TortoiseSVN

针对 Windows 平台的 Subversion 客户端

版本 1.6.16

Stefan Küng
Lübbe Onken
Simon Large

目录

插图清单

表格清单

前言

TortoiseSVN

- 你是否在一个团队中工作？
- 是否发生过这样的情况： 当你在修改一个文件时，其他人也在修改这个文件？而你是否因此丢失过自己所作的修改呢？
- 是否曾经保存完一个修改，然后又想把个文件恢复到修改以前的状态？是否曾经希望能够看到一个文件以前某个时间点的状态？
- 是否曾经在项目中发现了一个 BUG，然后想调查它是什么时候产生的？

如果这些问题中的任何一个回答“是”的话，那么 TortoiseSVN 就是为你准备的！请继续读下去，你就能知道怎样让 TortoiseSVN 对你的工作起到帮助，这其实并不困难。

1. □致读者

本书面向这样的计算机用户： 希望使用 Subversion 管理数据，但又不愿意使用 Subversion 的命令行客户端。因为 TortoiseSVN 是 Windows 的外壳扩展应用，所以我们假设用户很熟悉 Windows 资源管理器的使用。

2. □阅读指南

在前言里简单介绍了 TortoiseSVN 项目，以及其开发团体，还介绍了使用及分发该软件应遵循的许可证。

在第 1 章简介一章里解释了 TortoiseSVN 是什么，能够做什么，它的开发过程以及在你的个人电脑中安装它的基础知识。

在第 2 章Basic Version-Control Concepts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。

3. □TortoiseSVN 是完全免费的！

TortoiseSVN 是免费的，你不需要为使用它而付费，可以用任何你希望的方式使用它，它开发的许可证是 GNU General Public License (GPL)。

TortoiseSVN is an Open Source project. That means you have full read access to the source code of this program. You can browse it on this link http://code.google.com/p/tortoisesvn/source/browse/. You will be prompted to enter username and password. The username is guest, and the password must be left blank. The most recent version (where we're currently working) is located under /trunk/, and the released versions are located under /tags/.

4. 社区

TortoiseSVN 和 Subversion 由工作在这些项目的社区成员开发。他们来自全世界不同的国家，联合起来创造美妙的程序。

5. 致谢

Tim Kemp
: TortoiseSVN 项目的发起者

Stefan Küng
: TortoiseSVN 的主要开发者

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

Simon Large
: 帮助编写文档和跟踪错误

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
: 为了那些补丁，问题报告和新创意，以及在邮件列表里通过回答问题帮助别人。

我们的捐赠者
: 他们发送给我们的那些音乐带来了快乐

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。

User Actions are indicated using a bold font. Alt+A: press the Alt-Key on your keyboard and while holding it down press the A-Key as well. Right-drag: press the right mouse button and while holding it down drag the items to the new location.

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。

需要特别关注的地方，如果忽略这些警告，会导致数据损坏或其他令人讨厌的事情。

第□1□章□简介

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。

1.1.□什么是 TortoiseSVN?

TortoiseSVN 是 Subversion 版本控制系统的一个免费开源客户端，可以超越时间的管理文件和目录。文件保存在中央版本库，除了能记住文件和目录的每次修改以外，版本库非常像普通的文件服务器。你可以将文件恢复到过去的版本，并且可以通过检查历史知道数据做了哪些修改，谁做的修改。这就是为什么许多人将 Subversion 和版本控制系统看作一种“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。

1.2.□TortoiseSVN 的历史

在2002年，Tim Kemp 发现 Subversion 是一个很好的版本控制系统，但是没有好的图形化客户端，创建一个作为 Windows 外壳集成的 Subversion 客户端的创意来自 TortoiseCVS，一个非常类似的 CVS 客户端。

Tim 学习了 TortoiseCVS 的源代码，将其作为 TortoiseSVN 的基础，然后开始这个项目，注册了域名 tortoisesvn.org，并将源代码提交到网上。此时 Stefan Küng 正在寻找一个好的免费版本控制系统，他发现了 Subversion 和 TortoiseSVN，由于 TortoiseSVN 还不能够使用，他加入了这个项目并开始编程。很快他便重写了大多数代码，并且开始添加命令和特性，此时，最初的代码都已经不复存在了。

随着 Subversion 越来越稳定，吸引了越来越多的用户开始使用 TortoiseSVN 作为他们的 Subversion 客户端。用户群增长迅速(每天都持续增长)。Lübbe Onken 提供了许多漂亮图标和 TortoiseSVN 的 logo，细心照料网站并且管理翻译。

1.3.□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 与Windows 外壳(例如资源管理器)无缝集成，你可以保持在熟悉的工具上工作，不需要在每次使用版本控制功能时切换应用程序。

并且你不一定必须使用 Windows 资源管理器，TortoiseSVN 的右键菜单可以工作在其他文件管理器，以及文件/打开对话框等标准的 Windows 应用程序中。你必须牢记，TortoiseSVN 是有意作为 Windows 资源管理器的扩展开发，因此在其他程序可能集成的并不完整，例如重载图标可能不会显示。

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。

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。

良好的维护能力
: Subversion 没有历史负担，它由一系列良好的共享 C 库实现，具有定义良好的 API，这使 Subversion 非常容易维护，可以轻易的被其他语言和程序使用。

1.4. □安装 TortoiseSVN

1.4.1. □系统要求

TortoiseSVN 可以运行在 Windows 2000 SP2，Windows XP 或更高的版本。TortoiseSVN 1.2.0 以后不再支持 Windows 98, Windows ME 和 Windows NT4，但是如果需要的话，你仍旧可以下载以前的版本。

如果在安装 TortoiseSVN 时发现了任何问题，请首先参考附录 A,常见问题(FAQ)。

1.4.2. □安装

TortoiseSVN 提供一个容易使用的安装程序。双击安装程序文件并按照提示操作。安装程序将会完成剩余的步骤。

你需要管理员权限来安装 TortoiseSVN。

1.4.3. □语言包

TortoiseSVN 的界面已经翻译成了许多种语言，所以你可以下载符合你要求的语言包。你可以在我们的翻译状态页 [http://tortoisesvn.net/translation_status]里看到语言包。如果没有你需要的，为什么不加入我们的团队并且提交你的翻译呢？-)

每一种语言包都是一个 .exe 安装程序，只要根据向导运行安装程序，当你下一次启动程序时，翻译就会生效。

1.4.4. □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。

安装程序自动添加 US 和 UK 英语词典。如果你需要其他语言，最简单的方法是安装 TortoiseSVN 的语言包，这会安装合适的词典文件和 TortoiseSVN 的本地用户界面，当你下一次启动程序时，词典也将会生效。

或者你也可以自己安装词典。如果你安装了 OpenOffice 或 Mozilla，你可以复制这些词典，位于那些应用的安装目录。否则，你需要从 http://wiki.services.openoffice.org/wiki/Dictionaries 下载必要的词典文件。

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. 如果精确的 Windows 区域信息不起作用，可以试一下“基础”语言，例如将 de_CH(Swiss-German) 修改为 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。

第□2□章□Basic Version-Control Concepts

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。

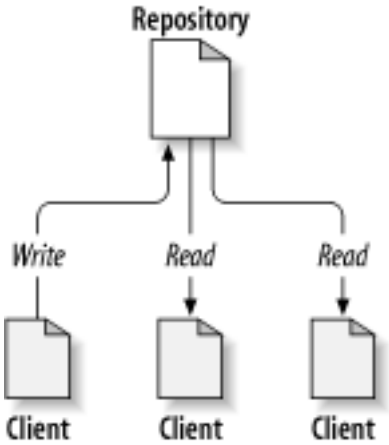


图 2.1. 一个典型的客户/服务器系统

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。

When a client reads data from the repository, it normally sees only the latest version of the filesystem tree. But the client also has the ability to view previous states of the filesystem. For example, a client can ask historical questions like, “what did this directory contain last Wednesday?”, or “who was the last person to change this file, and what changes did they make?” These are the sorts of questions that are at the heart of any version control system: systems that are designed to record and track changes to data over time.

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！

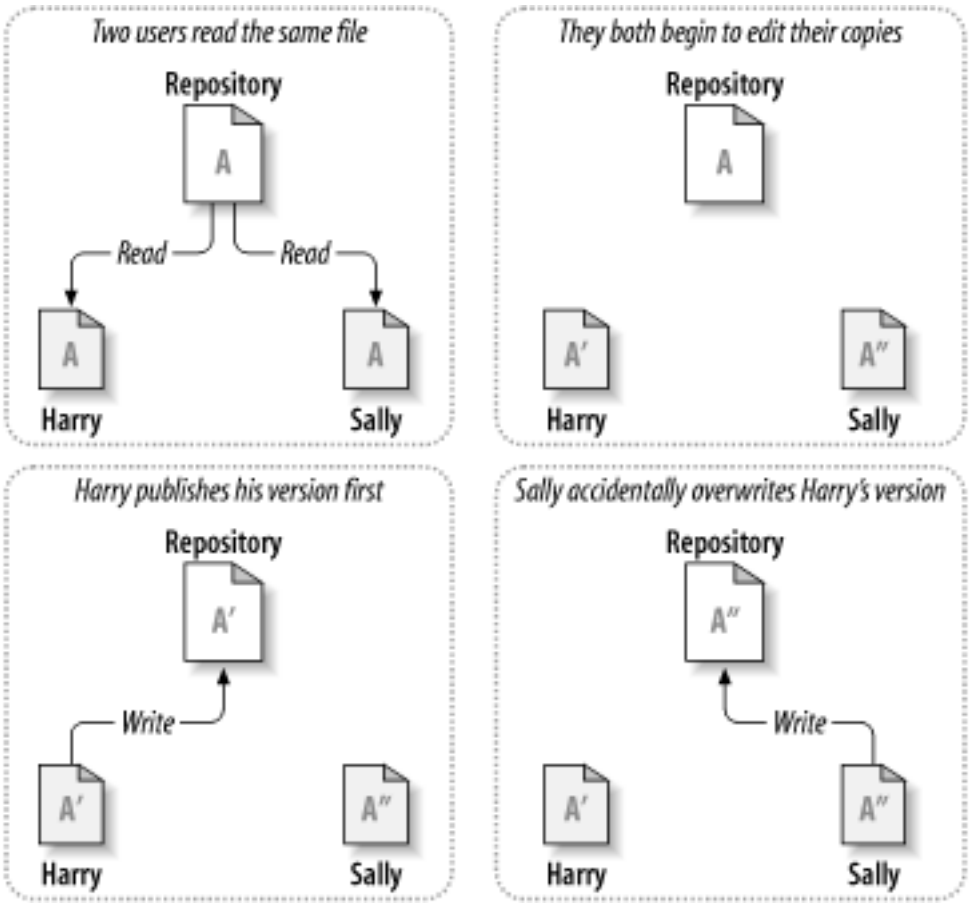


图 2.2. 需要避免的问题

2.2.2. □锁定-修改-解锁 方案

Many version control systems use a lock-modify-unlock model to address this problem, which is a very simple solution. In such a system, the repository allows only one person to change a file at a time. First Harry must lock the file before he can begin making changes to it. Locking a file is a lot like borrowing a book from the library; if Harry has locked a file, then Sally cannot make any changes to it. If she tries to lock the file, the repository will deny the request. All she can do is read the file, and wait for Harry to finish his changes and release his lock. After Harry unlocks the file, his turn is over, and now Sally can take her turn by locking and editing.

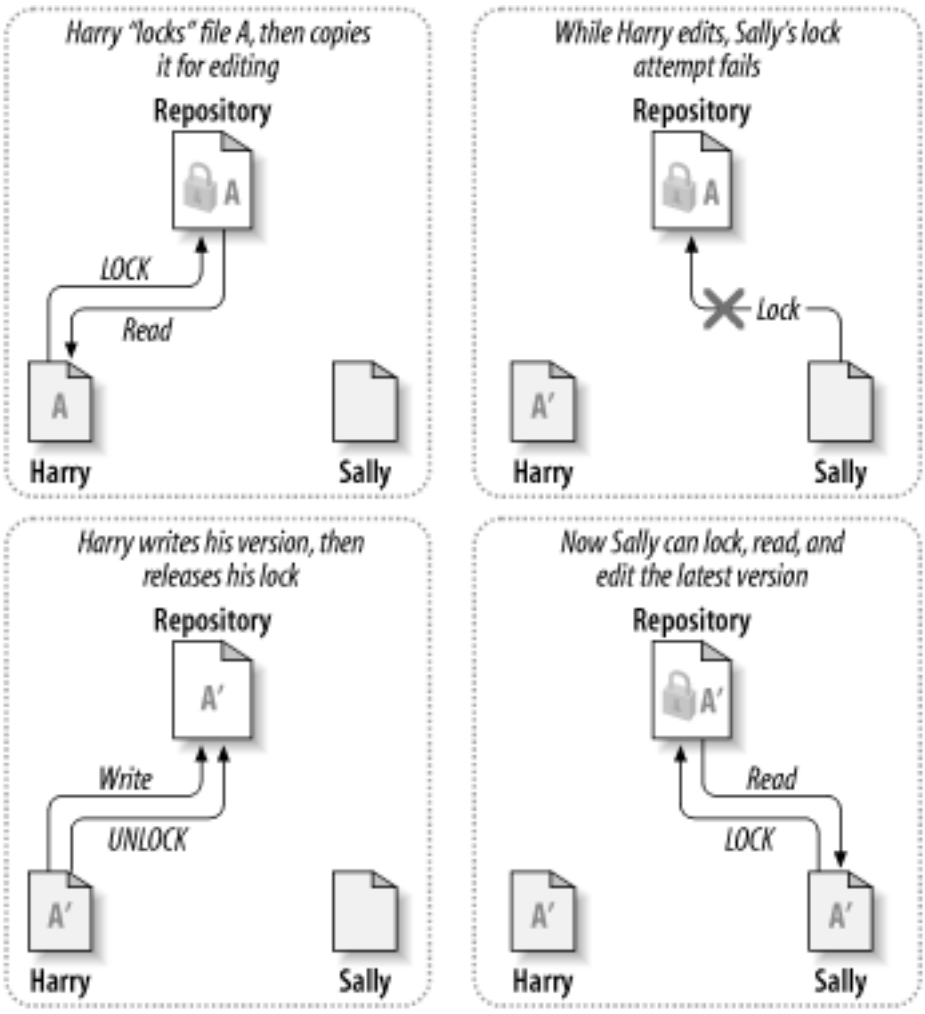


图 2.3. 锁定-修改-解锁 方案

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。

Here's an example. Say that Harry and Sally each create working copies of the same project, copied from the repository. They work concurrently, and make changes to the

same file A within their copies. Sally saves her changes to the repository first. When Harry attempts to save his changes later, the repository informs him that his file A is out-of-date. In other words, that file A in the repository has somehow changed since he last copied it. So Harry asks his client to merge any new changes from the repository into his working copy of file A. Chances are that Sally's changes don't overlap with his own; so once he has both sets of changes integrated, he saves his working copy back to the repository.

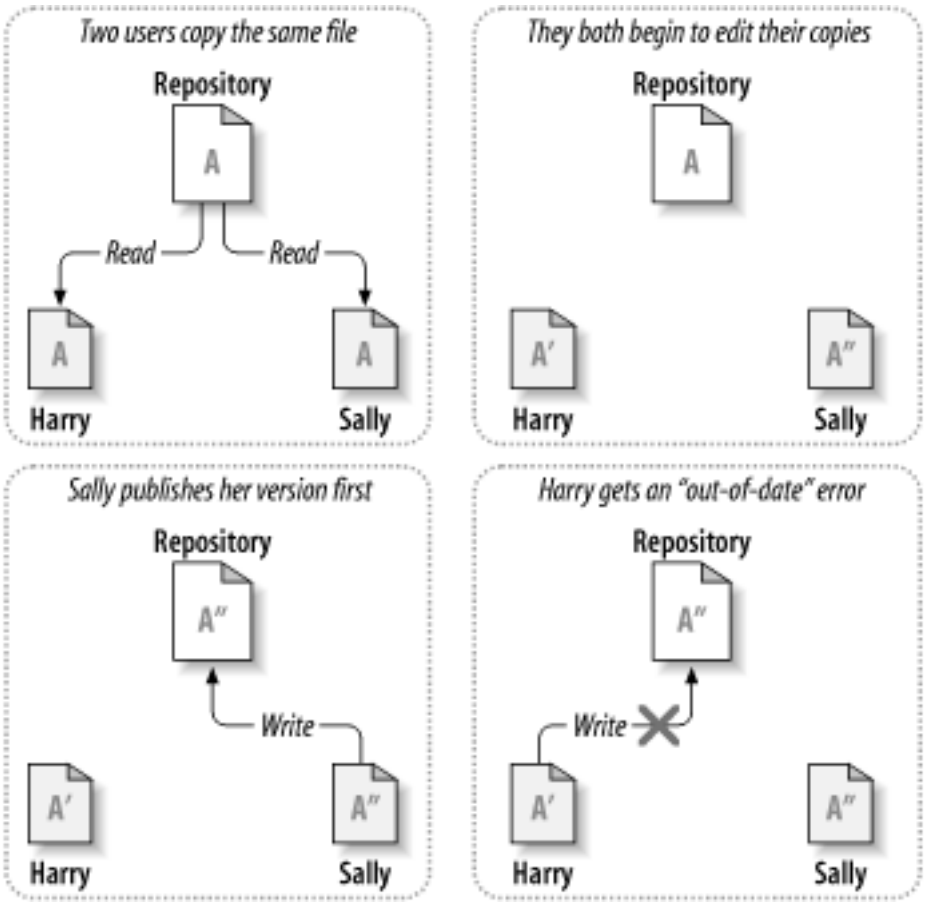


图 2.4. 复制-修改-合并 方案

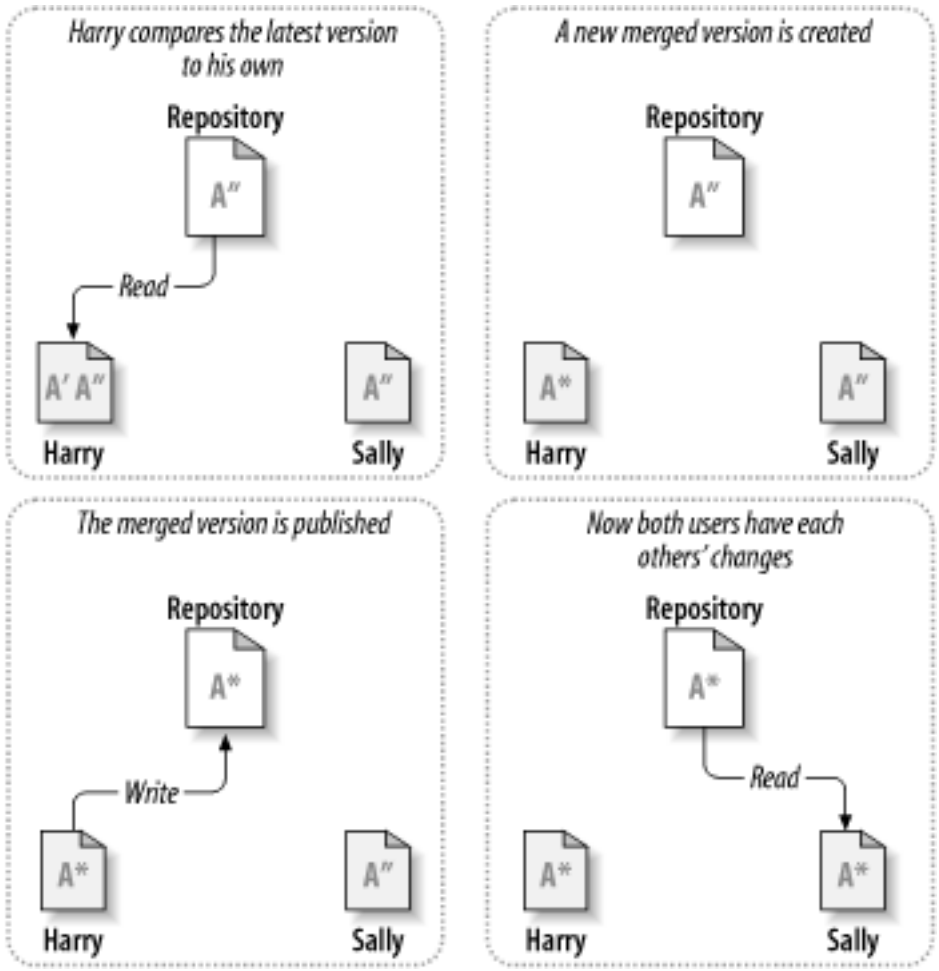


图 2.5. 复制-修改-合并 方案(续)

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。

After you've made some changes to the files in your working copy and verified that they work properly, Subversion provides you with commands to publish your changes to the other people working with you on your project (by writing to the repository). If other people publish their own changes, Subversion provides you with commands to merge those changes into your working directory (by reading from the repository).

一个工作副本也包括一些由 Subversion 创建并维护的额外文件，用来协助执行这些命令。通常情况下，你的工作副本每一个文件夹有一个以 .svn 为名的文件夹，也被叫做工作副本管理目录，这个目录里的文件能够帮助 Subversion 识别哪一个文件做过修改，哪一个文件相对于别人的工作已经过期了。

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。

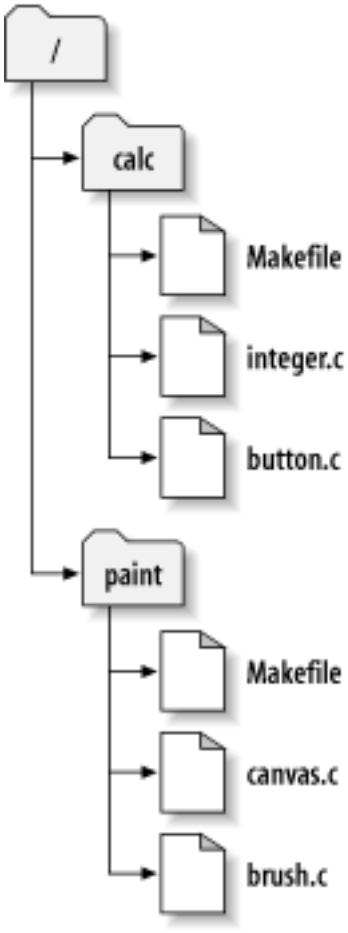


图 2.6. 版本库的文件系统

换句话说，版本库的根目录包含两个子目录：paint 和 calc。

To get a working copy, you must check out some subtree of the repository. (The term check out may sound like it has something to do with locking or reserving resources, but it doesn't; it simply creates a private copy of the project for you).

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

For the most part, Subversion's URLs use the standard syntax, allowing for server names and port numbers to be specified as part of the URL. The file:// access method is normally used for local access, although it can be used with UNC paths to a networked host. The URL therefore takes the form file://hostname/path/to/repos. For the local machine, the hostname portion of the URL is required to be either absent or localhost. For this reason, local paths normally appear with three slashes, file:///path/to/repos.

Also, users of the file:// scheme on Windows platforms will need to use an unofficially "standard" syntax for accessing repositories that are on the same machine, but on a different drive than the client's current working drive. Either of the two following URL path syntaxes will work where X is the drive on which the repository resides:

```
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”。

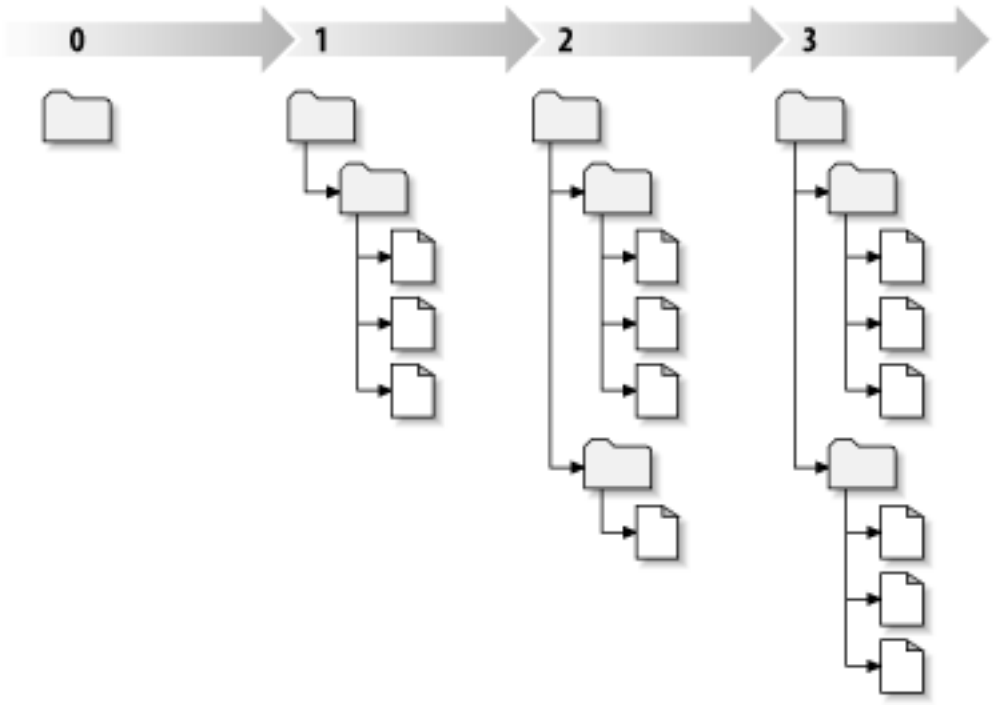


图 2.7. 版本库

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
: The file has been changed both in the working directory, and in the repository. A commit of the file will fail with an out-of-date error. The file should be updated first; an update command will attempt to merge the public changes with the local changes. If Subversion can't complete the merge in a plausible way automatically, it leaves it to the user to resolve the conflict.

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/1.5/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

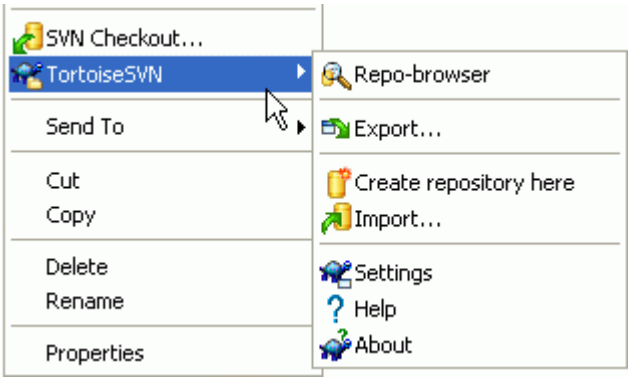


图 3.1. 未版本控制文件夹的 TortoiseSVN 菜单

1. 打开资源管理器

2. 创建一个新的文件夹，命名为SVNRepository

3. 右键点击新创建的目录，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。

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。

Future versions of TortoiseSVN will not support file:// access to BDB repositories due to these compatibility issues, although it will of course always support this repository format when accessed via a server through the svn://, http:// or https:// protocols. For this reason, we strongly recommend that any new repository which must be accessed using file:// protocol is created as FSFS.

Of course we also recommend that you don't use file:// access at all, apart from local testing purposes. Using a server is more secure and more reliable for all but single-developer use.

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 Berkeley DB 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 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。

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 “Accessing the Repository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。

There are some standard, recommended ways to organize a repository. Most people create a trunk directory to hold the “main line” of development, a branches directory to contain branch copies, and a tags directory to contain tag copies. If a repository holds only one project, then often people create these top-level directories:

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

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. 通过在那个文件夹右键，选择TortoiseSVN → 导入... 将这个结构导入到版本库中。这将导入临时文件夹到版本库的根目录形成一个基本的版本库布局。

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. If you are installing the command line tools on a Windows PC, the best way is to download the Windows installer version. It is compressed more efficiently than the .zip version, so the download is smaller, and it takes care of setting the paths for you. You can download the latest version of the Subversion command line client from http://subversion.apache.org/getting.html.

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/1.5/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。

版本库的hooks目录中有一些钩子的例子脚本，这些例子脚本适合于Unix/Linux服务器，在Windows下需要修改。钩子可以是批处理文件或可执行文件，下面是用来实现pre-revprop-change钩子的例子。

```
rem Only allow log messages to be changed.
if "%4" == "svn:log" exit 0
echo Property '%4' cannot be changed >&2
exit 1
```

请注意所有发送到标准输出的东西都会被忽略，如果你希望信息出现在拒绝提交对话框中，你需要将这些信息发送到标准错误，在一个批处理文件中使用>&2实现。

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。

To include such a link in your own html page, you need to add code which looks something like this:

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

Of course it would look even better if you included a suitable picture. You can use the TortoiseSVN logo [http://tortoisesvn.tigris.org/images/TortoiseCheckout.png] or you can provide your own image.

```
<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. □Accessing the Repository

To use TortoiseSVN (or any other Subversion client), you need a place where your repositories are located. You can either store your repositories locally and access them using the file:// protocol or you can place them on a server and access them with the http:// or svn:// protocols. The two server protocols can also be encrypted. You use https:// or svn+ssh://, or you can use svn:// with SASL.

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 。

If you don't have a server and you work alone, or if you are just evaluating Subversion and TortoiseSVN in isolation, then local repositories are probably your best choice. Just create a repository on your own PC as described earlier in 第 3 章版本库. You can skip the rest of this chapter and go directly to 第 4 章日常使用指南 to find out how to start using it.

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。

The next sections are a step-by-step guide on how you can set up such a server on a Windows machine. Of course you can also set up a server on a Linux machine, but that is beyond the scope of this guide. 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/1.5/svn.serverconfig.html].

3.6. □基于 svnserve 的服务器

3.6.1. □简介

在 Subversion 中包含 Svnserve - 一个轻型的独立服务器，它使用一个基于一般 TCP/IP 连接的定制协议。用于小型安装，或者不能使用全能 Apache 服务器的地方。

In most cases svnserve is easier to setup and runs faster than the Apache based server, although it doesn't have some of the advanced features. And now that SASL support is included it is easy to secure as well.

3.6.2. □安装 svnserve

1. 可以从这里获取最新版本的 Subversion http://subversion.apache.org/getting.html [http://subversion.tigris.org/getting.html]。另外，也可以从 ColabNet 获取一个打包好的安装程序 http://www.collab.net/downloads/subversion 。这个安装程序将会把 svnserve 设置为 Windows 服务，并且还包含了一些你需要的工具，如果你为了安全而使用 SASL。

2. 如果你已经安装了 Subversion，svnserve 已经运行，你需要在继续之前把它停下来。

3. 运行 Subversion 安装程序，如果你在你的服务器(推荐)上运行，可以跳过第 4 步。

4. 打开资源管理器，进入Subversion的安装目录(通常是C:\Program Files\Subversion)的bin目录，找到文件svnserve.exe，intl3_svn.dll，libapr.dll，libapriconv.dll，libapriutil.dll，libdb*.dll，libeay复制这些文件，或所有bin目录内的文件到你的服务器目录，例如c:\svnserve。

3.6.3. □运行 svnserve

现在svnserve已经安装了，你需要在你的server运行它，最简单的方法是在DOS窗口或者windows快捷方式输入:

```
svnserve.exe --daemon
```

svnserve将会在端口3690等待请求，--daemon选项告诉svnserve以守护进程方式运行，这样在手动终止之前不会退出。

如果你没有创建一个版本库，根据下面的Apache服务器设置指令第 3.7.4 节 “配置”。

为了验证svnserve正常工作，使用TortoiseSVN → 版本库浏览器来查看版本库。

假定你的版本库位于c:\repos\TestRepo，你的服务器叫做localhost，输入:

```
svn://localhost/repos/TestRepo
```

当被版本库浏览器提示输入。

你也可以使用 --root 选项设置根位置来限制访问服务器的目录，从而增加安全性和节约输入 svnserve URL 的时间:

```
svnserve.exe --daemon --root drive:\path\to\repository\root
```

以前面的测试为例，svnserve 现在的运行命令为:

```
svnserve.exe --daemon --root c:\repos
```

然后在 TortoiseSVN 中我们的版本库浏览器 URL 缩短为:

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

注意，当 svnserve 和版本库位于不同分区或盘符时也需要使用 --root 选项。

Svnserve 可以提供任意数量的版本库服务。只要将这些版本库放到你刚才定义的根目录下即可，然后使用相对于根的URL访问它们。

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。

3.6.3.1. □以服务形式运行 svnserve

使用普通用户直接运行 svnserve 通常不是最好的方法。它意味着你的服务器必须有一个用户登录，还要记着重新启动服务器后重新启动 svnserve。最好的方法是将 svnserve 作为 windows 服务运行。从 Subversion 1.4 开始，svnserve 可以安装为 windows 服务。

To install svnserve as a native windows service, execute the following command all on one line to create a service which is automatically started when windows starts.

```
sc create svnserve binpath= "c:\svnserve\svnserve.exe --service
    --root c:\repos" displayname= "Subversion" depend= tcpip
    start= auto
```

If any of the paths include spaces, you have to use (escaped) quotes around the path, like this:

```
sc create svnserve binpath= "
    \"C:\Program Files\Subversion\bin\svnserve.exe\"
    --service --root c:\repos" displayname= "Subversion"
    depend= tcpip start= auto
```

You can also add a description after creating the service. This will show up in the Windows Services Manager.

```
sc description svnserve "Subversion server (svnserve)"
```

注意 sc 的命令行很特殊。在 key= value 对中，key 与 = 之间不能有空格，但是在 value 之前，必须有空格。

Microsoft 现在建议服务程序使用本地服务或网络服务帐户运行，参考 The Services and Service Accounts Security Planning Guide [http://www.microsoft.com/technet/security/topics/serversecurity/serviceaccount/default.mspx]。以本地服务帐户创建服务，需要在上面的例子里追加下面几行。

```
obj= "NT AUTHORITY\LocalService"
```

请注意需要给本地服务帐户一些目录的适当权限，包括的 Subversion 和你的版本库，还有所有钩子脚本使用的应用。此帐号的内置组名是"LOCAL SERVICE"。

服务安装完毕后，你需要在服务管理器中启动它(仅此一次；当服务器重启后它会自动启动)。

为了得到更详细的信息，可参考 Windows Service Support for Svnserve [http://svn.collab.net/repos/svn/trunk/notes/windows-service.txt]。

如果你已经使用 SvnService 包装安装了早期的 svnserve，现在想使用内置服务，那么你需要将其从服务中删除(切记先停止服务！)。使用简单的命令

```
svnservice -remove
```

即可删除服务。

3.6.4. □svnserve 与基本认证

The default svnserve setup provides anonymous read-only access. This means that you can use an svn:// URL to checkout and update, or use the repo-browser in TortoiseSVN to view the repository, but you won't be able to commit any changes.

为了打开对版本库的写访问，你可以编辑版本库目录的conf/svnserve.conf文件，这个文件控制了svnserve守护进程的配置，也提供了有用的文档。

为了打开匿名的写访问，只需要简单得设置:

```
[general]
anon-access = write
```

然而，你不会知道谁修改了版本库，因为svn:author属性是空的，你也不能控制谁来修改版本库，这是一个很危险的设置。

解决这个问题的一个方法是创建密码数据库:

```
[general]
anon-access = none
auth-access = write
password-db = userfile
```

这里的 userfile 与 svnserve.conf 文件在同一个目录，这个文件也可以存在于文件系统的其他地方(当多个版本库使用相同的访问权限时尤其有用)，可以使用绝对路径，或者是 conf 的相对目录，使用 \ 或盘符不能工作。userfile 的结构如下:

```
[general]
anon-access = none
auth-access = write
```

```
password-db = userfile
```

这个例子拒绝所有的未认证用户(匿名)访问，给 userfile 中的用户读写权限。

If you maintain multiple repositories using the same password database, the use of an authentication realm will make life easier for users, as TortoiseSVN can cache your credentials so that you only have to enter them once. More information can be found in the Subversion book, specifically in the sections Create a 'users' file and realm [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnserve.html#svn.serverconfig.svnserve.auth.users] and Client Credentials Caching [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.netmodel.html#svn.serverconfig.netmodel.credcache]

3.6.5. □使用 SASL 以便更安全

3.6.5.1. □什么是 SASL?

Cyrus 简单的认证和安全层(The Cyrus Simple Authentication and Security Layer)是一个由卡耐基梅隆大学编写的开源软件。它可以为任何网络协议增加通用认证和加密的能力，并且从 Subversion 1.5 开始所有的后续版本，包括 svnserve 服务器和 TortoiseSVN 客户端知道如何使用这个库。

要获得这一选项可用性的更充分讨论，你应该看一下 Subversion 手册中的通过 SASL 使用 svnserve [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnserve.html#svn.serverconfig.svnserve.sasl] 。如果你仅仅是找一种简单的方法为 Windows 服务器设置安全的认证和加密，因此你的版本库可以安全的通过又大又乱的互联网访问，请继续阅读。

3.6.5.2. □SASL 认证

要在服务器上激活详尽的 SASL 功能，你需要做 3 件事。首先，在你的版本库的 svnserve.conf 文件中创建一个 [sasl] 节，包括这样一个配置项：

```
use-sasl = true
```

其次，在一个合适的地方创建一个名为 svn.conf 的文件 - 通常在 Subversion 的安装目录中。

第三，创建 2 个新的注册表项目来告诉 SASL 到哪里找到需要的东西。创建一个名为的 [HKEY_LOCAL_MACHINE\SOFTWARE\Carnegie Mellon\Project Cyrus\SASL Library] 注册表键并在其中创建两个新的字符串值：SearchPath 设置为 sasl*.dll 插件所在的目录路径(通常是 Subversion 的安装目录)，ConfFile 设置为 svn.conf 文件所在的目录。如果你使用 CollabNet 安装程序，这两个注册表键就已经为你创建好了。

编辑文件 svn.conf，使其包括下列内容：

```
pwcheck_method: auxprop
auxprop_plugin: sasldb
mech_list: DIGEST-MD5
sasldb_path: C:\TortoiseSVN\sasldb
```

最后一行指示认证数据库的位置，认证数据库是一个名为 sasldb 的文件。它可以在任何地方，不过一个方便的选择是在版本库的上层目录中。确认 svnserve 服务有读取这个文件的权限。

如果 svnserve 已经在运行，你需要重启服务，并确保它读取了更新后的配置参数。

现在所有的东西已经设置完成，你要做的事情就是创建用户和密码。你需要 saslpasswd2 程序来做这件事。如果你使用 CollabNet 安装程序，这个程序会在安装目录内。使用像这样的命令：

```
saslpasswd2 -c -f C:\TortoiseSVN\sasldb -u realm username
```

选项 -f 指明数据库的位置，realm 必须与版本库的 svnserve.conf 文件中设置相同的值，username 就是你要使用的用户名。注意 realm 不允许包含空字符。

你可以使用 sasldblistusers2 程序列出数据库中储存的用户名。

3.6.5.3. □SASL 加密

为了启用或禁用加密的不同等级，你可以设置版本库内文件 svnserve.conf 中的两个值：

```
[sasl]
use-sasl = true
min-encryption = 128
max-encryption = 256
```

变量 min-encryption 和 max-encryption 控制服务器所需要的加密强度。要完全禁用加密，就将这 2 个变量的值都设为 0。要启用简单的数据校验(例如，为了防止篡改和保证数据的完整，不加密)，就将这 2 个值都设为 1。如果你想允许(但不强制)加密，将最小值设为 0，最大值设为任意位数。要强制加密，将这 2 个值设为大于 1 的数字。在前面的例子中，我们要求客户端至少进行 128 位加密，但是不大于 256 位加密。

3.6.6. □使用 svn+ssh 认证

另一种 svnserve 认证的方法是使用 SSH 来建立请求通道。它不像设置 SASL 那样简单，但是某些场合却很有用。

通过此方法，svnserve 不会作为守护进程启动，而是 SSH 为你启动 svnserve，以 SSH 认证的用户身份运行，为此，你需要在你的服务器上有 SSH 守护进程。

设置服务器的基本方法请参见附录 G,用 SSH 使服务器更安全。你可以在常见问题(FAQ)中使用关键词 “SSH” 找到其它 SSH 相关的主题。

更多的关于svnserve的信息可以看《使用 Subversion 进行版本管理》 [http://svnbook.red-bean.com]。

3.6.7. □svnserve 基于路径的授权

从 Subversion 1.3 开始，svnserve 支持与 mod_authz_svn 相同的基于路径的授权模式，你需要编辑版本库路径下的 conf/svnserve.conf 引用的授权文件。

```
[general]
authz-db = authz
```

在这里，authz 是你创建用来定义访问权限的文件，你可以为每一个版本库使用单独的文件，或者为所有的版本库使用相同的文件，关于此文件的格式可以查看第 3.7.6 节 “路径为基础的授权”。

3.7. □基于 Apache 的服务器

3.7.1. □简介

所有可能的服务器当中，Apache为基础的服务器是最灵活的，尽管配置有一点复杂，但是提供了其他服务器没有的便利：

WebDAV
: The Apache based Subversion server uses the WebDAV protocol which is supported by many other programs as well. You could e.g. mount such a repository as a “Web folder” in the Windows explorer and then access it like any other folder in the file system.

