TortoiseSVN

Windows 下的一种 Subversion 客户端

Version 1.5.0 - r12606

Stefan Küng
Lübbe Onken
Simon Large

TortoiseSVN: Windows 下的一种 Subversion 客户端: Version 1.5.0 - r12606

由 Stefan Küng、Lübbe Onken和Simon Large
译者: Dongsheng Song (dongsheng.song@gmail.com), rocksun (daijun@gmail.com), Stan (thinkernel@yeah.net), xingyu.wang (xingyu.wang@gmail.com), Tan Ruyan (try876@gmail.com), liuheqi (liuheqi@gmail.com), Jaxx_chen (Jaxx.Chen@gmail.com), sunjing (sunjingzhi@gmail.com), kkeric (kkeric@sina.com), lulu2me (lu-liuyun@163.com), yasakya (yasakya@gmail.com), amo (amosir@gmail.com)

出版方 2008-06-22 23:40:52 +0800

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。

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。

阅读指南

This 前言 explains a little about the TortoiseSVN project, the community of people who work on it, and the licensing conditions for using it and distributing it.

The 第 1 章 简介 explains what TortoiseSVN is, what it does, where it comes from and the basics for installing it on your PC.

In 第 2 章 基本概念 we give a short introduction to the Subversion revision control system which underlies TortoiseSVN. This is borrowed from the documentation for the Subversion project and explains the different approaches to version control, and how Subversion works.

在服务器安装一章里介绍了如何安装一个版本控制服务器。虽然大部分的 Subversion 用户从来不需要自己安装服务器，但是对系统管理员来说这一章非常有用。

The chapter on 第 4 章 版本库 explains how to set up a local repository, which is useful for testing Subversion and TortoiseSVN using a single PC. It also explains a bit about repository administration which is also relevant to repositories located on a server.

The 第 5 章 日常使用指南 is the most important section as it explains all the main features of TortoiseSVN and how to use them. It takes the form of a tutorial, starting with checking out a working copy, modifying it, committing your changes, etc. It then progresses to more advanced topics.

第 6 章 SubWCRev 程序 is a separate program included with TortoiseSVN which can extract the information from your working copy and write it into a file. This is useful for including build information in your projects.

The 附录 B, 如何实现 ... section answers some common questions about performing tasks which are not explicitly covered elsewhere.

The section on 附录 D, TortoiseSVN 操作 shows how the TortoiseSVN GUI dialogs can be called from the command line. This is useful for scripting where you still need user interaction.

The 附录 E, 命令行交叉索引 give a correlation between TortoiseSVN commands and their equivalents in the Subversion command line client `svn.exe`.

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 是一个开源项目，那意味着你可以访问程序所有的源代码，你可以在 http://tortoisesvn.tigris.org/svn/tortoisesvn/ 浏览代码(用户名: guest，密码为空)。最新的版本(我们正在为之工作的版本)位于`/trunk/`，发布的版本位于`/tags/`。

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。

致谢

时间机器
: TortoiseSVN 项目的发起者

Stefan Küng
: TortoiseSVN 的主要开发者

Lübbe Onken
: for the beautiful icons, logo, bug hunting, translating and managing the translations

Simon Large
: for helping with the documentation and bug hunting

Subversion 权威指南
: 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

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`###`字体显示。

重要

使用图标标记的重要提示。

提示

技巧让你的生活更加简单。

小心

操作时需要小心的地方。

警告

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。

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。

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 studied the source code of TortoiseCVS and used it as a base for TortoiseSVN. He then started the project, registered the domain `tortoisesvn.org` and put the source code online. During that time, Stefan Küng was looking for a good and free version control system and found Subversion and the source for TortoiseSVN. Since TortoiseSVN was still not ready for use then he joined the project and started programming. Soon he rewrote most of the existing code and started adding commands and features, up to a point where nothing of the original code remained.

As Subversion became more stable it attracted more and more users who also started using TortoiseSVN as their Subversion client. The user base grew quickly (and is still growing every day). That's when Lübbe Onken offered to help out with some nice icons and a logo for TortoiseSVN. And he takes care of the website and manages the translation.

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，例如覆盖图标可能不会显示。

覆盖图标
: 每个版本控制的文件和目录的状态使用小的覆盖图标表示，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。

安装 TortoiseSVN

系统要求

TortoiseSVN runs on Windows 2000 SP2, Windows XP or higher. Windows 98, Windows ME and Windows NT4 are no longer supported since TortoiseSVN 1.2.0, but you can still download the older versions if you really need them.

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)。

安装

TortoiseSVN comes with an easy to use installer. Double click on the installer file and follow the instructions. The installer will take care of the rest.

重要

You need Administrator privileges to install TortoiseSVN.

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。

拼写检查器

TortoiseSVN 包括了一个拼写检查器，可以检查你的提交日志信息，当你的项目语言不是你的本地语言时尤其有用，拼写检查器使用 OpenOffice [http://openoffice.org] 和 Mozilla [http://mozilla.org] 相同的词典。

The installer automatically adds the US and UK English dictionaries. If you want other languages, the easiest option is simply to install one of TortoiseSVN's language packs. This will install the appropriate dictionary files as well as the TortoiseSVN local user interface. Next time you restart, the dictionary will be available too.

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"项目设置"一节。
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. If none of the above works, then the default language is English, which is included with the standard installation.

第 2 章 基本概念

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

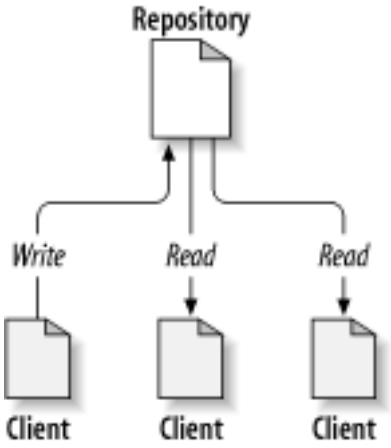
这一章是对 Subversion 一个简短随意的介绍，如果你对版本控制很陌生，这一章节完全是为你准备的，我们从讨论基本概念开始，深入理解 Subversion 的思想，然后展示许多简单的实例。

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

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.

