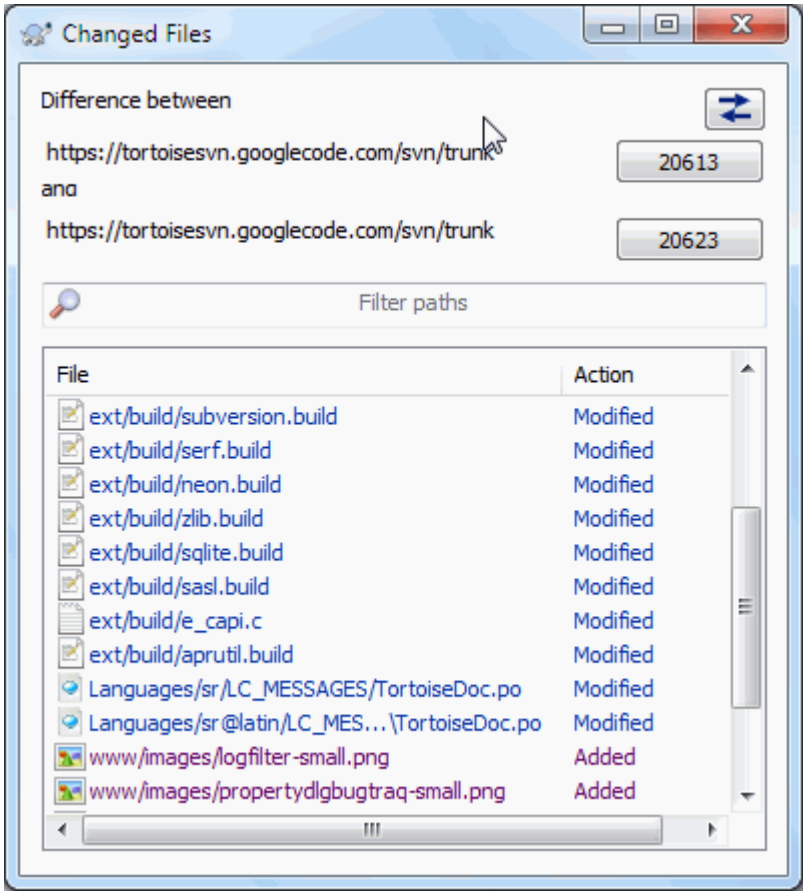


图 4.27. 比较修订版本对话框

当你在版本库浏览器中选择了两个树，或者在日志对话框中选择了一个文件夹的两个版本，就可以使用右键菜单 → 比较版本差异。

这个对话框显示一个所有已经修改的文件列表，允许你使用右键菜单单独的比较或追溯它们。

你可以导出修改树，当你需要将仅包含已修改文件的项目树结构发送给其它人的时候很有用。这个操作仅对选中的文件有效，所以需要选择你感兴趣的文件 - 通常这就意味着全部文件 - 然后使用右键菜单 → 导出选择项...。然后会提示你选择目录保存修改树。

你也可以通过右键菜单 → 保存选择文件列表将修改过的文件列表导出为文本文件。

如果你需要导出文件列表和动作(修改，增加，删除)，你可以使用右键菜单 → 复制到剪贴板。

顶部的按钮允许你改变比较的方向。你可以显示从A到B的修改，或者如果你喜欢，显示从B到A的修改。

有版本数字的按钮可以用来改变版本范围。当你改变范围时，两个版本不同的项目列表会自动更新。

如果列表中的文件非常多，你可以使用查找框来筛选文件名中包含指定文字的文件，从而减少列表中的文件。注意这里使用的是简单文本查找，所以当你需要显示 C 语言源码的列表时，你应该输入.c 而不是 *.c。

4.10.4. #使用 TortoiseIDiff 进行比较的图像

我们有许多有用的比较文本文件的工具，包括我们自带的TortoiseMerge，但是我们也需要查看图像文件的更改。这就是我们设计TortoiseIDiff的原因。

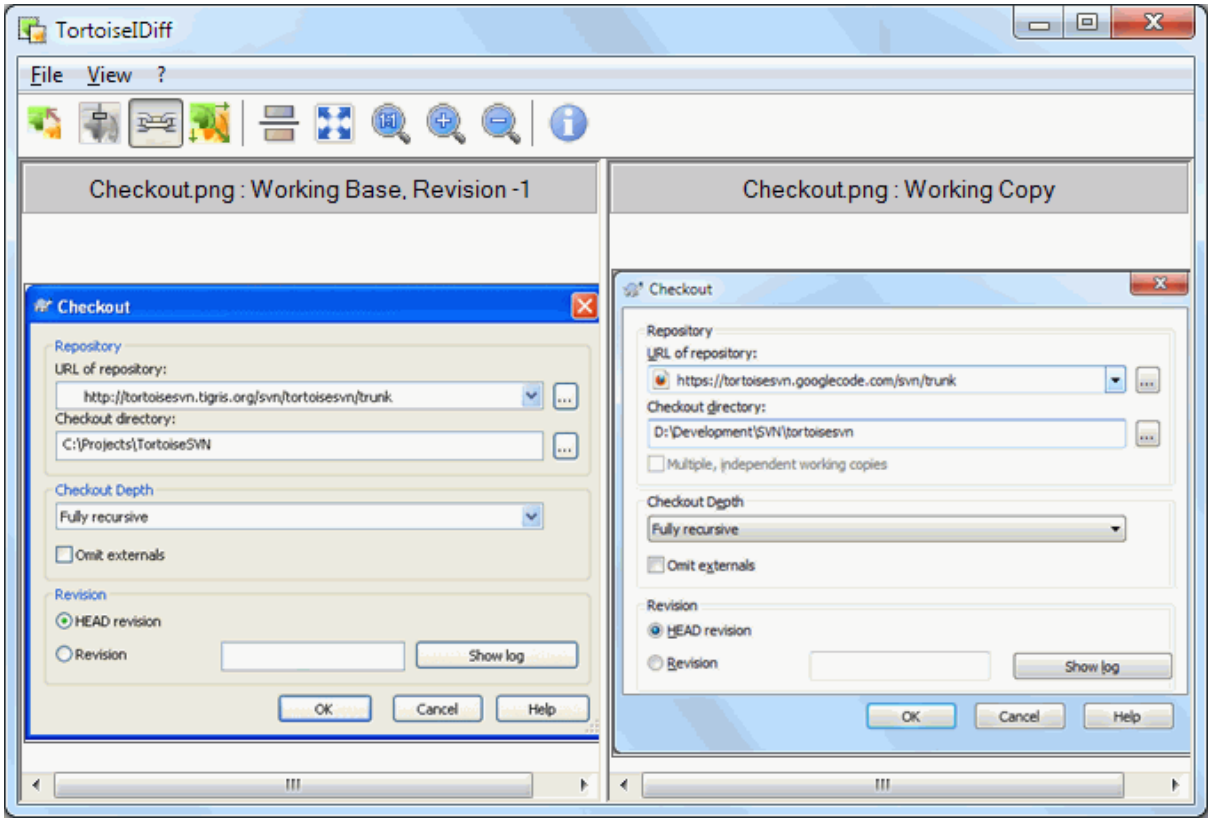


图 4.28. 差异察看器截图

TortoiseSVN → 比较差异 会启动 TortoiseIDiff 显示常用格式的图像差异。一般情况下是左右对称地显示两个图像，但你也可以通过视图菜单或者工具栏切换为上下显示的模式，或者如果你愿意，也可以重叠显示图像。

当然你也可以放大和缩小，或者移动图像。你也可以简单的通过左键拖拽来移动图像。如果你选择链接图像位置，则移动控制(滑动条，鼠标滚轮)对两幅图像同时生效。

在图像信息框中显示了图像的基本信息，比如像素的大小，颜色的深度。如果觉得这个框碍眼可以选择视图 → 图像信息来隐藏它。当你的鼠标指针位于图像标题栏上方时，可以从工具提示中获得同样的信息。

当图像叠加时，图像的相对亮度(alpha 混合)由左边的滑杆控制。你可以点击滑杆的任意地方来设置混合参数，或者通过拖曳滑杆来交互改变。Ctrl+Shift-鼠标滚轮也可以改变混合参数。

滑杆上方的按钮可以切换 0% 和 100% 混合，如果你双击按钮，会每隔两秒钟自动切换，直到再次点击按钮为止。当查看多处小更改时，这个功能很有用。

有时你想要查看不同但不是混合图像。也许你有两个版本的印刷电路板图像文件，想查看哪些线条改变了。如果你关闭 alpha 混合模式，不同之处会以像素颜色值的或非(XOR)结果显示。未修改的区域是纯白色的，修改的部分则是彩色的。

4.10.5. #比较Office文档

When you want to diff non-text documents you normally have to use the software used to create the document as it understands the file format. For the commonly used Microsoft Office and Open Office suites there is indeed some support for viewing differences and TortoiseSVN includes scripts to invoke these with the right settings when you diff files with the well-

known file extensions. You can check which file extensions are supported and add your own by going to TortoiseSVN → Settings and clicking Advanced in the External Programs section.

Office 2010的问题

If you installed the Click-to-Run version of Office 2010 and you try to diff documents you may get an error message from Windows Script Host something like this: "ActiveX component can't create object: word.Application". It seems you have to use the MSI-based version of Office to get the diff functionality.

4.10.6. #其他的比较/合并工具

如果我们提供的这些工具不是你所需要的，可以尝试使用一些其他开源的或者商业的软件。每个人都有不同喜好，下面列表虽不完全，或许有些你也会认可的:

WinMerge
: WinMerge [http://winmerge.sourceforge.net/]WinMerge也是一款很好的能处理目录的开源软件。

Perforce Merge
: Perforce 是一款商业 RCS，但是你也可以免费下载到。可以从Perforce [http://www.perforce.com/perforce/products/merge.html]获得更多信息。

KDiff3
: KDiff3也是一款能处理目录的免费比较工具。你可以从here [http://kdiff3.sf.net/]下载。

SourceGear DiffMerge
: SourceGear Vault is a commercial RCS, but you can download the diff/merge tool for free. Get more information from SourceGear [http://www.sourcegear.com/diffmerge/].

ExamDiff
: ExamDiff Standard是免费软件。它能处理文件但不能处理目录。ExamDiff Pro是共享软件，拥有一系列的功能包括目录比较和编辑的能力。对于以上体验，3.2及以上版本能处理二进制。你可以从PrestoSoft [http://www.prestosoft.com/]下载它们。

Beyond Compare
: 和ExamDiff Pro一样，这也是一款很不错的共享软件，同样也能进行目录比较和二进制处理。下载地址Scooter Software [http://www.scootersoftware.com/]。

Araxis Merge
: Araxis Merge是一款能对文件和文件夹进行比较和合并的商业软件。在合并时它进行三路比较，而且在你修改函数的顺序时可以进行异步链接。可以从这里下载 Araxis [http://www.araxis.com/merge/index.html]。

第 4.30.5 节 "外部程序设置这里可以了解到怎样起用TortoiseSVN来使用这些工具。

4.11. #添加新文件和目录

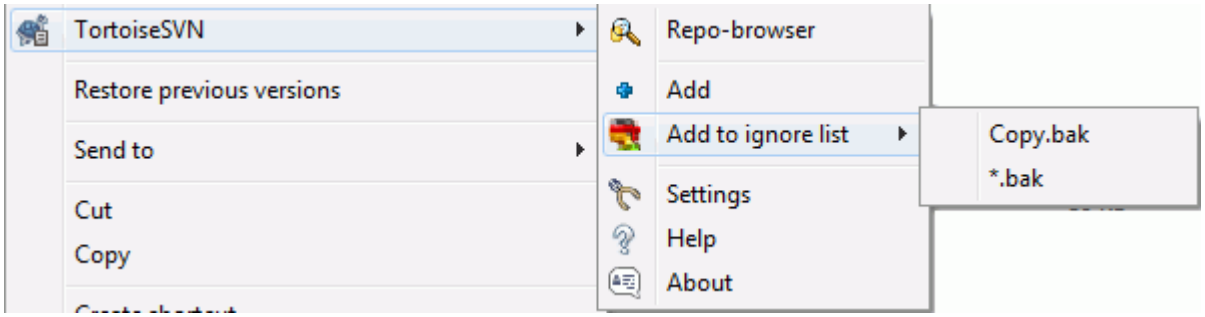


图 4.29. 未受版本控制的文件之资源管理器上下文菜单

如果在你的开发过程中你创建了新的文件或目录，那么你需要把他们加入你的版本控制中。选择那个文件或目录并使用TortoiseSVN → 添加(Add).

当你添加了指定的文件/目录到版本控制系统之后，这个文件上会出现一个added标志，这意味着你得先提交你的工作副本使该文件/目录对其他开发者来说成为有效的。添加一个文件/目录不会not影响版本库

更多

你也可以在已经版本控制的文件夹上使用增加命令。那样的话，增加对话框会显示该版本控制文件夹中所有未版本控制的文件。如果你有许多新文件需要一起增加的话，这是很有帮助的。

你可以使用鼠标拖拽的方式从你的工作副本外部添加进文件。

1. 选择你要添加的文件

2. 用鼠标右键拖拽它们到工作副本的新位置

3. 松开鼠标右键

4. 选择右键菜单 → SVN 增加文件到工作副本。这些文件会被复制到工作副本，加入版本控制。

你可以在工作副本中通过左拖，将文件放到提交对话框中，来增加文件。

如果你误增加了文件或文件夹，你可以在提交前撤销增加，使用TortoiseSVN → 撤销增加.

4.12. #复制/移动/重命名文件和文件夹

经常发生其它项目需要使用某个项目文件的情况，你只想简单的复制它们。你可以使用上述方法复制这些文件，然后增加到版本库，但是这种方法不会给你任何历史信息。并且当你修改了原始文件的问题后，你只能在那些原始文件与新副本在同一个 Subversion 版本库的情况下自动合并修改。

在工作副本中复制文件和目录的最简单的方法是使用右拖菜单。当你从一个工作副本<action>右拖</action>文件或目录到另一个工作副本，甚至是在同一个目录中，当你释放鼠标时，就会出现一个上下文菜单。<placeholder-1/> 现在你可以复制受版本控制的内容到新位置，可能同时有改名操作。

你也可以使用熟悉的剪切-粘贴方式在一个工作副本中或两个工作副本之间复制或移动版本控制的文件。使用标准的 Windows 复制 或 剪切来复制或移动版本控制的条目到剪贴板中。如果剪贴板中包含这样的版本控制的条目，就可以使用 TortoiseSVN → 粘贴(注意: 不是标准的 Windows 粘贴)来复制或移动这些条目到工作副本中的新位置。

你可以使用TortoiseSVN → 分支/标记将工作副本中的文件和文件夹复制到版本库中的另一个位置。参看 第 4.19.1 节 “创建一个分支或标记”获得更多信息。

你可以直接在日志对话框中定位一个文件或文件夹的旧版本并使用右键菜单 → 从版本创建分支/标记将其复制到版本库中的新位置。参看 第 4.9.3 节 “获得更多信息”获得更多信息。

你也可以使用版本库浏览器定位内容，并从版本库直接复制到你的工作副本中，或复制到版本库中的另一个位置。参看 第 4.24 节 “版本库浏览器”获得更多信息。

不能在版本库之间复制

虽然可以在同一个版本库中复制或移动文件和文件夹，你却不能使用 TortoiseSVN 从一个版本库中复制或移动到另一个版本库并保留完整的历史。即使版本库在同一个服务器上也不可以。你能做的就是复制它当前的状态并以新内容的形式添加到第二个版本库中。

如果你不能确定位于同一个服务器上的两个 URL 指向相同的还是不同的版本库，使用版本库浏览器打开其中一个URL，并且找到版本库的根。如果你能在这一个版本库浏览器中看到那两个地址，那么它们就在同一个版本库中。

4.13. #忽略文件和目录

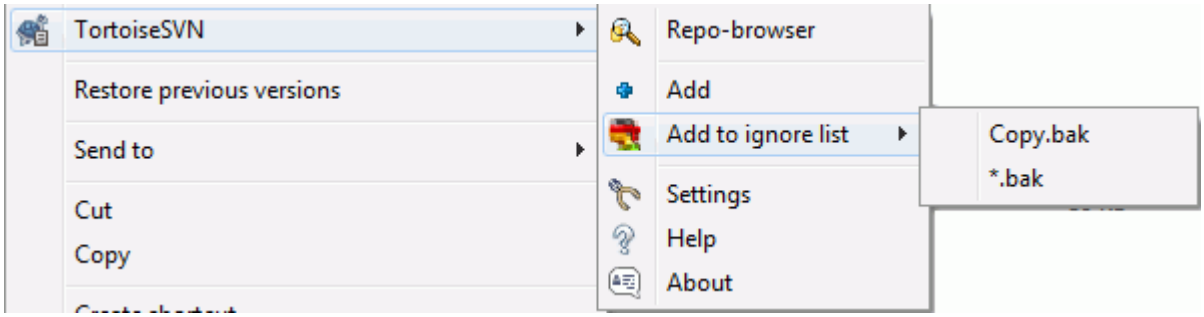


图 4.30. 未受版本控制的文件之资源管理器上下文菜单

在多数项目中你总会有文件和目录不需要进行版本控制。这可能包括一些由编译器生成的文件，*.obj,*.lst，或许是一个用于存放可执行程序的输出文件夹。只要你提交修改，TortoiseSVN 就会在提交对话框的文件列表中显示出未版本控制文件。当然你可以关闭这个显示，不过你可能会忘记添加新的源文件。

最好的避免类似问题的方法是添加参考文件到该项目的忽略列表。这样他们就永远不会出现在提交对话框中，而真正的未版本控制文件则仍然列出。

If you right click on a single unversioned file, and select the command TortoiseSVN → Add to Ignore List from the context menu, a submenu appears allowing you to select just that file, or all files with the same extension. Both submenus also have a (recursively) equivalent. If you select multiple files, there is no submenu and you can only add those specific files/folders.

If you choose the (recursively) version of the ignore context menu, the item will be ignored not just for the selected folder but all subfolders as well. However this requires SVN clients version 1.8 or higher.

如果你想从忽略列表中移除一个或多个条目，右击这些条目，选择TortoiseSVN → 从忽略列表删除。你也可以直接存取目录的svn:ignore属性。它允许你使用文件匹配来指定多个模式，这在下面的章节叙述，阅读第 4.17 节 “项目设置”获得更多关于直接设置属性的信息。请注意每个忽略模式占一行，不支持使用空格分割。

全局忽略列表

另一个忽略文件的方法是添加这些文件到global ignore list .他们最大的不同是全局忽略列表是一个客户端特性。它会作用到 所有的(all)subversion 项目。但只能在pc客户端使用。在全局尽可能更好的使用svn:ignore特性，因为他能够应用到特殊的项目区域，并却他作用于所有检出该项目的人。阅读第 4.30.1 节 “常规设置”获得更多信息。

忽略已版本控制的条目

已版本控制的文件或目录不能够忽略，这是subversion的一个特性。如果你错误的版本控制了一个文件，阅读第 B.8 节 “忽略已经版本控制的文件”介绍怎样“取消版本控制(unversion)”。

4.13.1. #忽略列表中的模式匹配

Subversion 的忽略模式使用了文件匹配，一种原先在Unix系统中使用meta字符作为通配符的技术。下面的字符有着特殊的意思:

*
匹配任何字符串，包括空串(没有字符)

?
匹配任何单字符

[...]
匹配任何单在方括号[]内的单字符，在方括号内，一对字符被“-”分隔，匹配任何词汇表(lexically)上在他们中间的字符。例如[AGm-p]匹配任何单个的A,G,m,n,o或者p。

模式匹配是大小写敏感的，这在 Windows 平台下会出问题。你可以使用成对的字符来强制忽略大小写。例如，忽略不记 *.tmp 的大小写，那么你可以使用像 *.[Tt][Mm][Pp] 这样的模式。

如果你想要一个官方定义的匹配规则。你可以在关于shell命令行语言的IEEE规范Pattern Matching Notation [http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_13]中找到。

全局忽略列表中不要使用路径

不应该在模式中包含路径信息。模式匹配的目标是纯文本的文件名和文件夹名。如果你想忽略所有的 CVS 文件夹，只要向忽略列表中添加 CVS 即可。不再需要像在早先的版本中那样指定 CVS */CVS。如果你想忽略所有在 prog 文件夹下的 tmp 文件夹，但不忽略在 doc 文件夹下的，你应该使用 svn:ignore 属性来替代。使用全局忽略模式不能可靠的完成这一目标。

4.14. #删除、移动和改名

Subversion allows renaming and moving of files and folders. So there are menu entries for delete and rename in the TortoiseSVN submenu.

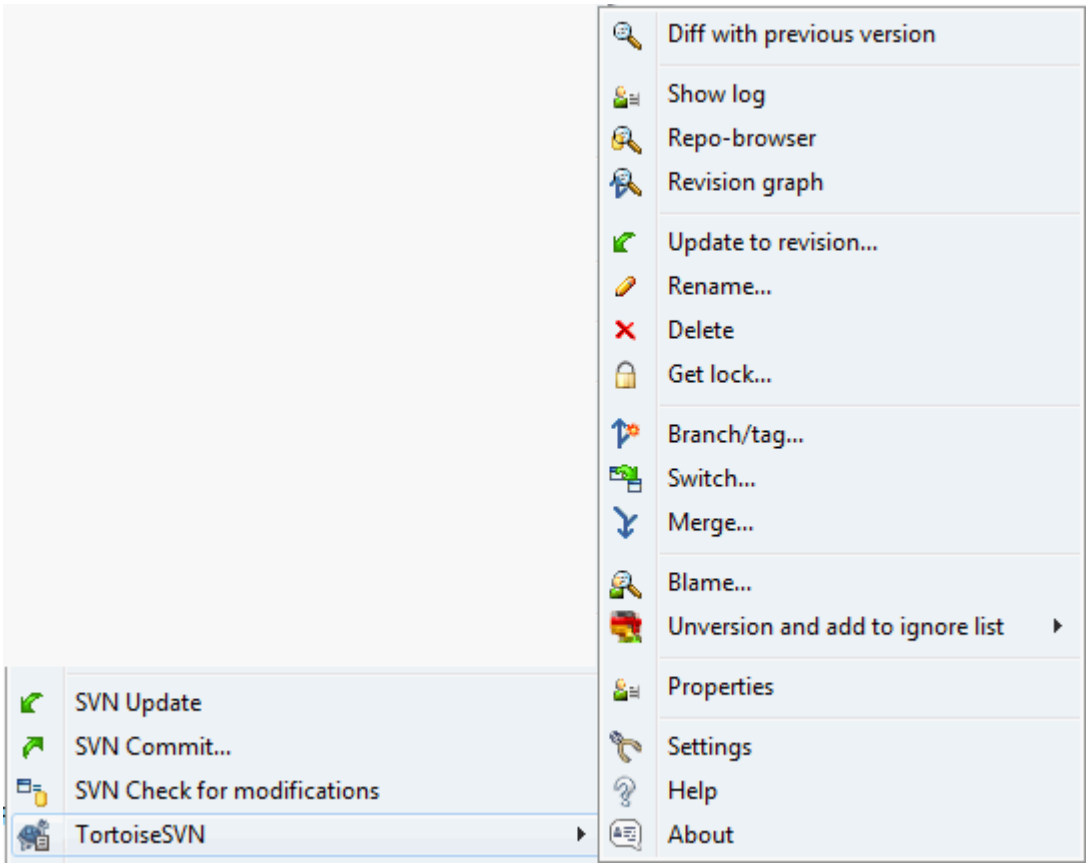


图 4.31. 版本控制文件的菜单浏览

4.14.1. #正在删除文件/文件夹

Use TortoiseSVN → Delete to remove files or folders from Subversion.

When you TortoiseSVN → Delete a file or folder, it is removed from your working copy immediately as well as being marked for deletion in the repository on next commit. The item's parent folder shows a "modified" icon overlay. Up until you commit the change, you can get the file back using TortoiseSVN → Revert on the parent folder.

如果你想从版本库删除项目，但是在本地作为非版本控制的文件/文件夹保留，可以使用 扩展右键菜单 → 删除(保留本地副本)。为了看到扩展右键菜单，当你在文件管理器列表窗格(右窗格)中的项目上点击右键时，必须同时按下 Shift 键。

If an item is deleted via the explorer instead of using the TortoiseSVN context menu, the commit dialog shows those items as missing and lets you remove them from version control too before the commit. However, if you update your working copy, Subversion will spot the missing item and replace it with the latest version from the repository. If you need to delete a version-controlled file, always use TortoiseSVN → Delete so that Subversion doesn't have to guess what you really want to do.

找回已删除的文件或目录

如果你删除了一个文件或目录并已经提交该删除操作到版本库，那么 一个常规的TortoiseSVN → 复原已不能再将其找回。但是该文件或目录并没有完全丢失。如果你知道该被删除文件或目录的版本(如果不能，使用日志对话框来查找出来)，打开数据仓库的浏览器，并选择那个版本。然后选择你删除的文件或目录，右键并选择Context Menu → 复制到...作为目标执行复制操作，然后选择你的工作副本的路径。

4.14.2. #移动文件和文件夹

如果你仅想重命名文件或文件夹，使用 右键菜单 → 改名... 为此条目输入新的名称就可以了。

如果你想在工作副本中移动文件，比如移动到一个不同的子文件夹下，那么使用鼠标右键拖拽:

1. 选择你要移动的文件或目录
2. 用鼠标右键拖拽它们到工作副本的新位置
3. 松开鼠标右键
4. 在弹出菜单选择右键菜单 → SVN 移动版本控制的条目到当前位置。

提交父目录

既然重命名和移动都是像添加之后跟随着删除一样被执行，你必需提交该重命名/移动文件的父文件夹，所以重命名/移动的删除部分将出现在提交对话框中。如果你不提交重命名/移动的已删除部分，他将保留在仓库中并且你的同组人更新工作副本时，该文件也不会被删除。例如，他们将有两个一老一新的副本。

你必须在重命名文件夹后立刻进行提交，在提交前不要更改文件夹下的任何文件，不然你的工作副本就会真的混淆。

另外一种复制或移动文件的方法是通过Windows 的复制/移动命令. 首先选择你需要复制的文件, 在资源管理器中右键点击并选择右键菜单 → 复制. 然后进入目标文件夹，右键点击并选择TortoiseSVN → 粘贴. 对于移动文件，选择 右键菜单 → 剪切而不是右键菜单 → 复制.

你也可以使用版本库浏览器在版本库中移动条目。阅读 第 4.24 节 “版本库浏览器”以获得更多信息。

不要使用 SVN 移动外部连接

你不应该用 TortoiseSVN 的移动或改名命令作用在用 svn:externals 创建的目录上。因为这个动作可能会导致外部元素(item)从它的父版本库中删除，这可能会使其它人烦恼。如果你需要移动外部目录，你应该使用普通的外壳移动，然后调整源和目的之父目录的 svn:externals 属性。

4.14.3. #处理文件名称大小写冲突

万一在你的版本库中有两个名字相同但大小拼写不同(例如: TEST.TXT 和 test.txt)的文件，你是不能在 Windows 客户端更新或者检出该包含该文件的目录的。当 Subversion 支持大小写敏感的文件名时，Windows 不支持。

它偶尔在两个人在独立的工作副本提交时发生，文件名称相同，只有大小写不同。它也会在具有大小写敏感的文件系统的系统中提交文件时发生，例如 Linux。

如果是那样的话，你得决定在这个版本库里的哪一个文件是你想保留的，哪一个是要删除(或重命名)的

防止两个文件名字相同

这有一个有用的服务器端脚本在http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/将会防止检入拼写(大小写)冲突文件。

4.14.4. #修复文件改名

有时候你的IDE会因为执行反射操作，改名文件，当然它不能告诉Subversion。如果你尝试提交修改，Subversion会发现丢失了老文件，新增了未版本控制的新文件。你可以简单的增加新文件，但是你将丢失历史记录，因为Subversion不知道这些文件的关系。

更好的方法是通知Subversion这实际上是改名，你可以在提交和检查修改对话框中做此操作。简单选择老文件(丢失的)和新文件(未版本控制的)，使用右键菜单 → 修复移动设置这两个文件是改名关系。

4.14.5. #删除未版本控制的文件

通常你可以在Subversion中设置自己的忽略列表，例如忽略所有产生的文件。但是你如何清理这些忽略的项目，从而产生一个干净的构建呢？通常你在makefile中清理，但是如果你在调试makefile，或者修改构建系统，那么有一个清理方法是极为有用的。

TortoiseSVN 提供了使用扩展上下文菜单 → 删除未版本控制的项目...来清理工作副本。你可以在目录上右键操作时，保持 Shift按下，就可以看到这个上下文菜单。它会出现一个对话框，列出工作副本中的所有未版本控制的文件。你可以选择或取消删除的项目。

当删除这些项目时，使用了垃圾箱。所以如果你犯了错误，删除了应该版本控制的文件，你仍旧可以恢复。

4.15. #撤消更改

如果你想要撤消一个文件自上次更新后的所有的变更，你需要选择该文件， 右键点击弹出右键菜单，然后选择TortoiseSVN → SVN 还原命令，将会弹出一个对话框显示你已经变更并能恢复的文件。选择那些你想要恢复的然后按确定。

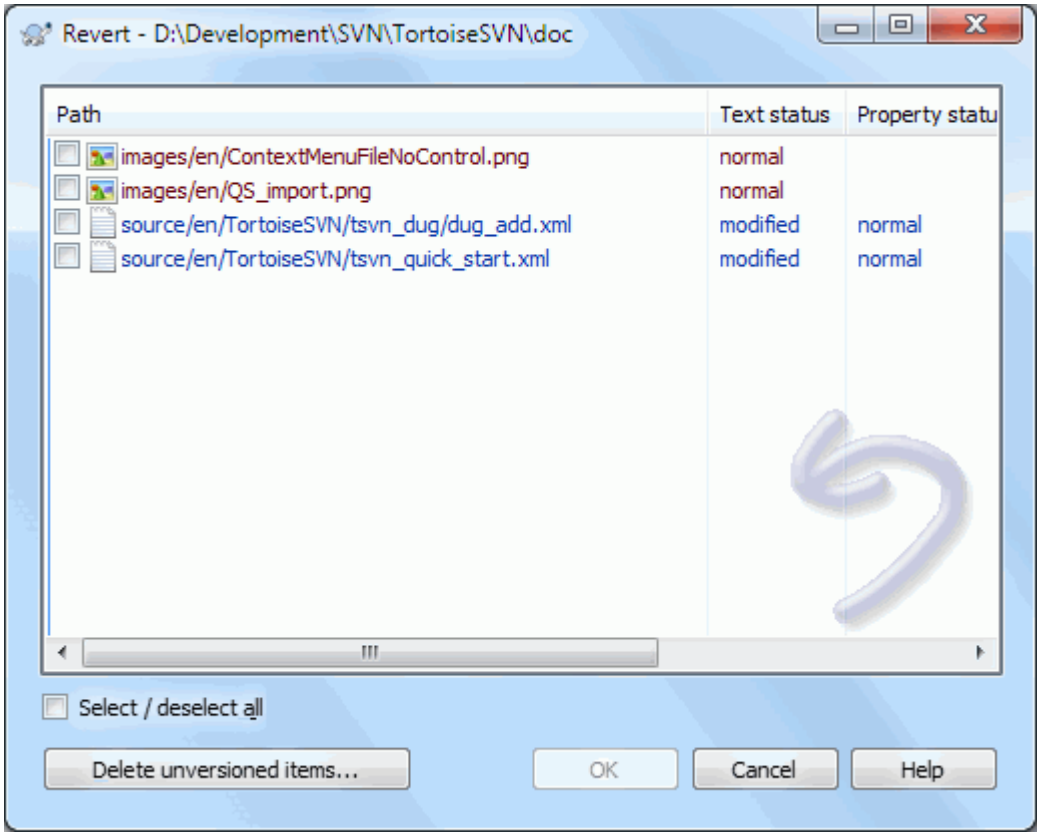


图 4.32. 恢复对话框

如果你想撤销删除或者更名操作，因为被删除的条目已经不存在了，你需要右键点击上一层文件夹来进行还原操作。

如果你想撤销增加条目，在右键菜单中选择 TortoiseSVN → 撤销增加。其实这就是还原操作，只是起了一个更显而易见的名字而已。

在这一对话框中，纵列和在 检查修改对话框中的纵列同样是可以定制的。更多细节请阅读第 4.7.4 节 “本地与远程状态”

还原命令有时被用来清理工作副本，因此这里有一个额外的按钮，通过这个按钮你可以删除未版本控制的条目。当你点击这个按钮，会出现另一个对话框列出所有的未版本控制的条目，你可以选择需要删除的内容。

取消已经提交的改变

Revert仅能撤消你本地的变更。他不能撤消已经提交的的变更。如果你想撤消所有的包括已经提交到一个特定版本的变更，请阅读 第 4.9 节 “版本日志对话框”获得更多信息。

还原较慢

当你进行还原修改时，你会发现操作所花的时间比你预期的要长。这是因为修改过的文件被扔到回收站，所以当误还原时你可以找回更改后的文件。不管怎样，如果你的回收站已经满了，Windows 会花一些时间来找个地方放置文件。解决方案很简单：或者清空回收站，或者在 TortoiseSVN 的设置中取消选中在恢复的时候使用垃圾箱。

4.16. #清理

也许由于服务器问题，一个Subversion指令不能成功地完成，你的工作副本因此被滞留在一个不一致的状态。 那样的话，你需要在该目录上使用TortoiseSVN → 清理命令。在工作副本的根目录使用它是一个好主意。

In the cleanup dialog, there are also other useful options to get the working copy into a clean state.

Clean up working copy status
: As stated above, this option tries to get an inconsistent working copy into a workable and usable state. This doesn't affect any data you have but only the internal states of the working copy database. This is the actual Cleanup command you know from older TortoiseSVN clients or other SVN clients.

Refresh shell overlays
: Sometimes the shell overlays, especially on the tree view on the left side of the explorer don't show the current status, or the status cache failed to recognize changes. In this situation, you can use this command to force a refresh.

Include externals
: If this is checked, then all actions are done for all files and folders included with the svn:externals property as well.

Delete unversioned files and folders, Delete ignored files and folders
: This is a fast and easy way to remove all generated files in your working copy. All files and folders that are not versioned are moved to the trash bin.

  Note: you can also do the same from the TortoiseSVN → Revert dialog. There you also get a list of all the unversioned files and folders to select for removal.

Revert all changes recursively
: This command reverts all your local modifications which are not committed yet.

  Note: it's better to use the TortoiseSVN → Revert command instead, because there you can first see and select the files which you want to revert.

4.17. #项目设置

4.17.1. #Subversion 属性

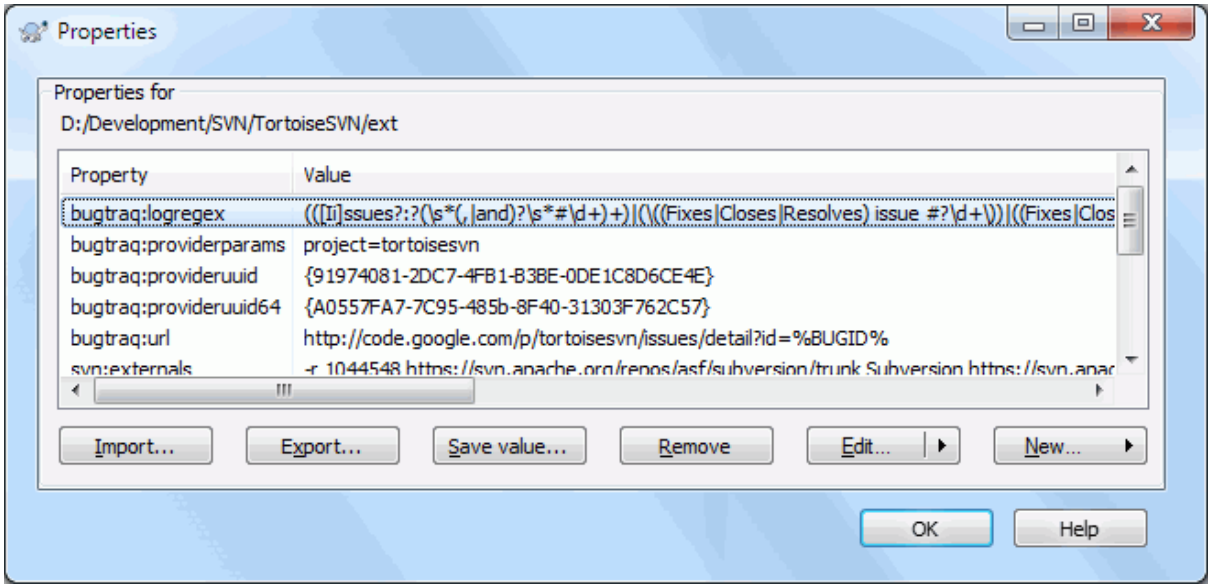


图 4.33. Subversion 属性页

你可以通过右键菜单 → 属性从 Windows 属性对话框来读写 Subversion 属性，这样也能得到 TortoiseSVN 的状态列表。也可以从TortoiseSVN → 属性来读写 Subversion 属性。

你可以添加你自己定义的属性，或者一些在 Subversion 中有特殊含义的属性。这些属性以 svn: 开头。svn:externals 就是这一类型的属性；关于如何引用外部条目，请参阅第 4.18 节 “外部条目”

4.17.1.1.#svn:keywords

Subversion 支持类似 CVS 的关键字扩展，用来在文件中嵌入文件名称和版本信息。当前支持的关键字有：

$Date$
: 已知最后提交的日期。它基于你更新工作副本时获得的信息。它不检查版本库查找最新的修改。

$Revision$
: 已知最后提交的版本。

$Author$
: 已知最后提交的作者。

$HeadURL$
: 此文件在版本库中的 URL。

Id
: 上述四个关键字的压缩组合。

To find out how to use these keywords, look at the svn:keywords section [http://svnbook.red-bean.com/en/1.8/svn.advanced.props.special.keywords.html] in the Subversion book, which gives a full description of these keywords and how to enable and use them.

For more information about properties in Subversion see the Special Properties [http://svnbook.red-bean.com/en/1.8/svn.advanced.props.html].

4.17.1.2. #增加和编辑属性

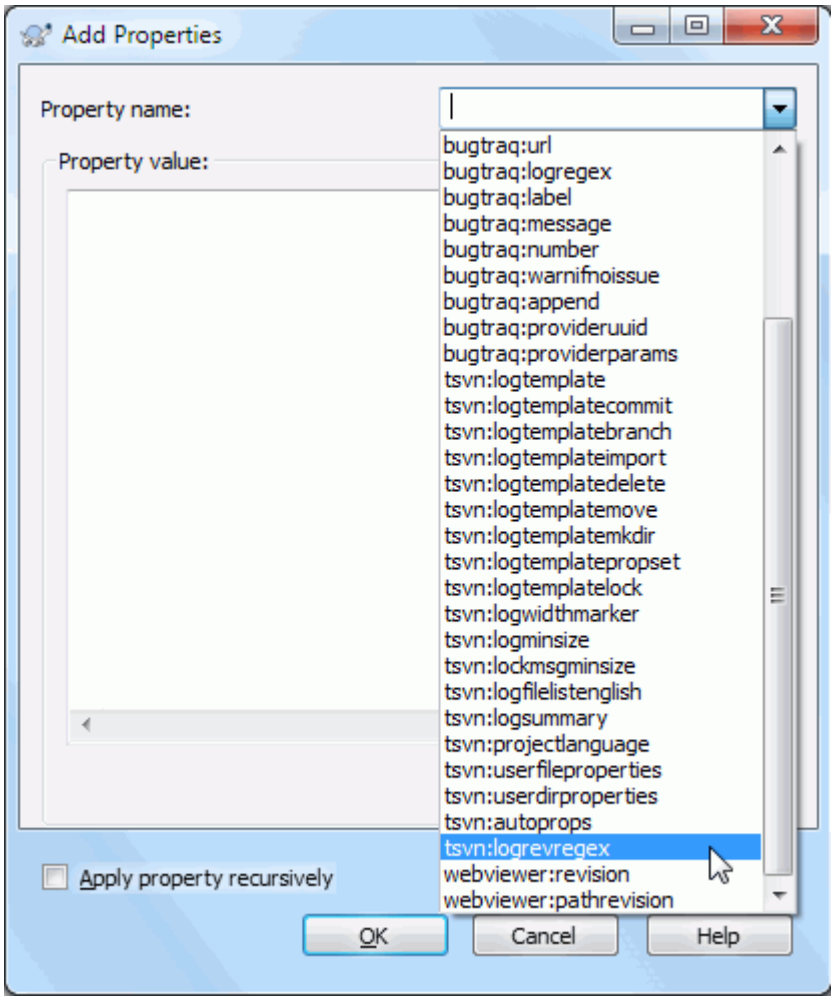


图 4.34. 增加属性

要添加一个新的属性，先点击新建... 。从菜单中选择需要的属性名称，然后在特定的属性对话框内填写所需的信息。这些特定的属性对话框中详细描述第 4.17.3 节 “属性编辑器”。

想添加没有自己的对话框的属性，从新建... 菜单中选择高级 。然后从组合框中选择已经存在的属性或者输入自定义的属性名称。

如果你想一次性设置许多文件的属性，在资源管理器中选择文件/文件夹，然后选择右键菜单 → 属性。

如果你想设置当前文件夹内的全部文件和文件夹，选中递归检查框。

如果你想编辑一个已有属性，在已有属性列表中选择它，然后单击编辑...即可。

如果你想删除已有属性，在已有属性列表中选择它，然后单击删除即可。

属性svn:externals可以用来下载位于同一版本库或不同版本库的其它工程。阅读第 4.18 节 “外部条目”以获得更多信息。

编辑最新修订版本的属性

因为属性是受版本控制的，因此您不能编辑以前修订版本的属性。如果您从日志对话框或库浏览器中的无头部修订版本来查看属性，您将会看到一个属性和值的列表，但没有编辑控件。

4.17.1.3. #导出和导入属性

通常，你会发现你要设置同一组属性很多遍，例如 bugtraq:logregex。要想简单的完成从一个项目复制属性到另一个项目，你可以使用导出/导入特性。

在已经设置了属性的文件或文件上使用TortoiseSVN → 属性，选择你想要导出的属性并单击导出...。然后会提示你输入文件名，属性名和属性值会保存在此文件中。

在你想要应用这些属性的文件夹上使用TortoiseSVN → 属性并单击导入...。然后会提示你选择从哪个文件导入，找到之前你保存导出文件的地方并选中它。属性会添加到文件夹，不递归。

如果你想递归的添加属性到目录树，按照上述的步骤，然后在属性对话框依次选中每一个需要递归的属性，单击编辑...，选中递归应用该属性并单击确定。

导入文件是二进制格式的，并且只被 TortoiseSVN 所使用。它唯一的用途是在导入和导出命令之间传递属性，所以不要编辑这些文件。

4.17.1.4. #二进制属性

TortoiseSVN可以处理文件的二进制属性。使用保存...到文件读取二进制属性值。使用十六进制编辑器或其它适当的工具创建文件，然后用从文件加载...设置二进制值为此文件的内容。

尽管二进制文件不经常使用，它们在一些程序中是有用的。举例来说，如果你存储了巨大的图形文件，或者用程序加载的文件巨大，你可能想将缩略图作为属性存储，于是你可以快速的预览。

4.17.1.5. #自动属性设置

你可以配置 Subversion 和 TortoiseSVN 在文件和文件夹添加到版本库时自动设置属性。这里有两种实现方法。

You can edit the Subversion configuration file to enable this feature on your client. The General page of TortoiseSVN's settings dialog has an edit button to take you there directly. The config file is a simple text file which controls some of Subversion's workings. You need to change two things: firstly in the section headed miscellany uncomment the line enable-auto-props = yes. Secondly you need to edit the section below to define which properties you want added to which file types. This method is a standard Subversion feature and works with any Subversion client. However it has to be defined on each client individually - there is no way to propagate these settings from the repository.

另一个办法是在文件夹上设置 tsvn:autoprops 属性，将会在下一节详细描述。这个方法仅对 TortoiseSVN 客户端有效，但是它可以在更新时应用于所有的工作副本。

无论您选择哪种方法，您应该注意 Auto-props 仅应用于本次被添加到工作副本的那些文件。Auto-props 将决不会更改已受版本控制的文件的属性。

如果你想确保所有的新文件设置了正确的属性，你应该设置版本库的 pre-commit 钩子脚本来拒绝那些没有设置必要属性的提交。

提交属性

Subversion 属性是受版本控制的。在你改变或增加属性后必须提交。

属性冲突

如果因为其他用户已经提交了同样的属性，提交时出现冲突，Subversion 会产生一个 .prej 文件。在你解决冲突后，请删除此文件。

4.17.2. #TortoiseSVN 项目属性

TortoiseSVN 有自己专用的几个属性，它们都有tsvn:前缀。

- tsvn:logminsize设置提交日志的最小长度。如果你输入的日志短于预设值，提交会被禁止。这个属性对于提醒你为每次提交提供一个适当的描述信息非常有用。如果不设置这个属性，或者设置为0，那么就允许空提交信息。

  tsvn:lockmsgminsize设置锁定日志的最小长度。如果你输入的日志短于预设值，加锁会被禁止。这个属性对于提醒你为每次加锁提供一个适当的描述信息非常有用。如果不设置这个属性，或者设置为0，那么就允许空加锁信息。

- tsvn:logwidthmarker用在要求日志信息被格式化为在最大宽度(典型是80字符)处换行非常有用。设置此属性为大于0的值会在日志消息对话框中做两件事： 放置一个标记指示最大宽度，和禁止自动换行，于是你可以看到输入的信息是否太长。注意: 这个特性仅在你选择的消息使用固定宽度字体时才能正确工作。

- tsvn:logtemplate在需要定义日志消息格式化规则的工程中使用。在你开始提交时，这个属性的多行消息会被插入日志消息编辑框。你可以编辑它以便包含需要的信息。注意: 如果你使用了tsvn:logminsize属性，请确认这个长度大于模版的长度，不然就会失去其保护作用。

  也会有一些可替代 tsvn:logtemplate 的操作特定模板以供使用。这些特定模板被设置后即可使用，但如果没有设置任何特定模板，则将会使用 tsvn:logtemplate。

  这些操作特定模板是:

  - tsvn:logtemplatecommit 用于所有来自工作副本的提交。
  - tsvn:logtemplatebranch 用于在您创建分支/标记或直接在版本库浏览器中复制文件或文件夹时。
  - tsvn:logtemplateimport 用于导入。
  - tsvn:logtemplatedelete 用于直接在版本库浏览中删除项目时。
  - tsvn:logtemplatemove 用于在版本库浏览器中重命名或移动项目时。
  - tsvn:logtemplatemkdir 用于在版本库浏览中创建目录时。
  - tsvn:logtemplatepropset 用于在版本库浏览器中修改属性时。
  - tsvn:logtemplatelock 用于获取一个锁定。

- Subversion allows you to set “autoprops” which will be applied to newly added or imported files, based on the file extension. This depends on every client having set appropriate autoprops in their Subversion configuration file. tsvn:autoprops can be set on folders and these will be merged with the user's local autoprops when importing or adding files. The format is the same as for Subversion autoprops, e.g. *.sh = svn:eol-style=native;svn:executable sets two properties on files with the .sh extension.

  如果本地 autoprops 与 tsvn:autoprops 冲突，项目设置优先(因为它们是针对此项目的)。

- 在提交对话框，你可以粘贴修改的文件列表，包含每个文件的状态(增加，修改等)。tsvn:logfilelistenglish定义了文件状态用英文插入，还是用本机语言插入。如果此属性没有设置，默认值是 true。

- TortoiseSVN可以使用OpenOffice和Mozilla使用的拼写检查模块。如果你安装了这些模块，那么这个属性将检测使用哪个拼写检查模块。也就是，你的项目的日志信息用的语言。tsvn:projectlanguage设置拼写检查引擎在你输入日志消息的时候应该使用什么语言模块。你可以在这个页面找到你的语言的取值： MSDN: 语言标示符 [http://msdn2.microsoft.com/en-us/library/ms776260.aspx]。

  你可以用十进制输入取值，如果用0x前缀的话，也可以用十六进制。例如英语(美国英语)可以输入0x0409或者1033。

- 属性 tsvn:logsummary 用于摘录日志的一部分，在日志对话框中显示为日志摘要。

属性 tsvn:logsummary 的值必须设置为一行包含一个正则组的正则字符串。匹配于这个正则组的任何内容被当作摘要。

例如: \[SUMMARY\]:\s+(.*) 将会抓取 “[SUMMARY]” 后面的所有内容并将其用作摘要。

- 属性 tsvn:logrevregex 定义了一个正则表达式，它匹配日志消息中的对版本号的引文。它的用处是，在日志对话框中，将这样的版本号变成链接，当点击链接的时候滚动到这一版本(如果此版本已经显示在日志对话框中被显示出来，或者它在日志缓存中)或者打开一个新的日志对话框显示那个版本。

  正则表达式必须匹配整个引文，而不仅仅是版本号。版本号可以自动的从匹配的引文字符串中提取出来。

  如果这个属性没有设置，一个默认的正则表达式被用来链接版本引文。

- There are several properties available to configure client-side hook scripts. Each property is for one specific hook script type.

  The available properties/hook-scripts are

  - tsvn:startcommithook
  - tsvn:precommithook
  - tsvn:postcommithook
  - tsvn:startupdatehook
  - tsvn:preupdatehook
  - tsvn:postupdatehook

  The parameters are the same as if you would configure the hook scripts in the settings dialog. See 第 4.30.8 节 “客户端钩子脚本”for the details.

  Since not every user has his or her working copy checked out at the same location with the same name, you can configure a script/tool to execute that resides in your working copy by specifying the URL in the repository instead, using %REPOROOT% as the part of the URL to the repository root. For example, if your hook script is in your working copy under contrib/hook-scripts/client-side/checkyear.js, you would specify the path to the script as %REPOROOT%/trunk/contrib/hook-scripts/client-side/checkyear.js. This way even if you move your repository to another server you do not have to adjust the hook script properties.