浏览版本库
: 你可以将浏览器指向版本库的URL，无需安装Subversion客户端就可以浏览内容，这样可以扩大访问你数据的用户圈。

认证
: 你可以使用所有Apache支持的认证机制，包括SSPI和LDAP。

安全
: 因为Apache非常稳定和安全，你的版本库可以自动获得同样的安全性，包括SSL加密。

3.7.2. □安装 Apache

安装 Apache 的先决条件是有一台安装了 Windows 2000，Windows XP SP1+，Windows 2003，Vista 或 Windows Server 2008 的计算机。

请注意，Windows XP 如果没有安装 SP1 将会导致不正常的网络传输，从而搞坏你的版本库！

1. 从 http://httpd.apache.org/download.cgi 下载最新版本的 Apache，请确认你下载的版本是 2.2.x — 1.3.xx 的版本不能工作！

   点击 other files，然后浏览 binaries/win32，可以找到 msi 格式的 Apache 安装文件。你可以选择 msi 文件 apache-2.2.x-win32-x86-openssl-0.9.x.msi (这个包含有 OpenSSL)。

2. 一旦你有了 Apache2 安装程序，你可以双击它，然后它会指导你的安装过程。请确认你输入的服务器 URL 正确(如果你的服务器没有 DNS 名称，请直接输入 IP 地址)。我建议为所有用户在 80 端口安装 Apache 服务。注意：如果你已经有了 IIS 或其他监听 80 端口的程序，安装会失败。如果发生这种情况，直接到程序的安装目录 \Apache Group\Apache2\conf，打开 httpd.conf。编辑文件的 Listen 80 为其他可用的端口，例如 Listen 81，然后重新启动 — 这样就不会那个问题了。

3. 现在可以测试 Apache 服务器是否正确运行，将浏览器指向 http://localhost/ — 将会看到一个预先配置的网站。

如果你决定将Apache安装为服务，缺省情况以本地系统帐户运行会发出警告，更安全的方法是为Apache创建一个单独的运行帐户。

请确认Apache运行的帐户是版本库目录的访问控制列表(右键目录|属性|安全)中一个明确的条目，对目录有完全的控制能力，否则，用户不能提交他们的修改。

即使Apache运行于本地系统，你仍然需要这个条目(这种情况下将是SYSTEM帐户)。

如果没有配置 Apache 的此访问权限，你的用户会得到“拒绝访问(Access denied)”的错误信息，在 Apache 的错误日志中的错误代码是 500。

3.7.3. □安装 Subversion

1. Download the latest version of the Subversion Win32 binaries for Apache. Be sure to get the right version to integrate with your version of Apache, otherwise you will get an obscure error message when you try to restart. If you have Apache 2.2.x go to http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=8100.

2. 运行Subversion安装程序，并根据指导安装，如果Subversion认识到你安装了Apache，你就几乎完成了工作，如果它没有找到Apache服务器，你还有额外的步骤。

3.

   使用Windwos资源管理器，来到Subversion的安装目录(通常是c:\program files\Subversion)，找到文件/httpd/mod_dav_svn.so和mod_authz_svn.so，复制这些文件到Apache的模块目录(通常是c:\program files\apache group\apache2\modules)。

4. 从 Subversion 安装目录将 /bin/libdb*.dll 和 /bin/intl3_svn.dll 复制到 Apache 的 bin 目录。

5. 使用记事本之类的文本编辑器修改Apache的配置文件(通常是 C:\Program Files\Apache Group\Apache2\conf\httpd.conf)，做出如下修改:

   去掉以下几行的注释(删除 '#'标记):

   ```
   #LoadModule dav_fs_module modules/mod_dav_fs.so
   #LoadModule dav_module modules/mod_dav.so
   ```

   将以下两行到 LoadModule 节的末尾。

   ```
   LoadModule dav_svn_module modules/mod_dav_svn.so
   LoadModule authz_svn_module modules/mod_authz_svn.so
   ```

3.7.4. □配置

现在你已经配置了 Apache 和 Subversion，但是 Apache 不知道如何处理 Subversion 客户端，例如TortoiseSVN。为了让 Apache 知道哪个 URL 是用作 Subversion 版本库，你需要使用任意文本编辑器(例如记事本)编辑 Apache 的配置文件(通常是 c:\program files\apache group\apache2\conf\httpd.conf):

1. At the end of the config file add the following lines:

   ```
   <Location /svn>
     DAV svn
     SVNListParentPath on
     SVNParentPath D:\SVN
     #SVNIndexXSLT "/svnindex.xsl"
     AuthType Basic
     AuthName "Subversion repositories"
     AuthUserFile passwd
     #AuthzSVNAccessFile svnaccessfile
     Require valid-user
   </Location>
   ```

   This configures Apache so that all your Subversion repositories are physically located below D:\SVN. The repositories are served to the outside world from the URL: http://MyServer/svn/ . Access is restricted to known users/passwords listed in the passwd file.

2. 为了创建 passwd 文件，再次打开命令行提示符(DOS 窗口)，进入 apache2 目录(通常是 c:\program files\apache group\apache2)，通过输入下面的命令创建文件

   ```
   bin\htpasswd -c passwd <username>
   ```

   它将会创建名为 passwd 的文件用于认证。用下面的命令增加其它用户

```
bin\htpasswd passwd <username>
```

3. 再次重启Apache服务。

4. 将浏览器指向http://MyServer/svn/MyNewRepository(MyNewRepository是你此前创建的版本库名)，如果一切正常，你会被提示输入用户名和密码，然后你会看到版本库的内容。

你刚才输入的简短解释是:

设置	解释
<Location /svn>	意思是Subversion版本库的URL是http://MyServer/svn/
DAV svn	告诉Apache是哪个模块响应那个URL的请求－此刻是Subversion模块。
SVNListParentPath on	对于 Subversion 1.3 或者更高版本，这个指示会列出所有SVNParentPath 中的版本库。
SVNParentPath D:\SVN	告诉Subversion需要查看的版本库位于D:\SVN之下
SVNIndexXSLT "/svnindex.xsl"	使用它可以在用浏览器浏览时更好看。
AuthType Basic	激活基本认证，就是用户名/密码
AuthName "Subversion repositories"	用来说明何时弹出要求用户输入认证信息的认证对话框
AuthUserFile passwd	指定使用的认证密码文件
AuthzSVNAccessFile	位置Subversion版本库的访问控制文件的路径
Require valid-user	指定只有输入了正确的用户/密码的用户可以访问URL

表 3.1. 设置 Apache 的httpd.conf

但是那只是一个例子。对于 Apache 你有很多可能的选择。

- 如果你希望所有人可以读你的版本库，但是只有特定用户可以写，你可以修改下面几行

  ```
  Require valid-user
  ```

  to

  ```
  <LimitExcept GET PROPFIND OPTIONS REPORT>
  Require valid-user
  </LimitExcept>
  ```

- 使用 passwd 可以整体的控制对版本库的访问，但是如果你希望精确的控制版本库目录访问，你可以去掉下行的注释

  ```
  #AuthzSVNAccessFile svnaccessfile
  ```

  ，并且创建 Subversion 的访问控制文件。Apache 将会确保只有有效的用户可以访问你的 /svn 目录，然后将用户名传递给 Subversion 的 AuthzSVNAccessFile 模块，这样就可以根据 Subversion 访问控制文件内的规则实现更细粒度的访问控制。请注意路径可以是 repos:path 或简单的 path，如果你不指定特定的版本库，访问控制规则会应用到

SVNParentPath 下所有的版本库。使用的授权策略文件的格式在第 3.7.6 节 “路径为基础的授权”描述。

- 如果要使浏览器浏览仓库时更“漂亮”，请将去掉下行注释

```
#SVNIndexXSLT "/svnindex.xsl"
```

，将文件 svnindex.xsl, svnindex.css 和 menucheckout.ico 放到你的文档根目录中(通常是 C:/Program Files/Apache Group/Apache2/htdocs)。 这个目录在 Apache 配置文件中用 DocumentRoot 指示设置。

You can get those three files directly from our source repository at http://tortoisesvn.googlecode.com/svn/trunk/contrib/svnindex. (第 3 节 “TortoiseSVN 是完全免费的！” explains how to access the TortoiseSVN source repository).

TortoiseSVN 版本库中的 XSL 文件有个特性：如果你用浏览器浏览版本库，那么每个版本库中的目录右边会有个图标。如果你点击此图标，那么 TortoiseSVN 会为此 URL 启动检出对话框。

3.7.5. □多版本库

如果你使用 SVNParentPath 指示，你就不必在每次添加新 Subversion 版本库时修改 Apache 的配置文件，只需要在第一个版本库所在的位置建立新的版本库就可以了。在我的公司，我可以使用 SMB(普通的 windows 文件访问)直接访问服务器的文件夹，所以我直接在那里创建一个目录，运行 TortoiseSVN 命令TortoiseSVN → 在此创建版本库...，然后一个新的项目建立了...

如果你使用 Subversion 1.3 或更高版本，可以使用 SVNListParentPath on 指示，这样当你使用浏览器访问父路径而不是具体某个版本库时 Apache 就会显示所有版本库列表。

3.7.6. □路径为基础的授权

mod_authz_svn 模块可以根据用户名和路径实现细粒度的权限控制，它对 Apache 服务器有效，在 Subversion 1.3 以上版本的 svnserve 中也实现了基于路径的授权。

一个可能的例子:

```
[groups]
admin = john, kate
devteam1 = john, rachel, sally
devteam2 = kate, peter, mark
docs = bob, jane, mike
training = zak
# Default access rule for ALL repositories
# Everyone can read, admins can write, Dan German is excluded.
[/]
* = r
@admin = rw
dangerman =
# Allow developers complete access to their project repos
[proj1:/]
@devteam1 = rw
[proj2:/]
@devteam2 = rw
[bigproj:/]
@devteam1 = rw
```

```
@devteam2 = rw
trevor = rw
# Give the doc people write access to all the docs folders
[/trunk/doc]
@docs = rw
# Give trainees write access in the training repository only
[TrainingRepos:/]
@training = rw
```

请注意，检查每一条路径是一件消耗极大的操作，特别是修订版本日志，服务器会检查在每一个修订版本的每一条路径是否可读，对于影响很多文件的修订将会花费很多时间。

认证和授权是不同的处理过程，如果用户希望获得对版本库的访问，他需要通过全部检查，即通常的认证需求和访问控制文件的授权需求。

3.7.7. □使用 Windows 域认证

你已经注意到了，你需要为每个用户在passwd文件中创建用户名/密码条目，如果(因为安全原因)他们希望周期性的修改他们的密码，你需要手动的做出修改。

但是对于此问题有另一个解决方案 — 至少是你在使用域控制器的 LAN 中访问版本库时: mod_auth_sspi!

最初的 SSPI 是由 Syneapps 提供的，包括源代码。但是它的开发已经终止。不过不要失望，社区重新拾起代码并进行了改进，它现在的新主页在SourceForge [http://sourceforge.net/projects/mod-auth-sspi/]。

- 下载此匹配你的 Apache 版本的模块，将文件mod_auth_sspi.so复制到 Apache 的 modules 目录。
- 编辑 Apache 的配置文件: 增加一行

  LoadModule sspi_auth_module modules/mod_auth_sspi.so

  到 LoadModule 节。确认你在下行之前插入此行

  LoadModule auth_module modules/mod_auth.so
- 为了让 Subversion 领域使用此认证类型，你需要将

  AuthType Basic

  修改为

  AuthType SSPI

  并且在 <Location /svn> 中增加

```
SSPIAuth On
SSPIAuthoritative On
SSPIDomain <domaincontroller>
SSPIOmitDomain on
SSPIUsernameCase lower
SSPIPerRequestAuth on
```

```
SSPIOfferBasic On
```

如果你没有域控制器，可以将域控制器的名称置为 <domaincontroller> 。

请注意，当你使用 SSPI 认证时，没有必要再使用 AuthUserFile 行定义密码文件，Apache 使用 Windows 域验证你的用户名和密码，你需要更新 svnaccessfile 中的用户列表来引用 DOMAIN\username。

只有使用 SSL 加密连接(https)时才可以启用 SSPI 认证。如果你只是用普通 http 协议连接到服务器，那么它不会工作。

要使你的服务器启用 SSL，请看：第 3.7.9 节 “用 SSL 使服务器更安全”

Subversion 的 AuthzSVNAccessFile 文件对用户名大小写敏感(JUser 与 juser 不同)。

在微软的世界，Windows 域和用户名不是大小写敏感。即使如此，一些网络管理员还是喜欢创建首字母大写的用户帐号(例如 JUser)。

使用SSPI的一个问题是用户名和密码是用户在提示输入时发送到Subversion的，而IE经常会不管你的帐户是如何建立的都会自动发送你的用户名。

结果就是你必须为每个用户在 AuthzSVNAccessFile 中至少创建两个条目：一个小写的条目和一个与 IE 传递给 Apache 一样的条目，你也需要训练你的用户在通过 TortoiseSVN 输入访问版本库的凭证时使用小写字母。

Apache 的错误和访问日志是你最好的朋友，例如帮助你检测传递给 Subversion 的 AuthzSVNAccessFile 模块的用户名，你或许需要试验 svnaccessfile 中用户字符串的精确格式(例如 DOMAIN\user 还是 DOMAIN//user)来使一切工作正常。

3.7.8. □多重认证源

也可以为Subversion使用不止一个的认证源，为此，你需要将每一种认证设置为non-authoritative，这样Apache会在多个源检查用户名/密码。

一个常见的场景就是同时使用Windows域和passwd文件认证，这样你可以为没有Windows域帐户的用户提供访问SVN的权限。

- To enable both Windows domain and passwd file authentication, add the following entries within the <Location> block of your Apache config file:

```
AuthBasicAuthoritative Off
SSPIAuthoritative Off
```

Here is an example of the full Apache configuration for combined Windows domain and passwd file authentication:

```
<Location /svn>
  DAV svn
  SVNListParentPath on
  SVNParentPath D:\SVN

  AuthName "Subversion repositories"
  AuthzSVNAccessFile svnaccessfile.txt

# NT Domain Logins.
```

```
  AuthType SSPI
  SSPIAuth On
  SSPIAuthoritative Off
  SSPIDomain <domaincontroller>
  SSPIOfferBasic On

# Htpasswd Logins.
  AuthType Basic
  AuthBasicAuthoritative Off
  AuthUserFile passwd

  Require valid-user
</Location>
```

3.7.9. □用 SSL 使服务器更安全

虽然 Apache 2.2.x 支持 OpenSSL，它默认却没有激活。你可以手动激活它。

1. 在 apache 配置文件中，取消这些行的注释:

   ```
   #LoadModule ssl_module modules/mod_ssl.so
   ```

   和最后面的

   ```
   #Include conf/extra/httpd-ssl.conf
   ```

   接着将这一行(在配置文件中这是一行)

   ```
   SSLMutex "file:C:/Program Files/Apache Software Foundation/\
   Apache2.2/logs/ssl_mutex"
   ```

   改为

   ```
   SSLMutex default
   ```

2. Next you need to create an SSL certificate. To do that open a command prompt (DOS-Box) and change to the Apache folder (e.g. C:\program files\apache group\apache2) and type the following command:

   ```
   bin\openssl req -config conf\openssl.cnf -new -out my-server.csr
   ```

   You will be asked for a passphrase. Please don't use simple words but whole sentences, e.g. a part of a poem. The longer the phrase the better. Also you have to enter the URL of your server. All other questions are optional but we recommend you fill those in too.

   Normally the privkey.pem file is created automatically, but if it isn't you need to type this command to generate it:

   ```
   bin\openssl genrsa -out conf\privkey.pem 2048
   ```

   Next type the commands

```
bin\openssl rsa -in conf\privkey.pem -out conf\server.key
```

and (on one line)

```
bin\openssl req -new -key conf\server.key -out conf\server.csr \
-config conf\openssl.cnf
```

and then (on one line)

```
bin\openssl x509 -in conf\server.csr -out conf\server.crt
                 -req -signkey conf\server.key -days 4000
```

This will create a certificate which will expire in 4000 days. And finally enter (on one line):

```
bin\openssl x509 -in conf\server.cert -out conf\server.der.crt
                 -outform DER
```

These commands created some files in the Apache conf folder (server.der.crt, server.csr, server.key, .rnd, privkey.pem, server.cert).

3. 重启 apache 服务。
4. 将你的浏览器指向https://servername/svn/project ...

SSL 和 Internet Explorer

如果你使用SSL保护你的服务器，并使用windows域来进行认证，你会发现不能使用IE浏览版本库了，不需要担心－那只是因为IE没有经过认证，其他浏览器没有这个问题，TortoiseSVN和其他Subversion客户端仍然可以得到认证。

如果你一直希望使用IE浏览你的版本库，你可以选择:

- 在 Apache 的配置文件定义一个单独的 <Location /path> 指示，增加 SSPIBasicPreferred On。这将使 IE 能够认证，但是其他浏览器和 Subversion 不能对这个领域认证。
- 也提供未加密(没有SSL)认证的浏览，奇怪的IE在没有使用SSL的认证时没有任何问题。
- 在 SSL 的“标准”配置中，通常在 apache 的虚拟 SSL 主机内有下面的内容:

```
SetEnvIf User-Agent ".*MSIE.*" \
             nokeepalive ssl-unclean-shutdown \
             downgrade-1.0 force-response-1.0
```

这种设置的充足理由参见 http://www.modssl.org/docs/2.8/ssl_faq.html#ToC49。但是如果你希望使用 NTLM 认证，就必须使用 keepalive。如果启用全部 SetEnvIf，你就可以使 IE 用 Windows 认证访问运行在 Win32 上加载了 mod_auth_sspi 模块的 Apache。

强制 SSL 访问

当你设置了 SSL 让你的版本库更安全，你一定希望关闭普通的非 SSL (http) 访问。为此，你需要在 Subversion 的 <Location> 增加指示: SSLRequireSSL。

```
An example <Location> block would look like this:


<Location /svn>
  DAV svn
  SVNParentPath D:\SVN
  SSLRequireSSL
  AuthType Basic
  AuthName "Subversion repositories"
  AuthUserFile passwd
  #AuthzSVNAccessFile svnaccessfile
  Require valid-user
</Location>
```

3.7.10. □在虚拟 SSL 主机中使用客户端证书

由 Nigel Green 发送到 TortoiseSVN 邮件列表。非常感谢!

在某些情况下你需由要 1 台服务器掌管 2 个虚拟的 SSL 主机: 第 1 个用于公共 web 浏览，不需要客户端证书。第 2 个必需通过客户端证书来保证安全，运行 Subversion 服务器。

在 Apache 配置文件的 per-server 部分中添加 SSLVerifyClient Optional 指令(例如，在任何 VirtualHost 和 Directory 块之外)强制 Apache 在开始 SSL 连接握手时索取客户端证书。由于 mod_ssl 中的一个错误，在这个时候索取客户端证书是非常有必要的，因为当 SSL 连接重新协商时，它就不会生效了。

解决方案就是在 Subversion 虚拟主机的目录配置部分添加下面的指令:

```
SSLRequire %{SSL_CLIENT_VERIFY} eq "SUCCESS"
```

这个指令确认只有在收到客户端证书并成功验证后才能有权读取目录。

总之，Apache 配置文件中的相关内容就是这样的:

```
SSLVerifyClient Optional

### 公共访问虚拟主机
### (不需要客户端证书)

<VirtualHost 127.0.0.1:443>
  <Directory "pathtopublicfileroot">
  </Directory>
</VirtualHost>

### SUBVERSION 虚拟主机
### (需要客户端证书)
<VirtualHost 127.0.0.1:443>
  <Directory "subversion host root path">
    SSLRequire %{SSL_CLIENT_VERIFY} eq "SUCCESS"
  </Directory>

  <Location /svn>
    DAV svn
    SVNParentPath /pathtorepository
  </Location>
</VirtualHost>
```

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. □开始

4.1.1. □图标重载

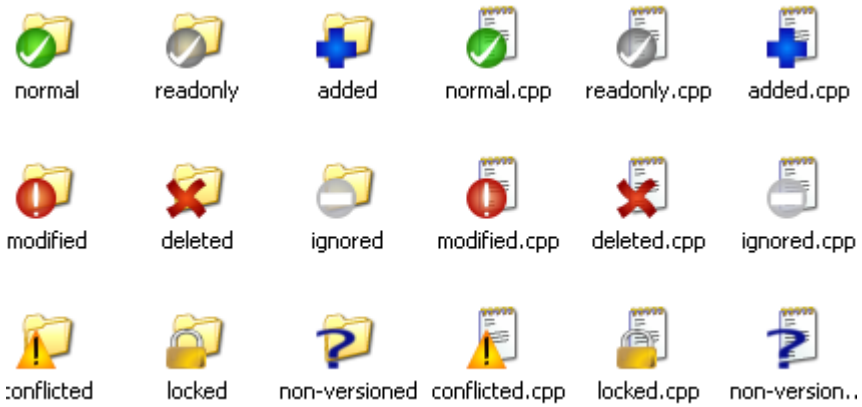


图 4.1. 显示重载图标的资源管理器

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

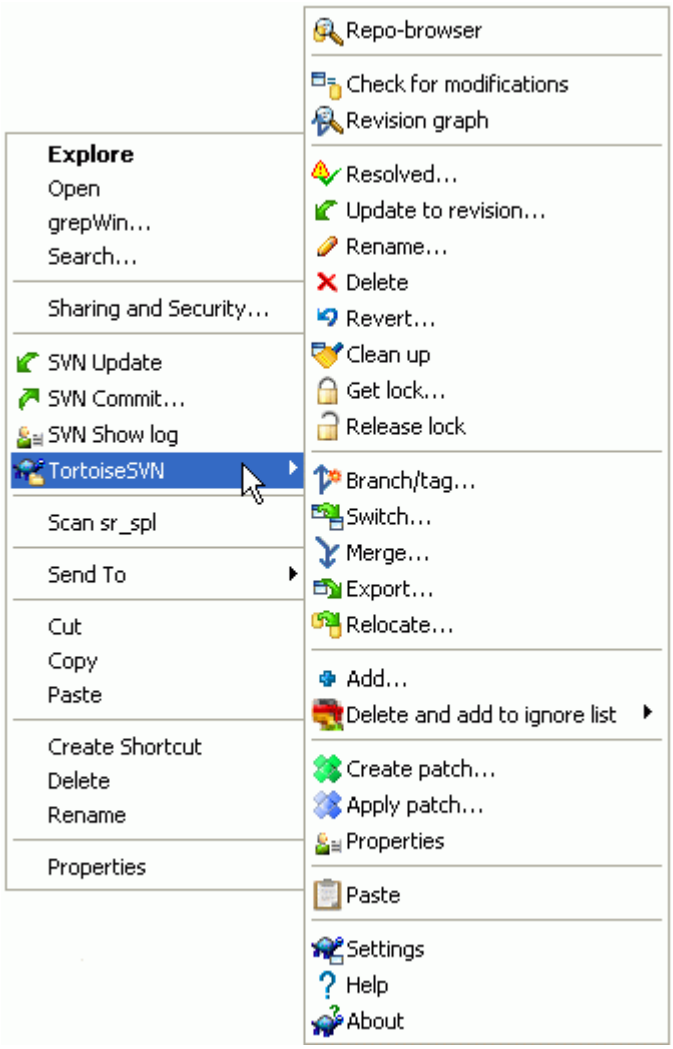


图 4.2. 版本控制下一个目录的右键菜单

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。

有些命令很少被用到，只有在扩展右键菜单中才能显示。要显示扩展右键菜单，请在单击右键时按住 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

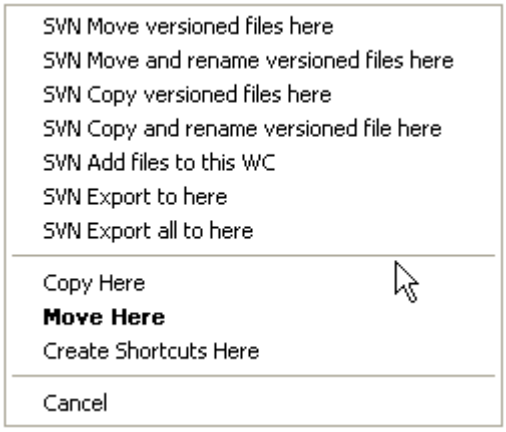


图 4.4. 版本控制下的一个目录的右键拖拽菜单

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。

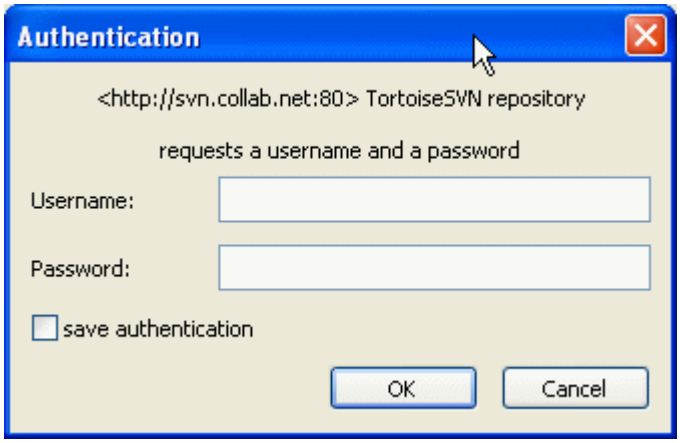


图 4.5. 认证对话框

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)。
- 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”

有些人喜欢在注销 Windows 时或关机时删除认证数据。使用关机脚本删除目录 %APPDATA%\Subversion\auth 可以达到此目的，例如:

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

你可以从 windows-help-central.com [http://www.windows-help-central.com/windows-shutdown-script.html] 找到如何安装此脚本的详细描述。

关于如何设置服务器的认证和权限的更多信息，请参考第 3.5 节 “Accessing the Repository”

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

如果你要将数据导入到已经包括一些项目的版本库中，那么版本库的结构应该已经确定了。如果你要将数据导入到一个新的版本库中，那么应该花点时间确定版本库的组织结构。阅读 第 3.1.5 节 “版本库布局”以获得更多的建议。

这一节介绍 Subversion 的导入(import)命令，它被设计成只需要 1 步就可以将目录结构导入到版本库中。尽管它可以使用，但是它有一些短处:

- 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。

因此，我们推荐你根本不要使用导入命令，而是使用这里介绍的需要 2 步的操作方法: 第 4.2.2 节 “导入适当的位置”既然你已经阅读到这里了，下面就介绍基础的导入操作...

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：

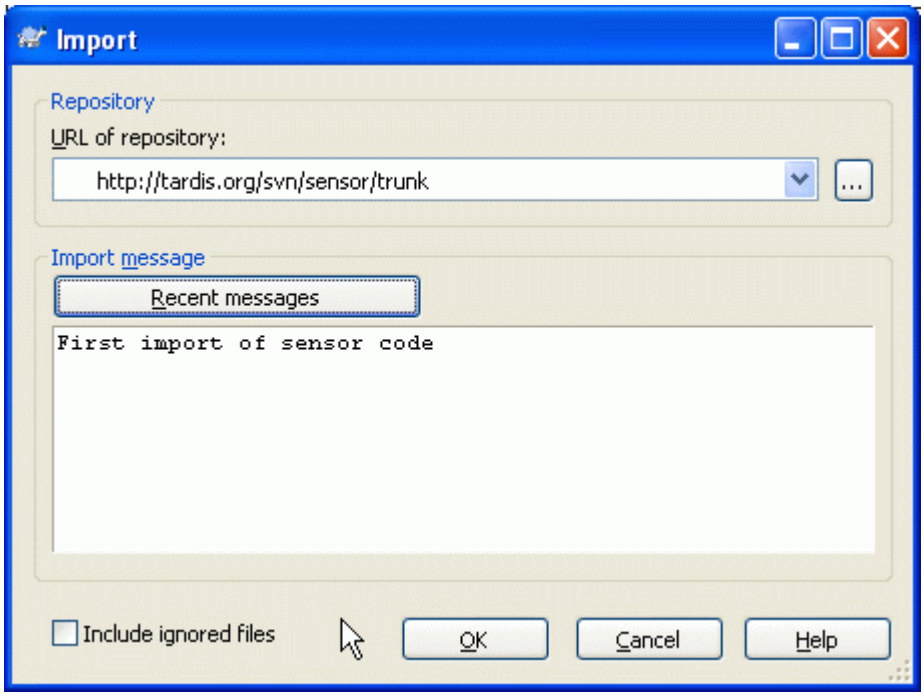


图 4.6. 导入对话框

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. 使用版本库浏览器直接在版本库中创建项目文件夹。
2. 在你要导入的文件夹检出新目录。你会得到一个本地目录为空的警告。现在你有一个版本控制的顶级目录，含有未版本控制的内容。

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：

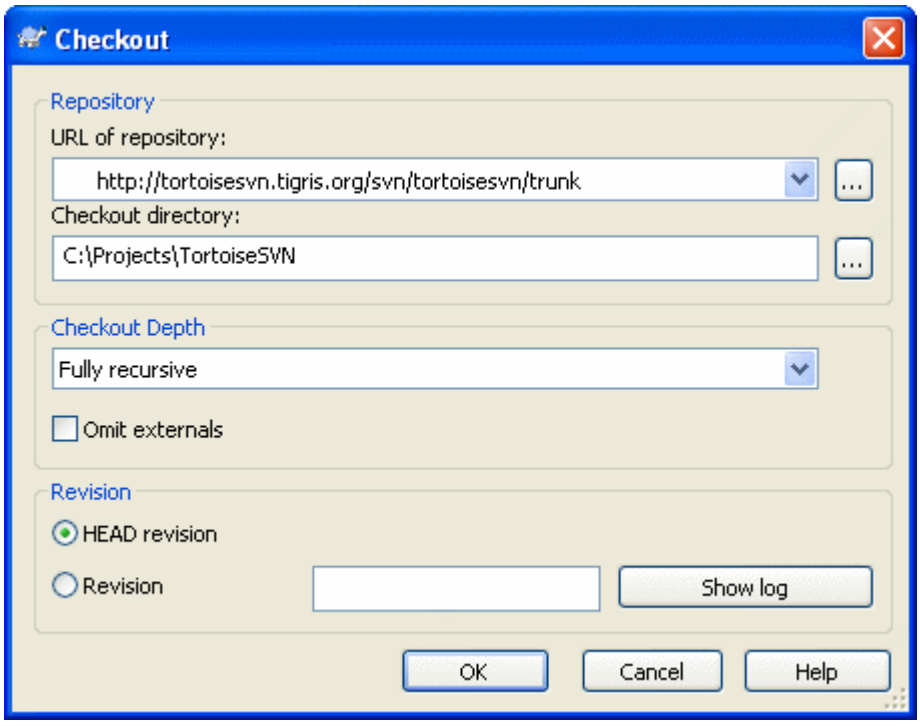


图 4.7. 检出对话框

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。

直接节点，包含目录。
: 检出目录，包含其中的文件或子目录，但是不递归展开子目录。

文件子节电
: 检出指定目录，包含所有文件，但是不检出任何子目录。

仅此项。
: 只检出目录。不包含其中的文件或子目录。

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。

如果你检出了一个稀疏的工作副本(例如，在签出时选择的签出深度不是全递归)，你可以使用版本库浏览器(第 4.24 节 “版本库浏览器”)或检查修改对话框(第 4.7.3 节 “本地与远程状态”)来获得其它子文件夹。

In the repository 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... That menu will only be visible if the selected item does not exist yet in your working copy, but the parent item does exist.

在检查修改对话框中，首先点击检查版本库按钮。对话框会将你未检出，但是位于版本库中的文件和文件夹显示为远程加入。右键单击你需要加入工作副本的文件夹，选择右键菜单 → 更新。

This feature is very useful when you only want to checkout parts of a large tree, but you want the convenience of updating a single working copy. Suppose you have a large tree which has sub-folders Project01 to Project99, and you only want to checkout Project03, Project25 and Project76/SubProj. Use these steps:

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. Select the new folder and use TortoiseSVN → Repo browser to display the repository content.

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

一旦你检出一个特定深度的工作副本，你可以使用右键菜单 → 更新项目至版本...更改深度获得更多和更少的内容。

使用旧版本服务器

1.5 版之前的服务器不支持设置工作副本深度的请求，所以它们不能有效的处理请求。不过该命令仍然可以工作，但是旧版的服务器会发送全部数据，由客户端过滤掉不需要的内容，这意味着会产生很多的网络数据流量。如果可能，应该升级服务器到 1.5 版

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。

If Omit externals is checked, or if you wish to increase the depth value, you will have to perform updates to your working copy using TortoiseSVN → Update to Revision... instead of TortoiseSVN → Update. The standard update will include all externals and keep the existing depth.

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....

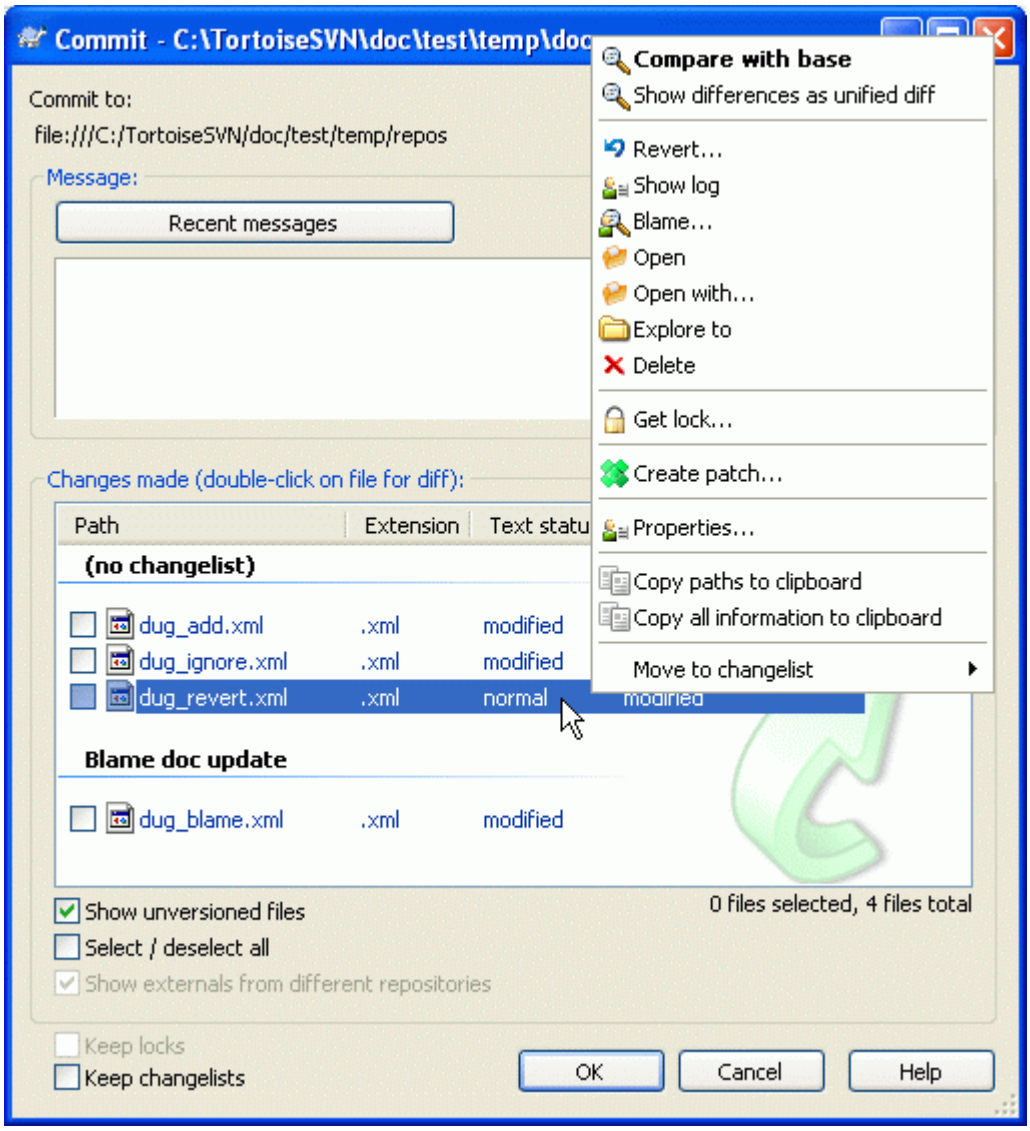


图 4.8. 提交对话框

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。

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。

如果你修改的文件是使用了svn:externals从别的版本库中包含进来的，那么这些改动不会被自动提交。在文件列表下方的警告符号会告诉你是否出现了这种状况，工具提示(tooltip)提示了外部文件必须要分开提交。

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。

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。

You can select or deselect items by clicking on the checkbox to the left of the item. For directories you can use Shift-select to make the action recursive.

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。

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。

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. □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。

4.4.4. □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。

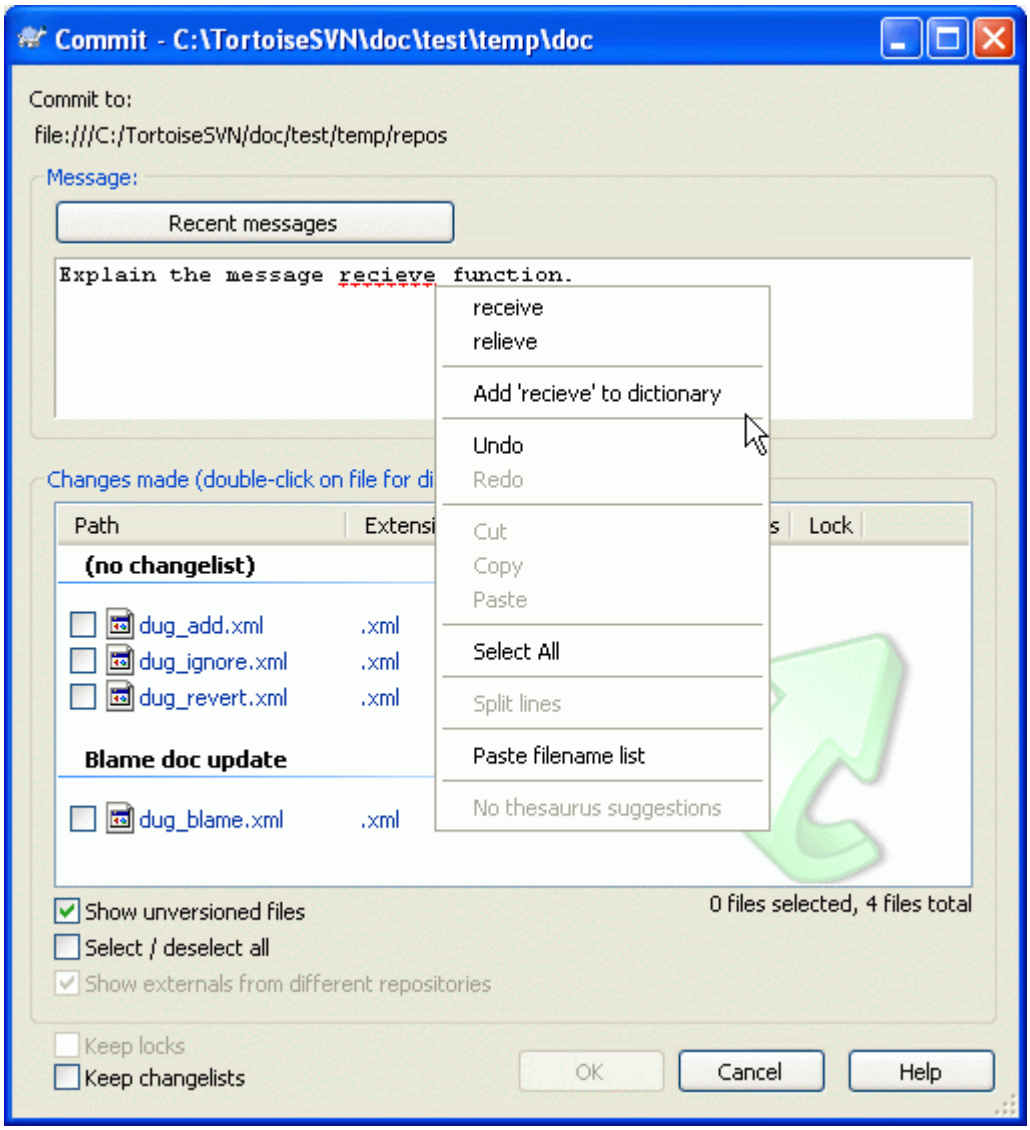


图 4.9. 提交对话框的拼写检查器

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。

The log message window also includes a filename and function auto-completion facility. This uses regular expressions to extract class and function names from the (text)

files you are committing, as well as the filenames themselves. If a word you are typing matches anything in the list (after you have typed at least 3 characters, or pressed Ctrl+Space), a drop-down appears allowing you to select the full name. The regular expressions supplied with TortoiseSVN are held in the TortoiseSVN installation bin folder. You can also define your own regexes and store them in %APPDATA%\TortoiseSVN\autolist.txt. Of course your private autolist will not be overwritten when you update your installation of TortoiseSVN. If you are unfamiliar with regular expressions, take a look at the introduction at http://en.wikipedia.org/wiki/Regular_expression, and the online documentation and tutorial at http://www.regular-expressions.info/.