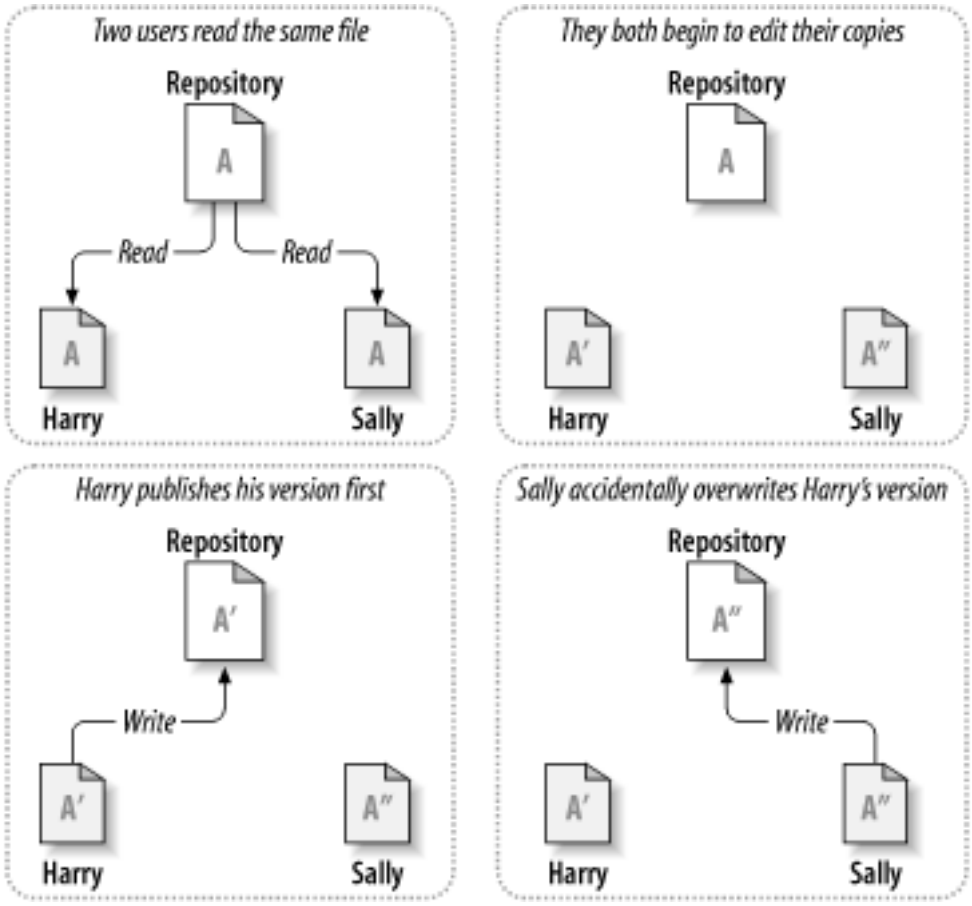
版本模型

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。

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

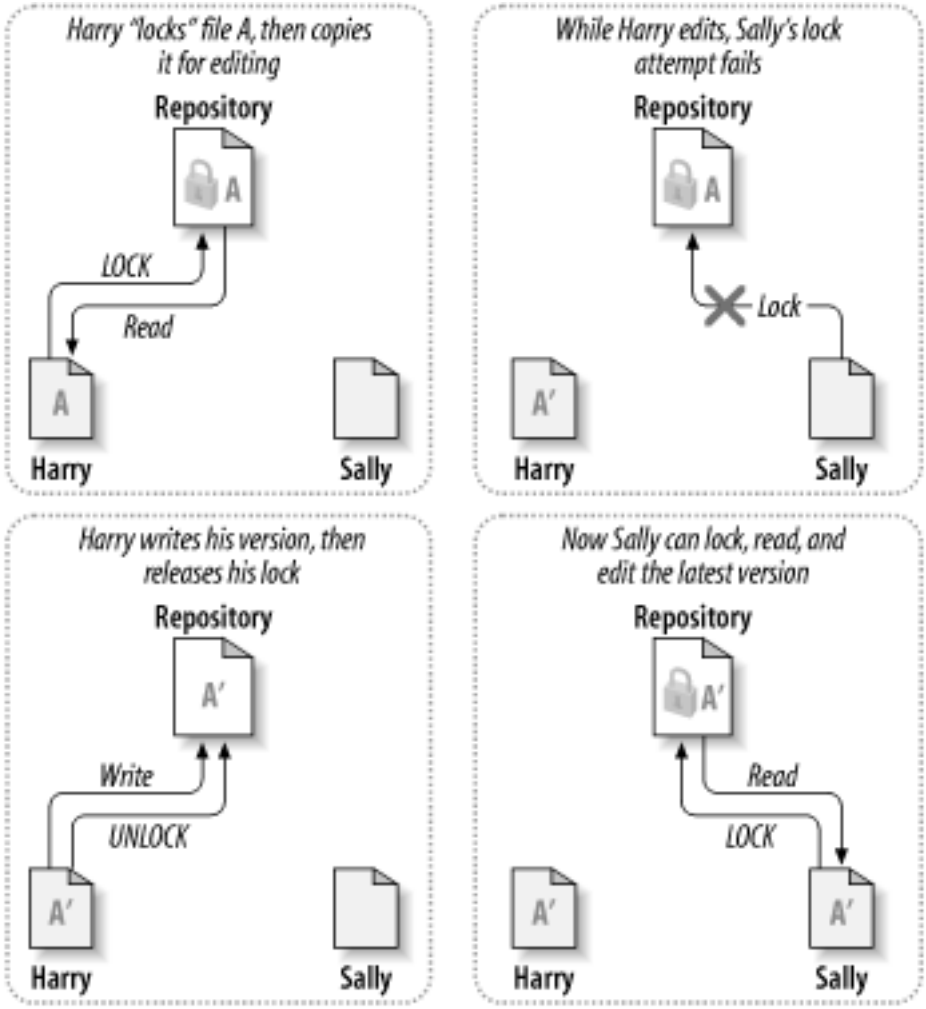
图 2.2. 需要避免的问题



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。

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

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

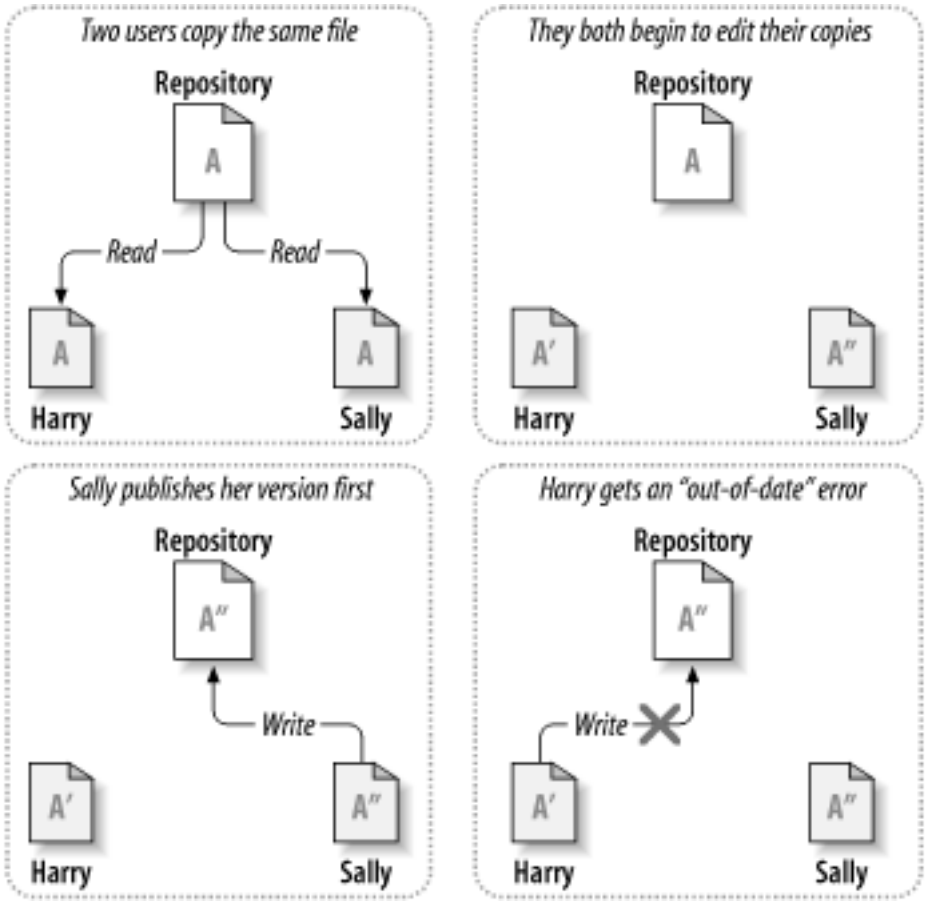
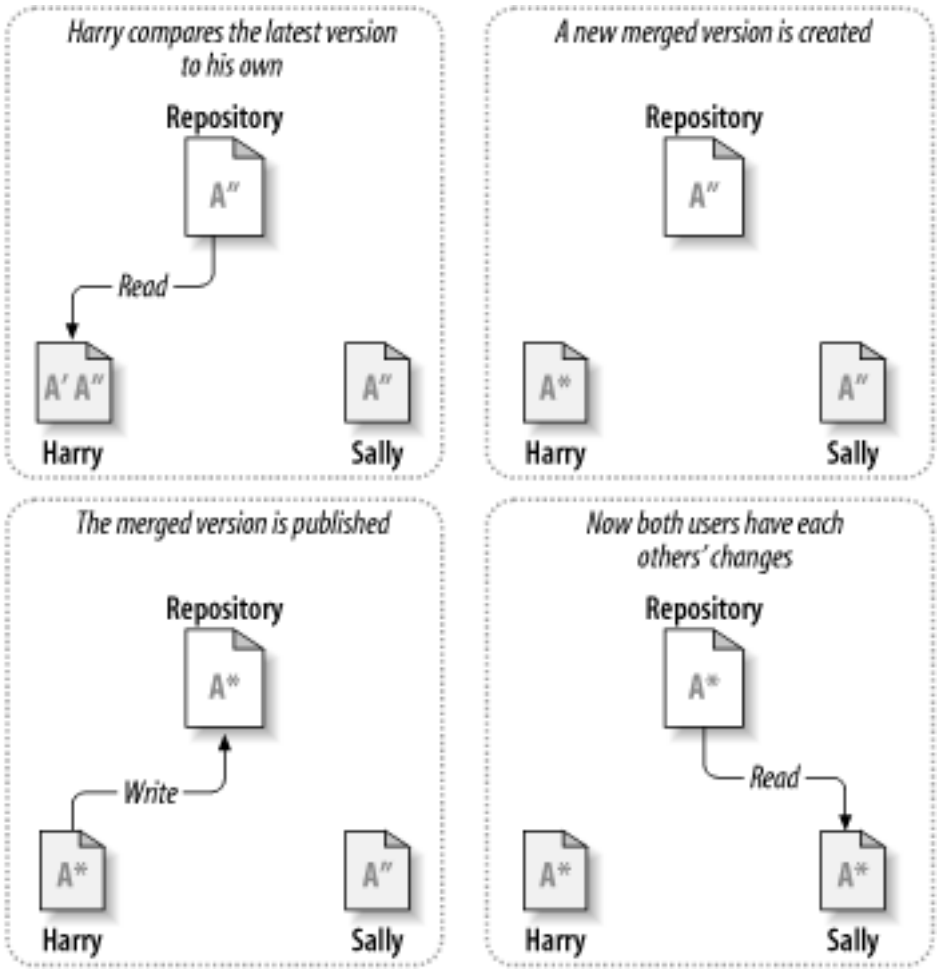


图 2.5. 复制-修改-合并 方案(续)



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。

There is one common situation where the lock-modify-unlock model comes out better, and that is where you have unmergeable files. For example if your repository contains some graphic images, and two people change the image at the same time, there is no way for those changes to be merged together. Either Harry or Sally will lose their changes.

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。

Subversion 实战

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。

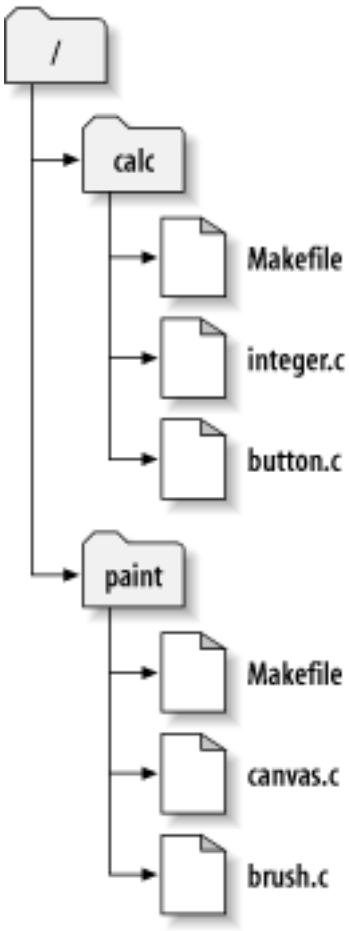
After you've made some changes to the files in your working copy and verified that they work properly, Subversion provides you with commands to publish your changes to the other people working with you on your project (by writing to the repository). If other people publish their own changes, Subversion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

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。

表 2.1. 版本库访问 URL

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://`	Unauthenticated TCP/IP access via custom protocol to a `svnserve` server.
`svn+ssh://`	authenticated, encrypted TCP/IP access via custom protocol to a `svnserve` server.

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`.

另外，在 Windows 平台的 `file:` 方案有时候需要使用一种非正式的"标准"协议访问本地但不是程序运行磁盘的版本库，以下两种 URL 路径语法都可以工作，其中的 `X` 是版本库所在的磁盘:

```
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。

You can safely access a FSFS repository via a network share, but you cannot access a BDB repository in this way.

> **警告**
>
> 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。

修订版本

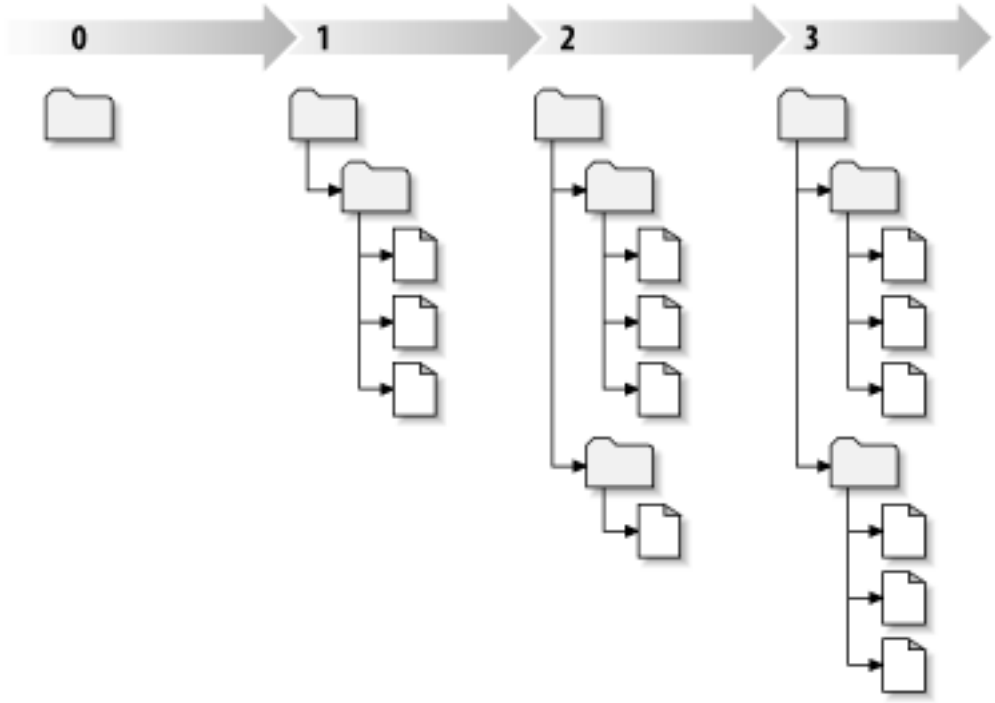
A svn commit operation can publish changes to any number of files and directories as a single atomic transaction. In your working copy, you can change files' contents, create, delete, rename and copy files and directories, and then commit the complete set of changes as a unit.

In the repository, each commit is treated as an atomic transaction: either all the commits changes take place, or none of them take place. Subversion retains this atomicity in the face of program crashes, system crashes, network problems, and other users' actions.

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。

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
: The file has been changed in the working directory, and no changes to that file have been committed to the repository since its base revision. There are local changes that have not been committed to the repository, thus a commit of the file will succeed in publishing your changes, and an update of the file will do nothing.

本地未修改且过时
: The file has not been changed in the working directory, but it has been changed in the repository. The file should eventually be updated, to make it current with the public revision. A commit of the file will do nothing, and an update of the file will fold the latest changes into your working copy.

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.

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 配置服务器

为了使用TortoiseSVN(或任何其他Subversion客户端)，你需要放置你的版本库，你可以将版本库存于本地并使用file://协议访问，也可以放置于服务器，使用http://或svn://访问，这两种服务器协议也可以被加密，使用https://或svn+ssh://。本章介绍了在Windows主机上设置一个服务器的步骤。

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/1.4/svn.serverconfig.html].

If you don't have a server and you work alone then local repositories are probably your best choice. You can skip this chapter and go directly to 第 4 章 版本库.

If you were thinking about setting up a multi-user repository on a network share, think again. Read "Accessing a Repository on a Network Share"一节 to find out why we think this is a bad idea.

基于 Apache 的服务器

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。

安装 Apache

The first thing you need before installing Apache is a computer with Windows 2000, Windows XP+SP1, Windows 2003, Vista or Server 2008.

> 警告
>
> Please note that Windows XP without the service pack 1 will lead to bogus network data and could therefore corrupt your repository!

1. Download the latest version of the Apache web server from http://httpd.apache.org/download.cgi. Make sure that you download the version 2.2.x - the version 1.3.xx won't work!

   The msi installer for Apache can be found by clicking on `other files`, then browse to `binaries/win32`. You may want to choose the msi file `apache-2.2.x-win32-x86-openssl-0.9.x.msi` (the one that includes OpenSSL).

2. Once you have the Apache2 installer you can double click on it and it will guide you through the installation process. Make sure that you enter the server-URL correctly (if you don't have a DNS name

for your server just enter the IP-address). I recommend to install Apache for All Users, on Port 80, as a Service. Note: if you already have IIS or any other program running which listens on port 80 the installation might fail. If that happens, go to the programs directory, `\Apache Group\Apache2\conf` and locate the file `httpd.conf`. Edit that file so that `Listen 80` is changed to a free port, e.g. `Listen 81`. Then restart the installation - this time it should finish without problems.

3. Now test if the Apache web server is running correctly by pointing your web browser to `http://localhost/` - a preconfigured Website should show up.

   小心

   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)。

   If Apache does not have this permission set up, your users will get “Access denied” error messages, which show up in the Apache error log as error 500.

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
   ```