  Instead of %REPOROOT% you can also specify %REPOROOT+%. The + is used to insert any number of folder paths necessary to find the script. This is useful if you want to specify your script so that if you create a branch the script is still found even though the url of the working copy is now different. Using the example above, you would specify the path to the script as %REPOROOT+%/contrib/hook-scripts/client-side/checkyear.js.

  The following screenshot shows how the script to check for current copyright years in source file headers is configured for TortoiseSVN.

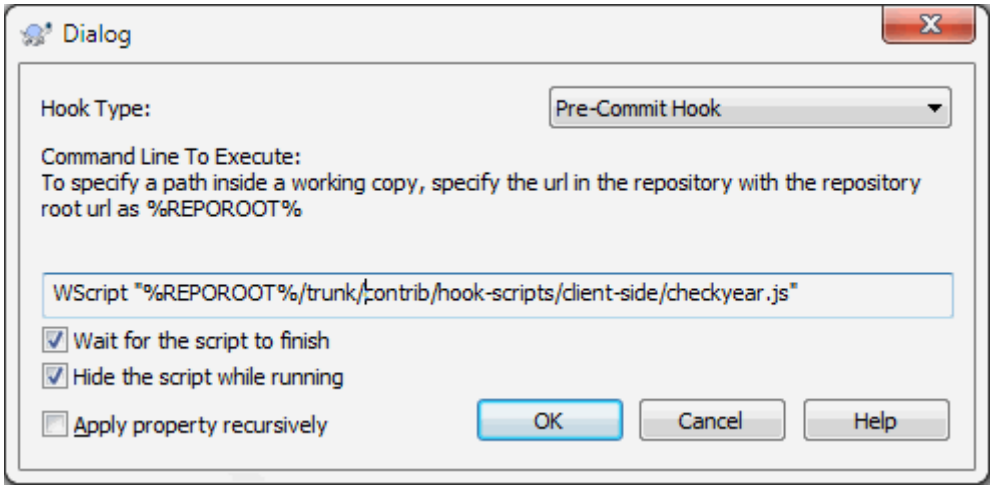


图 4.35. Property dialog for hook scripts

- 当你想增加新属性时，你可以从组合框的下拉列表选取，也可以输入你喜欢的任何属性名称。如果你的项目使用了自定义属性，并且想让这些属性出现在组合框的下拉列表中(避免输入时拼写错误)，你可以使用tsvn:userfileproperties和tsvn:userdirproperties创建自定义属性列表。对目录应用这些属性，当你编辑其任何子项属性时，你自定义的属性将会在预定义属性名称列表中出现。

  You can also specify whether a custom dialog is used to add/edit your property. TortoiseSVN offers four different dialog, depending on the type of your property.

  bool
  : If your property can only have two states, e.g., true and false, then you can configure your property as a bool type.

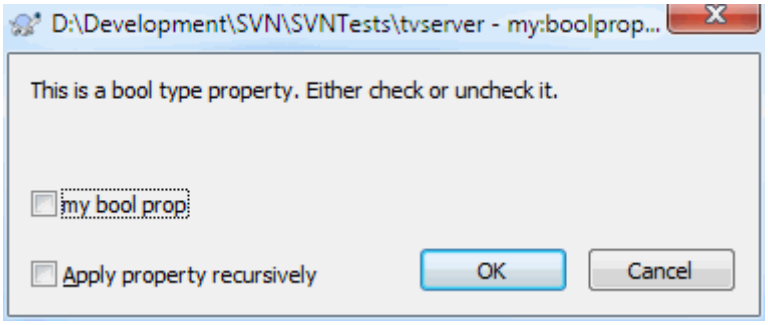


图 4.36. Property dialog boolean user types

  Specify your property like this:

  propertyname=bool;labeltext(YESVALUE;NOVALUE;Checkboxtext)

  the labeltext is the text shown in the dialog above the checkbox where you can explain the purpose and use of the property. The other parameters should be self explanatory.

  state
  : If your property represents one of many possible states, e.g., yes, no, maybe, then you can configure your property as a state

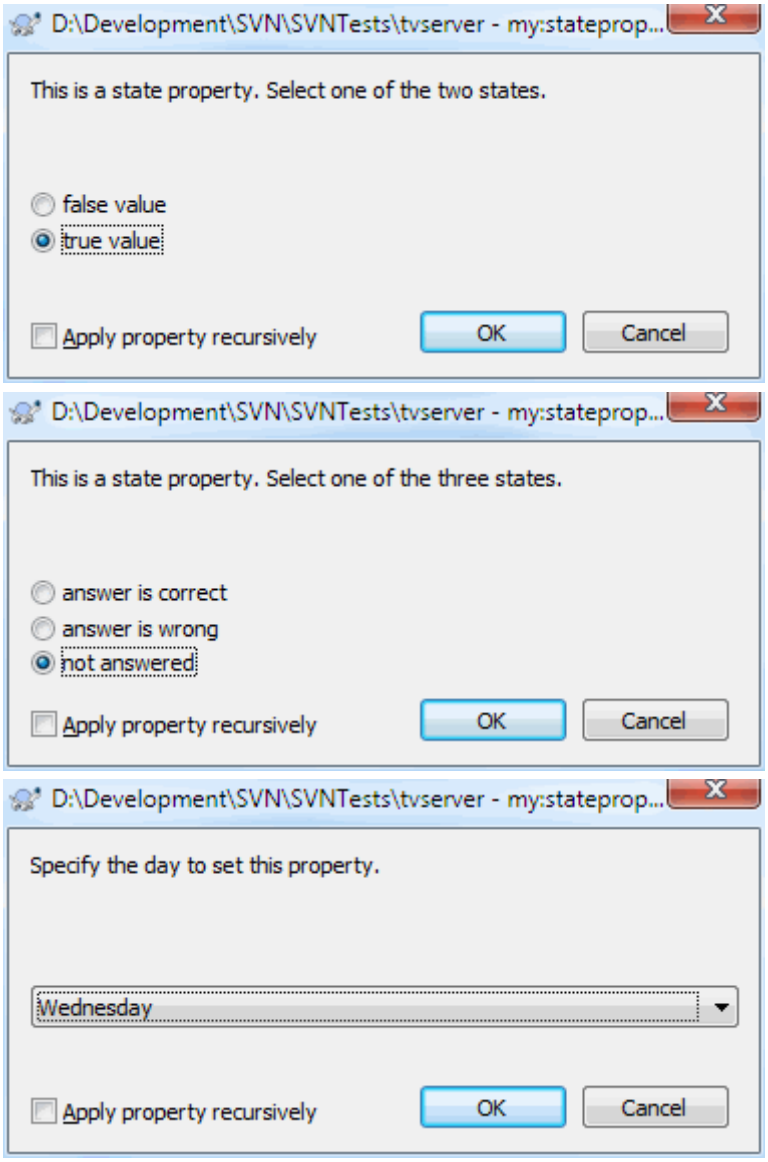


图 4.37. Property dialog state user types

property like this:

propertyname=state;labeltext(DEFVAL;VAL1;TEXT1;VAL2;TEXT2;VAL3;TEXT3;...)

The parameters are the same as for the bool property, with DEFVAL being the default value to be used if the property isn't set yet or has a value that's not configured.

For up to three different values, the dialog shows up to three radio buttons. If there are more values configured, it uses a combo box from where the user can select the required state.

singleline

For properties that consist of one line of text, use the singleline property type:

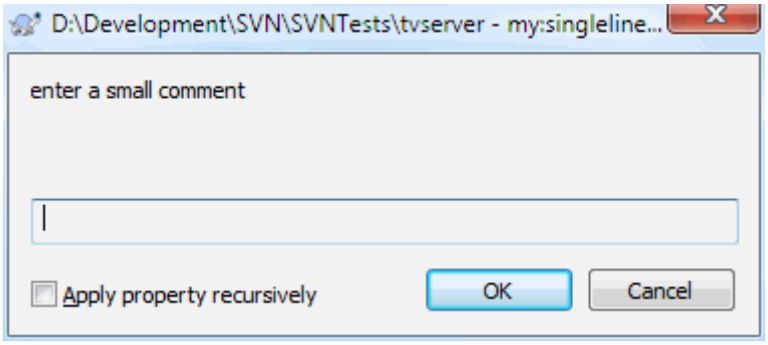


图 4.38. Property dialog single-line user types

propertyname=singleline;labeltext(regex)

the regex specifies a regular expression which is used to validate (match) the text the user entered. If the text does not match the regex, then the user is shown an error and the property isn't set.

multiline
For properties that consist of multiple lines of text, use the multiline property type:

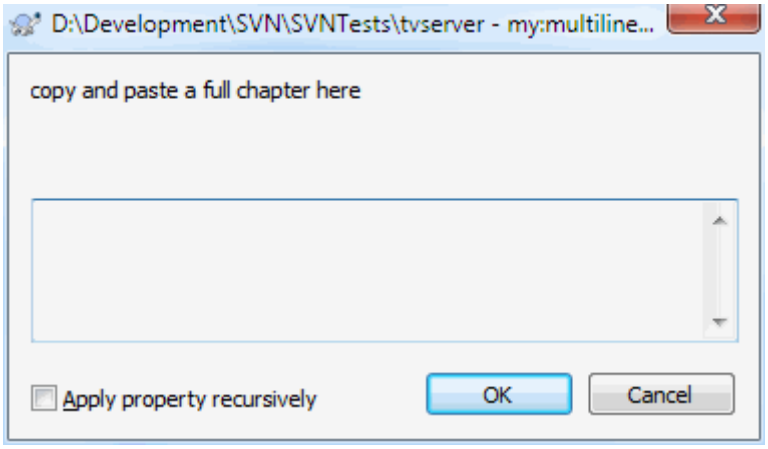


图 4.39. Property dialog multi-line user types

propertyname=multiline;labeltext(regex)

the regex specifies a regular expression which is used to validate (match) the text the user entered. Don't forget to include the newline (\n) character in the regex!
The screenshots above were made with the following tsvn:userdirproperties:

```
my:boolprop=bool;This is a bool type property. Either check or uncheck it.(true;false;my bool prop)
my:stateprop1=state;This is a state property. Select one of the two states.(true;true;true value;false;
my:stateprop2=state;This is a state property. Select one of the three states.(maybe;true;answer is corr
my:stateprop3=state;Specify the day to set this property.(1;1;Monday;2;Tuesday;3;Wednesday;4;Thursday;5
my:singlelineprop=singleline;enter a small comment(.*)
my:multilineprop=multiline;copy and paste a full chapter here(.*)
```

TortoiseSVN 可以与一些 BUG 跟踪工具集成。它使用 bugtraq: 开始的项目属性。阅读第 4.28 节 “与 BUG 跟踪系统/问题跟踪集成”以获得更多信息。

它也与一些基于 WEB 的版本库浏览器集成，使用 webviewer: 开始的项目属性。阅读第 4.29 节 “与基于 WEB 的版本库浏览器集成”以获得更多信息。

设置文件夹的项目属性

为了让系统更好的工作，某些特别的项目属性必须保存在 文件夹中。 当你使用了 TortoiseSVN 命令调用这些属性，属性值就从选中的文件夹读取出来。如果在文件夹中找不到属性值，TortoiseSVN 会沿着文件夹树一直向上搜索，直到出现未版本化的 文件夹或者到达根目录（例如：C:\）。 如果您可以肯定每个用户都是只从类似 trunk/的位置签出代码，而且没什么子目录， 那么只需要设置属性在 trunk/上就够了。 假如你并不确定，那就需要在每个子目录都设置属性。 如果你给每个目录设置了相同的属性但是不同的项目角色又要使用不同的值，那么从不同的目录签出的时候，会得到不同的结果。

对于tsvn:属性，例如 tsvn:，bugtraq: 和 webviewer:，你只能对于所有子文件夹使用递归复选框设置属性，不用将这些属性设置在文件上。

当你使用TortoiseSVN在工作副本中新建一个子目录， 上层目录的所有项目属性都会自动的被继承。.

限制使用代码库浏览器

远程获取项目属性是一项很耗时的操作，因此上面描述的某些功能查看工作副本时是没有问题的，但在浏览库的时候不一定会显示。

- 当你使用版本库浏览器添加一个属性时，在预定义列表中只会提供标准的 svn: 属性。任何其他的属性名称都必须手动输入。
- 属性不能设置或删除递归使用回购浏览器。
- 当使用版本库浏览器添加一个新的子文件夹时，项目属性将 不会 被自动传递到子文件夹。
- 当使用版本库浏览器添加文件时，tsvn:autoprops 将 不会 给这些添加的文件设定属性。

尽管 TortoiseSVN 的项目属性很有用，但它们只适用于 TortoiseSVN，而且一些只是对于新版本的 TortoiseSVN 才生效。如果你的项目中的人使用各种 Subversion 客户端，或者使用低版本的 TortoiseSVN，你可能要使用版本库钩子强制执行项目策略。项目属性只能帮助你实现策略，不能强制执行。

4.17.3. #属性编辑器

有些属性必须使用特定的值或以某种特定的方式进行格式化，以便能被自动使用。为了帮助正确地进行格式化，TortoiseSVN 为某些显示可能值或将属性分解给它的单独组件的特殊属性提供了编辑对话框。

4.17.3.1. #外部条目

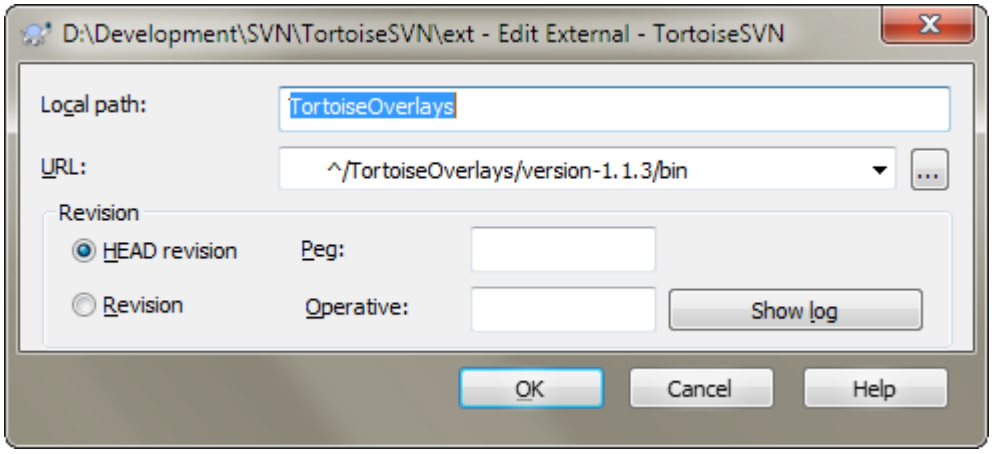


图 4.40. svn:externals 属性页

svn:外部 属性可用于从同一版本库或一个如 第 4.18 节 “外部条目”所述的完全不同的版本库中引入其他项目。

You need to define the name of the sub-folder that the external folder is checked out as, and the Subversion URL of the external item. You can check out an external at its HEAD revision, so when the external item changes in the repository, your working copy will receive those changes on update. However, if you want the external to reference a particular stable point then you can specify the specific revision to use. IN this case you may also want to specify the same revision as a peg revision. If the external item is renamed at some point in the future then Subversion will not be able to update this item in your working copy. By specifying a peg revision you tell Subversion to look for an item that had that name at the peg revision rather than at HEAD.

The button Find HEAD-Revision fetches the HEAD revision of every external URL and shows that HEAD revision in the rightmost column. After the HEAD revision is known, a simple right click on an external gives you the command to peg the selected externals to their explicit HEAD revision. In case the HEAD revision is not known yet, the right click command will fetch the HEAD revision first.

4.17.3.2. #SVN 关键字

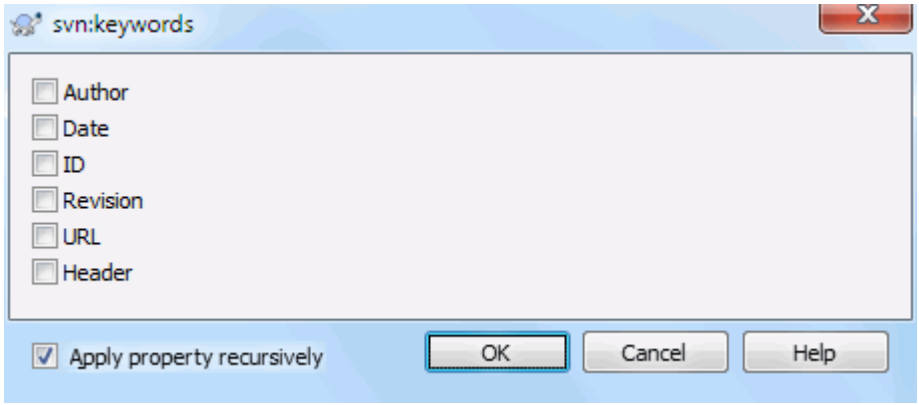


图 4.41. svn:keywords 属性页

选择你想要在文件中被扩展的关键字。

4.17.3.3. #EOL 样式

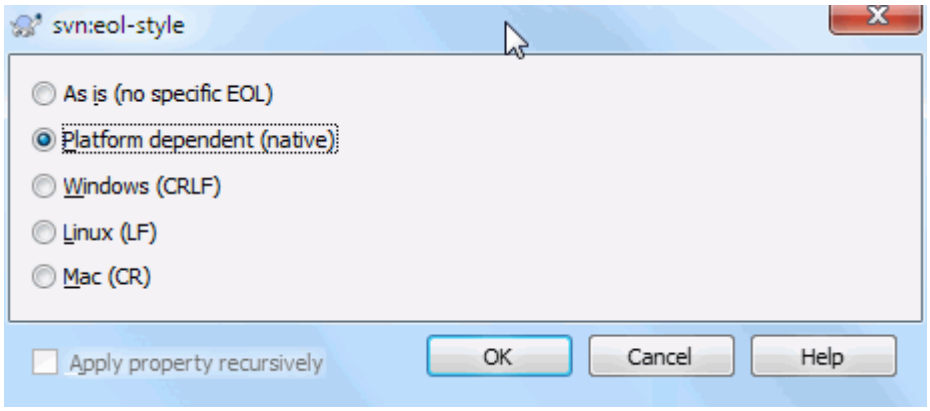


图 4.42. svn:eol-style 属性页

选择您想使用的并且 TortoiseSVN 会使用正确属性值的行结束样式。

4.17.3.4. #问题跟踪器集成

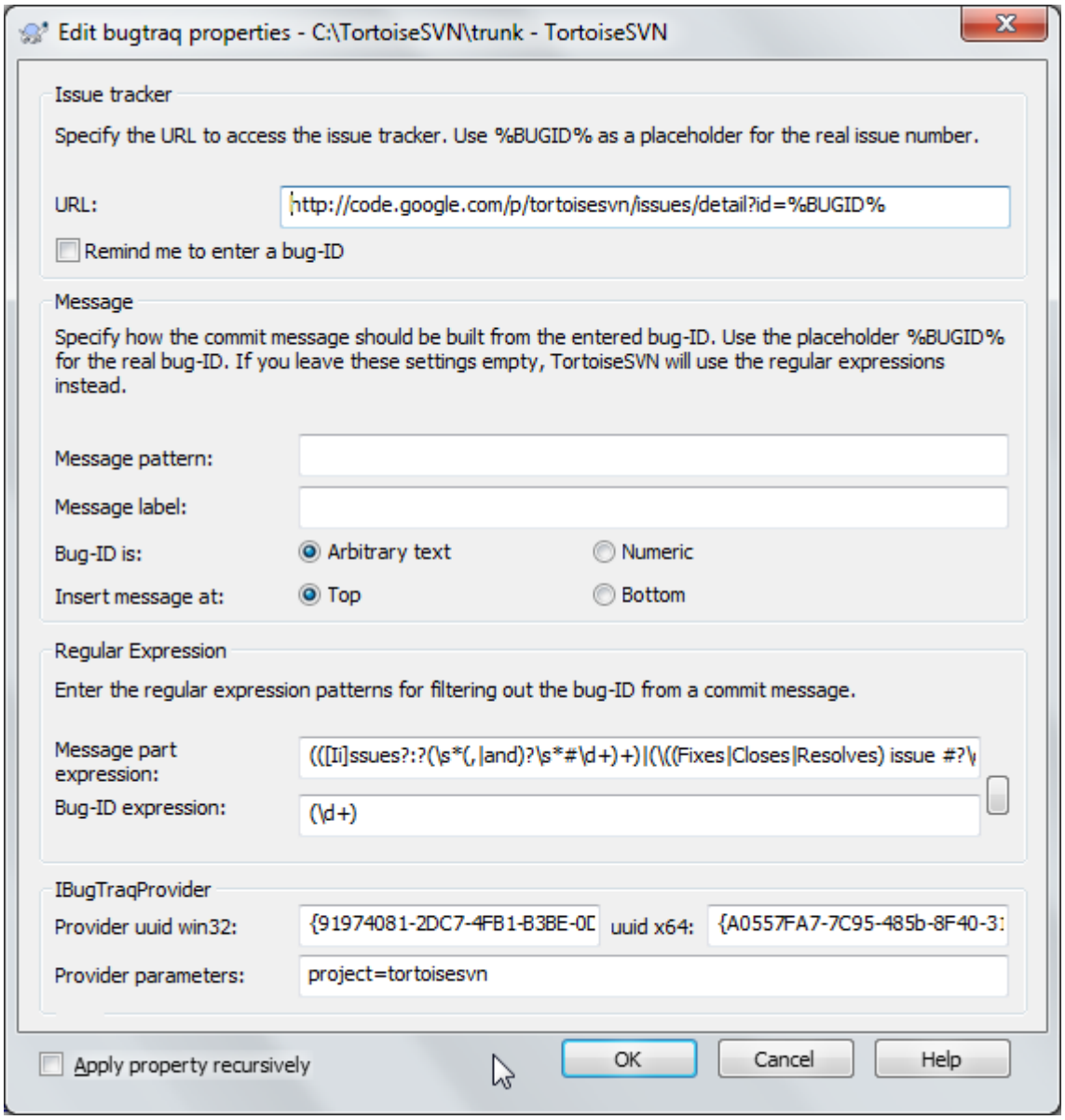


图 4.43. tsvn:bugtraq 属性页

4.17.3.5. #日志消息大小

图 4.44. 日志信息属性页的大小

These 3 properties control the formatting of log messages. The first 2 disable the OK in the commit or lock dialogs until the message meets the minimum length. The border position shows a marker at the given column width as a guide for projects which have width limits on their log messages. Setting a value to zero will delete the property.

4.17.3.6. #项目语言

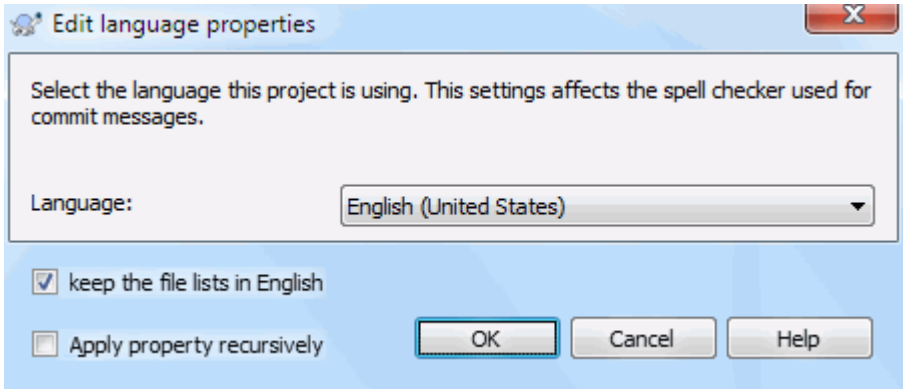


图 4.45. 语言属性页

Choose the language to use for spell-checking log messages in the commit dialog. The file lists checkbox comes into effect when you right click in the log message pane and select Paste file list. By default the Subversion status will be shown in your local language. When this box is checked the status is always given in English, for projects which require English-only log messages.

4.17.3.7.#MIME 类型

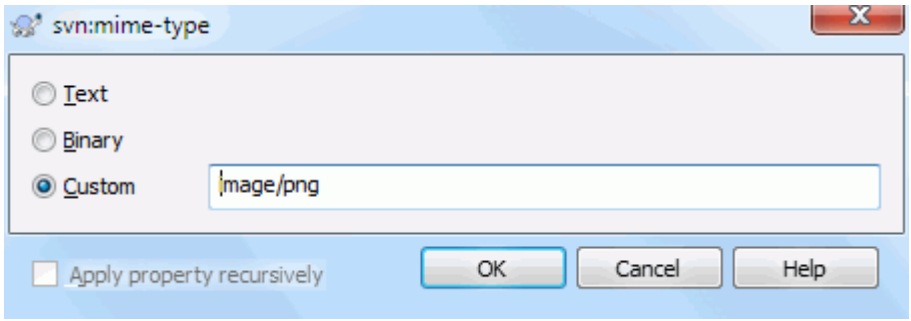


图 4.46. svn:mime-type 属性页

4.17.3.8.#svn:needs-lock

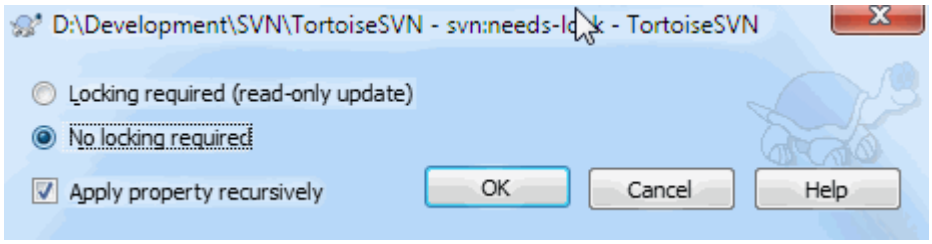


图 4.47. svn:needs-lock 属性页

此属性只是简单地控制一个文件是否要被检出为只读，如果没有在工作副本中锁定它。

4.17.3.9.#svn:executable

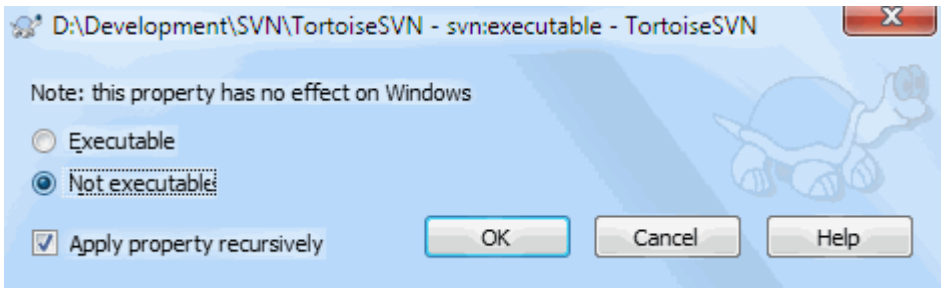


图 4.48. svn:executable 属性页

该属性控制一个文件在 Unix/Linux 系统上检出时是否将被给予可执行状态。在 Windows 签出时则没有任何影响。

4.17.3.10.#Merge log message templates

Whenever revisions are merged into a working copy, TortoiseSVN generates a log message from all the merged revisions. Those are then available from the Recent Messages button in the commit dialog.

You can customize that generated message with the following properties:

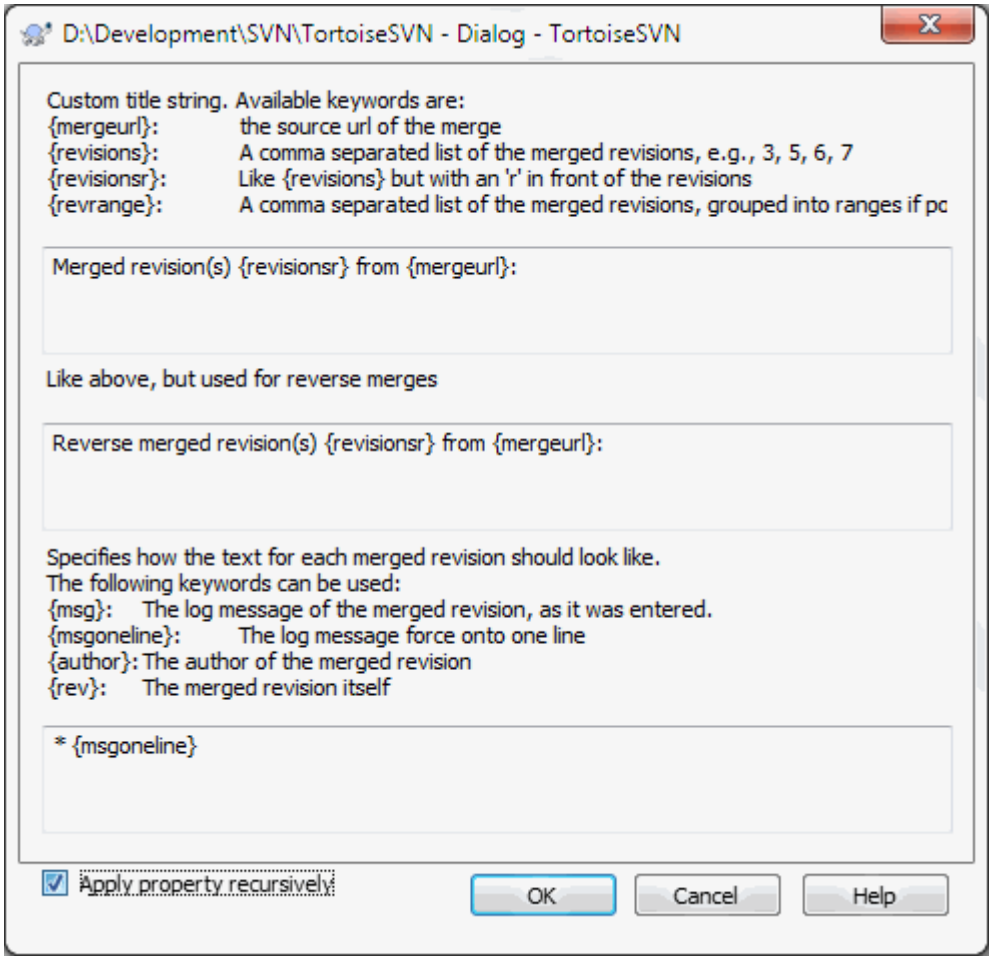


图 4.49. Property dialog merge log message templates

tsvn:mergelogtemplatetitle, tsvn:mergelogtemplatereversetitle
This property specifies the first part of the generated log message. The following keywords can be used:

{revisions}
A comma separated list of the merged revisions, e.g., 3, 5, 6, 7

{revisionsr}
Like {revisions}, but with each revision preceded with an r, e.g., r3, r5, r6, r7

{revrange}
A comma separated list of the merged revisions, grouped into ranges if possible, e.g., 3, 5-7

{mergeurl}
The source URL of the merge, i.e., where the revisions are merged from.

The default value for this string is Merged revision(s) {revrange} from {mergeurl}: with a newline at the end.

tsvn:mergelogtemplatemsg
This property specifies how the text for each merged revision should look like. The following keywords can be used:

{msg}
The log message of the merged revision, as it was entered.

{msgoneline}
Like {msg}, but all newlines are replaced with a space, so that the whole log message appears on one single line.

{author}
The author of the merged revision.

{rev}
The merged revision itself.

{bugids}
The bug IDs of the merged revision, if there are any.

This only works if the merged revisions are already in the log cache. If you have disabled the log cache or not shown the log first before the merge, the generated message won't contain any information about the merged revisions.

4.18. #外部条目

有时候，构建一个需要不同检出的工作目录是很有用的。举例来说，你也许需要来自版本库的不同位置的别的文件或子目录，或者可能完全来自不同的版本库。如果你需要每个用户具有相同的目录结构，你可以定义 svn:externals 属性来获取特定的资源到你需要的地方。

4.18.1. #外部文件夹

比方说您检出了一个 /project1 工作副本到 D:\dev\project1。选择文件夹 D:\dev\project1，右击该文件夹并从上下文菜单中选择 Windows 菜单 → 属性。属性对话框出现。然后转到 Subversion 选项卡。在那里您可以设置属性。单击 属性...。在属性对话框中，即可以双击 svn:externals 如果它已经存在，或点击 新建... 按钮并从菜单中选择 externals。要添加新的外部项目，单击 新建...，然后在显示的对话框中填写所需的信息。

URLs 必须正确转义否则将不会工作，例如必须将每个空格替换为 %20。

如果你需要在本地路径中包含空格或其它特殊字符，你可以使用双引号将它们括起来，或者你可以使用 Unix shell 风格的转义字符 -- 在特殊字符前添加 \ (反斜线)。当然这样就意味着你必须使用 / (正斜线)作为路径的分隔符。注意：这一特性是在 Subversion 1.6 中新引入的，不支持旧版本的客户端程序。

使用确定的版本号

你应当认真考虑在所有外部定义中使用确定的版本号，就像下面介绍的那样。这样做意味着当你下载扩展信息的个别快照时你已经作出决定，并且精确的指明了是哪个快照。而且你不会为第三方版本库的修改感到惊讶，这些版本库你可能没有任何控制，使用精确的版本号能使你回溯工作目录到以前的版本，你的外部定义也遵循此规则，看起来是以前的版本，即外部工作副本的更新匹配它们的老版本。对于软件工程，它是旧版本的复杂代码构建成功或失败的重要区别。

The edit dialog for svn:externals properties allows you to select the externals and automatically set them explicitly to the HEAD revision.

如果外部项目位于同一版本库，当你提交你的主要项目时，你在这儿所做的任何更改都将包含在提交列表中。

如果外部工程位于不同的版本库，当你向主项目提交你的修改时，你对外部工程做的修改会被通报，但是你必须单独的提交这些外部工程的修改。

如果你在 svn:externals 定义中使用绝对 URL 并且你不得不重定位你的工作副本(例如，版本库的 URL 改变了)，然后你的外部定义并不会改变，它可能就失效了。

要避免这样的问题，Subversion 客户端程序 1.5 版及更高版本支持相对外部 URL。四种不同的指定相对 URL 的方式被支持。在下面的例子中，假设我们有两个版本库: 一个位于 http://example.com/svn/repos-1，另一个位于 http://example.com/svn/repos-2。我们签出 http://example.com/svn/repos-1/project/trunk 到C:\Working 并且在 trunk 设置 svn:externals 属性。

相对于父目录
: 这些 URL 始终以字符串 ../ 开头，例如:

```
../../widgets/foo  common/foo-widget
```

这将会取出 http://example.com/svn/repos-1/widgets/foo 到 C:\Working\common\foo-widget。

注意: URL 是相对于具有 svn:externals 属性的目录所对应的 URL，而不是外部条目将要写入的硬盘目录。

相对于版本库的根
: 这些 URL 始终以字符串 ^/ 开头，例如:

```
^/widgets/foo  common/foo-widget
```

这将会取出 http://example.com/svn/repos-1/widgets/foo 到 C:\Working\common\foo-widget。

你可以很容易引用同一个 SVNParentPath (一个普通的目录包含多个版本库)下的其他版本库。例如:

```
^/../repos-2/hammers/claw  common/claw-hammer
```

这将会取出 http://example.com/svn/repos-2/hammers/claw 到 C:\Working\common\claw-hammer.

相对于方案
: 以字符串 // 开头的 URL 仅复制主版本库 URL 的方案部分。对于相同的主机，因为客户端所处网络的不同而需要使用不同的方案来访问时，这就很有用了；比如，内部网络的客户端使用 http:// 而外部客户端使用 svn+ssh://。例如:

```
//example.com/svn/repos-1/widgets/foo  common/foo-widget
```

这将会取出 http://example.com/svn/repos-1/widgets/foo 或者 svn+ssh://example.com/svn/repos-1/widgets/foo，这取决于签出 C:\Working 时使用哪种方式。

相对于服务器主机名称
: 以字符串 / 开头的 URL 复制主版本库 URL 的方案和主机名称部分。 for example:

```
/svn/repos-1/widgets/foo  common/foo-widget
```

这将会取出 http://example.com/svn/repos-1/widgets/foo 到 C:\Working\common\foo-widget。但如果你从另一个位于 svn+ssh://another.mirror.net/svn/repos-1/project1/trunk 的服务器签出工作副本，那么外部引用将会取出 svn+ssh://another.mirror.net/svn/repos-1/widgets/foo。

You can also specify a peg and operative revision for the URL if required. To learn more about peg and operative revisions, please read the corresponding chapter [http://svnbook.red-bean.com/en/1.8/svn.advanced.pegrevs.html] in the Subversion book.

如果你需要TortoiseSVN如何处理属性的更多信息，请阅读第 4.17 节 “项目设置”。

如果你需要知道存取公共子个我我你工程的不同方法，请阅读第 B.6 节 “包含一个普通的子项目”。

4.18.2. #外部文件

从 Subversion 1.6 版起，你可以将单独的文件作为外部引用添加到你的工作副本中，它使用和外部文件夹相同的语法格式。然而，这里有一些限制。

- 外部文件的路径必须将文件放置在一个存在的版本控制的文件夹下。通常情况下，将文件放在设置 svn:externals 属性的文件夹下是一个非常明智的举措，不过，如果需要，它可以放在一个版本控制的子文件夹下。相比之下，外部目录将会根据需要自动创建内部的未版本控制的文件夹。

- 外部文件的 URL 必须和插入外部文件的 URL 位于同一个版本库；不同版本库之间的外部文件不被支持。

外部文件行为在许多方面与其它版本控制的文件类似，但是它们不能使用普通的命令进行移动或删除；必须通过修改 svn:externals 来替代上述操作。

4.19. #分支/标记

版本控制系统的一个特性是能够把各种修改分离出来放在一个单独的开发线上。这条线被称为分支。分支经常被用来试验新的特性，而不会干扰正在修改编译器错误和 bug 的主开发线。当新的特性足够稳定之后，开发分支就可以合并回主分支里(主干).

版本控制系统的另一个特性是能够标记特殊的版本(例如一个发布版本)，所以你可以在任何时候重新建立一个特定的构建或环境。这个过程被称作标记。

Subversion 没有用于建立分支和标记的特殊命令，但是使用所谓的便宜复制来代替。便宜复制类似于 Unix 里的硬连接，它意思是代替一个版本库里的完整的复制，创建一个内部的链接，指向一个具体的版本树。结果分支和标记迅速被创建，并且几乎没有在版本库里占据任何额外的空间。

4.19.1. #创建一个分支或标记

如果你用推荐的目录结构导入了一个工程，那么创建分支或标记就非常简单:

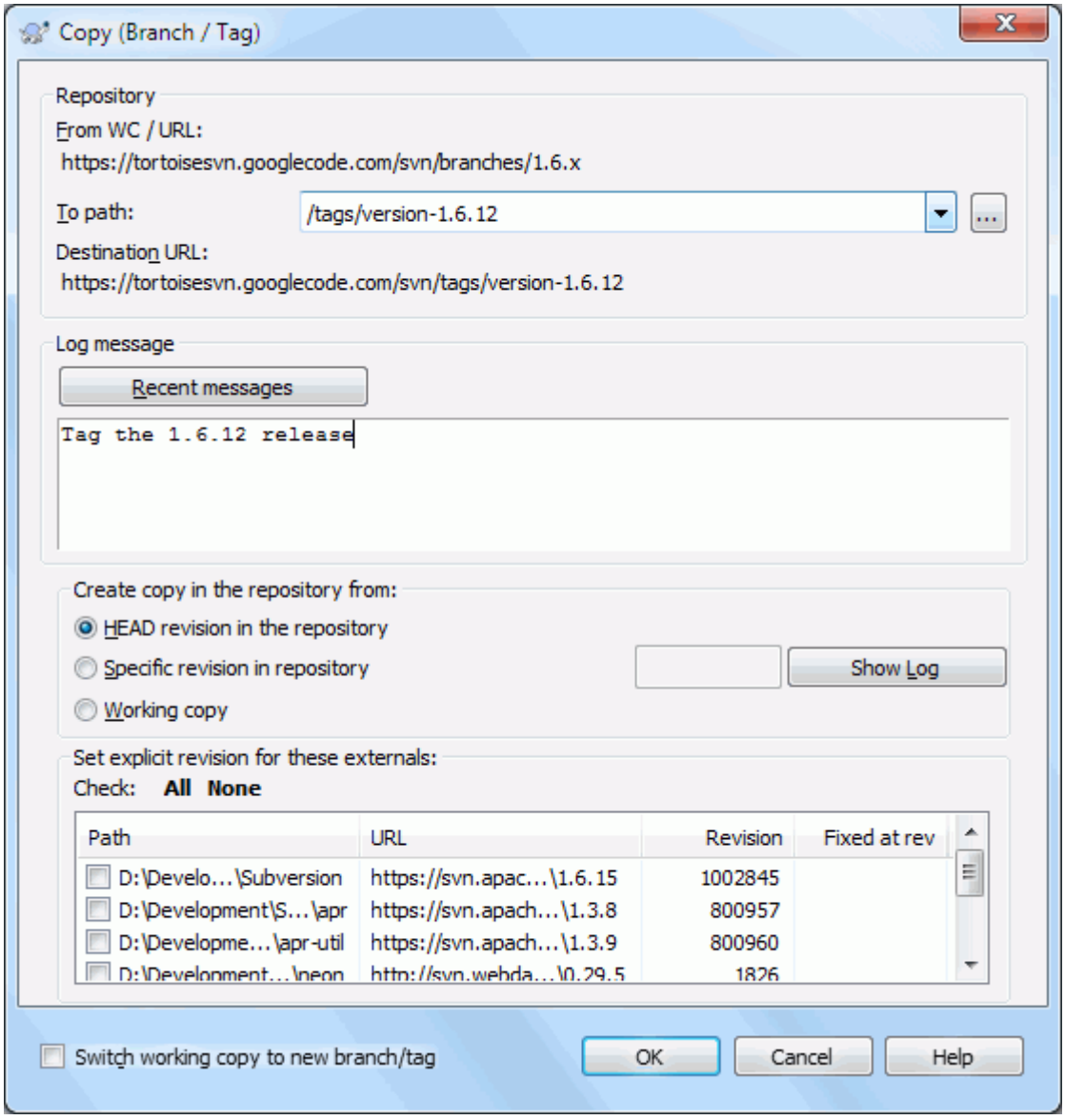


图 4.50. 分支/标记对话框

在你当前的工作副本中选择你想要复制的分支或标记的目录，然后选择命令TortoiseSVN → 分支/标记...。

默认的新分支的目标 URL 将会是你工作副本对应的源 URL。你需要修改这个 URL 为你的分支/标记的新路径。那么替代

```
http://svn.collab.net/repos/ProjectName/trunk
```

你也会使用类似这样的路径

```
http://svn.collab.net/repos/ProjectName/tags/Release_1.10
```

如果你不记得你上一次使用的命名规则，单击右边的按钮来打开版本库浏览器，这样你就可以查看现有的版本库结构。

intermediate folders

When you specify the target URL, all the folders up to the last one must already exist or you will get an error message. In the above example, the URL http://svn.collab.net/repos/ProjectName/tags/ must exist to create the Release_1.10 tag.

However if you want to create a branch/tag to an URL that has intermediate folders that don't exist yet you can check the option Create intermediate folders at the bottom of the dialog. If that option is activated, all intermediate folders are automatically created.

Note that this option is disabled by default to avoid typos. For example, if you typed the target URL as http://svn.collab.net/repos/ProjectName/Tags/Release_1.10 instead of http://svn.collab.net/repos/ProjectName/tags/Release_1.10, you would get an error with the option disabled, but with the option enabled a folder Tags would be automatically created, and you would end up with a folder Tags and a folder tags.

现在你需要选择从哪里复制。这里你有三个选择:

版本库中的最新版本
: 新的分支从最新版本复制到版本库中。没有数据需要从你的工作副本传递到版本库，而且分支非常迅速的被创建。

版本库中指定的版本
: 新的分支从你选选择的旧版本复制到版本库中。当你忘记为上一周发布的项目创建标记时，这就很有用了。如果你不记得版本号，点击右边的按钮来显示版本日志，然后从中选择版本号。也没有数据需要从你的工作副本传递到版本库，而且分支非常迅速的被创建。

工作副本
: 新的分支是与你本地工作副本一模一样的副本。如果你将工作副本中某些文件更新到某个早先的版本，或者你进行了本地修改，这些正是要进入副本的。当然，这些繁多的标记会作为传输数据从你的工作副本送回版本库，如果它们不存在于版本库中。

如果想要将工作副本自动切换到新创建分支，选中切换工作副本至新分支/标记复选框。但是如果你要这样做，首先确认你的工作副本中不包含修改。如果有，这些修改将会在你切换时合并到分支的工作副本中。

如果您的工作副本中有其他项目包含了 svn:externals 属性，在分支/标记对话框中的底部将列出这些外部对象。对于每个外部项目，目标路径、源地址和修订版本都会被显示。外部项目版本是从工作副本中确定的，这意味着它将显示外部项目实际指向的版本。

如果您要确保新的标记总是处于一致状态，请检查所有外部项目将它们的版本恢复到当前的工作副本版本。如果您不检查外部项目，这些外部项目会指向一个将来可能会更改的最新版本，检出新标记将会检出外部项目的最新版本并且你的标记可能不再编译。所以在创建一个标记时给外部对象设置一个明确的版本是一个不错的办法。

如果在创建分支或标记时给外部项目设置一个明确的版本，TortoiseSVN 将自动更改 svn:externals 属性。当从最新版本点或版本库中的指定修订版创建分支/标记时，TortoiseSVN 会首先创建分支/标记，然后调整属性。这将为每个属性创建额外的提交。当从工作副本创建见分支/标记时，首先修改属性，然后创建分支/标记，然后属性更改回其原始值。

按下确认提交新副本到版本库中。别忘了提供一条日志信息。需要注意的是这个副本是在版本库内部创建的。

需要注意除非你决定切换工作副本到新创建分支，建立一个分支或标记不会影响你的工作副本。即使你从工作副本创建分支，这些修改也会提交到新分支里，而不是到主干里，所以你的工作副本可能仍然标记为已修改状态来避免影响主干。

4.19.2. #创建分支或标记的其他方法

你可以在没有工作副本的情况下创建分支或标记。要这样做，打开版本库浏览器。你可以拖拽文件夹到新的位置。要创建副本，你必须在拖拽过程中按下 Ctrl 键，否则文件夹是被移动，不是被复制。

你可以使用鼠标右键拖拽文件夹。一旦你松开鼠标右键，你可以从右键菜单中选择移动或复制文件夹。当然，要创建分支或标记你必须复制文件夹，而不是移动它。

还有一个方法就是从日志对话框。你可以显示某个文件夹，例如 trunk，的日志对话框，选择一个版本(可以是位于顶端的最新版本，也可以是先前的版本)，右键单击并选择从版本创建分支/标记。

4.19.3. #检出或者切换

...这是个小问题。当从版本库中预期的分支检出所有数据到你的工作副本目录时，TortoiseSVN → 切换... 仅仅传输已经被修改的数据到你的工作副本中。有利于减轻网络负担，也有利于你的耐心。 :-)

为了能够使用你最新产生的副本或标记，你可以采用下面几种方法。你可以:

- TortoiseSVN → 检出一个最新的工作副本在一个空目录下。你可以在你的本地磁盘上的任意位置进行检出操作，同时你可以从版本库中按照你的意愿建立出任意数量的工作副本。
- 将你当前的工作副本切换到在版本库中新建立的副本。再一次选择你的项目所处的顶级文件夹然后在右键菜单中使用TortoiseSVN → 切换...。

  在接下来的对话框中输入你刚才建立的分支的 URL。选择最新版本单选按钮然后确认。你的工作副本就切换到了最新的分支/标记。

  切换操作起来就象更新，因为它没有丢弃你在本地做的修改。当你进行切换的时候，工作副本里任何没有提交过的修改都会被合并。如果你不想看到这样的结果，那么你可以有两种选择，要么在切换前提交修改，要么把工作副本恢复到一个已经提交过的版本(通常是最新版本)。
- 如果你需要在主干和分支上工作，但是又不想耗费资源来进行一个全新的检出操作，你可以使用 Windows 资源管理器来将你的主干工作副本复制到另一个文件夹，然后使用TortoiseSVN → 切换...这样就会复制新的分支。

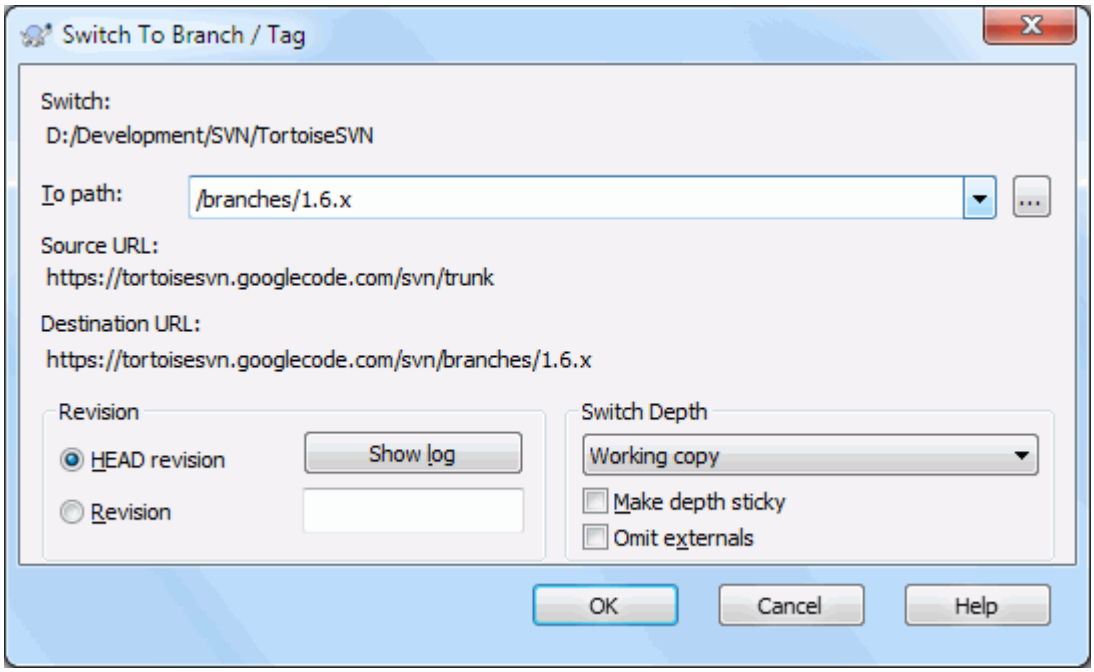


图 4.51. 切换对话框

尽管 Subversion 本身不区分标记和分支，但它们通常被应用的场合还是有些不同。

- 标记被用来建立一个项目在某个特殊的阶段的静态映像。通常情况下他们本不是用来进行开发的 - 分支是用来进行开发的，这就是我们推荐 /trunk /branches /tags 这样的版本库结构的原因。在标记上工作并不是一个好想法，因为你的本地文件没有写保护，没有什么办法防止你误操作。然而，如果你试着提交一个包含 /tags/ 的路径到版本库，TortoiseSVN 会警告你。

- 如果你需要在一个已经标记的发布版上做更多的修改。正确的操作方法是先从标记处建立一个新分支然后提交这个分支。在这个分支的基础上进行修改后再从这个新分支上建立一个新标记，例如 Version_1.0.1。

- 如果你修改了一个从分支建立的工作副本然后又提交了这个副本，那么所有的修改会转到一个新分支里而不是 主干。仅仅是存储了修改的数据。其余的数据还是便宜复制。

4.20. #合并

分支用来维护独立的开发支线，在一些阶段，你可能需要将分支上的修改合并到主干，或者相反。

It is important to understand how branching and merging works in Subversion before you start using it, as it can become quite complex. It is highly recommended that you read the chapter Branching and Merging [http://svnbook.red-bean.com/en/1.8/svn.branchmerge.html] in the Subversion book, which gives a full description and many examples of how it is used.