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。

If you want to include the checked paths in your log message, you can use the command Context Menu → Paste filename list in the edit control.

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

If you have activated the bug tracking system, you can set one or more Issues in the Bug-ID / Issue-Nr: text box. Multiple issues should be comma separated. Alternatively, if you are using regex-based bug tracking support, just add your issue references as part of the log message. Learn more in 第 4.28 节 “与 BUG 跟踪系统/问题跟踪集成”.

4.4.5. □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。

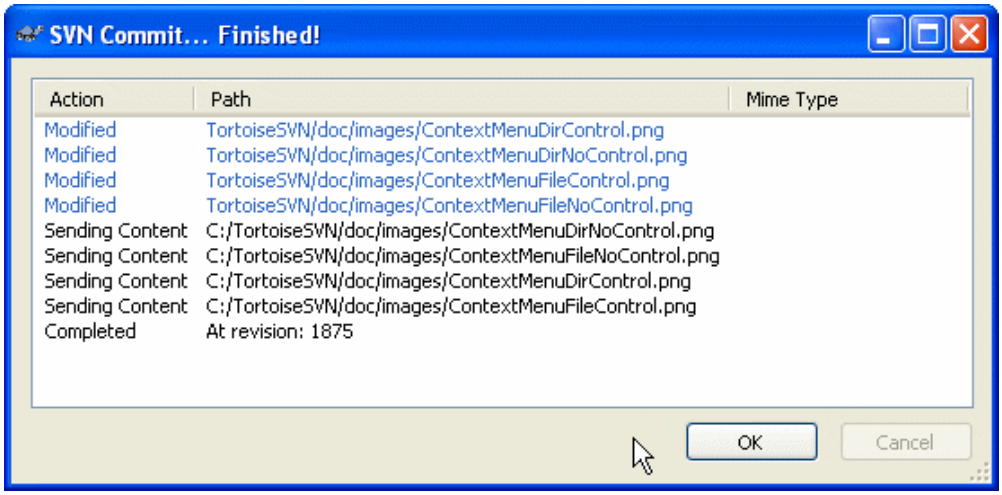


图 4.10. 显示提交进度的进度对话框

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　　4.30.1.4　　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

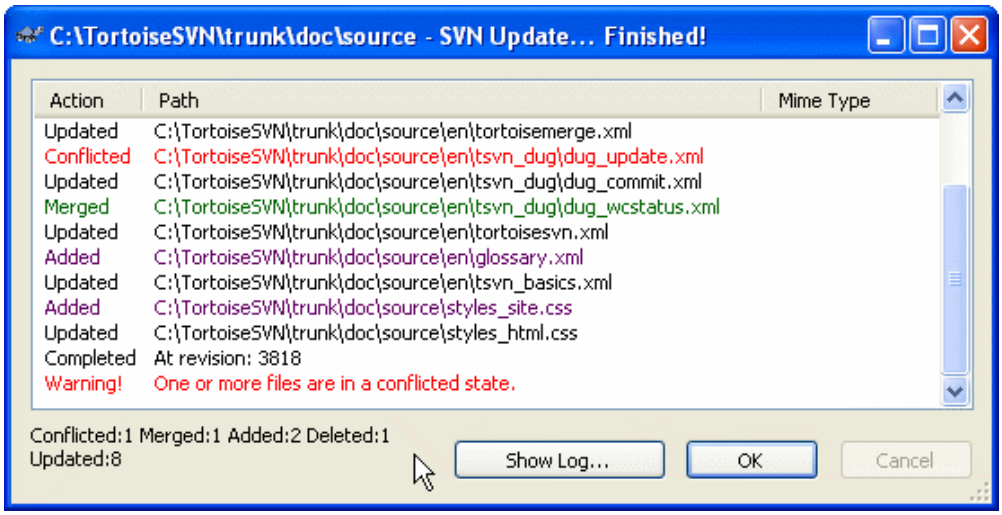


图　4.11.　已经完成更新的进度对话框

Periodically, you should ensure that changes done by others get incorporated in your local working copy. The process of getting changes from the server to your local copy is known as updating. Updating may be done on single files, a set of selected files, or recursively on entire directory hierarchies. To update, select the files and/or directories you want, right click and select TortoiseSVN → Update in the explorer context menu. A window will pop up displaying the progress of the update as it runs. Changes done by others will be merged into your files, keeping any changes you may have done to the same files. The repository is not affected by an update.

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WC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 4.30.1.4 节 “TortoiseSVN 颜色设置”获得详情。

If you get any conflicts during an update (this can happen if others changed the same lines in the same file as you did and those changes don't match) then the dialog shows those conflicts in red. You can double click on these lines to start the external merge tool to resolve the conflicts.

When the update is complete, the progress dialog shows a summary of the number of items updated, added, removed, conflicted, etc. below the file list. This summary information can be copied to the clipboard using Ctrl+C.

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

If you update a file or folder to a specific revision, you should not make changes to those files. You will get “out of date” error messages when you try to commit them! If you want to undo changes to a file and start afresh from an earlier revision, you can rollback to a previous revision from the revision log dialog. Take a look at 第 B.4 节 “Roll back (Undo) revisions in the repository” for further instructions, and alternative methods.

Update to Revision can occasionally be useful to see what your project looked like at some earlier point in its history. But in general, updating individual files to an earlier revision is not a good idea as it leaves your working copy in an inconsistent state. If the file you are updating has changed name, you may even find that the file just disappears from your working copy because no file of that name existed in the earlier revision. You should also note that the item will show a normal green overlay, so it is indistinguishable from files which are up-to-date.

If you simply want a local copy of an old version of a file it is better to use the Context Menu → Save revision to... command from the log dialog for that file.

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。

本地文件已经存在

有时在你试图更新的时候，更新失败，提示信息说已经有一个同名的本地文件。通常发生在Subversion试图检出一个新增的受控文件时，发现一个未受控的同名文件

> 已经在工作路径中存在。Subversion绝不会覆盖一个未受控的文件——因为它有可能有你需要的东西，却碰巧与另一个开发者新提交的文件重名了。
>
> 如果你得到这个错误信息，解决的方法就是把本地的未受控文件重命名。在完成更新之后，你再检查被重命名的文件是不是还需要。
>
> 如果你一直得到错误，使用TortoiseSVN → 检查修改来列出所有有问题的文件。这样你可以一次性解决它们。

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

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。由于 Subversion 不知道你的项目的具体情况，它把解决冲突的工作留给了开发人员。一旦出现冲突，你就应该打开有问题的文件，查找以字符串<<<<<<<开头的行。有冲突的区域用如下的方式标记:

```
<<<<<<< 文件名
  你的修改
=======
  合并自版本库中的代码
>>>>>>> 版本
```

对于每个冲突的文件 Subversion 在你的目录下放置了三个文件:

文件名.ext.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.ext.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.ext.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。

你可以通过TortoiseSVN → 编辑冲突运行外部合并工具/冲突编辑器，或者你可以使用任何别的编辑器手动解决冲突。你需要冲定哪些代码是需要的，做一些必要的修改然后保存。

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filename.ext.mine和filename.ext.r*两个文件，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filename.ext.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. □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。

4.6.2.1. □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。

4.6.2.2. □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。

4.6.2.3. □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 将文件 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。

4.6.2.4. □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。

4.6.2.5. □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。

4.6.2.6. □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 工作在分之上，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。

4.7. □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

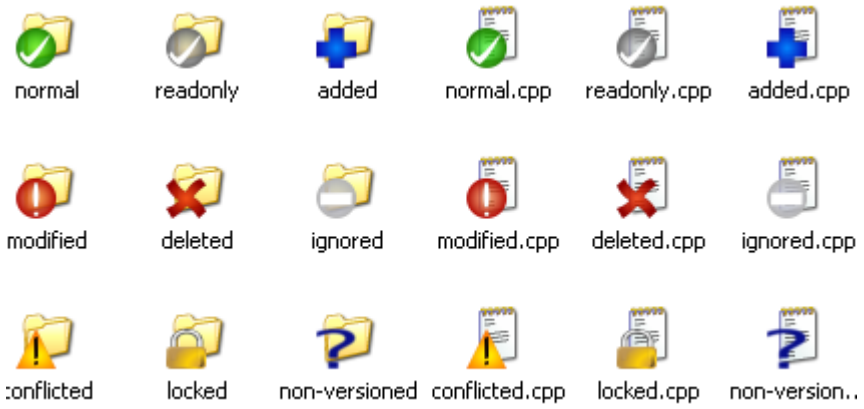


图 4.12. 显示重载图标的资源管理器

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。

现在有很多 Tortoise 客户端工具(TortoiseCVS, TortoiseHG, ...)所以图标限制成为一个实实在在的问题。为了在这种条件下工作，TortoiseSVN 项目引入了一种通用的共享的图标集，以动态链接库的形式加载，可以被所有的 Tortoise 客户端工具所使用。联系你的客户端程序提供者确认是否支持这一功能。

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. □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。

4.7.3. □本地与远程状态

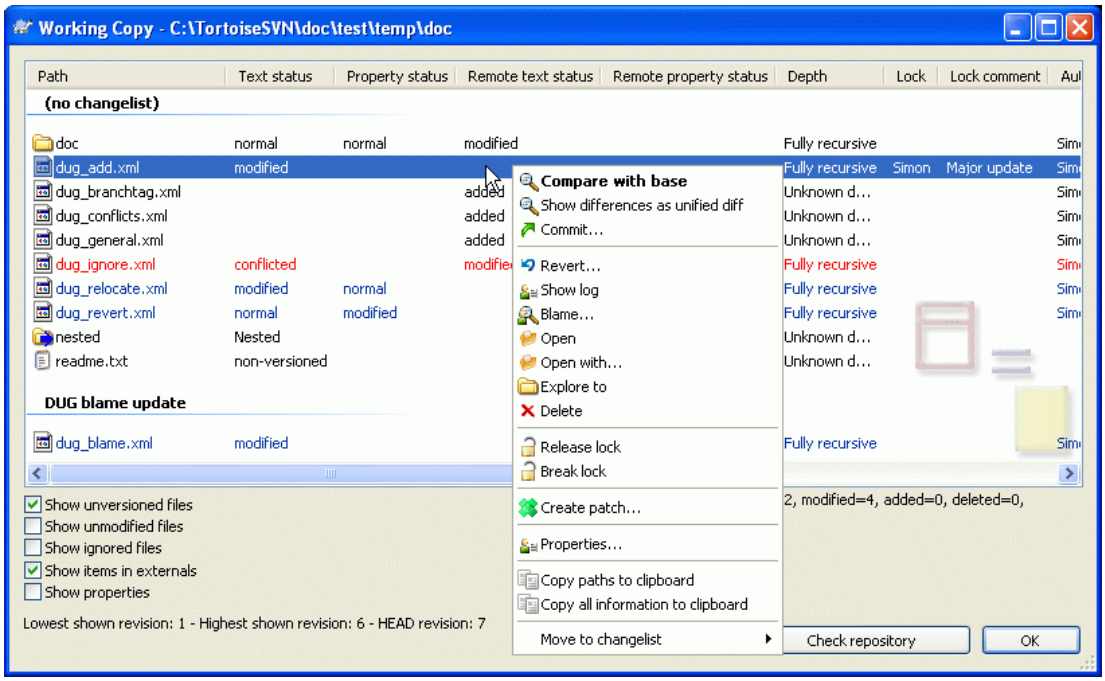


图 4.13. 检查修改

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 → Check For Modifications... 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。

If you click on the Check Repository then you can also look for changes in the repository. That way you can check before an update if there's a possible conflict. You can also update selected files from the repository without updating the whole folder. By default, the Check Repository button only fetches the remote status with the checkout depth of the working copy. If you want to see all files and folders in the repository, even those you have not checked out, then you have to hold down the Shift key when you click on the Check Repository button.

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 shows)显示了该项是从哪里复制来的。 where the item was copied from.

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 4.30.1.4 节 “TortoiseSVN 颜色设置”获得详情。

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！

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。

You can also revert changes in individual files. If you have deleted a file accidentally, it will show up as Missing and you can use Revert to recover it.

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。

如果你要查询一个文件的详细情况，你可以把它从这里拖到另一个应用程序，比如一个文本编辑器或是IDE中。

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。

At the bottom of the dialog you can see a summary of the range of repository revisions in use in your working copy. These are the commit revisions, not the update revisions; they represent the range of revisions where these files were last committed, not the revisions to which they have been updated. Note that the revision range shown applies only to the items displayed, not to the entire working copy. If you want to see that information for the whole working copy you must check the Show unmodified files checkbox.

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。

4.7.4. □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。

TortoiseSVN 中的修改列表依赖一个不存在于 Windows 2000 的外壳特性，所以它只能用于 Windows XP 或更新的系统。抱歉，但是现在 Windows 2000 真的很古老了，请不要抱怨。

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。

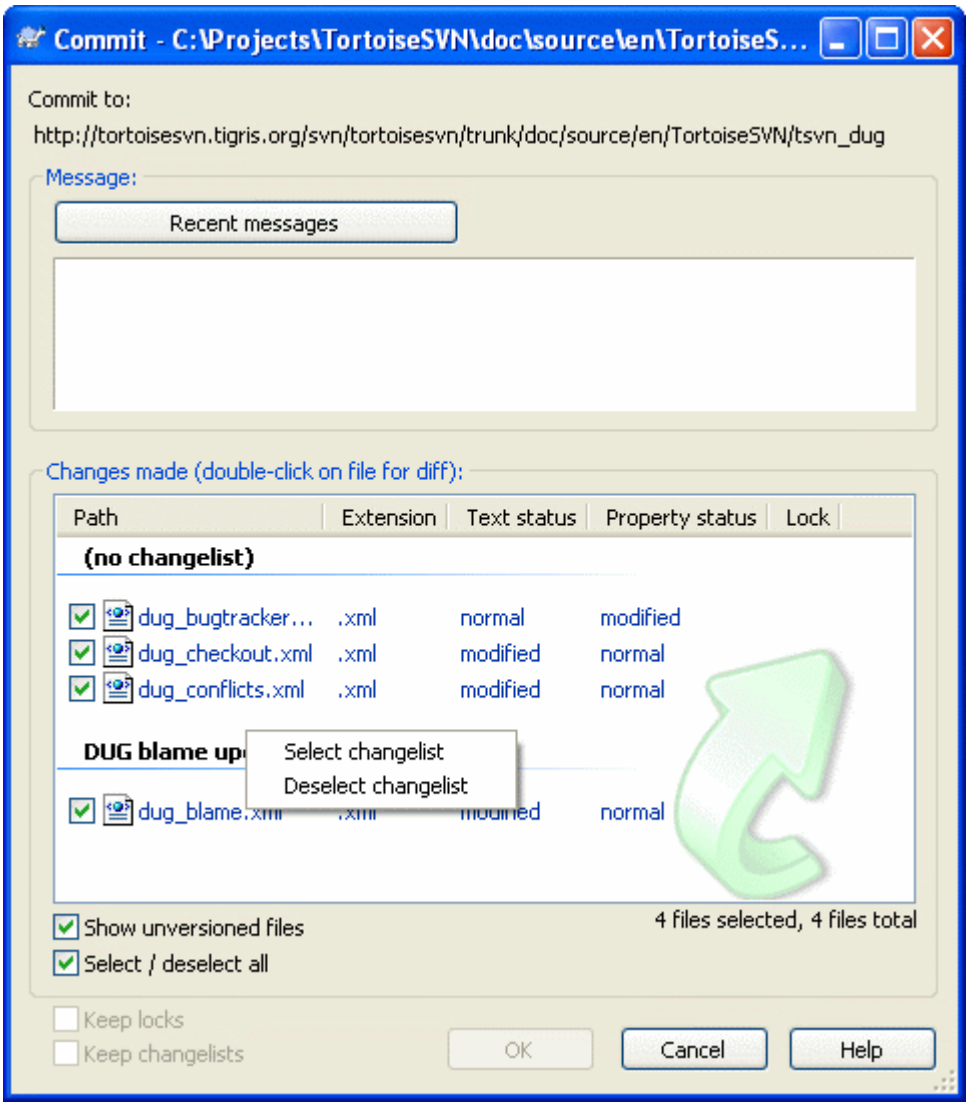


图 4.14. 带有修改列表的提交对话框

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 4.4.3 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。

Changelists are purely a local client feature. Creating and removing changelists will not affect the repository, nor anyone else's working copy. They are simply a convenient way for you to organise your files.

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

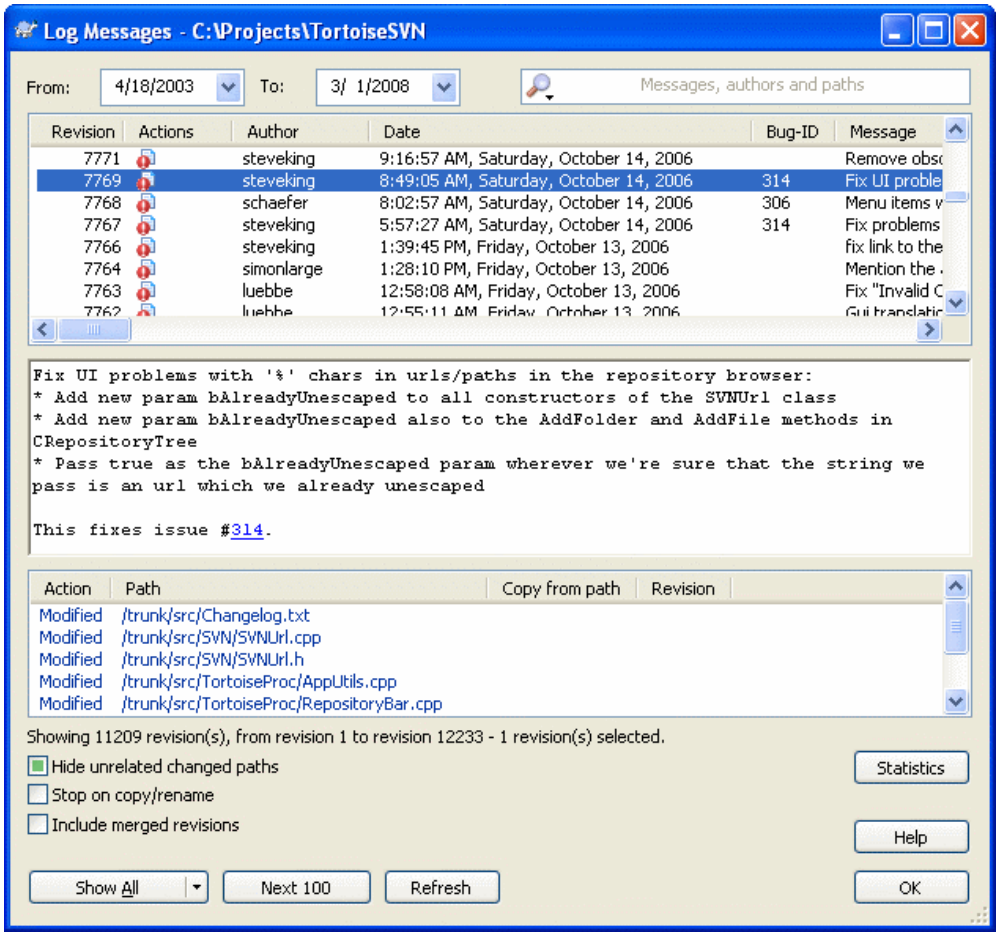


图 4.15. 版本日志对话框

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

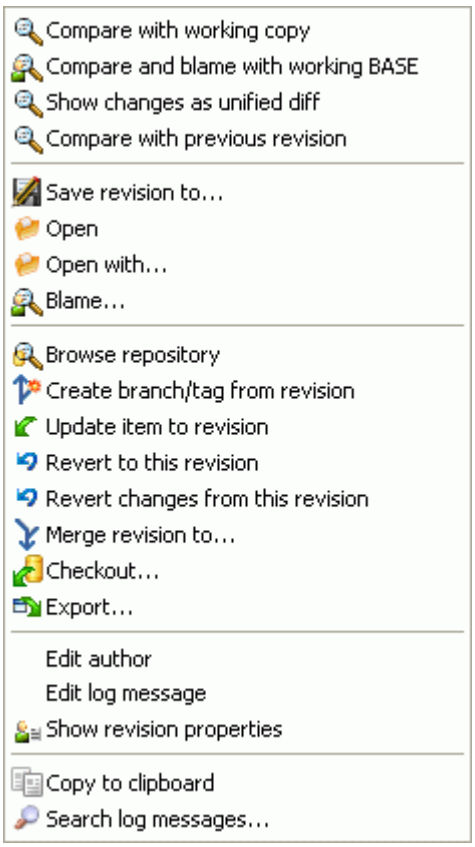


图 4.16. 版本日志对话框的顶部面板的右键菜单

日志对话框的顶部面板有一个右键菜单，通过此菜单你可以获得更多的信息。这些菜单条目中的有些内容只会在显示文件日志时出现，有些内容只会在显示文件夹日志时出现。

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
: 追溯选中的版本，和你的工作基础文件，使用可视化差异工具比较追溯报告。参阅 第 4.23.2 节 “追溯不同点”以获得更多信息(仅对于文件)。

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。

Compare with previous revision
: 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
: 显示已修改的文件对话框，让你选择文件。追溯选中的版本和旧版本，用可视化差异工具比较结果(仅对于文件夹)。

保存版本至...
: 将选中的版本保存到文件，你可以得到一份该文件的旧版本(仅对于文件)。

打开 / 打开方式...
: 用此文件类型的默认查看器，或你指定的程序打开选中的文件(仅对于文件)。

追溯...
: 追溯文件直到选中的版本(仅对于文件)。

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。

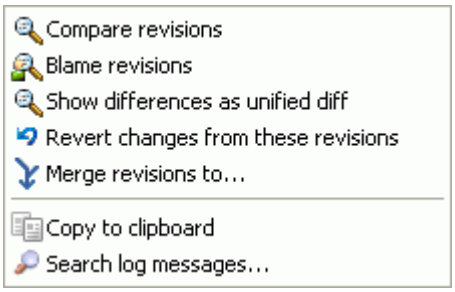


图 4.17. 选中两个版本的顶部面板的右键菜单

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。

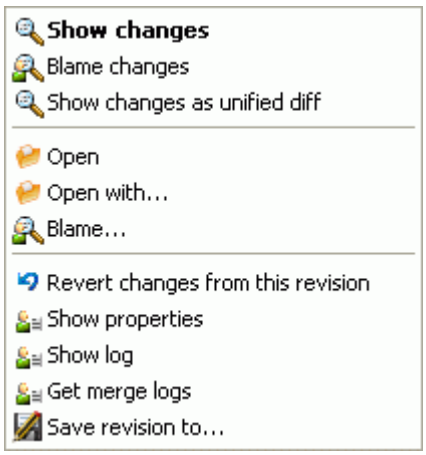


图　4.18.　日志对话框的底部面板的右键菜单

日志对话框的底部面板也有右键菜单，你可以:

显示改变
　　显示选中版本中的选中文件的改变。这个菜单条目只对显示为已修改的文件有效。

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。

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 使用版本号表达两个意思。版本号通常意味着版本库在那个时间的状态，也用来表达那个版本创建的结果集，例如 “r1234 所做” 意味着 r1234 版本提交的改变实现了特性 X。为了能更清晰的知道使用了哪个语义，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。

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。

When you show the log for a folder the revision highlighted is the highest revision found anywhere within that folder, which requires a crawl of the working copy. This can be a slow operation for large working copies, and the log messages are not displayed until the crawl completes. If you want to disable or limit this feature you need to set a registry key HKCU\Software\TortoiseSVN\RecursiveLogRev as described in 第 4.30.10 节 “注册表设置”.

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。

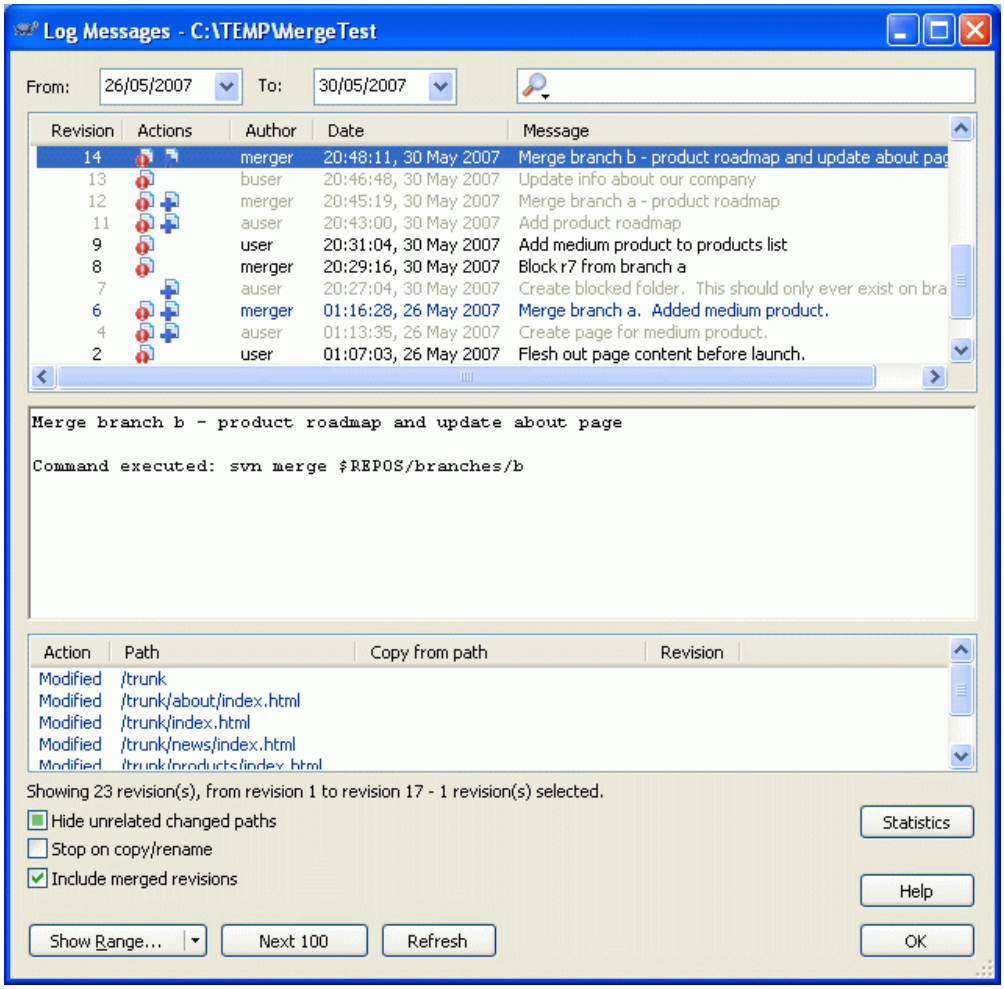


图 4.19. 日志对话框显示合并跟踪版本

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允许你在任何时候修改日志消息和作者。但这种改变不可还原(不在版本控制之下)，正因如此，这些功能默认是不可用的，如果要开启它，必须设置一个pre-revprop-change钩子。具体如何做，请参考《使用 Subversion 进行版本管理》的相关章节钩子脚本 [http://svnbook.red-bean.com/en/1.5/svn.reposadmin.create.html#svn.reposadmin.create.hooks]。在Windows机器上实现钩子的注意事项请阅读第 3.3 节 “服务器端钩子脚本”。

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。

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。

点击查找图标来选择你要查找信息的类型，并且选择正则表达式模式。一般情况下你只需要简单的文本查找，但是如果你需要更有弹性的查找规则，你就可以使用正则表达式。鼠标指针停留在文版框上面时，会显示如何使用正则表达式功能的的提示。你也可以在这里 http://www.regular-expressions.info/ 查看在线文档和指导。过滤器检查你的过滤字符串是否与日志条目匹配，并且只显示与过滤字符串匹配的条目。

要使过滤所有日志条目的结果中不包含过滤字符串，在字符串的前面加一个感叹号('!')。例如，过滤字符串 !username 将会显示那些不是由 username 提交的条目。

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。

你还可以使用隐藏无关的修改路径复选框来过滤底部面板中的路径名称。所谓相关路径，是指那些与日志相关的路径。对于一个文件夹的日志来说，相关路径就是该文件夹以其下的所有内容。对于一个文件的日志来说，相关路径就是与该文件的路径。该复选框是3态的: 可以显示所有的路径，将无关的内容灰色显示，或是完全隐藏无关路径。

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

因为格式化依赖于对 subversion 版本库的访问，所以只能在已签出的工作副本中看到结果。从远程获取属性是一个很慢的操作，所以你在从版本库浏览器中进行操作时不能看到这个特性。

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

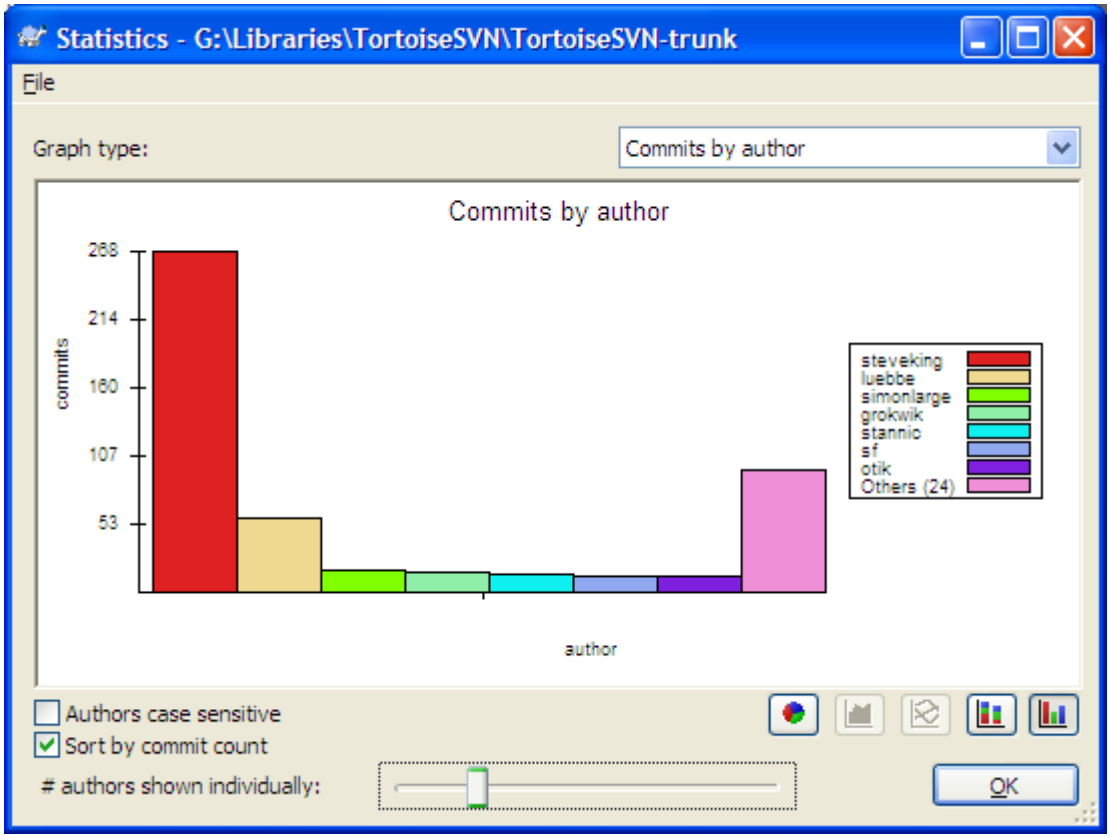


图 4.20. 作者提交次数统计柱状图

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。

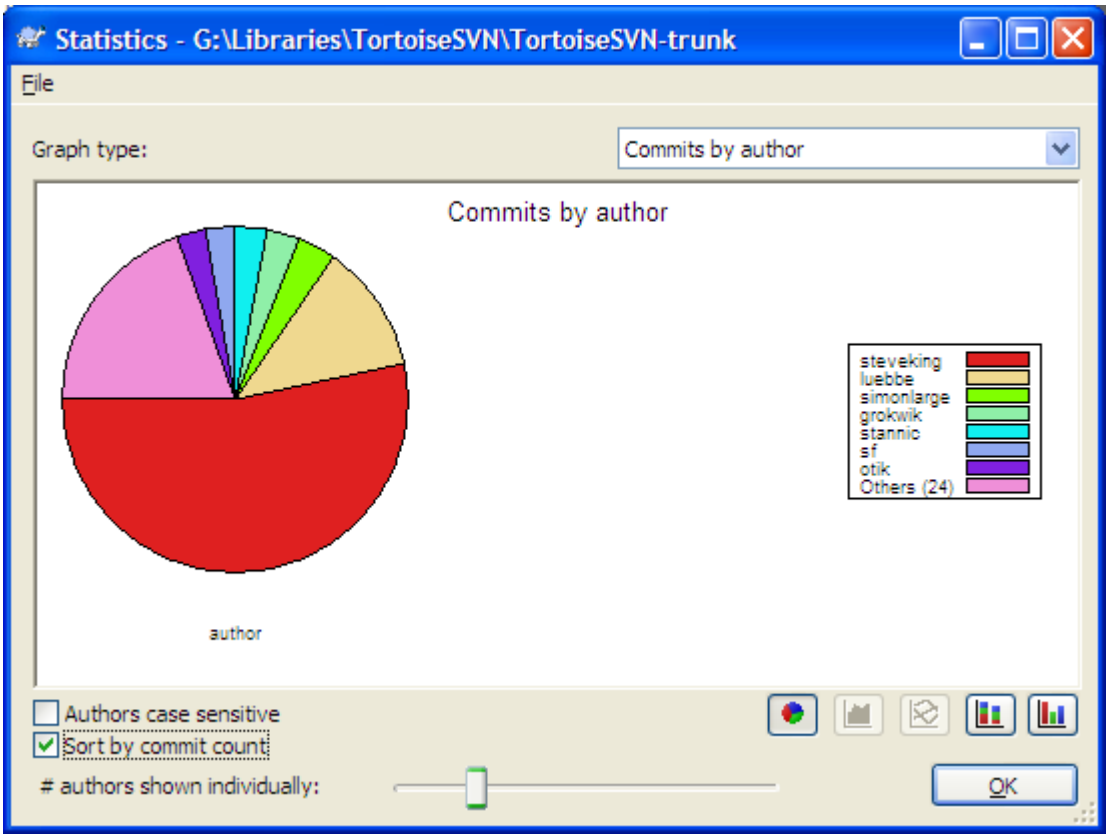


图 4.21. 作者提交次数统计饼图

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

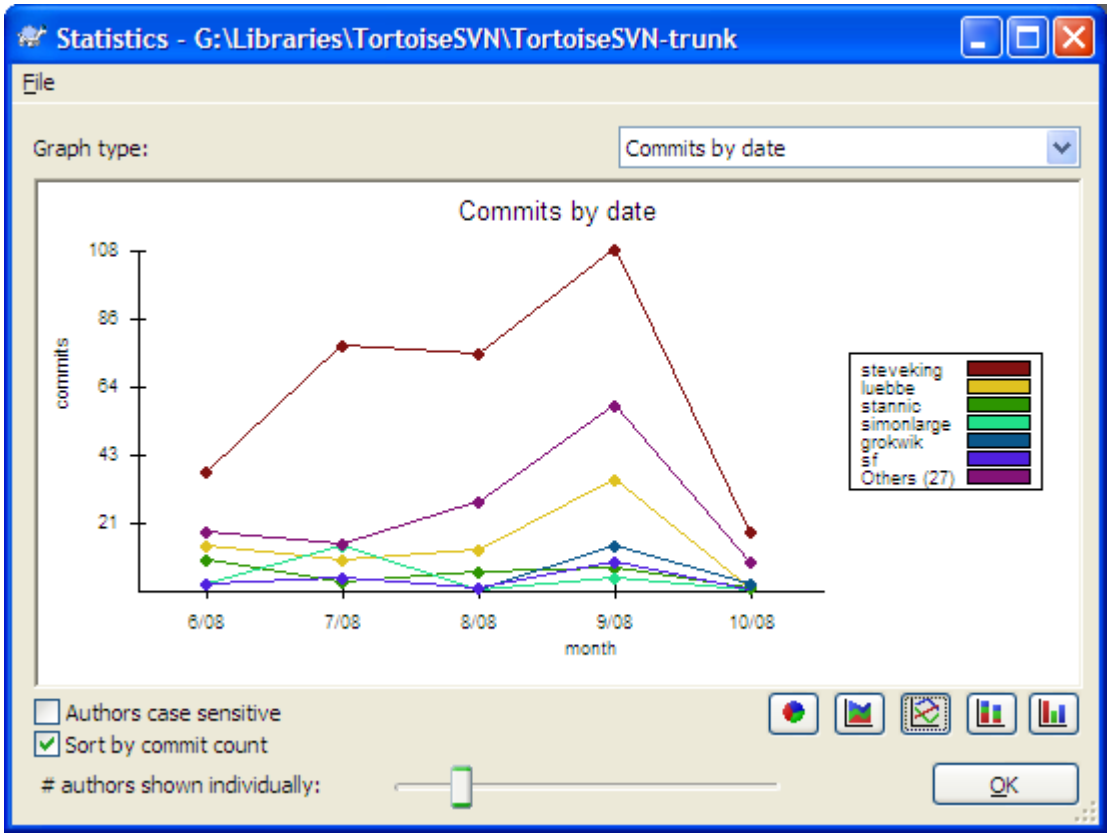


图 4.22. 按日期提交统计图

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

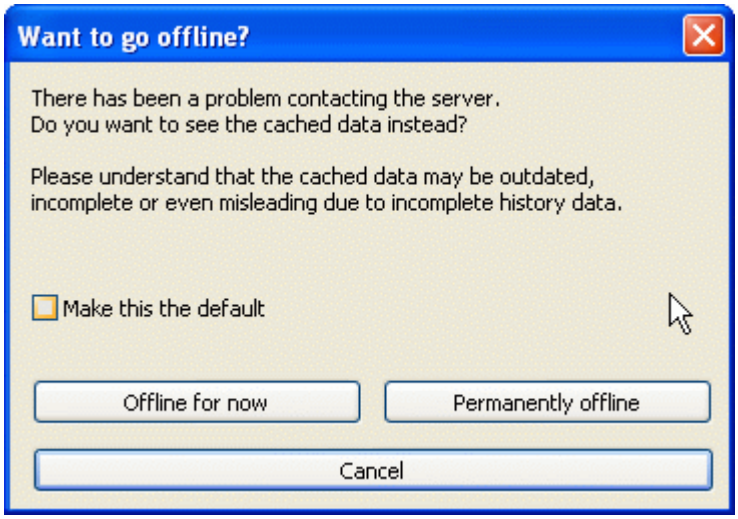


图 4.23. 要离线对话框

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 4.10.5 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