配置

Now you have set up Apache and Subversion, but Apache doesn't know how to handle Subversion clients like TortoiseSVN yet. To get Apache to know which URL will be used for Subversion repositories you have to edit the Apache configuration file (usually located in `c:\program files\apache group\apache2\conf\httpd.conf`) with any text editor you like (e.g. Notepad):

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. To create the `passwd` file, open the command prompt (DOS-Box) again, change to the `apache2` folder (usually `c:\program files\apache group\apache2`) and create the file by entering

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

This will create a file with the name `passwd` which is used for authentication. Additional users can be added with

```
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。

A short explanation of what you just entered:

表 3.1. Apache **`httpd.conf`** Settings

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	For Subversion version 1.3 and higher, this directive enables listing all the available repositories under `SVNParentPath`.
SVNParentPath D:\SVN	告诉Subversion需要查看的版本库位于`D:\SVN`之下
SVNIndexXSLT "/svnindex.xsl"	Used to make the browsing with a web browser prettier.
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

But that's just an example. There are many, many more possibilities of what you can do with the Apache web server.

- 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>
```

- Using a `passwd` file limits and grants access to all of your repositories as a unit. If you want more control over which users have access to each folder inside a repository you can uncomment the line

  ```
  #AuthzSVNAccessFile svnaccessfile
  ```

  and create a Subversion access file. Apache will make sure that only valid users are able to access your `/svn` location, and will then pass the username to Subversion's `AuthzSVNAccessFile` module so that it can enforce more granular access based upon rules listed in the Subversion access file. Note that paths are specified either as `repos:path` or simply `path`. If you don't specify a particular repository, that access rule will apply to all repositories under `SVNParentPath`. The format of the authorization-policy file used by `mod_authz_svn` is described in "路径为基础的授权"一节

- To make browsing the repository with a web browser 'prettier', uncomment the line

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

  and put the files `svnindex.xsl`, `svnindex.css` and `menucheckout.ico` in your document root directory (usually `C:/Program Files/Apache Group/Apache2/htdocs`). The directory is set with the `DocumentRoot` directive in your Apache config file.

  你可以直接在我们的代码库http://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/contrib/other/svnindex中拿到这三个文件。如果访问这个链接需要认证，输入用户名称 `guest`，无需密码。

  The XSL file from the TortoiseSVN repository has a nice gimmick: if you browse the repository with your web browser, then every folder in your repository has an icon on the right shown. If you click on that icon, the TortoiseSVN checkout dialog is started for this URL.

多版本库

If you used the `SVNParentPath` directive then you don't have to change the Apache config file every time you add a new Subversion repository. Simply create the new repository under the same location as the first repository and you're done! In my company I have direct access to that specific folder on the server via SMB (normal windows file access). So I just create a new folder there, run the TortoiseSVN command TortoiseSVN → Create repository here... and a new project has a home...

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就会显示所有版本库列表。

路径为基础的授权

The `mod_authz_svn` module permits fine-grained control of access permissions based on user names and repository paths. This is available with the Apache server, and as of Subversion 1.3 it is available with svnserve as well.

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。

Authentication and authorization are separate processes. If a user wants to gain access to a repository path, she has to meet both, the usual authentication requirements and the authorization requirements of the access file.

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。

But there's a solution for that problem - at least if you're accessing the repository from inside a LAN with a windows domain controller: `mod_auth_sspi`!

The original SSPI module was offered by Syneapps including source code. But the development for it has been stopped. But don't despair, the community has picked it up and improved it. It has a new home on SourceForge [http://sourceforge.net/projects/mod-auth-sspi/].

- Download the module which matches your apache version, then copy the file `mod_auth_sspi.so` into the Apache modules folder.
- Edit the Apache config file: add the line

  ```
  LoadModule sspi_auth_module modules/mod_auth_sspi.so
  ```

  to the `LoadModule` section. Make sure you insert this line before the line

  ```
  LoadModule auth_module modules/mod_auth.so
  ```

- To make the Subversion location use this type of authentication you have to change the line

  ```
  AuthType Basic
  ```

  to

  ```
  AuthType SSPI
  ```

  also you need to add

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

within the `<Location /svn>` block. If you don't have a domain controller, leave the name of the domain control as `<domaincontroller>`.

Note that if you are authenticating using SSPI, then you don't need the `AuthUserFile` line to define a password file any more. Apache authenticates your username and password against your windows domain instead. You will need to update the users list in your `svnaccessfile` to reference `DOMAIN\username` as well.

重要

The SSPI authentication is only enabled for SSL secured connections (https). If you're only using normal http connections to your server, it won't work.

To enable SSL on your server, see the chapter: "用 SSL 使服务器更安全"一节

提示

Subversion `AuthzSVNAccessFile` files are case sensitive in regard to user names (`JUser` is different from `juser`).

In Microsoft's world, Windows domains and user names are not case sensitive. Even so, some network administrators like to create user accounts in CamelCase (e.g. `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。

The end result is that you may need at least two entries in your `AuthzSVNAccessFile` for each user -- a lowercase entry and an entry in the same case that Internet Explorer passes to Apache. You will also need to train your users to also type in their credentials using lower case when accessing repositories via TortoiseSVN.

Apache's Error and Access logs are your best friend in deciphering problems such as these as they will help you determine the username string passed onto Subversion's `AuthzSVNAccessFile` module. You may need to experiment with the exact format of the user string in the `svnaccessfile` (e.g. `DOMAIN\user` vs. `DOMAIN//user`) in order to get everything working.

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:

```
AuthAuthoritative 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
  AuthAuthoritative Off
  AuthUserFile passwd

  Require valid-user
</Location>
```

用 SSL 使服务器更安全

Even though Apache 2.2.x has OpenSSL support, it is not activated by default. You need to activate this manually.

1. In the apache config file, uncomment the lines:

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

   and at the bottom

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

   then change the line (on one line)

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

   to

   ```
   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 bin\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:

```
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. Restart the Apache service.
4. 将你的浏览器指向`https://servername/svn/project` ...

SSL and 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:

- define a separate `<Location /path>` directive in the Apache config file, and add the `SSPIBasicPreferred On`. This will allow IE to authenticate again, but other browsers and Subversion won't be able to authenticate against that location.
- 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。
- In the SSL "standard" setup there's often the following statement in Apache's virtual SSL host:

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

There are (were?) good reasons for this configuration, see http://www.modssl.org/docs/2.8/ssl_faq.html#ToC49 But if you want NTLM authentication you have to use `keepalive`. If You uncomment the whole `SetEnvIf` you should be able to authenticate IE with windows authentication over SSL against the Apache on Win32 with included `mod_auth_sspi`.

强制 SSL 访问

When you've set up SSL to make your repository more secure, you might want to disable the normal access via non-SSL (http) and only allow https access. To do this, you have to add another directive to the Subversion `<Location>` block: `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>
```

Using client certificates with virtual SSL hosts

Sent to the TortoiseSVN mailing list by Nigel Green. Thanks!

In some server configurations you may need to setup a single server containing 2 virtual SSL hosts: The first one for public web access, with no requirement for a client certificate. The second one to be secure with a required client certificate, running a Subversion server.

Adding an `SSLVerifyClient Optional` directive to the per-server section of the Apache configuration (i.e. outside of any `VirtualHost` and `Directory` blocks) forces Apache to request a client Certificate in the initial SSL handshake. Due to a bug in `mod_ssl` it is essential that the certificate is requested at this point as it does not work if the SSL connection is re-negotiated.

The solution is to add the following directive to the virtual host directory that you want to lock down for Subversion:

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

This directive grants access to the directory only if a client certificate was received and verified successfully.

To summarise, the relevant lines of the Apache configuration are:

```
SSLVerifyClient Optional

### Virtual host configuration for the PUBLIC host
### (not requiring a certificate)

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

### Virtual host configuration for SUBVERSION
### (requiring a client certificate)
<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>
```

基于 svnserve 的服务器

简介

有一些情况下，不能使用Apache作为你的服务器，Subversion包括Svnserve - 一个轻型的独立服务器，使用普通TCP/IP连接之上的自定义协议。

大多数情况下svnserve的设置更加简单，也比Apache的服务器更加快。

安装 svnserve

1. 从http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91得到最新版本的Subversion。
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`，`libeay32.dll`和`ss`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`。

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"配置"一节。

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。

You can also increase security and save time entering URLs with svnserve by using the --root switch to set the root location and restrict access to a specified directory on the server:

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

Using the previous test as a guide, svnserve would now run as:

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

And in TortoiseSVN our repo-browser URL is now shortened to:

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

Note that the --root switch is also needed if your repository is located on a different partition or drive than the location of svnserve on your server.

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。

> **警告**
>
> 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。

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。

> ### 提示
>
> Microsoft now recommend services to be run as under either the Local Service or Network Service account. Refer to The Services and Service Accounts Security Planning Guide [http://www.microsoft.com/technet/security/topics/serversecurity/serviceaccount/default.mspx]. To create the service under the Local Service account, append the following to the example above.
>
> ```
> obj= "NT AUTHORITY\LocalService"
> ```
>
> Note that you would have to give the Local Service account appropriate rights to both Subversion and your repositories, as well as any applications which are used by hook scripts. The built-in group for this is called "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]。

If you installed an earlier version of svnserve using the `SVNService` wrapper, and you now want to use the native support instead, you will need to unregister the wrapper as a service (remember to stop the service first!). Simply use the command

```
svnservice -remove
```

to remove the service registry entry.

svnserve 的认证

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。

One way to overcome this is to create a password database:

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

Where `userfile` is a file which exists in the same directory as `svnserve.conf`. This file can live elsewhere in your file system (useful for when you have multiple repositories which require the same access rights) and may be referenced using an absolute path, or a path relative to the `conf` directory. If you include a path, it must be written `/the/unix/way`. Using \ or drive letters will not work. The `userfile` should have a structure of:

```
[users]
username = password
...
```

This example would deny all access for unauthenticated (anonymous) users, and give read-write access to users listed in `userfile`.

> 提示
>
> 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/1.4/svn.serverconfig.svnserve.html#svn.serverconfig.svnserve.auth.users] and Client Credentials Caching [http://svnbook.red-bean.com/en/1.4/svn.serverconfig.netmodel.html#svn.serverconfig.netmodel.credcache]

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。

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授权用户运行，为此，你需要在你的服务器上有SSH守护进程。

It is beyond the scope of this documentation to detail the installation and setup of a secure shell, however you can find further information in the TortoiseSVN FAQ, SSH How-To [http://tortoisesvn.net/ssh_howto]. You can find other SSH topics within the FAQ by searching for "SSH".

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]。

svnserve 以路径为基础的授权

从Subversion1.3开始，svnserve支持与`mod_authz_svn`相同的路径为基础的授权模式，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"路径为基础的授权"一节。

Cyrus SASL Support

Starting with Subversion 1.5.0, svnserve and TortoiseSVN support Cyrus SASL (Simple Authentication and Security Layer) for the `svn://` protocol. You can find further information in this subversion app note: Using Cyrus SASL Authentication with Subversion [http://svn.collab.net/repos/svn/trunk/notes/sasl.txt].

第 4 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。

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/1.4/svn.reposadmin.planning.html#svn.reposadmin.basics.backends] in the Subversion book for more information.

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
```