下一个需要注意的地方是，合并总是在工作副本中进行。如果你想要合并修改到分支，你必须检出该分支的工作副本，并且从这个工作副本使用TortoiseSVN → 合并...来调用合并向导。

通常来说，在没有修改的工作副本上执行合并是一个好想法。如果你在工作副本上做了修改，请先提交。如果合并没有按照你的想法执行，你可能需要撤销这些修改，命令SVN 还原 会丢弃包含你执行合并之前的所有修改。

这里有三个处理方法稍微不同的用例，如下所述。合并向导的第一页会让你选择你需要的方法。

合并一个版本范围
: 这个方法覆盖了你已经在分支(或者主干)上做出了一个或多个修改，并且你想将这些修改应用到不同分支的情况。

  你要 Subversion 做如下事情: “计算[从]分支 A 的版本 1 [到]分支 A 的版本 7 所需的修改，并将这些改变应用到(主干或分支 B 的)工作副本。”

  If you leave the revision range empty, Subversion uses the merge-tracking features to calculate the correct revision range to use. This is known as a reintegrate or automatic merge.

合并两个不同的树
: 这是复兴合并的通用情况。你要 Subversion 做如下事情: “计算[从]主干的最新版本[到]分支的最新版本所需要的修改，并将这些修改应用到(主干的)工作副本。”最终结果就是主干看起来与分支一模一样。

  If your server/repository does not support merge-tracking then this is the only way to merge a branch back to trunk. Another use case occurs when you are using vendor branches and you need to merge the changes following a new vendor drop into your trunk code. For more information read the chapter on vendor branches [http://svnbook.red-bean.com/en/1.8/svn.advanced.vendorbr.html] in the Subversion Book.

4.20.1. #合并指定版本范围

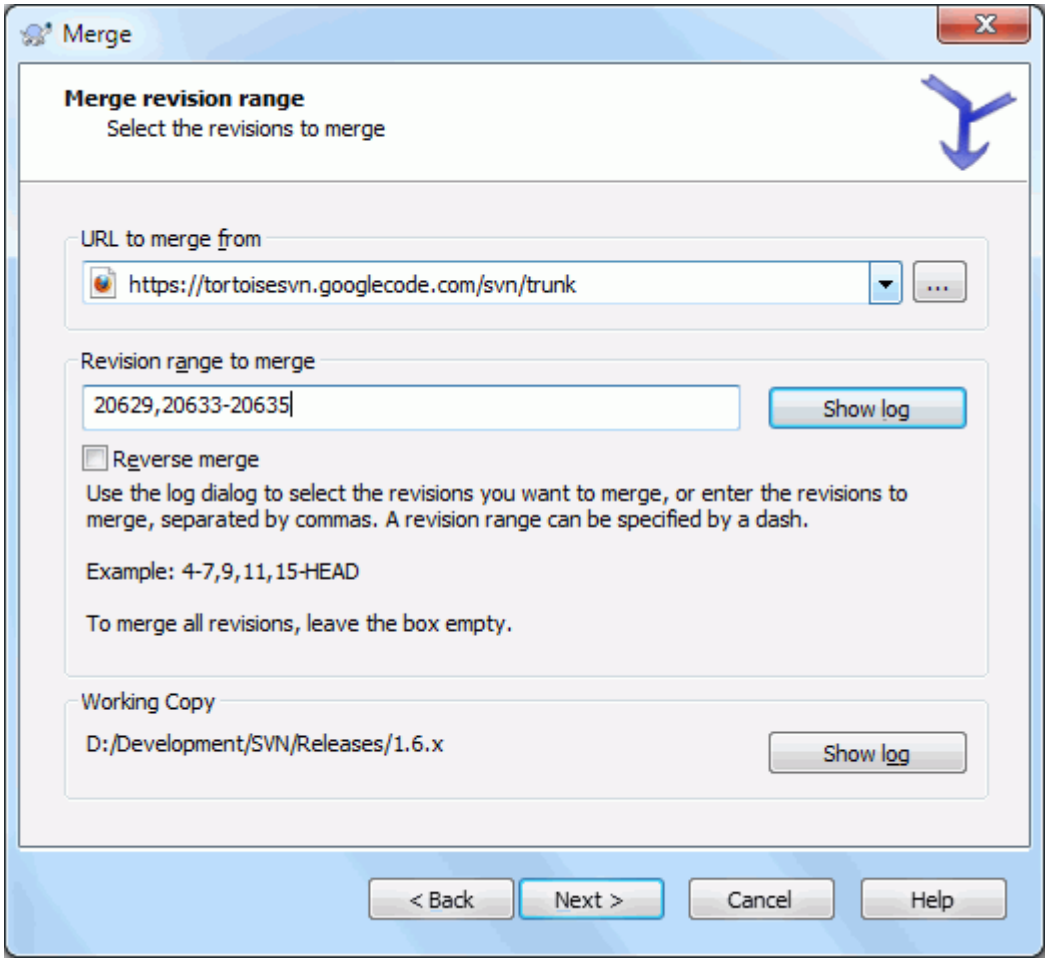


图 4.52. 合并向导 - 选择版本范围

在从:文本框中输入分支或标记文件夹的完整 URL，它包含了你想应用到工作副本的修改。你也可以点击...浏览版本库，找到想要的分支。如果你以前已经从这个分支合并过，可以直接从包含历史的下拉列表选择以前使用的 URL。

If you are merging from a renamed or deleted branch then you will have to go back to a revision where that branch still existed. In this case you will also need to specify that revision as a peg revision in the range of revisions being merged (see below), otherwise the merge will fail when it can't find that path at HEAD.

在待合并的版本范围文本框中输入你想合并的版本列表。它可以是单一版本，用逗号隔开的指定版本列表，或用横线(-)连接的版本范围，或这些形式的组合。

If you need to specify a peg revision for the merge, add the peg revision at the end of the revisions, e.g. 5-7,10@3. In the above example, the revisions 5,6,7 and 10 would be merged, with 3 being the peg revision.

在 TortoiseSVN 中指定版本范围的方法与命令行客户端大相径庭。用最简单的方法来说明这一点，设想一条栅栏，有支撑柱和栅栏板。

在命令行客户端，你使用两个“支撑柱”版本来指定需要合并的修改，这两个版本指明了修改前和修改后的点。

在 TortoiseSVN，你使用“栅栏板”来指定要合并的修改集。当你使用日志对话框来选择需要合并的版本时这样做的原因就很清晰，每个版本看上去就是一个修改集。

> If you are merging revisions in chunks, the method shown in the Subversion book will have you merge 100-200 this time and 200-300 next time. With TortoiseSVN you would merge 100-200 this time and 201-300 next time.
>
> 这个区别在邮件列表中引起了热烈的讨论。我们知道它与命令行客户端不同，但是我们相信，对于大多数的图形界面用户来说，我们制定的方法更容易理解。

选择版本范围最简单的方法是，点击显示日志，列出最近的修改和日志。如果你要合并单个版本的修改，直接选取那个版本。如果你要合并多个版本，就选择范围(使用通常的Shift键)。点击确认，就会为你填写要合并的版本号列表。

如果您要将更改往回合并 出 您的工作副本，以还原一个已提交的更改，选择要还原的版本并确保选中了 反向合并 框。

如果你已经从这个分支合并了一些修改，希望你在提交日志中注明最后一个合并的版本号。这时，你可以在工作副本上使用显示日志对话框跟踪日志。记住，我们将版本号视作修改集，你应该使用最后合并的版本之后的版本作为本次合并的开始版本。例如，上次你已经合并了版本37到39，那么本次合并你应该从版本40开始。

如果你在使用 Subversion 的合并跟踪特性，你就不需要记住已经合并了那些版本 - Subversion 将会帮你记录。如果不填写版本范围，所有未合并的版本将被选中。阅读第 4.20.5 节 “合并跟踪获得更多信息。

When merge tracking is used, the log dialog will show previously merged revisions, and revisions pre-dating the common ancestor point, i.e. before the branch was copied, as greyed out. The Hide non-mergeable revisions checkbox allows you to filter out these revisions completely so you see only the revisions which can be merged.

如果其他用户可能提交，那么要小心使用最新版本。如果有人在你最近更新之后提交了，它指代的版本可能就不是你想的那样了。

If you leave the range of revisions empty or have the radio button all revisions checked, then Subversion merges all not-yet merged revisions. This is known as a reintegrate or automatic merge.

复兴合并需要满足一些条件。首先，服务器必须支持合并跟踪。工作副本必须为完全递归 (不是有限的检出深度)，而且它不能具有任何本地修改、交换项目或已更新为最新版本之外的其他版本的项目。分支开发过程中的所有主干更改必须已合并到分支 (或标记为已合并)。要合并的版本范围将自动被计算。

点击下一页跳转到第 4.20.3 节 “合并选项”。

4.20.2. #合并两个不同的目录树

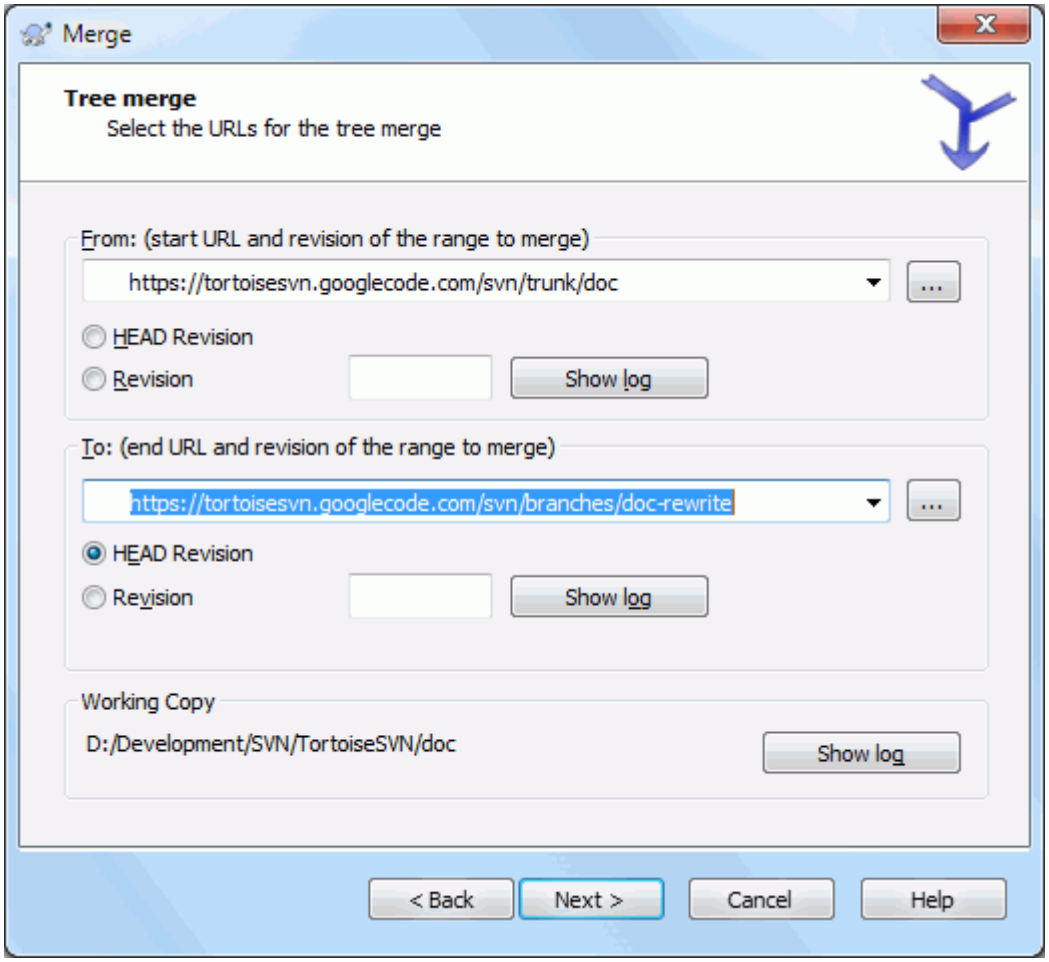


图 4.53. 合并向导 - 树合并

如果您正使用此方法将一个特性分支合并回主干，你需要从一个主干工作副本内启动合并向导。

在 起始: 区域输入 主干 的完整文件夹地址。这听起来像是错了，但请记住主干是要添加分支更改的起始点。你也可以点击 ... 浏览版本库。

在到:域输入关注的分支中文件夹的全路径。

在开始版本和结束版本 域，输入两个树被同步的最后一个版本号。如果你确信没有其他人提交，两个都可是输入 HEAD。如果在同步时可能有人提交的话，使用清楚的版本号以便面丢失最新提交。

也可以使用 显示日志 选择版本。

4.20.3. #合并选项

此向导页可以指定合并进程开始前的高级选项。一般情况下你可以直接使用默认设置。

您可以指定合并深度，也就是到工作副本的合并程度。深度条款在 第 4.3.1 节 “检出深度”中有所描述。默认深度是 工作副本，使用现有的深度设置，并且几乎能满足所需。

Most of the time you want merge to take account of the file's history, so that changes relative to a common ancestor are merged. Sometimes you may need to merge files which are perhaps related, but not in your repository. For example you may have imported versions 1 and 2 of a third party library into two separate directories. Although they are logically related, Subversion has no knowledge of this because it only sees the tarballs you imported.

If you attempt to merge the difference between these two trees you would see a complete removal followed by a complete add. To make Subversion use only path-based differences rather than history-based differences, check the Ignore ancestry box. Read more about this topic in the Subversion book, Noticing or Ignoring Ancestry [http://svnbook.red-bean.com/en/1.8/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry].

您可以指定处理行结束和空白变化的方式。这些选项在 第 4.10.2 节 “行结束符和空白选项”中被描述。默认情况是把所有的空白和行结束差异当做将被合并的真实更改。

标记为 强制合并 的复选框用于避免树冲突，传入的删除操作会影响到本地修改或不受版本控制的文件。如果删除了该文件，则无法恢复，这就是为什么在默认情况下未选中该选项。

如果您正在使用合并跟踪并且要将一个版本标记为已合并，而实际上并没有执行合并，那么就选中 只记录合并 复选框。您这样做可能有两个原因。合并过于复杂，合并算法表示压力很大，因此您选择手动更改代码，然后将更改标记为已合并，以便于合并跟踪算法也能识别到。或者您可能想要防止某个特定版本被合并，将其标记为已合并就可以用合并跟踪识别客户端来阻止合并的发生。

现在一切就绪，你要做的就是点击 合并 按钮。如果想要预览结果，测试合并 可以模拟合并操作，但完全 不会 修改工作副本。它会显示一份会被真正合并的文件列表，并通告那些 可能 发生冲突的文件。因为合并跟踪会使合并过程变得更为复杂，没有任何方法可以事先保证完成合并时是否会出现冲突，所以在测试合并中标记为冲突的文件可能在实际的合并中并不会出现任何问题。

合并进度对话框中会显示每个合并阶段，附带所涉及的版本范围。这可能意味着有超出你预料的更多版本。例如，如果你要求合并版本 123，进度对话框将报告为 “Merging revisions 122 through 123”。要理解这一点需要记住合并跟差异是密切相关的。合并进程的工作方式是在版本库两个点之间生成一份差异列表，并将这些差异应用至您的工作副本。进度对话框仅显示差异的开始点和结束点。

4.20.4. #预览合并结果

现在已完成合并。查看合并是否按预期完成是个不错的主意。合并通常是相当复杂的。如果分支跟主干偏差太大，往往会出现冲突。

Whenever revisions are merged into a working copy, TortoiseSVN generates a log message from all the merged revisions. Those are then available from the Recent Messages button in the commit dialog.

To customize that generated message, set the corresponding project properties on your working copy. See 第 4.17.3.10 节 “Merge log message templates”

对于 Subversion 客户端和服务器 1.5 之前的版本，不会存储合并信息并且合并的版本必须手动进行跟踪。当您测试完更改来提交这次修订版本时，您的提交日志信息应该会 始终 包括在合并中已导入的版本号。如果以后要应用另一个合并，您需要知道你已经合并过的，因为没人愿意多次地导入更改。这有关的详细信息，请参阅 Subversion 书册中的 最佳的合并实践 [http://svnbook.red-bean.com/en/1.4/svn.branchmerge.copychanges.html#svn.branchmerge.copychanges.bestprac]。

如果你的服务器和所有客户端都使用 Subversion 1.5 或更高版本，合并跟踪工具会记录已经合并的版本，避免某个版本被合并多次。它会使你的生活更简单，因为你每次都可以简单的合并全部版本范围，并且知道只有新版本才会实际被合并。

分支管理很重要。如果你要保持这个分支与最新版本同步，你应当经常合并，这样分支和最新版本的差别就不会太大。当然，你仍旧应该遵循上面的说明，避免重复合并修改。

如果你刚刚将一个特性分支合并到主干，那么主干现在包含所有新的特性代码，分支就已经无用了。如果需要的话，现在可以从版本库中将其删除。

Subversion 无法用一个文件夹来合并一个文件，反之亦然 - 只能是文件夹对文件夹，文件对文件。如果您点击一个文件并打开合并对话框，然后你必须在对话框中给出一个文件路径。如果您选择一个文件夹并打开对话框，然后您必须为合并指定一个文件夹地址。

4.20.5. #合并跟踪

Subversion 1.5 引入了合并跟踪特性。当你合并版本树时，版本号会被保存，此信息可以用于几个目的。

- 您可以避免合并同一版本两次的隐患 (重复合并问题)。一旦一个版本被标记为已合并，将来在版本范围中包含该版本的合并将会跳过它。
- 当您合并一个分支到主干时，日志对话框可以将分支提交作为主干日志的一部分来显示，这样可得到对更改更好的跟踪性。
- 当您从合并对话框内显示日志对话框时，已合并的版本显示为灰色。
- 当显示一个文件的追溯信息时，你可以选择显示已合并版本的原作者，而不是合并者。
- 您可以把版本标记为 不合并，通过把他们包含在已合并的版本列表中而实际上并没有进行合并。

执行合并时客户端将合并跟踪信息存储在 svn:mergeinfo 属性中。提交合并时，服务器会将该信息存储在数据库中，而当您请求合并、日志或追溯信息时，服务器可以作出恰当的响应。为了使系统正常工作，您必须确保服务器、版本库和所有客户端都是最新的。较早的客户端将不会存储 svn:mergeinfo 属性并且早期版本的服务器端不会提供新版客户端所请求的信息。

Find out more about merge tracking from Subversion's Merge tracking documentation [http://svn.apache.org/repos/asf/Subversion/trunk/notes/merge-tracking/index.html].

4.20.6. #子合并期间处理冲突

合并不会总是顺利的。有时会有冲突，如果您正在合并多个范围，您通常要在下一个范围的合并开始之前解决冲突。TortoiseSVN 会在进程中通过显示 合并冲突回调 对话框来帮助你解决。

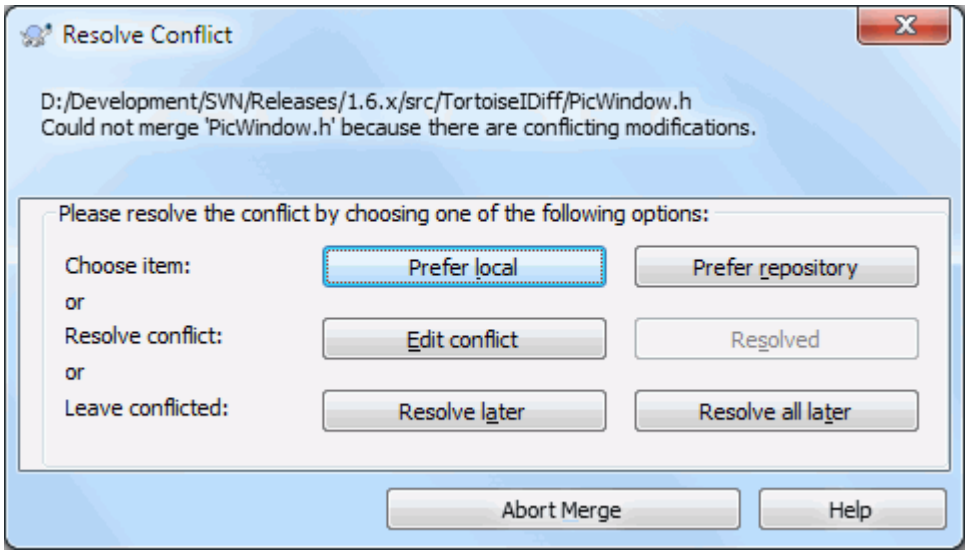


图 4.54. 合并冲突回调对话框

很可能某些更改将顺利进行合并，而其他的本地更改会跟已提交到版本库的更改发生冲突。所有可以合并的更改都将被合并。合并冲突回调对话框会给你您三种不同的方法来处理冲突的行。

1. If your merge includes binary files, merging of conflicts in those is not possible. You have to choose one complete file. Use Prefer local to select the local version as it was

in your working copy prior to the merge, or Prefer repository to select the incoming file from the merge source in the repository.

If you are merging text files then these first two buttons allow you to merge non-conflicting lines as normal and always prefer one version where there are conflicts. Choosing Prefer local will select your local version in every conflict, i.e. it will prefer what was already there before the merge over the incoming change from the merge source. Likewise, Prefer repository will select the repository changes in every conflict, i.e. it will prefer the incoming changes from the merge source over what was already in your working copy. This sounds easy, but the conflicts often cover more lines than you think they will and you may get unexpected results.

2. 通常情况下，你会想看看冲突和自己解决他们。在这种情况下，选择 编辑冲突 将启动合并工具。当你对结果满意时，请点击 已解决。

3. 最后一个选项就是延迟解决并继续进行合并。您可以为当前的冲突文件做这样的选择，或者为合并余下部分的所有文件。但是，如果该文件在接下来的合并中有进一步的更改，它将无法完成合并。

如果您不想使用交互式回调，在合并进度对话框中有一个 非交互式合并 复选框。如果选中它，若合并中出现了冲突，该文件会被标记为冲突，然后继续合并。你必须在整个合并完成后解决这些冲突。如果不选中它，那么在冲文件被标记为冲突之前你有机会在合并 过程中 解决冲突。它的优势是如果一个文件获得多个合并 （多个修订版本将更改应用于该文件），后续的合并是否成功取决于受影响的行。但这样在合并期间你可不能离开去尿尿哦 ;)

4.20.7. #合并已完成的分支

如果您要从分支合并所有更改至主干，您可以从扩展的上下文菜单（按住 Shift 键同时右击该文件）中使用 TortoiseSVN → 合并复兴分支...。

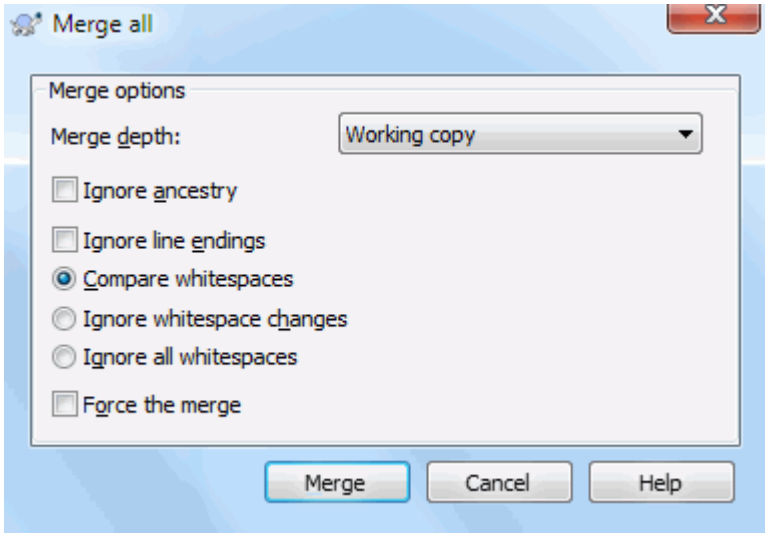


图 4.55. 合并复兴分支对话框

此对话框很简单。你所要做的就是设置合并选项，如 第 4.20.3 节 “合并选项”中所述。剩下部分是通过 TortoiseSVN 自动使用合并跟踪来完成。

4.20.8. #特性分支维护

当您在一个特性分支上开发一项新特性并完成时，为其重新整合至主干制定一个策略是一个好主意。如果同一时间 主干 上有其他工作也在继续，您可能会发现随着时间推移，差异会变得更显著，复兴合并将成为一场噩梦。

如果新特性较为简单并且开发时间较短，那么您可以采取一种简单的方法，就是将分支完全独立，直到完成该特性，然后将分支更改合并回主干。在合并向导中，选择简单的 合并一个版本范围，该版本范围即是分支的版本跨度。

如果该特性需要花费更长的时间，就要考虑到 主干 中的变化，那么你就需要保持分支同步。这仅仅意味着定期地将主干的修改合并到分支，使分支包含所有主干的修改并 加上 新特性。同步过程使用 合并一个版本范围 。当完成新特性时，那么你可以使用 复兴分支 或 合并两个不同的树 将其合并回 主干。

4.21. #锁

使用之前在第 2.2.3 节 “复制-修改-合并 方案”中描述的“复制-修改-合并” 的方法，Subversion 通常不需要锁就可以很好的工作。但是，在某些情况下你可能需要制定某种锁定策略。

- 例如，你使用图形文件等“不能合并”的文件。如果两个人修改同一个这样的文件，合并是不可能的，所以你丢失其中一个的修改。
- 您的公司过去一直使用一个锁定版本控制系统，并且已有一个 “锁定是最好的” 管理决策。

首先，你需要确保你的 Subversion 服务器端升级到至少 1.2 版本。早期版本不支持锁定。如果您正在使用 file:// 访问，那么当然只有你的客户端需要更新。

The Three Meanings of “Lock”

In this section, and almost everywhere in this book, the words “lock” and “locking” describe a mechanism for mutual exclusion between users to avoid clashing commits. Unfortunately, there are two other sorts of “lock” with which Subversion, and therefore this book, sometimes needs to be concerned.

The second is working copy locks, used internally by Subversion to prevent clashes between multiple Subversion clients operating on the same working copy. Usually you get these locks whenever a command like update/commit/... is interrupted due to an error. These locks can be removed by running the cleanup command on the working copy, as described in 第 4.16 节 “清理”

And third, files and folders can get locked if they're in use by another process, for example if you have a word document opened in Word, that file is locked and can not be accessed by TortoiseSVN.

You can generally forget about these other kinds of locks until something goes wrong that requires you to care about them. In this book, “lock” means the first sort unless the contrary is either clear from context or explicitly stated.

4.21.1. #锁定在Subversion中是如何工作的

默认情况下，所有的东西都没有锁定，只要有提交权限的人都可以在任何时候提交任何的文件。其他人会定时更新他们的工作副本，在库中的改变的东西都会与本地合并。

如果你对一个文件 取得锁定，那么只有你可以提交这个文件。其他用户的提交都会被拒绝，直到你释放了这个锁。一个被锁定的文件不能在库中进行任何形式的合并。所以它不能除锁的拥用者之外的人删除或更名。

锁定不是分配给一个特定用户，而是一个特定用户和一个工作副本。工作副本上的锁定也可以防止同一用户从另一个工作副本中提交已锁定的文件。

举一个例子，假设用户 Jon 在他的办公室电脑上有一个工作副本。他开始对一个图像进行工作，并因此获得了该文件的锁定。当他离开办公室时还没有完成该文件的工作，所以他不会解除该锁定。回到家里 Jon 也有一个工作副本，并决定继续该项目的工作。但是他却不能修改或提交相同的图像文件，因为该文件的锁定驻留在办公室里的工作副本中。

但是，其他用户不必知道你已经增加了锁定，除非他们定期地检查锁定的状态。这其实没什么意义，因为他们发现提交失败的时候就可以知道锁定了。为了更容易管理锁，而设置了一个新的Subversion属性svn:needs-lock。当一个文件的这个属性被设置(成任意值)的时候，每当该文件检出或更新时，本地的副本都被设成只读，除非该工作副本就是拥有锁的那个用户的。这么做是为了能警告你，你不应该修改这个文件，除非你申请到了锁定。受控只读的文件在TortoiseSVN中用一个特殊的图标来表示你需要在编辑前取得锁定。

锁除了按所有者记录外，还在工作副本中记录。如果你有多个工作副本(在家，在单位)，那么在这些工作副本中，只允许对其中一份拥有锁。

如果你的合作者之一请求一个锁，但却外出旅游去了，你怎么办？Subversion提供了一种强制锁。释放别人拥有的锁被称为破坏锁定，强制获得别人拥有的锁称为窃取锁定。当然，如果你想要与你的合作者保持良好的关系，轻易不要这么做。

锁在库中进行记录，一个锁定令牌建立在你的本地工作副本中。如果有矛盾，比如某人破坏了锁下，那么本地的锁定令牌将不可用。库中的记录将是最权威的参考。

4.21.2. #取得锁定

选择工作副本中你想要获取锁定的文件，然后选择命令TortoiseSVN → 取得锁定....

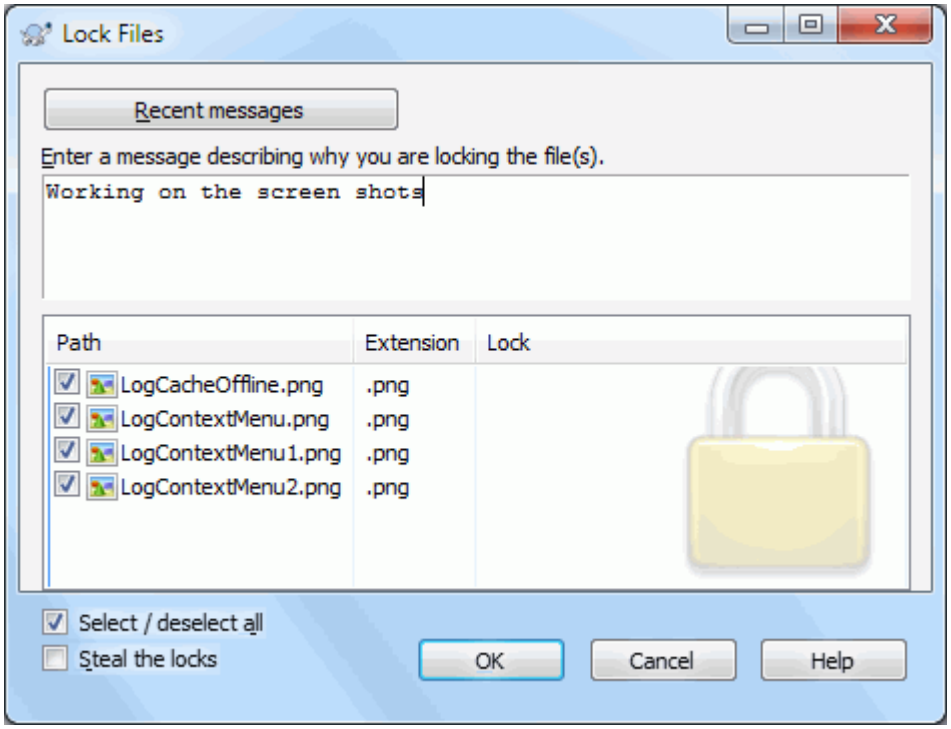


图 4.56. 锁定对话框

出现一个对话框，允许你输入注释，这样别人可以知道你为什么锁定这个文件。注释是可选的，并且只用于基于Svnserve的库。当(且仅当)你需要窃取别人的锁的时候，勾选偷取此锁定复选框，然后点击确定。

您可以设置项目属性 tsvn:logtemplatelock 给要填写如锁定信息的用户提供一个信息模板。请参阅第 4.17 节 “项目设置”有关如何设置属性的说明。

如果你选择一个文件夹，使用TortoiseSVN → 获取锁定...锁定对话框将显示所有子文件夹中的所有文件 。如果你真的要锁定整个目录，就这么做，但如果你这么做，可能会很不受你的合作者的欢迎。小心使用...

4.21.3. #释放锁定

为了确保你不会忘记释放锁，你不需要做别的事，在提交对话框中，总是会显示锁定的文件，并总是默认被选中。如果你继续提交，选中的文件中的锁就被移除了，就算你从没有修改过。如果你不希望释放某文件的锁，你可以取消选中它(如果你没有修改过)。如果你希望保持一个修改过的文件的锁，你需要在提交之前选中保持锁定复选框。

要手动释放锁定，选中工作副本中要释放的文件，选择命令TortoiseSVN → 释放锁定。不需要输入什么TortoiseSVN会联系版本库并释放锁。你可以对一个文件夹来使用这个命令释放其中的所有锁定项。

4.21.4. #检查锁定状态

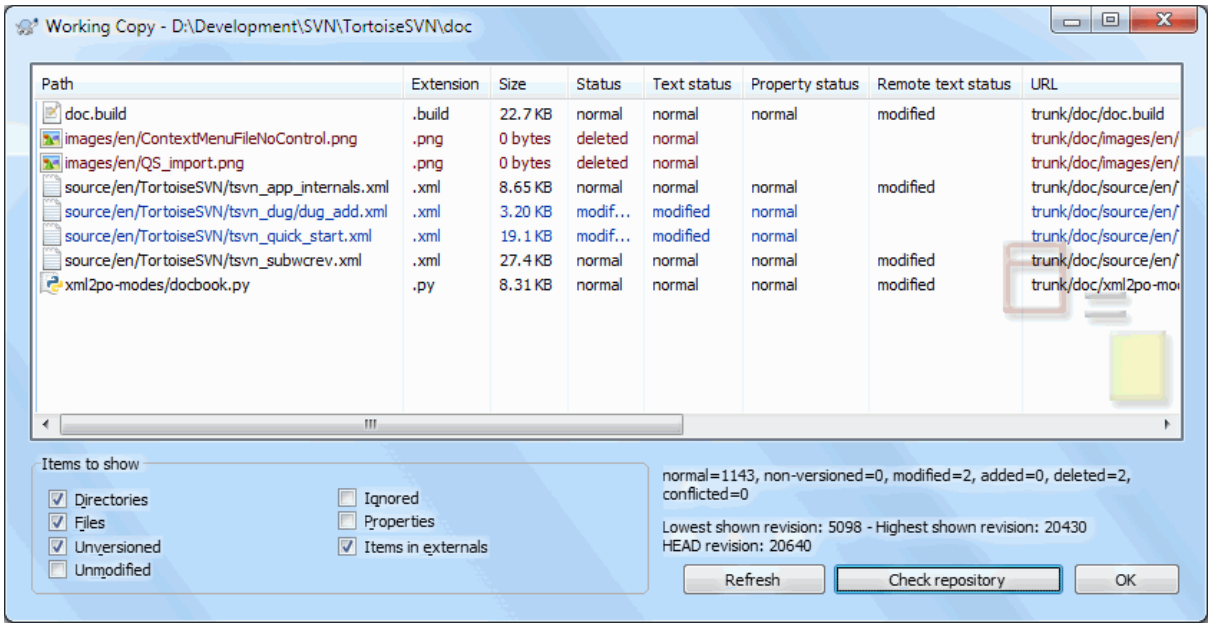


图 4.57. 检查修改对话框

要查看你和他人所拥有的锁，你可以使用TortoiseSVN → 检查修改...命令。本地的锁定令牌会立即显示出来，要查看别人拥用的锁定(或是检查你的锁是否被破坏或窃取)你需要点击检查版本库。

从此处的右键菜单中，你可以获取锁或是释放锁，也可以破坏或是窃取别人的锁。

避免破坏和窃取锁定

如果你在破坏或是窃取某人的锁的时候没有告诉他，你可能会丢掉工作。如果你正在写一个不可合并的文件类型，并且你窃取了某人的锁，一旦你释放了锁，他们就可以随意检入他们的修改以覆盖你的。Subversion并不会丢弃数据，但你却失去了锁带来的对团队工作的保护。

4.21.5. #让非锁定的文件变成只读

正如上面提到的，最有效的使用锁的方式是对一个文件设置svn:needs-lock属性。参考第 4.17 节 “项目设置”如何设置属性。带有此属性的文件将总被按只读的方式检出和更新(除非你的工作副本拥有锁定)。

作为提醒的方式之一，TortoiseSVN使用一个特殊的图标表示。

如果你想实行一个每个文件必须被锁定的策略，那么您可能会发现使用 Subversion 的 auto-props 功能来为你每次新添加的文件自动设置属性是更容易的。自动设置属性每次添加新的文件。阅读第 4.17.1.5 节 “自动属性设置”以进一步了解信息。

4.21.6. #锁定钩子脚本

当你使用Suversion1.2或更新的版本建立新的版本库的时候，建立了四个钩子模板在版本库中的hooks目录下。这些钩子模板在取得/释放一个锁定之前和之后会被分别调用。

安装一个 post-lock和post-unlock钩子，在钩子中为锁定和解锁事件发送邮件，是个好主意 。有了这种脚本，你的所有用户都会在一个文件被锁定/解锁的时候被通知。在你的版本库文件夹下，你可以找到一个示例钩子脚本hooks/post-lock.tmpl。

你可能也使用hooks来拒绝破坏或是窃取锁定，或是限制只有管理员可以，或是当一个用户破坏或窃取锁定时通知原来的锁定拥有者。

更多内容请参考 第 3.3 节 “服务器端钩子脚本”

4.22. #创建并应用补丁

对开源工程(比如本工程)来说，每个人对仓库都有读访问权，并且任何人都可以对该工程做出修改。那么如何控制这些修改呢？如果任何人都可以提交自己的修改，那么这个工程可能永远都会处于不稳定状态，而且很有可能永远的瘫痪下去。在这种情况下，修改需要以补丁文件的形式先递交到有写访问权限的开发组。开发组可以先对该补丁文件进行审查，然后决定将其提交到仓库里或者是退还给作者。

补丁文件只是简单地用统一的差异描述文件显示出你的工作副本和基础版本的不同点。

4.22.1. #创建一个补丁文件

首先你需要做出修改并测试这个修改的内容。然后在父目录上使用TortoiseSVN → 创建补丁...代替TortoiseSVN → 提交...。

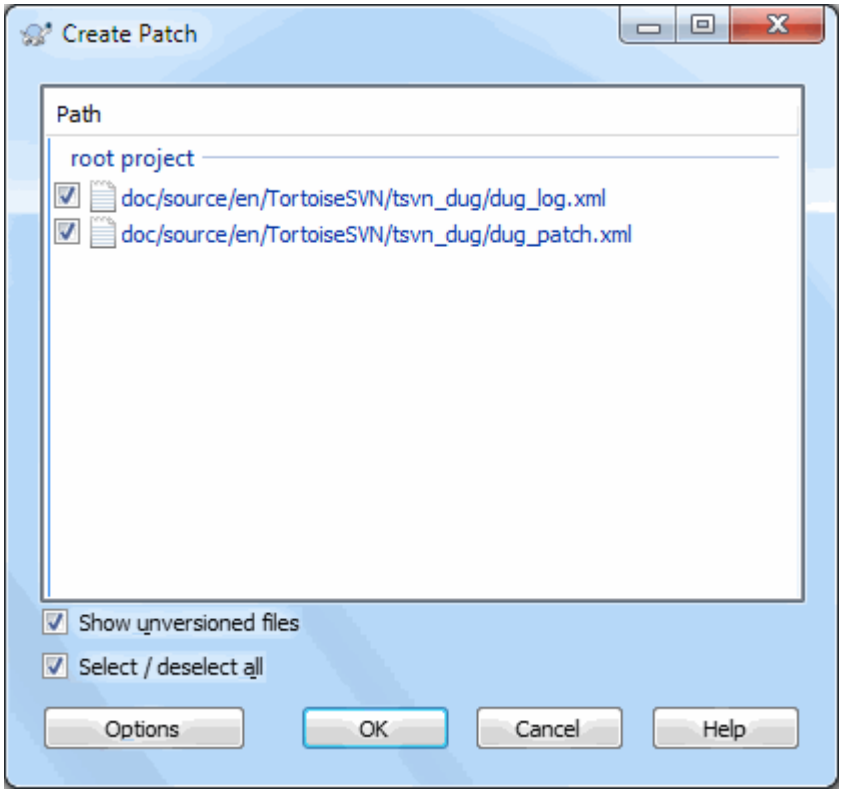


图 4.58. 创建补丁的对话框

现在你可以选择要包含在补丁中的文件了，就像你要做一个完整的提交一样。这样会产生一个单一的文件，该文件包括一份自从最后一次从仓库更新后你对所选择文件做的全部修改的摘要。

在这一对话框中，纵列和在 检查修改对话框中的纵列同样是可以定制的。更多细节请阅读第 4.7.4 节“本地与远程状态”

By clicking on the Options button you can specify how the patch is created. For example you can specify that changes in line endings or whitespaces are not included in the final patch file.

你可以创建包含对不同文件集合修改的相互独立的补丁。当然如果你创建了一个补丁文件，对于同一份文件的更多修改会创建另外一个补丁文件，第二份补丁文件包含了全部的修改。

你可以用一个自己选择的文件名来保存这个补丁文件，补丁文件可以有任意的扩展名，但是按人一般习惯，人们都是用.patch或者.diff作扩展名，你现在已经做好提交你的补丁文件的准备了。

你也可以将补丁保存到剪贴板，而不是文件。这样你可以粘贴到电子邮件中，让他人审核。或者你在机器上有两个工作副本，想将修改传递到另外的副本。在剪贴板上的补丁就是为这些操作提供便利。

If you prefer, you can create a patch file from within the Commit or Check for Modifications dialogs. Just select the files and use the context menu item to create a patch from those files. If you want to see the Options dialog you have to hold shift when you right click.

4.22.2.#应用一个补丁文件

补丁文件应用到你的工作副本。这应该从用来创建补丁的相同层级文件夹中完成。如果你不确定是什么文件夹，只需查看补丁文件中的第一行。例如，如果被执行的第一个文件是 doc/source/english/chapter1.xml 并且补丁文件的第一行是 Index: english/chapter1.xml 那么您需要应用此补丁至 doc/source/ 文件夹。然而，假如你在正确的工作副本中却选择了错误的层级文件夹，TortoiseSVN 会通知你并提示正确的文件夹。

为了给你的工作副本打补丁，你至少需要对代码库的读权限。因为合并程序必须要参考修订版本中其他开发人员做的修改。

从那个目录的右键菜单，点击TortoiseSVN → 应用补丁... 系统会弹出一个打开文件的对话框，让你选择要应用的补丁文件。默认情况下只显示.patch或者.diff文件，但是你可以选择"所有文件"。如果你以前将补丁保存到了剪贴板，可以使用打开文件对话框的从剪贴板打开...

如果补丁文件以.patch 或者 .diff 为扩展名的话，你可以选择直接右键点击该补丁文件，选择TortoiseSVN → 应用补丁... 在这种情况下，系统会提示你输入工作副本的位置。

这两种方法只是提供了做同一件事的不同方式。第一种方法，你先选择工作副本，然后浏览补丁文件。第二种方法，你先选择补丁文件，然后浏览工作副本。

一旦你选定了补丁文件和工作副本的位置，TortoiseMerge就会把布丁文件合并到你的工作副本中。系统会弹出一个小窗口列出所有被更改了的文件。依次双击每一个文件，检查所做的改变，然后保存合并后的文件。远程开发者的补丁现在已经应用到了你的工作副本上，你需要提交它以使每一个人都可以从代码库访问到这些修改。

远程开发者的补丁现在已经应用到了你的工作副本上，你需要提交它以使每一个人都可以从代码库访问到这些修改。

4.23.#谁修改了哪一行？

有时你不仅要知道哪一行做了修改，还要精确地知道谁修改了一个文件中的哪一行。这就是TortoiseSVN → 追溯...命令，有时候也叫做 评注 命令派上用场的时候了。

对一个文件中的每一行，这个命令列出了作者和该行修改时的版本。

4.23.1. #追溯文件

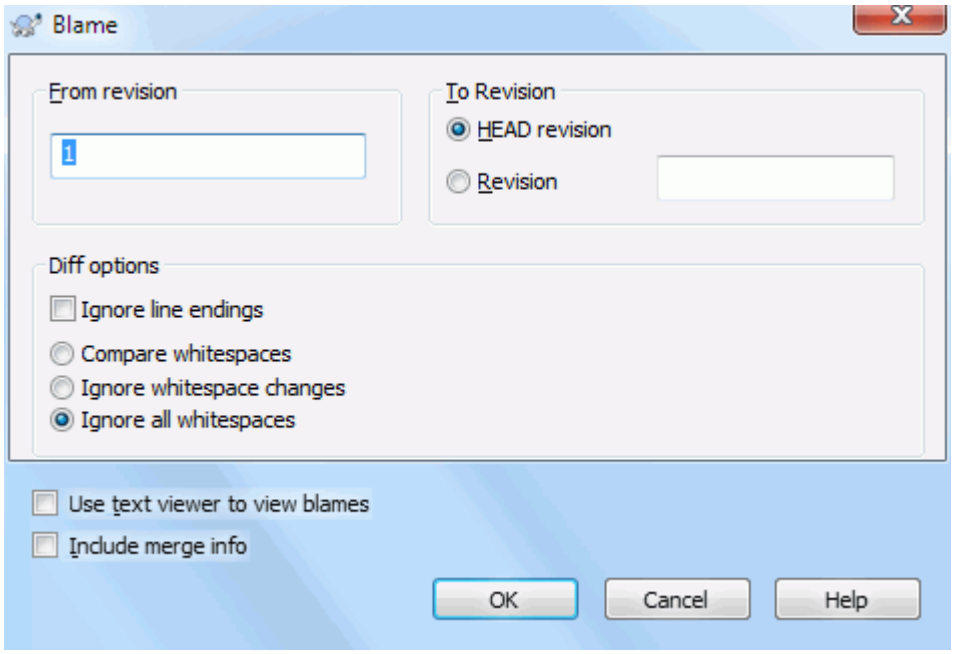


图 4.59. 评注/追溯对话框

如果对早期版本的修改不感兴趣，你可以设置从哪个版本开始追溯。如果你想追溯每一个版本，你可以把那个数值设置为1。

默认情况下追溯文件是使用 TortoiseBlame 来查看，突出显示不同的版本以使其更易于阅读。如果你想打印或者编辑追溯文件，选择 使用文本查看器查看追溯文件。

你可以指定处理行结束符和空白改变的方法。这些选项在 第 4.10.2 节 “行结束符和空白选项” 描述。默认是将行结束符和空白改变视为实际改变，但是如果你想忽略缩进改变和找到原始作者，那么可以在这里选择适当的处理方法。

You can include merge information as well if you wish, although this option can take considerably longer to retrieve from the server. When lines are merged from another source, the blame information shows the revision the change was made in the original source as well as the revision when it was merged into this file.