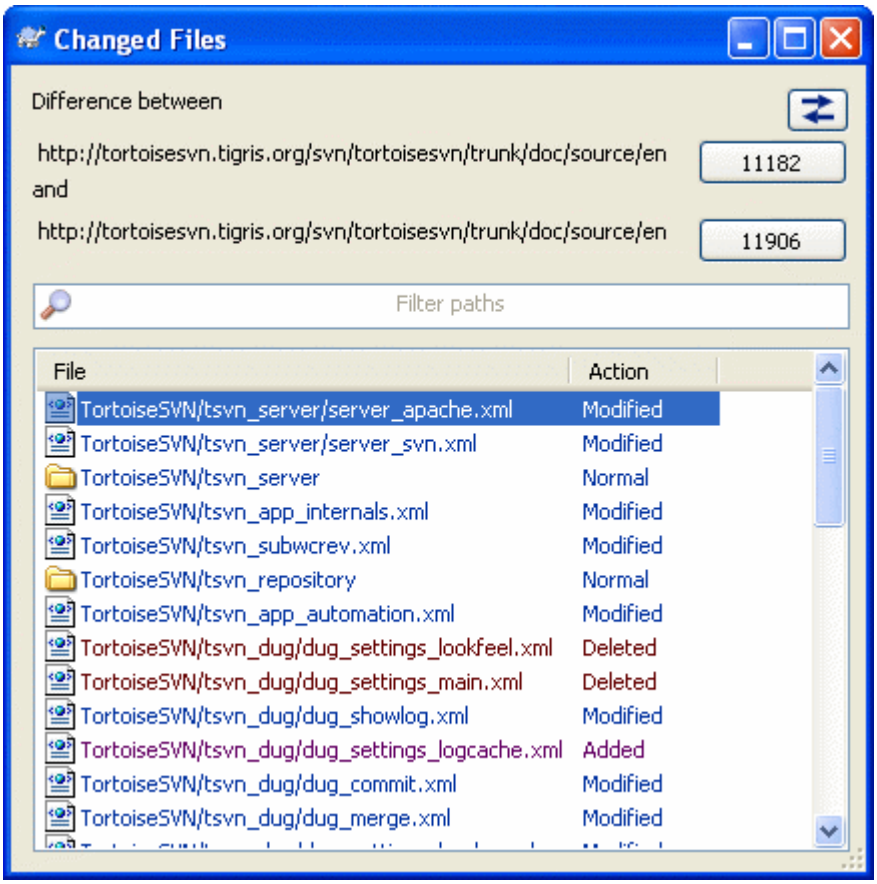


图 4.24. 比较修订版本对话框

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。

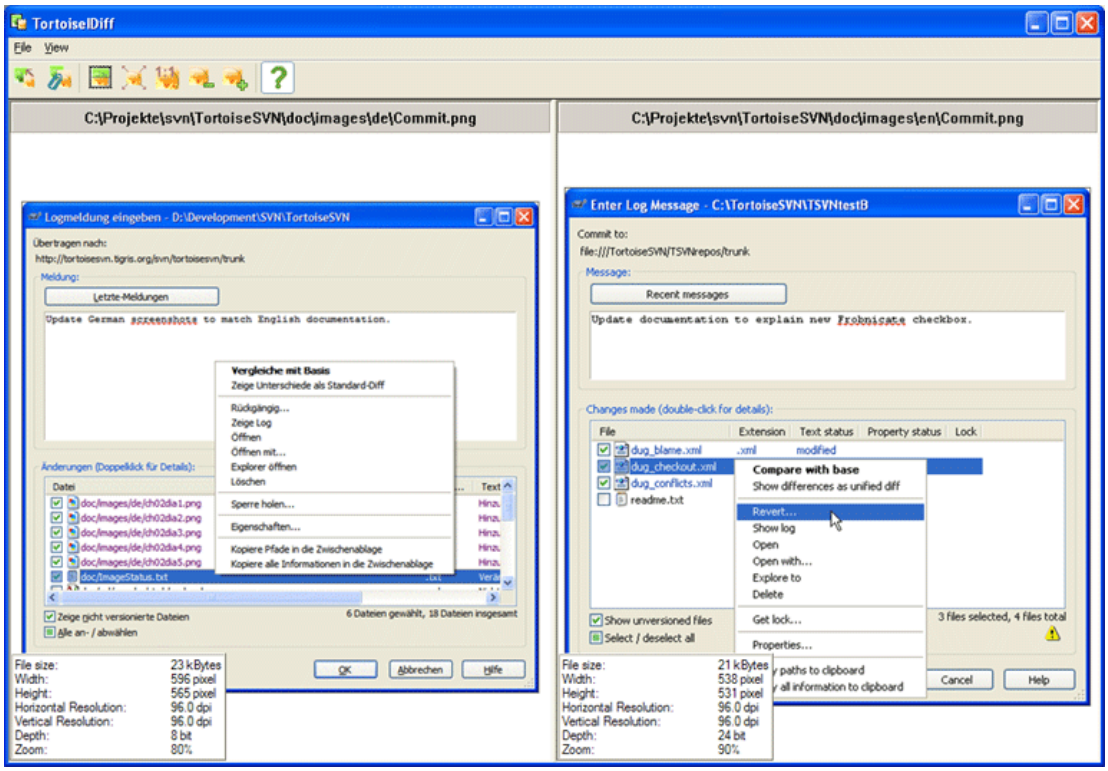


图 4.25. 差异察看器截图

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. □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。

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]。

SciTE
: 这款文本编译器在统一比较时提供语法显示，读起来更加容易。可以从这里下载Scintilla [http://www.scintilla.org/SciTEDownload.html]。

Notepad2
: Notepad2的设计旨在替代 Windows自带的记事本的功能，它以开源编译控制为基础。在查看统一比较时，它能实现比Windows自带的记事本更多功能。免费下载here [http://www.flos-freeware.ch/notepad2.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

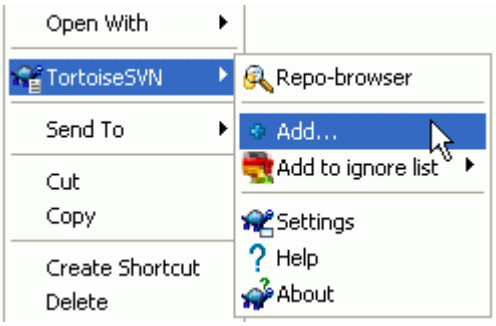


图 4.26. 未受版本控制的文件之资源管理器上下文菜单

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右拖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。

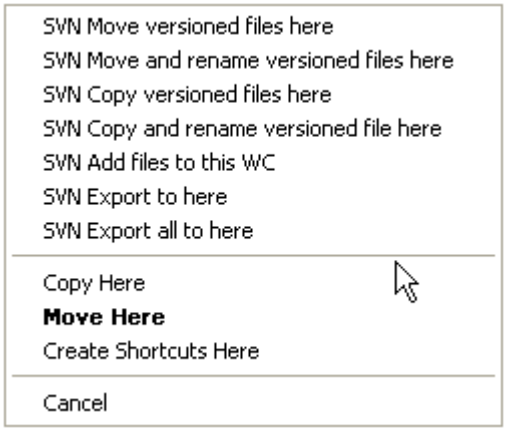


图 4.27. 版本控制下的一个目录的右键拖拽菜单

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

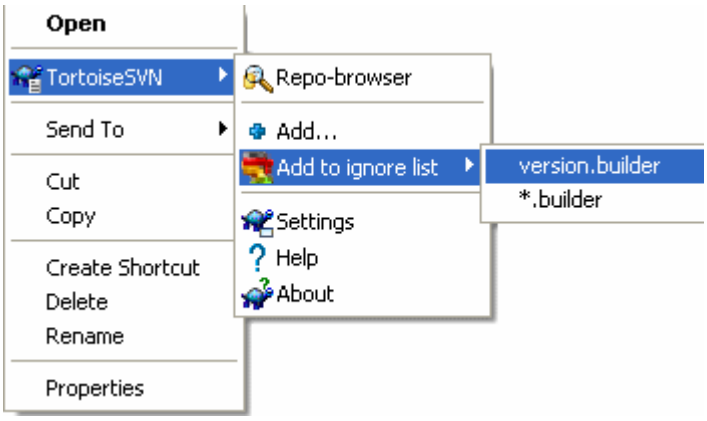


图 4.28. 未受版本控制的文件之资源管理器上下文菜单

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。

如果你右键一个单独的未版本控制文件，并从菜单栏选择TortoiseSVN → (加入忽略列表)Add to Ignore List，会出现一个子菜单允许你仅选择该文件，或者所有具有相同后缀的文件。如果你选择多种文件，那么就没有子菜单了，你仅能添加这些特定的文件/目录。

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

不像 CVS，Subversion 允许重命名和移动文件和目录。因此在 TortoiseSVN 的子菜单中有删除和更名的菜单项。

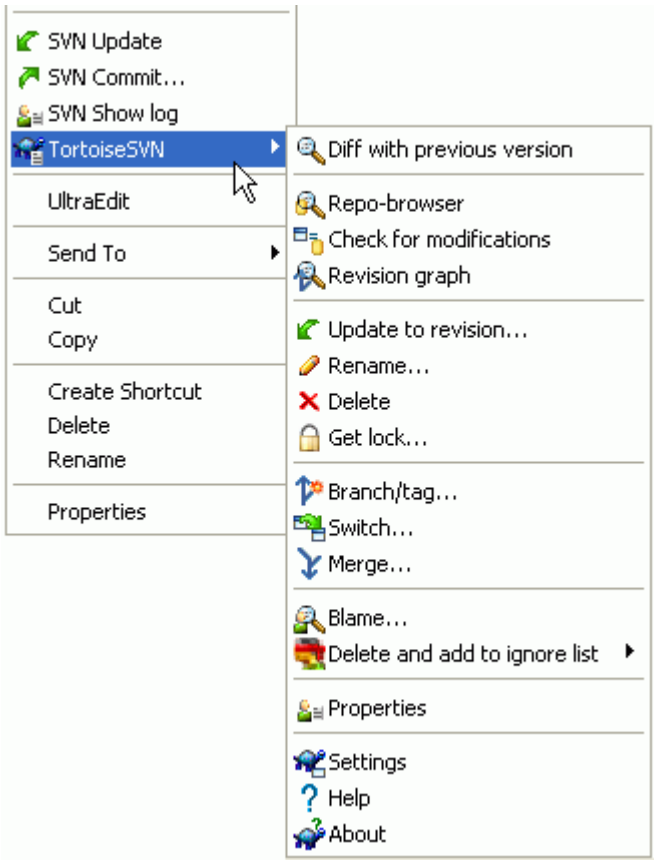


图 4.29. 版本控制文件的菜单浏览

4.14.1. □正在删除文件/文件夹

使用 TortoiseSVN → 删除 从 subversion 中删除文件或文件夹。

当你 TortoiseSVN → 删除 一个文件后，该文件立刻就从工作副本中移除，并且被标记，在下次提交时从版本库删除。该文件的父文件夹会覆盖上一个“已删除”重载图标。在你提交修改之前，你可以在父文件夹通过 TortoiseSVN → SVN 还原 命令找回文件。

当你TortoiseSVN → 删除一个文件夹，它仍旧在你的工作副本中，但是重载图标指出了它标记为删除。在你提交修改之前，你可以通过 TortoiseSVN → SVN 还原 命令找回文件夹。文件和文件夹处理方式的区别是 Subversion 的特性，与 TortoiseSVN 无关。

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。

如果一个 文件 是通过浏览器而不是使用TortoiseSVN 快捷菜单被删除，提交对话框也会显示这些文件并让你在提交前把他们从版本控制中移除。可是,如果你更新你的工作副本，Subversion 将会混淆这个丢失文件并替换他为版本库中的最新版本。因此，如果你需要删除一个版本控制下的文件，请始终使用TortoiseSVN → Delete保证 Subversion不去猜测你到底想干什么。

如果一个 目录 是通过浏览器而不是使用TortoiseSVN 快捷菜单被删除，你的工作副本将回被损坏，并且你将不能提交。如果你更新你的工作副本，如果你更新你的工作副本， Subversion 将用版本库中的最新版本替换已丢失目录。接下来你就可以使用TortoiseSVN → Delete这种正确的方法来删除它了。

找回已删除的文件或目录

如果你删除了洋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 → Revert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 → Copy to...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，可能是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。

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. □改变文件名称大小写

在 Windows 中，使用 Subversion 修改文件名称的大小写需要小技巧，因为在改名期间，两个文件名称需要同时存在。因为 Windows 的文件系统是大小写不敏感的，所以使用平常的改名命令是不能工作的。

这里(至少)有两种可能的解决方案来重命名文件而不丢失它的日志记录。在 subversion 里重命名是很重要的。仅在 Windows 资源管理器中重命名将会损坏你的工作副本！

解决方案 A)(推荐)

1. 提交你工作副本中的改变到版本库
2. 使用版本库的浏览器立即重命名该文件的大写(小写)为小写(大写)
3. 更新你的工作副本

解决方案 B)

1. 使用TortoiseSVN子菜单中的重命名命令将UPPERcase重命名为UPPERcase_ 格式
2. 提交该更改
3. 将UPPERcase_重命名为upperCASE格式
4. 提交该更改

4.14.4. □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。

4.14.5. □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。

4.14.6. □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。

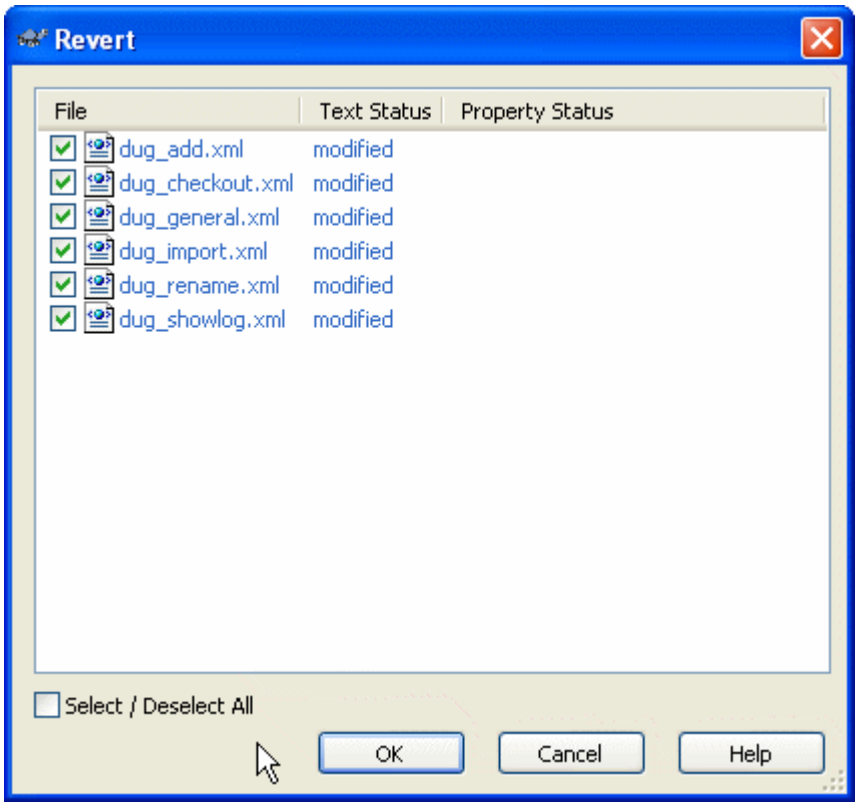


图 4.30. 恢复对话框

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　4.7.3　节　“本地与远程状态”

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。

清理有另外的一个有用的副作用。如果一个文件日期变化了但是它的内容没变，Subversion 除了逐字节地将该文件和原副本进行对照，不能分清它是否真的变更。 如果你有很多这种状态下的文件，将会使获得状态非常慢，还会导致许多会话响应变慢。在你的工作副本上执行一个清理命令将会修正这些“坏掉的”时戳，从而可以全速检查它们的状态。

使用提交时戳

Subversion的一些早期发布中存在一个bug，当你使用使用提交时戳选项检出的时候会造成时戳混乱。使用清理命令可以修正工作副本中的这些问题。

4.17. □项目设置

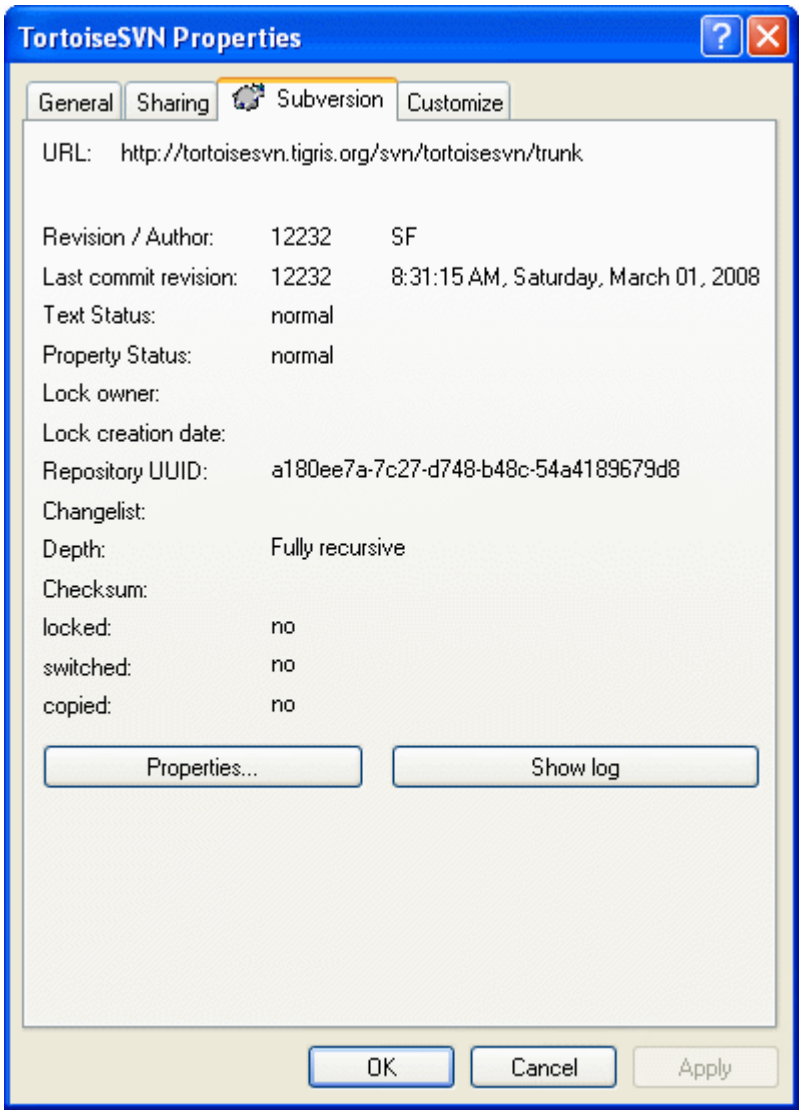


图 4.31. 资源管理器属性页，Subversion 页面

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。

4.17.1. □Subversion 属性

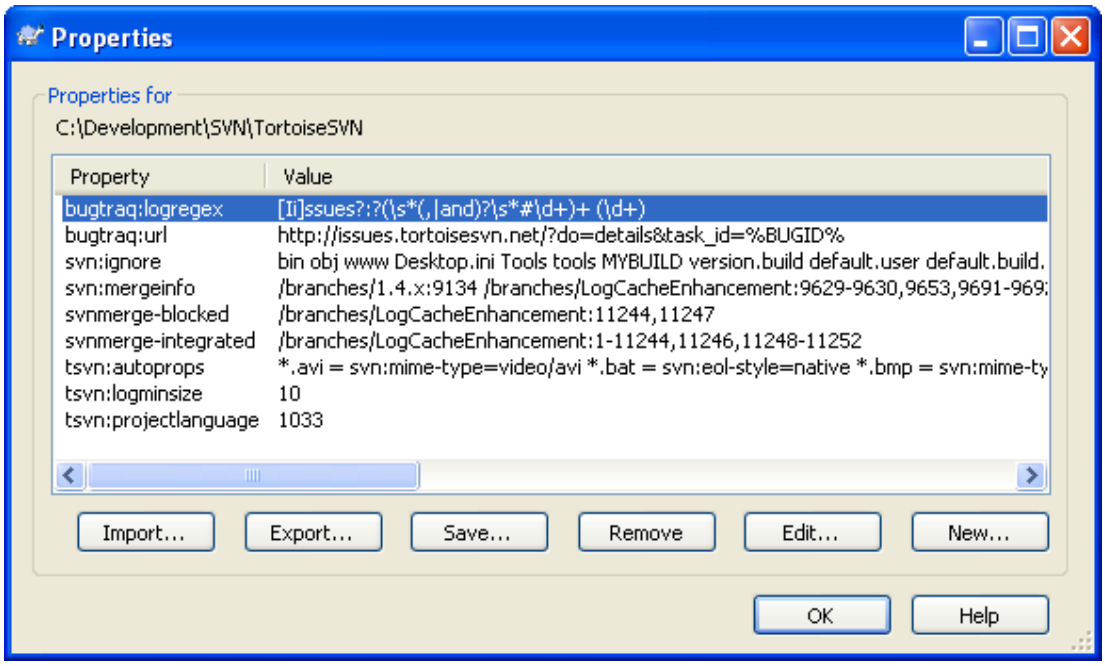


图 4.32. Subversion 属性页

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。

关键字的用法参见 Subversion 手册的 svn:keywords section [http://svnbook.red-bean.com/en/1.5/svn.advanced.props.special.keywords.html]，它给出了这些关键字的完整描述、如何启用和使用的方法。

更多的关于 Subversion 中属性的信息可以看Special Properties [http://svnbook.red-bean.com/en/1.5/svn.advanced.props.html]。

4.17.1.2. □增加和编辑属性

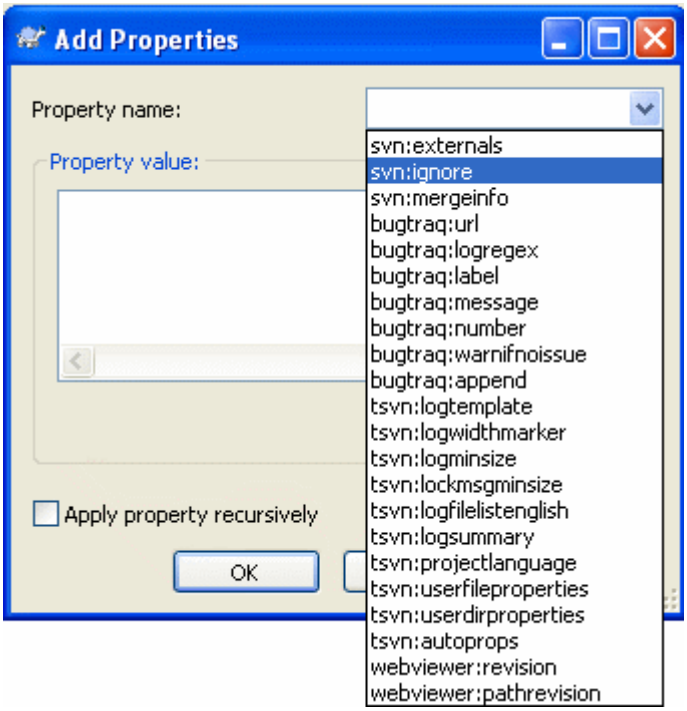


图 4.33. 增加属性

为了增加新属性，先单击增加...，从组合框中选择需要的属性名称，或者输入你自定义的名称，然后在下面的编辑框内输入取值。有多个取值的属性，例如忽略列表，肯呢个输入多行。单极确认将属性增加到属性列表。

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，然后选择上下文菜单→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。

一些属性，例如svn:needs-lock只能用于文件，所以它们在文件夹的属性下拉列表内不会出现。你仍旧可以递归的设置目录树中所有文件的属性，但是需要你自己输入属性名称。

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。

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。

你可以在编辑 subversion 配置文件来在客户端启用这一特性。在 TortoiseSVN 配置对话框的常规设置页中有一个编辑按钮可以直接打开这个文件。配置文件是一个普通的文本文件，它控制着 subversion 的某些运作。你需要修改两处：首先在 miscellany 打头的这一节中取消注释这一行：enable-auto-props = yes。其次，你需要编辑后面的一节，定义你想为哪些文件添加哪些属性。这个方法是标准的 subversion 特性而且对任何 subversion 客户端有效。不过它需要在每一个客户端进行定义 - 没有办法从版本库来设置这些参数。

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。

无论你选择哪一种方法，你应该注意： 自动属性设置仅在文件被添加到版本库时生效。自动属性设置不会改变已经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。

- Subversion 允许你设置 “自动属性(autoprops)”，它基于文件扩展名，对新增文件或导入的文件生效。这依赖于每个客户端在 Subversion 配置文件中设置适当的自动属性。tsvn:autoprops 可以对文件夹设置，当新增文件或导入文件时，它会与用户的本地自动属性合并。其格式与 subversion 的自动属性相同，例如，*.sh = svn:eol-style=native;svn:executable为扩展名为 .sh 的文件设置两个属性。

  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。

- 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。

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。

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

这些特别的项目属性只能设置在文件夹上才能生效。当你提交文件或文件夹时，这些属性从文件夹读取。如果没有发现这些属性，TortoiseSVN 会向上级文件夹搜索，直到未版本控制的文件夹，或到达根目录(例如 C:\) 。如果你确认每个用户都是从同一个文件夹检出，例如 trunk/，而不是其它子文件夹，那么只在 trunk/ 设置属性就足够了。如果你不能确定，那么应当递归的设置每个子文件夹。深层的设置覆盖高层的设置 (靠近 trunk/)。

> 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 增加新的目录时，任何父目录的项目属性都会被自动增加到子目录。

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。

4.18. □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

比如说，你检出了 /project1 到 D:\dev\project1。选择文件夹D:\dev\project1，右键单击并选择Windows 菜单 → 属性。出现属性对话框。在 Subversion 页，你可以设置属性。点击增加...。从组合框选择 svn:externals 属性，在编辑框按照 URL 文件夹 格式输入版本库URL，或者你需要一个指定的版本，那么使用 -r版本 URL 文件夹。你可以增加多个外部工程，每行一个。假设你为D:\dev\project1设置了这些属性:

```
http://sounds.red-bean.com/repos  sounds
http://graphics.red-bean.com/repos/fast%20graphics  "quick graphs"
-r21 http://svn.red-bean.com/repos/skin-maker  skins/toolkit
```

现在点击 设置，提交你的修改。当你(或其他用户)更新工作副本时，Subversion 将会创建子文件夹 D:\dev\project1\sounds，并且检出 sounds 工程，另一个子文件夹 D:\dev\project1\quick_graphs 包含 graphics 工程，最后一个嵌套的子文件夹 D:\dev\project1\skins\toolkit 包含版本为 21 的 skin-maker 工程。

URL 必须使用转义字符，否则它们不能正常工作，例如，你必须使用 %20 替代每一个空格符，就像在上面示范的第二条那样。

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。

旧的 svn:externals 定义

这里示范的格式是在 Subversion 1.5 中引入的。你会发现在旧格式中，相同的信息以不同的顺序排列。新格式要好一些，因为它支持一些有用的特性，下面会详细

> 描述这些特性，但是它不能用于旧版本的客户端程序。新旧格式的区别请参见 Subversion 手册 [http://svnbook.red-bean.com/en/1.5/svn.advanced.externals.html]。

如果一个外部工程位于同一版本库中，当你向主项目提交你的修改时，你对外部工程做的修改也会包含在提交列表中。

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。

如果需要，你也可以在 URL 后面指定一个固定的版本号，比如 http://sounds.red-bean.com/repos@19。

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。

在 Subversion 1.6 中外部文件支持不完整

在 subversion 1.6 中，一但你添加了外部文件，subversion 不会从你的工作副本中删除外部文件，即使你完全删除了 svn:externals 属性。你必须检出一个全新的工作副本来删除文件。

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:

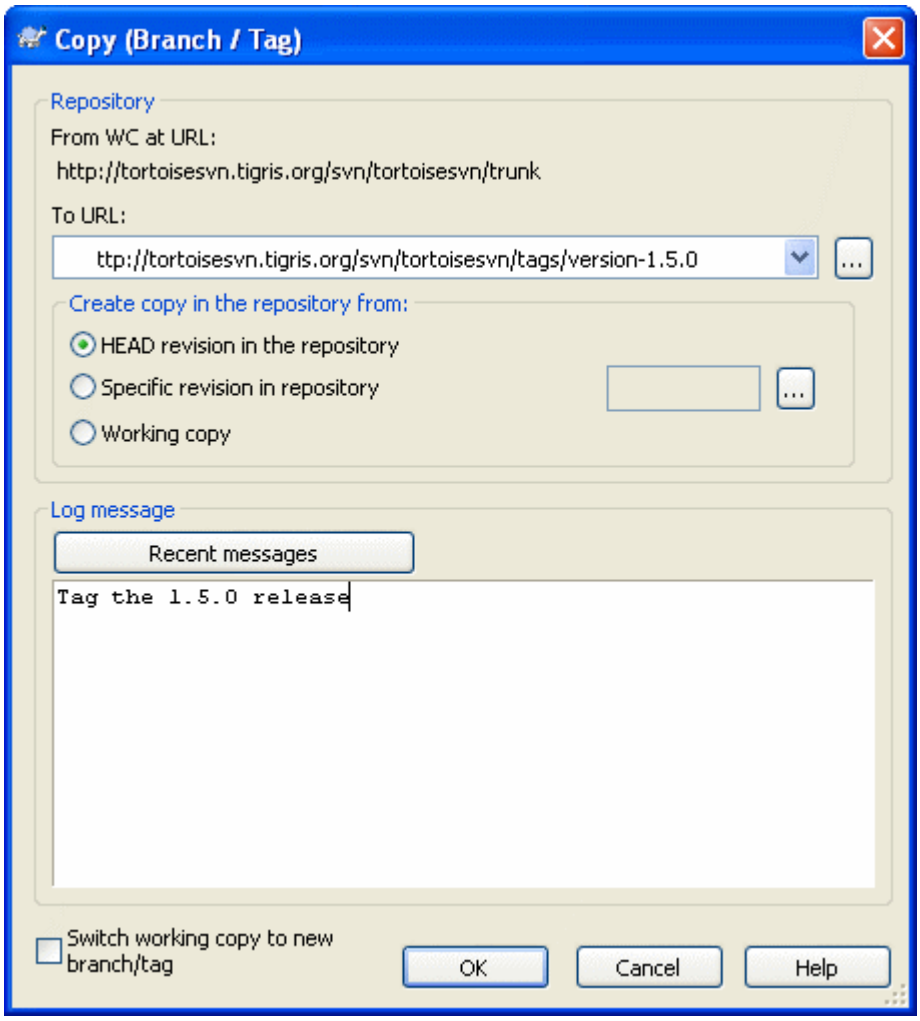


图 4.34. 分支/标记对话框

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。

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。

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. □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。

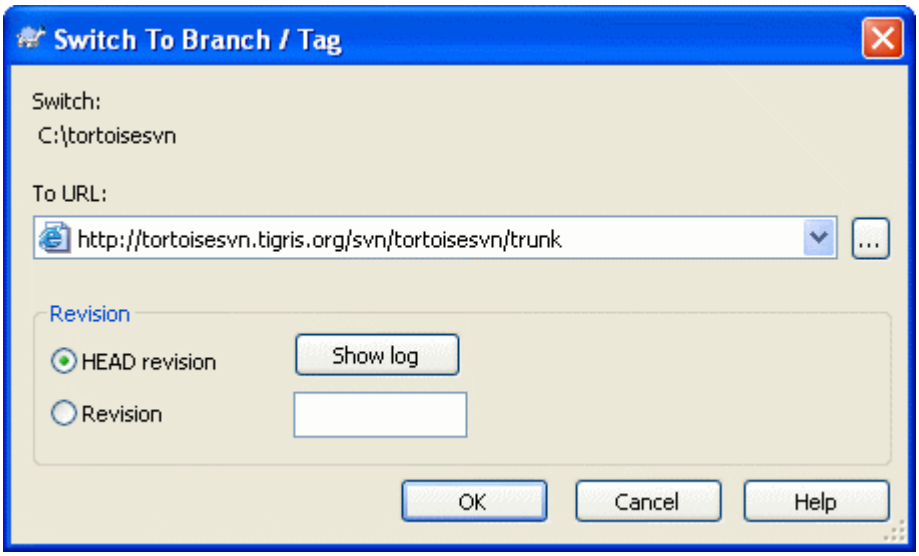


图 4.35. 切换对话框

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。

因为分支与合并很复杂，所以在你开始使用之前，请先理解它们是怎么工作的。强烈建议你阅读《使用 Subversion 进行版本管理》的分支与合并 [http://svnbook.red-bean.com/en/1.5/svn.branchmerge.html]章节，它给出了全面的描述，和许多使用举例。

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。”

复兴分支
这个用例覆盖了这种情况，当你象《Subversion手册》里讨论的那样，创建了一个新特性分支。所有主干的修改都要每周一次合并到新特性分支，等新特性完成后，你想将它合并到主干。因为你已经保持了新特性分支和主干同步，所以除了你在分支做的修改，其它部分在分支和最新版本应该是相同的。

这是下面讨论的树合并的特殊情况，它只需要你(通常情况下)想要合并的开发分支的 URL。它使用 Subversion 的合并跟踪特性来计算正确的版本范围，并且执行附加的检查来确保分支按照主干的修改进行了完整的更新。这样会确保你不会意外的撤销提交内容，这些内容是其他人在你上次同步修改后提交到主干的。

合并之后，分支的所有开发被完整的合并到主开发版本。现在的分支已经是多余的，可以删除。

一旦你执行了复兴合并，你就不应该继续使用它来进行开发。这样做的原因是，如果过后你试图再次对现有的分支与主干进行同步，合并跟踪将会你的复兴合并视为未合并到分支的主干修改，并且试图将分支到主干的合并再合并回分支！解决方案很简单，从主干创建一个新的分支来进行下一阶段的开发。

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。

如果服务器/版本库不支持合并跟踪，那么这是将分支合并回主干的唯一办法。另一个使用它的情况是当你使用供方分支时你需要合并沿着新供方的工作产生的修改到主干代码中。要了解更多的信息，请阅读《Subversion 手册》的供方分支 [http://svnbook.red-bean.com/en/1.5/svn.advanced.vendorbr.html]一章。

4.20.1. □合并指定版本范围

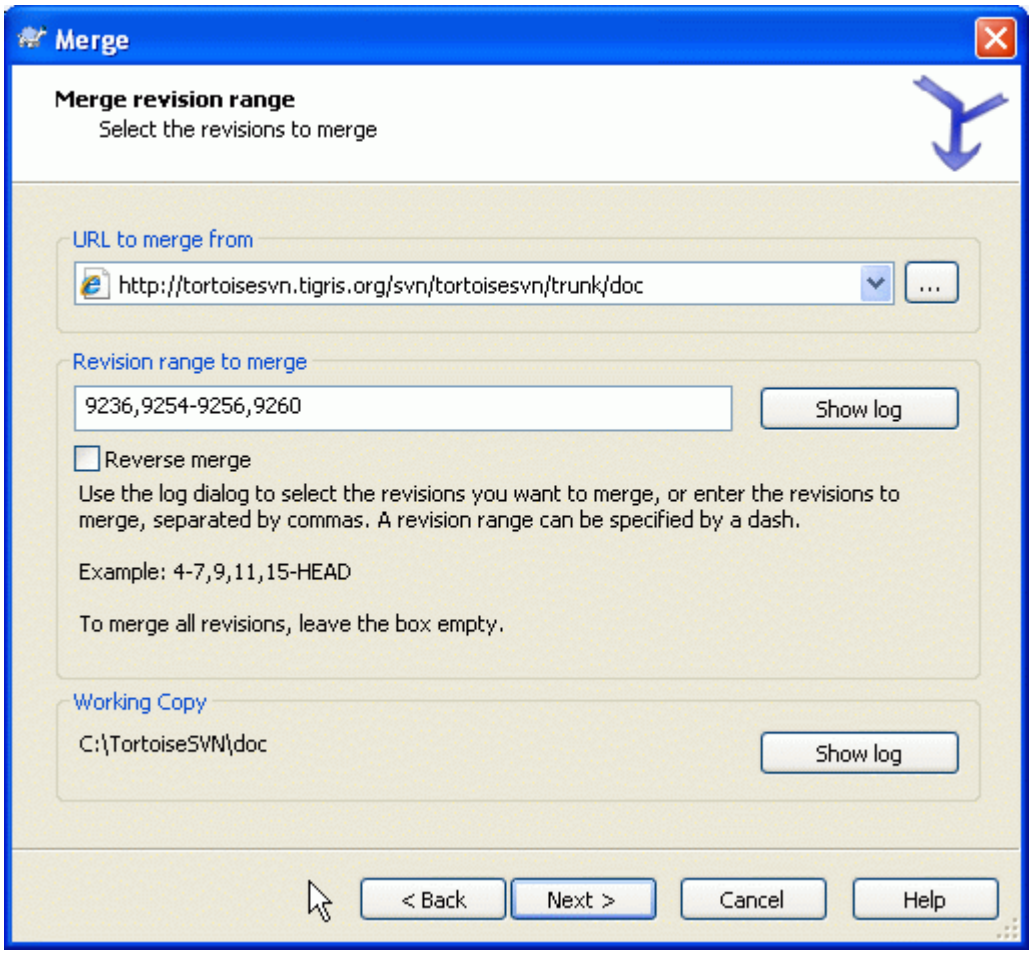


图 4.36. 合并向导 - 选择版本范围

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。

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。

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。

如果你要合并大量的版本，按照在 subversion 手册中所示的方法，你要在这次合并 100-200 下次合并 200-300。如果使用 TortoiseSVN，你要在这次合并 100-200 下次合并 201-300。

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。

If you want to merge changes back out of your working copy, to revert a change which has already been committed, select the revisions to revert and make sure the Reverse merge box is checked.

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 4.20.6 节 “合并跟踪获得更多信息。

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。

点击 下一步 进入 第 4.20.4 节 “合并选项”

4.20.2. □复兴分支

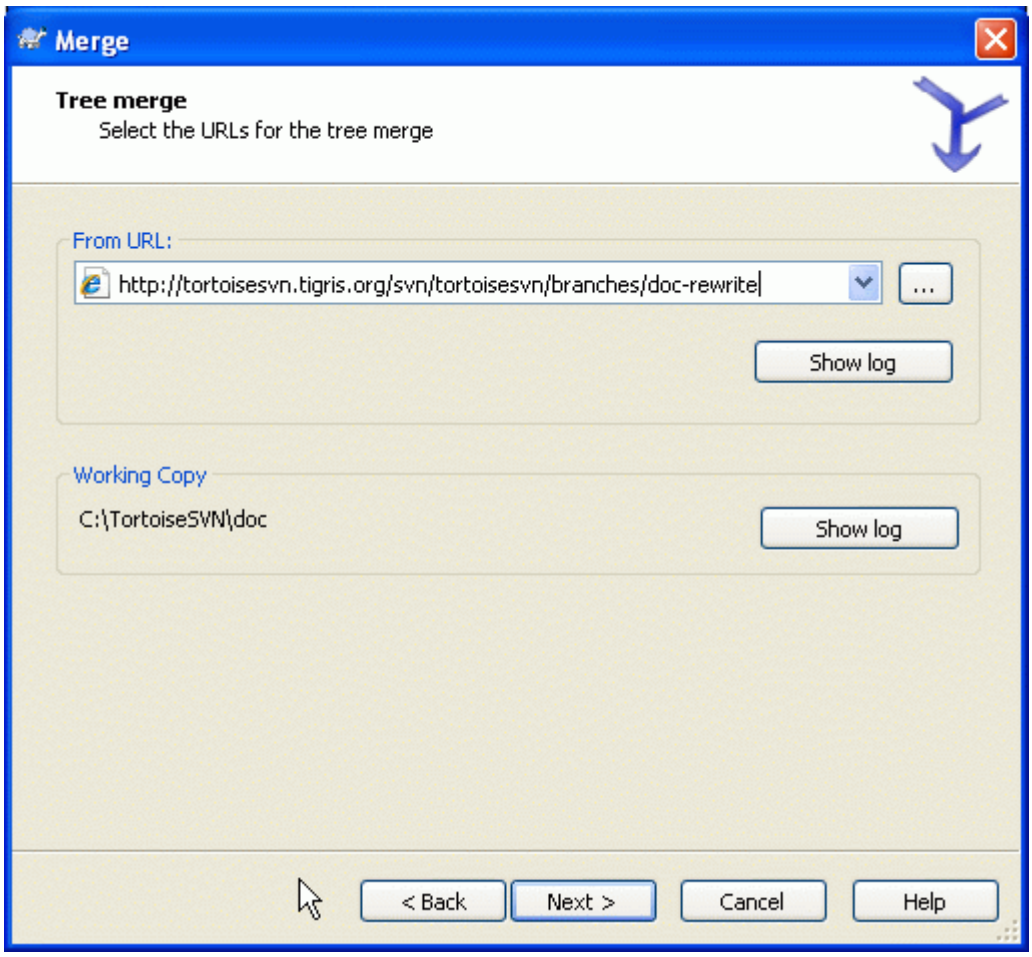


图 4.37. The Merge Wizard - Reintegrate Merge

To merge a feature branch back into the trunk you must start the merge wizard from within a working copy of the trunk.