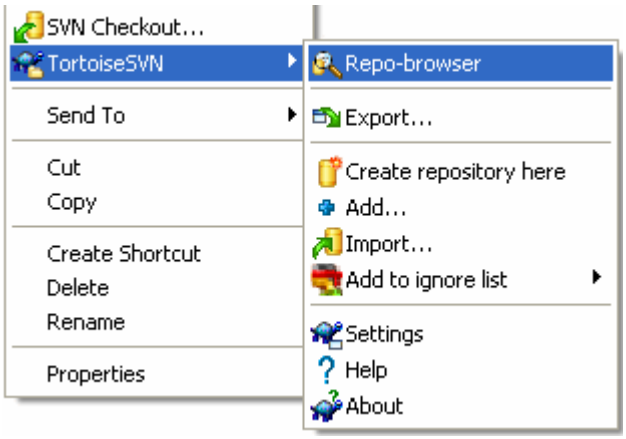
或

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

现在你在`D:\SVN\MyNewRepository`创建了一个新的版本库。

使用 TortoiseSVN 创建版本库

图 4.1. 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。

   > **提示**
   >
   > 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。
   >
   > 由于这些兼容性问题，将来的 TortoiseSVN 不再支持使用 `file:///` 访问 BDB 版本库，尽管我们继续支持使用 `svn://`，`http://` 或 `https://` 协议访问 BDB 版本库。因此，我们强烈建议任何新的必须使用 `file:///` 协议访问的版本库使用 FSFS 格式。
   >
   > 当然除了本地测试之外，我们也建议你根本不要使用 `file:///` 协议访问。除非是单个开发人员，使用服务器模型更为安全可靠。

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。

> **警告**
>
> Do not create or access a Berkeley DB repository on a network share. It cannot exist on a remote file system. Not even if you have the network drive mapped to a drive letter. If you attempt to use Berkeley DB on a network share, the results are unpredictable - you may see mysterious errors right away, or it may be months before you discover that your repository database is subtly corrupted.

Accessing a Repository on a Network Share

Although in theory it is possible to put a FSFS repository on a network share and have multiple users access it using `file:///` protocol, this is most definitely not recommended. In fact we would strongly discourage it, and do not support such use.

Firstly you are giving every user direct write access to the repository, so any user could accidentally delete the entire repository or make it unusable in some other way.

Secondly not all network file sharing protocols support the locking that Subversion requires, so you may find your repository gets corrupted. It may not happen straight away, but one day two users will try to access the repository at the same time.

Thirdly the file permissions have to be set just so. You may just about get away with it on a native Windows share, but SAMBA is particularly difficult.

`file:///` access is intended for local, single-user access only, particularly testing and debugging. When you want to share the repository you really need to set up a proper server, and it is not nearly as difficult as you might think. Read 第 3 章 配置服务器 for guidelines on choosing and setting up a server.

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。

`svnadmin`在安装Subversion命令行客户端时会自动安装，如果你在Windows PC上安装这个命令行工具，最好的方式是下载Windows安装版本，它比`.zip`版本的压缩效率更好，所以更小，并且小心的为你设定路径。你可以从http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91下载最新版本的命令行客户端。

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/1.4/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 "客户端钩子脚本"一节 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。

检出链接

If you want to make your Subversion repository available to others you may want to include a link to it from your website. One way to make this more accessible is to include a checkout link for other TortoiseSVN users.

When you install TortoiseSVN, it registers a new `tsvn:` protocol. When a TortoiseSVN user clicks on such a link, the checkout dialog will open automatically with the repository URL already filled in.

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

```
<a href="tsvn:https://tortoisesvn.tigris.org/svn/tortoisesvn/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:https://tortoisesvn.tigris.org/svn/tortoisesvn/trunk">
<img src=TortoiseCheckout.png></a>
```

第 5 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 。

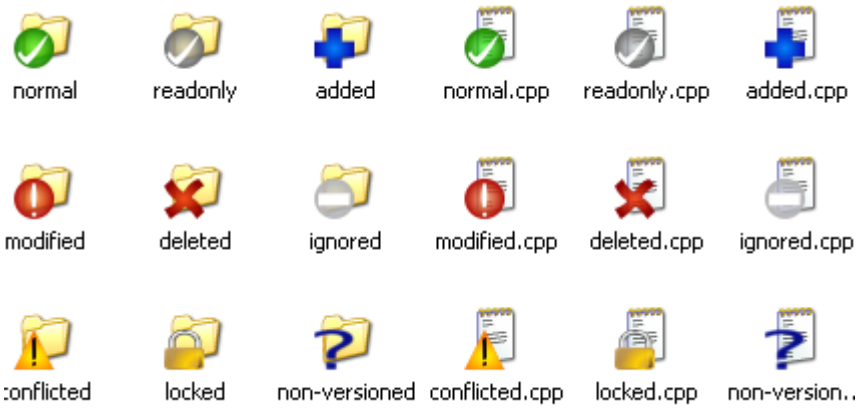
为了获得比每日用户指南更多的信息:

- 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。

开始

图标重载

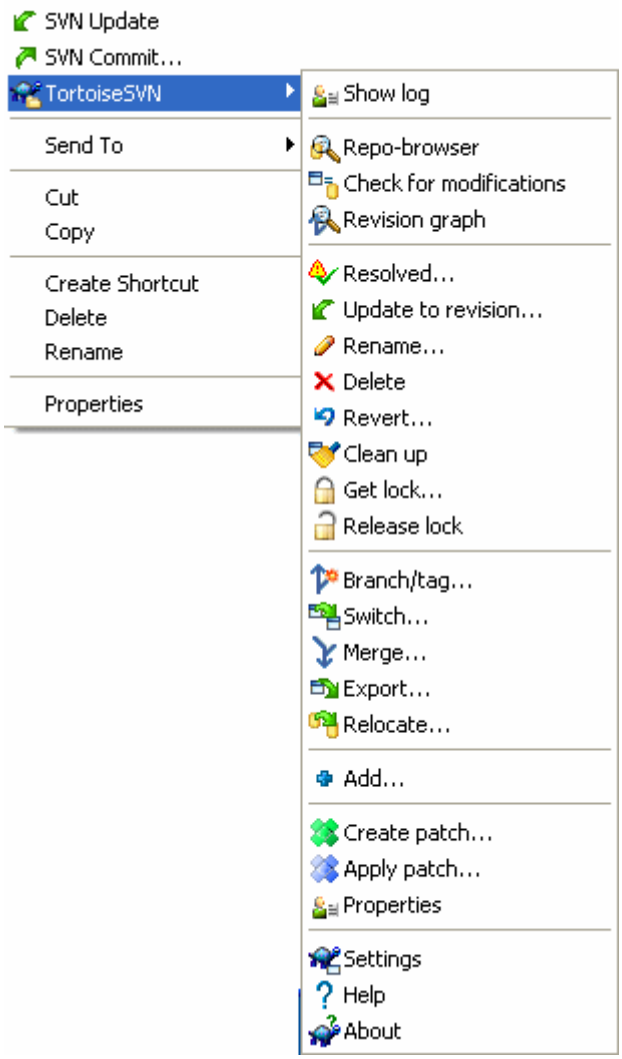
图 5.1. 显示重载图标的资源管理器



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 "图标重载"一节 查阅不同的重载图标含义。

右键菜单

图 5.2. 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。

提示

Some commands which are very rarely used are only available in the extended context menu. To bring up the extended context menu, hold down the Shift key when you right-click.

在某些情况下，你可能看到多个TortoiseSVN条目。这不是BUG！

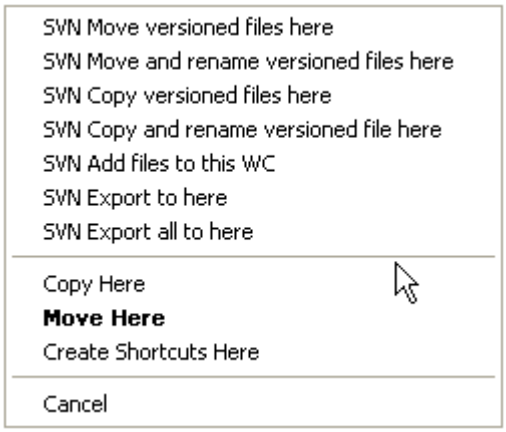
图 5.3. 在一个版本控制的文件夹下资源管理器文件菜单中的快捷方式。

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。

If you are using Windows 2000 you will find that the context menus are shown as plain text, without the menu icons shown above. We are aware that this was working in previous versions, but Microsoft has changed the way its icon handlers work for Vista, requiring us to use a different display method which unfortunately does not work on Windows 2000.

拖放

图 5.4. 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。

常用快捷方式

Some common operations have well-known Windows shortcuts, but do not appear on buttons or in menus. If you can't work out how to do something obvious, like refreshing a view, check here.

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。

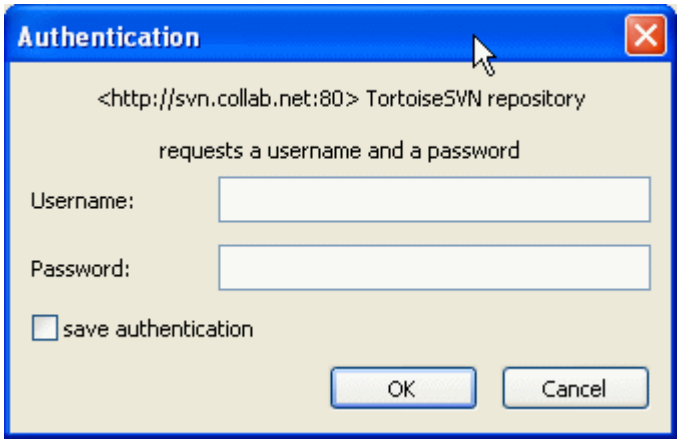
Ctrl-A
: 全选。可用于在得到一个错误消息并想要复制粘贴到电子邮件时。使用Ctrl-A to选择错误错误，然后...

Ctrl-C
: ... 复制选中的文本。

认证

If the repository that you are trying to access is password protected, an authentication Dialog will show up.

图 5.5. 认证对话框



Enter your username and password. The checkbox will make TortoiseSVN store the credentials in Subversion's default directory: `%APPDATA%\Subversion\auth` in three subdirectories:

- `svn.simple`文件里包含了基本认证方式所需要的认证信息(用户名/口令)。
- `svn.ssl.server`文件里包含了SSL服务器证书。
- `svn.username`文件里包含了用户名认证的认证信息(不需要提供密码)。

If you want to clear the authentication cache for all servers, you can do so from the Saved Data page of TortoiseSVN's settings dialog. That button will clear all cached authentication data from the Subversion `auth` directories, as well as any authentication data stored in the registry by earlier versions of TortoiseSVN. Refer to "已保存数据的设置"一节.