一旦你按了 OK 按钮，TortoiseSVN就开始从版本库获取数据创建追溯文件。注意：视修改的文件的多少和你的网络连接情况，可能会花掉几分钟到几十分钟不等。追溯过程完成后，其结果将会写到一个临时文件中，你可以读到这个结果文件。

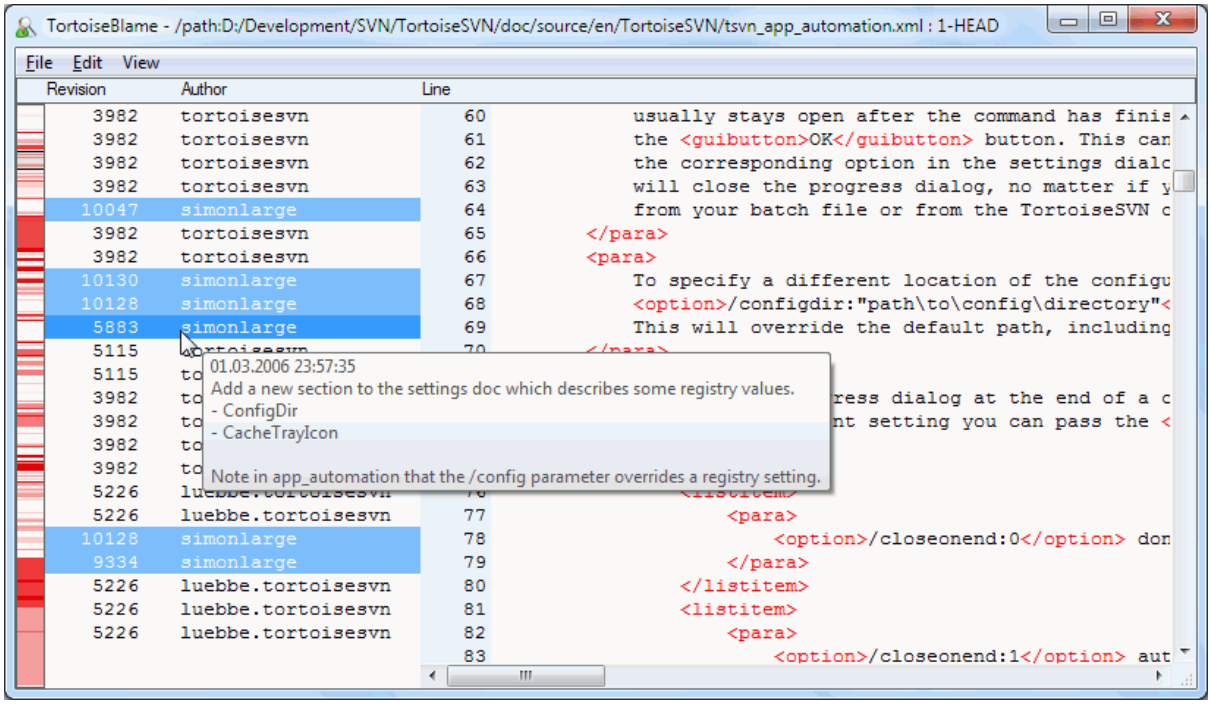


图 4.60. TortoiseBlame

TortoiseBlame,包含在TortoiseSVN中,使得追溯文件更加容易阅读。当你的鼠标在追溯信息列的某一行上盘旋时，所有和该列具有相同版本号的行都会用一个比较暗的背景色显示。同一作者修改的其他版本的行会用一个亮一些的背景色显示。如果你的显示设置为256色模式，那么颜色显示的可能不会很清楚。

如果在某一行上左击，具有同一版本号的所有行都会高亮显示，同一作者的其他版本的行会用一个更亮的颜色高亮显示。这个高亮具有粘性，也就是说允许你移动鼠标但原先高亮的地方仍然保持高亮。再次点击该版本将关闭高亮显示。

只要鼠标逗留在追溯信息列，版本注释(日志信息)就会作为提示出现。如果你想复制此版本的日志信息，使用在追溯信息列显示的右键菜单。

你可以在追溯报告里用编辑 → 查找...来搜索想要的内容。它允许你搜索版本号，作者还有文件的内容。这个搜索不包含日志信息－你可以在日志对话框里搜索日志信息。

你也可以使用 编辑→转到行 ... 来跳到指定行。

当鼠标滑过追溯信息列时，有个上下文菜单可以帮助你比较版本和检查历史，它用鼠标下方行的版本号作为参考版本。上下文菜单 → 追溯以前版本产生同一个文件的追溯报告，它使用以前的版本作为上限，给出了你看到最后修改之前的文件状态追溯报告。上下文菜单 → 显示修改启动差异察看器，显示在参考版本中的修改。上下文菜单→显示日志从参考版本开始显示版本日志。

如果您需要对最早和最新变化之处有一个更好的视觉指示，选择 View→Color age of lines。这将使用一个颜色渐变来显示新行为红色、旧行为蓝色。默认着色很浅，但你可以使用 TortoiseBlame 设置来改变它。

If you are using Merge Tracking and you requested merge info when starting the blame, merged lines are shown slightly differently. Where a line has changed as a result of merging from another path, TortoiseBlame will show the revision and author of the last change in the original file rather than the revision where the merge took place. These lines are indicated by showing the revision and author in italics. The revision where the merge took place is

shown separately in the tooltip when you hover the mouse over the blame info columns. If you do not want merged lines shown in this way, uncheck the Include merge info checkbox when starting the blame.

If you want to see the paths involved in the merge, select View → Merge paths. This shows the path where the line was last changed, excluding changes resulting from a merge.

The revision shown in the blame information represents the last revision where the content of that line changed. If the file was created by copying another file, then until you change a line, its blame revision will show the last change in the original source file, not the revision where the copy was made. This also applies to the paths shown with merge info. The path shows the repository location where the last change was made to that line.

TortoiseBlame 设置可以使用 TortoiseBlame 选项卡上的 TortoiseSVN → 设置... 来访问。请参阅第 4.30.9 节 “TortoiseBlame 的设置”。

4.23.2. #追溯不同点

One of the limitations of the Blame report is that it only shows the file as it was in a particular revision, and the last person to change each line. Sometimes you want to know what change was made, as well as who made it. If you right click on a line in TortoiseBlame you have a context menu item to show the changes made in that revision. But if you want to see the changes and the blame information simultaneously then you need a combination of the diff and blame reports.

在版本日志对话框里包含了以下几个选项支持你做这样的操作。

追溯版本
: 在顶部窗口，选择两个版本，然后选择上下文菜单 → 追溯版本。它将取出两个版本的追溯数据，然后使用差异察看器比较这两个追溯文件。

追溯修改
: 在上面的面板里选择一个版本，然后在下面的面板选择一个文件然后选择右键菜单 → 追溯改变。这会为选定版本及其上一个版本获取追溯数据，然后用差异阅读器比较两份追溯文件。

与工作副本比较并追溯
: 为一个单一文件显示日志，在上面的面板选择一个版本，然后选择上下文菜单 → 与工作副本比较并追溯。这会为文件的选择版本和工作副本获取追溯数据，然后使用差异阅读器比较两份追溯文件。

4.24. #版本库浏览器

有时候我们需要在版本库中直接进行操作，而不是在工作副本中。这就是我们的版本库浏览器可以做到的。正如资源管理器和能浏览你的工作副本一样，版本库浏览器允许你浏览版本库的结构和状态。

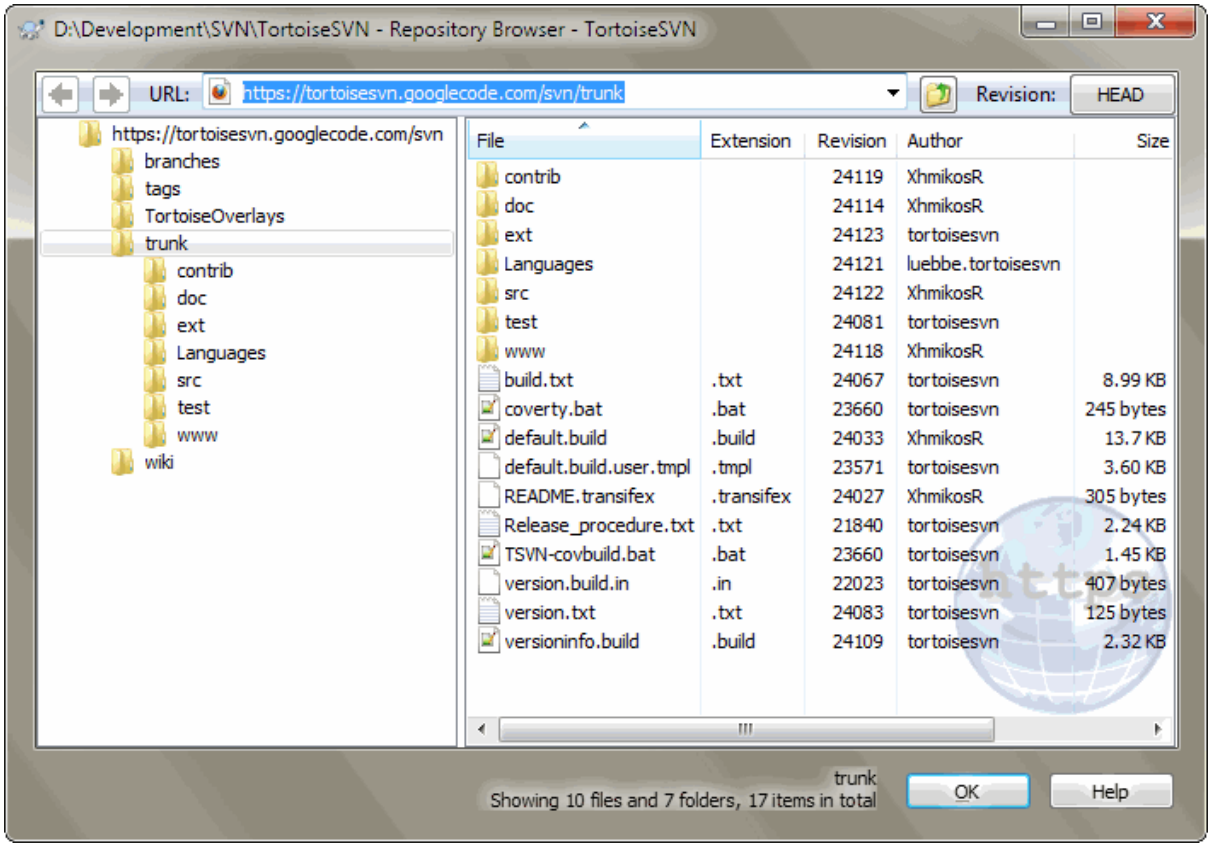


图 4.61. 版本库浏览器

在版本库浏览器中你可以执行比如复制，转移，重命名、、、直接操作在版本库上。

除了版本库浏览器不是显示计算机中的文件，而是显示版本库中特定版本的内容之外，它看起来很象 Windows 资源管理器。在左边的窗格显示目录树，右边的窗格显示选定目录的内容。在版本库浏览器窗口的顶部，你可以输入你想浏览的版本库 URL和版本。

在版本库浏览器中，含有 svn:externals 属性的文件夹也会被显示。这些文件夹上会有一个小箭头，以表明他们不是版本库结构的一部分，而只是链接。

就象 Windows 资源管理器一样，如果你想设置排列顺序，可以点击右边窗格中的列首。并且所有窗格都有上下文菜单。

文件的上下文菜单允许你：

- 用默认查看器或你指定的程序打开选中文件的选中版本。
- Edit the selected file. This will checkout a temporary working copy and start the default editor for that file type. When you close the editor program, if changes were saved then a commit dialog appears, allowing you to enter a comment and commit the change.
- 显示此文件的版本日志，或者显示所有版本图，从而你可以知道其来源。
- 追溯这个文件，查看是谁在何时修改哪些行。
- Checkout a single file. This creates a “sparse” working copy which contains just this one file.
- 删除或改名文件。
- 在你的硬盘上保存此文件的一个未版本控制的副本。

- Copy the URL shown in the address bar to the clipboard.
- 复制这个文件，目的可以是版本库中的其它地方，也可以是同一版本库中的工作副本。
- 查看/编辑文件属性。
- Create a shortcut so that you can quickly start repo browser again, opened directly at this location.

目录的上下文菜单允许你：

- 显示此目录的版本日志，或者显示所有版本图，从而你可以知道其来源。
- 导出此目录的未版本控制副本到你的本地硬盘。
- 检出此目录到你的硬盘，产生一个本地工作副本。
- 在版本库创建新目录。
- Add unversioned files or folders directly to the repository. This is effectively the Subversion Import operation.
- 删除或改名文件夹。
- 创建一个文件夹的副本至版本库中的不同部分或根于同一版本库中的工作副本。这也可以用来创建一个分支/标记，而不需要检出工作副本。
- 察看/编辑目录的属性
- 为比较标记目录。已标记的目录用粗体显示。
- 将此文件夹与以前标记的文件夹比较，用统一差异，或者是一个可以用默认比较工具可视化显示差异的文件改变列表。它对于比较两个标签，或者最新版本与分支，查看修改详情时尤其有用。

如果你在右窗格选择两个文件夹，你可以用统一差异，或者是一个可以用默认比较工具可视化显示差异的文件改变列表来查看其区别。

如果你在右边窗格中选择了多个目录，你可以将它们同时检出到同一个父目录之下。

如果你选择两个拥有相同历史的标签(特别是/主干/)，你可以使用右键菜单 → 显示日志...来查看

External items (referenced using svn:externals are also shown in the repository browser, and you can even drill down into the folder contents. External items are marked with a red arrow over the item.

您通常可以使用 F5 来刷新视图。这将刷新当前的所有显示。如果你想预取或刷新还未被打开的节点信息，使用 CTRL-F5。之后，扩展的任何节点将会立即出现而没有网络延迟，同时提取信息。

您还可以使用版本库浏览器进行拖放操作。如果您从资源管理器中拖动一个文件夹至版本库浏览器，它将被导入至版本库。请注意，如果你拖动多个项目，他们将以单独的提交导入。

如果你想在版本库内移动一个项目，只需 左键拖拽 到新的位置。如果你想创建一个拷贝而不是移动项目，用 CTRL-左键拖拽 代替。拷贝时，光标上有一个 “+” 符号，就像在资源管理器中那样。

如果你想复制/移动文件或文件夹到另一个位置，并同时命一个新名称，你可以用 右键拖拽 或 CTRL-右键拖拽 而不是使用 左键拖拽。在这种情况下，将会显示一个重命名对话框，您可以为文件或文件夹输入一个新的名称。

无论什么时候对版本库的任意操作，都要求加入它的操作日志。这为你操作失误提供了返回的机会。

有时候当你访问一个路径时，会跳出一个错误信息。也许是不可用的URL，也许是你没有访问权限，或者其他的一些原因，如果你要把这些信息发送出去，只用点击它然后右键 → 复制错误信息到剪贴板，或者简单地用CTRL+C。

4.25. #版本分支图

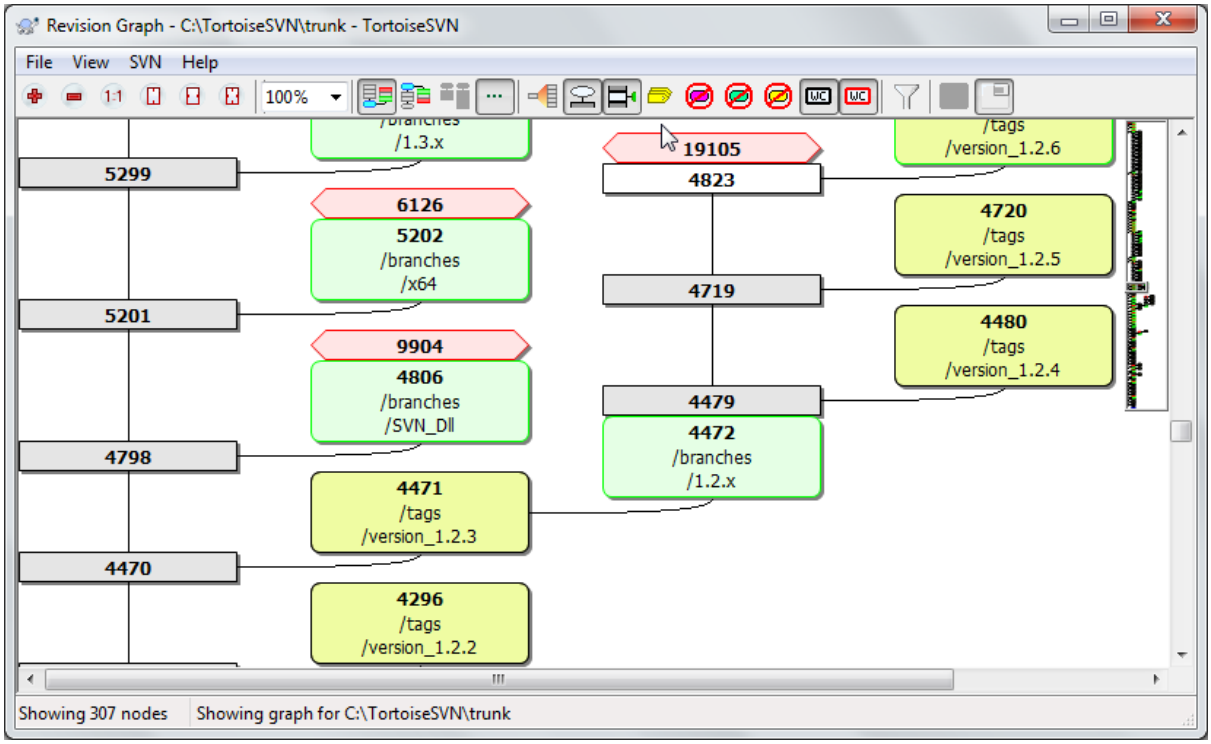


图 4.62. 一个版本分支

有时候，我们需要知道从哪开始有了分支和标签，同时想知道这条支路是单独的分支还是树型结构。如果需要你可以使用TortoiseSVN → 版本分支图...。

这个版本历史分析图能够显示分支/标签从什么地方开始创建，以及什么时候删除。

为了生成图示，TortoiseSVN 必须从版本库根目录中提取所有的日志信息。无需言明，对于拥有几千个修订版本的版本库来说得需要花上数分钟时间，这取决于服务器速度，网络带宽等。如果您尝试对像超过有 500,000 个修改版本的 Apache 之类的项目来进行此操作，您可能得等待一些时候。

好消息是，如果你正使用日志缓存，你只需遇到一次这种延迟。之后，日志数据被保存到本地。日志缓存在 TortoiseSVN 设置中可被启用。

4.25.1. #版本图节点

每个版本图节点代表版本库中的一个修订版本，在那儿你可以查看到树状的变化。不同类型的节点可通过形状和颜色来区分。形状是固定的，但颜色可以使用 TortoiseSVN → 设置

新增或复制的项目
: 已添加或通过复制另一个文件/文件夹而创建的项目使用一个圆角矩形来显示。默认颜色为绿色。
标签和主干都被视为一个特殊的情况并使用不同的色调，这取决于 TortoiseSVN → 设置。

已删除的节点
: 已删除的项目，例如一个不再需要的分支，显示为一个八边形（切角的矩形）。默认颜色为红色。

重命名的项目
: 重命名的项目也使用一个八边形，但默认颜色是蓝色的。

分支最新版本
: 图示通常只限于显示分支点，但通常也可用于查看每个分支各自的 HEAD 版本。如果您选择 显示 HEAD 版本，每个 HEAD 版本节点将显示为一个椭圆形。请注意，这里的 HEAD 是指在本路径上所提交的最新版本，而不是版本库的 HEAD 版本。

当前工作副本的版本
: 如果您从工作副本中调用版本图，你可以使用 显示工作副本的版本号 在图示上选择显示当前版本号，这使用了一个粗体边框来标记当前版本节点。

已修改的工作副本
: 如果您从工作副本中调用版本图，你可以使用 显示工作副本的本地更新 选择显示附加节点来表明你修改过的工作副本，附加节点默认有一个红色的粗体边框。

一般的文件/文件夹
: 其他一般的文件用plain rectangle标示。

来设置。

注意默认情况下图示只显示被添加、复制或删除的项目点。非常状况下显示每一个项目版本将生成一个非常巨大的图示。如果你真的想看到 所有 发生更改的版本，在 查看 菜单和工具栏上的某个选项可以满足您的要求。

默认视图 (分组关闭) 的节点置放主要是让其垂直位置按照严格的版本顺序排列，所以你会对已完成项目的顺序产生一个视觉提示。同列中两个节点的顺序是非常明显的。当两个节点处于相邻列时偏移量会更小，因为不必防止节点相互重叠，因此顺序并不太明显。这种优化是必要的，可以将复杂的图示保持到一个合理的规模。请注意，这种排序使用了 较老 的一边的节点 边缘 作为参照，也就是当用底部最老的节点显示图示时的节点底部边缘。这种参考的边缘是很显著的，因为节点的形状并不都是相同的高度。

4.25.2.#更改视图

因为版本图经常很复杂，所以有很多方法剪裁视图。你可以在视图菜单或工具栏找到它们。

分组分支
: 默认行为 (分组关闭) 让所有的行严格按照版本号进行排序。结果，只有稀少的提交而有较长寿命的分支仅仅为少数的变化而占据一整列，并且版本图变得非常宽广。

  This mode groups changes by branch, so that there is no global revision ordering: Consecutive revisions on a branch will be shown in (often) consecutive lines. Sub-branches, however, are arranged in such a way that later branches will be shown in the same column above earlier branches to keep the graph slim. As a result, a given row may contain changes from different revisions.

最老的在顶部
: 正常情况下，在图形底部显示最老的版本，版本树向上生长。使用这个选项，版本树从顶部向下生长。

顶部对齐树
: 当一个版本图形被拆分成几个较小的树结构时，这些树结构既可能按自然的修订版本顺序显示，也可能对齐于窗口的底部，这取决于您是否使用了 Group Branches 选项。使用该选项可让所有树结构自顶向下延伸。

减少交叉行
: This option is normally enabled and avoids showing the graph with a lot of confused crossing lines. However this may also make the layout columns appear in less logical places, for example in a diagonal line rather than a column, and the graph may require a larger area to draw. If this is a problem you can disable the option from the View menu.

差异路径名称
: 长路径名称会占据大量空间并且让节点框变得非常大。使用此选项可以只显示路径的变化部分，而用小数点来代替共同部分。例如，如果您从 /trunk/doc/html 创建了一个分支 /branches/1.2.x/doc/html，可以用简洁形式将该分支显示为 /branches/1.2.x/..，因为最后两级 doc 和 html 并无变化。

显示所有版本
: 这会如你所愿并显示每一个发生更改(您正在绘制的树结构中)的修订版本。如果历史版本繁长，这可能会生成一个相当巨大的图形。

显示最新版本
: 它确保每个分支的最新版本永远在图中显示。

精确复制源
: 当创建一个分支/标签时，默认行为是将分支显示为取自最后一个发生更改的节点。严格地讲，这并不准确。因为分支往往创建自当前 HEAD 而非某个特定的修订版本。因此，有可能显示更为正确的(但用处不大)用来创建副本的修订版本。请注意，此修订版本可能比源分支的 HEAD 版本更新。

折叠标签
: When a project has many tags, showing every tag as a separate node on the graph takes a lot of space and obscures the more interesting development branch structure. At the same time you may need to be able to access the tag content easily so that you can compare revisions. This option hides the nodes for tags and shows them instead in the tooltip for the node that they were copied from. A tag icon on the right side of the source node indicates that tags were made. This greatly simplifies the view.

  Note that if a tag is itself used as the source for a copy, perhaps a new branch based on a tag, then that tag will be shown as a separate node rather than folded.

隐藏已删除的路径
: 隐藏掉已经不在最新版本的路径，例如已经删除的分支。

  If you have selected the Fold tags option then a deleted branch from which tags were taken will still be shown, otherwise the tags would disappear too. The last revision that was tagged will be shown in the colour used for deleted nodes instead of showing a separate deletion revision.

  If you select the Hide tags option then these branches will disappear again as they are not needed to show the tags.

Hide unused branches
: 隐藏那些没有更改提交到相应文件或子文件夹中的分支。这并不一定表示该分支不被使用，仅表示它的这部分没有发生更改。

显示最新版本
: 在图形上标记修订版本，其对应于您抓取图形而得到的该项目的更新修订版本。如果您刚刚更新，这将会是 HEAD。但如果有其他人自您上次更新后提交了更改，您的工作副本可能会处在一些修订版本之后。该节点用粗边框来标记。

显示工作拷贝的修改
: 如果您的工作副本包含本地更改，此选项会将它绘制为单独的椭圆形节点，其被链接回您的工作副本最后更新至的节点。默认的边框颜色为红色。您可能需要使用 F5 刷新图形来捕获最近的更改。

过滤器
: 有时修订版本图形显示的修订版本比您想看的还要多。此选项打开一个对话框，可允许您限制修订版本的显示范围，并通过名称隐藏特定的路径。

  If you hide a particular path and that node has child nodes, the children will be shown as a separate tree. If you want to hide all children as well, use the Remove the whole subtree(s) checkbox.

修剪树
: 当版本图中包含不同的版本树时，在背景上用不同的颜色区分是很有用的。

显示概述
: 将整个图形显示为小图片，当前视图窗口如同可以拖动的矩形。这可以使您更轻松地导航图形。请注意，对于相当大的图形，概览可能会由于极端的缩放系数而变得无用，并且在这种情况下会因此不被显示。

4.25.3. #使用图

概览窗口可以使遍历大图更容易。它在小窗口中显示整个图形，高亮显示当前详细显示的部分。你可以通过拖曳高亮区域来改变当前详细显示的部分。

任何时候只要鼠标在版本框上滑过，都会显示该版本的日期、作者和备注信息。

选择两个版本(用ctrl+左击)，你可以在右键菜单下查看这两个版本的差异。你可以在分支的起始点查看差异，但一般你会在一个分支的结束点来查看，比如，在最新版本。

你可以用查看单一差异文件的方式来查看差异，它在单一的文件中显示差异。如果你选右键菜单 → 比较版本所有修改过的文件都会呈现在列表中。双击一个文件名可以返回文件版本号和他们差异。

如果右击一个版本你可以使用右键菜单 → 显示日志 来查看它的历史。

您也可以将所选修订版本的更改合并到不同的工作副本。可以通过文件夹选择对话框来确定合并到哪一个工作副本中，但是此操作没有确认对话框，也没有尝试测试合并的机会。合并到未修改的工作副本是一个好主意，这样当其行不通时您还可以恢复这些更改！当您想要将所选版本从一个分支合并到另一个分支时，这个功能很有用。

学习怎样看版本关系图

初次使用的用户可能会被修订版本图形不符合用户心理模式的显示所惊倒。比如，如果一个修订版本更改了一个文件或文件夹的多个副本或分支，然后这单个修订版本将会有多个节点。建议您从工具栏最左边的选项开始并一步步自定义图形直到它慢慢接近您的心理模式为止。

所有过滤选项都会尝试尽可能漏掉小信息。这可能会导致某些节点更改它们的颜色等。只要结果不符合预期，就应撤销上次过滤操作并尝试理解特定修订版本或分支的特别之处。在大多数情况下，过滤操作的最初预期结果既不准确也可能误导。

4.25.4. #刷新视图

如果你想再次检查服务器是否有新的消息，可以使用 F5 键来刷新视图。如果使用日志缓存(默认启用)，这将会检查版本库是否有新的提交并且只获取新的。如果日志缓存处于离线模式，这将会尝试回到在线模式。

如果您正在使用日志缓存并且认为消息内容或作者可能已更改，您应该使用日志对话框来刷新所需消息。由于修订版本图形起于版本库根目录，我们将不得不使整个日志缓存失效，而重新填满它可能会花上非常长的时间。

4.25.5. #修剪树结构

一个大的树结构可能难以导航，有时您会想隐藏其中一部分，或者把它拆分成更小的树结构集合。如果您将鼠标悬停于节点链接进入或离开节点的地方，您将会看到一个或更多的弹出式按钮来允许您这样做。

在减号上单击收起关联子树。

点击加号按钮来展开折叠的树结构。当一个树结构已经折叠时，此按钮将保持可见以表明已隐藏的子树。

点击十字按钮来拆开所连接的子树，并在图形上将其显示为单独的树。

点击圆形按钮以重新连接一个拆开的树。当一个树已被拆离时，此按钮将保持可见以表明有一个单独的子树。

点击图形背景以获取主上下文菜单，它可提供选项来 全部展开 和 全部拼合。如果没有分支被折叠或拆开，该上下文菜单将不会被显示。

4.26. #导出一个Subversion工作副本

Sometimes you may want a clean copy of your working tree without the .svn directory, e.g. to create a zipped tarball of your source, or to export to a web server. Instead of making a copy and then deleting the .svn directory manually, TortoiseSVN offers the command TortoiseSVN → Export.... Exporting from a URL and exporting from a working copy are treated slightly differently.

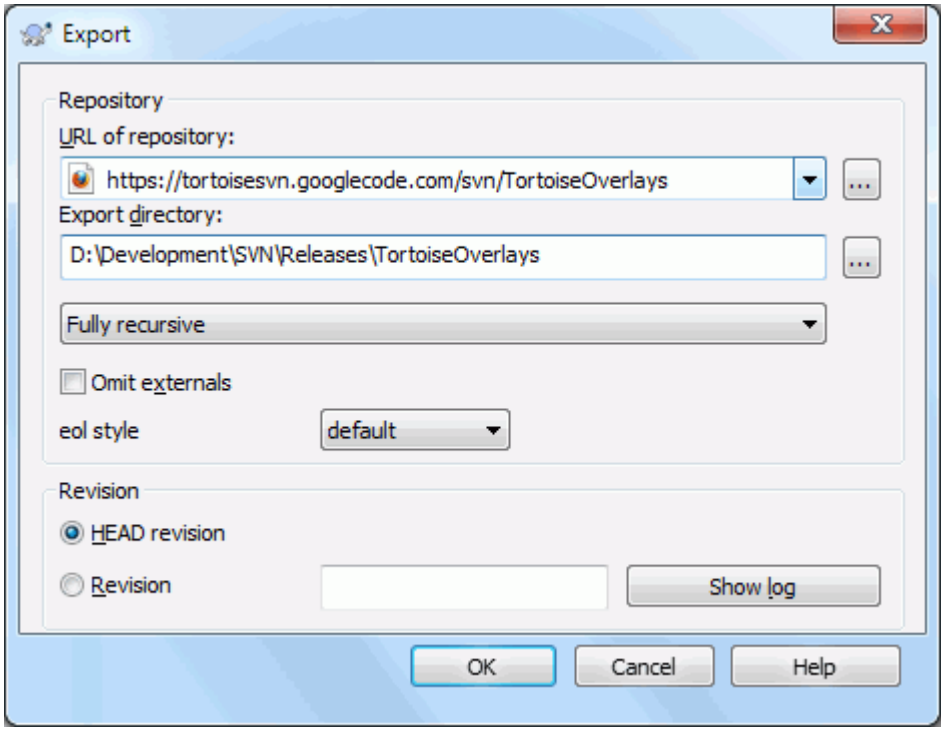


图 4.63. 从 URL 导出对话框

如果你在未版本控制的目录执行此命令，TortoiseSVN会假定此目录是目标，弹出对话框让你输入要导出的URL和版本。这个对话框有只导出顶级目录，省略外部引用，以及不管svn:eol-style的取值，重新设置行结束样式等选项。

当然，你也可以直接从版本库导出。使用版本库浏览器浏览有关子树，然后使用右键菜单 → 导出。就会出现上面所说的从URL导出对话框。

If you execute this command on your working copy you'll be asked for a place to save the clean working copy without the .svn folder. By default, only the versioned files are exported, but you can use the Export unversioned files too checkbox to include any other unversioned files which exist in your WC and not in the repository. External references using svn:externals can be omitted if required.

Another way to export from a working copy is to right drag the working copy folder to another location and choose Context Menu → SVN Export versioned items here or Context Menu → SVN Export all items here or Context Menu → SVN Export changed items here. The second option includes the unversioned files as well. The third option exports only modified items, but maintains the folder structure.

当从一个工作副本导出时，如果目标文件夹已包含了与您所导出的具有相同名称的文件夹，将为您提供此选项来重写现有内容，或者创建一个自动生成名称的新文件夹，例如 目标 (1)。

导出单个的文件

导出对话框不允许展出一个单独的文件，尽管 Subversion 可以

要用 TortoiseSVN 导出单个文件，您必须使用版本库浏览器 (第 4.24 节 “版本库浏览器”)。简单地将您想从版本库浏览器中导出的文件拖动到想要存放至的资源管理器位置，或者使用版本库浏览器的上下文菜单来导出文件。

导出更改树

如果您想导出项目树结构的副本，但只包含在某特定修订版本中或任意两个修订版本之间发生更改的文件，请使用在 第 4.10.3 节 “比较文件夹”中所描述的版本比较功能。

If you want to export your working copy tree structure but containing only the files which are locally modified, refer to SVN Export changed items here above.

4.26.1. #从版本控制里移除删除工作副本

Sometimes you have a working copy which you want to convert back to a normal folder without the .svn directory. All you need to do is delete the .svn directory from the working copy root.

Alternatively you can export the folder to itself. In Windows Explorer right drag the working copy root folder from the file pane onto itself in the folder pane. TortoiseSVN detects this special case and asks if you want to make the working copy unversioned. If you answer yes the control directory will be removed and you will have a plain, unversioned directory tree.

4.27. #重新定位工作副本

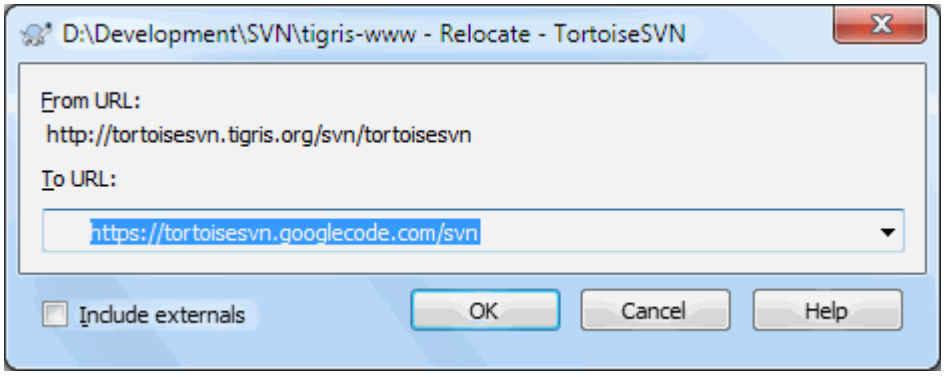


图 4.64. 重定位对话框

If your repository has for some reason changed it's location (IP/URL). Maybe you're even stuck and can't commit and you don't want to checkout your working copy again from the new location and to move all your changed data back into the new working copy, TortoiseSVN → Relocate is the command you are looking for. It basically does very little: it rewrites all URLs that are associated with each file and folder with the new URL.

> 此操作仅适用于工作副本的根目录。所以此上下文菜单项仅显示于工作副本根目录。

您可能会惊讶地发现 TortoiseSVN 会将版本库牵扯到此操作中来。这样做完全是为了执行一些简单的检查以确保新地址确实不会将相同的版本库当作现有的工作副本。

这是一个极少使用的操作.重定位只能 在版本库路径更改时使用。可能的原因是:

- 服务器的IP地址已更改。
- 协议已更改(比如从http://改为 https://)。
- 版本库在服务器的路径已更改。

换种说法，如果你要重定位，只有当你的工作副本仍然还在同一个版本库中定位，但版本库本身已经没有了。

以下情况不支持:

- 你要移到一个完全不同的版本库。这种情况下，你必须从新的版本库里执行一次干净的检出。
- 你要在同一个版本库中切换一个分支或目录。这么做你可以用TortoiseSVN → 切换....

如果你使用以上任意一种重定位方式，它将破坏你的工作副本，在你更新、提交等操作时会提示一些令人费解的错误信息。一旦发生这种情况，唯一的办法就是检出最新版本。

4.28. #与 BUG 跟踪系统/问题跟踪集成

在软件开发中，修改依赖于一个bug或问题编号是很常见的。bug跟踪系统的用户(问题跟踪者)喜欢在问题跟踪中将Subversion的修改与一个指定编号联系起来。因此很多问题跟踪者提供了一个预提交钩子脚本，分析日志，查找提交相关的bug编号。这稍微有些不可靠，因为它依赖于用户写完全的日志，预提交钩子才能正确分析。

TortoiseSVN可以在两个方面帮助用户:

1. 当用户输入日志信息时，一个定义良好，包含问题编号，与此提交相关的的行，会自动增加。这样减少了用户输入的问题编号不能比bug跟踪系统正确分析的风险。

   或者TortoiseSVN高亮显示日志消息中能被问题跟踪者识别的部分。这样，用户就知道日志消息能被正确解析。

2. 当用户浏览日志信息，TortoiseSVN在日志信息中创建指向每个bug标示的链接，它可以用浏览器打开。

4.28.1. #在日志消息中增加问题号

您可以在 TortoiseSVN 中为您的选择集成一个错误跟踪工具。为此，您必须定义一些属性，用 bugtraq: 开头。它们必须设置于文件夹: (第 4.17 节 “项目设置”)

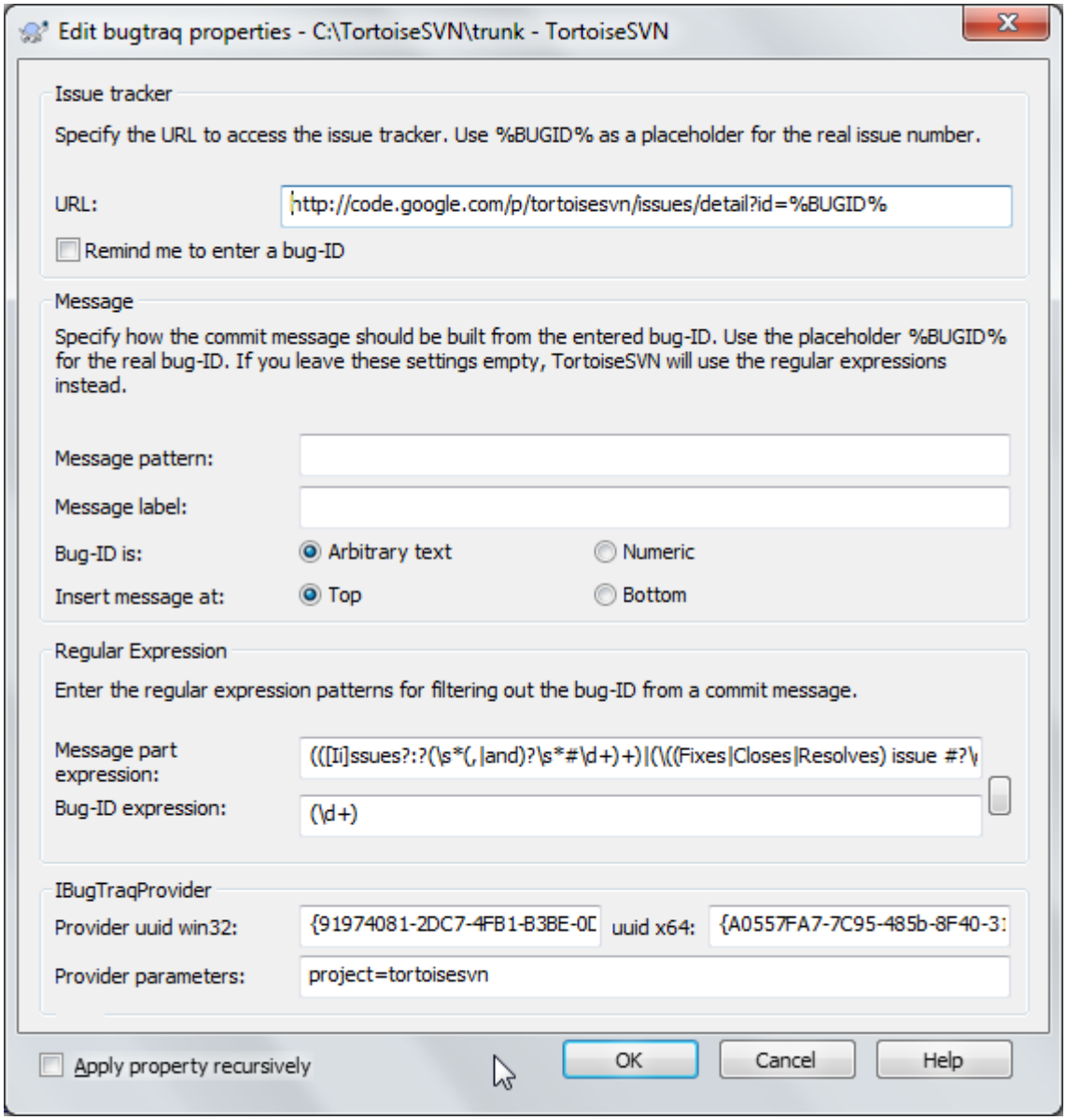


图 4.65. The Bugtraq Properties Dialog

When you edit any of the bugtraq properties a special property editor is used to make it easier to set appropriate values.

有两种方式将问题跟踪器集成于 TortoiseSVN。一种基于简单的字符串，另一种基于 正则表达式。两种方法使用的属性为:

bugtraq:url
将这个属性设置为你的bug跟踪工具的地址。它必须编码并且包含<literal>%BUGID%</literal>。<literal>%BUGID%</literal>用你输入的问题编号替换。它允许TortoiseSVN 在日志对话框中显示链接，于是你可以在察看版本日志时直接进入bug跟踪工具。你可以不提供这个属性，但是这样TortoiseSVN就不能显示链接了，只能显示问题编号。例如TortoiseSVN 使用<literal>http://issues.tortoisesvn.net/?do=details&id=%BUGID%</literal>。

您也可以使用相对地址来代替绝对地址。当您的问题跟踪器和源版本库在同一域/服务器上时，这非常有用。即使在域名称发生更改的情况下，您也不必调整 bugtraq:url 属性。有两种方式来指定相对地址:

如果它以字符串 ^/ 开头，它被假定为对应于版本库的根目录。例如，如果您的版本库位于 http://tortoisesvn.net/svn/trunk/，^/../?do=details&id=%BUGID% 将解析为 http://tortoisesvn.net/?do=details&id=%BUGID%。

以字符串 / 开头的地址被假定为相对于服务器的主机名。例如，如果您的版本库位于 http://tortoisesvn.net 的任何地方，/?do=details&id=%BUGID% 将解析为 http://tortoisesvn.net/?do=details&id=%BUGID%。

bugtraq:warnifnoissue
如果你想TortoiseSVN给出空问题编号的警告，就设置为 <literal>真</literal>。有效取值是<literal>真/假</literal>。<emphasis>如果没有定义，那么假定为 <literal>假</literal></emphasis>。

4.28.1.1. #文本框中的问题单编号

在最简单的方法里，TortoiseSVN为用户显示了一个单独的bug ID输入字段，然后后面预计会追加一个用户输入日志信息的行。

bugtraq:message
此属性以输入字段模式激活问题跟踪系统。如果设置了此属性，在提交更改时，TortoiseSVN 将提示您输入问题单编号。它通常会在日志信息后面添加一行。必须包含 %BUGID%，在提交时会被替换为问题单编号。这确保了您的提交日志包含了对总是保持一致格式并且可被问题跟踪工具解析的问题单编号的引用，从而将问题与特定提交相关联。例如，您可能使用 Issue : %BUGID%，但是这取决于您的工具。

bugtraq:label
是TortoiseSVN的提交对话框中用来输入问题单号码的输入项，如果没有设置，将会显示<literal>Bug-ID / Issue-Nr:</literal>，要记住窗口不会为适应标签而改变大小，所以请保持标签的小于20-25个字符。

bugtraq:number
如果设置为 true，在问题单编号文本字段中只允许输入数字。逗号除外，因此您可以使用逗号来分隔多个数字。有效的值为 true/false。如果未定义，假定为 true。

bugtraq:append
这个属性定义了bug-ID。是追加到(true)日志信息的末尾，还是插入到(false)日志信息的开始。有效的值包括true/false，如果没有定义，默认是true ，所以现存的项目不会被打破。

4.28.1.2. #问题号使用正则表达式

在含正则表达式的方法中，TortoiseSVN 不会显示一个单独的输入字段，而是标记用户所输入的日志信息的一部分，这可以被问题追踪工具所识别。这是在用户编写日志信息的时候完成的，这也意味着 bug ID 可以出现在日志信息的任何位置！这种方法更为灵活，也是 TortoiseSVN 项目本身使用的方法。

bugtraq:logregex
此属性以正则表达式模式激活问题追踪系统。它既可包含单个正则表达式，也可包含以新行来分隔的两个正则表达式。

如果设置了两个表达式，那么第一个表达式用作为一个预过滤器来查找包含 bug ID 的表达式。然后第二个表达式从第一个正则表达式的结果中提取显现出的 bug ID。这允许您使用 bug ID 列表和自然语言表达式，如果您愿意的话。例如，您可能修复几个 bug 并包括像这样的字符串: “This change resolves issues #23, #24 and #25”。

如果您想像上述日志信息中的表达式那样捕获 bug ID，您可以使用下面的 TortoiseSVN 项目所使用的正则表达式字符串: [Ii]ssues?:?(\s*(,|and)?\s*#\d+)+ 和 (\d+)。

第一个表达式从日志消息中筛选出 “issues #23, #24 and #25”。第二个正则表达式从第一个正则表达式的输出中提取纯粹的十进制数， 因此它将返回 “23”, “24” 和 “25” 并用作为 bug ID。

稍微分解一下第一个正则表达式，它必须以 “issue” 开头，可以大写。其后也可以跟一个 “s”(多个问题) 和一个冒号。这之后跟一个或多个组，每个都有零个或多个前导空格、可选的逗号或 “and” 以及更多的可选空格。最后强制使用 “#” 和十进制数。

如果只设置了一个表达式，则出现的 bug ID 必须按正则表达式字符串组匹配。例如: [Ii]ssue(?:s)? #?(\d+) 此方法是一些问题跟踪器所必需的，比如 trac，但它构建正则表达式会更难。我们建议您仅使用此方法，如果您的问题跟踪器文档告诉你如此。

<ulink url="http://www.regular-expressions.info/"><citetitle>http://www.regular-expressions.info/</citetitle></ulink>上的在线文档和教程。 如果你不熟悉正则表达式，可以看一下 http://en.wikipedia.org/wiki/Regular_expression, 和在线教程 http://www.regular-expressions.info/.

It's not always easy to get the regex right so to help out there is a test dialog built into the bugtraq properties dialog. Click on the button to the right of the edit boxes to bring it up. Here you can enter some test text, and change each regex to see the results. If the regex is invalid the edit box background changes to red.

如果同时设置了bugtraq:message和bugtraq:logregex属性，日志正则表达式会优先使用。

即使你的问题追踪工具没有pre-commit钩子来解析日志信息，你仍然可以使用这个功能将日志信息中的问题单转化为链接！

即使您不需要链接，问题单编号会在日志对话框中显示为一个单独的列，可更容易找到相对于某特定问题单的变动。

一些 tsvn: 属性需要 true/false 值。TortoiseSVN 也理解 yes 是 true 的同义词，no 是 false 的同义词。

设置文件夹的属性

为了系统能够工作，这个属性必须设置到文件夹上。当您提交文件或文件夹时，属性会从文件夹上读取。如果没有发现属性，TortoiseSVN 会向上级搜索整个目录树，直到一个没有版本控制的文件夹或发现根目录(例如C:\)为止。如果您能够确定每个用户只从 trunk/ 检出，而不是一些子目录，那么您只在 trunk/ 上设置属性就足够了。如果不能确定，您必须为每个子目录设置这些属性。在较深的项目层次结构中的属性设置将重写较高级别上的设置(更靠近 trunk/ 的)。

对于tsvn:属性，例如 tsvn:, bugtraq: 和 webviewer:，你只能对于所有子文件夹使用递归复选框设置属性，不用将这些属性设置在文件上。

当你使用TortoiseSVN在工作副本中新建一个子目录， 上层目录的所有项目属性都会自动的被继承。.

没有来自版本库浏览器的问题跟踪器信息

Because the issue tracker integration depends upon accessing Subversion properties, you will only see the results when using a checked out working copy. Fetching properties remotely is a slow operation, so you will not see this feature in action from the repo browser unless you started the repo browser from your working copy. If you started the repo browser by entering the URL of the repository you won't see this feature.

出于同样的原因，当使用版本库浏览器添加子文件夹时，项目属性将不会被自动传递到子文件夹。

This issue tracker integration is not restricted to TortoiseSVN; it can be used with any Subversion client. For more information, read the full Issue Tracker Integration

Specification [http://tortoisesvn.googlecode.com/svn/trunk/doc/notes/issuetrackers.txt] in the TortoiseSVN source repository. (第 3 节 “许可协议”explains how to access the repository.)