In the From URL: field enter the full folder URL of the branch that you want to merge back. You may also click ... to browse the repository.

There are some conditions which apply to a reintegrate merge. Firstly, the server must support merge tracking. The working copy must be of depth infinite (no sparse checkouts), and it must not have any local modifications, switched items or items that have been updated to revisions other than HEAD. All changes to trunk made during branch development must have been merged across to the branch (or marked as having been merged). The range of revisions to merge will be calculated automatically.

4.20.3. □合并两个不同的目录树

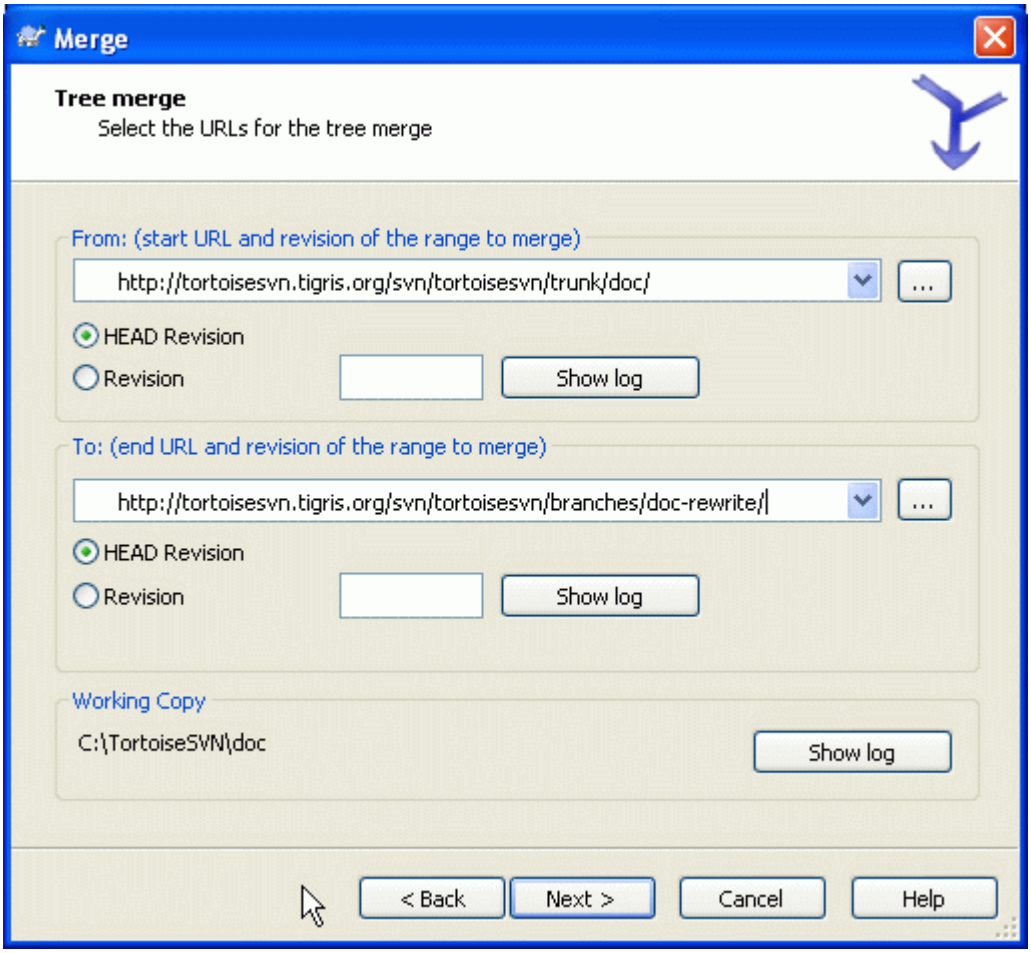


图 4.38. 合并向导 - 树合并

If you are using this method to merge a feature branch back to trunk, you need to start the merge wizard from within a working copy of trunk.

In the From: field enter the full folder URL of the trunk. This may sound wrong, but remember that the trunk is the start point to which you want to add the branch changes. You may also click ... to browse the repository.

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。

4.20.4. □合并选项

This page of the wizard lets you specify advanced options, before starting the merge process. Most of the time you can just use the default settings.

You can specify the depth to use for the merge, i.e. how far down into your working copy the merge should go. The depth terms used are described in 第 4.3.1 节 “检出深度”. The default depth is Working copy, which uses the existing depth setting, and is almost always what you want.

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/1.5/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry]

You can specify the way that line ending and whitespace changes are handled. These options are described in 第 4.10.2 节 “行结束符和空白选项”. The default behaviour is to treat all whitespace and line-end differences as real changes to be merged.

If you are using merge tracking and you want to mark a revision as having been merged, without actually doing the merge here, check the Only record the merge checkbox. There are two possible reasons you might want to do this. It may be that the merge is too complicated for the merge algorithms, so you code the changes by hand, then mark the change as merged so that the merge tracking algorithm is aware of it. Or you might want to prevent a particular revision from being merged. Marking it as already merged will prevent the merge occurring with merge-tracking-aware clients.

Now everything is set up, all you have to do is click on the Merge button. If you want to preview the results Test Merge performs the merge operation, but does not modify the working copy at all. It shows you a list of the files that will be changed by a real merge, and notes those areas where conflicts will occur.

The merge progress dialog shows each stage of the merge, with the revision ranges involved. This may indicate one more revision than you were expecting. For example if you asked to merge revision 123 the progress dialog will report “Merging revisions 122 through 123”. To understand this you need to remember that Merge is closely related to Diff. The merge process works by generating a list of differences between two points in the repository, and applying those differences to your working copy. The progress dialog is simply showing the start and end points for the diff.

4.20.5. □预览合并结果

The merge is now complete. It's a good idea to have a look at the merge and see if it's as expected. Merging is usually quite complicated. Conflicts often arise if the branch has drifted far from the trunk.

For Subversion clients and servers prior to 1.5, no merge information is stored and merged revisions have to be tracked manually. When you have tested the changes and come to commit this revision, your commit log message should always include the revision numbers which have been ported in the merge. If you want to apply

another merge at a later time you will need to know what you have already merged, as you do not want to port a change more than once. For more information about this, refer to Best Practices for Merging [http://svnbook.red-bean.com/en/1.4/svn.branchmerge.copychanges.html#svn.branchmerge.copychanges.bestprac] in the Subversion book.

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。

If you have just merged a feature branch back into the trunk, the trunk now contains all the new feature code, and the branch is obsolete. You can now delete it from the repository if required.

Subversion can't merge a file with a folder and vice versa - only folders to folders and files to files. If you click on a file and open up the merge dialog, then you have to give a path to a file in that dialog. If you select a folder and bring up the dialog, then you must specify a folder URL for the merge.

4.20.6. □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。

- You can avoid the danger of merging the same revision twice (repeated merge problem). Once a revision is marked as having been merged, future merges which include that revision in the range will skip over it.
- When you merge a branch back into trunk, the log dialog can show you the branch commits as part of the trunk log, giving better traceability of changes.
- When you show the log dialog from within the merge dialog, revisions already merged are shown in grey.
- When showing blame information for a file, you can choose to show the original author of merged revisions, rather than the person who did the merge.
- You can mark revisions as do not merge by including them in the list of merged revisions without actually doing the merge.

Merge tracking information is stored in the svn:mergeinfo property by the client when it performs a merge. When the merge is committed the server stores that information in a database, and when you request merge, log or blame information, the server can respond appropriately. For the system to work properly you must ensure that the server, the repository and all clients are upgraded. Earlier clients will not store the svn:mergeinfo property and earlier servers will not provide the information requested by new clients.

Find out more about merge tracking from Subversion's Merge tracking documentation [http://subversion.tigris.org/merge-tracking/index.html].

4.20.7. □子合并期间处理冲突

Merging does not always go smoothly. Sometimes there is a conflict, and if you are merging multiple ranges, you generally want to resolve the conflict before merging of the next range starts. TortoiseSVN helps you through this process by showing the merge conflict callback dialog.

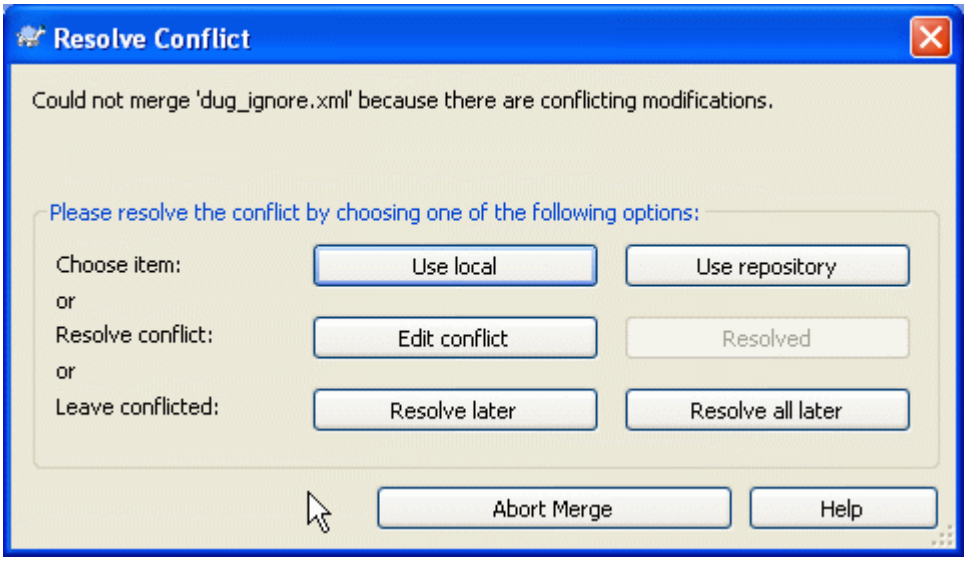


图 4.39. 合并冲突回调对话框

When a conflict occurs during the merge, you have three ways to handle it.

1. You may decide that your local changes are much more important, so you want to discard the version from the repository and keep your local version. Or you might discard your local changes in favour of the repository version. Either way, no attempt is made to merge the changes - you choose one or the other.

2. Normally you will want to look at the conflicts and resolve them. In that case, choose the Edit Conflict which will start up your merge tool. When you are satisfied with the result, click Resolved.

3. The last option is to postpone resolution and continue with merging. You can choose to do that for the current conflicted file, or for all files in the rest of the merge. However, if there are further changes in that file, it will not be possible to complete the merge.

If you do not want to use this interactive callback, there is a checkbox in the merge progress dialog Merge non-interactive. If this is set for a merge and the merge would result in a conflict, the file is marked as in conflict and the merge goes on. You will have to resolve the conflicts after the whole merge is finished. If it is not set, then before a file is marked as conflicted you get the chance to resolve the conflict during the merge. This has the advantage that if a file gets multiple merges (multiple revisions apply a change to that file), subsequent merges might succeed depending on which lines are affected. But of course you can't walk away to get a coffee while the merge is running ;)

4.20.8. □Merge a Completed Branch

If you want to merge all changes from a feature branch back to trunk, then you can use the TortoiseSVN → Merge reintegrate... from the extended context menu (hold down the Shift key while you right click on the file).

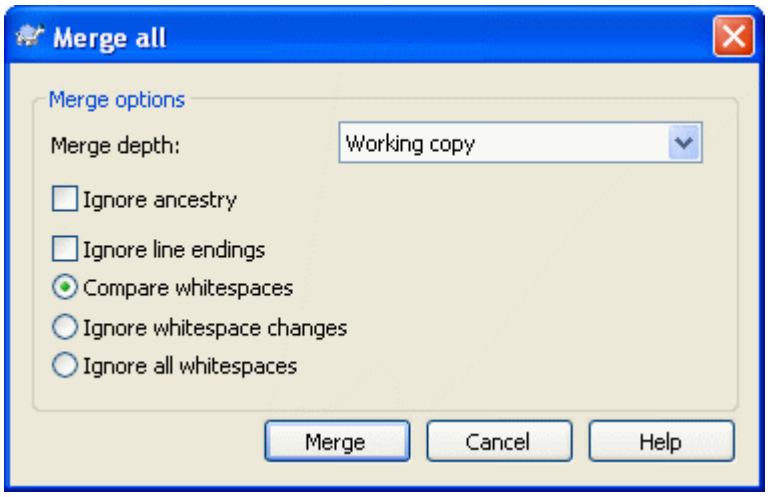


图 4.40. The Merge reintegrate Dialog

This dialog is very easy. All you have to do is set the options for the merge, as described in 第 4.20.4 节 “合并选项”. The rest is done by TortoiseSVN automatically using merge tracking.

4.20.9. □Feature Branch Maintenance

When you develop a new feature on a separate branch it is a good idea to work out a policy for re-integration when the feature is complete. If other work is going on in trunk at the same time you may find that the differences become significant over time, and merging back becomes a nightmare.

If the feature is relatively simple and development will not take long then you can adopt a simple approach, which is to keep the branch entirely separate until the feature is complete, then merge the branch changes back into trunk. In the merge wizard this would be a simple Merge a range of revisions, with the revision range being the revision span of the branch.

If the feature is going to take longer and you need to account for changes in trunk, then you need to keep the branch synchronised. This simply means that periodically you merge trunk changes into the branch, so that the branch contains all the trunk changes plus the new feature. The synchronisation process uses Merge a range of revisions. When the feature is complete then you can merge it back to trunk using either Reintegrate a branch or Merge two different trees.

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。
- Your company has always used a locking revision control system in the past and there has been a management decision that “locking is best”.

Firstly you need to ensure that your Subversion server is upgraded to at least version 1.2. Earlier versions do not support locking at all. If you are using file:// access, then of course only your client needs to be updated.

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。

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....

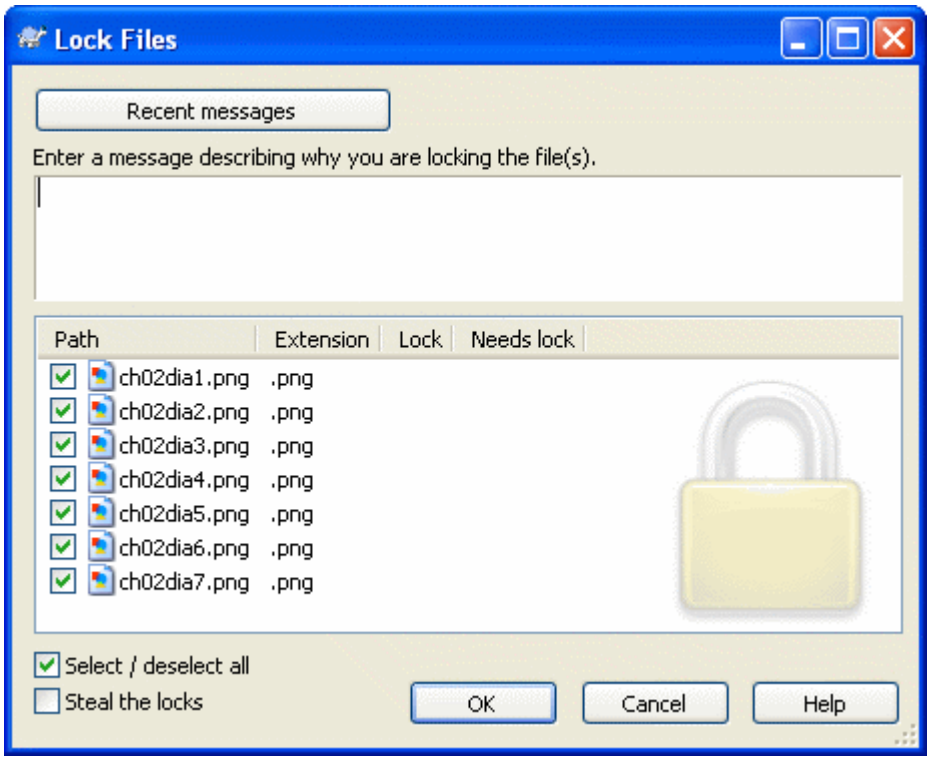


图 4.41. 锁定对话框

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。

If you select a folder and then use TortoiseSVN → Get Lock... the lock dialog will open with every file in every sub-folder selected for locking. If you really want to lock an entire hierarchy, that is the way to do it, but you could become very unpopular with your co-workers if you lock them out of the whole project. Use with care ...

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

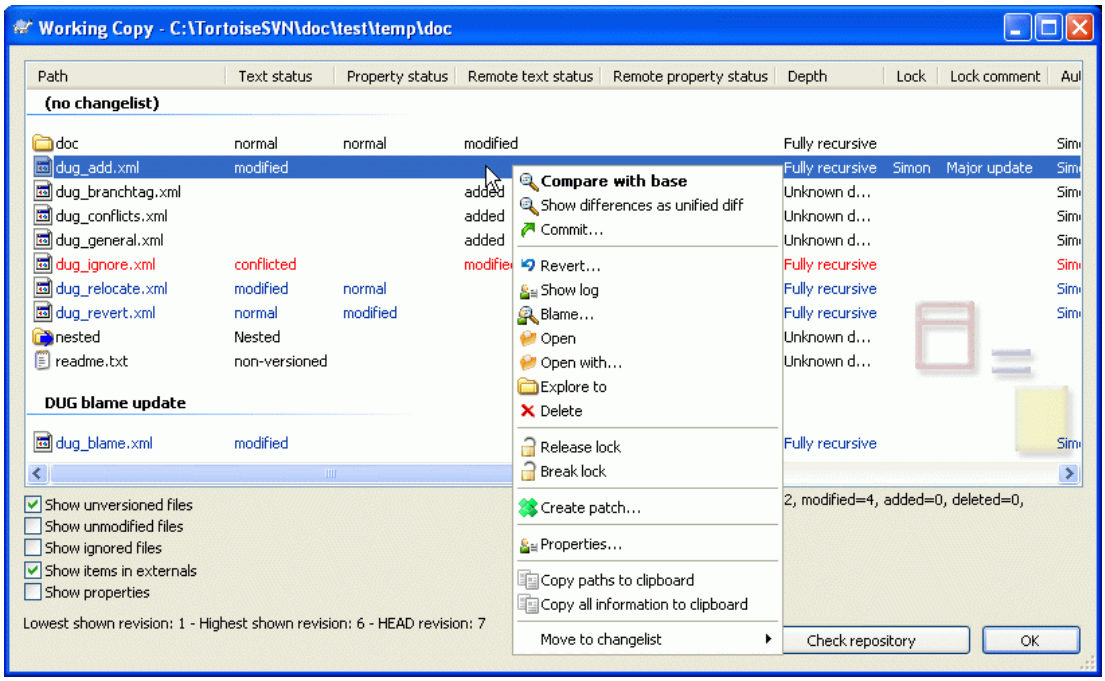


图 4.42. 检查修改对话框

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。

If you operate a policy where every file has to be locked then you may find it easier to use Subversion's auto-props feature to set the property automatically every time you add new files. Read 第 4.17.1.5 节 “自动属性设置”for further information.

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...。

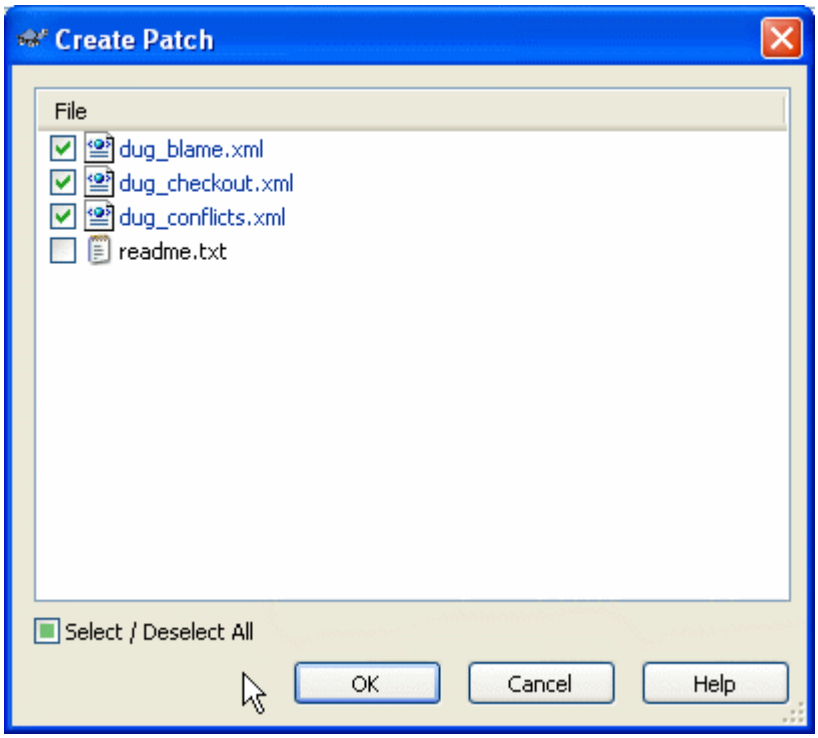


图 4.43. 创建补丁的对话框

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 4.7.3 节 “本地与远程状态”

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。

4.22.2. □应用一个补丁文件

Patch files are applied to your working copy. This should be done from the same folder level as was used to create the patch. If you are not sure what this is, just look at the first line of the patch file. For example, if the first file being worked on was doc/source/english/chapter1.xml and the first line in the patch file is Index: english/chapter1.xml then you need to apply the patch to the doc/source/ folder. However, provided you are in the correct working copy, if you pick the wrong folder level, TortoiseSVN will notice and suggest the correct level.

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。

From the context menu for that folder, click on TortoiseSVN → Apply Patch... This will bring up a file open dialog allowing you to select the patch file to apply. By

default only .patch or .diff files are shown, but you can opt for “All files”. If you previously saved a patch to the clipboard, you can use Open from clipboard... in the file open dialog.

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

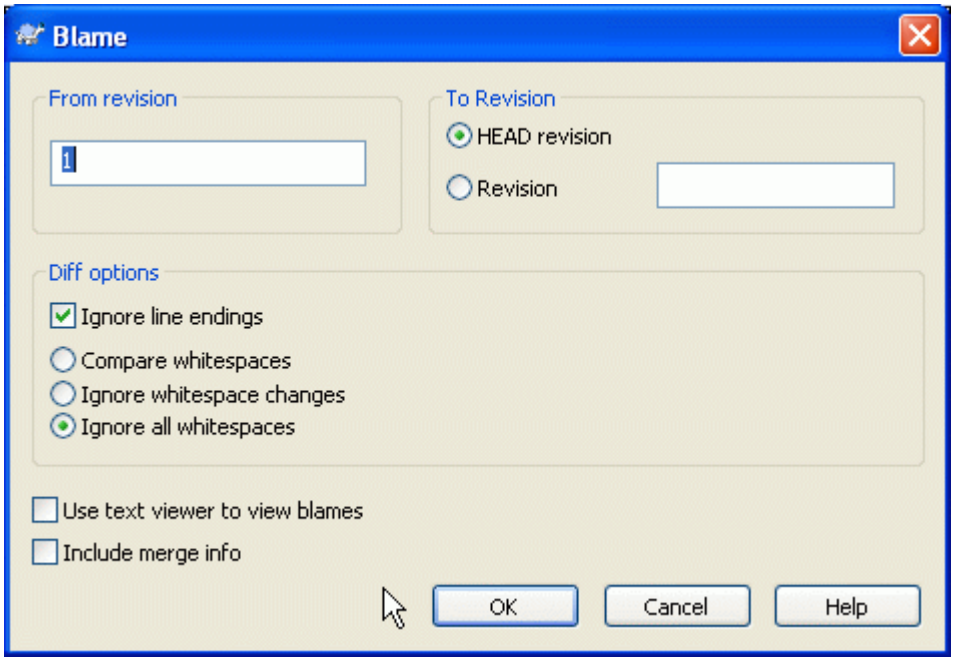


图 4.44. 评注/追溯对话框

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，追溯文件使用TortoiseBlame,这个工具可以高亮显示不同版本从而使阅读更加容易。如果想打印或者编辑追溯文件，复选使用文字编辑器查看追溯信息。

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。

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。

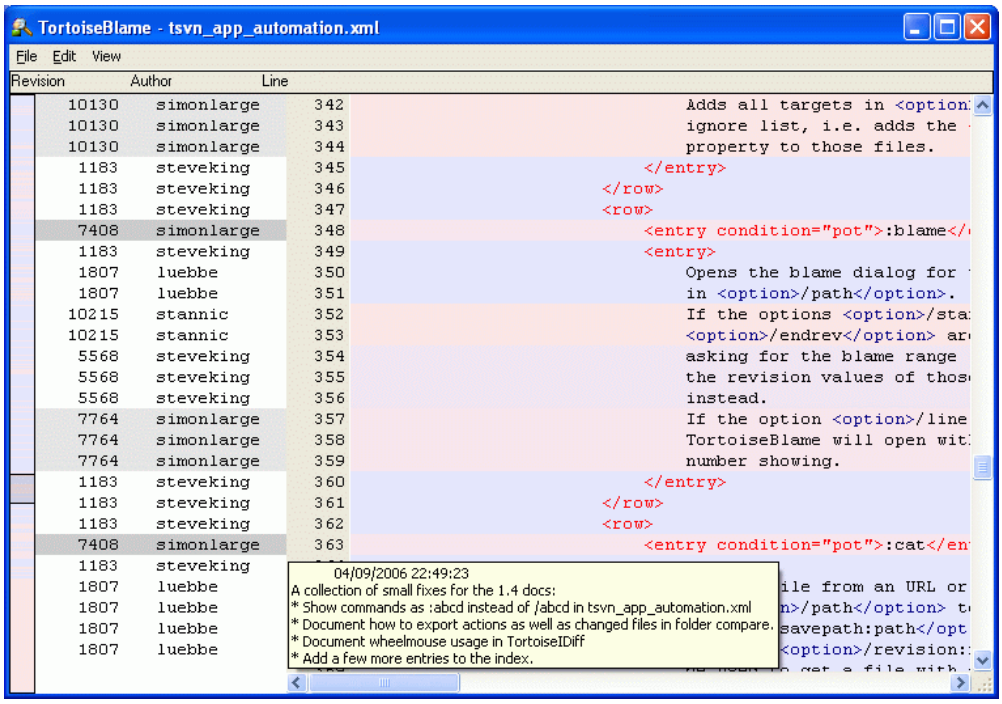


图 4.45. TortoiseBlame

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。

If you need a better visual indicator of where the oldest and newest changes are, select View → Color age of lines. This will use a colour gradient to show newer lines

in red and older lines in blue. The default colouring is quite light, but you can change it using the TortoiseBlame settings.

If you are using Merge Tracking, where lines have 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. If you do not want merged lines shown in this way, uncheck the Include merge info checkbox.

If you want to see the paths involved in the merge, select View → Merge paths.

The settings for TortoiseBlame can be accessed using TortoiseSVN → Settings... on the TortoiseBlame tab. Refer to 第 4.30.9 节 “TortoiseBlame 的设置”.

4.23.2. □追溯不同点

追溯报告的一个局限性就是它仅仅象显示一个特殊版本一样显示文件，并显示出修改每一行的最后一个人。有时候你想知道谁做了什么修改。你所需要的就是把差异和追溯报告结合起来。

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
: Select one revision in the top pane, then pick one file in the bottom pane and select Context menu → Blame changes. This will fetch the blame data for the selected revision and the previous revision, then use the diff viewer to compare the two blame files.

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。

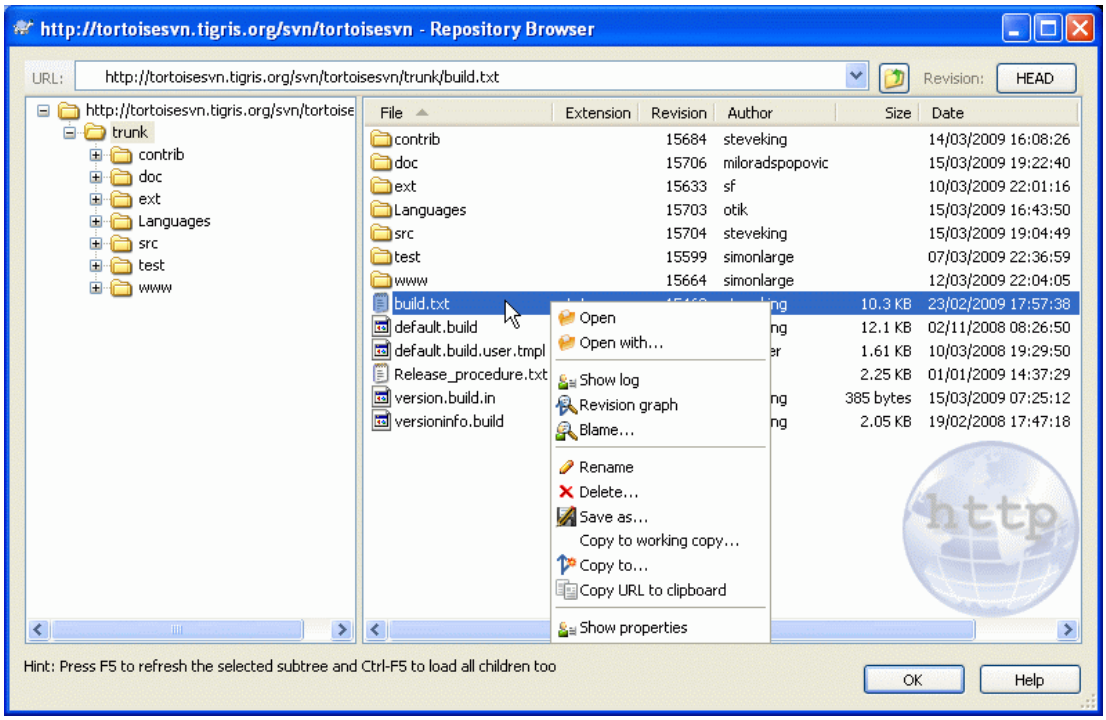


图 4.46. 版本库浏览器

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。

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。
- 在你的硬盘上保存此文件的一个未版本控制的副本。
- 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。
- 删除或改名文件。
- 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。
- View/Edit the file's properties.

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。
- 直接向版本库增加文件或目录。
- 删除或改名文件夹。
- 复制这个目录，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。
- 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

You can use F5 to refresh the view as usual. This will refresh everything which is currently displayed. If you want to pre-fetch or refresh the information for nodes which have not been opened yet, use Ctrl-F5. After that, expanding any node will happen instantly without a network delay while the information is fetched.

You can also use the repository browser for drag-and-drop operations. If you drag a folder from explorer into the repo-browser, it will be imported into the repository. Note that if you drag multiple items, they will be imported in separate commits.

If you want to move an item within the repository, just left drag it to the new location. If you want to create a copy rather than moving the item, Ctrl-left drag instead. When copying, the cursor has a “plus” symbol on it, just as it does in Explorer.

If you want to copy/move a file or folder to another location and also give it a new name at the same time, you can right drag or Ctrl-right drag the item instead of using left drag. In that case, a rename dialog is shown where you can enter a new name for the file or folder.

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。

Sometimes when you try to open a path you will get an error message in place of the item details. This might happen if you specified an invalid URL, or if you don't have access permission, or if there is some other server problem. If you need to copy this message to include it in an email, just right click on it and use Context Menu → Copy error message to clipboard, or simply use Ctrl+C.

4.25. □版本分支图

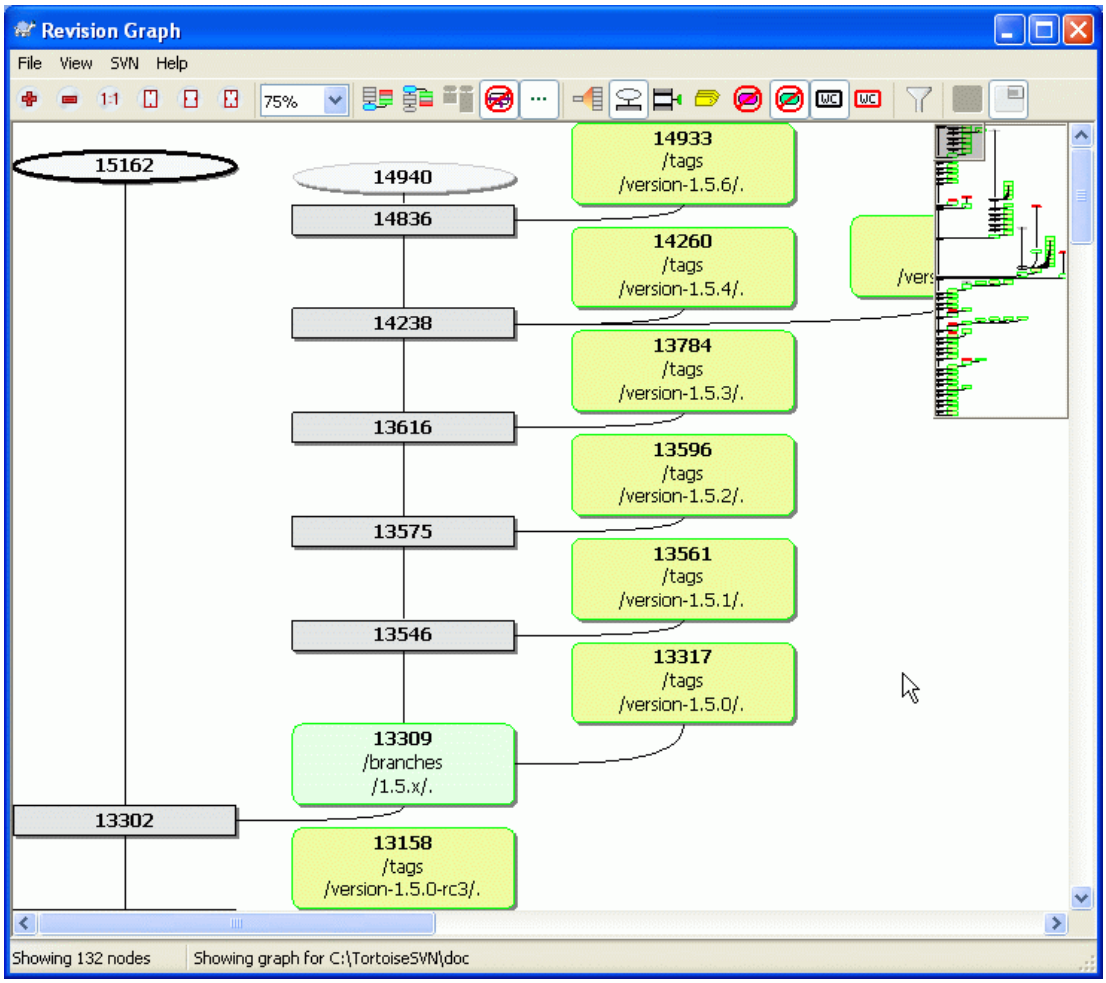


图 4.47. 一个版本分支

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。

In order to generate the graph, TortoiseSVN must fetch all log messages from the repository root. Needless to say this can take several minutes even with a repository of a few thousand revisions, depending on server speed, network bandwidth, etc. If you try this with something like the Apache project which currently has over 500,000 revisions you could be waiting for some time.

The good news is that if you are using log caching, you only have to suffer this delay once. After that, log data is held locally. Log caching is enabled in TortoiseSVN's settings.

4.25.1. □版本图节点

Each revision graph node represents a revision in the repository where something changed in the tree you are looking at. Different types of node can be distinguished by shape and colour. The shapes are fixed, but colours can be set using TortoiseSVN → Settings

Added or copied items
: Items which have been added, or created by copying another file/folder are shown using a rounded rectangle. The default colour is green. Tags and trunks are treated as a special case and use a different shade, depending on the TortoiseSVN → Settings

Deleted items
: Deleted items eg. a branch which is no longer required, are shown using an octagon (rectangle with corners cut off). The default colour is red.

Renamed items
: Renamed items are also shown using an octagon, but the default colour is blue.

分支最新版本
: The graph is normally restricted to showing branch points, but it is often useful to be able to see the respective HEAD revision for each branch too. If you select Show HEAD revisions, each HEAD revision nodes will be shown as an ellipse. Note that HEAD here refers to the last revision committed on that path, not to the HEAD revision of the repository.

Working copy revision
: If you invoked the revision graph from a working copy, you can opt to show the BASE revision on the graph using Show WC revision, which marks the BASE node with a bold outline.

Modified working copy
: If you invoked the revision graph from a working copy, you can opt to show an additional node representing your modified working copy using Show WC modifications. This is an elliptical node with a bold outline in red by default.

Normal item
: 其他一般的文件用plain rectangle标示。

Note that by default the graph only shows the points at which items were added, copied or deleted. Showing every revision of a project will generate a very large graph for non-trivial cases. If you really want to see all revisions where changes were made, there is an option to do this in the View menu and on the toolbar.

The default view (grouping off) places the nodes such that their vertical position is in strict revision order, so you have a visual cue for the order in which things were done. Where two nodes are in the same column the order is very obvious. When two nodes are in adjacent columns the offset is much smaller because there is no need to prevent the nodes from overlapping, and as a result the order is a little less obvious. Such optimisations are necessary to keep complex graphs to a reasonable size. Please note that this ordering uses the edge of the node on the older side as a reference, i.e. the bottom edge of the node when the graph is shown with oldest node at the bottom. The reference edge is significant because the node shapes are not all the same height.

4.25.2. □Changing the View

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
: The default behavior (grouping off) has all rows sorted strictly by revision. As a result, long-living branches with sparse commits occupy a whole column for only a few changes and the graph becomes very broad.

  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 older 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。

Align trees on top
When a graph is broken into several smaller trees, the trees may appear either in natural revision order, or aligned at the bottom of the window, depending on whether you are using the Group Branches option. Use this option to grow all trees down from the top instead.

减少交叉行
If the layout of the graph has produced a lot of crossing lines, use this option to clean it up. This may 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.

Differential path names
Long path names can take a lot of space and make the node boxes very large. Use this option to show only the changed part of a path, replacing the common part with dots. E.g. if you create a branch /branches/1.2.x/doc/html from /trunk/doc/html the branch could be shown in compact form as /branches/1.2.x/.. because the last two levels, doc and html, did not change.

Show all revisions
This does just what you expect and shows every revision where something (in the tree that you are graphing) has changed. For long histories this can produce a truly huge graph.

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
When a branch/tag is made, the default behaviour is to show the branch as taken from the last node where a change was made. Strictly speaking this is inaccurate since the branches are often made from the current HEAD rather than a specific revision. So it is possible to show the more correct (but less useful) revision that was used to create the copy. Note that this revision may be younger than the HEAD revision of the source branch.

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.

Hide deleted paths
Hides paths which are no longer present at the HEAD revision of the repository, e.g. deleted branches.

Hide unchanged branches
Hides branches where no changes were committed to the respective file or sub-folder. This does not necessarily indicate that the branch was not used, just that no changes were made to this part of it.

Show WC revision
: Marks the revision on the graph which corresponds to the update revision of the item you fetched the graph for. If you have just updated, this will be HEAD, but if others have committed changes since your last update your WC may be a few revisions lower down. The node is marked by giving it a bold outline.

Show WC modifications
: If your WC contains local changes, this option draws it as a separate elliptical node, linked back to the node that your WC was last updated to. The default outline colour is red. You may need to refresh the graph using F5 to capture recent changes.

过滤器
: Sometimes the revision graph contains more revisions than you want to see. This option opens a dialog which allows you to restrict the range of revisions displayed, and to hide particular paths by name.