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 章 配置服务器

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。

导入数据到版本库

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。

所以，如果你还没创建一个基本的文件夹结构到你的版本库中，你现在可以这样做:

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
```

你还可以使用版本库浏览器直接在版本库中创建文件夹。

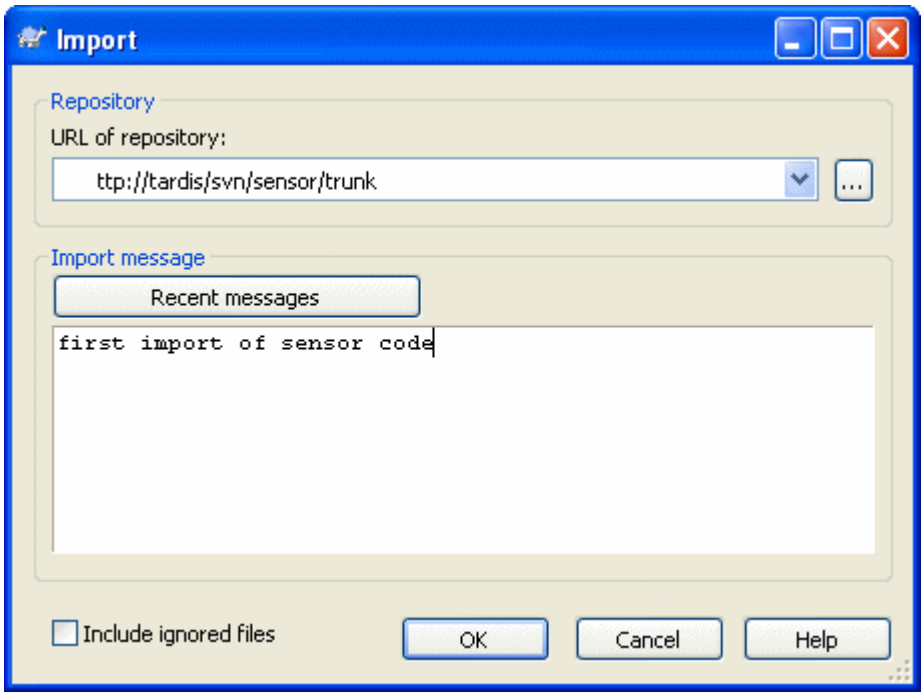
导入

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. Organize the files in folders and sub-folders. Although it is possible to rename/move files later it is highly recommended to get your project's structure straight before importing!

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 5.6. 导入对话框



在这个对话框中，输入你的项目导入到版本库的URL。

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"常规设置"一节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。如前所述，你导入的文件夹名称不会在版本库中出现，只有这个文件夹的内容会在版本库中出现。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目录。或者继续阅读，找到如何导入文件夹到合适的位置。

导入适当的位置

Subversion 导入命令功能有限 - 你不能容易的选择导入的项目。而且，你导入的文件夹不会成为 Subversion 工作副本 - 你不得不单独做检出。但是不要担心，有其它方法克服这些缺陷 :-)

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。

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。

In such cases we suggest to use template files. You create a file which contains all the data your developers will need, add that file to version control and let the developers check this file out. Then, each developer has to make a copy of that file and rename that copy. After that, modifying the copy is not a problem anymore.

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。

引用的工程

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，你需要不同的子目录来自版本库的不同位置。，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。

Let's say you check out a working copy of `/project1` to `D:\dev\project1`. Select the folder `D:\dev\project1`, right click and choose Windows Menu → Properties from the context menu. The Properties Dialog comes

up. Then go to the Subversion tab. There, you can set properties. Click Add.... Select the `svn:externals` property from the combobox and write in the edit box the repository URL in the format `name url` or if you want to specify a particular revision, `name -rREV url` You can add multiple external projects, 1 per line. Note that URLs must be properly escaped or they will not work properly. For example you must replace each space with `%20`. Note that it is not possible to use folder names with spaces in them. Suppose that you have set these properties on `D:\dev\project1`:

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

Now click Set and commit your changes. When you (or any other user) update your working copy, Subversion will create a sub-folder `D:\dev\project1\sounds` and checkout the sounds project, another sub-folder `D:\dev\project1\quick graphs` containing the graphics project, and finally a nested sub-folder `D:\dev\project1\skins\toolkit` containing revision 21 of the skin-maker project.

提示

You should strongly consider using explicit revision numbers in all of your externals definitions, as described above. Doing so means that you get to decide when to pull down a different snapshot of external information, and exactly which snapshot to pull. Besides the common sense aspect of not being surprised by changes to third-party repositories that you might not have any control over, using explicit revision numbers also means that as you backdate your working copy to a previous revision, your externals definitions will also revert to the way they looked in that previous revision, which in turn means that the external working copies will be updated to match they way they looked back when your repository was at that previous revision. For software projects, this could be the difference between a successful and a failed build of an older snapshot of your complex code base.

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项目的修改。

Note that if you change the URL in an `svn:externals` property, then next time you update your working copy Subversion will delete the old external folder and make a fresh checkout, so you will see files being Added, rather than being Updated as you might have expected. This situation might occur if you reference a tag from another project. When a new version of that project is released, you change your external reference to point to the new tag.

外部定义`svn:externals`中的URL是绝对路径。如果你重新定位工作副本，或者外部版本库重新定位了，这些URL不会自动更新。而且，如果你分支了一个工程，它的外部定义位于同一版本库中，分支中的这些URL也不会更新；你可能需要用对分支的引用替换对最新版本的引用。

如果你需要TortoiseSVN如何处理属性的更多信息，请阅读"项目设置"一节。

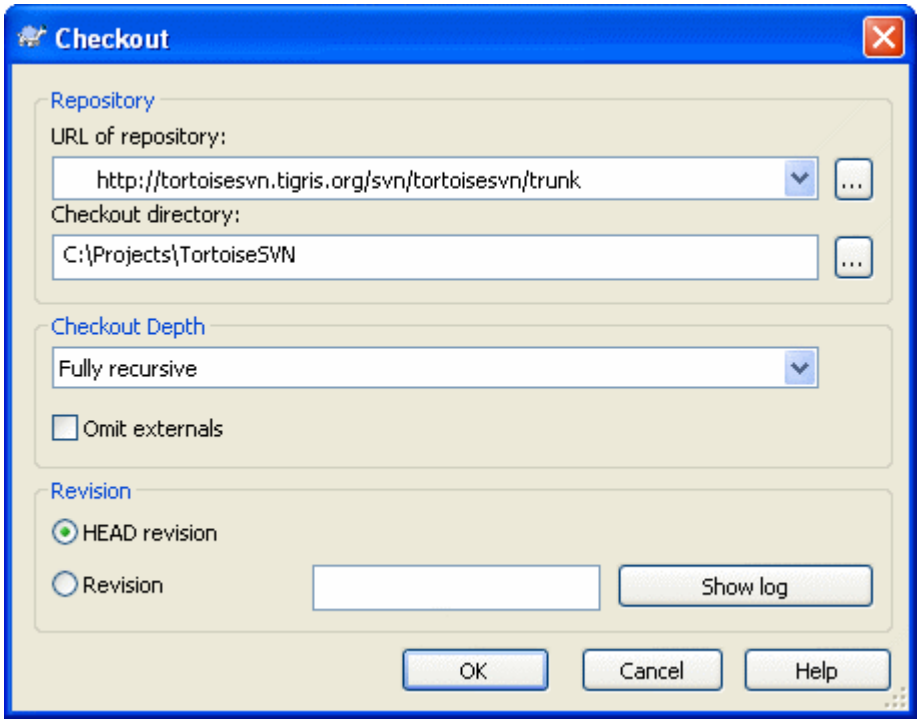
如果你需要知道存取公共子个我我你工程的不同方法，请阅读"包含一个普通的子项目"一节.

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 5.7. 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。

Checkout Depth

You can choose the depth you want to checkout, which allows you to specify the depth of recursion into child folders. If you want just a few sections of a large tree, You can checkout the top level folder only, then update selected folders recursively.

Fully recursive
: Checkout the entire tree, including all child folders and sub-folders.

Immediate children, including folders
: Checkout the specified directory, including all files and child folders, but do not populate the child folders.

Only file children
: Checkout the specified directory, including all files but do not checkout any child folders.