4.28.2. #从问题跟踪器中获取信息

前一节介绍了在日志信息中添加问题单信息，那如何从问题跟踪器中获取信息呢？提交对话框有一个 COM 接口，可允许集成一个能够跟跟踪器交互的外部程序。通常，您可能想查询跟踪器以得到分配给您的开放问题列表，以便于您能选取提交中正在解决的问题。

任何此类接口当然都是高度针对于您的问题跟踪系统，所以我们无法提供这部分，描述如何创建这样的程序已超出了本手册的范围。C# 和 C++/ATL 接口定义和示例插件可以从TortoiseSVN 版本库 [http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins] 中的 contrib 目录中获得。(第 3 节 “许可协议”解释了如何访问版本库。) 第 6 章BugtraqProvider 接口 中也给出了 API 摘要。另一个 C# 示例插件(开发中)是 Gurtle [http://code.google.com/p/gurtle/]，它实现了可与 Google Code [http://code.google.com/hosting/] 问题跟踪器交互的所需 COM 接口。

为便于说明，假设您的系统管理员已向您提供了已安装的问题跟踪器插件，并且您已让一些工作副本使用了 TortoiseSVN 设置对话框中的插件。当您从已分配了插件的工作副本中打开提交对话框时，您会在对话框顶部看到一个新的按钮。

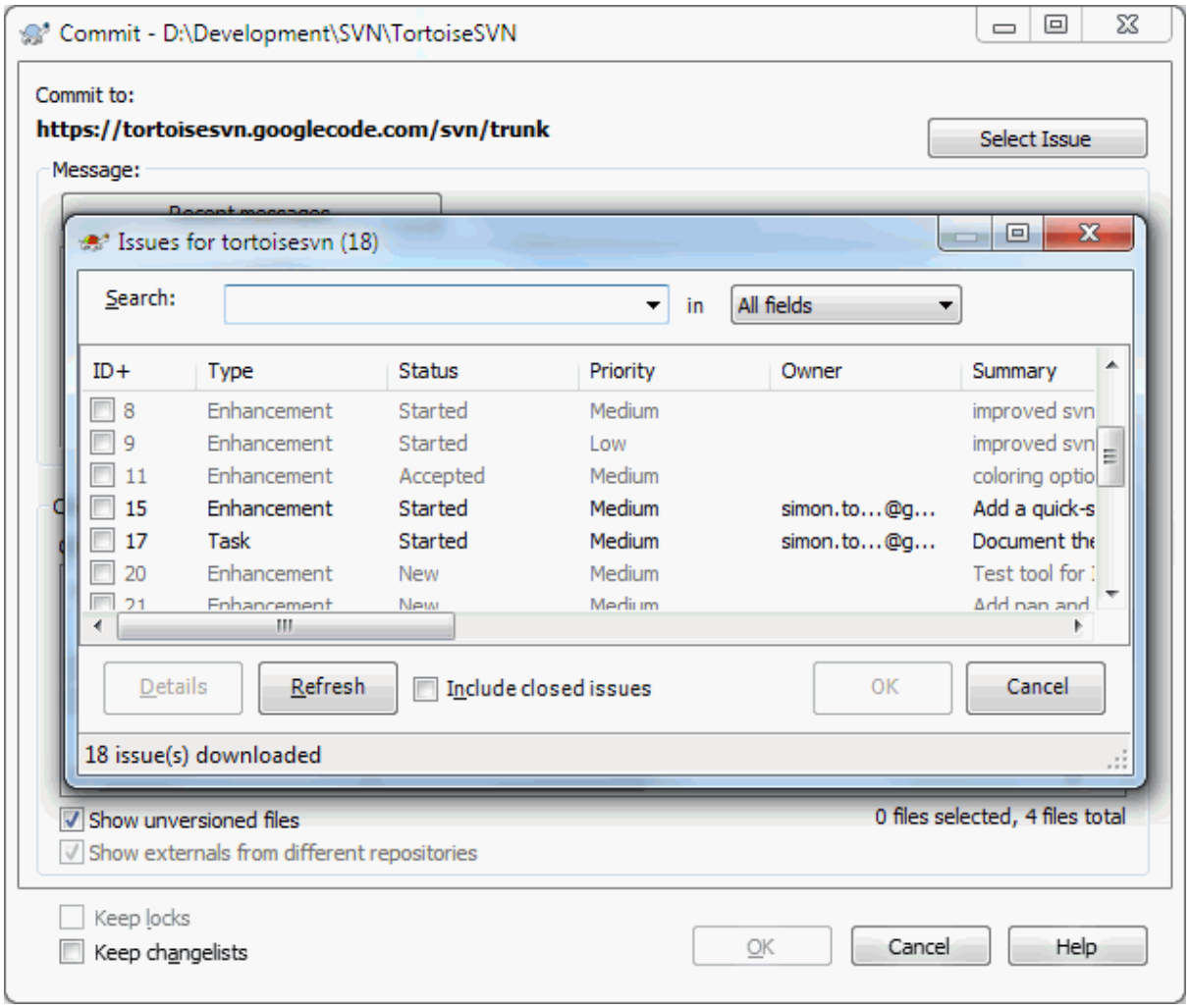


图 4.66. 问题跟踪查询对话框示例

在此示例中，您可以选择一个或多个开放问题。然后，该插件可以生成特殊格式的文本，并将其添加到您的日志消息中。

4.29. #与基于 WEB 的版本库浏览器集成

有许多web为基础的版本库浏览器，例如ViewVC [http://www.viewvc.org/] and WebSVN [http://websvn.tigris.org/]，TortoiseSVN提供了链接这些浏览器的方法。

您可以在 TortoiseSVN 中集成一个您自己选择的版本库查看器。为此，您必须定义一些属性，这些属性定义了连接。它们必须设置于文件夹: (第 4.17 节 “项目设置”)

webviewer:revision
: 将这个属性设置为版本库浏览器显示所有本版本修改内容的url，它必须是URI编码的，也必须包含%REVISION%。在问题单中%REVISION%将会被替换成版本号。这允许TortoiseSVN在日志对话框的增加这样的条目右键菜单 → 在版浏览器中查看此修订

webviewer:pathrevision
: 为您的版本库查看器的地址设置此属性以查看某特定版本中的某特定文件的更改。它必须是完全的 URI 编码并且要包含 %REVISION% 和 %PATH%。%PATH% 用相对于版本库根目录的路径来替换。这允许 TortoiseSVN 在日志对话框中显示一个上下文菜单项 上下文菜单 → 在网络查看器中查看路径版本 例如，如果您在日志对话框的底部窗格中右击一个文件项 /trunk/src/file，那么地址中的 %PATH% 将被 /trunk/src/file 所替换。

您也可以使用相对地址来代替绝对地址。当您的网络查看器和源版本库在同一域/服务器上时，这非常有用。即使在域名称发生更改的情况下，您也不必调整 webviewer:revision 和 webviewer:pathrevision 属性。地址格式与 bugtraq:url 属性的一样，请查 第 4.28 节 “与 BUG 跟踪系统/问题跟踪集成”。

设置文件夹的属性

为了系统能够工作，这个属性必须设置到文件夹上。当您提交文件或文件夹时，属性会从文件夹上读取。如果没有发现属性，TortoiseSVN 会向上级搜索整个目录树，直到一个没有版本控制的文件夹或发现根目录(例如C:\)为止。如果您能够确定每个用户只从 trunk/ 检出，而不是一些子目录，那么您只在 trunk/ 上设置属性就足够了。如果不能确定，您必须为每个子目录设置这些属性。在较深的项目层次结构中的属性设置将重写较高级别上的设置(更靠近 trunk/ 的)。

对于tsvn:属性，例如 tsvn:，bugtraq: 和 webviewer:，你只能对于所有子文件夹使用递归复选框设置属性，不用将这些属性设置在文件上。

当你使用TortoiseSVN在工作副本中新建一个子目录， 上层目录的所有项目属性都会自动的被继承。.

限制使用代码库浏览器

Because the repo viewer integration depends upon accessing Subversion properties, you will only see the results when using a checked out working copy. Fetching properties remotely is a slow operation, so you will not see this feature in action from the repo browser unless you started the repo browser from your working copy. If you started the repo browser by entering the URL of the repository you won't see this feature.

出于同样的原因，当使用版本库浏览器添加子文件夹时，项目属性将不会被自动传递到子文件夹。

4.30. #TortoiseSVN的设置

想知道不同的设置是干什么用的，你只需将鼠标指针在编辑框/选项框上停留一秒钟...一个帮助提示气泡就会弹出来。

4.30.1.#常规设置

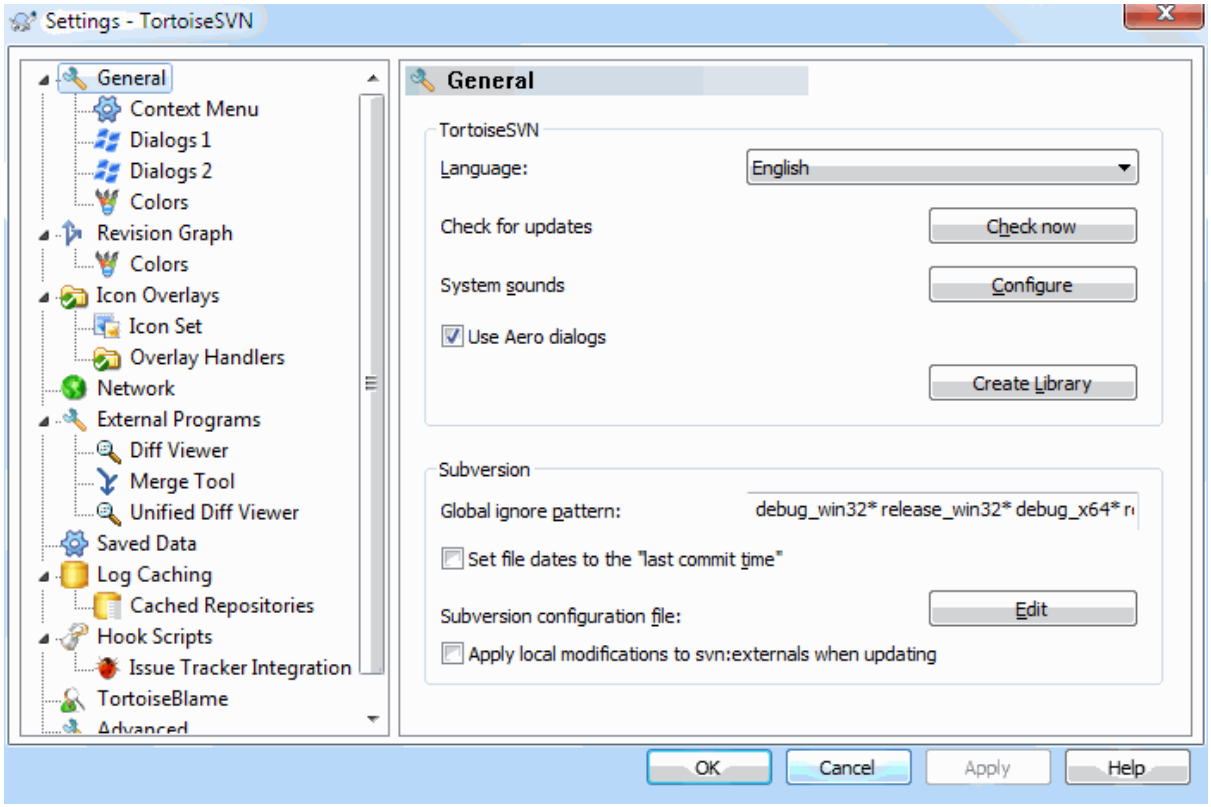


图 4.67. 设置对话框，常规设置页面

这个对话框允许你指定自己喜欢的语言，同时也可做那些与Subversion相关的特殊设置。

语言
选择你TSVN的用户界面语言。不然你还期望从这里得到啥别的？

检查更新
TortoiseSVN 将定期联系它的下载站点以查看是否有更新的程序版本可用。如果有，它将在提交对话框中显示一个通知链接。如果您想立即知晓，请使用现在检查。新版本将不会被下载；您只是简单地接收到一个信息对话框来告诉您有新的版本可用。

系统声音
TSVN已经默认安装了三个自定义声音。

- 错误
- 提示
- 警告

你可以使用Windows控制面板中的声音属性，来选择不同的声音(或是把这些声音完全关掉)。配置按钮是一个打开控制面板声音属性的快捷方式。

Use Aero Dialogs
On Windows Vista and later systems this controls whether dialogs use the Aero styling.

Create Library
On Windows 7 you can create a Library in which to group working copies which are scattered in various places on your system.

全局忽略样式

全局忽略模式是用来防止非版本控制的文件被显示出来，比如在提交对话框中等等。那些符合模式的文件在执行导入操作时同样被忽略。通过输入文件名或扩展名来忽略文件或文件夹。模式之间以空格分隔，例如 bin obj *.bak *.~?? *.jar *.[Tt]mp。这些模式不应该包含任何路径分隔符。还请注意没有办法来区分文件和目录。请阅读 第 4.13.1 节 “忽略列表中的模式匹配”以获得更多模式匹配语法信息。

值得注意的是，你在这里指定的忽略样式将同样作用于你本机上的其他Subversion客户端，包括命令行客户端。

如果你象下面段落那样使用Subversion配置文件来设置一个 全局－忽略 样式，那么它将覆盖你在这里做的设置。该Subversion配置文件可以象下面段落描述的那样，通过 编辑 按钮来访问。

This ignore pattern will affect all your projects. It is not versioned, so it will not affect other users. By contrast you can also use the versioned svn:ignore or svn:global-ignores property to exclude files or directories from version control. Read 第 4.13 节 “忽略文件和目录” for more information.

将文件日期设置为“最后提交时间”

该选项通知 TortoiseSVN 在做检出或更新操作时，把文件日期设置为最后提交的时间。否则 TortoiseSVN 将使用当前日期。如果您正在进行软件开发，总的来说最好使用当前日期，因为软件生成系统通常通过查看时间戳来决定需要编译哪些文件。如果您使用“最后提交时间”并恢复到了一个更旧的文件版本，您的项目可能无法如您预期的那样编译。

Subversion配置文件

Use Edit to edit the Subversion configuration file directly. Some settings cannot be modified directly by TortoiseSVN, and need to be set here instead. For more information about the Subversion config file see the Runtime Configuration Area [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html]. The section on Automatic Property Setting [http://svnbook.red-bean.com/en/1.8/svn.advanced.props.html#svn.advanced.props.auto] is of particular interest, and that is configured here. Note that Subversion can read configuration information from several places, and you need to know which one takes priority. Refer to Configuration and the Windows Registry [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] to find out more.

Apply local modifications to svn:externals when updating

This option tells TortoiseSVN to always apply local modifications to the svn:externals property when updating the working copy.

4.30.1.1. #右键菜单配置

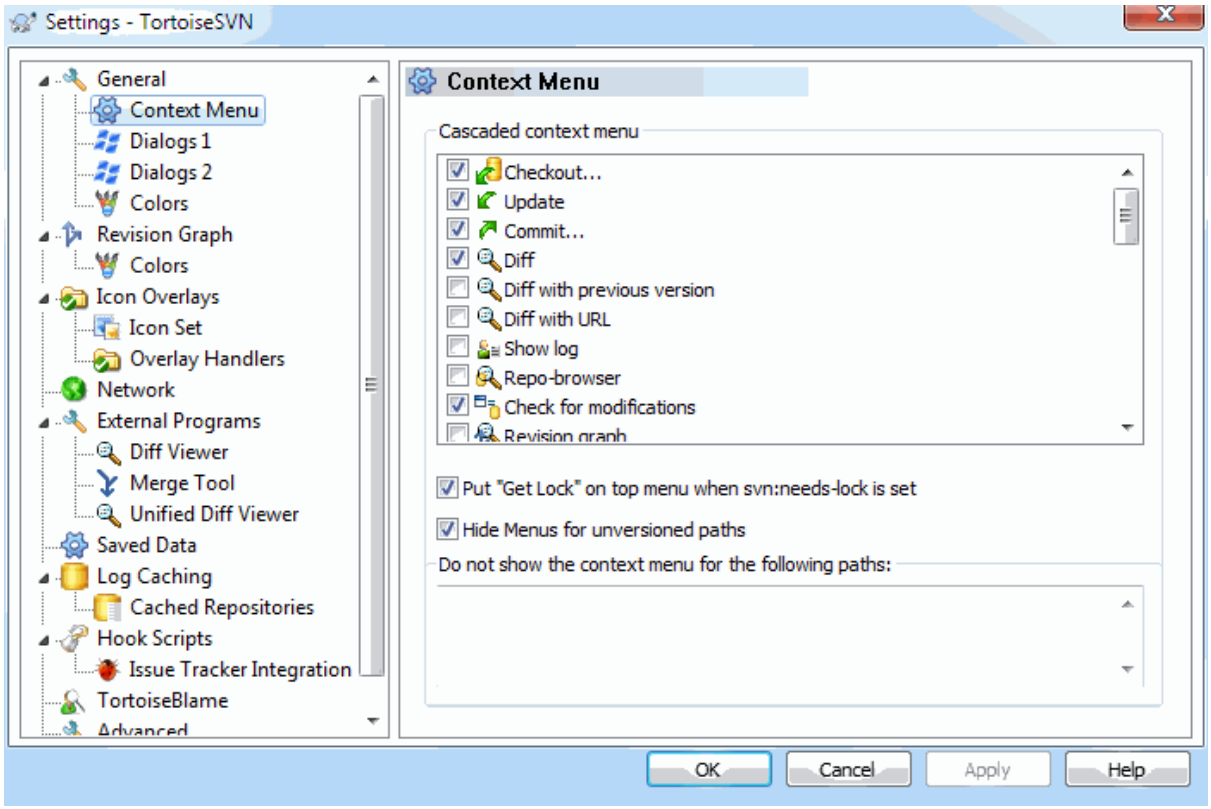


图 4.68. 设置对话框，右键菜单页面

该页面允许你指定： 在TortoiseSVN的主上下文菜单中哪些条目可以直接在鼠标右键菜单显示，哪些在TortoiseSVN子菜单显示。默认情况下很多项未被勾选，只在子菜单显示。

获得锁会有一个特别的情况，你可以将其提升到顶级带但，但是大多数文件不需要锁定，这样做只是添加了混乱。然而，一个标记为svn:needs-lock属性的文件每次编辑前都需要那个操作，所以这个菜单会进入顶级菜单会比较方便。选定这个选项，会使设置svn:needs-lock属性的文件的Get Lock出现在顶级菜单中。

Most of the time, you won't need the TortoiseSVN context menu, apart for folders that are under version control by Subversion. For non- versioned folders, you only really need the context menu when you want to do a checkout. If you check the option Hide menus for unversioned paths, TortoiseSVN will not add its entries to the context menu for unversioned folders. But the entries are added for all items and paths in a versioned folder. And you can get the entries back for unversioned folders by holding the Shift key down while showing the context menu.

如果您不想在计算机中的某些路径中显示 TortoiseSVN 的上下文菜单，您可以在底部的框中列出它们。

4.30.1.2. #TSVN对话框设置一

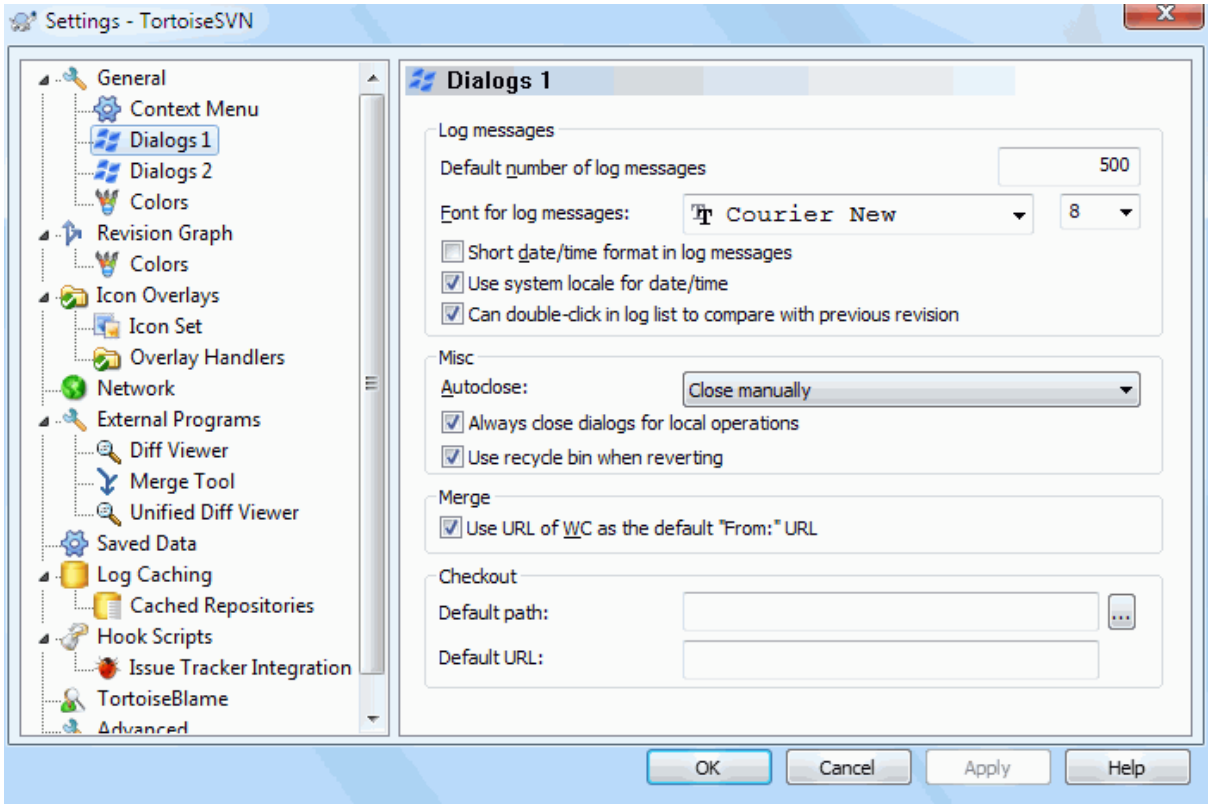


图 4.69. 设置对话框，对话框一页面

此对话框允许你按照喜欢的方式去配置一些TSVN的对话框。

默认的日志信息数
: 限制你首次选择 TortoiseSVN → 显示日志 时，TortoiseSVN 从服务器获取的日志信息数。在服务器连接缓慢时很有用。你可以使用 全部获取 或 下100条 来获得更多信息。

日志信息字体
: 选择日志信息显示的字体样式和大小，作用域为版本日志对话框的中间窗格，以及提交对话框时填写日志信息的窗格。

日志信息使用短日期/时间格式
: 如果标准长度的日期/时间信息占在用了过多的屏幕空间，可以使用短格式。

可以在日志列表中双击以便与以前的版本比较
: 如果您经常在日志对话框的顶部窗格中进行版本比较，您可以使用此选项来允许双击操作。默认情况下是不启用的，因为获取差异是相当漫长的过程，并且许多人在意外双击后都不愿等待，这就是此选项默认禁用的原因。

自动关闭
: 当一个动作正确无误地完成时，TSVN可以自动关闭所有的进程对话框。这项设置允许你选择在何种情况下关闭对话框。默认(推荐)的设置是 手动关闭 ，允许你重新浏览所有信息并检查发生了什么。当然，你可能会决定忽略某些类型的信息并在你的操作没做出什么重大改变的情况下让对话框自动关闭。

  如无合并、添加、删除操作，自动关闭 意味着如果有简单更新的话，进程对话框将关闭。但如果版本库的更改和你的内容进行了合并，或若有任何文件被添加或删除，对话框将保持打开。若操作中发生什么冲突和错误这些对话框也将同样保持打开。

  无冲突时自动关闭 更放宽了标准，即使在无合并、添加、删除操作时也同样关闭对话框。当然，如果操作发生了任何冲突或错误，对话框将保持打开。

如无错误，自动关闭　即使在有冲突发生时也会关闭。维持对话框打开的唯一条件是发生了错误，使得Subversion无法完成任务。举个例子，一个更新操作由于服务器不可达而失败了，或是一个提交操作因为工作副本已经过期而失败。

对本地操作始终关闭对话框
像添加文件或恢复更改这样的本地操作不需要联系版本库并且可快速完成，因此进度对话框通常意义不大。如果您想要进程对话框在这些操作后自动关闭，则选择此选项，除非有错误发生。

在还原的时候使用回收站
当您恢复本地修改时，您的更改将被丢弃。TortoiseSVN 通过在恢复至原始副本之前将已修改文件发送至回收站来赋予您额外的安全保证。如果您想跳过回收站，则不要选择此选项。

将工作副本 URL 用作默认的 “来源:” URL
在合并对话框里，默认行为是在每次合并中记忆 起始: 的URL。无论如何，都有某些人喜欢在他们的版本进化树中从很多不同的位置执行合并操作，他们发现从当前工作副本的URL开始更方便些。该URL可以随后被编辑来指向一个同级路径或另一个分支。

缺省检出路径
你可以指定缺省的检出路径。如果你保持所有检出在同一个地方，那么预先填写的路径是极为有用的，这样你只需要在路径末尾增加新的目录名称即可。

缺省检出URL
你可以指定缺省的检出URL。如果你经常检出一些大项目的子工程，那么预先填写的URL是极为有用的，这样你只需要在路径末尾增加新的工程名称即可。

4.30.1.3. #TSVN对话框设置二

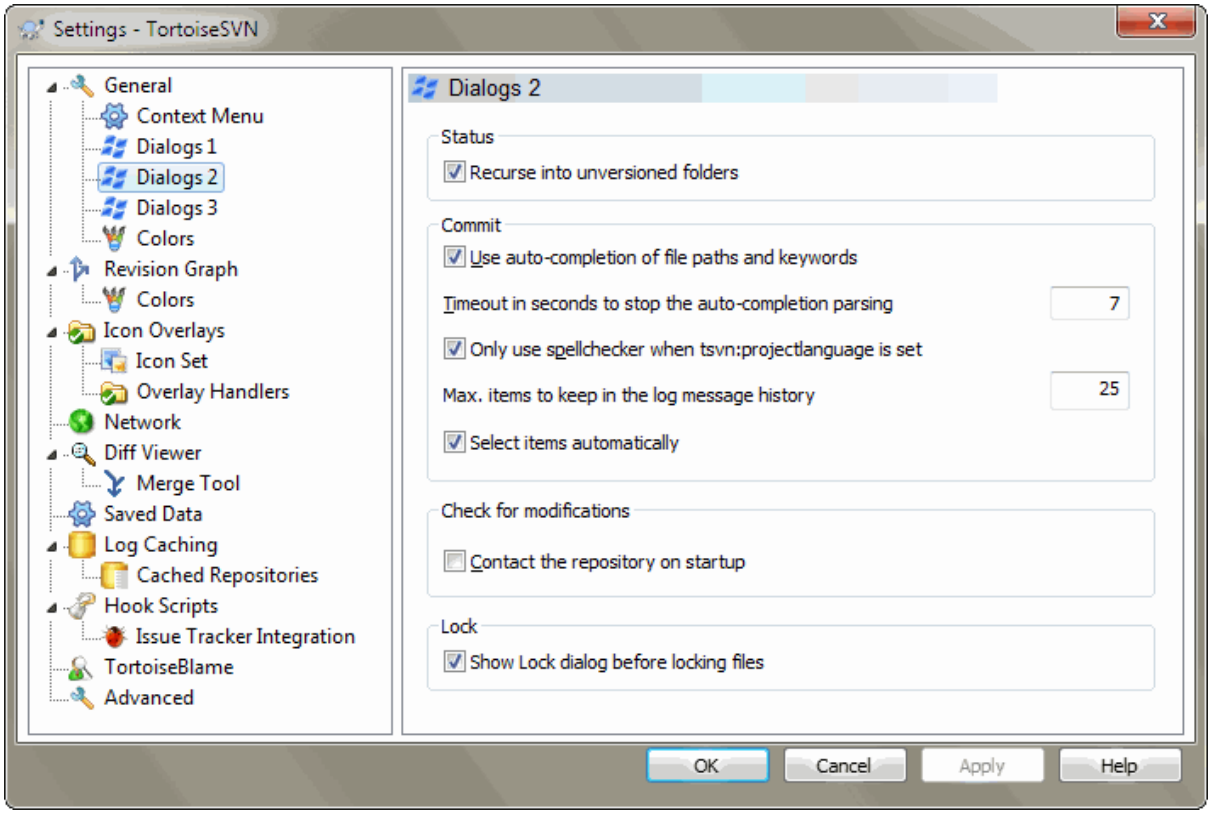


图　4.70.　设置对话框，对话框二页面

递归处理未进行版本控制的文件夹
若这个选项框被选中(默认状态)，那么一个非版本控制的文件夹，不论在 添加，提交 或 检查修改 时显示的是什么状态，它的每个子文件和子文件夹都要同样显示。取消选择将减少这些对话框中的混乱程度。这样一来如果你选择添加一个非版本控制的文件夹，将会非递归地添加。

在 检查修改 对话框中您可以选择查看被忽略的项目。如果选中该复选框，那么每当找到一个被忽略的文件夹，所有的子项也将被显示。

自动完成文件路径和关键词
提交对话框包含了一项功能，可以解析被提交的文件名列表。当您输入列表中某个项目的前三个字母时，自动完成框就会弹出，您就可以按回车来完成该文件名。选中该框以启用此功能。

自动完成分析的超时时间(秒)
如果有许多大文件要检查，自动完成解析器可能会非常慢。该超时时间设置可以防止提交对话框被长时间挂起。若您错过了某些重要的自动完成信息，您可以延长该超时时间。

仅在设置了 tsvn:projectlanguage 时才进行拼写检查
若你不愿意在所有提交操作时都进行拼写检查，就选择该选项。而后拼写检查功能将在项目属性做出明确要求时才生效。

日志中保留的最大条目数量
当您在提交对话框中输入日志消息时，TortoiseSVN 会存储起来以便以后可能重新使用。默认情况下，它将为每个版本库保存最后 25 条日志信息，但是您可以在此自定义该数目。若您有很多不同的版本库，您可能会希望减少该数目以防止向注册表中填入过多信息。

请注意此设置仅应用于您在此计算机上键入的消息。它跟日志缓存毫不相关。

如果提交失败，自动重新打开提交和分支/标签对话框
When a commit fails for some reason (working copy needs updating, pre-commit hook rejects commit, network error, etc), you can select this option to keep the commit dialog open ready to try again.

自动选择项目
在提交对话框中的正常行为就是选择所有更改的(受版本控制的)项目以备自动提交。如果您想在启动时不选择任何项目并且手动挑选项目以备提交，则不要选择此框。

启动时连接版本库
“检查修改”对话框将默认检查工作副本，但仅当你点击 检查版本库 时才连接你的版本库做检查。若你想总是去检查版本库，就可以使用该设置来使版本库检查的动作每次都自动启动。

在锁定文件之前显示加锁对话框
当你选择一个或多个文件，然后选择 TortoiseSVN → 加锁 后，一些项目的惯例是写加锁信息，解释你为什么锁定这些文件。如果你不使用加锁信息，可以取消此选择框，从而略过对话框，直接锁定文件。

如果你在目录上使用加锁命令，一定会出现加锁对话框，因为它要让你选择加锁的文件。

如果你的项目使用了 tsvn:lockmsgminsize 属性，那么不管你如何设置，都会看到加锁对话框，因为此项目需要加锁信息。

4.30.1.4.#TortoiseSVN Dialog Settings 3

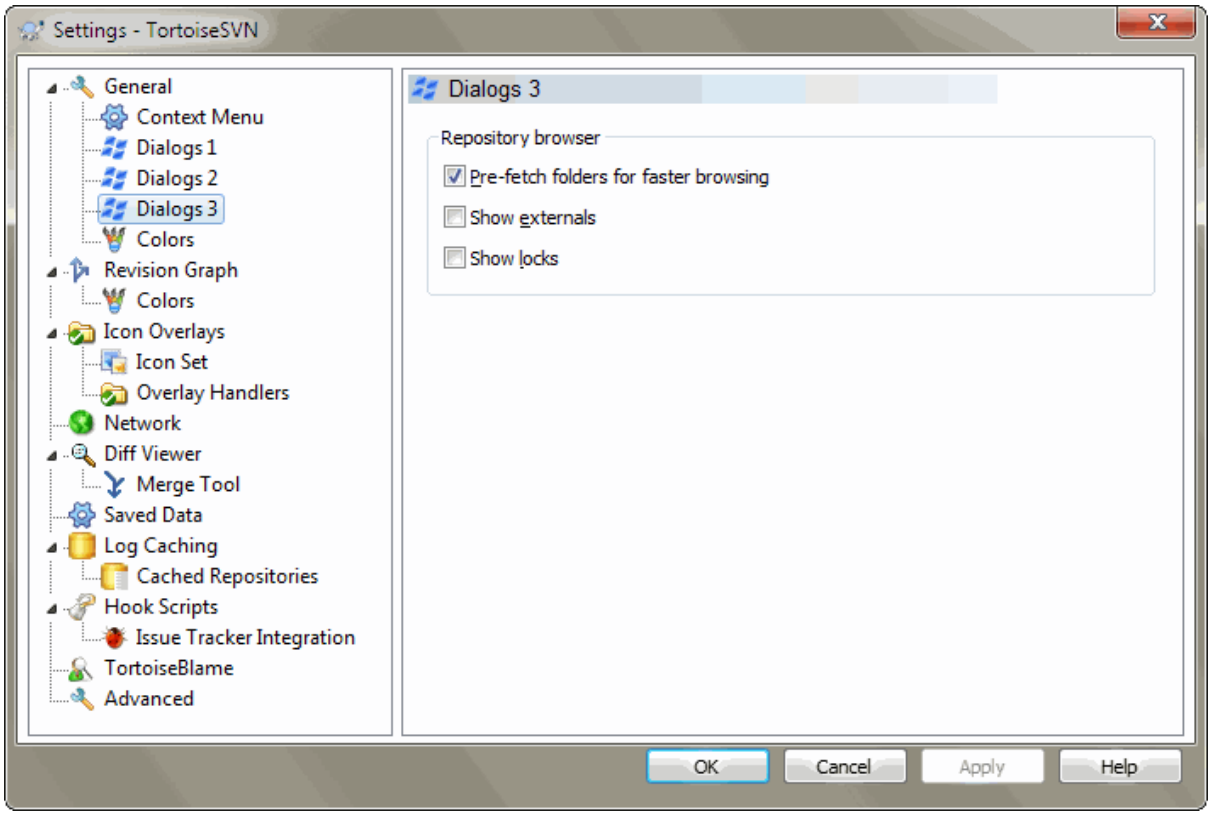


图 4.71. The Settings Dialog, Dialogs 3 Page

预取文件夹以加速浏览

If this box is checked (default state), then the repository browser fetches information about shown folders in the background. That way as soon as you browse into one of those folders, the information is already available.

Some servers however can't handle the multiple requests this causes or when not configured correctly treat so many requests as something bad and start blocking them. In this case you can disable the pre-fetching here.

显示外部

If this box is checked (default state), then the repository browser shows files and folders that are included with the svn:externals property as normal files and folders, but with an overlay icon to mark them as from an external source.

As with the pre-fetch feature explained above, this too can put too much stress on weak servers. In this case you can disable this feature here.

4.30.1.5. #TortoiseSVN 颜色设置

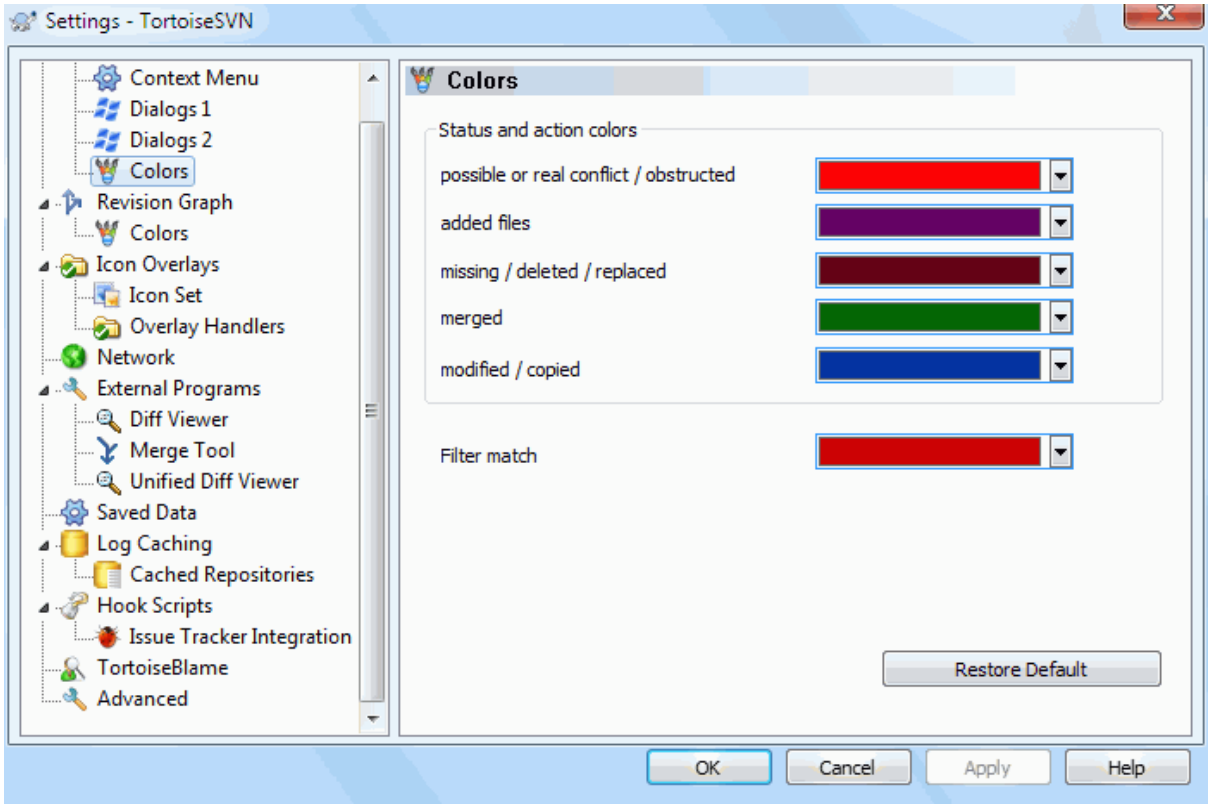


图 4.72. 设置对话框，颜色页面

此对话框允许你按照你喜欢的方式来配置TSVN对话框使用的文本颜色。

可能或确实有冲突/问题
: 当更新时或合并时发生了冲突。如果对应于版本控制下的文件/文件夹，存在一个同名的非版本控制的文件/文件夹，此时做更新将被阻碍。

  此颜色同样被用在进程对话框的错误信息中。

添加文件
: 向版本库添加的条目。

丢失/已删除/已替换
: 已从工作副本中遗失的条目；已从版本库中删除；或已经从工作副本删除并且被另一个同名文件替换。

已合并
: 从版本库所做的更改被成功地合并到工作副本，并无任何冲突产生。

已修改/已复制
: 已经增加(现在只是修改)，或者在版本库中复制。也在包含复制条目的日志对话框中使用。

删除的节点
: 一个已经从版本库中删除了的条目。

添加的节点
: 一个通过添加、复制或移动操作，已经被添加到版本库的条目。

重命名的节点
: 一个在版本库中已经被重命名的条目。

替换的节点
    该原始条目已经被删除，且有同名条目替换了的条目。

过滤器匹配
    当在日志对话框中使用筛选时，在结果中使用颜色来突出显示搜索对象。

4.30.2. #版本图设置

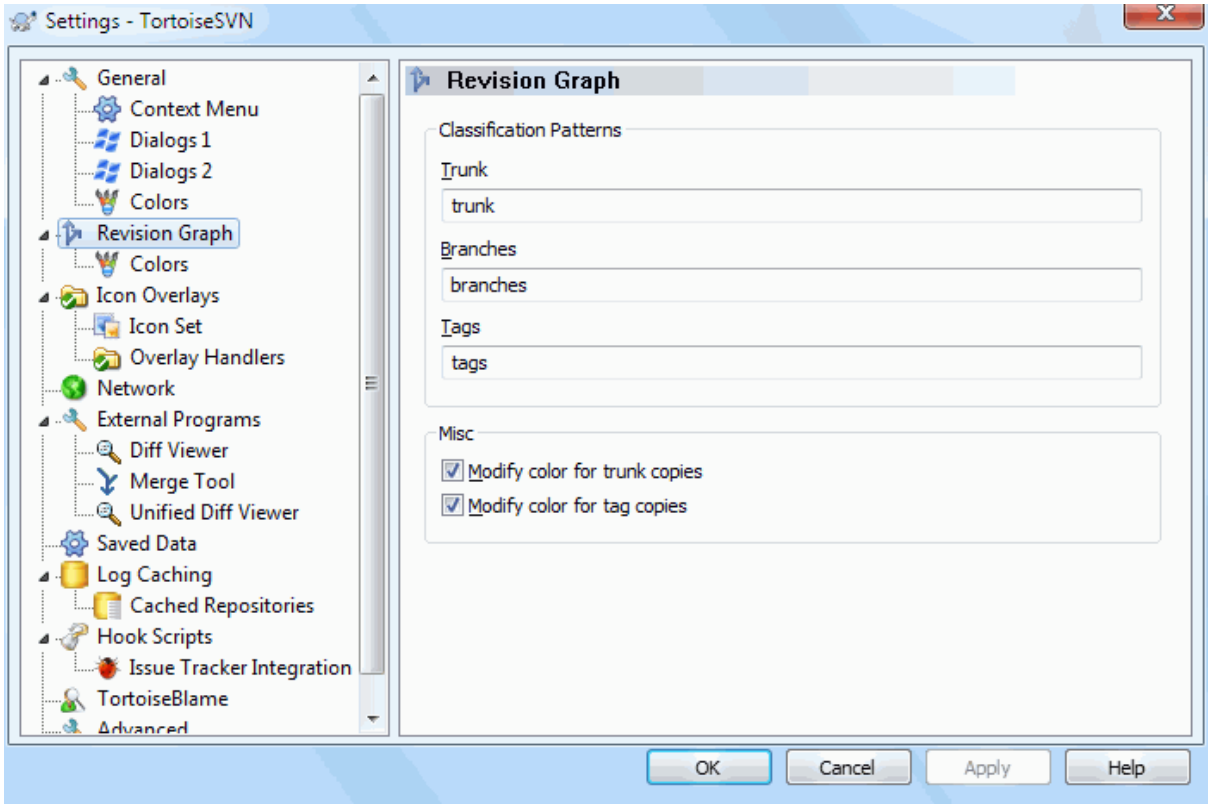


图 4.73. 设置对话框，版本图页面

分类模式
    版本图会通过区分主干，分支和标签来为版本库结构显示更为清晰的图示。由于 Subversion 没有内置这样的分类，此信息主要提取自路径名称。默认设置假设您使用 Subversion 文档中所建议的常规英文名称，当然您的使用情况也可能发生改变。

    在所提供的三个框中指定用来识别这些路径的模式。模式不会区分大小写，但您必须以小写来指定它们。通配符 * 和 ? 将照常工作，并且可以使用 ; 来分隔多个模式。不要包括任何额外的空格，因为它将包含在匹配规范中。

修改颜色
    版本图使用颜色来表示节点类型，即节点是否是添加的，删除的，重命名的。为了帮助分辨出节点分类，您可以允许版本图混合颜色以表示即是节点类型又是分类。如果选中该复选框，将使用颜色混合。若未选中此框，颜色仅用于表示节点类型。使用颜色选择对话框来分配要使用的特定颜色。

4.30.2.1. #版本图颜色

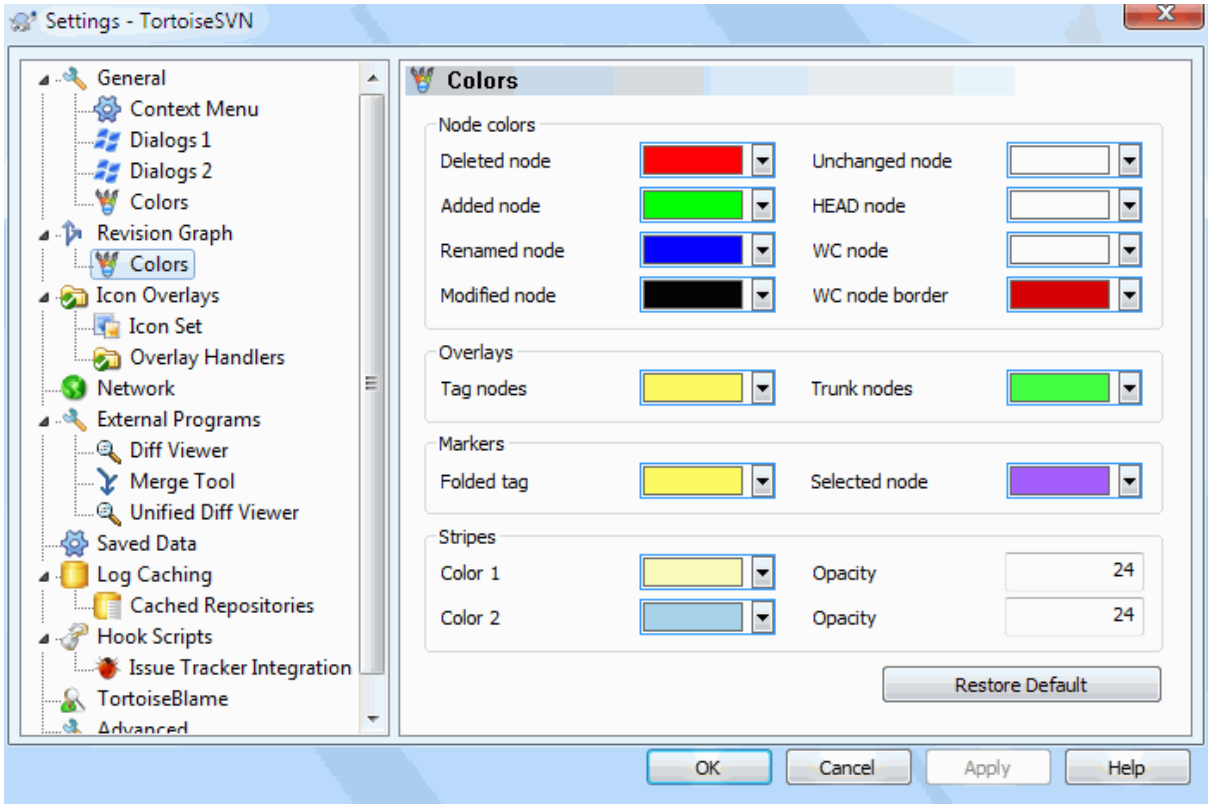


图 4.74. 设置对话框，版本图颜色页面

此页面允许您配置要使用的颜色。请注意，这里指定的颜色是纯色。大多数节点都是使用节点类型颜色、背景颜色和分类颜色(可选)的混合色。

删除的节点
: 在同一修订版本中已被删除并且不会复制到任何别的地方的项目。

添加的节点
: 新添加的或复制的(用历史添加)项目

重命名的节点
: 在同一修订版本中从一个位置删除并且在另一位置添加的项目。

已修改的节点
: 无任何添加或删除的简单的修改。

未修改的节点
: 可用于显示用作副本源的修订版本，即使修订版本中所绘制的项目没有发生更改。

主干节点
: 版本库中的当前 HEAD 版本。

工作副本节点
: 如果您选择为您修改后的工作副本(在图形中连接至它最后提交的修订版本)显示额外的节点，使用此颜色。

工作副本节点边框
: 如果您选择显示工作副本是否被修改，当发现修改时为工作副本节点使用此颜色边框。

标签节点
: 分类为标签的节点也可以用此颜色混合。

主干节点
分类为主干的节点也可以用此颜色混合。

折叠式的标签标记符
如果您使用标签折叠以节省空间，使用此颜色块在副本源上来标记标签。

所选节点标记符
当你左键点击节点来选择它时，用于表示选定的标记符为此颜色块。

修剪
当版本图被拆分成子树结构时将使用这些颜色，并且用交替的条纹颜色来标示背景，以便分辨出各个树结构。

4.30.3. #图标叠加设置

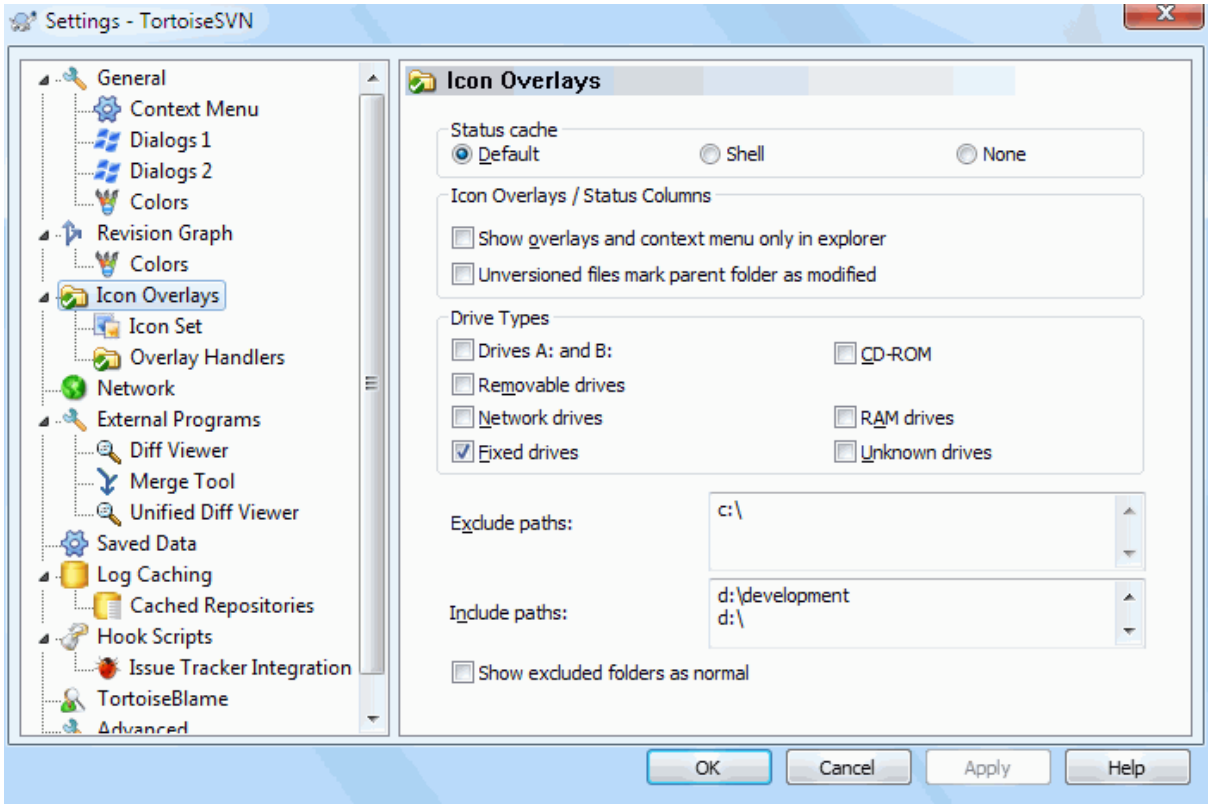


图 4.75. 设置对话框，图标覆盖页面

此页面允许您选择 TortoiseSVN 要显示图标覆盖的项目。

因为它要花费一段时间来获取工作副本的状态，TortoiseSVN 将使用一个缓存来存储这些状态，从而使资源管理器在显示图标覆盖时，不会消耗太多资源。您可以根据您的系统和工作副本大小来选择让 TortoiseSVN 使用哪种类型的缓存：

默认
以一个单独的进程(TSVNCache.exe)来缓存所有的状态信息。该进程监视所有驱动器的更改，并在工作副本中的文件被修改时重新获取其状态。该进程以最低优先级运行，所以其他程序不会被它挤占。这也意味着状态信息并不是 实时 的，因为它需要几秒钟时间处理图标重载的变化。

优点： 图标覆盖递归地显示状态，就是说，如果一个处在工作副本深处的文件被修改了，所有上级目录直到工作副本的根目录都会显示出已修改的覆盖图标。由于该进程可以向 Windows 外壳发送通知，所以资源管理器左侧树视图通常也会更改。

缺点： 即使你已经不在项目下工作了，该进程仍然持续运行。取决于你工作副本的数量和大小，它将占用10-50 MB的RAM内存空间。

Windows 外壳
: 缓存在外壳扩展dll中直接完成，但仅仅是为那些当前可见的文件夹。每次你浏览到其他文件夹，状态信息就会被重新获取。

  优点: 仅仅需要很少的内存(大约 1 MB)，并且可以 实时 显示状态。

  缺点: 因为仅有一个文件夹被缓存，图标重载不会递归地显示状态。在大一些的工作副本下，它在浏览器中显示一个文件夹将比默认缓存模式花费更多时间。而且 mime-type 列将无效。

无
: 在这种设置下，TSVN在浏览器里就完全不去获取状态了。因此，版本控制下的文件将不会获得任何图标重载。文件夹也仅仅有个“正常”状态的图标重载，其他的不会显示，也不会有其他额外的列可用。

  优点: 绝对不会占用任何额外的内存，也完全不会减慢浏览器的浏览速度。

  缺点: 文件和文件夹的状态信息不会显示在资源管理器中。要获知工作副本是否被修改，您需要使用 “检查修改” 对话框。

默认情况下，覆盖图标和上下文菜单将出现在所有的打开/保存对话框中，就像在 Windows 资源管理器中一样。如果您想让它们仅出现在 Windows 资源管理器中，选择 仅在资源管理器中显示图标覆盖和上下文菜单。

您还可以选择将文件夹标记为已修改，如果它们包含非版本控制的项目。这很有用，可以提醒您已经创建了尚未受版本控制的新文件。此选项仅适用于在使用 默认 状态缓存选项时(见下文)。

下一组允许您选择要显示图标覆盖的存储器种类。默认情况下，仅硬盘驱动器被选中。您甚至可以禁用所有的图标覆盖，但这样做还有何乐趣?