Tree stripes
: Where the graph contains several trees, it is sometimes useful to use alternating colours on the background to help distinguish between trees.

Show overview
: Shows a small picture of the entire graph, with the current view window as a rectangle which you can drag. This allows you to navigate the graph more easily. Note that for very large graphs the overview may become useless due to the extreme zoom factor and will therefore not be shown in such cases.

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。

If you select two revisions (Use Ctrl-left click), you can use the context menu to show the differences between these revisions. You can choose to show differences as at the branch creation points, but usually you will want to show the differences at the branch end points, i.e. at the HEAD revision.

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。

You can also merge changes in the selected revision(s) into a different working copy. A folder selection dialog allows you to choose the working copy to merge into, but after that there is no confirmation dialog, nor any opportunity to try a test merge. It is a good idea to merge into an unmodified working copy so that you can revert the changes if it doesn't work out! This is a useful feature if you want to merge selected revisions from one branch to another.

Learn to Read the Revision Graph

First-time users may be surprised by the fact that the revision graph shows something that does not match the user's mental model. If a revision changes multiple copies or branches of a file or folder, for instance, then there will be multiple nodes for that single revision. It is a good practice to start with the leftmost options in the toolbar and customize the graph step-by-step until it comes close to your mental model.

> All filter options try lose as little information as possible. That may cause some nodes to change their color, for instance. Whenever the result is unexpected, undo the last filter operation and try to understand what is special about that particular revision or branch. In most cases, the initially expected outcome of the filter operation would either be inaccurate or misleading.

4.25.4. □刷新视图

If you want to check the server again for newer information, you can simply refresh the view using F5. If you are using the log cache (enabled by default), this will check the repository for newer commits and fetch only the new ones. If the log cache was in offline mode, this will also attempt to go back online.

If you are using the log cache and you think the message content or author may have changed, you should use the log dialog to refresh the messages you need. Since the revision graph works from the repository root, we would have to invalidate the entire log cache, and refilling it could take a very long time.

4.25.5. □Pruning Trees

A large tree can be difficult to navigate and sometimes you will want to hide parts of it, or break it down into a forest of smaller trees. If you hover the mouse over the point where a node link enters or leaves the node you will see one or more popup buttons which allow you to do this.

Click on the minus button to collapse the attached sub-tree.

Click on the plus button to expand a collapsed tree. When a tree has been collapsed, this button remains visible to indicate the hidden sub-tree.

Click on the cross button to split the attached sub-tree and show it as a separate tree on the graph.

Click on the circle button to reattach a split tree. When a tree has been split away, this button remains visible to indicate that there is a separate sub-tree.

Click on the graph background for the main context menu, which offers options to Expand all and Join all. If no branch has been collapsed or split, the context menu will not be shown.

4.26. □导出一个Subversion工作副本

有时候你需要一个没有.svn目录的工作目录树，例如，创建一份源代码的压缩文件，或者导出一份用作WEB服务器。不用先复制，然后手工删除所有.svn目录。TortoiseSVN提供命令TortoiseSVN → 导出...。如果你要用这个功能操作拥有一份工作副本，将会在要求你保存一份干净的文件。从URL或工作副本导出有少许不同。

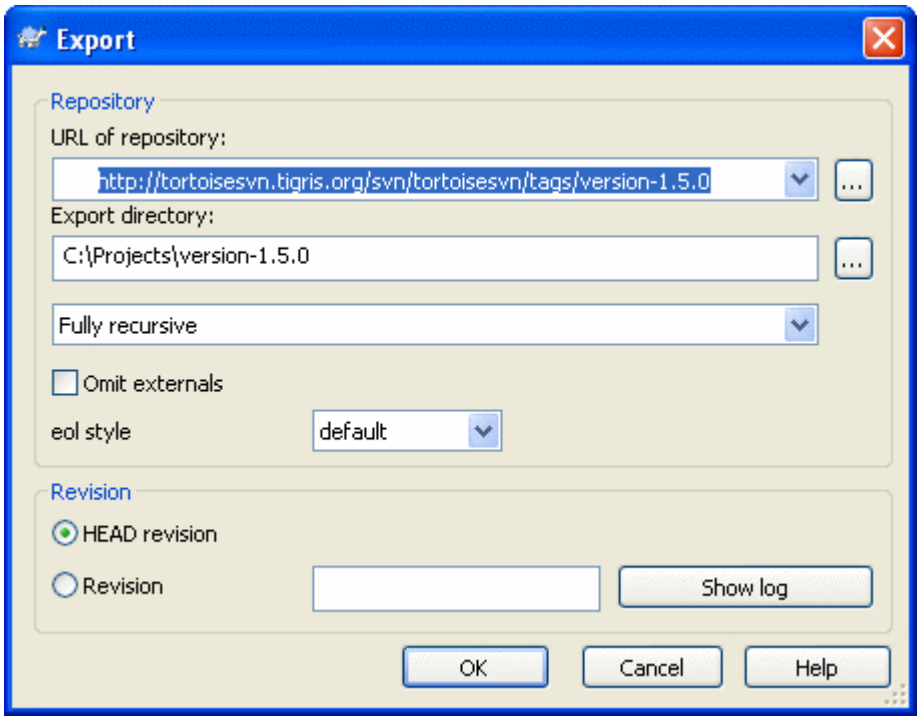


图 4.48. 从 URL 导出对话框

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。

如果你要用这个功能操作工作副本，将会询问你要保存干净而没有.svn目录的副本到何处。默认情况下，只导出被版本控制的文件，但你可以使用同时导出未受版本控制的文件来将版本库中没有但在你的本地副本中存在的文件导出来。另外可以使用svn:externals来忽略外部引用。

另外一个导出的方法是 右键拖工作副本到另外的保存位置，然后选择右键菜单 → SVN导出到这儿或者右键菜单 → SVN 导出全部到这儿。后者可以把未受版本控制的文件也导出。

导出工作副本时，如果目标目录包含了和你导出的名称相同的目录，你需要使用此选项重写已经存在的内容，或者使用自动生成的名称，例如目标 (1)。

Exporting single files

The export dialog does not allow exporting single files, even though Subversion can.

To export single files with TortoiseSVN, you have to use the repository browser (第 4.24 节 “版本库浏览器”). Simply drag the file(s) you want to export from the repository browser to where you want them in the explorer, or use the context menu in the repository browser to export the files.

Exporting a Change Tree

If you want to export a copy of your project tree structure but containing only the files which have changed in a particular revision, or between any

two revisions, use the compare revisions feature described in 第 4.10.3 节 “比较文件夹”.

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 directories. What you really need is an export-in-place command, that just removes the control directories rather than generating a new clean directory tree.

The answer is surprisingly simple - export the folder to itself! TortoiseSVN detects this special case and asks if you want to make the working copy unversioned. If you answer yes the control directories will be removed and you will have a plain, unversioned directory tree.

4.27. □重新定位工作副本

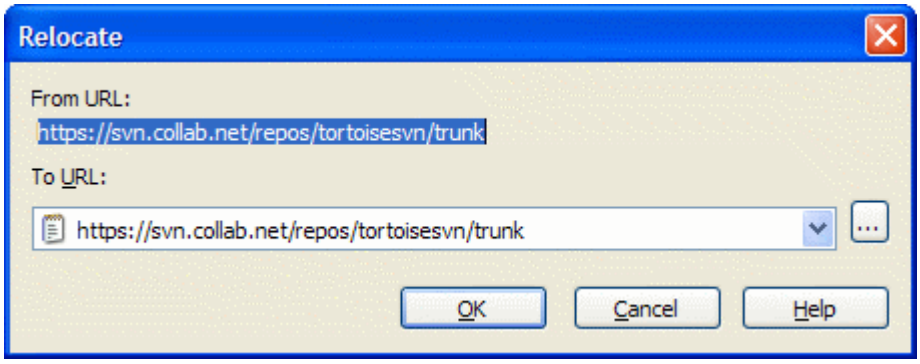


图 4.49. 重定位对话框

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 scans all entries files in the .svn folder and changes the URL of the entries to the new value.

You may be surprised to find that TortoiseSVN contacts the repository as part of this operation. All it is doing is performing some simple checks to make sure that the new URL really does refer to the same repository as the existing working copy.

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. □Adding Issue Numbers to Log Messages

You can integrate a bug tracking tool of your choice in TortoiseSVN. To do this, you have to define some properties, which start with bugtraq:. They must be set on Folders: (第 4.17 节 “项目设置”)

There are two ways to integrate TortoiseSVN with issue trackers. One is based on simple strings, the other is based on regular expressions. The properties used by both approaches are:

bugtraq:url
: Set this property to the URL of your bug tracking tool. It must be properly URI encoded and it has to contain %BUGID%. %BUGID% is replaced with the Issue number you entered. This allows TortoiseSVN to display a link in the log dialog, so when you are looking at the revision log you can jump directly to your bug tracking tool. You do not have to provide this property, but then TortoiseSVN shows only the issue number and not the link to it. e.g the TortoiseSVN project is using http://issues.tortoisesvn.net/?do=details&id=%BUGID%

  You can also use relative URLs instead of absolute ones. This is useful when your issue tracker is on the same domain/server as your source repository. In case the domain name ever changes, you don't have to adjust the bugtraq:url property. There are two ways to specify a relative URL:

  If it begins with the string ^/ it is assumed to be relative to the repository root. For example, ^/../?do=details&id=%BUGID% will resolve to http://tortoisesvn.net/?do=details&id=%BUGID% if your repository is located on http://tortoisesvn.net/svn/trunk/.

  A URL beginning with the string / is assumed to be relative to the server's hostname. For example /?do=details&id=%BUGID% will resolve to http://tortoisesvn.net/?do=details&id=%BUGID% if your repository is located anywhere on http://tortoisesvn.net.

bugtraq:warnifnoissue
Set this to true, if you want TortoiseSVN to warn you because of an empty issue-number text field. Valid values are true/false. If not defined, false is assumed.

4.28.1.1. □Issue Number in Text Box

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
This property activates the bug tracking system in Input field mode. If this property is set, then TortoiseSVN will prompt you to enter an issue number when you commit your changes. It's used to add a line at the end of the log message. It must contain %BUGID%, which is replaced with the issue number on commit. This ensures that your commit log contains a reference to the issue number which is always in a consistent format and can be parsed by your bug tracking tool to associate the issue number with a particular commit. As an example you might use Issue : %BUGID%, but this depends on your Tool.

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。

bugtraq:label
This text is shown by TortoiseSVN on the commit dialog to label the edit box where you enter the issue number. If it's not set, Bug-ID / Issue-Nr: will be displayed. Keep in mind though that the window will not be resized to fit this label, so keep the size of the label below 20-25 characters.

bugtraq:number
If set to true only numbers are allowed in the issue-number text field. An exception is the comma, so you can comma separate several numbers. Valid values are true/false. If not defined, true is assumed.

4.28.1.2. □Issue Numbers Using Regular Expressions

In the approach with regular expressions, TortoiseSVN doesn't show a separate input field but marks the part of the log message the user enters which is recognized by the issue tracker. This is done while the user writes the log message. This also means that the bug ID can be anywhere inside a log message! This method is much more flexible, and is the one used by the TortoiseSVN project itself.

bugtraq:logregex
This property activates the bug tracking system in Regex mode. It contains either a single regular expressions, or two regular expressions separated by a newline.

If two expressions are set, then the first expression is used as a pre-filter to find expressions which contain bug IDs. The second expression then extracts the bare bug IDs from the result of the first regex. This allows you to use a list of bug IDs and natural language expressions if you wish. e.g. you might fix several bugs and include a string something like this: “This change resolves issues #23, #24 and #25”

If you want to catch bug IDs as used in the expression above inside a log message, you could use the following regex strings, which are the ones used by the TortoiseSVN project: [Ii]ssues?:?(\s*(,|and)?\s*#\d+)+ and (\d+)

The first expression picks out “issues #23, #24 and #25” from the surrounding log message. The second regex extracts plain decimal numbers from the output of the first regex, so it will return “23”, “24” and “25” to use as bug IDs.

Breaking the first regex down a little, it must start with the word “issue”, possibly capitalised. This is optionally followed by an “s” (more than one issue) and optionally a colon. This is followed by one or more groups each having zero or more leading whitespace, an optional comma or “and” and more optional space. Finally there is a mandatory “#” and a mandatory decimal number.

If only one expression is set, then the bare bug IDs must be matched in the groups of the regex string. Example: [Ii]ssue(?:s)? #?(\d+) This method is required by a few issue trackers, e.g. trac, but it is harder to construct the regex. We recommend that you only use this method if your issue tracker documentation tells you to.

If you are unfamiliar with regular expressions, take a look at the introduction at http://en.wikipedia.org/wiki/Regular_expression, and the online documentation and tutorial at http://www.regular-expressions.info/.

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！

And even if you don't need the links, the issue numbers show up as a separate column in the log dialog, making it easier to find the changes which relate to a particular issue.

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

These properties must be set on folders for the system to work. When you commit a file or folder the properties are read from that folder. If the properties are not found there, TortoiseSVN will search upwards through the folder tree to find them until it comes to an unversioned folder, or the tree root (eg. C:\) is found. If you can be sure that each user checks out only from e.g trunk/ and not some sub-folder, then it's enough if you set the properties on trunk/. If you can't be sure, you should set the properties recursively on each sub-folder. A property setting deeper in the project hierarchy overrides settings on higher levels (closer to trunk/).

For tsvn: properties only you can use the Recursive checkbox to set the property to all sub-folders in the hierarchy, without also setting it on all files.

No Issue Tracker Information from Repository Browser

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.

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/issuetrackers.txt] in the TortoiseSVN source repository. (第 3 节 “TortoiseSVN 是完全免费的！” explains how to access the repository).

4.28.2. Getting Information from the Issue Tracker

The previous section deals with adding issue information to the log messages. But what if you need to get information from the issue tracker? The commit dialog has a COM interface which allows integration an external program that can talk to your tracker. Typically you might want to query the tracker to get a list of open issues assigned to you, so that you can pick the issues that are being addressed in this commit.

Any such interface is of course highly specific to your issue tracker system, so we cannot provide this part, and describing how to create such a program is beyond the scope of this manual. The interface definition and sample plugins in C# and C++/ATL can be obtained from the contrib folder in the TortoiseSVN repository [http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins]. (第 3 节 “TortoiseSVN 是完全免费的！”explains how to access the repository). A summary of the API is also given in 第 6 章BugtraqProvider interface Another (working) example plugin in C# is Gurtle [http://code.google.com/p/gurtle/] which implements the required COM interface to interact with the Google Code [http://code.google.com/hosting/] issue tracker.

For illustration purposes, let's suppose that your system administrator has provided you with an issue tracker plugin which you have installed, and that you have set up some of your working copies to use the plugin in TortoiseSVN's settings dialog. When you open the commit dialog from a working copy to which the plugin has been assigned, you will see a new button at the top of the dialog.

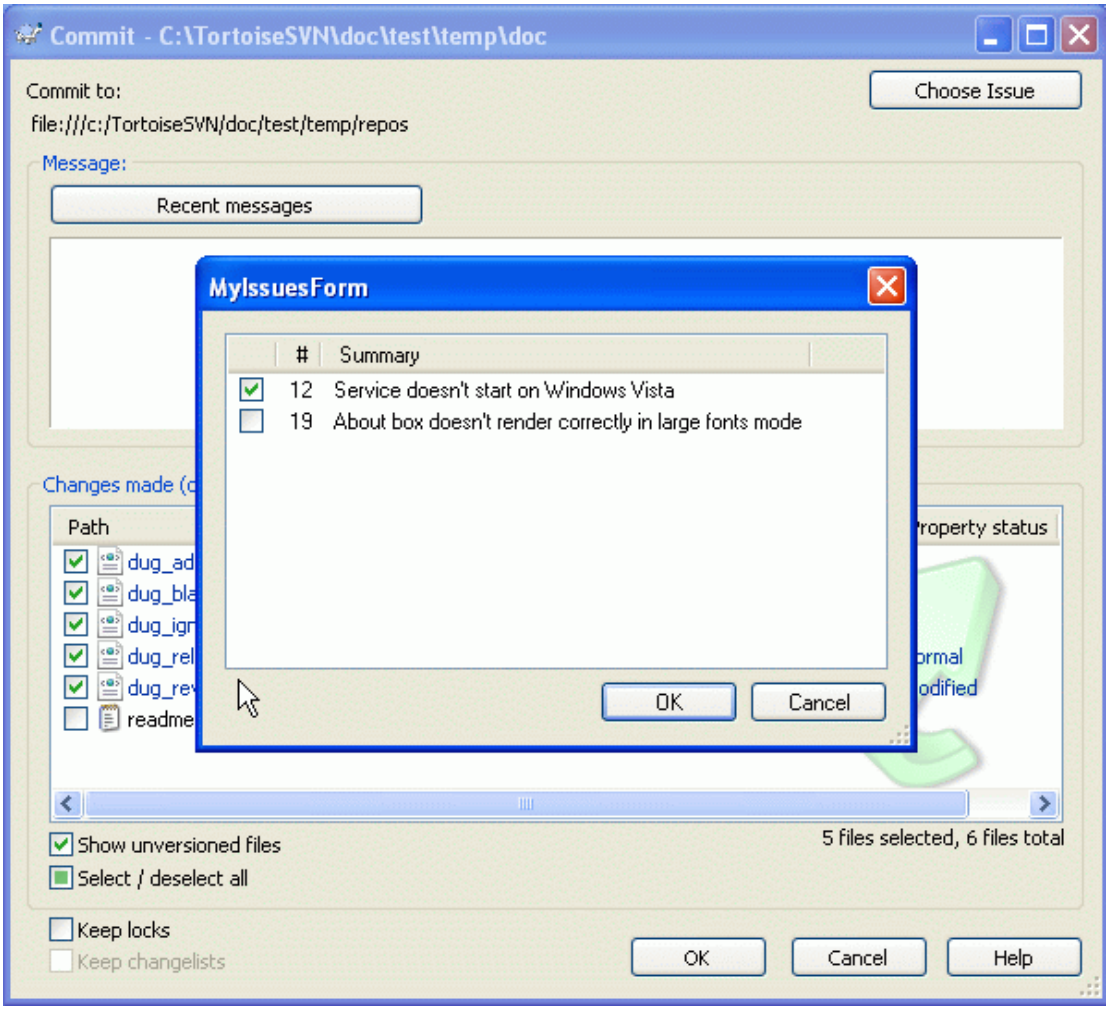


图 4.50. Example issue tracker query dialog

In this example you can select one or more open issues. The plugin can then generate specially formatted text which it adds to your log message.

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。

You can integrate a repo viewer of your choice in TortoiseSVN. To do this, you have to define some properties which define the linkage. They must be set on Folders: (第 4.17 节 “项目设置”)

webviewer:revision
: Set this property to the URL of your repo viewer to view all changes in a specific revision. It must be properly URI encoded and it has to contain %REVISION%. %REVISION% is replaced with the revision number in question. This allows TortoiseSVN to display a context menu entry in the log dialog Context Menu → View revision in webviewer

webviewer:pathrevision
: Set this property to the URL of your repo viewer to view changes to a specific file in a specific revision. It must be properly URI encoded and it has to contain %REVISION% and %PATH%. %PATH% is replaced with the path relative to the repository root. This allows TortoiseSVN to display a context menu entry in the log dialog Context Menu → View revision and path in webviewer For example, if you right-click in the log dialog bottom pane on a file entry /trunk/src/file then the %PATH% in the URL will be replaced with /trunk/src/file.

You can also use relative URLs instead of absolute ones. This is useful in case your web viewer is on the same domain/server as your source repository. In case the domain name ever changes, you don't have to adjust the webviewer:revision and webviewer:pathrevision property. The format is the same as for the bugtraq:url property. See 第 4.28 节 “与 BUG 跟踪系统/问题跟踪集成”.

设置文件夹的属性

These properties must be set on folders for the system to work. When you commit a file or folder the properties are read from that folder. If the properties are not found there, TortoiseSVN will search upwards through the folder tree to find them until it comes to an unversioned folder, or the tree root (eg. C:\) is found. If you can be sure that each user checks out only from e.g trunk/ and not some sub-folder, then it's enough if you set the properties on trunk/. If you can't be sure, you should set the properties recursively on each sub-folder. A property setting deeper in the project hierarchy overrides settings on higher levels (closer to trunk/).

For tsvn: properties only you can use the Recursive checkbox to set the property to all sub-folders in the hierarchy, without also setting it on all files.

No Repo Viewer Links from Repository Browser

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.

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

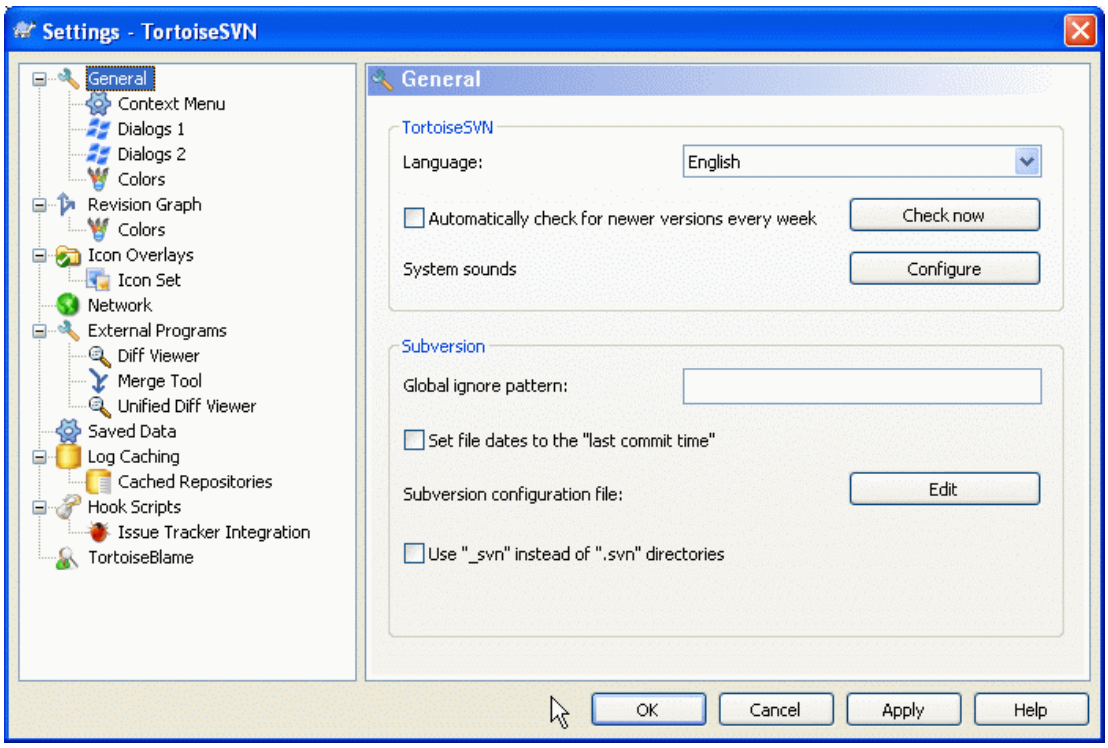


图 4.51. 设置对话框，常规设置页面

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？

每周自动检查新版本
如果检查过，TSVN将每周联系它的下载站点一次，来看看程序是否有个可用的新版本。若你想马上得到结果，使用 立即检查 按钮。无论如何，新版本不会被自动下载，而只是你将收到一条提示信息对话框，告诉你有新版本可用。

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。

全局忽略样式

Global ignore patterns are used to prevent unversioned files from showing up e.g. in the commit dialog. Files matching the patterns are also ignored by an import. Ignore files or directories by typing in the names or extensions. Patterns are separated by spaces e.g. bin obj *.bak *.~?? *.jar *.[Tt]mp. These patterns should not include any path separators. Note also that there is no way to differentiate between files and directories. Read 第 4.13.1 节 “忽略列表中的模式匹配”for more information on the pattern-matching syntax.

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。

忽略样式将作用于你所有的项目工程。因为它是非版本控制的，所以它将不会对其他的用户起作用。相对而言，你也可以使用可版本控制的 svn:ignore 属性来把要忽略的文件或文件夹排斥在版本控制之外。阅读 第 4.13 节 “忽略文件和目录”以获得更多信息。

Set file dates to the “last commit time”

This option tells TortoiseSVN to set the file dates to the last commit time when doing a checkout or an update. Otherwise TortoiseSVN will use the current date. If you are developing software it is generally best to use the current date because build systems normally look at the date stamps to decide which files need compiling. If you use “last commit time” and revert to an older file revision, your project may not compile as you expect it to.

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/1.5/svn.advanced.confarea.html]. The section on Automatic Property Setting [http://svnbook.red-bean.com/en/1.5/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/1.5/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] to find out more.

Use _svn instead of .svn directories

在使用VS.NET环境做web工程时，将无法处理 .svn 文件夹，但Subversion是要用这些文件夹来储存自己的内部信息的。这可不是Subversion的bug，这bug是VS.NET和它使用的frontpage扩展带来的。阅读 第 4.30.11 节 “Subversion 的工作文件夹”来获得有关此问题的更多信息。

若你想改变Subversion和TSVN的这些行为，就可以使用这个选项框来设置控制这些的环境变量。

你应该注意到： 改变该选项将不会使已存在的工作副本中的管理文件夹从“_svn”自动转换到“.svn”。你需要使用一个脚本(查看我们的FAQ)来自行完成这项工作，或是简单地重新检出一个新的工作副本。

4.30.1.1. □右键菜单配置

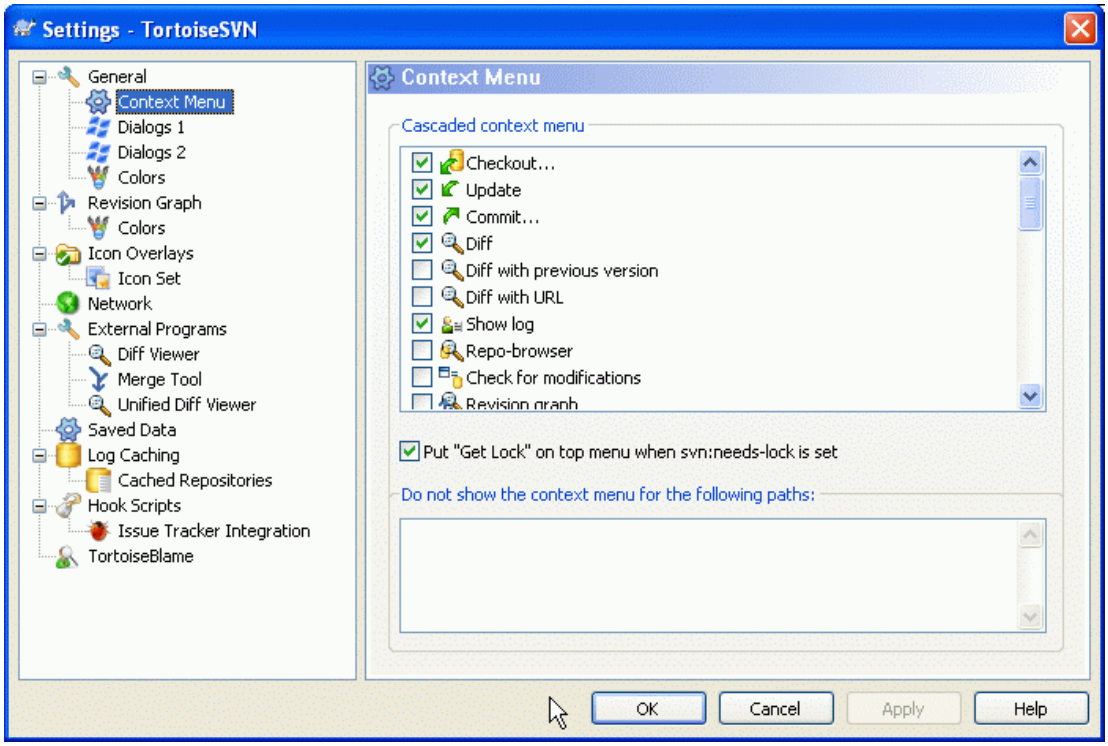


图 4.52. 设置对话框，右键菜单页面

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。

If there are some paths on your computer where you just don't want TortoiseSVN's context menu to appear at all, you can list them in the box at the bottom.

4.30.1.2. □TSVN对话框设置一

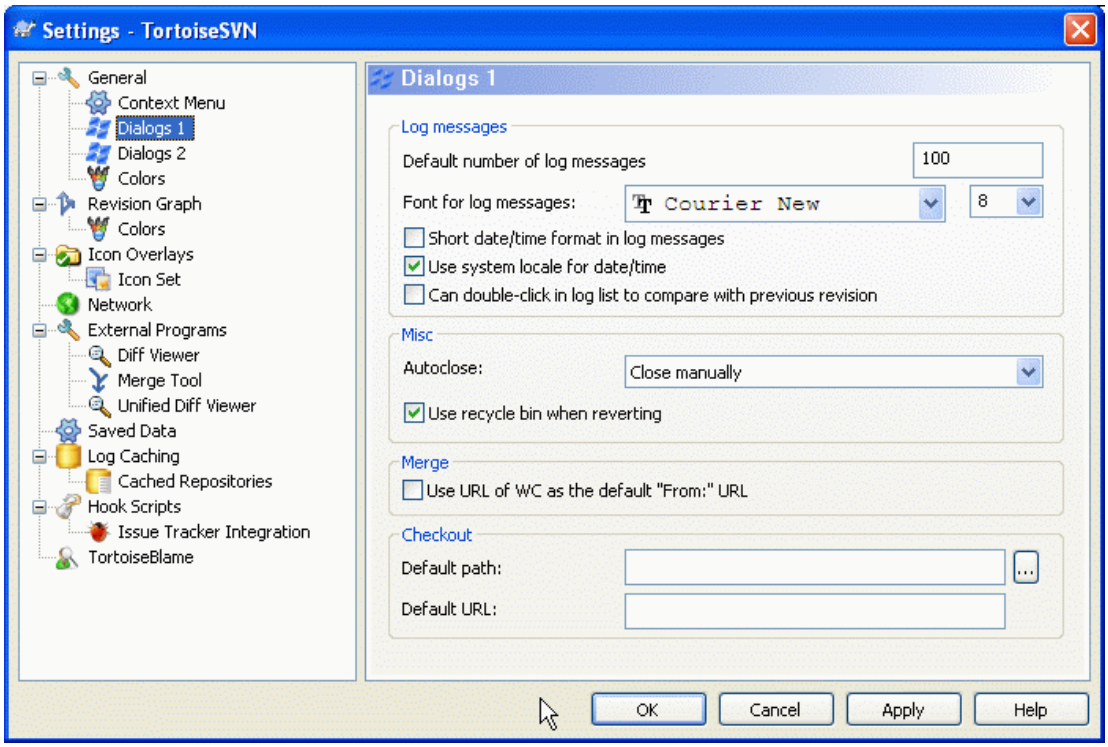


图 4.53. 设置对话框，对话框一页面

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
: Limits the number of log messages that TortoiseSVN fetches when you first select TortoiseSVN → Show Log Useful for slow server connections. You can always use Show All or Next 100 to get more messages.

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。

Can double-click in log list to compare with previous revision
: If you frequently find yourself comparing revisions in the top pane of the log dialog, you can use this option to allow that action on double-click. It is not enabled by default because fetching the diff is often a long process, and many people prefer to avoid the wait after an accidental double-click, which is why this option is not enabled by default.

进程对话框
: 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。

对本地操作自动关闭(如无合并、添加或删除操作，自动关闭) 意味着进程对话框当 如无合并、添加或删除操作 时自动关闭，但仅限于那些如添加文件、还原等本地的操作。在做远程操作时对话框将保持打开。

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。

Use recycle bin when reverting
When you revert local modifications, your changes are discarded. TortoiseSVN gives you an extra safety net by sending the modified file to the recycle bin before bringing back the pristine copy. If you prefer to skip the recycle bin, uncheck this option.

Use URL of WC as the default “From:” 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

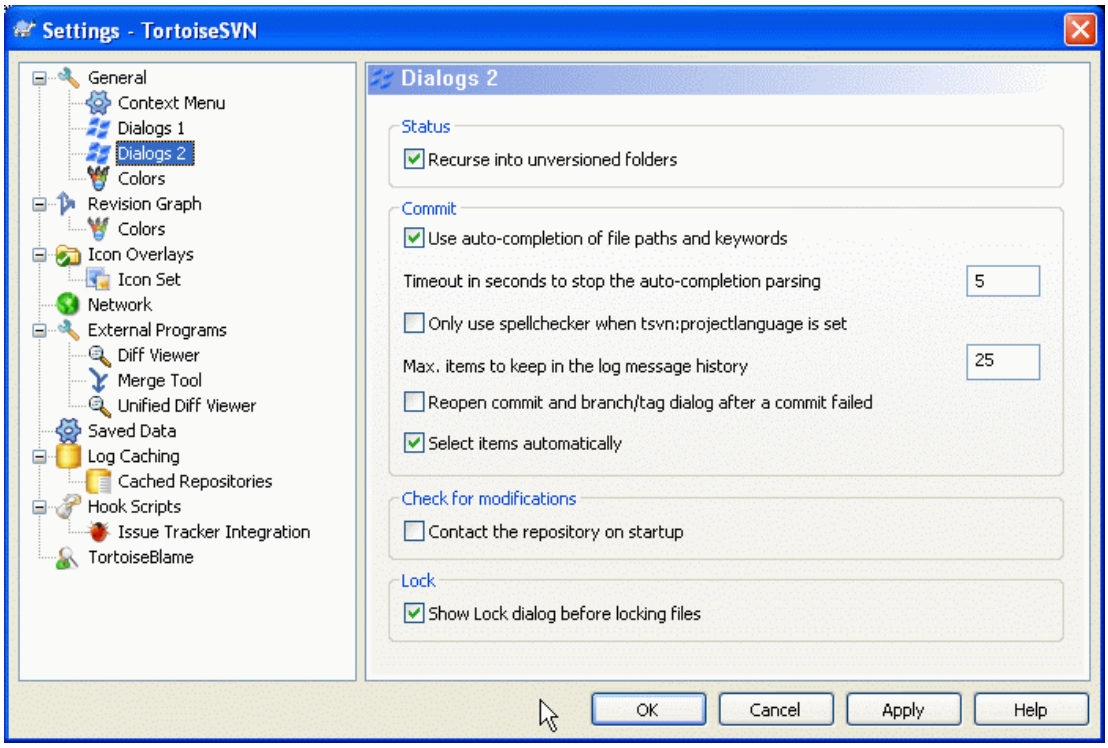


图 4.54. 设置对话框，对话框二页面

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。

自动完成文件路径和关键词
The commit dialog includes a facility to parse the list of filenames being committed. When you type the first 3 letters of an item in the list, the auto-completion box pops up, and you can press Enter to complete the filename. Check the box to enable this feature.

自动完成分析的超时时间(秒)
The auto-completion parser can be quite slow if there are a lot of large files to check. This timeout stops the commit dialog being held up for too long. If you are missing important auto-completion information, you can extend the timeout.

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
When you type in a log message in the commit dialog, TortoiseSVN stores it for possible re-use later. By default it will keep the last 25 log messages for each repository, but you can customize that number here. If you have many different repositories, you may wish to reduce this to avoid filling your registry.

Note that this setting applies only to messages that you type in on this computer. It has nothing to do with the log cache.

如果提交失败，自动重新打开提交和分支/标签对话框
当一个提交操作由于某些原因(工作副本需要更新、pre-commit钩子程序拒绝了提交、网络错误等等)失败了，你可以选择该选项来使提交对话框保持打开，以便重新操作。当然，你应该注意到这可能会导致一些问题。若发生的错误意味着你需要更新你的工作副本，而此更新操作将导致冲突，那么你必须先解决这些事情再说。

自动选择项目
The normal behaviour in the commit dialog is for all modified (versioned) items to be selected for commit automatically. If you prefer to start with nothing selected and pick the items for commit manually, uncheck this box.

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 颜色设置

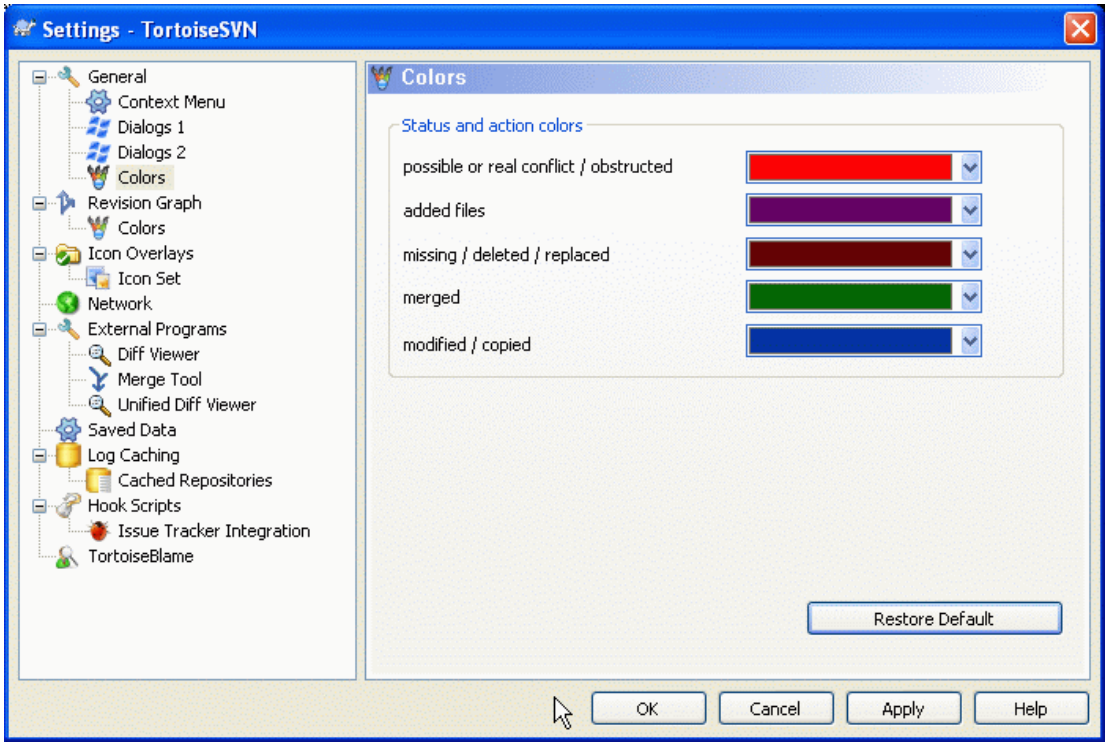


图 4.55. 设置对话框，颜色页面

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。

4.30.2. □Revision Graph Settings

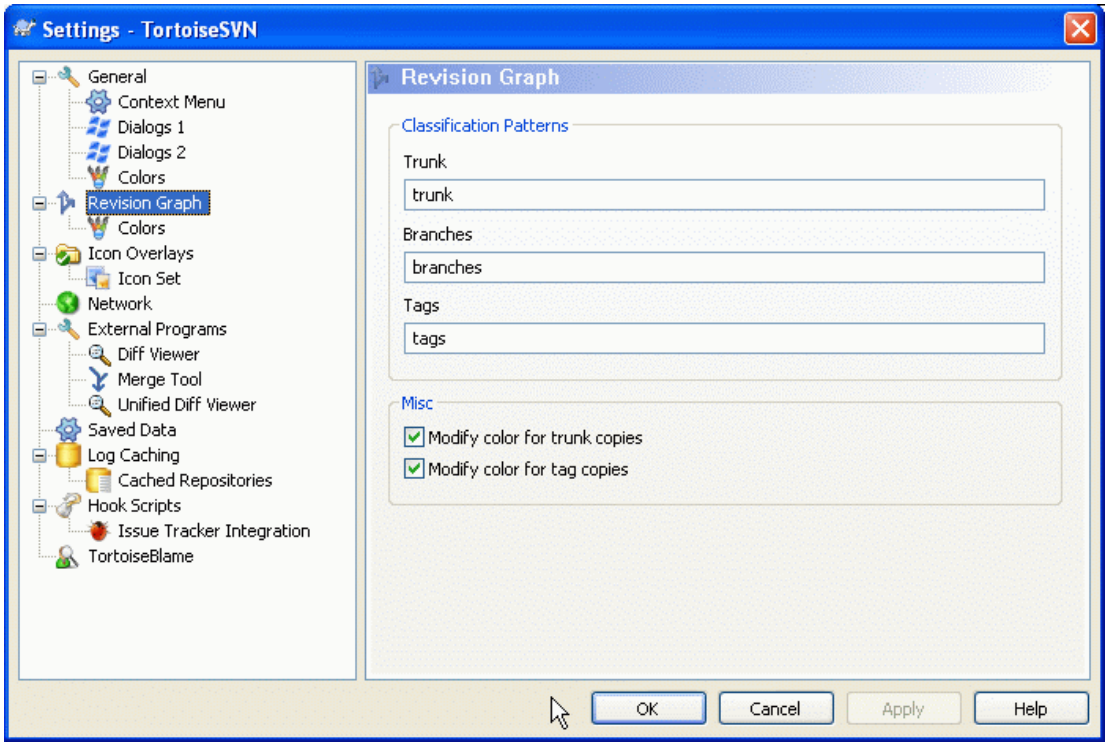


图 4.56. The Settings Dialog, Revision Graph Page

Classification Patterns
The revision graph attempts to show a clearer picture of your repository structure by distinguishing between trunk, branches and tags. As there is no such classification built into Subversion, this information is extracted from the path names. The default settings assume that you use the conventional English names as suggested in the Subversion documentation, but of course your usage may vary.