Only this item
: Checkout the directory only. Do not populate it with files or child folders.

工作副本
: Retain the depth specified in the working copy. This option is not used in the checkout dialog, but it is the default in all other dialogs which have a depth setting.

If you check out a sparse working copy (i.e., by choosing something other than `fully recursive` for the checkout depth), you can fetch additional sub-folders by using the repository browser (“版本库浏览器”一节) or the check for modifications dialog (“本地与远程状态”一节).

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.

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

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

重要

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.

It is recommended that you check out only the `trunk` part of the directory tree, or lower. If you specify the parent path of the directory tree in the URL then you might end up with a full hard disk since you will get a copy of the entire repository tree including every branch and tag of your project!

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"导出一个Subversion工作副本"一节。

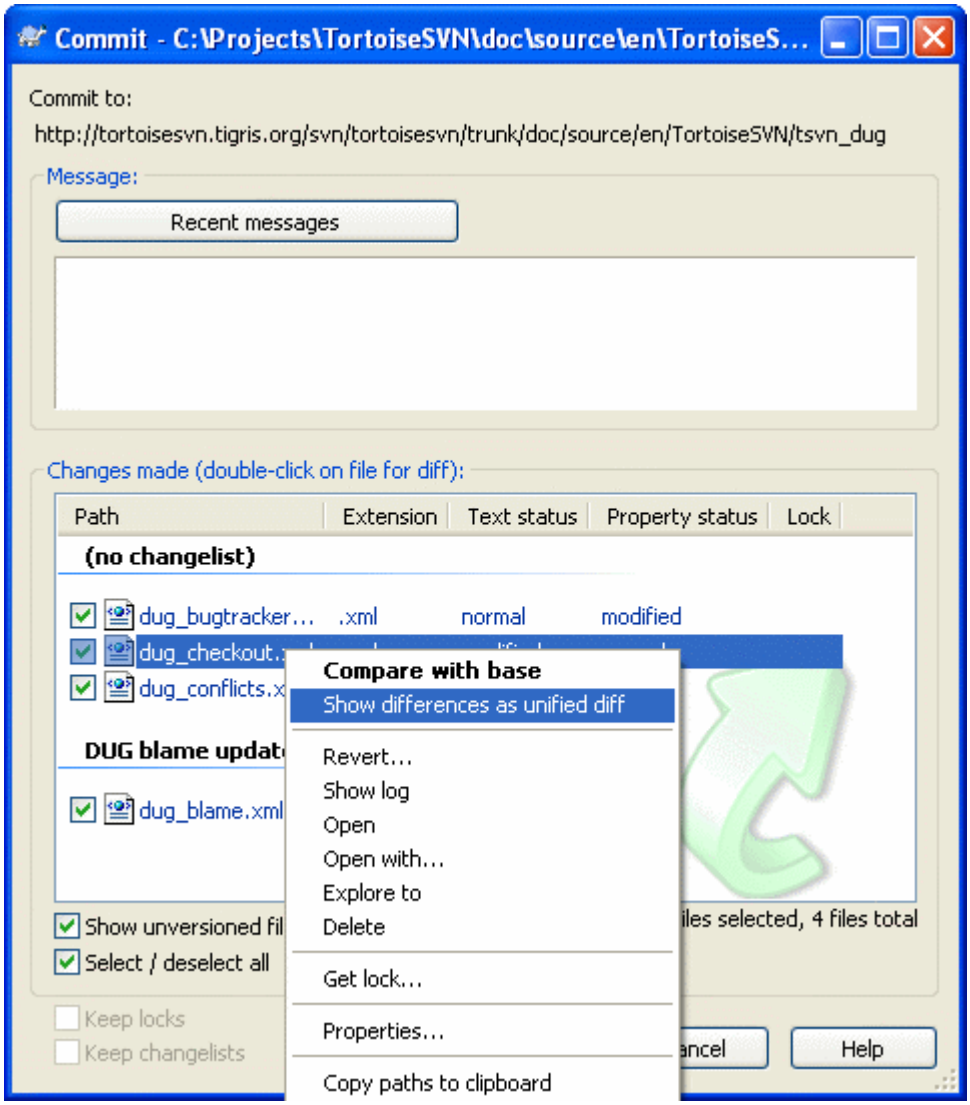
将你的修改提交到版本库

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。

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 5.8. 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

If you think that the commit dialog shows you too many unversioned (e.g. compiler generated or editor backup) files, there are several ways to handle this. You can:

- 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。

Read "忽略文件和目录"一节 for more information.

Double clicking on any modified file in the commit dialog will launch the external diff tool to show your changes. The context menu will give you more options, as shown in the screenshot. You can also drag files from here into another application such as a text editor or an 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。

By default when you commit changes, any locks that you hold on files are released automatically after the commit succeeds. If you want to keep those locks, make sure the Keep locks checkbox is checked. The default state of this checkbox is taken from the `no_unlock` option in the Subversion configuration file. Read "常规设置"一节 for information on how to edit the Subversion configuration file.

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。

修改列表

The commit dialog supports Subversion's changelist feature to help with grouping related files together. Find out about this feature in "修改列表"一节.

Excluding Items from the Commit List

Sometimes you have versioned files that change frequently but that you really don't want to commit. Sometimes this indicates a flaw in your build process - why are those files versioned? should you be using template files? But occasionally it is inevitable. A classic reason is that your IDE changes a timestamp in the project file every time you build. The project file has to be versioned as it includes all the build settings, but it doesn't need to be committed just because the timestamp changed.

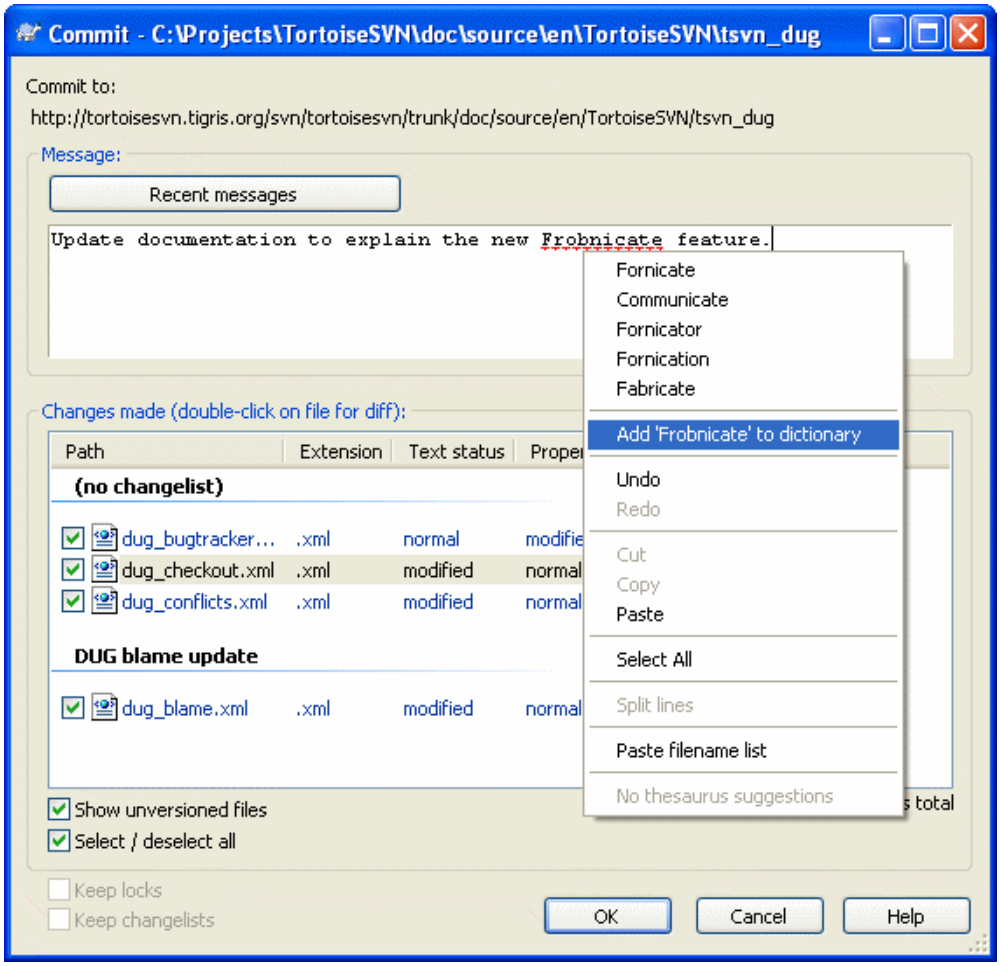
To help out in awkward cases like this, we have reserved a changelist called `ignore-on-commit`. Any file added to this changelist will automatically be unchecked in the commit dialog. You can still commit changes, but you have to select it manually in the commit dialog.

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##采用这些样式，使用*##*表示粗体，_##_表示下划线，^##^表示斜体。

图 5.9. 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/.

You can re-use previously entered log messages. Just click on Recent messages to view a list of the last few messages you entered for this working copy. The number of stored messages can be customized in the TortoiseSVN settings dialog.

You can clear all stored commit messages from the Saved data page of TortoiseSVN's settings, or you can clear individual messages from within the Recent messages dialog using the Delete key.

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"项目设置"一节以了解详情。

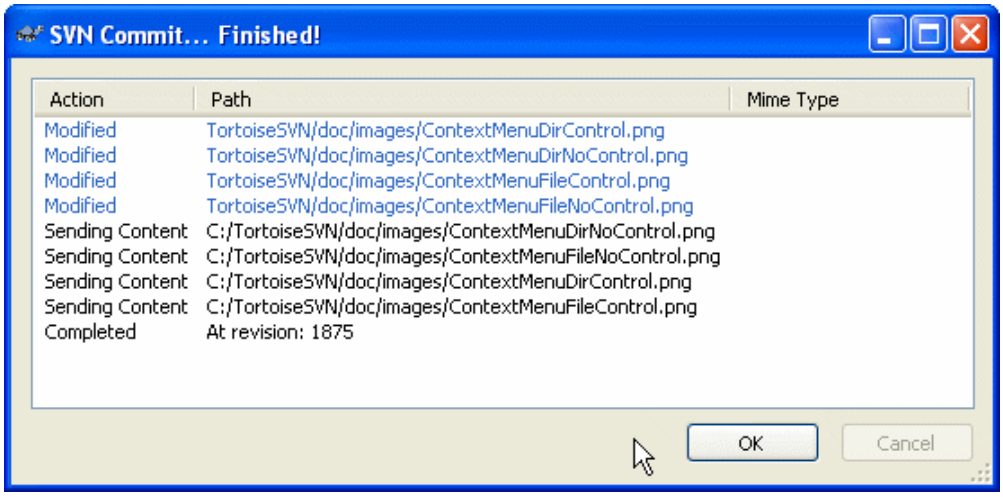
Integration with Bug Tracking Tools

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 "Integration with Bug Tracking Systems / Issue Trackers"一节.

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 5.10. 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。

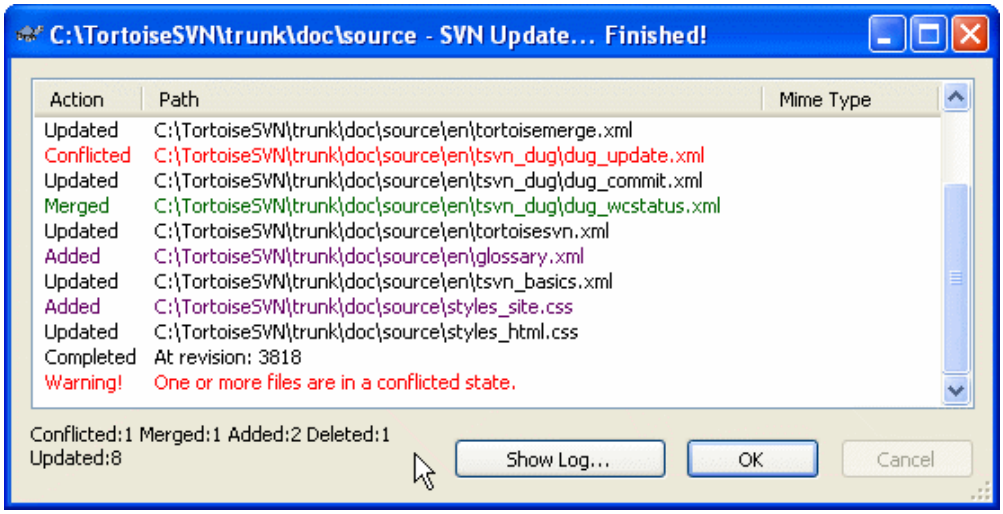
深红
: 提交一个删除或是替换。

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"TortoiseSVN 颜色设置"一节获得详情。

用来自别人的修改更新你的工作副本

图 5.11. 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"TortoiseSVN 颜色设置"一节获得详情。

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 "Checkout Depth"一节. 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`).

小心

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 "Roll back (Undo) revisions in the repository"一节 for further instructions, and alternative methods.

更新到版本在你偶尔要看看你的项目在早前某时刻是什么样子的时候很有用。但通常，更新单个文件到之前的版本不是一个好主意，因为这会使你的工作副本处于不一致的状态。如果你要更新的文件已经改了名，你可能甚至发现该文件从你的工作副本中消失了，因为早期的版本中不存在这个名字的文件。如果你只是简单地想要一个旧版本文件的本地副本，最好是在该文件的日志对话框中使用右键菜单 → 另存版本为...命令。

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。

解决冲突

有时当你从版本库中更新你的文件时，会有冲突。冲突出现的原因是两个开发人员修改了文件中相同的几行。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:

```
<<<<<<< filename
    ####
=======
    #########
>>>>>>> revision
```

对于每个冲突的文件Subversion在你的目录下放置了三个文件:

filename.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。

filename.ext.rOLDREV
: 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。

filename.ext.rNEWREV
: 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。这个文件对应与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。

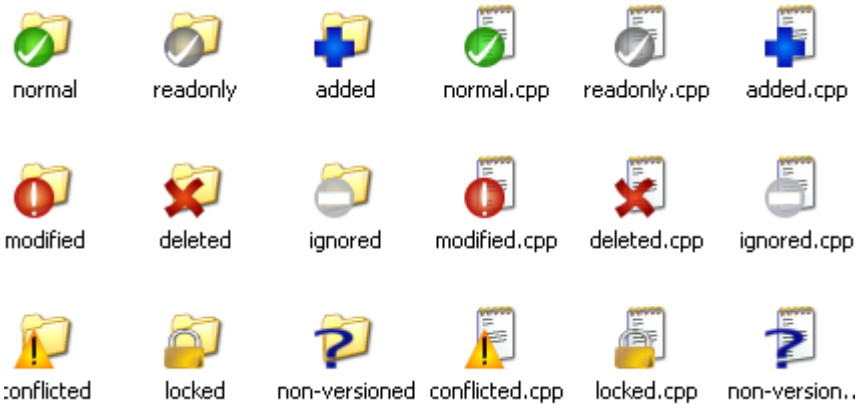
你可以右击父文件夹，选择TortoiseSVN → 已解决...，使用"已解决"命令来解决多个文件。这个操作会出现一个对话框，列出文件夹下所有有冲突的文件，你可以选择将哪些标记成已解决。

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。

图标重载

图 5.12. 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。


A fresh checked out working copy has a green checkmark as overlay. That means the Subversion status is normal.


As soon as you start editing a file, the status changes to modified and the icon overlay then changes to a red exclamation mark. That way you can easily see which files were changed since you last updated your working copy and need to be committed.


If during an update a conflict occurs then the icon changes to a yellow exclamation mark.


If you have set the `svn:needs-lock` property on a file, Subversion makes that file read-only until you get a lock on that file. Such files have this overlay to indicate that you have to get a lock first before you can edit that file.

If you hold a lock on a file, and the Subversion status is normal, this icon overlay reminds you that you should release the lock if you are not using it to allow others to commit their changes to the file.

This icon shows you that some files or folders inside the current folder have been scheduled to be deleted from version control or a file under version control is missing in a folder.

The plus sign tells you that a file or folder has been scheduled to be added to version control.

The bar sign tells you that a file or folder is ignored for version control purposes. This overlay is optional.

This icon shows files and folders which are not under version control, but have not been ignored. This overlay is optional.

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 very limited and if you are also using an old version of TortoiseCVS, then 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 too.

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

For a description of how icon overlays correspond to Subversion status and other technical details, read "图标重载"一节.

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)。

Simply right click on one of the headings of a column, choose More... from the context menu displayed. A dialog will appear where you can specify the columns and their order, which is displayed in the "Detailed View". Scroll down until the entries starting with SVN come into view. Check the ones you would like to have displayed and close the dialog by pressing OK. The columns will be appended to the right of those currently displayed. You can reorder them by drag and drop, or resize them, so that they fit your needs.

重要

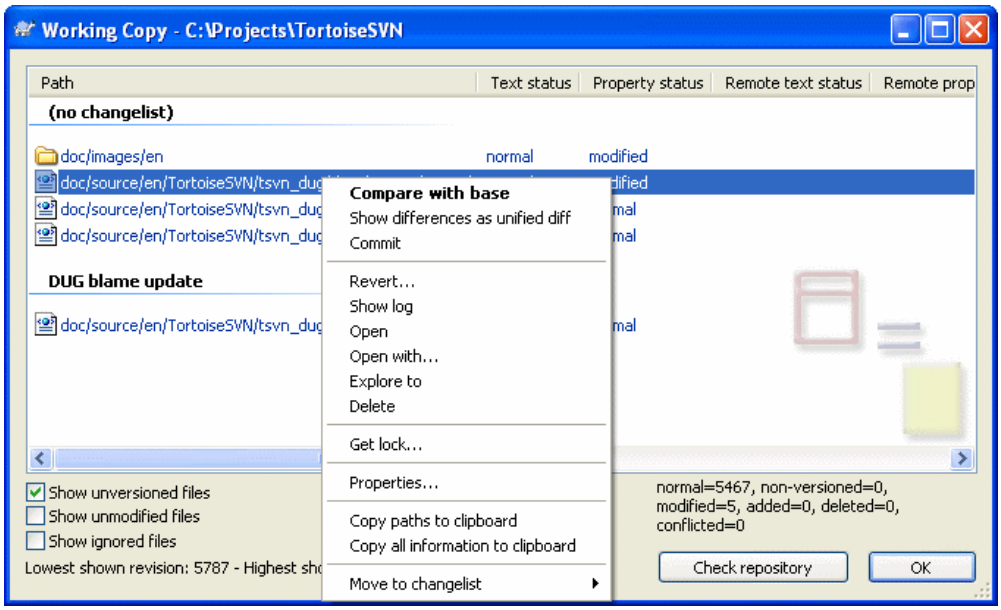
The additional columns in the Windows Explorer are not available on Vista, since Microsoft decided to not allow such columns for all files anymore but only for specific file types.

提示

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。

本地与远程状态

图 5.13. 检查所作的修改



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。

如果你点击检查版本库，那你还可以看到版本库里的改动。这样，你就可以在提交之前检查是否有存在冲突的可能。你也可以从版本库中更新选中的文件而用不着更新整个文件夹。

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"TortoiseSVN 颜色设置"一节获得详情。

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.

可以使用邮件菜单 → 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"修改列表"一节以获得更多信息。

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.

提示

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对话框。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。

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(##版本)。

提示

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。

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。

重要

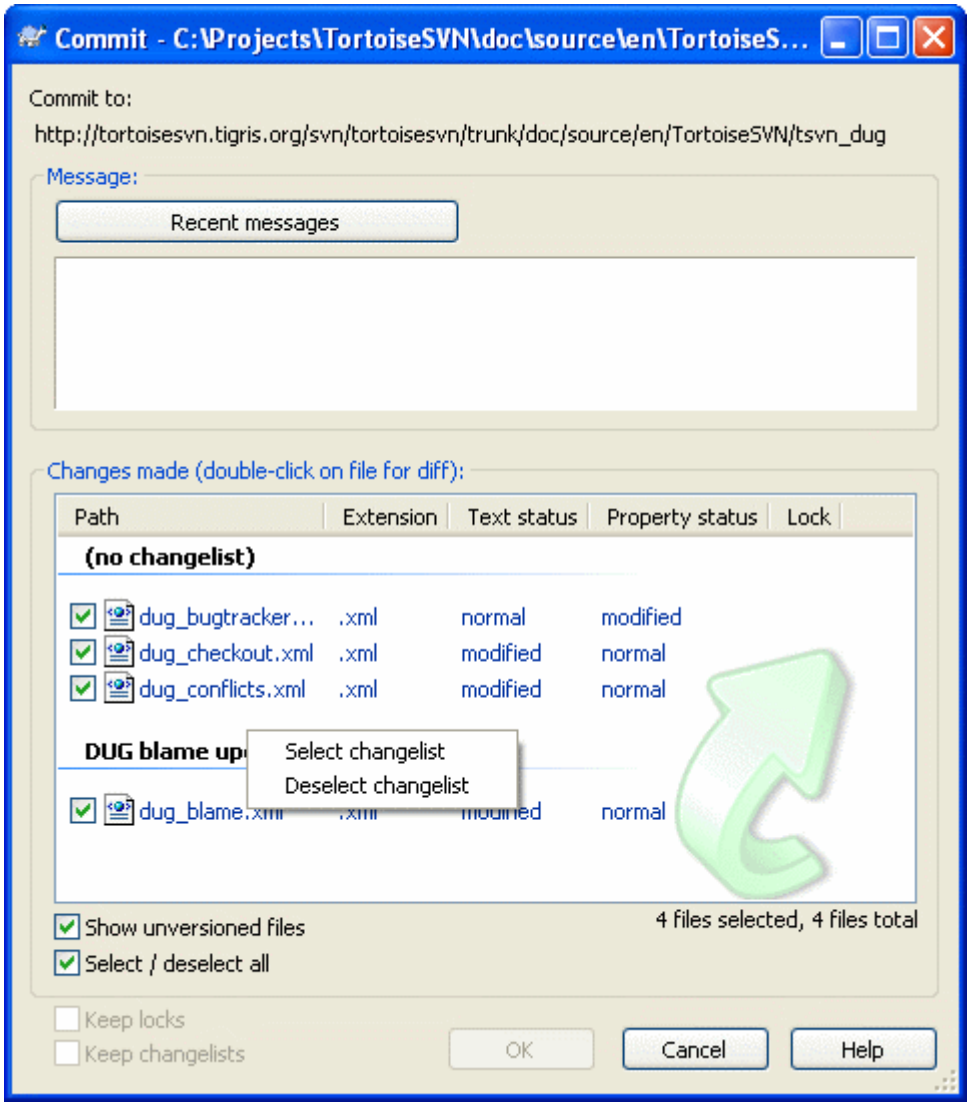
The changelist feature in TortoiseSVN is only available in Windows XP and later, as it depends on a shell capability which is not present in Windows 2000. Sorry, but Win2K is really quite old now, so please don't complain.

You can see changelists in several places, but the most important ones are the commit dialog and the check-for-modifications dialog. Let's start in the check-for-modifications dialog after you have worked on several features and many files. When you first open the dialog, all the changed files are listed together. Suppose you now want to organise things and group those files according to feature.

Select one or more files and use Context Menu → Move to changelist to add an item to a changelist. Initially there will be no changelists, so the first time you do this you will create a new changelist. Give it name which describes what you are using it for, and click OK. The dialog will now change to show groups of items.

Once you have created a changelist you can drag and drop items into it, either from another changelist, or from Windows Explorer. Dragging from Explorer can be useful as it allows you to add items to a changelist before the file is modified. You could do that from the check-for-modifications dialog, but only by displaying all unmodified files.

图 5.14. 带有修改列表的提交对话框



In the commit dialog you can see those same files, grouped by changelist. Apart from giving an immediate visual indication of groupings, you can also use the group headings to select which files to commit.

On XP, there is a context menu when you right click on a group heading which gives you the choice to check or uncheck all group entries. On Vista however the context menu is not necessary. Click on the group header to select all entries, then check one of the selected entries to check all.

TortoiseSVN reserves one changelist name for its own use, namely `ignore-on-commit`. This is used to mark versioned files which you almost never want to commit even though they have local changes. This feature is described in "Excluding Items from the Commit List"一节.

When you commit files belonging to a changelist then normally you would expect that the changelist membership is no longer needed. So by default, files are removed from changelists automatically on

commit. If you wish to retain the file in its changelist, use the Keep changelists checkbox at the bottom of the commit dialog.

> **提示**
>
> 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.

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。

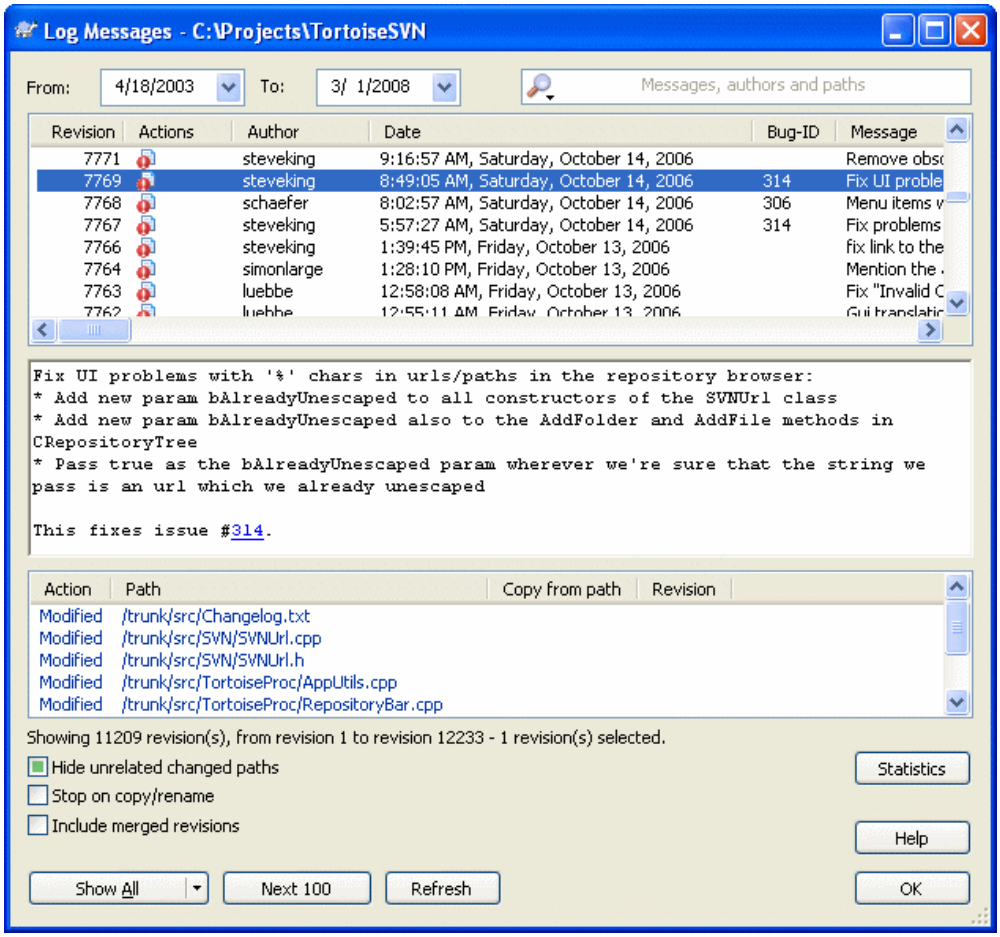
  以蓝色显示的行表示某些内容被复制到该开发版本中(可能是从一个分支中复制而来)。

- 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。

调用版本日志对话框

图 5.15. 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。

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。


If a revision modified a file or directory, the modified icon is shown in the first column.

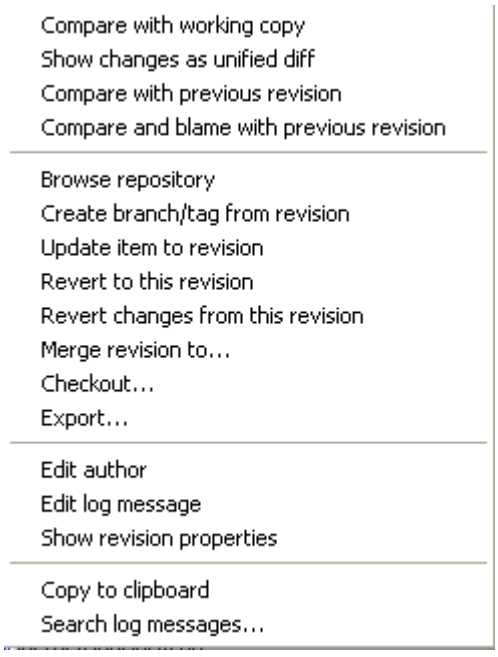

If a revision added a file or directory, the added icon is shown in the second column.


If a revision deleted a file or directory, the deleted icon is shown in the third column.


If a revision replaced a file or directory, the replaced icon is shown in the fourth column.

获得更多信息

图 5.16. 版本日志对话框的顶部面板的右键菜单

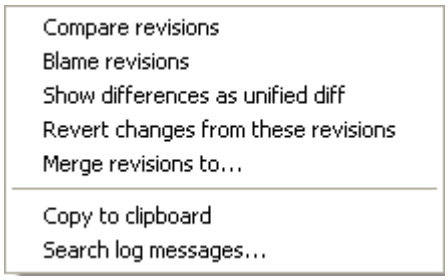


The top pane of the Log dialog has a context menu that allows you to access much more information. Some of these menu entries appear only when the log is shown for a file, and some only when the log is shown for a folder.