网络驱动器可能会很慢，所以默认情况下不会为位于网络共享上的工作副本显示图标。

USB闪存看上去是个特殊情况，因为驱动类型是设备自主标识的。于是有些显示为固定驱动器，而有些显示为可移动磁盘。

排除路径 是用来告诉 TortoiseSVN 不用 在哪些路径下显示图标覆盖和状态列。如果您有些很大的工作副本，而这些工作副本仅仅包含您完全不会改变的库文件，因此也就不需要显示图标覆盖，或者您只想 TortoiseSVN 在特定的目录中查看。

您在此指定的任何路径都被假定为递归应用，因此子文件夹也都不会显示图标覆盖。如果您 只 想排除有名称的目录，在路径后追加 ?。

包含路径 也使用同样的语法。除了有些反例: 即使该路径处在某个取消图标重载显示的特定驱动类型下，或是处在上面的排除路径之下， 也依然会显示图标重载。

用户有时会问这三种设置如何相互作用。对于任何给定的路径，检查包含和排除列表，向上搜寻整个目录结构直到找到匹配。当找到第一个匹配时，遵循包含或排除规则。如果有冲突，单个目录规范的优先级高于递归规范，然后包含优先级高于排除。

此例将有助理解:

排除:

```
C:
C:\develop\?
C:\develop\tsvn\obj
C:\develop\tsvn\bin
```

包含:

```
C:\develop
```

这些设置为 C: 驱动器禁用图标覆盖，除了 c:\develop外。该目录下的所有项目都将显示图标覆盖，除了 c:\develop 目录自身外，其被专门忽略。高变动的二进制文件夹也被排除在外。

TSVNCache.exe 同样使用这些路径来限制它的扫描。如果你想让它仅仅在某些特定文件夹里监视，就取消所有的驱动器类型，并仅仅包含你允许被扫描的文件夹。

排除 SUBST 磁盘

使用 SUBST 驱动器访问您的工作副本通常很方便，比如，使用命令

```
subst T: C:\TortoiseSVN\trunk\doc
```

然而这可能会导致图标覆盖不更新，因为 TSVNCache 在文件更改时只会收到一个通知，这对于原始路径来说是正常的。这意味着您在 subst 路径上的图标覆盖可能从不会更新。

解决此问题的一个简单方法就是排除原始路径，使其不显示图标覆盖，以便于图标覆盖显示于 subst 路径。

Sometimes you will exclude areas that contain working copies, which saves TSVNCache from scanning and monitoring for changes, but you still want a visual indication that a folder contains a working copy. The Show excluded root folders as 'normal' checkbox allows you to do this. With this option, working copy root folders in any excluded area (drive type not checked, or specifically excluded) will show up as normal and up-to-date, with a green check mark. This reminds you that you are looking at a working copy, even though the folder overlays may not be correct. Files do not get an overlay at all. Note that the context menus still work, even though the overlays are not shown.

作为一个特殊例外， 从不为 将排除文件夹显示为 '正常' 选项考虑驱动器 A: 和 B:。这是因为 Windows 强制查看驱动器，这可能导致在资源管理器启动时延迟数秒钟，即使您的 PC 确实有一个软盘驱动器。

4.30.3.1. #图标集选择

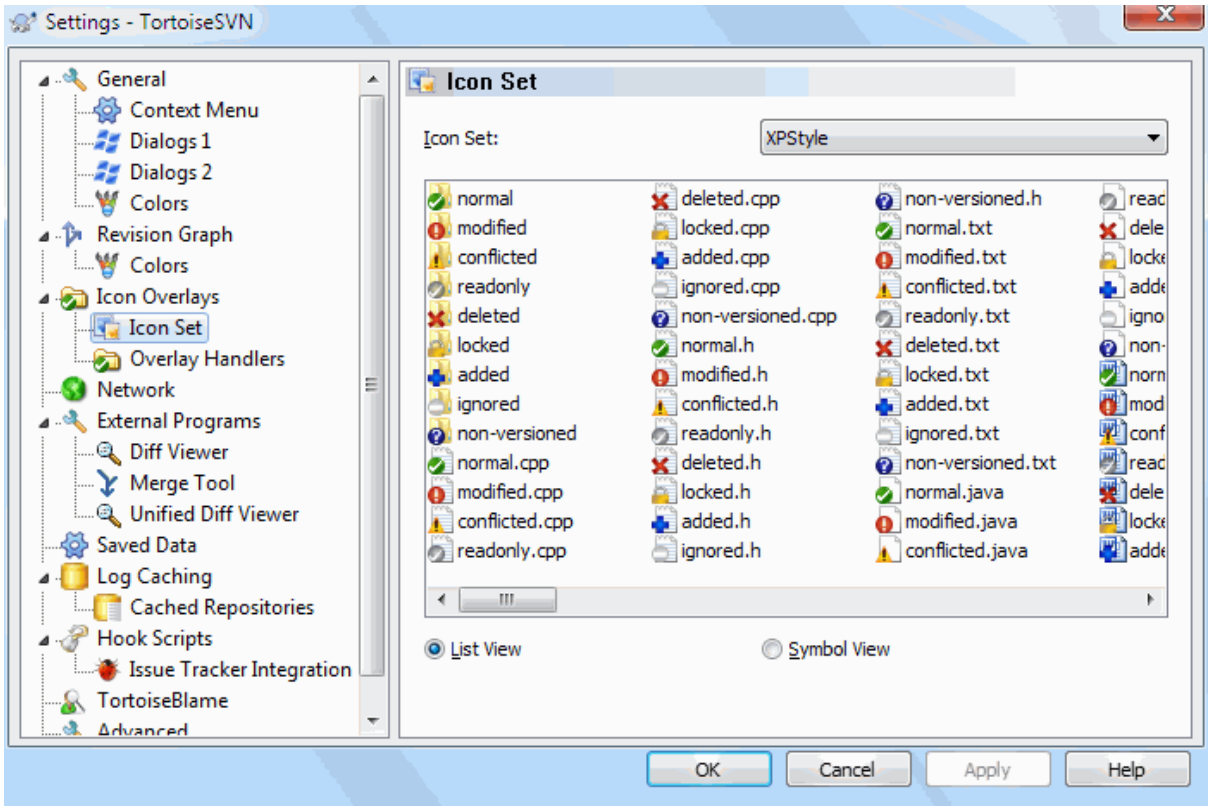


图 4.76. 设置对话框，图标集页面

你可以选择你最喜欢的重载图标集。要注意的是，倘若改变了重载图标集，你可能需要重启计算机使更改生效。

4.30.3.2. #启用的图标覆盖

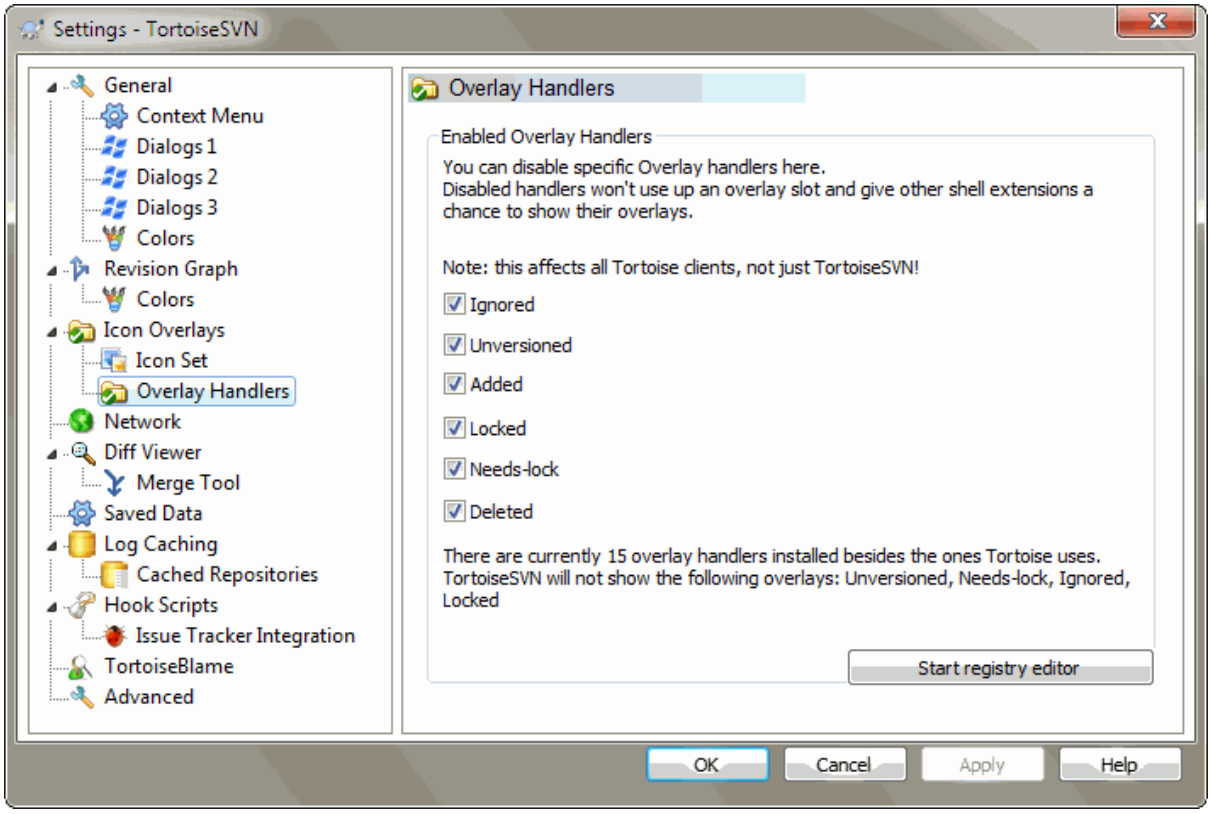


图 4.77. 设置对话框，图标处理器页面

Because the number of overlays available is severely restricted, you can choose to disable some handlers to ensure that the ones you want will be loaded. Because TortoiseSVN uses the common TortoiseOverlays component which is shared with other Tortoise clients (e.g. TortoiseCVS, TortoiseHg) this setting will affect those clients too.

4.30.4. #网络设置

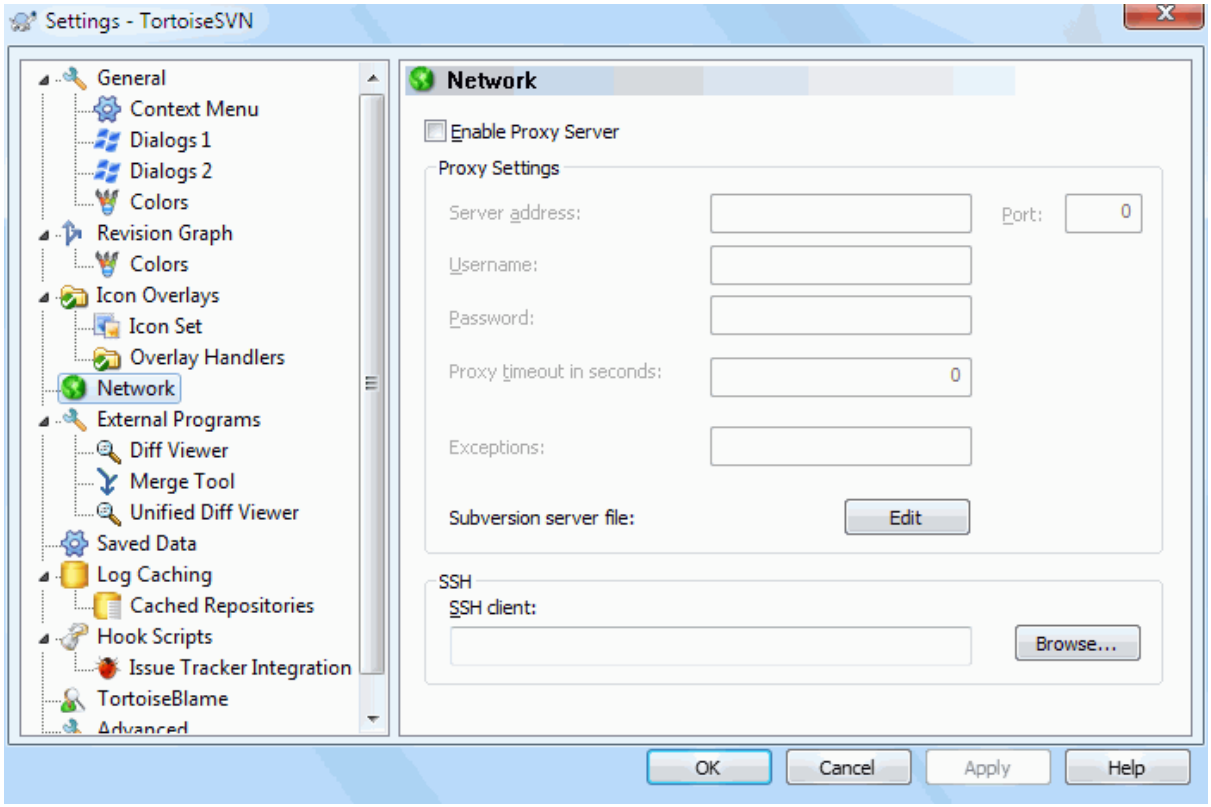


图 4.78. 设置对话框，网络设置页面

如果需要穿透你公司的防火墙，在这里可以配置你的代理服务器。

If you need to set up per-repository proxy settings, you will need to use the Subversion servers file to configure this. Use Edit to get there directly. Consult the Runtime Configuration Area [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html] for details on how to use this file.

你同样可以在此指定SSH客户端程序，用来支持TortoiseSVN同使用svn+ssh协议的版本库建立安全连接。我们推荐您使用TortoisePlink.exe。这是著名的Plink程序的一个定制版本， 并且业已包含在TortoiseSVN之中，但它被编译成了一个无窗口的应用，因此当你每次认证的时候将不会看到弹出的DOS窗口。

您必须指定可执行文件的完整路径。对于 TortoisePlink.exe 这是在标准的 TortoiseSVN bin 目录。使用 浏览 按钮来帮助找到它。请注意，如果路径中包含空格，则必须将其用引号括起来，例如

"C:\Program Files\TortoiseSVN\bin\TortoisePlink.exe"

这里有个不弹出窗口的副作用：将没有什么错误信息可供你追踪。因此倘若认证失败你将得到一个信息说：“Unable to write to standard output”。这样一来，我们就推荐你第一次设置时使用原始的Plink程序；而当一切工作正常之时，再使用定制版的TortoisePlink，并且重复利用那些相同的参数。

TortoisePlink 自身没有任何文档，因为它只是 Plink.的小变种。从 PuTTY 网站 [http://www.chiark.greenend.org.uk/~sgtatham/putty/] 了解有关命令行参数。

为了避免反复提示输入密码，您也可以考虑使用如 Pageant 之类的密码缓存工具。这也可从 PuTTY 网站下载。

最后，在服务器和客户端上设置 SSH 是一个重要的过程，这已经超出了本帮助文件的范畴。然而，您可以在下面列出的 TortoiseSVN FAQ 中找到指南 Subversion/TortoiseSVN SSH How-To [http://tortoisesvn.net/ssh_howto]。

4.30.5. #外部程序设置

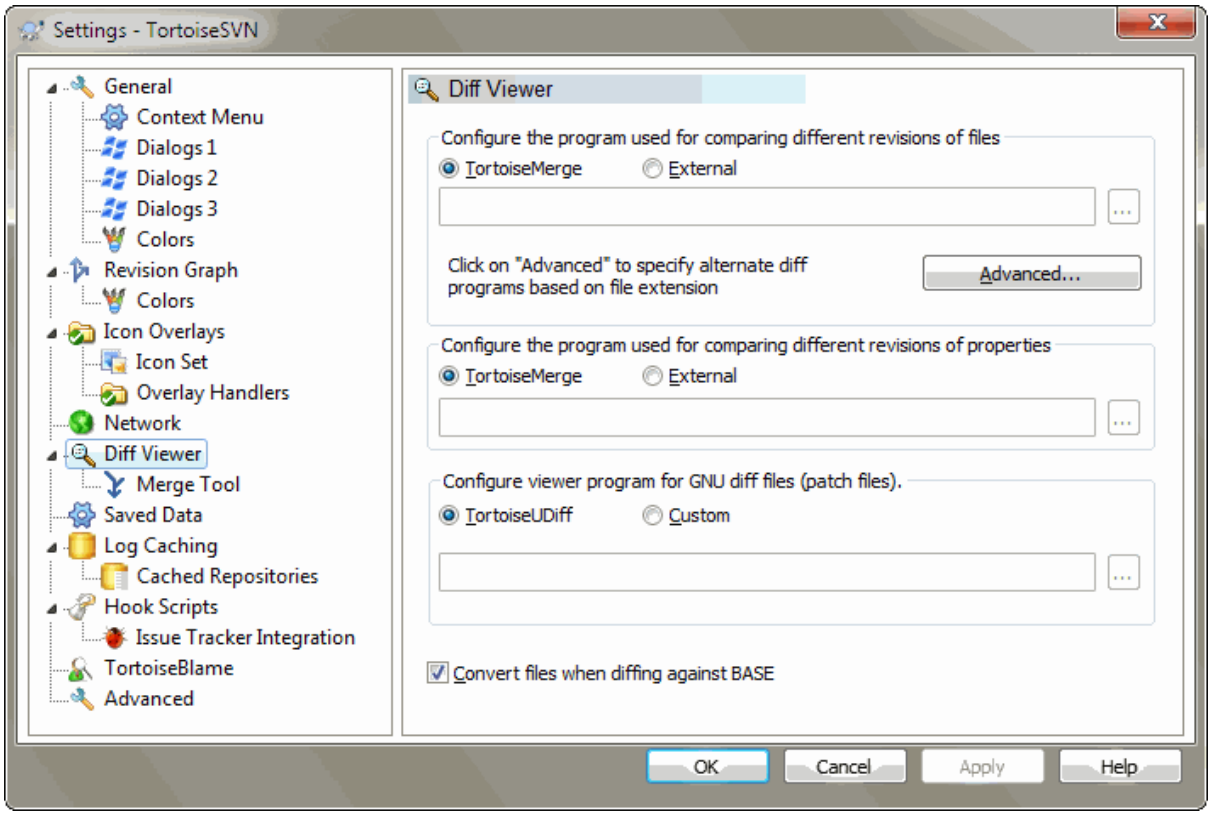


图 4.79. 设置对话框，差异查看页面

在这里你可以定义你自己的差异查看/合并工具. 默认设置是使用与TortoiseSVN一同安装的TortoiseMerge。

阅读 第 4.10.6 节 “其他的比较/合并工具”来了解人们为配合TortoiseSVN工作而使用的外部差异查看/合并程序列表。

4.30.5.1. #差异查看器

有时你可能需要一个外部的差异查看程序来比较不同版本的文件。在为你的命令行填写各种可选参数的同时，要确保这些外部程序从中获得文件名。在TortoiseSVN编辑命令行时，使用以 % 开头的替代参数。当外部程序执行至遇到这些替代参数，它将从TortoiseSVN那里获取那些实际的值。参数的填写顺序将依赖于你使用的差异查看程序。

%base
没更改的原始文件

%bname
原始文件的窗口标题

%mine
你更改过的新文件

%yname
你新文件的窗口标题

%burl
The URL of the original file, if available

%yurl
    The URL of the second file, if available

%brev
    The revision of the original file, if available

%yrev
    The revision of the second file, if available

%peg
    The peg revision, if available

窗口标题并不纯是文件名。TortoiseSVN 将其当作名称来显示并创建相应的名称。因此，假设您正在将版本 123 中的一个文件同工作副本中的一个文件做差异对比时，名称将显示为 文件名 : 版本 123 和 文件名 : 工作副本。

For example, with ExamDiff Pro:

```
C:\Path-To\ExamDiff.exe %base %mine --left_display_name:%bname
    --right_display_name:%yname
```

or with KDiff3:

```
C:\Path-To\kdiff3.exe %base %mine --L1 %bname --L2 %yname
```

or with WinMerge:

```
C:\Path-To\WinMerge.exe -e -ub -dl %bname -dr %yname %base %mine
```

or with Araxis:

```
C:\Path-To\compare.exe /max /wait /title1:%bname /title2:%yname
    %base %mine
```

or with UltraCompare:

```
C:\Path-To\uc.exe %base %mine -title1 %bname -title2 %yname
```

or with DiffMerge:

```
C:\Path-To\DiffMerge.exe -nosplash -t1=%bname -t2=%yname %base %mine
```

如果您使用 svn:keywords 属性来扩展关键词，特别是一个文件的修订版本，那么只要关键词不同，文件之间也会存在差异。同样如果您使用 svn:eol-style = native，原 BASE 文件只有纯 LF 行结尾，而您的文件将会是 CR-LF 行结尾。TortoiseSVN 将通常在做 diff 操作之前先解析 BASE 文件以扩展关键词和行结尾，从而自动隐藏这些差异。然而，对于大文件这可能要花上很长的时间。如果不选择 与基础版本比较时转换文件，那么 TortoiseSVN 将跳过对这些文件的预处理。

你也可以使用 Subversion 属性来指定其它的比较工具。既然这些是简短的文本，你可能想要使用简单的查看器。

如果您已经配置了备用的 diff 工具，您可以从上下文菜单访问 TortoiseMerge 和第三方工具。上下文菜单 → Diff 使用主 diff 工具，and Shift+ 上下文菜单 → Diff 使用第二 diff 工具。

At the bottom of the dialog you can configure a viewer program for unified-diff files (patch files). No parameters are required. The Default setting is to use TortoiseUDiff which is installed alongside TortoiseSVN, and colour-codes the added and removed lines.

Since Unified Diff is just a text format, you can use your favourite text editor if you prefer.

4.30.5.2. #合并工具

外部合并程序被用来解决冲突的文件。像差异查看程序那样，替代参数同样被用在命令行中。

%base
    没有被你或他人更改的原始文件

%bname
    原始文件的窗口标题

%mine
    你更改过的新文件

%yname
    你新文件的窗口标题

%theirs
    档案库中存放的文件

%tname
    档案库中文件的窗口标题

%merged
    发生冲突的文件，同时将被合并后的文件替换

%mname
    合并文件的窗口标题

For example, with Perforce Merge:

```
C:\Path-To\P4Merge.exe %base %theirs %mine %merged
```

or with KDiff3:

```
C:\Path-To\kdiff3.exe %base %mine %theirs -o %merged
    --L1 %bname --L2 %yname --L3 %tname
```

or with Araxis:

```
C:\Path-To\compare.exe /max /wait /3 /title1:%tname /title2:%bname
    /title3:%yname %theirs %base %mine %merged /a2
```

or with WinMerge (2.8 or later):

```
C:\Path-To\WinMerge.exe %merged
```

or with DiffMerge:

```
C:\Path-To\DiffMerge.exe -caption=%mname -result=%merged -merge
    -nosplash -t1=%yname -t2=%bname -t3=%tname %mine %base %theirs
```

4.30.5.3. #差异查看/合并工具的高级设置

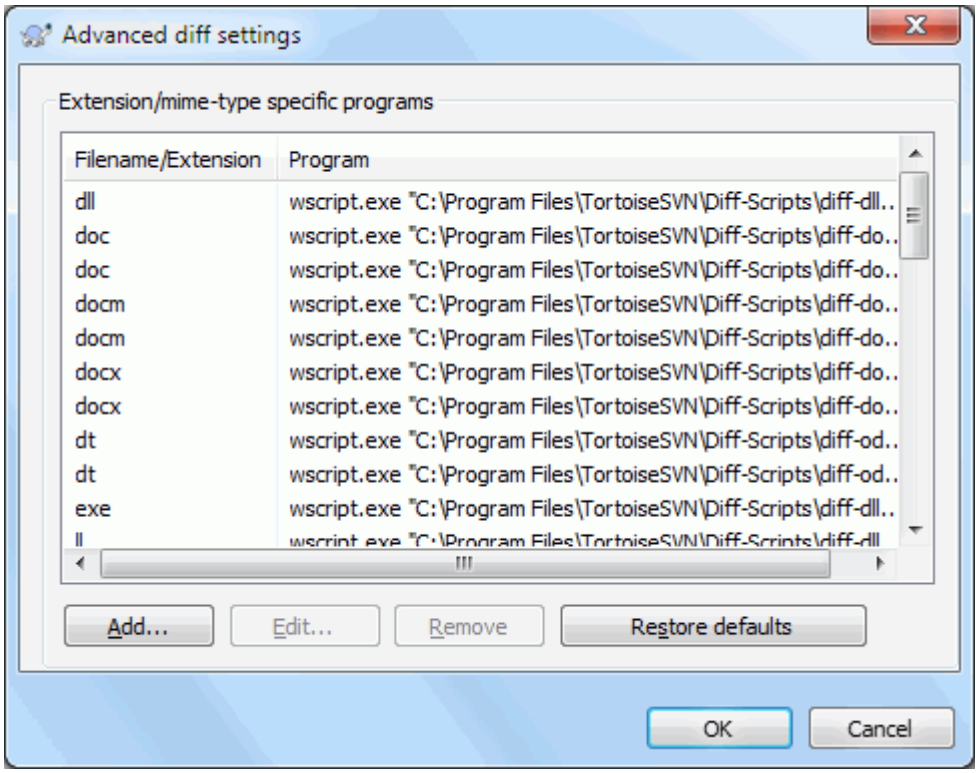


图 4.80. 高级差异比较设置/高级合并设置的对话框

在高级设置中，您可以为每种文件类型都定义一个不同的差异比较及合并程序。例如，您可以将 Photoshop 关联为 .jpg 文件的 “Diff” 程序 :-) 也可以将 svn:mime-type 属性同差异比较/合并程序关联起来。

为了使用文件扩展，你需要指定扩展。使用 .BMP 来描述 Windows 位图文件。如果使用svn:mime-type属性，要指定多媒体文件类型，包含斜线，例如text/xml。

4.30.6. #已保存数据的设置

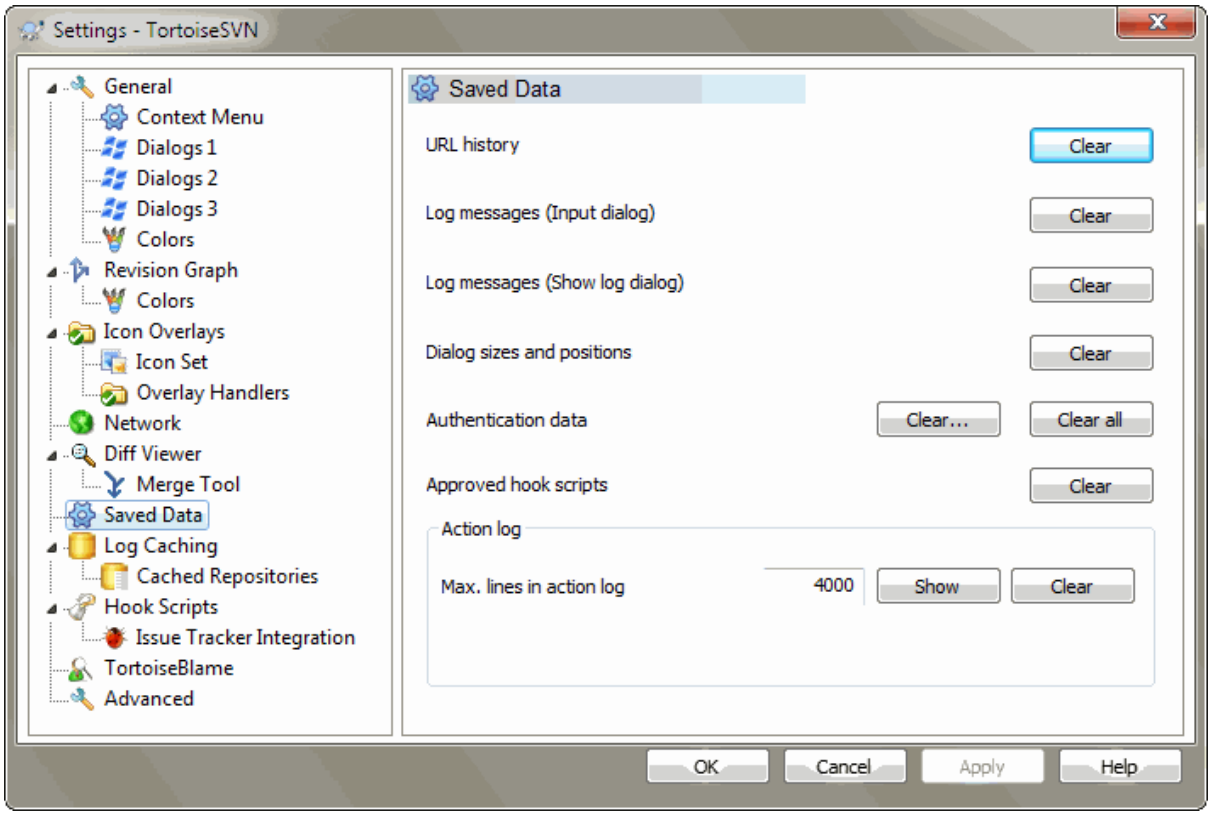


图 4.81. 设置对话框，已保存数据设置页面

为您方便着想，TortoiseSVN保存了很多你用过的设置，并记录你最近浏览过的地址。如果你想清空这些数据缓存，就在这里操作。

URL历史记录
: 每次你检出一个工作副本，合并那些更改的文件，或仅仅是在使用版本库浏览器时，TortoiseSVN都将保存一个记录，记录那些最近使用过的URL，并在一个下拉列表框中显示出来。有时列表会被逐渐增多的过期URL弄得乱糟糟的，所以有定期清理一下的必要。

  如果您想要从一个组合框中删除单个项目，只要点击箭头向下拉动组合框，将鼠标移动到要删除的项目上并按 Shift+Del。

日志信息(输入对话框)
: TortoiseSVN同时也储存你最近提交时填写的日志信息。对应每个版本库都要储存这些信息，所以如果你访问过很多版本库，这个列表将变得非常大。

日志信息(显示日志对话框)
: TortoiseSVN 将通过显示日志对话框而获取的日志消息缓存起来，从而节省下次显示日志的时间。如果其他人编辑了日志消息而您已缓存了该消息，您将无法看到更改，除非您清除缓存。可在 日志缓存 标签中启用日志消息缓存。

窗口大小及位置
: 许多对话框都可以记录你最后一次使用时的窗口大小和位置。

认证数据
: 当你在登陆某个Subversion服务器，填写认证信息时，用户名和密码也可以被保存在本地，你也就不用每次都输入了。但考虑到一些安全因素，你可能会有清除这些认证信息的愿望，或者你仅仅是想换个不同的用户名登陆...John知道你正在用他的机器么？(规范点儿，用你自己的用户名登陆版本库吧，伙计 *by Jax)

If you want to clear authentication data for one particular server only, use the Clear... instead of the Clear all button.

动作日志
TortoiseSVN 保存了一个日志，包含了进度对话框的一切写入。这很有用，比如您想要检查在最近的更新命令中都发生了什么。

日志文件有长度限制，文件过大时最旧的内容将被丢弃。默认保留 4000 行，但是您可以自定义该数字。

您可以从这里查看日志文件内容，也可清理它。

4.30.7. #日志缓存

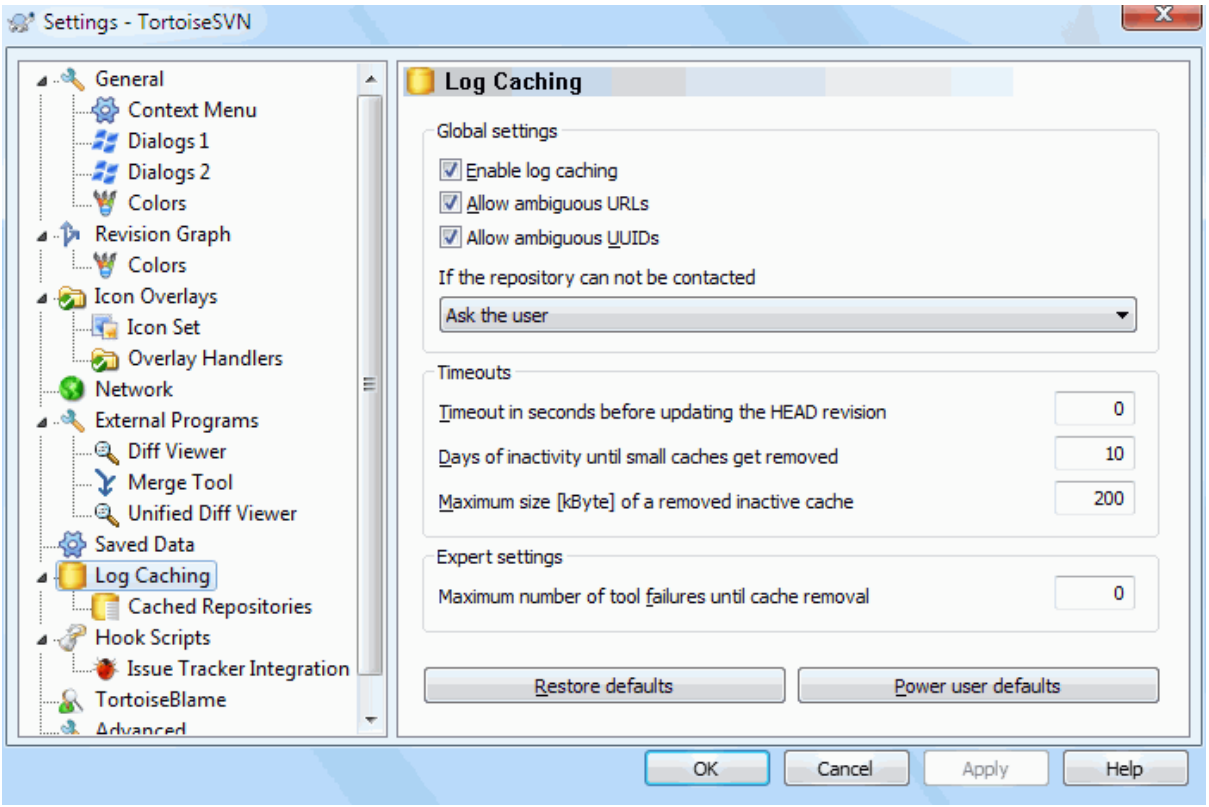


图 4.82. 设置对话框，日志缓存页面

此对话框允许您配置 TortoiseSVN 的日志缓存功能，其保留了日志消息和变更路径的本地副本以便避免从服务器耗时的下载过程。使用日志缓存可以大大加快日志对话框和版本图。另一个有用的功能就是在离线时仍然可以访问日志消息。

启用日志缓存
每当请求日志数据时启用日志缓存。如果选中，在可用时将从缓存中检索数据，任何不在缓存中的消息都将从服务器中检索并添加至缓存中。

如果禁用缓存，总是直接从服务器中检索数据并且不会存储在本地。

允许不明确的 URL
有时您可能必须连接到一个为所有版本库使用同一 URL 的服务器。svnbridge 的旧版本就会这样做。如果您需要访问此类版本库，就必须选中此选项。如果不需要，则不用选择以便提高性能。

允许不明确的 UUID
某些主机服务会为其所有版本库给予相同的 UUID。您甚至可能也这样做过，当通过复制一个版本库文件夹来创建一个新的版本库时。出于各种原因，这不是个好主意 - UUID 应该是唯一的。然

而，如果选中此复选框，日志缓存仍然会在这种情况下工作。如果不需要它，请不要选择以提高性能。

如果不能连接版本库
: 如果您正在离线工作或者版本库服务器已关闭，日志缓存仍可用来提供保留在缓存中的日志消息。当然缓存可能不是最新的，因此有选项可供您选择是否使用此功能。

  当从缓存中提取日志数据而不联系服务器时，使用这些消息的对话框将在标题栏中显示离线状态。

更新 HEAD 版本之前的超时时间
: 当您调用日志对话框时，您通常会希望联系服务器以检查是否有任何新的日志消息。如果此处设置的超时值非零，那么将只在自最后一次联系起超时时间已过时联系服务器。这样可以减少服务器往返过程，如果您经常打开日志对话框并且该服务器很缓慢的话，但所显示的数据可能无法完全保持最新。如果您想要使用此功能，我们建议使用值 300 (5 分钟) 作为权衡考量。

不活动的小缓存过期天数
: 如果您浏览过很多版本库，您将积累大量的日志缓存。如果您不常使用它们，缓存将不会变得很大，因此 TortoiseSVN 将在默认的设定时间后清除它们。使用此项目来控制缓存清除。

删除不活动缓存的最大尺寸
: 较大的缓存要重新获取可不容易，因此 TortoiseSVN 仅清除小的缓存。用此值来微调阈值。

在删除缓存之前的最大失败次数
: 有时缓存出错并导致崩溃。如果出现这种情况，缓存通常被自动删除以避免问题复发。如果您使用不太稳定的每日编译版本，您可以选择总是保留缓存。

4.30.7.1.#缓存的版本库

在此页面上，您可以看到已经缓存至本地的版本库列表，以及用作缓存的空间。如果您选择一个版本库，您就可以使用下面的按钮。

点击 更新 可完全刷新缓存并填满所有空隙。对于大版本库，这可能会耗用很长时间。但这很有用，比如您转至离线并希望获得最佳的可用缓存。

点击 导出 按钮以导出整个缓存作为 CSV 文件集。这很有用，比如您想使用外部程序来处理日志数据，尽管它主要用于开发人员。

点击 删除 为所选版本库删除所有缓存的数据。这不会禁用版本库缓存，因此您下次请求日志数据时，一个新的缓存将被创建。

4.30.7.2. #日志缓存统计

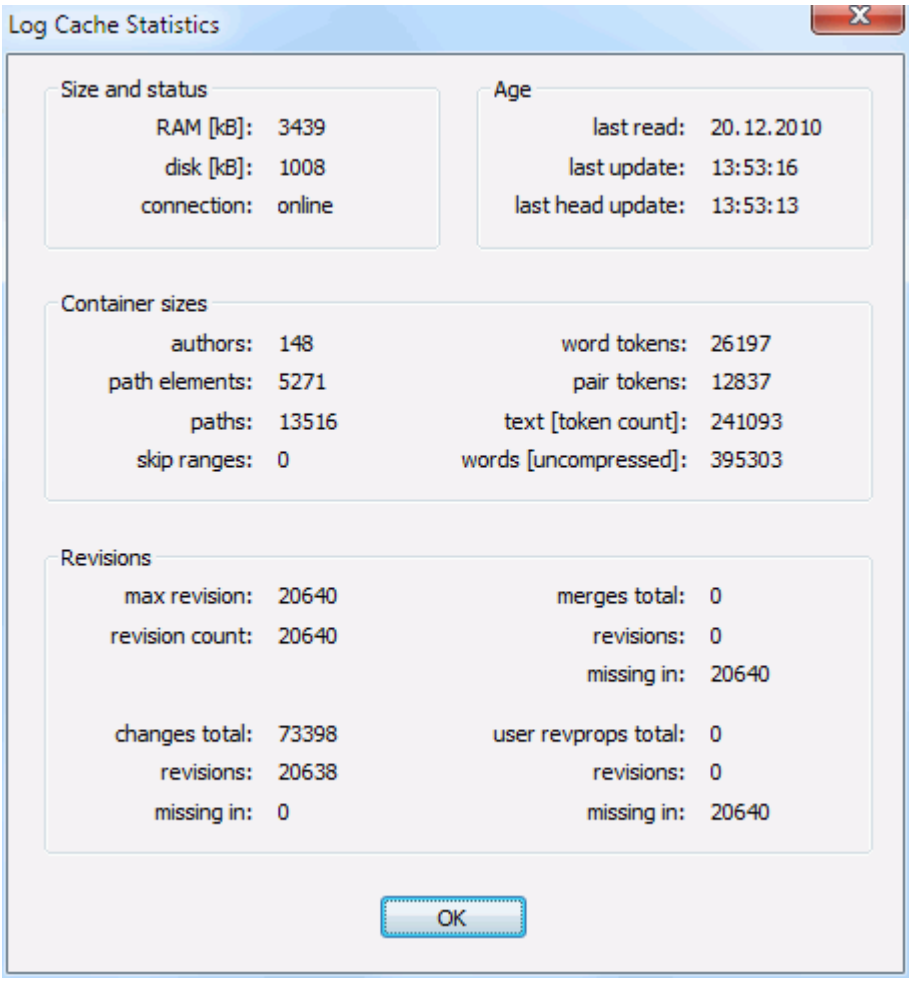


图 4.83. 设置对话框，日志缓存统计

点击 详细信息 按钮来查看特定缓存的详细统计。此处显示的许多字段主要对 TortoiseSVN 的开发者有意义，因此不会全部对它们进行详细说明。

RAM
缓存服务所需的内存量。

磁盘
用于缓存的磁盘空间量。数据被压缩，因此磁盘使用率通常相当适度。

连接
显示版本库在最后一次使用缓存时是否可用。

最近更新
最后一次缓存内容更改时间。

最后一次 HEAD 更新
最后一次我们从服务器请求 HEAD 版本的时间。

作者
缓存中记录的消息的不同作者的数量。

路径
列出的路径数量，正如您使用 svn log -v 所看到的一样。

跳过范围
: 我们还未获取的版本范围数量，只是因为它们还没有被请求。这是对缓存中空位数量的测量。

最大版本号
: 缓存中存储的最高版本号。

版本计数
: 存储在缓存中的版本数量。这是缓存完整性的另一项测量。

4.30.8. #客户端钩子脚本

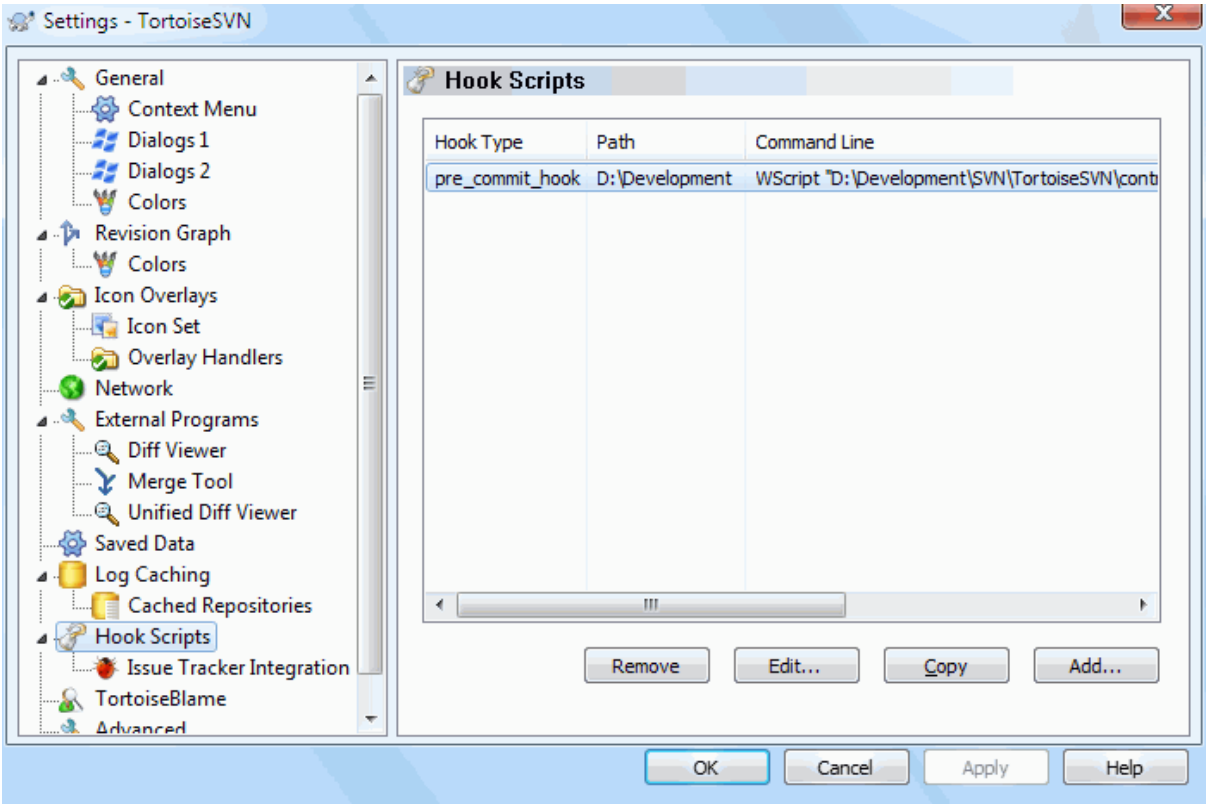


图 4.84. 设置对话框，钩子脚本页

这个对话框允许你指定当特定 Subversion 动作执行时，自动执行的钩子脚本。与 第 3.3 节 “服务器端钩子脚本” 中说明的钩子脚本相反，这些脚本在客户端本地执行。

应用程序，例如钩子，可能调用如SubWCRev.exe这样的程序，来更新提交后的版本号，可能还会出发重新构建。

Note that you can also specify such hook scripts using special properties on your working copy. See the section 第 4.17.2 节 “TortoiseSVN 项目属性”for details.

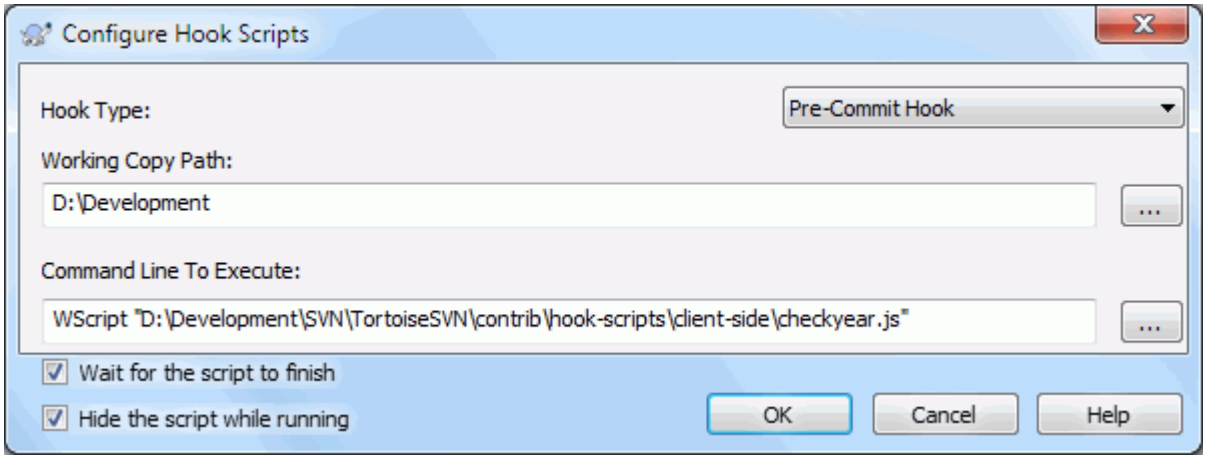


图 4.85. 设置对话框，配置钩子脚本页面

要增加钩子脚本，直接点击 增加 ，然后输入脚本即可。

现在有六种钩子脚本类型可用

开始提交
在提交对话框显示之前被调用.当钩子修改了一个受版本控制的文件,而且影响了需要提交的文件或者要提交的内容,可能需要使用它.然而,需要注意,钩子是在之前的某个时候被调用的,所以被选中提交的完整文件列表是不可用的.

提交之前
在用户点击了提交对话框的确定按钮,并且实际的提交过程开始后被调用.钩子包含所有要提交的内容的详细信息.

提交之后
在提交结束后调用(无论成功或失败)

开始更新
在更新到版本对话框显示之前调用

更新之前
在 Subversion 真正开始更新或切换前调用。

更新之后
在更新、切换或检出完成（无论成功与否）后调用。

连接之前
在尝试连接版本库前调用。每五分钟最多调用一次。

为特定工作目录定义的钩子。你只要指定顶级路径；如果在子目录内执行提交，TortoiseSVN 会自动向上搜索匹配路径。

接下来需要指定要执行的命令行,以钩子脚本或可执行文件的路径开始.它可以是批处理文件,可执行文件,或者有效的windows关联的其它文件,例如Perl脚本.注意,此命令不能使用UNC路径,Windows Shell由于安全限制而不允许这样的命令执行.

命令行包含一些由TortoiseSVN填写的参数,这些参数依赖于调用的钩子.每个钩子有对应的参数,按照以下的顺序调用:

开始提交
PATHMESSAGEFILECWD

提交之前
    PATHDEPTHMESSAGEFILECWD

提交之后
    PATHDEPTHMESSAGEFILEREVISIONERRORCWD

开始更新
    PATHCWD

更新之前
    PATHDEPTHREVISIONCWD

更新之后
    PATHDEPTHREVISIONERRORCWD

连接之前
    no parameters are passed to this script. You can pass a custom parameter by appending it to the script path.