Specify the patterns used to recognise these paths in the three boxes provided. The patterns will be matched case-insensitively, but you must specify them in lower case. Wild cards * and ? will work as usual, and you can use ; to separate multiple patterns. Do not include any extra white space as it will be included in the matching specification.

Modify Colors
Colors are used in the revision graph to indicate the node type, i.e. whether a node is added, deleted, renamed. In order to help pick out node classifications, you can allow the revision graph to blend colors to give an indication of both node type and classification. If the box is checked, blending is used. If the box is unchecked, color is used to indicate node type only. Use the color selection dialog to allocate the specific colors used.

4.30.2.1. Revision Graph Colors

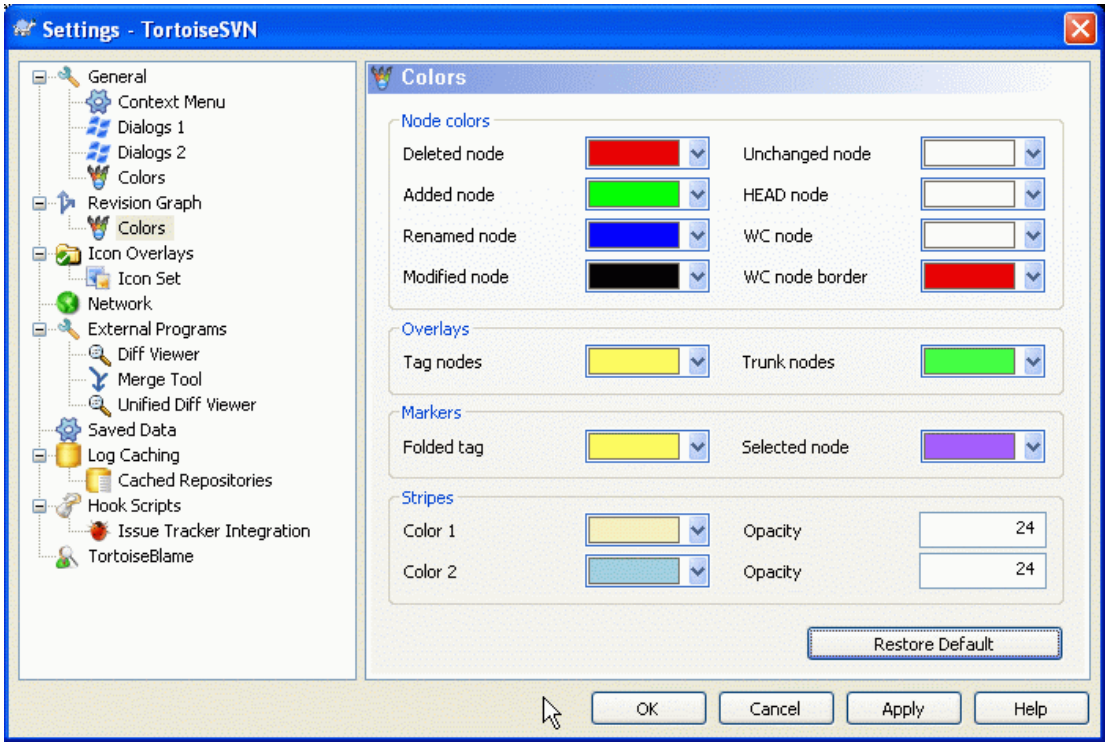


图 4.57. The Settings Dialog, Revision Graph Colors Page

This page allows you to configure the colors used. Note that the color specified here is the solid color. Most nodes are colored using a blend of the node type color, the background color and optionally the classification color.

Deleted Node
: Items which have been deleted and not copied anywhere else in the same revision.

Added Node
: Items newly added, or copied (add with history).

Renamed Node
: Items deleted from one location and added in another in the same revision.

Modified Node
: Simple modifications without any add or delete.

Unchanged Node
: May be used to show the revision used as the source of a copy, even when no change (to the item being graphed) took place in that revision.

HEAD node
: Current HEAD revision in the repository.

WC Node
: If you opt to show an extra node for your modified working copy, attached to its last-commit revision on the graph, use this color.

WC Node Border
: If you opt to show whether the working copy is modified, use this color border on the WC node when modifications are found.

Tag Nodes
    Nodes classified as tags may be blended with this color.

Trunk Nodes
    Nodes classified as trunk may be blended with this color.

Folded Tag Markers
    If you use tag folding to save space, tags are marked on the copy source using a block in this color.

Selected Node Markers
    When you left click on a node to select it, the marker used to indicate selection is a block in this color.

Stripes
    These colors are used when the graph is split into sub-trees and the background is colored in alternating stripes to help pick out the separate trees.

4.30.3. □图标叠加设置

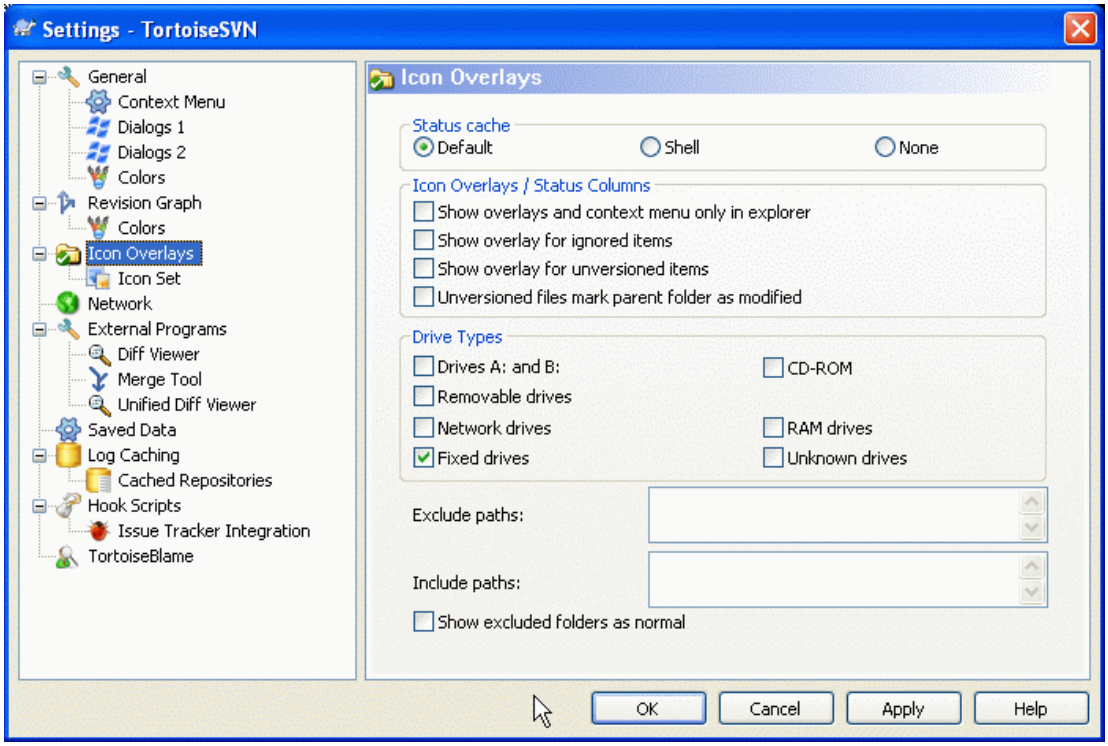


图 4.58. The Settings Dialog, Icon Overlays Page

This page allows you to choose the items for which TortoiseSVN will display icon overlays.

By default, overlay icons and context menus will appear in all open/save dialogs as well as in Windows Explorer. If you want them to appear only in Windows Explorer, check the Show overlays and context menu only in explorer box.

Ignored items and Unversioned items are not usually given an overlay. If you want to show an overlay in these cases, just check the boxes.

You can also choose to mark folders as modified if they contain unversioned items. This could be useful for reminding you that you have created new files which are

not yet versioned. This option is only available when you use the default status cache option (see below).

Since it takes quite a while to fetch the status of a working copy, TortoiseSVN uses a cache to store the status so the explorer doesn't get hogged too much when showing the overlays. You can choose which type of cache TortoiseSVN should use according to your system and working copy size here:

默认
: Caches all status information in a separate process (TSVNCache.exe). That process watches all drives for changes and fetches the status again if files inside a working copy get modified. The process runs with the least possible priority so other programs don't get hogged because of it. That also means that the status information is not real time but it can take a few seconds for the overlays to change.

  Advantage: the overlays show the status recursively, i.e. if a file deep inside a working copy is modified, all folders up to the working copy root will also show the modified overlay. And since the process can send notifications to the shell, the overlays on the left tree view usually change too.

  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。

  Disadvantage: Status information of files and folders is not shown in Explorer. To see if your working copies are modified, you have to use the “Check for modifications” dialog.

The next group allows you to select which classes of storage should show overlays. By default, only hard drives are selected. You can even disable all icon overlays, but where's the fun in that?

Network drives can be very slow, so by default icons are not shown for working copies located on network shares.

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 是被用来告诉TSVN 不用 在哪些路径下显示图标重载和状态列。如果你有些很大的工作副本，而这些工作副本仅仅包含你完全不想改变的库文件，从而你也不需要显示图标重载，这时该功能将会很有用。举个例子:

填写 f:\development\SVN\Subversion 将 仅仅 在这个特殊文件夹上取消图标覆盖。 你仍然可以在该路径下的所有文件、文件夹上看到图标重载。

填写 f:\development\SVN\Subversion* 将在路径以 f:\development\SVN\Subversion 开始的 所有 文件和文件夹上取消图标重载。这意味着你在该路径下的任何文件/文件夹上都将看不到图标重载了。

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。

Users sometimes ask how these three settings interact, and the definitive answer is:

```
if (path is in include list)
  show overlays
if (path is allowed drive type) AND (path is not in exclude list)
  show overlays
```

The include list always makes the overlays show. Otherwise, overlays are shown for all marked drive types unless the path is excluded.

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

It is often convenient to use a SUBST drive to access your working copies, e.g. using the command

```
subst T: C:\TortoiseSVN\trunk\doc
```

However this can cause the overlays not to update, as TSVNCache will only receive one notification when a file changes, and that is normally for the original path. This means that your overlays on the subst path may never be updated.

An easy way to work around this is to exclude the original path from showing overlays, so that the overlays show up on the subst path instead.

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 such folders are versioned. The Show excluded folders as 'normal' checkbox allows you to do this. With this option, versioned 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.

As a special exception to this, drives A: and B: are never considered for the Show excluded folders as 'normal' option. This is because Windows is forced to look on the drive, which can result in a delay of several seconds when starting Explorer, even if your PC does have a floppy drive.

4.30.3.1. □图标集选择

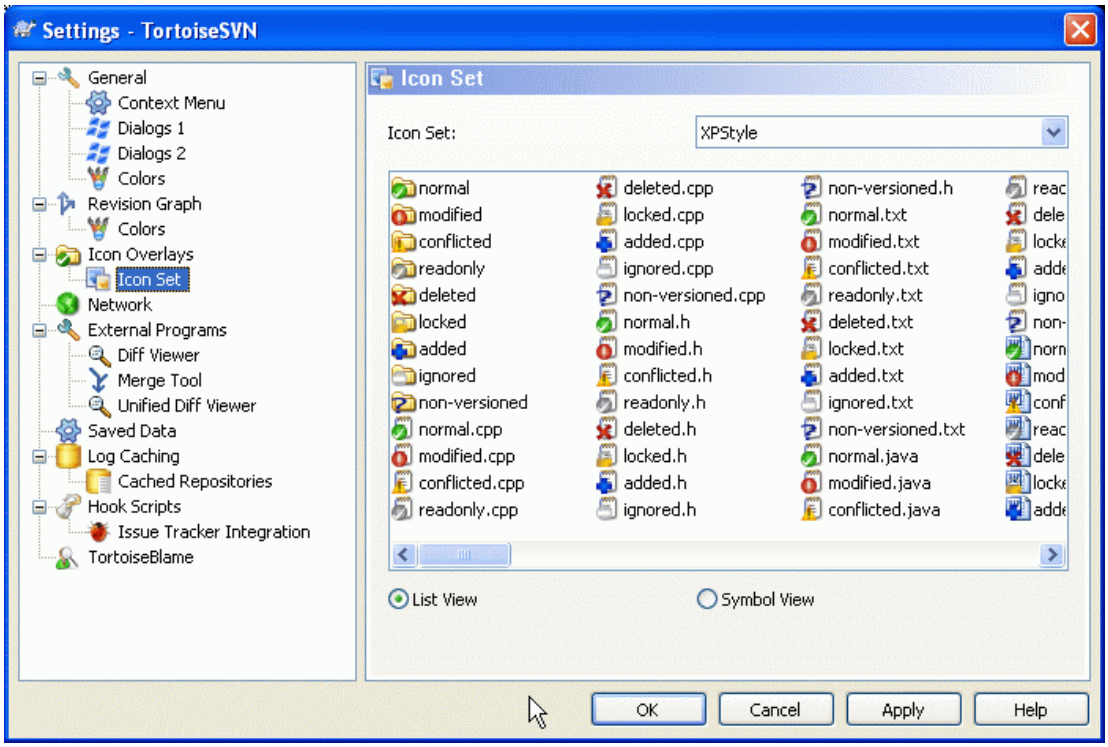


图 4.59. 设置对话框，图标集页面

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。

4.30.4. □网络设置

<placeholder-1> 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。</placeholder-1>

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/1.5/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。

You must specify the full path to the executable. For TortoisePlink.exe this is the standard TortoiseSVN bin directory. Use the Browse button to help locate it. Note that if the path contains spaces, you must enclose it in quotes, e.g.

"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 does not have any documentation of its own because it is just a minor variant of Plink. Find out about command line parameters from the PuTTY website [http://www.chiark.greenend.org.uk/~sgtatham/putty/]

To avoid being prompted for a password repeatedly, you might also consider using a password caching tool such as Pageant. This is also available for download from the PuTTY website.

Finally, setting up SSH on server and clients is a non-trivial process which is beyond the scope of this help file. However, you can find a guide in the TortoiseSVN FAQ listed under Subversion/TortoiseSVN SSH How-To [http://tortoisesvn.net/ssh_howto].

4.30.5. □外部程序设置

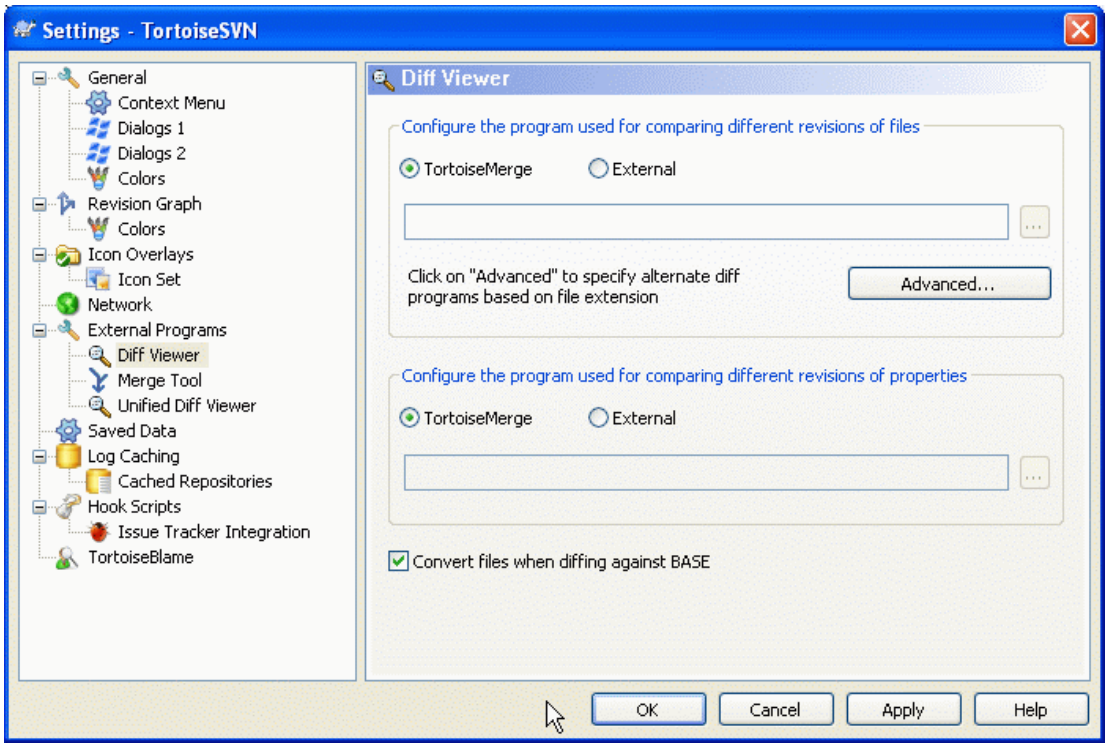


图 4.60. 设置对话框，差异查看页面

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 4.10.5 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

窗口标题并不一定代表真正的文件名。TortoiseSVN把它伪装成一个名字用来创建和显示。因此，倘若你在对比一个版本为123的文件和你当前工作副本中的文件，名字将显示为 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
```

If you use the svn:keywords property to expand keywords, and in particular the revision of a file, then there may be a difference between files which is purely due to the current value of the keyword. Also if you use svn:eol-style = native the BASE file will have pure LF line endings whereas your file will have CR-LF line endings. TortoiseSVN will normally hide these differences automatically by first parsing the BASE file to expand keywords and line endings before doing the diff operation. However, this can take a long time with large files. If Convert files when diffing against BASE is unchecked then TortoiseSVN will skip pre-processing the files.

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。

If you have configured an alternate diff tool, you can access TortoiseMerge and the third party tool from the context menus. Context menu → Diff uses the primary diff tool, and Shift+ Context menu → Diff uses the secondary diff tool.

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
```

4.30.5.3. □差异查看/合并工具的高级设置

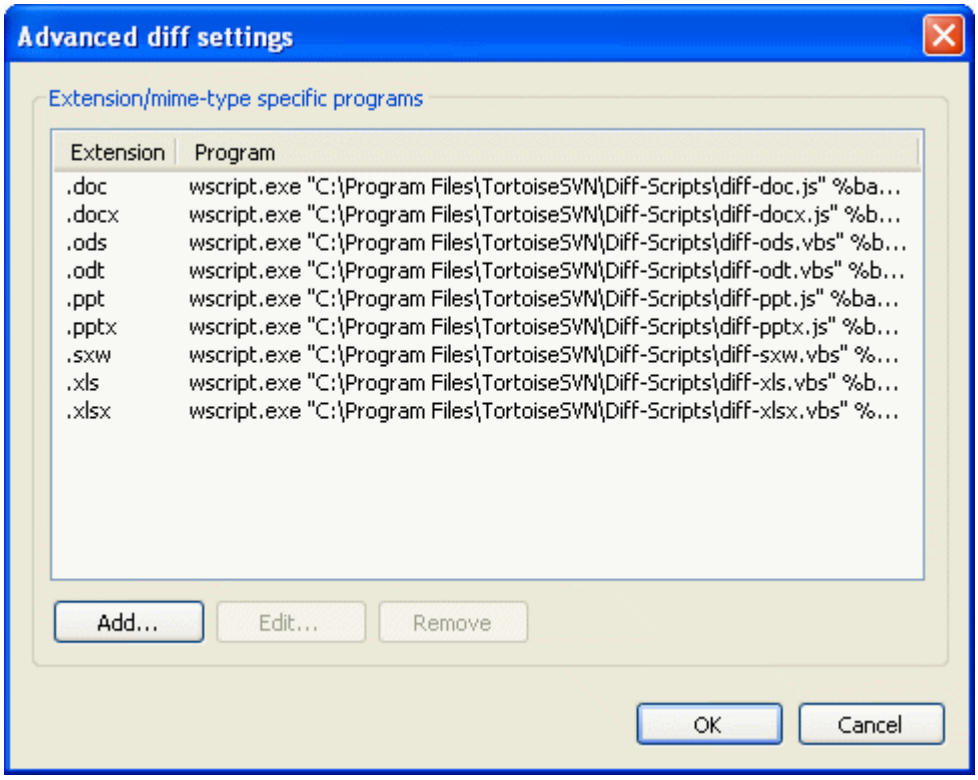


图　4.61.　高级差异比较设置/高级合并设置的对话框

In the advanced settings, you can define a different diff and merge program for every file extension. For instance you could associate Photoshop as the "Diff" Program for .jpg files :-) You can also associate the svn:mime-type property with a diff or merge program.

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。

4.30.5.4. □统一的差异查看器

一个统一差异文件(补丁文件)的查看程序。不需要任何参数。默认 选项遵循先检查 .diff 文件，再检查 .txt 文件的顺序。如果你没有 .diff 文件的查看器，就需要用记事本来查看了。

原始Windows记事本程序对未使用标准"回车-换行"结束符的文件支持的并不好。而很多统一差异文件都仅仅使用"换行"结束符，因此他们的格式在记事本中显示的并不好。无论如何，你可以下载一个免费的记事本2 Notepad2 [http://www.flos-freeware.ch/notepad2.html]，它不但可以正确地显示结束符，更可以为差异文件中添加和删除的那些行做颜色标记。

4.30.6. □已保存数据的设置

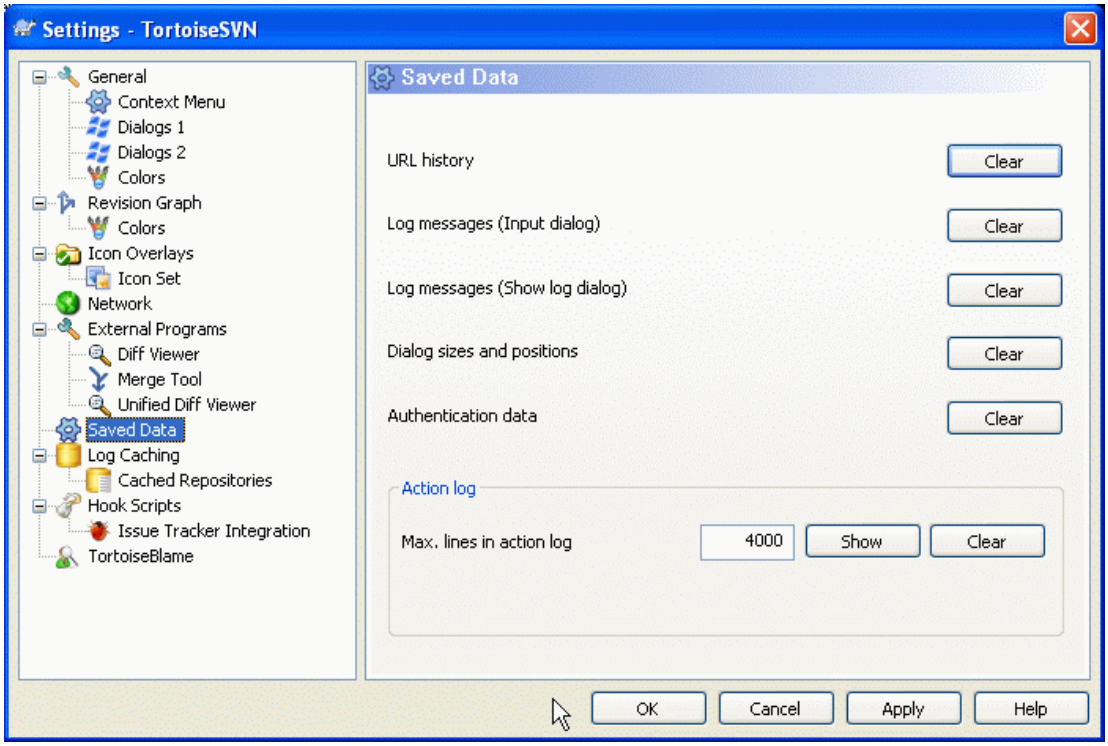


图 4.62. 设置对话框，已保存数据设置页面

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。

If you want to remove a single item from one of the combo boxes you can do that in-place. Just click on the arrow to drop the combo box down, move the mouse over the item you want to remove and type 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 caches log messages fetched by the Show Log dialog to save time when you next show the log. If someone else edits a log message and you already have that message cached, you will not see the change until you clear the cache. Log message caching is enabled on the Log Cache tab.

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, read 第 4.1.5 节 “认证”for instructions on how to find the cached data.

动作日志
TortoiseSVN keeps a log of everything written to its progress dialogs. This can be useful when, for example, you want to check what happened in a recent update command.

The log file is limited in length and when it grows too big the oldest content is discarded. By default 4000 lines are kept, but you can customize that number.

From here you can view the log file content, and also clear it.

4.30.7. □日志缓存

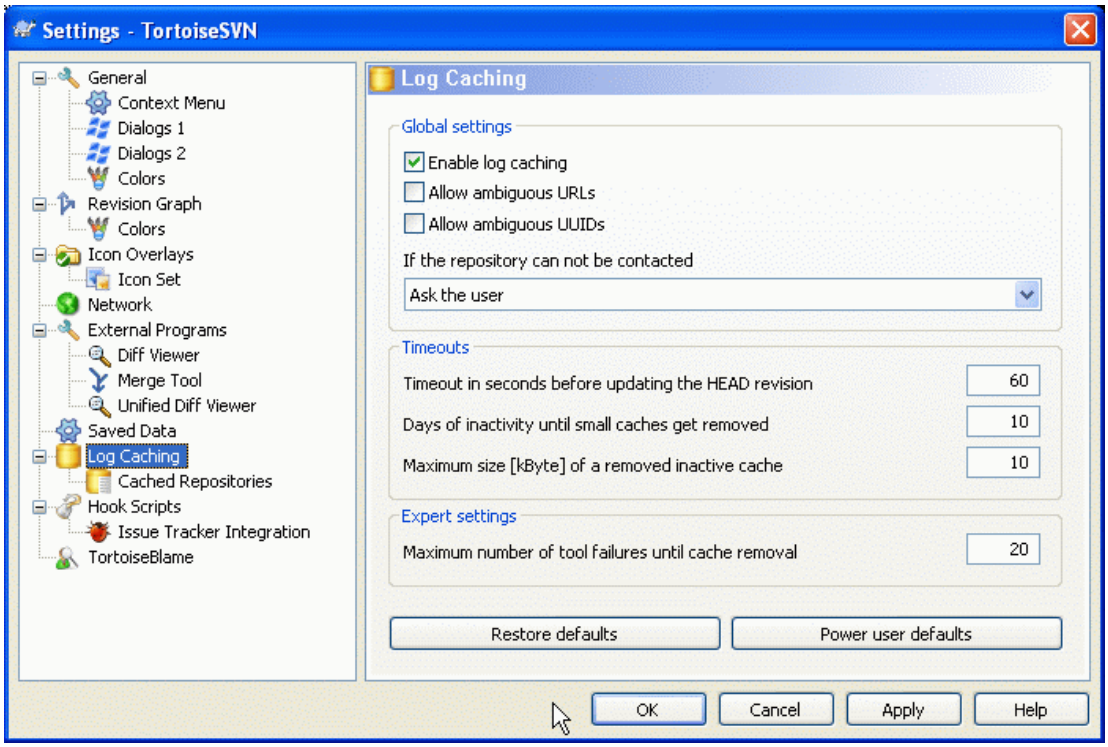


图 4.63. 设置对话框，日志缓存页面

This dialog allows you to configure the log caching feature of TortoiseSVN, which retains a local copy of log messages and changed paths to avoid time-consuming downloads from the server. Using the log cache can dramatically speed up the log dialog and the revision graph. Another useful feature is that the log messages can still be accessed when offline.

启用日志缓存
: Enables log caching whenever log data is requested. If checked, data will be retrieved from the cache when available, and any messages not in the cache will be retrieved from the server and added to the cache.

  If caching is disabled, data will always be retrieved directly from the server and not stored locally.

Allow ambiguous URLs
: Occasionally you may have to connect to a server which uses the same URL for all repositories. Older versions of svnbridge would do this. If you need to access such repositories you will have to check this option. If you don't, leave it unchecked to improve performance.

Allow ambiguous UUIDs
: Some hosting services give all their repositories the same UUID. You may even have done this yourself by copying a repository folder to create a new one. For all sorts of reasons this is a bad idea - a UUID should be unique. However, the log cache will still work in this situation if you check this box. If you don't need it, leave it unchecked to improve performance.

如果不能连接版本库
: If you are working offline, or if the repository server is down, the log cache can still be used to supply log messages already held in the cache. Of course the cache may not be up-to-date, so there are options to allow you to select whether this feature should be used.

  When log data is being taken from the cache without contacting the server, the dialog using those message will show the offline state in its title bar.

Timeout before updating the HEAD revision
: When you invoke the log dialog you will normally want to contact the server to check for any newer log messages. If the timeout set here is non-zero then the server will only be contacted when the timeout has elapsed since the last time contact. This can reduce server round-trips if you open the log dialog frequently and the server is slow, but the data shown may not be completely up-to-date. If you want to use this feature we suggest using a value of 300 (5 minutes) as a compromise.

Days of inactivity until small caches get removed
: If you browse around a lot of repositories you will accumulate a lot of log caches. If you're not actively using them, the cache will not grow very big, so TortoiseSVN purges them after a set time by default. Use this item to control cache purging.

Maximum size of removed inactive caches
: Larger caches are more expensive to reacquire, so TortoiseSVN only purges small caches. Fine tune the threshold with this value.

Maximum number of tool failures before cache removal
: Occasionally something goes wrong with the caching and causes a crash. If this happens the cache is normally deleted automatically to prevent a recurrence of the problem. If you use the less stable nightly build you may opt to keep the cache anyway.

4.30.7.1. □Cached Repositories

On this page you can see a list of the repositories that are cached locally, and the space used for the cache. If you select one of the repositories you can then use the buttons underneath.

Click on the Update to completely refresh the cache and fill in any holes. For a large repository this could be very time consuming, but useful if you are about to go offline and want the best available cache.

Click on the Export button to export the entire cache as a set of CSV files. This could be useful if you want to process the log data using an external program, although it is mainly useful to the developers.

Click on Delete to remove all cached data for the selected repositories. This does not disable caching for the repository so the next time you request log data, a new cache will be created.

4.30.7.2. □日志缓存统计

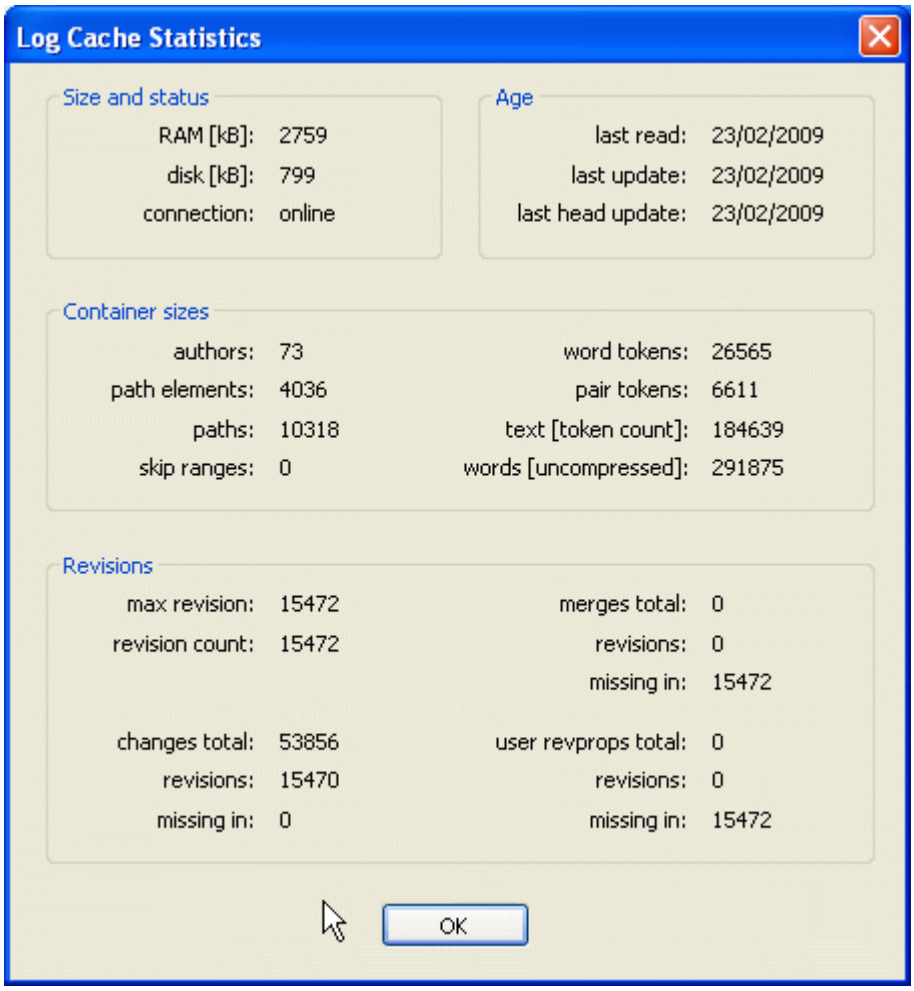


图 4.64. 设置对话框，日志缓存统计

Click on the Details button to see detailed statistics for a particular cache. Many of the fields shown here are mainly of interest to the developers of TortoiseSVN, so they are not all described in detail.

RAM
: The amount of memory required to service this cache.

磁盘
The amount of disk space used for the cache. Data is compressed, so disk usage is generally fairly modest.

连接
Shows whether the repository was available last time the cache was used.

最近更新
The last time the cache content was changed.

Last head update
The last time we requested the HEAD revision from the server.

作者
The number of different authors with messages recorded in the cache.

路径
The number of paths listed, as you would see using svn log -v.

Skip ranges
The number of revision ranges which we have not fetched, simply because they haven't been requested. This is a measure of the number of holes in the cache.

最大版本号
The highest revision number stored in the cache.

版本计数
The number of revisions stored in the cache. This is another measure of cache completeness.

4.30.8. □客户端钩子脚本

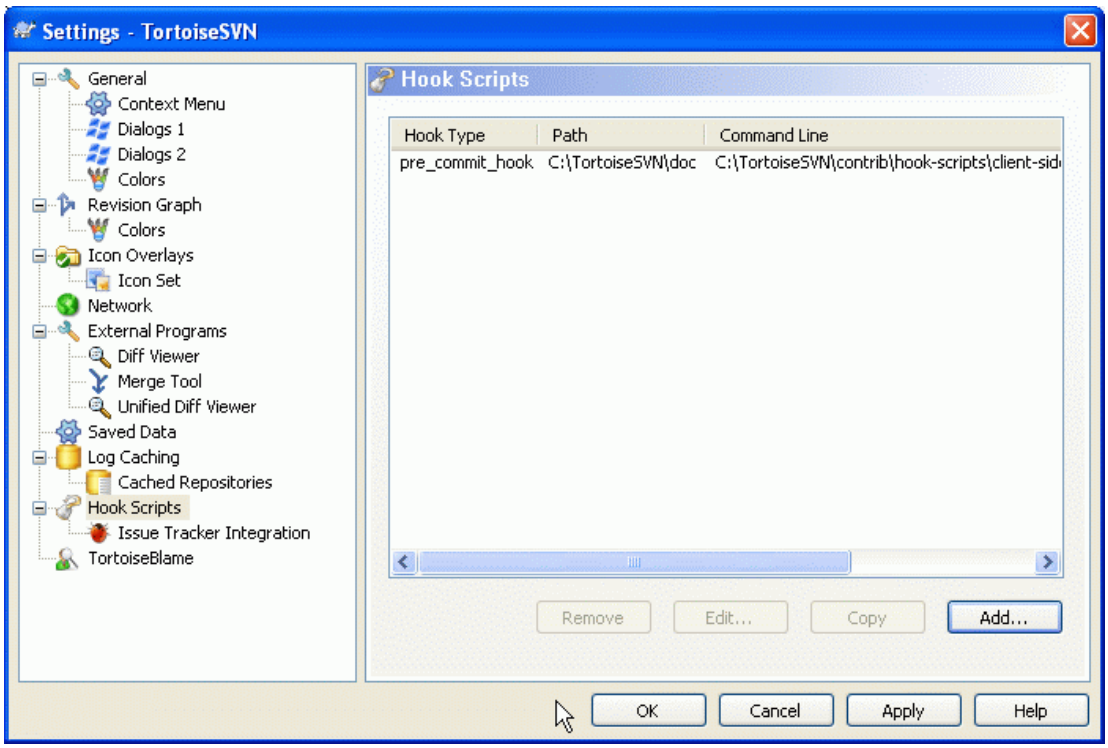


图 4.65. 设置对话框，钩子脚本页

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。

由于各种安全理由和实现问题，钩子脚本在本地机器定义，而不是象工程属性那样。不管是谁提交，都可以定义它做什么事情。当然，你也可以选择调用一个受版本控制的脚本。

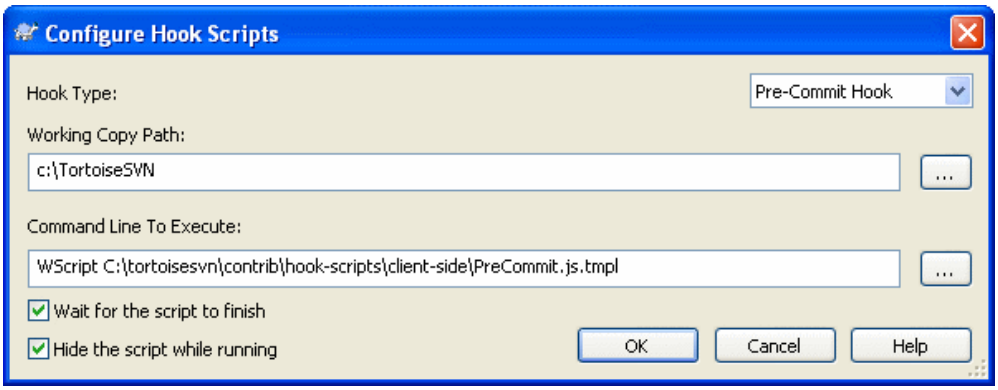


图 4.66. 设置对话框，配置钩子脚本页面

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
: Called before the commit dialog is shown. You might want to use this if the hook modifies a versioned file and affects the list of files that need to be committed and/or commit message. However you should note that because the hook is called at an early stage, the full list of objects selected for commit is not available.

提交之前
: Called after the user clicks OK in the commit dialog, and before the actual commit begins. This hook has a list of exactly what will be committed.

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 更新实际开始之前调用

更新之后
: 在更新之后调用(无论成功或失败)

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。

然后你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关联的其它文件类型，例如perl文件。

The command line includes several parameters which get filled in by TortoiseSVN. The parameters passed depend upon which hook is called. Each hook has its own parameters which are passed in the following order:

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

The meaning of each of these parameters is described here:

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。

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
    Path to a file containing the log message for the commit. The file contains the text in UTF-8 encoding. After successful execution of the start-commit hook, the log message is read back, giving the hook a chance to modify it.

REVISION
    更新或提交完成后的版本库的版本

ERROR
    Path to a file containing the error message. If there was no error, the file will be empty.

CWD
    The current working directory with which the script is run. This is set to the common root directory of all affected paths.