上述各参数的含义如下：

PATH
    指向临时文件的路径，此文件包含了操作开始时的所有路径。在临时文件中，每个路径占一行。

    Note that for operations done remotely, e.g. in the repository browser, those paths are not local paths but the urls of the affected items.

DEPTH
    提交/更新的深度。

    可能的取值是：

    -2
        svn_depth_unknown

    -1
        svn_depth_exclude

    0
        svn_depth_empty

    1
        svn_depth_files

    2
        svn_depth_immediates

    3
        svn_depth_infinity

MESSAGEFILE
    指向包含日志信息的提交文件.此文件使用UTF-8编码.在成功执行开始提交钩子后,日志信息会回显,以便于钩子修改.

REVISION
    更新或提交完成后的版本库的版本

ERROR
    指到包含错误信息的文件的路径， 如果没有错误的话，文件将是空的

CWD
脚本正在运行的工作目录,设置为所有受影响的路径的公用根目录.

注意,尽管为了方便提供了这些参数名称,但是设置钩子的时候不是必须使用它们.不管是否需要,为特定钩子列出的参数都会传递. ;-)

如果你想Subversion 操作直到钩子完成才结束，就选择等待脚本结束。

当运行脚本的时候通常希望隐藏DOS窗口,所以默认检查隐藏运行中的脚本

钩子脚本实例可以在 TortoiseSVN 代码库 [http://tortoisesvn.googlecode.com/svn/trunk/contrib/hook-scripts]的 contrib文件夹下面找到.(第 3 节 “许可协议解释了如何访问这个代码库.)

When debugging hook scripts you may want to echo progress lines to the DOS console, or insert a pause to stop the console window disappearing when the script completes. Because I/O is redirected this will not normally work. However you can redirect input and output explicitly to CON to overcome this. e.g.

```
echo Checking Status > con
pause < con > con
```

在TortoiseSVN 的安装文件夹下有一个名为 ConnectVPN.exe的小工具.在TortoiseSVN连接到代码库之前,可以使用这个连接前的钩子自动连接到VPN.

4.30.8.1. #问题跟踪器集成

TortoiseSVN可以使用COM插件在提交对话框中查询问题跟踪器.在第 4.28.2 节 “从问题跟踪器中获取信息”中描述了这类插件的使用.如果系统管理员提供了插件,用户也安装和注册了,这里就是指定插件和工作副本相结合的地方.

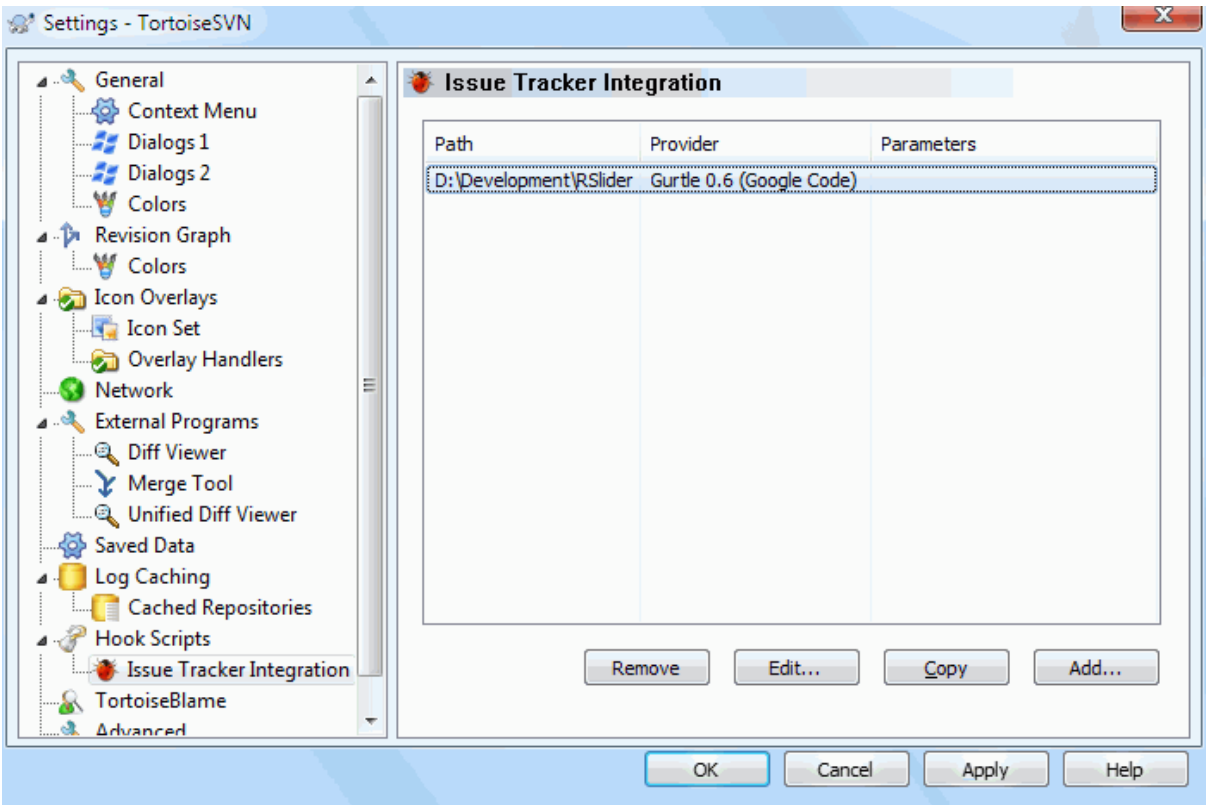


图 4.86. 设置对话框，问题跟踪集成页

点击 增加...按钮,在工作副本上使用这个插件.可以指定工作副本的路径.可以在所有注册过的问题跟踪器插件列表中,选择要使用的插件和要传递的参数.这些参数指定到插件,但是可能包含在问题跟踪器上的用户名,这样插件才能查找分配给用户的问题.

如果希望项目中的所有的用户使用同样的COM组件,可以为插件指定如下的属性,bugtraq:provideruuid, bugtraq:provideruuid64 和bugtraq:providerparams.

bugtraq:provideruuid
这个属性为COM组件的IBugtraqProvider接口指定UUID,例如, {91974081-2DC7-4FB1-B3BE-0DE1C8D6CE4E}.(这个例子是 Gurtle bugtraq provider [http://code.google.com/p/gurtle/]的UUID,它是Google Code [http://code.google.com/hosting/]的问题追踪器.)

bugtraq:provideruuid64
这和 bugtraq:provideruuid相同,但是是针对IBugtraqProvider接口的64位版本.

bugtraq:providerparams
这个属性指定了传递给IBugtraqProvider的参数.

请检查IBugtraqProvider 插件的相关文档,查明如何设置这两个属性.

4.30.9.#TortoiseBlame 的设置

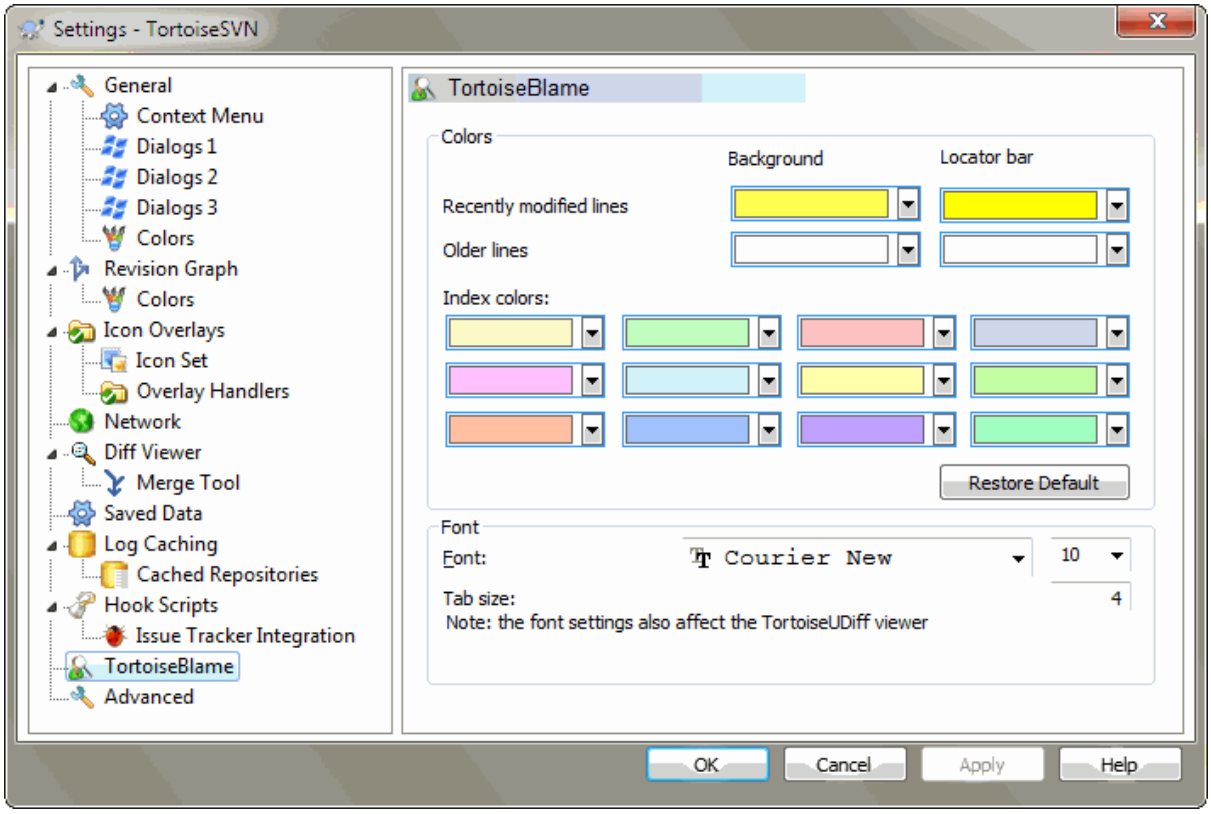


图 4.87. 设置对话框，TortoiseBlame 页面

TortoiseBlame 使用的配置被主上下文菜单控制，不是被 TortoiseBlame 自己直接控制。

颜色
TortoiseBlame 可以使用背景色指示文件中行的年龄。你设置最新和最旧版本的颜色后，TortoiseBlame 使用线性插补算法根据每行的版本设置其颜色。

You can specify different colours to use for the locator bar. The default is to use strong contrast on the locator bar while keeping the main window background light so that you can still read the text.

字体
你可以选择显示文本的字体和大小。它同时对文件内容，在左窗格显示的作者和版本信息等生效。

制表
定义在文件中出现的制表字符用多少空格扩展。

4.30.10. #高级设置

一些不常用的设置只在设置对话框的高级页面上,这些设置直接修改注册表.需要知道这些设置是做什么用的以及它们做出了什么改动.除非确定需要修改这些设置,否则不要修改它们.

AllowAuthSave
Sometimes multiple users use the same account on the same computer. In such situations it's not really wanted to save the authentication data. Setting this value to false disables the save authentication button in the authentication dialog.

AllowUnversionedObstruction
如果版本更新从版本库增加了一个新的文件,而此文件在本地是非版本控制的文件,默认的处理是保留本地文件,同时把它当作新文件的一个(可能的)修改版本.如果更希望TortoiseSVN在此情况下产生一个冲突,设置参数值为 false.

AlwaysExtendedMenu
和桌面一样,当上下文菜单打开的时候,按下Shift键,TortoiseSVN就会显示附件的命令.要强制TortoiseSVN总是显示这些拓展的命令,需要设置参数值为 true.

AutoCompleteMinChars
The minimum amount of chars from which the editor shows an auto-completion popup. The default value is 3.

AutocompleteRemovesExtensions
The auto-completion list shown in the commit message editor displays the names of files listed for commit. To also include these names with extensions removed, set this value to true.

BlockStatus
当另一个TortoiseSVN命令(如,更新,提交,...)正在执行的时候,如果不希望图形管理器更新TortoiseSVN图标状态,可以设置参数值为true.

CacheTrayIcon
要为TSVNCache程序增加一个缓存托盘图标,可以设置这个值为true.这实际只是对开发人员有作用,它允许开发人员以优雅的方式终止程序.

ColumnsEveryWhere
TortoiseSVN增加到明细查看窗口里额外的列,通常只在工作副本显示.如果希望到处都可以访问,而不仅仅是在工作副本,设置参数值为true. 需要注意的是,多余的列只在XP操作系统可用,Vista和后续的版本不再支持这个特性.

ConfigDir
您可以为 Subversion 配置文件指定其他位置。这将影响到 TortoiseSVN 的所有操作。

CtrlEnter
In most dialogs in TortoiseSVN, you can use Ctrl+Enter to dismiss the dialog as if you clicked on the OK button. If you don't want this, set this value to false.

Debug
如果希望在每一次在启动TortoiseProc.exe的命令行时都弹出对话框,可以设置这为true.

DebugOutputString
Set this to true if you want TortoiseSVN to print out debug messages during execution. The messages can be captured with special debugging tools only.

DialogTitles
The default format (value of 0) of dialog titles is url/path - name of dialog - TortoiseSVN. If you set this value to 1, the format changes to name of dialog - url/path - TortoiseSVN.

DiffBlamesWithTortoiseMerge
TortoiseSVN allows you to assign an external diff viewer. Most such viewers, however, are not suited for change blaming (第 4.23.2 节 “追溯不同点”), so you might wish to fall back to TortoiseMerge in this case. To do so, set this value to true.

FixCaseRenames
Some apps change the case of filenames without notice but those changes aren't really necessary nor wanted. For example a change from file.txt to FILE.TXT wouldn't bother normal Windows applications, but Subversion is case sensitive in these situations. So TortoiseSVN automatically fixes such case changes.

If you don't want TortoiseSVN to automatically fix such case changes for you, you can set this value to false.

FullRowSelect
这个状态列表控件在多个对话框中使用(例如,commit, check-for-modifications, add, revert, ...),可以选择整行(例如,如果选择一个项,这一行会被选中,而不仅仅是这一列).这很好,但是选中的行也覆盖了底部右侧的背景图片.设置参数值false可以取消选中整行.

GroupTaskbarIconsPerRepo
This option determines how the Win7 taskbar icons of the various TortoiseSVN dialogs and windows are grouped together. This option has no effect on Windows XP or Vista!

1. The default value is 0. With this setting, the icons are grouped together by application type. All dialogs from TortoiseSVN are grouped together, all windows from TortoiseMerge are grouped together, ...

图 4.88. Taskbar with default grouping

2. If set to 1, then instead of all dialogs in one single group per application, they're grouped together by repository. For example, if you have a log dialog and a commit dialog open for repository A, and a check-for-modifications dialog and a log dialog for repository B, then there are two application icon groups shown in the Win7 taskbar, one group for each repository. But TortoiseMerge windows are not grouped together with TortoiseSVN dialogs.

图 4.89. Taskbar with repository grouping

3. If set to 2, then the grouping works as with the setting set to 1, except that TortoiseSVN, TortoiseMerge, TortoiseBlame, TortoiseIDiff and TortoiseUDiff windows are all grouped together. For example, if you have the commit dialog open and then double click on a modified file, the opened TortoiseMerge diff window will be put in the same icon group on the taskbar as the commit dialog icon.

图 4.90. Taskbar with repository grouping

4. If set to 3, then the grouping works as with the setting set to 1, but the grouping isn't done according to the repository but according to the working copy. This is useful if you have all your projects in the same repository but different working copies for each project.

5. If set to 4, then the grouping works as with the setting set to 2, but the grouping isn't done according to the repository but according to the working copy.

HideExternalInfo
If this is set to false, then every svn:externals is shown during an update separately.

If it is set to true (the default), then update information for externals is only shown if the externals are affected by the update, i.e. changed in some way. Otherwise nothing is shown as with normal files and folders.

GroupTaskbarIconsPerRepoOverlay
This has no effect if the option GroupTaskbarIconsPerRepo is set to 0 (see above).

If this option is set to true, then every icon on the Win7 taskbar shows a small colored rectangle overlay, indicating the repository the dialogs/windows are used for.

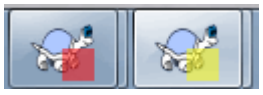


图 4.91. Taskbar grouping with repository color overlays

IncludeExternals
By default, TortoiseSVN always runs an update with externals included. This avoids problems with inconsistent working copies. If you have however a lot of externals set, an update can take quite a while. Set this value to false to run the default update with externals excluded. To update with externals included, either run the Update to revision... dialog or set this value to true again.

LogFindCopyFrom
When the log dialog is started from the merge wizard, already merged revisions are shown in gray, but revisions beyond the point where the branch was created are also shown. These revisions are shown in black because those can't be merged.

If this option is set to true then TortoiseSVN tries to find the revision where the branch was created from and hide all the revisions that are beyond that revision. Since this can take quite a while, this option is disabled by default. Also this option doesn't work with some SVN servers (e.g., Google Code Hosting, see issue #5471 [http://code.google.com/p/support/issues/detail?id=5471]).

LogStatusCheck
日志对话框显示了版本修订信息,其中工作路径以粗体显示.这需要日志对话框获取此路径的状态信息.对于一个非常大的工作副本来来说,这可能需要一些时间.设置参数值为false可以使这个特性无效.

MergeLogSeparator
When you merge revisions from another branch, and merge tracking information is available, the log messages from the revisions you merge will be collected to make up a commit log message. A pre-defined string is used to separate the individual log messages of the merged revisions. If you prefer, you can set this to a value containing a separator string of your choice.

NumDiffWarning
If you want to show the diff at once for more items than specified with this settings, a warning dialog is shown first. The default is 10.

OldVersionCheck
不管是否有新版本可用,TortoiseSVN默认每周检查一次更新.如果有可用的更新,提交对话框会显示链接信息.如果更希望按照以前的方式来显示,也就是弹出对话框通知有更新,设置参数值为 true.

RepoBrowserTrySVNParentPath
The repository browser tries to fetch the web page that's generated by an SVN server configured with the SVNParentPath directive to get a list of all repositories. To disable that behavior, set this value to false.

ScintillaDirect2D
This option enables the use of Direct2D accelerated drawing in the Scintilla control which is used as the edit box in e.g. the commit dialog, and also for the unified diff viewer. With some graphic cards however this sometimes doesn't work properly so that the cursor to enter text isn't always visible. If that happens, you can turn this feature off by setting this value to false.

OutOfDateRetry
If you don't want TortoiseSVN to ask you to update the working copy automatically after an Out of date error, set this value to false.

ShellMenuAccelerators
TortoiseSVN 为上下文菜单设置快捷键.这可能导致快捷键重复(例如,SVN Commit使用快捷键Alt-C,但是这也是 Copy的快捷键).如果不希望使用或者不需要使用TortoiseSVN快捷键,设置参数值为false.

ShowContextMenuIcons
如果使用windows桌面或者遇到上下文菜单显示不正确的问题时,这很有用.如果不希望TortoiseSVN为shell上下文菜单选项显示图标,设置参数值为false.设置参数值为true则可以再次显示图标.

ShowAppContextMenuIcons
如果不希望TortoiseSVN为上下文菜单对话框显示图标,设置参数值为false.

StyleCommitMessages
提交和日志对话框在提交信息中使用不同的字体风格(例如,粗体,斜体) (详情请见第 4.4.5 节“提交日志信息”).如果不希望这样显示,可以设置参数值为 false.

UpdateCheckURL
这个值包含了URL,TortoiseSVN尝试从这个URL下载文本文件,以确定是否有可用的更新.这对于公司的管理员来说可能有用.他们更希望核准后才让用户更新TortoiseSVN.

VersionCheck
不管是否有新版本可用,TortoiseSVN默认每周检查一次更新.如果不希望TortoiseSVN做此检查,设置参数值为false.

4.30.11.#正在导出TSVN设置

If you want to export all your client settings to use on another computer you can do so using the Windows registry editor regedt32.exe. Go to the registry key HKCU\Software\TortoiseSVN and export it to a reg file. On the other computer, just import that file again (usually, a double click on the reg file will do that).

Remember to save Subversion's general settings, which you can find in the Subversion configuration file %APPDATA%\Subversion\config.

4.31.#最后步骤

捐赠!

Even though TortoiseSVN and TortoiseMerge are free, you can support the developers by sending in patches and playing an active role in the development. You can also help to cheer us up during the endless hours we spend in front of our computers.

While working on TortoiseSVN we love to listen to music. And since we spend many hours on the project we need a lot of music. Therefore we have set up some wish-lists with our favourite music CDs and DVDs: http://tortoisesvn.net/donate.html Please also have a look at the list of people who contributed to the project by sending in patches or translations.

第#5#章#SubWCRev 程序

SubWCRev是Windows的命令行工具，可以阅读Subversion工作副本的状态，可以在模版中随意执行关键字替换。这通常是构建过程的一部分，将工作副本信息结合到创建的对象当中。通常情况下，可能是用来将修订版本号存入“关于”窗口。

5.1.#SubWCRev 命令行

SubWCRev reads the Subversion status of all files in a working copy, excluding externals by default. It records the highest commit revision number found, and the commit timestamp of that revision, it also records whether there are local modifications in the working copy, or mixed update revisions. The revision number, update revision range and modification status are displayed on stdout.

SubWCRev.exe从命令行或脚本中运行，使用命令行参数控制。

SubWCRev WorkingCopyPath [SrcVersionFile DstVersionFile] [-nmdfe]

WorkingCopyPath是要检查的工作副本路径，你可以只对工作副本使用SubWCRev，而不是直接对版本库，这个路径可以是绝对路径，也可以是工作目录的相对路径。

如果你想让SubWCRev执行关键字替换，象版本库版本，地址等字段保存到文本文件，就需要提供一个模版文件SrcVersionFile，输出文件DstVersionFile就是模版替换之后的版本。

There are a number of optional switches which affect the way SubWCRev works. If you use more than one, they must be specified as a single group, e.g. -nm, not -n -m.

切换	描述
-n	If this switch is given, SubWCRev will exit with ERRORLEVEL 7 if the working copy contains local modifications. This may be used to prevent building with uncommitted changes present.
-N	If this switch is given, SubWCRev will exit with ERRORLEVEL 11 if the working copy contains unversioned items that are not ignored.
-m	If this switch is given, SubWCRev will exit with ERRORLEVEL 8 if the working copy contains mixed revisions. This may be used to prevent building with a partially updated working copy.
-d	If this switch is given, SubWCRev will exit with ERRORLEVEL 9 if the destination file already exists.
-f	如果给出这个开关，SubWCRev 就会包含文件夹的最后修改版本。默认行为是取得版本号时只考虑文件。
-e	If this switch is given, SubWCRev will examine directories which are included with svn:externals, but only if they are from the same repository. The default behaviour is to ignore externals.
-E	If this switch is given, same as -e, but it ignores the externals with explicit revisions, when the revision range inside of them is only the given explicit revision in the properties. So it doesn't lead to mixed revisions.
-x	如果给出这个开关，SubWCRev 就会以十六进制输出修订版本号。
-X	如果给出这个开关，SubWCRev 就会以十六进制输出修订版本号，并且加上 'OX' 前缀。
-q	If this switch is given, SubWCRev will perform the keyword substitution without showing working copy status on stdout.

表 5.1. 列出可用的命令行开关

If there is no error, SubWCRev returns zero. But in case an error occurs, the error message is written to stderr and shown in the console. And the returned error codes are:

Error Code	描述
1	Syntax error. One or more command line parameters are invalid.
2	The file or folder specified on the command line was not found.
3	The input file could not be opened, or the target file could not be created.
4	Could not allocate memory. This could happen if e.g. the source file is too big.
5	The source file can not be scanned properly.
6	SVN error: Subversion returned with an error when SubWCRev tried to find the information from the working copy.
7	The working copy has local modifications. This requires the -n switch.
8	The working copy has mixed revisions. This requires the -m switch.
9	The output file already exists. This requires the -d switch.
10	The specified path is not a working copy or part or one.
11	此工作副本中有未版本控制的文件或文件夹。这需要 -N 切换。

表 5.2. List of SubWCRev error codes

5.2. #关键字替换

如果提供了源文件和目的文件，SubWCRev 会复制源文件到目标文件，执行如下所属的关键字替换:

关键字	描述
$WCREV$	用工作副本中最高的提交版本来替换
$WCREV&$	Replaced with the highest commit revision in the working copy, ANDed with the value after the & char. For example: $WCREV&0xFFFF$
$WCREV-$, $WCREV+$	Replaced with the highest commit revision in the working copy, with the value after the + or - char added or subtracted. For example: $WCREV-1000$
$WCDATE$, $WCDATEUTC$	Replaced with the commit date/time of the highest commit revision. By default, international format is used: yyyy-mm-dd hh:mm:ss. Alternatively, you can specify a custom format which will be used with strftime(), for example: $WCDATE=%a %b %d %I:%M:%S %p$. For a list of available formatting characters, look at the 在线引用 [http://msdn.microsoft.com/en-us/library/fe06s4ak.aspx].
$WCNOW$, $WCNOWUTC$	Replaced with the current system date/time. This can be used to indicate the build time. Time formatting can be used as described for $WCDATE$.
$WCRANGE$	Replaced with the update revision range in the working copy. If the working copy is in a consistent state, this will be a single revision. If the working copy contains mixed revisions, either due to being out of date, or due to a deliberate update-to-revision, then the range will be shown in the form 100:200.
$WCMIXED$	当有混合版本时用 TText 替换 $WCMIXED?TText:FText$，否则用 FText 替换。

关键字	描述
$WCMODS$	若本地存在修改，就用 TText 替换 $WCMODS?TText:FText$，否则用 FText 替换。
$WCUNVER$	$WCUNVER?TText:FText$ is replaced with TText if there are unversioned items in the working copy, or FText if not.
$WCEXTALLFIXED$	$WCEXTALLFIXED?TText:FText$ is replaced with TText if all externals are fixed to an explicit revision, or FText if not.
$WCISTAGGED$	$WCISTAGGED?TText:FText$ is replaced with TText if the repository URL contains the tags classification pattern, or FText if not.
$WCURL$	用传递给SubWCRev的工作目录的版本库地址替换。
$WCINSVN$	$WCINSVN?TText:FText$ is replaced with TText if the entry is versioned, or FText if not.
$WCNEEDSLOCK$	$WCNEEDSLOCK?TText:FText$ is replaced with TText if the entry has the svn:needs-lock property set, or FText if not.
$WCISLOCKED$	$WCISLOCKED?TText:FText$ is replaced with TText if the entry is locked, or FText if not.
$WCLOCKDATE$, $WCLOCKDATEUTC$	Replaced with the lock date. Time formatting can be used as described for $WCDATE$.
$WCLOCKOWNER$	Replaced with the name of the lock owner.
$WCLOCKCOMMENT$	Replaced with the comment of the lock.

表 5.3. List of available keywords

SubWCRev does not directly support nesting of expressions, so for example you cannot use an expression like:

```
#define SVN_REVISION    "$WCMIXED?$WCRANGE$:$WCREV$$"
```

But you can usually work around it by other means, for example:

```
#define SVN_RANGE       $WCRANGE$
#define SVN_REV         $WCREV$
#define SVN_REVISION    "$WCMIXED?SVN_RANGE:SVN_REV$"
```

Some of these keywords apply to single files rather than to an entire working copy, so it only makes sense to use these when SubWCRev is called to scan a single file. This applies to $WCINSVN$, $WCNEEDSLOCK$, $WCISLOCKED$, $WCLOCKDATE$, $WCLOCKOWNER$ and $WCLOCKCOMMENT$.

5.3. #关键字例子

下面的例子显示了模版文件中的关键字是如何在输出文件中被替换的。

```
// Test file for SubWCRev

char *Revision      = "$WCREV$";
char *Revision16    = "$WCREV&0xFF$";
char *Revisionp100  = "$WCREV+100$";
char *Revisionm100  = "$WCREV-100$";
```

```
char *Modified      = "$WCMODS?Modified:Not modified$";
char *Unversioned   = "$WCUNVER?Unversioned items found:no unversioned items$";
char *Date          = "$WCDATE$";
char *CustDate      = "$WCDATE=%a, %d %B %Y$";
char *DateUTC       = "$WCDATEUTC$";
char *CustDateUTC   = "$WCDATEUTC=%a, %d %B %Y$";
char *TimeNow       = "$WCNOW$";
char *TimeNowUTC    = "$WCNOWUTC$";
char *RevRange      = "$WCRANGE$";
char *Mixed         = "$WCMIXED?Mixed revision WC:Not mixed$";
char *ExtAllFixed   = "$WCEXTALLFIXED?All externals fixed:Not all externals fixed$";
char *IsTagged      = "$WCISTAGGED?Tagged:Not tagged$";
char *URL           = "$WCURL$";
char *isInSVN       = "$WCINSVN?versioned:not versioned$";
char *needslck      = "$WCNEEDSLOCK?TRUE:FALSE$";
char *islocked      = "$WCISLOCKED?locked:not locked$";
char *lockdateutc   = "$WCLOCKDATEUTC$";
char *lockdate      = "$WCLOCKDATE$";
char *lockcustutc   = "$WCLOCKDATEUTC=%a, %d %B %Y$";
char *lockcust      = "$WCLOCKDATE=%a, %d %B %Y$";
char *lockown       = "$WCLOCKOWNER$";
char *lockcmt       = "$WCLOCKCOMMENT$";

#if $WCMODS?1:0$
#error Source is modified
#endif

// End of file
```

After running SubWCRev.exe path\to\workingcopy testfile.tmpl testfile.txt, the output file testfile.txt would looks like this:

```
// Test file for SubWCRev

char *Revision      = "22837";
char *Revision16    = "53";
char *Revisionp100  = "22937";
char *Revisionm100  = "22737";
char *Modified      = "Modified";
char *Unversioned   = "no unversioned items";
char *Date          = "2012/04/26 18:47:57";
char *CustDate      = "Thu, 26 April 2012";
char *DateUTC       = "2012/04/26 16:47:57";
char *CustDateUTC   = "Thu, 26 April 2012";
char *TimeNow       = "2012/04/26 20:51:17";
char *TimeNowUTC    = "2012/04/26 18:51:17";
char *RevRange      = "22836:22837";
char *Mixed         = "Mixed revision WC";
char *ExtAllFixed   = "All externals fixed";
char *IsTagged      = "Not tagged";
char *URL           = "https://tortoisesvn.googlecode.com/svn/trunk";
char *isInSVN       = "versioned";
char *needslck      = "FALSE";
char *islocked      = "not locked";
char *lockdateutc   = "1970/01/01 00:00:00";
char *lockdate      = "1970/01/01 01:00:00";
char *lockcustutc   = "Thu, 01 January 1970";
```

```
char *lockcust      = "Thu, 01 January 1970";
char *lockown       = "";
char *lockcmt       = "";

#if 1
#error Source is modified
#endif

// End of file
```

A file like this will be included in the build so you would expect it to be versioned. Be sure to version the template file, not the generated file, otherwise each time you regenerate the version file you need to commit the change, which in turn means the version file needs to be updated.

5.4. #COM 接口

如果你需要在其它程序中访问 Subversion 版本信息，可以使用 SubWCRev 的 COM 接口。创建的对象名称是 SubWCRev.object，支持下述方法：

方法	描述
.GetWCInfo	This method traverses the working copy gathering the revision information. Naturally you must call this before you can access the information using the remaining methods. The first parameter is the path. The second parameter should be true if you want to include folder revisions. Equivalent to the -f command line switch. The third parameter should be true if you want to include svn:externals. Equivalent to the -e command line switch.
.GetWCInfo2	The same as GetWCInfo() but with a fourth parameter that sets the equivalent to the -E command line switch.
.Revision	The highest commit revision in the working copy. Equivalent to $WCREV$.
.Date	The commit date/time of the highest commit revision. Equivalent to $WCDATE$.
.Author	最高提交版本的作者，也就是工作副本中最后提交修改的人。
.MinRev	最小的更新版本，在 $WCRANGE$ 中描述
.MaxRev	最大的更新版本，在 $WCRANGE$ 中描述
.HasModifications	若本地存在修改，就为真
.HasUnversioned	如果是未版本控制项目则为真
.Url	Replaced with the repository URL of the working copy path used in GetWCInfo. Equivalent to $WCURL$.
.IsSvnItem	True if the item is versioned.
.NeedsLocking	True if the item has the svn:needs-lock property set.
.IsLocked	True if the item is locked.
.LockCreationDate	String representing the date when the lock was created, or an empty string if the item is not locked.
.LockOwner	String representing the lock owner, or an empty string if the item is not locked.
.LockComment	The message entered when the lock was created.

表 5.4. 支持 COM/自动化 方法

下述例子显示了如何使用接口。

```
// testCOM.js - javascript file
// test script for the SubWCRev COM/Automation-object

filesystem = new ActiveXObject("Scripting.FileSystemObject");

revObject1 = new ActiveXObject("SubWCRev.object");
revObject2 = new ActiveXObject("SubWCRev.object");
revObject3 = new ActiveXObject("SubWCRev.object");
revObject4 = new ActiveXObject("SubWCRev.object");

revObject1.GetWCInfo(
    filesystem.GetAbsolutePathName("."), 1, 1);
revObject2.GetWCInfo(
    filesystem.GetAbsolutePathName(".."), 1, 1);
revObject3.GetWCInfo(
    filesystem.GetAbsolutePathName("SubWCRev.cpp"), 1, 1);
revObject4.GetWCInfo2(
    filesystem.GetAbsolutePathName("..\\.."), 1, 1, 1);

wcInfoString1 = "Revision = " + revObject1.Revision +
                "\nMin Revision = " + revObject1.MinRev +
                "\nMax Revision = " + revObject1.MaxRev +
                "\nDate = " + revObject1.Date +
                "\nURL = " + revObject1.Url + "\nAuthor = " +
                revObject1.Author + "\nHasMods = " +
                revObject1.HasModifications + "\nIsSvnItem = " +
                revObject1.IsSvnItem + "\nNeedsLocking = " +
                revObject1.NeedsLocking + "\nIsLocked = " +
                revObject1.IsLocked + "\nLockCreationDate = " +
                revObject1.LockCreationDate + "\nLockOwner = " +
                revObject1.LockOwner + "\nLockComment = " +
                revObject1.LockComment;
wcInfoString2 = "Revision = " + revObject2.Revision +
                "\nMin Revision = " + revObject2.MinRev +
                "\nMax Revision = " + revObject2.MaxRev +
                "\nDate = " + revObject2.Date +
                "\nURL = " + revObject2.Url + "\nAuthor = " +
                revObject2.Author + "\nHasMods = " +
                revObject2.HasModifications + "\nIsSvnItem = " +
                revObject2.IsSvnItem + "\nNeedsLocking = " +
                revObject2.NeedsLocking + "\nIsLocked = " +
                revObject2.IsLocked + "\nLockCreationDate = " +
                revObject2.LockCreationDate + "\nLockOwner = " +
                revObject2.LockOwner + "\nLockComment = " +
                revObject2.LockComment;
wcInfoString3 = "Revision = " + revObject3.Revision +
                "\nMin Revision = " + revObject3.MinRev +
                "\nMax Revision = " + revObject3.MaxRev +
                "\nDate = " + revObject3.Date +
                "\nURL = " + revObject3.Url + "\nAuthor = " +
                revObject3.Author + "\nHasMods = " +
                revObject3.HasModifications + "\nIsSvnItem = " +
                revObject3.IsSvnItem + "\nNeedsLocking = " +
                revObject3.NeedsLocking + "\nIsLocked = " +
                revObject3.IsLocked + "\nLockCreationDate = " +
```

```
                revObject3.LockCreationDate + "\nLockOwner = " +
                revObject3.LockOwner + "\nLockComment = " +
                revObject3.LockComment;
wcInfoString4 = "Revision = " + revObject4.Revision +
                "\nMin Revision = " + revObject4.MinRev +
                "\nMax Revision = " + revObject4.MaxRev +
                "\nDate = " + revObject4.Date +
                "\nURL = " + revObject4.Url + "\nAuthor = " +
                revObject4.Author + "\nHasMods = " +
                revObject4.HasModifications + "\nIsSvnItem = " +
                revObject4.IsSvnItem + "\nNeedsLocking = " +
                revObject4.NeedsLocking + "\nIsLocked = " +
                revObject4.IsLocked + "\nLockCreationDate = " +
                revObject4.LockCreationDate + "\nLockOwner = " +
                revObject4.LockOwner + "\nLockComment = " +
                revObject4.LockComment;

WScript.Echo(wcInfoString1);
WScript.Echo(wcInfoString2);
WScript.Echo(wcInfoString3);
WScript.Echo(wcInfoString4);
```

The following listing is an example on how to use the SubWCRev COM object from C#:

```
using LibSubWCRev;
SubWCRev sub = new SubWCRev();
sub.GetWCInfo("C:\\PathToMyFile\\MyFile.cc", true, true);
if (sub.IsSvnItem == true)
{
    MessageBox.Show("versioned");
}
else
{
    MessageBox.Show("not versioned");
}
```

第#6#章#IBugtraqProvider 接口

怎样通过问题跟踪器而不是简单的利用bugtraq:属性去获得更紧密的集成呢？TortoiseSVN可以利用COM插件来实现这个目的。这类插件的使用令直接从问题追踪者那里获取信息成为可能，还可以和用户互动，并把有关开放问题的信息返回给TortoiseSVN，也可以验校用户输入的日志信息并且在成功提交后运行某些动作，比如关闭一个问题。

We can't provide information and tutorials on how you have to implement a COM object in your preferred programming language, but we have example plugins in C++/ATL and C# in our repository in the contrib/issue-tracker-plugins folder. In that folder you can also find the required include files you need to build your plugin. (第 3 节 “许可协议”explains how to access the repository.)

您需要同时提供32位和64位的插件。因为x64版本的TortoiseSVN不能使用32位的插件，反之亦然。

6.1.#命名规范

If you release an issue tracker plugin for TortoiseSVN, please do not name it Tortoise<Something>. We'd like to reserve the Tortoise prefix for a version control client integrated into the windows shell. For example: TortoiseCVS, TortoiseSVN, TortoiseHg, TortoiseGit and TortoiseBzr are all version control clients.

Please name your plugin for a Tortoise client Turtle<Something>, where <Something> refers to the issue tracker that you are connecting to. Alternatively choose a name that sounds like Turtle but has a different first letter. Nice examples are:

- Gurtle - 用于Google code的问题跟踪插件
- TurtleMine - 用于Redmine的问题跟踪插件
- VurtleOne - 用于VersionOne的问题跟踪插件

6.2.#IBugtraqProvider 接口

TortoiseSVN 1.5及以后的版本可以使用实现了IBugtraqProvider接口的插件。利用这些接口提供的方法，插件可以和问题追踪器互动。

```
HRESULT ValidateParameters (
  // UI 的父窗体
  // 该UI需要在验证的过程中一直显示。
  [in] HWND hParentWnd,

  // 需要被验证的参数字符串。
  [in] BSTR parameters,

  // 字符串有效吗？
  [out, retval] VARIANT_BOOL *valid
);
```

这个方法是被一个设置对话框调用的，在这个对话框中，用户可以添加和设置插件。parameters字符串可以被某个插件使用，用以获得附加需求的信息，例如，问题追踪器的URL、登录信息等等。该插件需要验证parameters字符串并且当验证不通过时显示错误对话框。hParentWnd参数用来指定插件显示的对话框属于哪个父窗体。当parameters字符串验证通过时，该插件必须返回一个“TRUE”。如果返回的是“FALSE”，则设置对话框将不会允许用户添加插件到工作副本目录中。

```
HRESULT GetLinkText (
  // 任何需要被显示的（错误）UI的父窗体。
  [in] HWND hParentWnd,

  // 参数字符串， 仅仅当需要和web服务器会话时使用。
  // 例如，去查找正确的文本是什么。
  [in] BSTR parameters,

  // 你想显示那些字符串？
  // 使用当前线程区域。
  [out, retval] BSTR *linkText
);
```

该插件可以提供一个可以在TortoiseSVN提交对话框的按钮（这个按钮需要调用该插件）上显示的字符串，例如，"Choose issue" 或者 "Select ticket"。请确定这个字符串不会超长，否则会在按钮中显示不下。如果这个方法返回一个错误（例如。 E_NOTIMPL），按钮上会显示一个默认的字符串。

```
HRESULT GetCommitMessage (
  // 您提供者的UI的父窗体。
  [in] HWND hParentWnd,

  // 为您的提供者准备的参数。
  [in] BSTR parameters,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,

  // 在提交信息中已经显示过的文字。
  // 您的提供者应该在适当的地方包含这些文字。
  [in] BSTR originalMessage,

  // 新的提交信息文本。
  // 它将替换原始的提交信息。
  [out, retval] BSTR *newMessage
);
```

这是这个插件的主要方法，这个方法将在用户点击插件按钮的时候，被TortoiseSVN提交对话框调用。

parameters是一个字符串，用户在设置插件时必须将这个字符串填入设置对话框。通常，插件将通过这个字符串去寻找问题追踪器、登录信息等等。

commonRoot字符串包含所有被选择的项目（在这些项目上打开的提交对话框）的父路径。请注意： 不是 所有已被用户在提交对话框中选中的项目的根路径。对分支/表情对话框来说，这个字符串就代表将被复制的路径。

pathList参数包含一个数组（字符串类型的），数组里存放用户选中的将要提交的项目的路径。

originalMessage参数包含在提交对话框的日志信息框中登入的文本信息。如果用户还没有登入任何信息，这个字符串将为空。

newMessage返回的字符串被复制到提交对话框的日志信息编辑框中，之前显示在编辑框中的所有信息都将被覆盖。如果插件没有修改originalMessage字符串，在这里，它必须再次返回相同的字符串，否则用户填写的所有文本信息将会丢失。

6.3.#IBugtraqProvider2 接口

在 TortoiseSVN 1.6 中增加了一个可以为插件提供更多功能的接口。这个接口就是IBugtraqProvider2，它继承自 IBugtraqProvider。

```
HRESULT GetCommitMessage2 (
  // 您提供者的UI的父窗体。
  [in] HWND hParentWnd,

  // 为您的提供者准备的参数。
  [in] BSTR parameters,
  // 提交的通用URL
  [in] BSTR commonURL,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,

  // 在提交信息中已经显示过的文字。
  // 您的提供者应该在适当的地方包含这些文字。
  [in] BSTR originalMessage,

  // 通过下面两个参数，
  // 你可以给提交指定版本属性。
  // 注意：所有的 SAFEARRAY 必须是同一长度。
  //       对于每一个属性名，必须有属性值！

  // bugID 字段的内容（如果显示）
  [in] BSTR bugID,

  // bugID 字段中被修改的内容。
  [out] BSTR * bugIDOut,

  // 版本属性名的列表。
  [out] SAFEARRAY(BSTR) * revPropNames,

  // 版本属性值的列表。
  [out] SAFEARRAY(BSTR) * revPropValues,

  // 新的提交信息文本。
  // 它将替换原始的提交信息。
  [out, retval] BSTR * newMessage
);
```

当用户点击插件按钮时，这个方法被 TortoiseSVN 提交对话框调用。这个方法代替了 GetCommitMessage() 被调用。有些属性在 GetCommitMessage 中使用，具体请参考 GetCommitMessage 文档。

commonURL 属性是所有被选中的项目的父URL（在这些项目上打开的提交对话框）。从根本上说，这个URL就是 commonRoot 路径的URL。

The parameter bugID contains the content of the bug-ID field (if it is shown, configured with the property bugtraq:message).

The return parameter bugIDOut is used to fill the bug-ID field when the method returns.

The revPropNames and revPropValues return parameters can contain name/value pairs for revision properties that the commit should set. A plugin must make sure that both arrays have the same size on return! Each property name in revPropNames must also have a corresponding value in revPropValues. If no revision properties are to be set, the plugin must return empty arrays.

```
HRESULT CheckCommit (
  [in] HWND hParentWnd,
  [in] BSTR parameters,
```

```
  [in] BSTR commonURL,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,
  [in] BSTR commitMessage,
  [out, retval] BSTR * errorMessage
);
```

This method is called right before the commit dialog is closed and the commit begins. A plugin can use this method to validate the selected files/folders for the commit and/or the commit message entered by the user. The parameters are the same as for GetCommitMessage2(), with the difference that commonURL is now the common URL of all checked items, and commonRoot the root path of all checked items.

For the branch/tag dialog, the commonURL is the source URL of the copy, and commonRoot is set to the target URL of the copy.

The return parameter errorMessage must either contain an error message which TortoiseSVN shows to the user or be empty for the commit to start. If an error message is returned, TortoiseSVN shows the error string in a dialog and keeps the commit dialog open so the user can correct whatever is wrong. A plugin should therefore return an error string which informs the user what is wrong and how to correct it.

```
HRESULT  OnCommitFinished (
  // Parent window for any (error) UI that needs to be displayed.
  [in] HWND hParentWnd,

  // The common root of all paths that got committed.
  [in] BSTR commonRoot,

  // All the paths that got committed.
  [in] SAFEARRAY(BSTR) pathList,


  // The text already present in the commit message.
  [in] BSTR logMessage,

  // The revision of the commit.
  [in] ULONG revision,


  // An error to show to the user if this function
  // returns something else than S_OK
  [out, retval] BSTR * error
);
```

This method is called after a successful commit. A plugin can use this method to e.g., close the selected issue or add information about the commit to the issue. The parameters are the same as for GetCommitMessage2.

```
HRESULT HasOptions(
  // Whether the provider provides options
  [out, retval] VARIANT_BOOL *ret
);
```

This method is called from the settings dialog where the user can configure the plugins. If a plugin provides its own configuration dialog with ShowOptionsDialog, it must return TRUE here, otherwise it must return FALSE.

```
HRESULT ShowOptionsDialog(
  // Parent window for the options dialog
  [in] HWND hParentWnd,

  // Parameters for your provider.
  [in] BSTR parameters,

  // The parameters string
  [out, retval] BSTR * newparameters
);
```

This method is called from the settings dialog when the user clicks on the "Options" button that is shown if HasOptions returns TRUE. A plugin can show an options dialog to make it easier for the user to configure the plugin.

The parameters string contains the plugin parameters string that is already set/entered.

The newparameters return parameter must contain the parameters string which the plugin constructed from the info it gathered in its options dialog. That paramameters string is passed to all other IBugtraqProvider and IBugtraqProvider2 methods.

附录#A. #常见问题(FAQ)

因为TortoiseSVN一直在开发过程中，有的时候真的很难控制文档的版本完全是最新的。我们维护一个在线常见问题集(FAQ) [http://tortoisesvn.net/faq.html] 这里包括一些在TortoiseSVN邮件列表中提问的常见问题集合<dev@tortoisesvn.tigris.org> and <users@tortoisesvn.tigris.org>.

We also maintain a project Issue Tracker [http://code.google.com/p/tortoisesvn/wiki/IssueTracker?tm=3] which tells you about some of the things we have on our To-Do list, and bugs which have already been fixed. If you think you have found a bug, or want to request a new feature, check here first to see if someone else got there before you.

If you have a question which is not answered anywhere else, the best place to ask it is on one of the mailing lists:

- <users@tortoisesvn.tigris.org> is the one to use if you have questions about using TortoiseSVN.
- If you want to help out with the development of TortoiseSVN, then you should take part in discussions on <dev@tortoisesvn.tigris.org>.
- If you want to help with the translation of the TortoiseSVN user interface or the documentation, send an e-mail to <translators@tortoisesvn.tigris.org>.

附录#B. #如何实现 …

这个附录包括了 TortoiseSVN 使用中可能碰到的一些困难或疑问的解决方法。

B. 1. #一次移动或复制多个文件

移动或复制单个文件可以通过 TortoiseSVN → 重命名…实现。但是如果想移动或复制很多文件，这种方法就显得太慢而且工作量太大了。

推荐的方法是：用鼠标右键拖拽这些文件到新的位置。就是用鼠标右键选中想要移动或复制的文件，不要释放鼠标右键。把选中的文件拖拽到新的位置后释放鼠标右键。在出现的右键菜单里可以选择右键菜单 → SVN 复制版本控制文件到当前位置或右键菜单 → SVN 移动版本控制文件到当前位置。

B. 2. #强制用户写日志

有两种方法可以防止用户在不写日志的情况下进行提交操作。一种方式只对TortoiseSVN有效，另外一种方法对任何Subversion的客户端都有效，但是需要直接访问服务器。

B. 2. 1. #服务器端的钩子脚本(Hook-script)

如果能够直接访问服务器，可以安装一个pre-commit钩子脚本，通过这个脚本可以阻止所有空白日志或者日志太简短的提交操作。

In the repository folder on the server, there's a sub-folder hooks which contains some example hook scripts you can use. The file pre-commit.tmpl contains a sample script which will reject commits if no log message is supplied, or the message is too short. The file also contains comments on how to install/use this script. Just follow the instructions in that file.

除了TortoiseSVN，如果还要同时使用其他的Subversion客户端，推荐使用这种方法。缺点是提交是被服务器端拒绝的，因此用户会看到一个错误消息。客户端无法在提交之前就知道会被拒绝。如果希望在日志的内容达到足够长之前，TortoiseSVN 的 OK 按钮处于无效的状态，请使用下面的方法。

B. 2. 2. #工程(Project)属性

TortoiseSVN 使用属性来控制它的一些特性。这其中有一个 tsvn:logminsize 属性。

如果给一个文件夹设置了这个属性，在提交对话框里的日志信息达到属性里定义的长度之前，提交对话框的 OK 按钮会处于无效状态。

For detailed information on those project properties, please refer to 第 4.17 节 “项目设置”

B. 3. #从版本库里更新选定的文件到本地

通常，我们可以使用TortoiseSVN → 更新把工作副本更新到最新版。但是，如果只想更新某位同事添加的文件，而保留工作副本里其他文件的状态，就必须使用其它的方法。

选择 TortoiseSVN → 查看更新，然后点击查看版本库按钮，就能够看到上次更新以后版本库里发生了哪些变化。选中想更新到本地的文件，然后用右键菜单更新这些文件。

B. 4. #Roll back (Undo) revisions in the repository

B. 4. 1. #使用版本日志对话框

By far the easiest way to revert the changes from one or more revisions, is to use the revision log dialog.

1. 选中想要恢复变更的文件或者文件夹。如果想要恢复所有的变更，需要选中最顶层的文件夹。

2. 选择TortoiseSVN → 显示日志，显示出版本列表。有可能需要使用全部显示或者下100(条) 按钮，把想要恢复的版本显示出来。

3. Select the revision you wish to revert. If you want to undo a range of revisions, select the first one and hold the Shift key while selecting the last one. If you want to pick out individual revisions and ranges, use the Ctrl key while selecting revisions. Right click on the selected revision(s), then select Context Menu → Revert changes from this revision.

4. 如果想要把以前的某个版本变成最新版本，右键点击选中的版本，然后选择右键菜单 → 恢复到此版本。就能够撤销被选中版本后面所有的变更。

工作副本已经恢复到了变更以前的状态。检查恢复后的结果，然后提交变更。

B.4.2.#使用合并对话框

If you want to enter revision numbers as a list, you can use the Merge dialog. The previous method uses merging behind the scenes; this method uses it explicitly.

1. 在工作副本上选择 TortoiseSVN → 合并。

2. In the Merge Type dialog select Merge a range of revisions.

3. In the From: field enter the full repository URL of your working copy folder. This should come up as the default URL.

4. In the Revision range to merge field enter the list of revisions to roll back (or use the log dialog to select them as described above).

5. Make sure the Reverse merge checkbox is checked.

6. In the Merge options dialog accept the defaults.

7. Click Merge to complete the merge.

You have reverted the changes within your working copy. Check that the results are as expected, then commit the changes.

B.4.3.#使用 svndumpfilter

因为TortoiseSVN绝不会丢弃数据，所以那些被回滚的版本仍然以中间版本的形式被保留在版本库里。只是最新版本已经回到了以前的状态。如果想让版本库里的某些版本彻底消失，擦去这些版本曾经存在过的所有痕迹，就必须采取更极端的手段。不推荐使用这种方法，除非有很好的理由。比如某人向一个公开的版本库里提交了一份机密文件。

The only way to remove data from the repository is to use the Subversion command line tool svnadmin. You can find a description of how this works in the Repository Maintenance [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.maint.html].

B.5.#Compare two revisions of a file or folder

如果希望比较某个文件的两个历史版本，比如同一个文件修订版本100和200，可以用TortoiseSVN → 显示日志列出这个文件的历史版本纪录，选择希望比较的两个版本，然后使用右键菜单 → 比较版本差异。

如果希望比较两个不同目录树下的同一个文件，比如主干和分支，可以使用版本库浏览器打开两个目录树，在两个目录树下选择同一个文件，然后使用 右键菜单 → 比较版本差异。

如果希望比较两个目录树下的所有变化，比如主干和某个发布标签，可以使用TortoiseSVN → 版本分支图。选择两个想要比较的节点，然后使用右键菜单 → 比较最新版本，就会列出一个变更文件列表。在列表上选择单个文件就能够浏览该文件的具体变更内容。另外一个方法是使用右键菜单 → 最新版本的标准差异(Unified diff)显示所有变更的汇总和最少限度的上下文。

B.6.#包含一个普通的子项目

Sometimes you will want to include another project within your working copy, perhaps some library code. There are at least 4 ways of dealing with this.

B.6.1.#使用 svn:externals

Set the svn:externals property for a folder in your project. This property consists of one or more lines; each line has the name of a sub-folder which you want to use as the checkout folder for common code, and the repository URL that you want to be checked out there. For full details refer to 第 4.18 节 “外部条目”

Commit the new folder. Now when you update, Subversion will pull a copy of that project from its repository into your working copy. The sub-folders will be created automatically if required. Each time you update your main working copy, you will also receive the latest version of all external projects.

如果外部项目位于同一版本库，当你提交你的主要项目时，你在这儿所做的任何更改都将包含在提交列表中。

如果外部工程位于不同的版本库，当你向主项目提交你的修改时，你对外部工程做的修改会被通报，但是你必须单独的提交这些外部工程的修改。

在已述的三种方法中，这是唯一不需要在客户端设置的方法。一旦在目录属性中指定了外部资源，所有客户端在更新时都会创建相应的目录。

B.6.2.#使用嵌套工作副本

Create a new folder within your project to contain the common code, but do not add it to Subversion.

在新目录下选择 TortoiseSVN → 检出，在其中检出普通代码的副本，现在你在主要的工作副本有了一个独立的嵌套的工作副本。

两个工作副本是独立的，当你在父目录提交修改，嵌套的工作副本会被忽略，同样当你更新你的父目录，嵌套的工作副本不会被更新。

B.6.3.#使用相对位置

If you use the same common core code in several projects, and you do not want to keep multiple working copies of it for every project that uses it, you can just check it out to a separate location which is related to all the other projects which use it. For example:

```
C:\Projects\Proj1
C:\Projects\Proj2
C:\Projects\Proj3
C:\Projects\Common
```

and refer to the common code using a relative path, e.g. ..\..\Common\DSPcore.

If your projects are scattered in unrelated locations you can use a variant of this, which is to put the common code in one location and use drive letter substitution to map that

location to something you can hard code in your projects, e.g. Checkout the common code to D:\Documents\Framework or C:\Documents and Settings\{login}\My Documents\framework then use

```
SUBST X: "D:\Documents\framework"
```

to create the drive mapping used in your source code. Your code can then use absolute locations.

```
#include "X:\superio\superio.h"
```

这个方法职能工作在完全PC的环境，你所做的就是记录必须的磁盘映射，所以你的团队知道这些神秘文件的位置，这个方法只能用于紧密地开发环境，在普通的使用中并不推荐。

B.6.4. #增加此项目到版本库

The maybe easiest way is to simply add the project in a subfolder to your own project working copy. However this has the disadvantage that you have to update and upgrade this external project manually.

To help with the upgrade, TortoiseSVN provides a command in the explorer right-drag context menu. Simply right-drag the folder where you unzipped the new version of the external library to the folder in your working copy, and then select Context Menu → SVN Vendorbranch here. This will then copy the new files over to the target folder while automatically adding new files and removing files that aren't in the new version anymore.

B.7. #创建到版本库的快捷方式

如果你经常需要从某个位置打开版本库浏览器，你可以使用 TortoiseProc 自动化接口创建一个快捷方式。只需要创建一个新的快捷方式，将目标设置为:

```
TortoiseProc.exe /command:repobrowser /path:"url/to/repository"
```

当然你需要包含真实的版本库 URL。

B.8. #忽略已经版本控制的文件

如果你不小心添加了一些应该被忽略的文件，你如何将它们从版本控制中去除而不会丢失它们？或许你有自己的IDE配置文件，不是项目的一部分，但将会花费很多时间使之按照自己的方式工作。

If you have not yet committed the add, then all you have to do is use TortoiseSVN → Undo Add... to undo the add. You should then add the file(s) to the ignore list so they don't get added again later by mistake.

If the files are already in the repository, they have to be deleted from the repository and added to the ignore list. Fortunately TortoiseSVN has a convenient shortcut for doing this. TortoiseSVN → Unversion and add to ignore list will first mark the file/folder for deletion from the repository, keeping the local copy. It also adds this item to the ignore list so that it will not be added back into Subversion again by mistake. Once this is done you just need to commit the parent folder.

B.9. #从工作副本删除版本信息

If you have a working copy which you want to convert back to a plain folder tree without the .svn directories, you can simply export it to itself. Read 第 4.26.1 节 “从版本控制里移除删除工作副本” to find out how.

B.10. #删除工作副本

If you have a working copy which you no longer need, how do you get rid of it cleanly? Easy - just delete it in Windows Explorer! Working copies are private local entities, and they are self-contained. Deleting a working copy in Windows Explorer does not affect the data in the repository at all.

附录#C.#Useful Tips For Administrators

附录包含了将TortoiseSVN部署到多个客户端电脑时可能发生问题的解决方案。

C.1.#通过组策略部署 TortoiseSVN

The TortoiseSVN installer comes as an MSI file, which means you should have no problems adding that MSI file to the group policies of your domain controller.

A good walk-through on how to do that can be found in the knowledge base article 314934 from Microsoft: http://support.microsoft.com/?kbid=314934.

TortoiseSVN must be installed under Computer Configuration and not under User Configuration. This is because TortoiseSVN needs the CRT and MFC DLLs, which can only be deployed per computer and not per user. If you really must install TortoiseSVN on a per user basis, then you must first install the MFC and CRT package version 11 from Microsoft on each computer you want to install TortoiseSVN as per user.

You can customize the MSI file if you wish so that all your users end up with the same settings. TSVN settings are stored in the registry under HKEY_CURRENT_USER\Software\TortoiseSVN and general Subversion settings (which affect all Subversion clients) are stored in config files under %APPDATA%\Subversion. If you need help with MSI customization, try one of the MSI transform forums or search the web for "MSI transform".

C.2.#重定向升级检查

TortoiseSVN checks if there's a new version available every few days. If there is a newer version available, a notification is shown in the commit dialog.

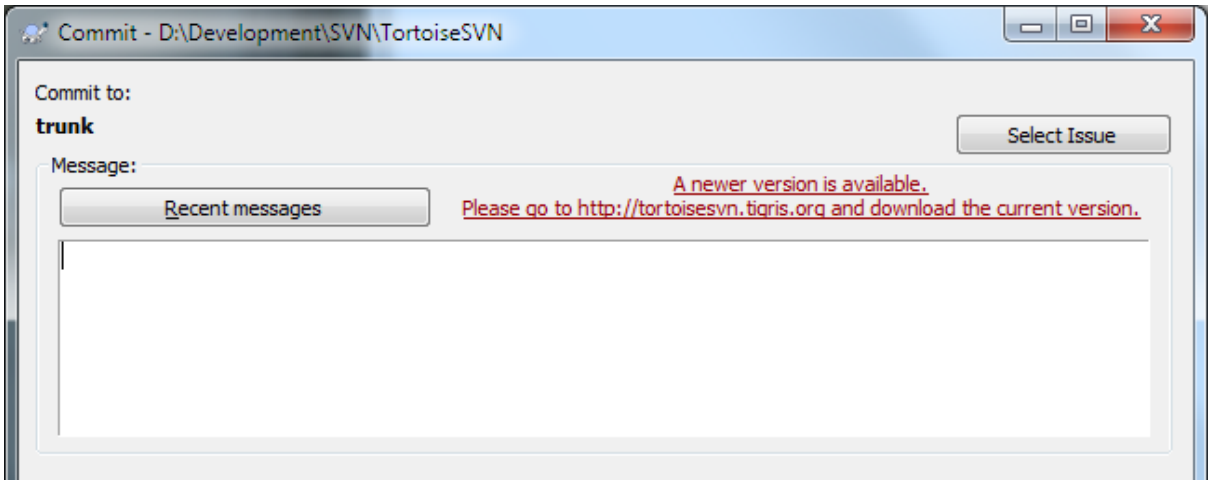


图 C.1. The commit dialog, showing the upgrade notification

If you're responsible for a lot of users in your domain, you might want your users to use only versions you have approved and not have them install always the latest version. You probably don't want that upgrade notification to show up so your users don't go and upgrade immediately.

Versions 1.4.0 and later of TortoiseSVN allow you to redirect that upgrade check to your intranet server. You can set the registry key HKCU\Software\TortoiseSVN\UpdateCheckURL (string value) to an URL pointing to a text file in your intranet. That text file must have the following format:

```
1.4.1.6000
A new version of TortoiseSVN is available for you to download!
http://192.168.2.1/downloads/TortoiseSVN-1.4.1.6000-svn-1.4.0.msi
```

The first line in that file is the version string. You must make sure that it matches the exact version string of the TortoiseSVN installation package. The second line is a custom text, shown in the commit dialog. You can write there whatever you want. Just note that the space in the commit dialog is limited. Too long messages will get truncated! The third line is the URL to the new installation package. This URL is opened when the user clicks on the custom message label in the commit dialog. You can also just point the user to a web page instead of the MSI file directly. The URL is opened with the default web browser, so if you specify a web page, that page is opened and shown to the user. If you specify the MSI package, the browser will ask the user to save the MSI file locally.

C.3.#设置 SVN_ASP_DOT_NET_HACK 环境变量

As of version 1.4.0 and later, the TortoiseSVN installer doesn't provide the user with the option to set the SVN_ASP_DOT_NET_HACK environment variable anymore, since that caused many problems and confusion for users who always install everything no matter whether they know what it is for.

But that option is only hidden for the user. You still can force the TortoiseSVN installer to set that environment variable by setting the ASPDOTNETHACK property to TRUE. For example, you can start the installer like this:

```
msiexec /i TortoiseSVN-1.4.0.msi ASPDOTNETHACK=TRUE
```

Please note that this hack is only necessary if you're still using VS.NET2002. All later versions of Visual Studio do not require this hack to be activated! So unless you're using that ancient tool, DO NOT USE THIS!

C.4.#禁用上下文菜单

As of version 1.5.0 and later, TortoiseSVN allows you to disable (actually, hide) context menu entries. Since this is a feature which should not be used lightly but only if there is a compelling reason, there is no GUI for this and it has to be done directly in the registry. This can be used to disable certain commands for users who should not use them. But please note that only the context menu entries in the explorer are hidden, and the commands are still available through other means, e.g. the command line or even other dialogs in TortoiseSVN itself!

控制上下文菜单显示的注册表键是HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskLow 和 HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskHigh。

每个注册表条目都是 DWORD 类型，每位对应一个指定的菜单项。置位意味着对应的菜单项禁用。

取值	菜单入口
0x0000000000000001	检出
0x0000000000000002	更新
0x0000000000000004	提交
0x0000000000000008	添加
0x0000000000000010	恢复
0x0000000000000020	清理

取值	菜单入口
0x0000000000000040	解决
0x0000000000000080	切换
0x0000000000000100	导入
0x0000000000000200	输出
0x0000000000000400	在当前位置创建版本库
0x0000000000000800	分支/标记
0x0000000000001000	合并
0x0000000000002000	删除
0x0000000000004000	改名
0x0000000000008000	更新到版本
0x0000000000010000	差异
0x0000000000020000	显示日志
0x0000000000040000	编辑冲突
0x0000000000080000	重新定位
0x0000000000100000	检查修改
0x0000000000200000	忽略
0x0000000000400000	版本库浏览器
0x0000000000800000	追溯
0x0000000001000000	创建补丁
0x0000000002000000	应用补丁(Apply Patch)
0x0000000004000000	版本图
0x0000000008000000	锁
0x0000000010000000	删除锁
0x0000000020000000	属性
0x0000000040000000	与 URL 比较
0x0000000080000000	删除未版本控制的项目
0x0000000100000000	合并所有
0x0000000200000000	与前一版本比较差异
0x0000000400000000	粘贴
0x0000000800000000	升级工作副本
0x2000000000000000	设置
0x4000000000000000	帮助
0x8000000000000000	关于

表 C.1. 菜单入口和取值

Example: to disable the “Relocate” the “Delete unversioned items” and the “Settings” menu entries, add the values assigned to the entries like this:

```
  0x0000000000080000
+ 0x0000000080000000
+ 0x2000000000000000
```

= 0x2000000080080000

The lower DWORD value (0x80080000) must then be stored in HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskLow, the higher DWORD value (0x20000000) in HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskHigh.

简单删除这两个注册键，即可重新激活菜单项。

附录#D.#TortoiseSVN 操作

因为所有的命令使用命令行参数控制，你可以使用特定的批处理脚本或从其它程序(例如你喜欢的文本编辑器)启动。

请记住TortoiseSVN是一个GUI客户端，这个自动化指导为你展示了让TortoiseSVN对话框显示并收集客户输入，如果你希望编写不需要输入的脚本，你应该使用官方的Subversion命令行客户端。

D.1.#TortoiseSVN 命令

TortoiseSVN的GUI程序叫做TortoiseProc.exe。所有的命令通过参数/command:abcd指定，其中abcd是必须的命令名。大多数此类命令至少需要一个路径参数，使用/path:"some\path"指定。在下面的命令表格中，命令引用的是/command:abcd参数，余下的代表了/path:"some\path"参数。

因为一些命令需要一个目标路径的列表(例如提交一些特定的文件)，/path参数可以接收多个路径，使用*分割。

You can also specify a file which contains a list of paths, separated by newlines. The file must be in UTF-16 format, without a BOM [http://en.wikipedia.org/wiki/Byte-order_mark]. If you pass such a file, use /pathfile instead of /path. To have TortoiseProc delete that file after the command is finished, you can pass the parameter /deletepathfile.

The progress dialog which is used for commits, updates and many more commands usually stays open after the command has finished until the user presses the OK button. This can be changed by checking the corresponding option in the settings dialog. But using that setting will close the progress dialog, no matter if you start the command from your batch file or from the TortoiseSVN context menu.

To specify a different location of the configuration file, use the parameter /configdir:"path\to\config\directory". This will override the default path, including any registry setting.

To close the progress dialog at the end of a command automatically without using the permanent setting you can pass the /closeonend parameter.

- /closeonend:0 不自动关闭对话框
- /closeonend:1 如果没发生错误则自动关闭对话框
- /closeonend:2 如果没发生错误和冲突则自动关闭对话框
- /closeonend:3如果没有错误、冲突和合并，会自动关闭

To close the progress dialog for local operations if there were no errors or conflicts, pass the /closeforlocal parameter.

下面的列表列出了所有可以使用TortoiseProc.exe访问的命令，就像上面的描述，必须使用/command:abcd的形式，在列表中，因为节省空间的关系省略了/command的前缀。

命令	描述
:about	显示关于对话框。如果没有给命令也会显示。
:log	Opens the log dialog. The /path specifies the file or folder for which the log should be shown. Additional options can be set: /startrev:xxx, /endrev:xxx, /strict enables the 'stop-on-copy' checkbox, /merge enables the 'include merged revisions' checkbox, /findstring:"filterstring" fills in the filter text, /findtext forces the filter to use text, not regex, or /findregex forces the filter to use regex, not simple text search, and /findtype:X with X being

命令	描述
	a number between 0 and 511. The numbers are the sum of the following options: • /findtype:0 filter by everything • /findtype:1 filter by messages • /findtype:2 filter by path • /findtype:4 按作者筛选 • /findtype:8 filter by revisions • /findtype:16 not used • /findtype:32 filter by bug ID • /findtype:64 not used • /findtype:128 filter by date • /findtype:256 filter by date range If /outfile:path\to\file is specified, the selected revisions are written to that file when the log dialog is closed. The revisions are written in the same format as is used to specify revisions in the merge dialog.
:checkout	Opens the checkout dialog. The /path specifies the target directory and the /url specifies the URL to checkout from. If you specify the key /blockpathadjustments, the automatic checkout path adjustments are blocked. The /revision:XXX specifies the revision to check out.
:import	Opens the import dialog. The /path specifies the directory with the data to import. You can also specify the /logmsg switch to pass a predefined log message to the import dialog. Or, if you don't want to pass the log message on the command line, use /logmsgfile:path, where path points to a file containing the log message.
:update	Updates the working copy in /path to HEAD. If the option /rev is given then a dialog is shown to ask the user to which revision the update should go. To avoid the dialog specify a revision number /rev:1234. Other options are /nonrecursive, /ignoreexternals and /ignoreexternals. The /stickydepth indicates that the specified depth should be sticky, creating a sparse checkout. The /skipprechecks can be set to skip all checks that are done before an update. If this is specified, then the 显示日志 button is disabled, and the context menu to show diffs is also disabled after the update.
:commit	打开提交对话框，/path 指定了目标路径或需要提交的文件列表，你也可以使用参数 /logmsg 给提交窗口传递预定义的日志信息，或者你不希望将日志传递给命令行，你也可以使用 /logmsgfile:path，path 指向了保存日志信息的文件。为了预先填入bug的ID(如果你设置了集成bug追踪属性)，你可以使用/bugid:"the bug id here"完成这个任务。
:add	将/path的文件添加到版本控制 。
:revert	恢复工作副本的本地修改，/path说明恢复哪些条目。
:cleanup	Cleans up interrupted or aborted operations and unlocks the working copy in /path. Use /noui to prevent the result dialog from popping

命令	描述
	up (either telling about the cleanup being finished or showing an error message). /noprogressui also disables the progress dialog. /nodlg disables showing the cleanup dialog where the user can choose what exactly should be done in the cleanup. The available actions can be specified with the options /cleanup for status cleanup, /revert, /delunversioned, /delignored, /refreshshell and /externals.
:resolve	将/path指定文件的冲突标示为解决，如果给定/noquestion，解决不会向用户确认操作。
:repocreate	在/path创建一个版本库。
:switch	Opens the switch dialog. The /path specifies the target directory and /url the URL to switch to.
:export	Exports the working copy in /path to another directory. If the /path points to an unversioned directory, a dialog will ask for an URL to export to the directory in /path. If you specify the key /blockpathadjustments, the automatic export path adjustments are blocked.
:dropexport	Exports the working copy in /path to the directory specified in /droptarget. This exporting does not use the export dialog but executes directly. The option /overwrite specifies that existing files are overwritten without user confirmation, and the option /autorename specifies that if files already exist, the exported files get automatically renamed to avoid overwriting them. The option /extended can specify either localchanges to only export files that got changed locally, or unversioned to also export all unversioned items as well.
:dropvendor	Copies the folder in /path recursively to the directory specified in /droptarget. New files are added automatically, and missing files get removed in the target working copy, basically ensuring that source and destination are exactly the same.
:merge	Opens the merge dialog. The /path specifies the target directory. For merging a revision range, the following options are available: /fromurl:URL, /revrange:string. For merging two repository trees, the following options are available: /fromurl:URL, /tourl:URL, /fromrev:xxx and /torev:xxx.
:mergeall	Opens the merge all dialog. The /path specifies the target directory.
:copy	Brings up the branch/tag dialog. The /path is the working copy to branch/tag from. And the /url is the target URL. To already check the option Switch working copy to new branch/tag you can pass the /switchaftercopy switch. You can also specify the /logmsg switch to pass a predefined log message to the branch/tag dialog. Or, if you don't want to pass the log message on the command line, use /logmsgfile:path, where path points to a file containing the log message.
:settings	打开设置对话框。
:remove	从版本控制里移除/path中的文件。
:rename	重命名/path的文件，会在对话框中询问新文件，为了防止一个步骤中询问相似文件，传递/noquestion。
:diff	Starts the external diff program specified in the TortoiseSVN settings. The /path specifies the first file. If the option /path2 is set, then the diff program is started with those two files. If /path2 is omitted, then the diff is done between the file in /

命令	描述
	path and its BASE. To explicitly set the revision numbers use /startrev:xxx and /endrev:xxx, and for the optional peg revision use /pegrevision:xxx. If /blame is set and /path2 is not set, then the diff is done by first blaming the files with the given revisions. The parameter /line:xxx specifies the line to jump to when the diff is shown.
:showcompare	Depending on the URLs and revisions to compare, this either shows a unified diff (if the option unified is set), a dialog with a list of files that have changed or if the URLs point to files starts the diff viewer for those two files. The options url1, url2, revision1 and revision2 must be specified. The options pegrevision, ignoreancestry, blame and unified are optional.
:conflicteditor	Starts the conflict editor specified in the TortoiseSVN settings with the correct files for the conflicted file in /path.
:relocate	打开重定位对话框，/path指定了重定位的工作副本路径。
:help	打开帮助文件
:repostatus	打开检查修改对话框。在 /path 指定工作副本目录。如果已指定 /remote，对话框将在启动时立即联系版本库，如同用户点击了 检查版本库按钮。
:repobrowser	Starts the repository browser dialog, pointing to the URL of the working copy given in /path or /path points directly to an URL. An additional option /rev:xxx can be used to specify the revision which the repository browser should show. If the /rev:xxx is omitted, it defaults to HEAD. If /path points to an URL, the /projectpropertiespath:path/to/wc specifies the path from where to read and use the project properties. If /outfile:path\to\file is specified, the selected URL and revision are written to that file when the repository browser is closed. The first line in that text file contains the URL, the second line the revision in text format.
:ignore	将/path中的对象加入到忽略列表，也就是将这些文件添加到 svn:ignore 属性。
:blame	为 /path 选项指定的文件打开追溯对话框。 如果设置了 /startrev 和 /endrev 选项，不会显示询问追溯范围对话框，直接使用这些选项中的版本号。 如果设置了 /line:nnn 选项，TortoiseBlame 会显示指定行数。 也支持 /ignoreeol，/ignorespaces 和 /ignoreallspaces 选项。
:cat	将/path指定的工作副本或URL的文件保存到/savepath:path，修订版本号在/revision:xxx，这样可以得到特定修订版本的文件。
:createpatch	Creates a patch file for the path given in /path. To skip the file Save-As dialog you can pass /savepath:path to specify the path where to save the patch file to directly. To prevent the unified diff viewer from being started showing the patch file, pass /noview.
:revisiongraph	按照在/path中给出的路径显示版本分支图。

命令	描述
	To create an image file of the revision graph for a specific path, but without showing the graph window, pass /output:path with the path to the output file. The output file must have an extension that the revision graph can actually export to. These are: .svg, .wmf, .png, .jpg, .bmp and .gif. Since the revision graph has many options that affect how it is shown, you can also set the options to use when creating the output image file. Pass these options with /options:XXXX, where XXXX is a decimal value. The best way to find the required options is to start the revision graph the usual way, set all user-interface options and close the graph. Then the options you need to pass on the command line can be read from the registry HKCU\Software\TortoiseSVN\RevisionGraphOptions.
:lock	Locks a file or all files in a directory given in /path. The 'lock' dialog is shown so the user can enter a comment for the lock.
:unlock	Unlocks a file or all files in a directory given in /path.
:rebuildiconcache	Rebuilds the windows icon cache. Only use this in case the windows icons are corrupted. A side effect of this (which can't be avoided) is that the icons on the desktop get rearranged. To suppress the message box, pass /noquestion.
:properties	Shows the properties dialog for the path given in /path. For dealing with versioned properties this command requires a working copy. Revision properties can be viewed/changed if /path is an URL and /rev:XXX is specified. To open the properties dialog directly for a specific property, pass the property name as /property:name.

表 D.1. 有效命令及选项列表

Examples (which should be entered on one line):

```
TortoiseProc.exe /command:commit
                 /path:"c:\svn_wc\file1.txt*c:\svn_wc\file2.txt"
                 /logmsg:"test log message" /closeonend:0

TortoiseProc.exe /command:update /path:"c:\svn_wc\" /closeonend:0

TortoiseProc.exe /command:log /path:"c:\svn_wc\file1.txt"
                 /startrev:50 /endrev:60 /closeonend:0
```

D.2. #Tsvncmd URL handler

Using special URLs, it is also possible to call TortoiseProc from a web page.

TortoiseSVN registers a new protocol tsvncmd: which can be used to create hyperlinks that execute TortoiseSVN commands. The commands and parameters are the same as when automating TortoiseSVN from the command line.

The format of the tsvncmd: URL is like this:

tsvncmd:command:cmd?parameter:paramvalue?parameter:paramvalue

with cmd being one of the allowed commands, parameter being the name of a parameter like path or revision, and paramvalue being the value to use for that parameter. The list of parameters allowed depends on the command used.

下列命令可以使用tsvncmd: URLs:

- :update
- :commit
- :diff
- :repobrowser
- :checkout
- :export
- :blame
- :repostatus
- :revisiongraph
- :showcompare
- :log

A simple example URL might look like this:

```
<a href="tsvncmd:command:update?path:c:\svn_wc?rev:1234">Update</a>
```

or in a more complex case:

```
<a href="tsvncmd:command:showcompare?
url1:https://stexbar.googlecode.com/svn/trunk/StExBar/src/setup/Setup.wxs?
url2:https://stexbar.googlecode.com/svn/trunk/StExBar/src/setup/Setup.wxs?
revision1:188?revision2:189">compare</a>
```

D.3. #TortoiseIDiff 命令

图形比较工具有几个用于控制启动的命令行选项。程序名称是TortoiseIDiff.exe。

下表列出可以在命令行传递给图形比较工具的所有选项。

选项	描述
:left	显示在左边的文件路径。
:lefttitle	标题字符串。此字符串用于图形视图标题，替换图形文件的全路径名称。
:right	显示在右边的文件路径。
:righttitle	标题字符串。此字符串用于图形视图标题，替换图形文件的全路径名称。
:overlay	如果指定，图形比较工具切换到重载模式(alpha 混合)。
:fit	如果指定，图形比较工具最大化所有图形。
:showinfo	显示图片信息框。

表 D.2. 可用选项列表

样例(应该在同一行输入):

```
TortoiseIDiff.exe /left:"c:\images\img1.jpg" /lefttitle:"image 1"
                  /right:"c:\images\img2.jpg" /righttitle:"image 2"
                  /fit /overlay
```

附录#E. #命令行交叉索引

有时候，本手册会参考Subversion的文档，会以命令行方式(CLI)描述Subversion术语，为了理解TortoiseSVN后台的操作，我们编辑了一份列表，用来展示命令行命令和对应的TortoiseSVN的GUI操作的关系。

> 即使有命令行对应TortoiseSVN的操作，请记住TortoiseSVN没有调用命令行，而是直接使用了Subversion库。

If you think you have found a bug in TortoiseSVN, we may ask you to try to reproduce it using the CLI, so that we can distinguish TortoiseSVN issues from Subversion issues. This reference tells you which command to try.

E.1. #约定和基本规则

In the descriptions which follow, the URL for a repository location is shown simply as URL, and an example might be http://tortoisesvn.googlecode.com/svn/trunk/. The working copy path is shown simply as PATH, and an example might be C:\TortoiseSVN\trunk.

因为TortoiseSVN是一个Windows外壳扩展，它不能使用当前工作副本的概念，所有的工作副本必须使用绝对路径，而不是相对的。

特定项目是可选的，TortoiseSVN里这是通过多选项和单选项控制的，这些选项是在命令行定义的[方括号]里显示的。

E.2. #TortoiseSVN 命令

E.2.1. #检出

```
svn checkout [-depth ARG] [--ignore-externals] [-r rev] URL PATH
```

The depth combo box items relate to the -depth argument.

如果希望忽略外部被选中，使用--ignore-externals选型。

如果你正在检出特定的修订版本，在URL后使用-r指定。

E.2.2. #更新

```
svn info URL_of_WC
svn update [-r rev] PATH
```

更新多个项目在Subversion还不是原子操作，所以TortoiseSVN会首先找到版本库的HEAD修订版本，然后将所有项目更新到特定修订版本，防止出现混合修订版本的工作副本。

如果只有一个项目被选中更新，或选中的项目来自不同的版本库，TortoiseSVN只会更新到HEAD。

没有使用命令行选项，更新到修订版本也实现了更新命令，但提供了更多的选项。

E.2.3. #更新到版本

```
svn info URL_of_WC
svn update [-r rev] [-depth ARG] [--ignore-externals] PATH
```

The depth combo box items relate to the -depth argument.

如果希望忽略外部被选中，使用--ignore-externals选型。

E.2.4. #提交

在TortoiseSVN，提交对话框使用Subversion命令，第一部分是检查工作副本哪些文件可能被提交，然后你可以检查列表，比较与BASE的区别，选择你希望提交包含的项目。

```
svn status -v PATH
```

如果选择了显示未版本控制的文件，TortoiseSVN会遵循忽略规则显示工作目录中所有未版本控制的文件和文件夹。这个特性在Subversion中没有等价操作，因为svn status 命令不扫描未版本控制的文件夹。

如果你选择了未版本控制的文件和文件夹，这些项目都会先增加到你的工作副本。

```
svn add PATH...
```

当你点击确认，开始执行Subversion提交。如果你不修改所有的文件检查框，TortoiseSVN 会递归提交工作副本。如果你取消选择一些文件，那么就必须使用非递归提交 (-N) ，每个路径都必须在命令行上单独指定。

```
svn commit -m "LogMessage" [-depth ARG] [--no-unlock] PATH...
```

日志消息是日志编辑框的内容。它可以为空。

如果选择了保持锁，就使用--no-unlock开关。

E.2.5. #差异

```
svn diff PATH
```

If you use Diff from the main context menu, you are diffing a modified file against its BASE revision. The output from the CLI command above also does this and produces output in unified-diff format. However, this is not what TortoiseSVN is using. TortoiseSVN uses TortoiseMerge (or a diff program of your choosing) to display differences visually between full-text files, so there is no direct CLI equivalent.

你可以使用TortoiseSVN,比较任意两个文件的差异，不管他们是否受版本控制。TortoiseSVN只是把这两个文件传递给已经选择的比较差异程序，让它比较差异。

E.2.6. #显示日志

```
svn log -v -r 0:N --limit 100 [--stop-on-copy] PATH
  或者
svn log -v -r M:N [--stop-on-copy] PATH
```

默认情况下，TortoiseSVN尝试用--limit方法取得100个日志消息。如果设置了让它使用旧借口，那么就使用第二种个是获得100个日志消息。

如果选择了停止于复制/改名，就使用--stop-on-copy开关。

E.2.7. #检查修改

```
svn status -v PATH
  或者
svn status -u -v PATH
```

只在你的工作副本执行初始的状态检查。如果你点击检查版本库，那么也检查版本库，察看哪些文件会被更新操作修改，它需要-u开关。

如果选择了显示未版本控制的文件，TortoiseSVN会遵循忽略规则显示工作目录中所有未版本控制的文件和文件夹。这个特性在Subversion中没有等价操作，因为svn status 命令不扫描未版本控制的文件夹。

E.2.8. #版本图

版本图是TortoiseSVN特有的，命令行客户端没有等价实现。

TortoiseSVN执行了这些操作

```
svn info URL_of_WC
svn log -v URL
```

其中URL是版本库的 根，返回分析数据。

E.2.9. #版本库浏览器

```
svn info URL_of_WC
svn list [-r rev] -v URL
```

你可以使用svn info检查版本库的根，它在版本库浏览器的顶级显示。你不能浏览它的上级目录。同样，这个命令返回所有显示在版本库浏览器的锁信息。

给出URL和可选的版本号，svn list列出目录中的内容。

E.2.10. #编辑冲突

这个命令没有控制台等价实现。它调用TortoiseMerge或者外部三路差异/合并工具察看棘手的冲突，挑选出冲突行。

E.2.11. #已解决

```
svn resolved PATH
```

E.2.12. #改名

```
svn rename CURR_PATH NEW_PATH
```

E.2.13. #删除

```
svn delete PATH
```

E.2.14. #恢复

```
svn status -v PATH
```

首先开始状态检查，察看你的工作副本有哪些项目可以被撤销。你可以复审文件列表，检查这些文件的修改，然后选择你要撤销的项目。

当你点击确认时，开始Subversion撤销操作。如果你不修改所有的文件检查框，TortoiseSVN 会递归撤销 (-R)工作副本的修改。如果你取消选择一些文件，那么就必须使用非递归撤销 ，每个路径都必须在命令行上单独指定。"

```
svn revert [-R] PATH...
```

E.2.15.#清理

```
svn cleanup PATH
```

E.2.16.#获得锁

```
svn status -v PATH
```

首先开始状态检查，察看你的工作副本有哪些项目可以被加锁。你可以选择想加锁的项目。

```
svn lock -m "LockMessage" [--force] PATH...
```

加锁信息是加锁编辑框的内容。它可以为空。"

如果选择了强制锁定 ，就使用--force开关。

E.2.17.#释放锁

```
svn unlock PATH
```

E.2.18.#分支/标记

```
svn copy -m "LogMessage" URL URL
  或
svn copy -m "LogMessage" URL@rev URL@rev
  或
svn copy -m "LogMessage" PATH URL
```

分支/标签对话框在版本库执行复制。有三个单选按钮:

- 版本库中的最新版本
- 指定版本库中的版本
- 工作副本

对应上面的三个命令行参数。

日志消息是日志编辑框的内容。它可以为空。

E.2.19.#切换

```
svn info URL_of_WC
svn switch [-r rev] URL PATH
```

E.2.20.#合并

```
svn merge [--dry-run] --force From_URL@revN To_URL@revM PATH
```

测试合并与使用--dry-run选项的合并相同。

```
svn diff From_URL@revN To_URL@revM
```

Unified diff显示了用来合并的区别操作。

E.2.21. #输出

```
svn export [-r rev] [--ignore-externals] URL Export_PATH
```

这个形式是当从一个未版本控制目录访问，并且文件夹作为目标。

导出一个工作副本到一个目录没有使用Subversion的库，所以没有等同的命令行匹配。

TortoiseSVN做的只是将所有文件复制到一个新的位置，并且会显示操作的过程。未版本控制的文件/文件夹也可以被导出。

在两种情况下，如果Omit externals被选中，就相当于使用了--ignore-externals选项。

E.2.22. #重新定位

```
svn switch --relocate From_URL To_URL
```

E.2.23. #在当前位置创建版本库

```
svnadmin create --fs-type fsfs PATH
```

E.2.24. #添加

```
svn add PATH...
```

如果选择了一个文件夹，TortoiseSVN会首先会递归的访问可以添加的条目。

E.2.25. #导入

```
svn import -m LogMessage PATH URL
```

日志消息是日志编辑框的内容。它可以为空。

E.2.26. #追溯

```
svn blame -r N:M -v PATH
svn log -r N:M PATH
```

如果你使用TortoiseBlame来查看追溯信息，文件日志也需要在工具提示上显示日志信息，如果你以文件方式查看追溯，这个信息是不需要的。

E.2.27. #加入忽略列表

```
svn propget svn:ignore PATH > tempfile
{编辑新的忽略内容到tempfile文件中}
```

```
svn propset svn:ignore -F tempfile PATH
```

因为svn:ignore属性通常是多行的，这里是通过文件显示所修改的内容，而不是直接使用命令行操作。

E.2.28.#创建补丁

```
svn diff PATH > patch-file
```

TortoiseSVN通过比较工作拷贝和基础版本(BASE version),使用统一的diff(差异)格式创一个补丁文件。

E.2.29.#应用补丁(Apply Patch)

如果补丁和工作副本不是同一版本的话，那么应用补丁会是一件很棘手的事情。幸运的是，你可以使用TortoiseMerge(在Subversion中没有等同的工具)。

附录#F. #实现细节

这篇附录包括更详细的关于TortoiseSVN特性实现的讨论

F.1. #图标重载

Every file and folder has a Subversion status value as reported by the Subversion library. In the command line client, these are represented by single letter codes, but in TortoiseSVN they are shown graphically using the icon overlays. Because the number of overlays is very limited, each overlay may represent one of several status values.

The Conflicted overlay is used to represent the conflicted state, where an update or switch results in conflicts between local changes and changes downloaded from the repository. It is also used to indicate the obstructed state, which can occur when an operation is unable to complete.

The Modified overlay represents the modified state, where you have made local modifications, the merged state, where changes from the repository have been merged with local changes, and the replaced state, where a file has been deleted and replaced by another different file with the same name.

The Deleted overlay represents the deleted state, where an item is scheduled for deletion, or the missing state, where an item is not present. Naturally an item which is missing cannot have an overlay itself, but the parent folder can be marked if one of its child items is missing.

The Added overlay is simply used to represent the added status when an item has been added to version control.

The In Subversion overlay is used to represent an item which is in the normal state, or a versioned item whose state is not yet known. Because TortoiseSVN uses a background caching process to gather status, it may take a few seconds before the overlay updates.

The Needs Lock overlay is used to indicate when a file has the svn:needs-lock property set.

已经锁定覆盖用于本地工作副本持有此文件的锁。

The Ignored overlay is used to represent an item which is in the ignored state, either due to a global ignore pattern, or the svn:ignore property of the parent folder. This overlay is optional.

The Unversioned overlay is used to represent an item which is in the unversioned state. This is an item in a versioned folder, but which is not under version control itself. This overlay is optional.

If an item has Subversion status none (the item is not within a working copy) then no overlay is shown. If you have chosen to disable the Ignored and Unversioned overlays then no overlay will be shown for those files either.

每个条目只有一个 Subversion 状态。例如一个文件可以被本地修改，同时被标记为删除。Subversion 返回一个状态 - 即已经删除。这些优先级是 Subversion 自己定义的。

当 TortoiseSVN 递归显示状态时(默认配置)，每个目录用重载图标显示自己的状态和所有子孙的状态。为了显示单个概要重载，我们使用上述优先级决定使用哪个重载，冲突重载使用最高优先级。

In fact, you may find that not all of these icons are used on your system. This is because the number of overlays allowed by Windows is limited to 15. Windows uses 4 of those, and the remaining 11 can be used by other applications. If there are not enough overlay slots available, TortoiseSVN tries to be a Good Citizen (TM) and limits its use of overlays to give other apps a chance.

Since there are Tortoise clients available for other version control systems, we've created a shared component which is responsible for showing the overlay icons. The technical details are not important here, all you need to know is that this shared component allows all Tortoise clients to use the same overlays and therefore the limit of 11 available slots isn't used up by installing more than one Tortoise client. Of course there's one small drawback: all Tortoise clients use the same overlay icons, so you can't figure out by the overlay icons what version control system a working copy is using.

- Normal, Modified and Conflicted are always loaded and visible.
- Deleted is loaded if possible, but falls back to Modified if there are not enough slots.
- Read-Only is loaded if possible, but falls back to Normal if there are not enough slots.
- 只读只要有可能就载入，但如果没有足够的空位就使用正常来代替。
- Added is loaded if possible, but falls back to Modified if there are not enough slots.

附录#G. #语言包和拼写检查器

标准的安装包是只支持英文的,但是你可以在安装好之后。下载分别的安装包(language packs)以及拼写词典(spell check dictionaries)。

G.1. #语言包

The TortoiseSVN user interface has been translated into many different languages, so you may be able to download a language pack to suit your needs. You can find the language packs on our translation status page [http://tortoisesvn.net/translation_status]. And if there is no language pack available, why not join the team and submit your own translation ;-)

Each language pack is packaged as a .msi installer. Just run the install program and follow the instructions. After the installation finishes, the translation will be available.

The documentation has also been translated into several different languages. You can download translated manuals from the support page [http://tortoisesvn.net/support] on our website.

G.2. #拼写检查器

TortoiseSVN 包括了一个拼写检查器，可以检查你的提交日志信息，当你的项目语言不是你的本地语言时尤其有用，拼写检查器使用 OpenOffice [http://openoffice.org] 和 Mozilla [http://mozilla.org] 相同的词典。

The installer automatically adds the US and UK English dictionaries. If you want other languages, the easiest option is simply to install one of TortoiseSVN's language packs. This will install the appropriate dictionary files as well as the TortoiseSVN local user interface. After the installation finishes, the dictionary will be available too.

Or you can install the dictionaries yourself. If you have OpenOffice or Mozilla installed, you can copy those dictionaries, which are located in the installation folders for those applications. Otherwise, you need to download the required dictionary files from http://wiki.services.openoffice.org/wiki/Dictionaries.

一旦你得到了词典文件，你可能需要重命名文件，这样文件名只包含位置信息，例如:

- en_US.aff
- en_US.dic

然后把它们复制到TortoiseSVN 安装目录的 bin 子目录，通常情况下，可能是在 C:\Program Files\TortoiseSVN\bin。如果你不希望弄乱bin子目录，你可以将拼写检查文件放置在C:\Program Files\TortoiseSVN\Languages，如果那个目录不存在，你可以自己创建，当你下次启动TortoiseSVN 时，就可以使用拼写检查器。

如果你安装了多个词典，TortoiseSVN 使用下面的规则选择一个。

1. 检查 tsvn:projectlanguage 设置，关于设置项目属性可以参考第 4.17 节 “项目设置”
2. 如果没有设置项目语言，或者那个语言没有安装，尝试使用对应 Windows 区域信息的语言。
3. If the exact Windows locale doesn't work, try the “Base” language, e.g. de_CH (Swiss-German) falls back to de_DE (German).
4. 如果以上都没有效果，则缺省语言是英语，包含在标准安装中。

术语表

BDB	伯克利DB(Berkeley DB)，版本库可以使用的一种经过充分测试的后台数据库实现，不能在通过网络共享的文件系统上使用，伯克利DB是Subversion 1.2 版本以前的缺省版本库格式。
FSFS	一个专用于 Subversion 版本库的文件系统后端，可以使用网络文件系统(例如 NFS 或 SMBFS)。是 1.2 版本及其后的版本库之默认后端。
GPO	组策略对象。
Revision	每当你提交一组修改，你会在版本库创建一个“修订版本”，每个修订代表了版本库树在历史上某一点的状态，如果你希望回到历史，你可以回到以前的修订版本N。 另一种情况下，你可以把修订看作修订版本建立的修改集。
SVN	Subversion 的常见缩写形式。 Subversion 的“svnserve”版本库服务器使用的自定义协议名称。
冲突	当版本库的修改合并到本地修改，有时候修改发生在同一行，这种情况下，Subversion 不能自动决定使用文件的那一行，在提交之前，你需要手工编辑文件解决冲突。
分支	一个经常被开发者在特定时期跟踪两个独立路径时使用的术语，它描述了在版本控制系统中发生的事情。你可以从主开发线创建分支，从而开发新特性，不会使主开发线陷入不稳定状态。或者你创建一个稳定发布版的分支，仅用于修复问题，新的开发位于不稳定的主开发线。在 Subversion 中，分支用“廉价拷贝”实现。
切换	就像“Update-to-revision”从历史上改变了工作副本的视点，“Switch”改变了工作副本空间上的视点。当主干和分支只有微小差别时，这个命令非常有用，你可以在目录之间跳转，而只会有很小区别需要传输。
删除	当你删除一个受版本控制的项(并且提交了修改)，在那么在此提交版本后，它就消失于版本库中。当然，它仍旧包含在版本库的早期版本中，所以你还可以访问它。如果需要，你可以复制一个已删除项，包含历史的彻底“复活”它。
历史	显示文件或目录的历史修订，也被称为“Log”。
合并	这个过程会查看版本库添加到工作副本的的修改，而不会破坏你在本地的修改，有时候这些修改可能不会自动的结合，也就是冲突了。 在你更新工作副本时会自动合并，你也可以使用TortoiseSVN的合并命令从另一条分支进行合并。
基础版本(BASE revision)	当前工作副本里的文件或目录的基础版本。是文件或目录最后被检出、更新或者提交时的版本。基础版本通常和HEAD版本不一致。
复制	在 Subversion 版本库，你可以创建一个文件或整个目录树的副本，这是通过“廉价复制”实现的，看起来很像链接到原来的位置，几乎不占用任何空间。创建一个保存历史的副本，这样你就可以跟踪副本之前的修改。
导入	在一个修订里将整个目录导入到版本库的 Subversion 命令。
属性	除了版本控制文件和目录，Subversion 允许你添加版本控制的元数据 - 被称作每个文件和目录的“属性” 。每个属性都有一个

	名称和一个值，非常类似于注册表键。Subversion 有一些内置的特别属性，例如 svn:eol-style。TortoiseSVN 也有一些类似的，例如 tsvn:logminsize，你可以选择名称和值添加你自己的属性。
工作副本	这是你的本地“沙盒”，这个区域是你工作在版本控制文件的地方，它一般存在于你的本地磁盘，你可以使用“Checkout”从版本库创建一个工作副本，然后使用“Commit”将修改传递回版本库。
差异	“显示区别”的快捷方式，当你希望查看你所做修改的时候非常有用。
恢复	Subversion 会为每个更新到工作副本的文件保留一份“原始”副本。如果你做出了修改，并希望取消修改，你可以使用“revert”回到原始状态。
提交	一个 Subversion 操作，用来将本地修改的内容传递回版本库，创建一个新的版本库修订版本。
日志	显示一个文件或是文件夹的版本历史。也就是“历史”。
更新	这个命令将最新的修改从版本库下载到工作副本，合并其他人的修改和工作副本的本地修改。
最新版本(HEAD revision)	版本库里文件或目录的最新版本。
检出	一个 Subversion 命令在空目录通过从版本库下载版本控制的文件来创建本地工作副本。
添加	向你的工作副本中增加文件或者目录时使用的 Subversion 命令。在你提交的时候新的项就会被加入到版本库中。
清理	摘自 Subversion 手册：“ 递归的清理工作目录，删除锁，恢复未完成的工作。如果你遇到工作目录已经锁定错误，运行此命令删除过时的锁，使你的工作副本恢复到可用状态。”注释：在此上下文中“锁”指的是本地文件系统中的锁，不是版本库的锁。
版本属性(revprop)	就像文件，版本库的每个修订也可以有属性。一些特殊的修订属性会在修订版本创建时自动生成，例如: svn:date svn:author svn:log代表了提交的时间，提交者和日志信息。这些属性可以编辑，但是这些属性都不是版本控制的，所以任何修改都是永久的，不可回退的。
版本库	版本库是进行数据存储和维护的中心。版本库既可以由分布在网络上的若干数据库或者文件组成，也可以存放在用户不需要通过网络就可以直接访问的某个位置。
补丁	如果工作副本只有文本文件有修改，也可以使用 Subversion 的 Diff 命令生成标准区别格式的单文件的修改摘要。这种文件通常被叫做“补丁”，可以邮寄给任何人，使之可以应用到另一个工作副本。一些没有提交访问的人可以通过提交补丁文件给有授权的人来应用补丁，或者是在不确定修改时提交补丁给别人进行评审。
解决	当合并之后版本库的文件进入了冲突状态，必须有人用编辑器解决冲突(或者是TortoiseMerge)，这个过程称作“解决冲突”，当此过程结束，你可以将冲突文件标示为解决，将会运行提交这个文件。
输出	这个命令创建了一个版本控制目录的副本，就像工作副本，但是没有.svn目录。
追溯	这个命令只能用于文本文件，它将会标记每一行来显示版本库修订版本的最后修改的修订和作出修改的人。我们的GUI实现叫做

	TortoiseBlame，在你将鼠标移到修订版本号码上时，它也会显示时间和日志信息。
重新定位	如果你的版本库移动了，或许是因为移动到了一个新的目录，或者是域名改变，你需要“relocate”你的工作副本，这样你的版本库URL指向新的地址。 注意： 工作副本必须是指向同一个版本库的同一个位置，是版本库本身移动了。在其他几种情况下，你很有可能是需要“Switch”命令。
锁	当你加锁一个版本控制项目，除了在发出加锁命令的工作目录，它在版本库中被标记为不可提交的。

索引

符号