Note that although we have given these parameters names for convenience, you do not have to refer to those names in the hook settings. All parameters listed for a particular hook are always passed, whether you want them or not ;-)

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。

Normally you will want to hide ugly DOS boxes when the script runs, so Hide the script while running is checked by default.

Sample client hook scripts can be found in the contrib folder in the TortoiseSVN repository [http://tortoisesvn.googlecode.com/svn/trunk/contrib/hook-scripts]. (第 3 节 “TortoiseSVN 是完全免费的！”explains how to access the repository).

4.30.8.1. □Issue Tracker Integration

TortoiseSVN can use a COM plugin to query issue trackers when in the commit dialog. The use of such plugins is described in 第 4.28.2 节 “Getting Information from the Issue Tracker”. If your system administrator has provided you with a plugin, which you have already installed and registered, this is the place to specify how it integrates with your working copy.

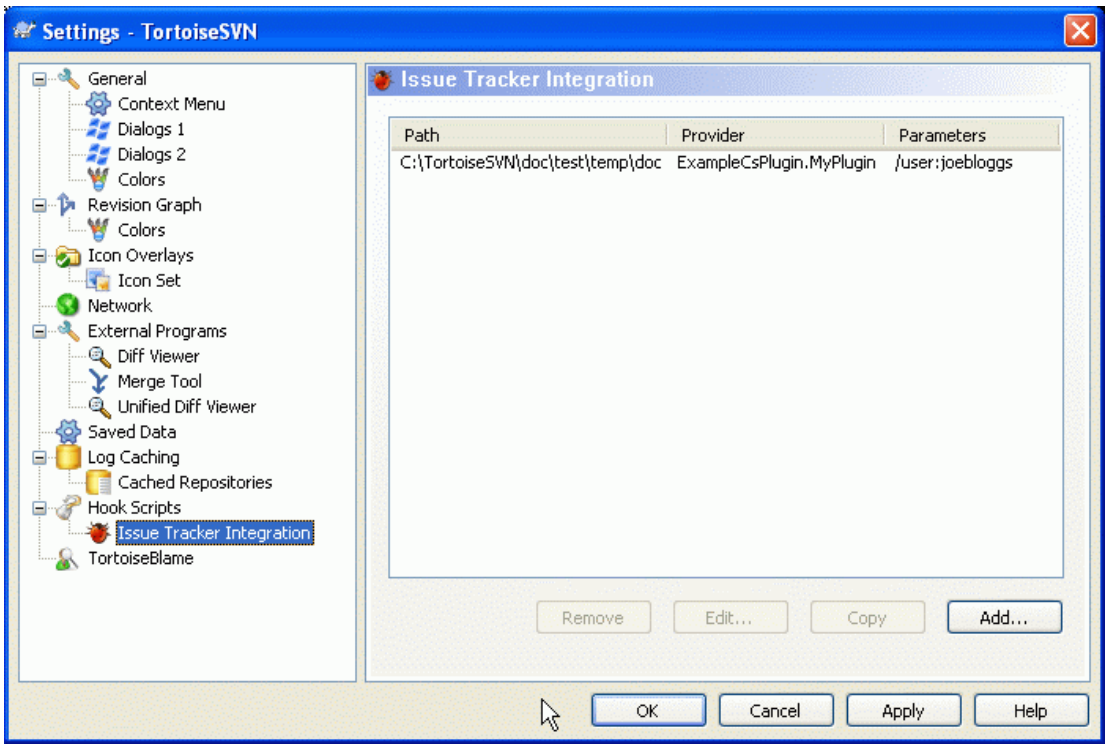


图 4.67. The Settings Dialog, Issue Tracker Integration Page

Click on Add... to use the plugin with a particular working copy. Here you can specify the working copy path, choose which plugin to use from a drop down list of all registered issue tracker plugins, and any parameters to pass. The parameters will be specific to the plugin, but might include your user name on the issue tracker so that the plugin can query for issues which are assigned to you.

If you want all users to use the same COM plugin for your project, you can specify the plugin also with the properties bugtraq:provideruuid and bugtraq:providerparams.

bugtraq:provideruuid
: This property specifies the COM UUID of the IBugtraqProvider, for example {91974081-2DC7-4FB1-B3BE-0DE1C8D6CE4E}. (this example is the UUID of the Gurtle

bugtraq provider [http://code.google.com/p/gurtle/], which is a provider for the Google Code [http://code.google.com/hosting/] issue tracker).

bugtraq:providerparams
This property specifies the parameters passed to the IBugtraqProvider.

Please check the documentation of your IBugtraqProvider plugin to find out what to specify in these two properties.

4.30.9. TortoiseBlame 的设置

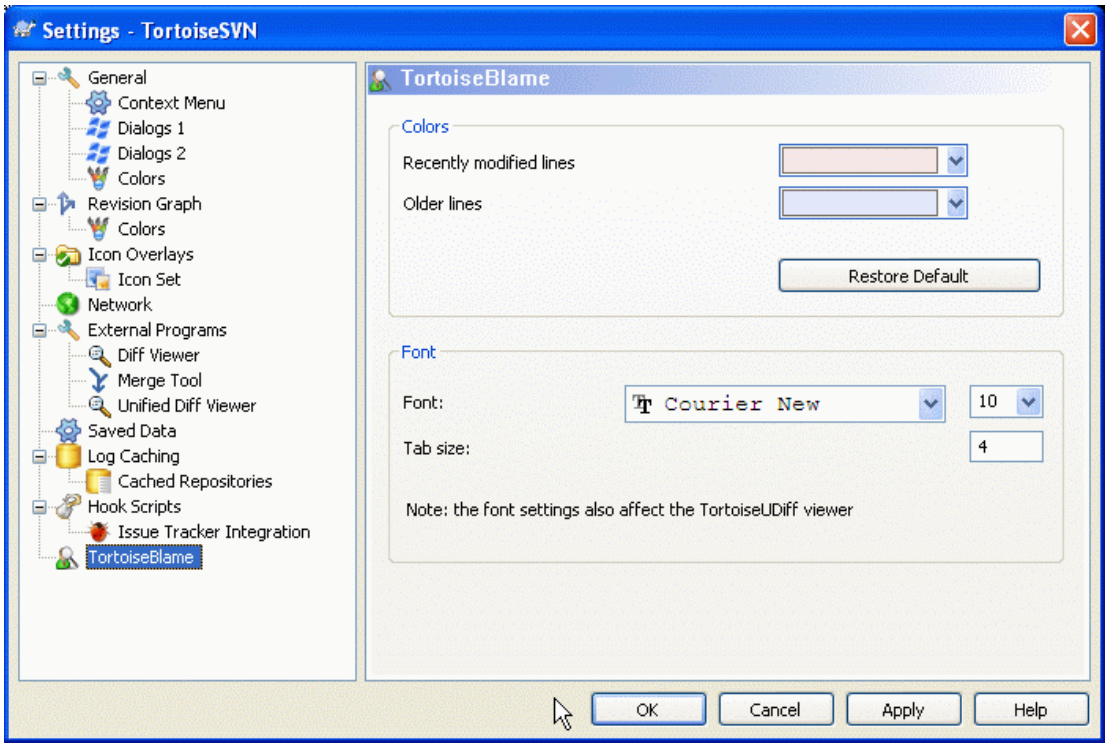


图 4.68. 设置对话框，TortoiseBlame 页面

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。

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. 注册表设置

A few infrequently used settings are available only by editing the registry directly. It goes without saying that you should only edit registry values if you know what you are doing.

配置
通过编辑注册表 HKCU\Software\TortoiseSVN\ConfigDir，你可以为Subversion的配置文件指定一个不同位置。这将影响到TSVN的所有操作。

缓存托盘图标
要为TSVNCache程序添加一个缓存托盘图标，先在 HKCU\Software\TortoiseSVN\CacheTrayIcon 的位置，创建一个 DWORD 值，取值为1。这确实只对开发者才有点用处，因为它允许你来优雅地关闭TSVNCache，而不是在进程列表里kill掉它。(托盘图标可以显示当前已缓存了的文件夹数目 *by Jax)

Debug
To show the command line parameters passed from the shell extension to TortoiseProc.exe create a DWORD key with a value of 1 at HKCU\Software\TortoiseSVN\Debug.

Context Menu Icons
This can be useful if you use something other than the windows explorer or if you get problems with the context menu displaying correctly. create a DWORD key with a value of 0 at HKCU\Software\TortoiseSVN\ShowContextMenuIcons if you don't want TortoiseSVN to not show icons for the shell context menu items. Set this value to 1 to show the icons again.

Block Overlay Status
If you don't want the explorer to update the status overlays while another TortoiseSVN command is running (e.g. Update, Commit, ...) then create a DWORD key with a value of 1 at HKCU\Software\TortoiseSVN\BlockStatus.

Update Check URL
HKCU\Software\TortoiseSVN\UpdateCheckURL contains the URL from which TortoiseSVN tries to download a text file to find out if there are updates available. You can also set this under HKLM instead of HKCU if you want, but HKCU overwrites the setting in HKLM. This might be useful for company admins who don't want their users to update TortoiseSVN until they approve it.

在自动完成列表的文件名称没有扩展名
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, create a DWORD key with a value of 1 at HKCU\Software\TortoiseSVN\AutocompleteRemovesExtensions.

Explorer columns everywhere
The extra columns the TortoiseSVN adds to the details view in Windows Explorer are normally only active in a working copy. If you want those to be accessible everywhere, not just in working copies, create a DWORD key with a value of 1 at HKCU\Software\TortoiseSVN\ColumnsEveryWhere.

合并日志分隔符
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 create a SZ key at HKCU\Software\TortoiseSVN\MergeLogSeparator containing a separator string of your choice.

始终用 TortoiseMerge 追溯改变
TortoiseSVN allows you to assign 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, create a DWORD key with a value of 1 at HKCU\Software\TortoiseSVN\DiffBlamesWithTortoiseMerge.

Current revision highlighting for folders in log dialog
The log dialog highlights the current working copy revision when the log is shown for a file. To do the same thing for a folder requires a working copy crawl, which is the default action, but it can be a slow operation for large working copies.

If you want to change the operation of this feature you must create a DWORD registry key at HKCU\Software\TortoiseSVN\RecursiveLogRev. A value of 0 disables the feature (no highlighting for folders), a value of 1 (default) will fetch the status recursively (find the highest revision in the working copy tree), and a value of 2 will check the revision of the selected folder itself, but will not check any child items.

Make checkout fail if an item of the same name exists

By default, if you checkout a working copy over an existing unversioned folder structure, as you might do after import, then any existing which differ from the repository content will be left unchanged and marked as modified. When you come to commit, it is your local copy which will then be sent back to the repository. Some people would prefer the checkout to fail if the existing content differs, so that if two people add the same file the second person's version does not overwrite the original version by mistake. If you want to force checkouts to fail in this instance you must create a DWORD registry key with value 0 at HKCU\Software\TortoiseSVN\AllowUnversionedObstruction.

4.30.11. □Subversion 的工作文件夹

VS.NET 2003 when used with web projects can't handle the .svn folders that Subversion uses to store its internal information. This is not a bug in Subversion. The bug is in VS.NET 2003 and the frontpage extensions it uses.

> Note that the bug is fixed in VS2005 and later versions.

As of Version 1.3.0 of Subversion and TortoiseSVN, you can set the environment variable SVN_ASP_DOT_NET_HACK. If that variable is set, then Subversion will use _svn folders instead of .svn folders. You must restart your shell for that environment variable to take effect. Normally that means rebooting your PC. To make this easier, you can now do this from the general settings page using a simple checkbox - refer to 第 4.30.1 节 “常规设置”.

For more information, and other ways to avoid this problem in the first place, check out the article about this in our FAQ [http://tortoisesvn.net/aspdotnethack].

4.31. □最后步骤

捐赠!

尽管TSVN和TortoiseMerge是免费的，你也可以通过提交补丁和在项目开发中扮演一些积极角色，来支持我们的开发人员。你同样也可以给点鼓励，它们会让每天在计算机前没日没夜拼命工作的我们感到非常的振奋。

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.tigris.org/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阅读工作副本中所有文件的状态，缺省会忽略外部引用。它记录找到的最高修订版本号，以及那个修订版本的提交时间戳，它也会记录在本地工作副本是否有修改，或混合的修订版本。修订版本号码，更新修订版本范围和修改状态会显示在标准输出。

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。

有几个开关影响 SubWCRev工作。如果使用多个，必须用单个组指定，例如要用-nm，不能用-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.
-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.
-x	If this switch is given, SubWCRev will output the revision numbers in HEX.
-X	If this switch is given, SubWCRev will output the revision numbers in HEX, with 'OX' prepended.

表 5.1. 列出可用的命令行开关

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

关键字	描述
$WCDATE$	用最高提交版本的日期/时间替换。默认使用国际化格式：yyyy-mm-dd hh:mm:ss。作为选择，你可以指定 strftime() 使用自定义格式，例如：$WCDATE=%a %b %d %I:%M:%S %p$。格式字符的列表参见 在线引用 [http://www.cppreference.com/stddate/strftime.html]。
$WCNOW$	Replaced with the current system date/time. This can be used to indicate the build time. Time formatting can be used as described for $WCDATE$.
$WCRANGE$	在工作目录用更新版本范围替换。如果工作目录处于一致的状态，它是一个单一版本。如果工作目录包含混合版本，或者是过时，或者是故意更新到版本，那么这个范围会用象100:200这样的格式来显示。
$WCMIXED$	当有混合版本时用 TText 替换 $WCMIXED?TText:FText$，否则用 FText 替换。
$WCMODS$	若本地存在修改，就用 TText 替换 $WCMODS?TText:FText$，否则用 FText 替换。
$WCURL$	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.
$WCLOCKDATE$	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 5.2. 列出可用的命令行开关

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: testfile.tmpl

char *Revision = "$WCREV$";
char *Modified = "$WCMODS?Modified:Not modified$";
char *Date     = "$WCDATE$";
char *Range    = "$WCRANGE$";
char *Mixed    = "$WCMIXED?Mixed revision WC:Not mixed$";
char *URL      = "$WCURL$";

#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: testfile.txt

char *Revision = "3701";
char *Modified = "Modified";
char *Date     = "2005/06/15 11:15:12";
char *Range    = "3699:3701";
char *Mixed    = "Mixed revision WC";
char *URL      = "http://project.domain.org/svn/trunk/src";

#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.
.Revision	工作副本中的最高提交版本。等价于 $WCREV$
.Date	最高提交版本的提交日期/时间。等价于 $WCDATE$
.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
.Url	在 GetWCInfo 中使用，用工作副本的版本库之 URL 替换。等价于 $WCURL$
.IsSvnItem	True if the item is versioned.
.NeedsLocking	True if the item has the svn:needs-lock property set.

方法	描述
.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 5.3. 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.GetWCInfo(
    filesystem.GetAbsolutePathName("..\\.."), 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);
```

第□6□章□IBugtraqProvider interface

To get a tighter integration with issue trackers than by simply using the bugtraq: properties, TortoiseSVN can make use of COM plugins. With such plugins it is possible to fetch information directly from the issue tracker, interact with the user and provide information back to TortoiseSVN about open issues, verify log messages entered by the user and even run actions after a successful commit to e.g, close an issue.

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 “TortoiseSVN 是完全免费的！” explains how to access the repository).

6.1. □The IBugtraqProvider interface

TortoiseSVN 1.5 can use plugins which implement the IBugtraqProvider interface. The interface provides a few methods which plugins can use to interact with the issue tracker.

```
HRESULT ValidateParameters (
  // Parent window for any UI that needs to be
  // displayed during validation.
  [in] HWND hParentWnd,

  // The parameter string that needs to be validated.
  [in] BSTR parameters,

  // Is the string valid?
  [out, retval] VARIANT_BOOL *valid
);
```

This method is called from the settings dialog where the user can add and configure the plugin. The parameters string can be used by a plugin to get additional required information, e.g., the URL to the issue tracker, login information, etc. The plugin should verify the parameters string and show an error dialog if the string is not valid. The hParentWnd parameter should be used for any dialog the plugin shows as the parent window. The plugin must return TRUE if the validation of the parameters string is successful. If the plugin returns FALSE, the settings dialog won't allow the user to add the plugin to a working copy path.

```
HRESULT GetLinkText (
  // Parent window for any (error) UI that needs to be displayed.
  [in] HWND hParentWnd,

  // The parameter string, just in case you need to talk to your
  // web service (e.g.) to find out what the correct text is.
  [in] BSTR parameters,

  // What text do you want to display?
  // Use the current thread locale.
  [out, retval] BSTR *linkText
);
```

The plugin can provide a string here which is used in the TortoiseSVN commit dialog for the button which invokes the plugin, e.g., "Choose issue" or "Select ticket". Make sure the string is not too long, otherwise it might not fit into the button. If the method returns an error (e.g., E_NOTIMPL), a default text is used for the button.

```
HRESULT GetCommitMessage (
  // Parent window for your provider's UI.
  [in] HWND hParentWnd,

  // Parameters for your provider.
  [in] BSTR parameters,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,

  // The text already present in the commit message.
  // Your provider should include this text in the new message,
  // where appropriate.
  [in] BSTR originalMessage,

  // The new text for the commit message.
  // This replaces the original message.
  [out, retval] BSTR *newMessage
);
```

This is the main method of the plugin. This method is called from the TortoiseSVN commit dialog when the user clicks on the plugin button. The parameters string is the string the user has to enter in the settings dialog when he configures the plugin. Usually a plugin would use this to find the URL of the issue tracker and/or login information or more. The commonRoot string contains the parent path of all items selected to bring up the commit dialog. Note that this is not the root path of all items which the user has selected in the commit dialog. The pathList parameter contains an array of paths (as strings) which the user has selected for the commit. The originalMessage parameter contains the text entered in the log message box in the commit dialog. If the user has not yet entered any text, this string will be empty. The newMessage return string is copied into the log message edit box in the commit dialog, replacing whatever is already there. If a plugin does not modify the originalMessage string, it must return the same string again here, otherwise any text the user has entered will be lost.

6.2. □The IBugtraqProvider2 interface

In TortoiseSVN 1.6 a new interface was added which provides more functionality for plugins. This IBugtraqProvider2 interface inherits from IBugtraqProvider.

```
HRESULT GetCommitMessage2 (
  // Parent window for your provider's UI.
  [in] HWND hParentWnd,

  // Parameters for your provider.
  [in] BSTR parameters,
  // The common URL of the commit
  [in] BSTR commonURL,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,

  // The text already present in the commit message.
  // Your provider should include this text in the new message,
```

```
  // where appropriate.
  [in] BSTR originalMessage,

  // You can assign custom revision properties to a commit
  // by setting the next two params.
  // note: Both safearrays must be of the same length.
  //       For every property name there must be a property value!

  // The content of the bugID field (if shown)
  [in] BSTR bugID,

  // Modified content of the bugID field
  [out] BSTR * bugIDOut,

  // The list of revision property names.
  [out] SAFEARRAY(BSTR) * revPropNames,

  // The list of revision property values.
  [out] SAFEARRAY(BSTR) * revPropValues,

  // The new text for the commit message.
  // This replaces the original message
  [out, retval] BSTR * newMessage
);
```

This method is called from the TortoiseSVN commit dialog when the user clicks on the plugin button. This method is called instead of GetCommitMessage(). Please refer to the documentation for GetCommitMessage for the parameters that are also used there. The parameter commonURL is the parent URL of all items selected to bring up the commit dialog. This is basically the URL of the commonRoot path. 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. 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)

Because TortoiseSVN is being developed all the time it is sometimes hard to keep the documentation completely up to date. We maintain an online FAQ [http://tortoisesvn.tigris.org/faq.html] which contains a selection of the questions we are asked the most on the TortoiseSVN mailing lists <dev@tortoisesvn.tigris.org> and <users@tortoisesvn.tigris.org>.

We also maintain a project Issue Tracker [http://issues.tortoisesvn.net] 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>.

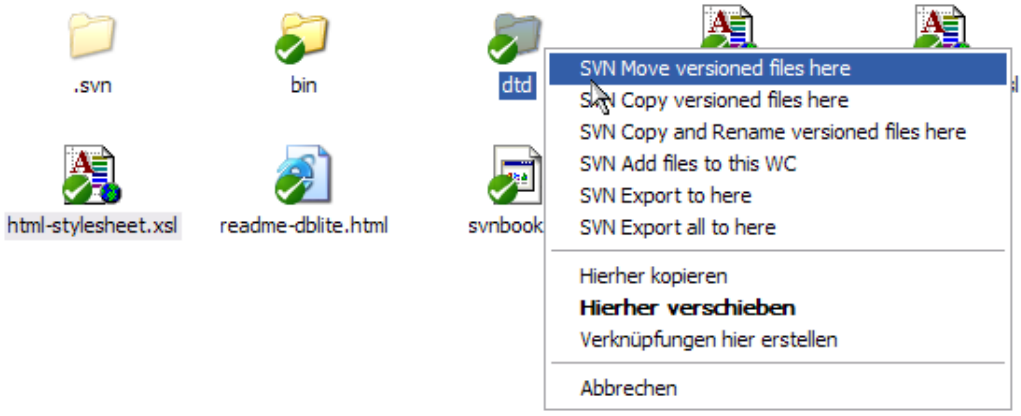
附录□B. □如何实现 …

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。

The recommended way is by right-dragging the files to the new location. Simply right-click on the files you want to move/copy without releasing the mouse button. Then drag the files to the new location and release the mouse button. A context menu will appear where you can either choose Context Menu → SVN Copy versioned files here. or Context Menu → SVN Move versioned files here.



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。

关于工程属性的具体信息，请参照 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

如果想恢复某个版本或者版本范围的变更，最简单的方法是使用版本日志对话框。这种方法也可以用来撤销最近的若干次变更，把以前的某个版本变成最新版。

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. Select TortoiseSVN → Show Log to display a list of revisions. You may need to use Show All or Next 100 to show the revision(s) you are interested in.
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. Note that for multiple revisions, the range must be unbroken with no gaps. Right click on the selected revision(s), then select Context Menu → Revert changes from this revision.
4. Or if you want to make an earlier revision the new HEAD revision, right click on the selected revision, then select Context Menu → Revert to this revision. This will discard all changes after the selected revision.

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

如果要撤销更大版本范围的变更，可以使用合并对话框。上一个方法在后台使用了合并的机制，在这个方法里我们直接使用合并功能。

1. 在工作副本上选择TortoiseSVN → 合并。
2. In the From: field enter the full folder URL of the branch or tag containing the changes you want to revert in your working copy. This should come up as the default URL.
3. 在起始版本文本框里输入当前工作副本的版本号。如果能够保证没有其他人会提交变更，可以使用最新版本。
4. 确认使用“起始：”的 URL检查框处于被选中的状态。
5. In the To Revision field enter the revision number that you want to revert to, namely the one before the first revision to be reverted.
6. 点击合并按钮完成合并。

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.4.3. □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/1.5/svn.reposadmin.maint.html].

B.5. □Compare two revisions of a file or folder

If you want to compare two revisions in an item's history, for example revisions 100 and 200 of the same file, just use TortoiseSVN → Show Log to list the revision history for that file. Pick the two revisions you want to compare then use Context Menu → Compare Revisions.

If you want to compare the same item in two different trees, for example the trunk and a branch, you can use the repository browser to open up both trees, select the file in both places, then use Context Menu → Compare Revisions.

If you want to compare two trees to see what has changed, for example the trunk and a tagged release, you can use TortoiseSVN → Revision Graph Select the two nodes to compare, then use Context Menu → Compare HEAD Revisions. This will show a list of changed files, and you can then select individual files to view the changes in detail. You can also export a tree structure containing all the changed files, or simply a list of all changed files. Read 第 4.10.3 节 “比较文件夹”for more information. Alternatively use Context Menu → Unified Diff of HEAD Revisions to see a summary of all differences, with minimal context.

B.6. □包含一个普通的子项目

有时候你希望在你的工作副本中引入另一个项目，或许是一些库代码，你不必在你的版本库复制一份，因为你会失去与原始(且维护的)代码的联系，或者你可能有多个项目共享同一份核心代码，有至少三种方法处理这个问题。

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.

如果一个外部工程位于同一版本库中，当你向主项目提交你的修改时，你对外部工程做的修改也会包含在提交列表中。

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

在你的项目创建一个包含普通代码的新目录，但不将其添加到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

如果你在多个项目中使用共同的代码，而你不想为每个项目保存一份副本，你可以将其检出到一个单独的位置，与其他项目关联，例如:

```
C:\Projects\Proj1
C:\Projects\Proj2
C:\Projects\Proj3
C:\Projects\Common
```

然后使用相对路径例如..\..\Common\DSPcore引用通用代码。

如果你的项目分散到不相关的位置，你可以使用一个变种方式，可以将通用代码放到一个位置，且使用盘符映射到你可以在项目硬编码的内容，例如将通用代码检出到D:\Documents \Framework或C:\Documents and Settings\{login}\My Documents\framework，然后使用

```
SUBST X: "D:\Documents\framework"
```

在你的源代码创建磁盘映射，你的代码可以使用绝对位置。

```
#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.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。

如果你还没有提交，你只需要TortoiseSVN → Revert...来取消添加，你需要将这个文件添加到忽略列表，这样它们才不会被再次误添加近来。

如果文件已经存在于版本库，你需要做更多的工作。

1. 用 Shift 键获得扩展上下文菜单，使用 TortoiseSVN → 删除(保留版本副本) 标记文件/目录从版本库删除，但是保留版本副本。

2. TortoiseSVN → 提交父目录。

3. 将文件/目录增加到忽略列表，所以你不会再遇到同样的麻烦。

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.

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.

Versions 1.3.0 and later of TortoiseSVN must be installed under Computer Configuration and not under User Configuration. This is because those versions need the new 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 8 from Microsoft on each computer you want to install TortoiseSVN as per user.

C.2. □重定向升级检查

TortoiseSVN会每隔几天检查是否有新版本可以下载，如果有新版本存在，会给用户显示相关信息的对话框。

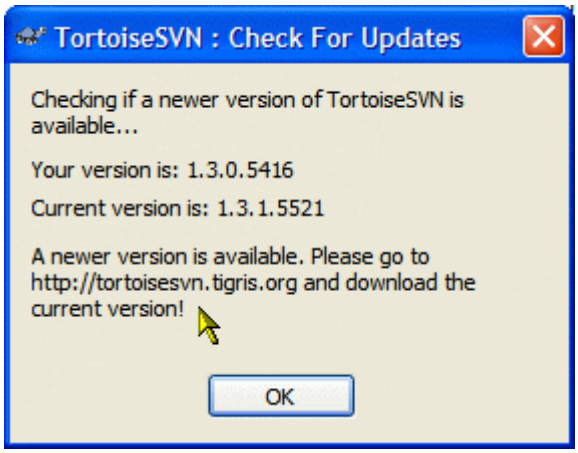


图 C.1. 升级对话框

如果你为你的域中的许多用户负责，你或许希望所有的用户使用你允许的版本，而不是最新的版本，你可能不希望显示升级对话框而使得用户立刻升级。

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 upgrade dialog. You can write there whatever you want. Just note that the space in the upgrade 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 upgrade 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 confusions with users which always install everything no matter if 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

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
0x0000000000000040	解决
0x0000000000000080	切换

取值	菜单入口
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
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。

TortoiseSVN 使用临时文件在 shell 扩展和主程序之间传递多个参数。从 TortoiseSVN 1.5.0 开始，废弃/notempfile参数，不再需要增加此参数。

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.

如果想在进度对话框执行完毕后自动关闭，而又不必设置永久性的参数，可以传递/closeonend参数。

- /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
- /closeonend:4如果没有错误、冲突和合并，会自动关闭

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	打开日志对话框，/path 指定了显示日志的文件或目录，另外还有三个选项可以设置: /startrev:xxx、/endrev:xxx和/strict
:checkout	打开检出对话框，/path指定了目标路径，而/url制定了检出的URL。
:import	打开导入对话框，/path 指定了数据导入路径。
:update	将工作副本的/path更新到HEAD，如果给定参数/rev，就会弹出一个对话框询问用户需要更新到哪个修订版本。为了防止指定修订版本

命令	描述
	号/rev:1234的对话框，需要选项/nonrecursive和/ignoreexternals。
: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	清理中断和终止的操作，将工作副本的/path解锁。
: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	打开选项对话框。/path 指定目标目录。
:export	将/path的工作副本导出到另一个目录，如果/path指向另一个未版本控制目录，对话框会询问要导出到/path的URL。
: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. These pre-fill the relevant fields in the merge dialog.
: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. 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 /path and its BASE. To explicitly set the revision numbers use /startrev:xxx and /endrev:xxx. If /blame is set and /path2 is not set, then the diff is done by first blaming the files with the given revisions.
: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.

命令	描述
	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	打开为修改检出对话框，/path 指定了工作副本目录。
: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.
: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	创建/path下的补丁文件。
:revisiongraph	显示/path目录下的版本变化图。
: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	显示 /path 给出的路径之属性对话框。

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. □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 [-N] [--ignore-externals] [-r rev] URL PATH
```

如果希望只检出顶级目录被选中，使用-N选项。

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] [-N] [--ignore-externals] PATH
```

如果希望只更新顶级目录，使用-N选项。

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" [-N] [--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
```

The Test Merge performs the same merge with the --dry-run switch.

```
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
```

If you use TortoiseBlame to view the blame info, the file log is also required to show log messages in a tooltip. If you view blame as a text file, this information is not required.

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
```

Because the svn:ignore property is often a multi-line value, it is shown here as being changed via a text file rather than directly on the command line.

E.2.28. □创建补丁

```
svn diff PATH > patch-file
```

TortoiseSVN creates a patch file in unified diff format by comparing the working copy with its BASE version.

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

This appendix contains a more detailed discussion of the implementation of some of TortoiseSVN's features.

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. For working copies which were created using Subversion 1.4.0 and later, the svn:needs-lock status is cached locally by Subversion and this is used to determine when to show this overlay. For working copies which are in pre-1.4.x format, TortoiseSVN shows this overlay when the file has read-only status. Note that Subversion automatically upgrades working copies when you update them, although the caching of the svn:needs-lock property may not happen until the file itself is updated.

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.

- 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.
- Locked is only loaded if there are fewer than 13 overlays already loaded. It falls back to Normal if there are not enough slots.
- Added is only loaded if there are fewer than 14 overlays already loaded. It falls back to Modified if there are not enough slots.

附录□G. □用 SSH 使服务器更安全

This section provides a step-by-step guide to setting up Subversion and TortoiseSVN to use the svn+ssh protocol. If you already use authenticated SSH connections to login to your server, then you are already there and you can find more detail in the Subversion book. If you are not using SSH but would like to do so to protect your Subversion installation, this guide gives a simple method which does not involve creating a separate SSH user account on the server for every subversion user.

In this implementation we create a single SSH user account for all subversion users, and use different authentication keys to differentiate between the real Subversion users.

In this appendix we assume that you already have the subversion tools installed, and that you have created a repository as detailed elsewhere in this manual. Note that you should not start svnserve as a service or daemon when used with SSH.

Much of the information here comes from a tutorial provided by Marc Logemann, which can be found at www.logemann.org [http://www.logemann.org/2007/03/13/subversion-tortoisesvn-ssh-howto/] Additional information on setting up a Windows server was provided by Thorsten Müller. Thanks guys!

G.1. □配置 Linux 服务器

You need to have SSH enabled on the server, and here we assume that you will be using OpenSSH. On most distributions this will already be installed. To find out, type:

```
ps xa | grep sshd
```

and look for ssh jobs.

One point to note is that if you build Subversion from source and do not provide any argument to ./configure, Subversion creates a bin directory under /usr/local and places its binaries there. If you want to use tunneling mode with SSH, you have to be aware that the user logging in via SSH needs to execute the svnserve program and some other binaries. For this reason, either place /usr/local/bin into the PATH variable or create symbolic links of your binaries to the /usr/sbin directory, or to any other directory which is commonly in the PATH.

To check that everything is OK, login in as the target user with SSH and type:

```
which svnserve
```

This command should tell you if svnserve is reachable.

Create a new user which we will use to access the svn repository:

```
useradd -m svnuser
```

Be sure to give this user full access rights to the repository.

G.2. □配置 Windows 服务器

Install Cygwin SSH daemon as described here: http://pigtail.net/LRP/printsrv/cygwin-sshd.html

Create a new Windows user account svnuser which we will use to access the repository. Be sure to give this user full access rights to the repository.

If there is no password file yet then create one from the Cygwin console using:

```
mkpasswd -l > /etc/passwd
```

G.3. □用于 TortoiseSVN 的 SSH 客户端工具

Grab the tools we need for using SSH on the windows client from this site: http://www.chiark.greenend.org.uk/~sgtatham/putty/ Just go to the download section and get Putty, Plink, Pageant and Puttygen.

G.4. □创建 OpenSSH 证书

The next step is to create a key pair for authentication. There are two possible ways to create keys. The first is to create the keys with PuTTYgen on the client, upload the public key to your server and use the private key with PuTTY. The other is to create the key pair with the OpenSSH tool ssh-keygen, download the private key to your client and convert the private key to a PuTTY-style private key.

G.4.1. □使用 ssh-keygen 创建密钥

Login to the server as root or svnuser and type:

```
ssh-keygen -b 1024 -t dsa -N passphrase -f keyfile
```

substituting a real pass-phrase (which only you know) and key file. We just created a SSH2 DSA key with 1024 bit key-phrase. If you type

```
ls -l keyfile*
```

you will see two files, keyfile and keyfile.pub. As you might guess, the .pub file is the public key file, the other is the private one.

Append the public key to those in the .ssh folder within the svnuser home directory:

```
cat keyfile.pub >> /home/svnuser/.ssh/authorized_keys
```

In order to use the private key we generated, we have to convert it to a putty format. This is because the private key file format is not specified by a standards body. After you download the private key file to your client PC, start PuTTYgen and use Conversions → Import key. Browse to your file keyfile which you got from the server the passphrase you used when creating the key. Finally click on Save private key and save the file as keyfile.PPK.

G.4.2. □使用 PuTTYgen 创建密钥

Use PuTTYgen to generate a public-key/private-key pair and save it. Copy the public key to the server and append it to those in the .ssh folder within the svnuser home directory:

```
cat keyfile.pub >> /home/svnuser/.ssh/authorized_keys
```

G.5. □使用 PuTTY 测试

To test the connection we will use PuTTY. Start the program and on the Session tab set the hostname to the name or IP address of your server, the protocol to SSH and save the session as SvnConnection or whatever name you prefer. On the SSH tab set the preferred SSH protocol version to 2 and from Auth set the full path to the .PPK private key file you converted earlier. Go back to the Sessions tab and hit the Save button. You will now see SvnConnection in the list of saved sessions.

Click on Open and you should see a telnet style login prompt. Use svnuser as the user name and if all is well you should connect directly without being prompted for a password.

You may need to edit /etc/sshd_config on the server. Edit lines as follows and restart the SSH service afterwards.

```
PubkeyAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
```

G.6. □使用 TortoiseSVN 测试 SSH

So far we have only tested that you can login using SSH. Now we need to make sure that the SSH connection can actually run svnserve. On the server modify /home/svnuser/.ssh/authorized_keys as follows to allow many subversion authors to use the same system account, svnuser. Note that every subversion author uses the same login but a different authentication key, thus you have to add one line for every author.

Note: This is all on one very long line.

```
command="svnserve -t -r <ReposRootPath> --tunnel-user=<author>",
            no-port-forwarding,no-agent-forwarding,no-X11-forwarding,
            no-pty ssh-rsa <PublicKey> <Comment>
```

There are several values that you need to set according to your setup.

<ReposRootPath> should be replaced with the path to the directory containing your repositories. This avoids the need to specify full server paths within URLs. Note that you must use forward slashes even on a Windows server, e.g. c:/svn/reposroot. In the examples below we assume that you have a repository folder within the repository root called repos.

<author> should be replaced with the svn author that you want to be stored on commit. This also allows svnserve to use its own access rights within svnserve.conf.

<PublicKey> should be replaced with the public key that you generated earlier.

<Comment> can be any comment you like, but it is useful for mapping an svn author name to the person's real name.

Right click on any folder in Windows Explorer and select TortoiseSVN → Repo-Browser. You will be prompted to enter a URL, so enter one in this form:

```
svn+ssh://svnuser@SvnConnection/repos
```

What does this URL mean? The Schema name is svn+ssh which tells TortoiseSVN how to handle the requests to the server. After the double slash, you specify the user to

connect to the server, in our case svnuser. After the @ we supply our PuTTY session name. This session name contains all details like where to find the private key and the server's IP or DNS. Lastly we have to provide the path to the repository, relative to the repository root on the server, as specified in the authorized_keys file.

Click on OK and you should be able to browse the repository content. If so you now have a running SSH tunnel in conjunction with TortoiseSVN.

Note that by default TortoiseSVN uses its own version of Plink to connect. This avoids a console window popping up for every authentication attempt, but it also means that there is nowhere for error messages to appear. If you receive the error "Unable to write to standard output", you can try specifying Plink as the client in TortoiseSVN's network settings. This will allow you to see the real error message generated by Plink.

G.7. □SSH 配置参数

One way to simplify the URL in TortoiseSVN is to set the user inside the PuTTY session. For this you have to load your already defined session SvnConnection in PuTTY and in the Connection tab set Auto login user to the user name, e.g. svnuser. Save your PuTTY session as before and try the following URL inside TortoiseSVN:

```
svn+ssh://SvnConnection/repos
```

This time we only provide the PuTTY session SvnConnection to the SSH client TortoiseSVN uses (TortoisePlink.exe). This client will check the session for all necessary details.

At the time of writing PuTTY does not check all saved configurations, so if you have multiple configurations with the same server name, it will pick the first one which matches. Also, if you edit the default configuration and save it, the auto login user name is not saved.

Many people like to use Pageant for storing all their keys. Because a PuTTY session is capable of storing a key, you don't always need Pageant. But imagine you want to store keys for several different servers; in that case you would have to edit the PuTTY session over and over again, depending on the server you are trying to connect with. In this situation Pageant makes perfect sense, because when PuTTY, Plink, TortoisePlink or any other PuTTY-based tool is trying to connect to an SSH server, it checks all private keys that Pageant holds to initiate the connection.

For this task, simply run Pageant and add the private key. It should be the same private key you defined in the PuTTY session above. If you use Pageant for private key storage, you can delete the reference to the private key file in your saved PuTTY session. You can add more keys for other servers, or other users of course.

If you don't want to repeat this procedure after every reboot of your client, you should place Pageant in the auto-start group of your Windows installation. You can append the keys with complete paths as command line arguments to Pageant.exe

The last way to connect to an SSH server is simply by using this URL inside TortoiseSVN:

```
svn+ssh://svnuser@100.101.102.103/repos
svn+ssh://svnuser@mydomain.com/repos
```

As you can see, we don't use a saved PuTTY session but an IP address (or domain name) as the connection target. We also supply the user, but you might ask how the private key file will be found. Because TortoisePlink.exe is just a modified version

of the standard Plink tool from the PuTTY suite, TortoiseSVN will also try all the keys stored in Pageant.

If you use this last method, be sure you do not have a default username set in PuTTY. We have had reports of a bug in PuTTY causing connections to close in this case. To remove the default user, simply clear HKEY_CURRENT_USER\Software\SimonTatham\Putty\Sessions\Default%20Settings\HostName

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	A proprietary Subversion filesystem backend for repositories. Can be used on network shares. Default for 1.2 and newer repositories.
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	A term frequently used in revision control systems to describe what happens when development forks at a particular point and follows 2 separate paths. You can create a branch off the main development line so as to develop a new feature without rendering the main line unstable. Or you can branch a stable release to which you make only bug fixes, while new developments take place on the unstable trunk. In Subversion a branch is implemented as a “cheap copy”.
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	When you delete a versioned item (and commit the change) the item no longer exists in the repository after the committed revision. But of course it still exists in earlier repository revisions, so you can still access it. If necessary, you can copy a deleted item and “resurrect” it complete with history.
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	To quote from the Subversion book: “ Recursively clean up the working copy, removing locks and resuming unfinished operations. If you ever get a working copy locked error, run this command to remove stale locks and get your working copy into a usable state again. ” Note that in this context lock refers to local filesystem locking, not repository locking.
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	When you take out a lock on a versioned item, you mark it in the repository as non-committable, except from the working copy where the lock was taken out.

索引

符号

A

B

C

D

G

I

M

N

P

R

S

T

U

V

W