- 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。
- 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。

- 比较选择的版本和以前版本。它与你比较工作副本类似。
- Blame the selected revision, and the file in your working BASE and compare the blame reports using a visual diff tool. Read "追溯不同点"一节 for more detail. (files only).
- Blame the selected revision, and the previous revision, and compare the results using a visual diff tool. (folders only).
- Save the selected revision to a file so you have an older version of that file. (files only).
- Open the selected file, either with the default viewer for that file type, or with a program you choose. (files only).
- Blame the file up to the selected revision. (files only).
- Open the repository browser to examine the selected file or folder in the repository as it was at the selected revision.
- 从选中的版本建立一个分支/标记。这个选项很有用。比如: 如果你提交了某些你不想使其进入发行版的修改，却忘记了为此建立标记。
- 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，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，你的工作副本就可能不一致，从而导致你无法提交任何修改。
- 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。如果你做了本地修改，此命令将会合并修改到工作副本。
- 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。再次说明，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。
- Merge 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 dry run. 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.
- 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。
- 编辑之前提交时的日志信息或是作者。请阅读"修改日志消息和作者"一节，了解其工作原理。
- 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。
- 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 5.17. 选种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:

- 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: “比较文件夹”一节。

- Blame the two revisions and compare the blame reports using a visual difference tool. Read “追溯不同点”一节 for more detail.

- 使用单一差异文件显示差异。这对文件和文件夹都有效。

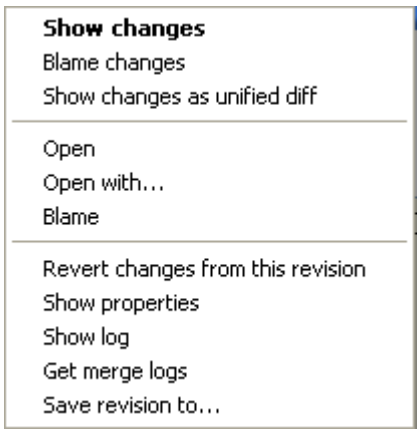
- 如前所述将日志消息复制到剪贴板。

- 如前所述可以搜索日志消息。

If you select two or more revisions (using the usual Ctrl or Shift modifiers), the context menu will include an entry to Revert all changes which were made in the selected revisions. This is the easiest way to rollback a group of revisions in one go.

You can also choose to merge the selected revisions to another working copy, as described above.

图 5.18. 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:

- Show changes made in the selected revision for the selected file. This context menu is only available for files shown as modified.

- Blame the selected revision and the previous revision for the selected file, and compare the blame reports using a visual diff tool. Read “追溯不同点”一节 for more detail.

- 用默认查看器或你指定的程序打开选中文件的选中版本。

- 还原选中文件的选中版本所作的变更。

- 查看选中项的Subversion属性。

- 显示选中的单个文件的版本日志。

- 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。

提示

You may notice that sometimes we refer to changes and other times to differences. What's the difference?

Subversion uses revision numbers to mean 2 different things. A revision generally represents the state of the repository at a point in time, but it can also be used to represent the changeset which created that revision, eg. "Done in r1234" means that the changes committed in r1234 implement feature X. To make it clearer which sense is being used, we use two different terms.

If you select two revisions N and M, the context menu will offer to show the difference between those two revisions. In Subversion terms this is `diff -r M:N`.

If you select a single revision N, the context menu will offer to show the changes made in that revision. In Subversion terms this is `diff -r N-1:N` or `diff -c N`.

The bottom pane shows the files changed in all selected revisions, so the context menu always offers to show changes.

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 ("TortoiseSVN的设置"一节)，
- 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。

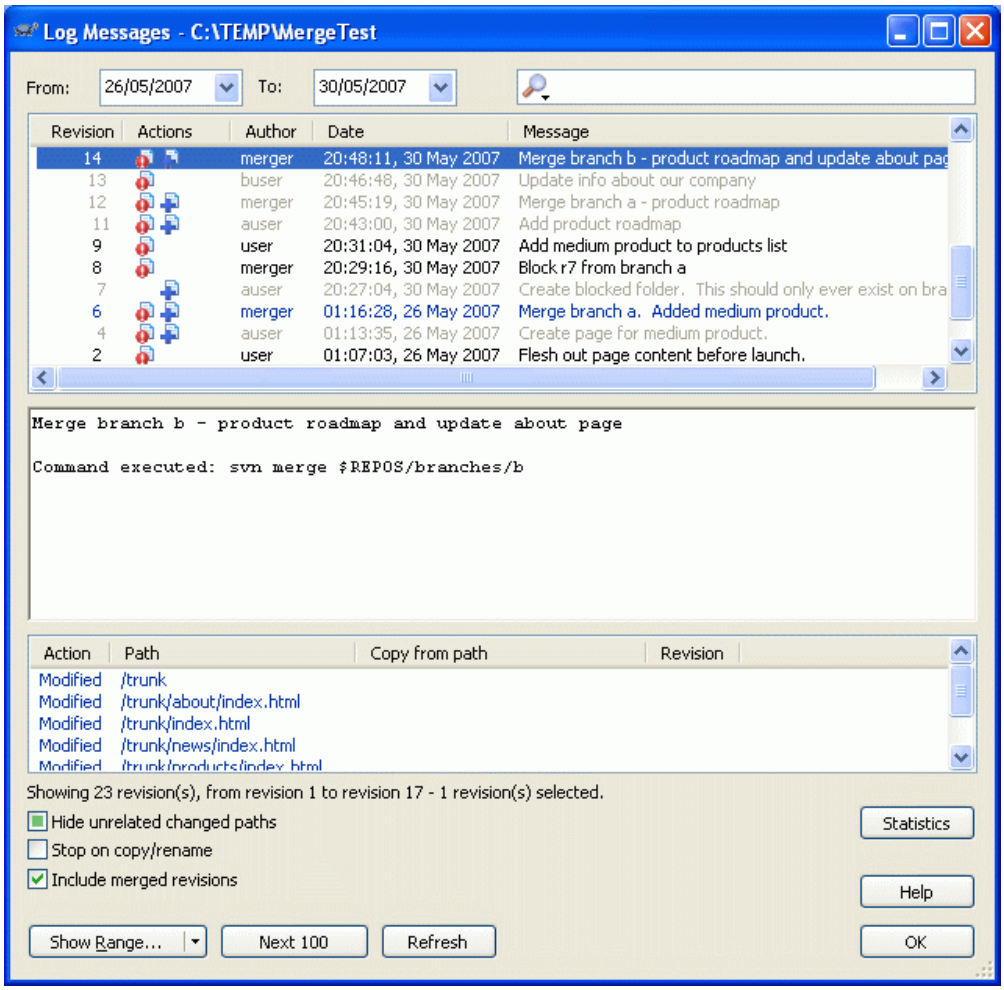
如果你要查询指定范围的版本，使用显示范围 ...。这会出现一个对话框，要求输入开始和结束的版本。

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。

Merge Tracking Features

Subversion 1.5 and later keeps a record of merges using properties. This allows us to get a more detailed history of merged changes. For example, if you develop a new feature on a branch and then merge that branch back to trunk, the feature development will show up on the trunk log as a single commit for the merge, even though there may have been 1000 commits during branch development.

图 5.19. The Log Dialog Showing Merge Tracking Revisions



If you want to see the detail of which revisions were merged as part of that commit, use the Include merged revisions checkbox. This will fetch the log messages again, but will also interleave the log messages from revisions which were merged. Merged revisions are shown in grey because they represent changes made on a different part of the tree.

Of course, merging is never simple! During feature development on the branch there will probably be occasional merges back from trunk to keep the branch in sync with the main line code. So the merge history of the branch will also include another layer of merge history. These different layers are shown in the log dialog using indentation levels.

修改日志消息和作者

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

Subversion lets you change both the log message and the author of revisions any time you want. But since such changes can't be undone (those changes are not versioned) this feature is disabled by default. To make this work, you must set up a pre-revprop-change hook. Please refer to the chapter on Hook Scripts [http://svnbook.red-bean.com/en/1.4/svn.reposadmin.create.html#svn.reposadmin.create.hooks] in the Subversion Book for details about how to do that. Read "钩子脚本"一节 to find some further notes on implementing hooks on a Windows machine.

一旦你按需要为服务器设置了钩子，你就可以使用日志对话框顶部面板的右键菜单来修改任意版本的作者和日志信息了。

警告

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

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。

Click on the search icon to select which information you want to search in, and to choose regex mode. Normally you will only need a simple text search, but if you need to more flexible search terms, you can use regular expressions. If you hover the mouse over the box, a tooltip will give hints on how to use the regex functions. You can also find online documentation and a tutorial at http://www.regular-expressions.info/.

要注意的是，这些过滤器只对已经获取的信息有效。它们并不从版本库中下载信息。

You can also filter the path names in the bottom pane using the Hide unrelated changed paths checkbox. Related paths are those which contain the path used to display the log. If you fetch the log for a folder, that means anything in that folder or below it. For a file it means just that one file. The checkbox is tristate: you can show all paths, grey out the unrelated ones, or hide the unrelated paths completely.

Sometimes your working practices will require log messages to follow a particular format, which means that the text describing the changes is not visible in the abbreviated summary shown in the top pane. The property `tsvn:logsummary` can be used to extract a portion of the log message to be shown in the top pane. Read “TortoiseSVN Project Properties”一节 to find out how to use this property.

统计信息

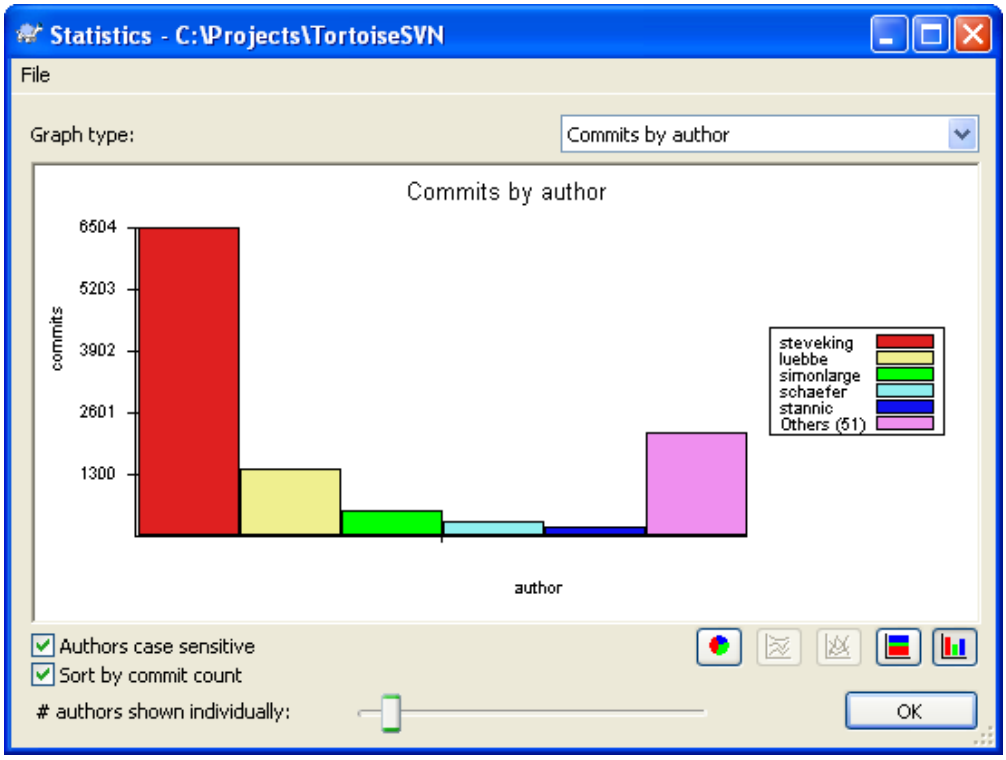
统计按扭，可以显示一些你感兴趣的关于日志对话框中版本的信息。可以显示已经有几个作者做了工作，他们各提交了几次，按周的统计，等等。现在，你可以发现一个大概情况: 谁最勤快，谁偷懒。;-)

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。

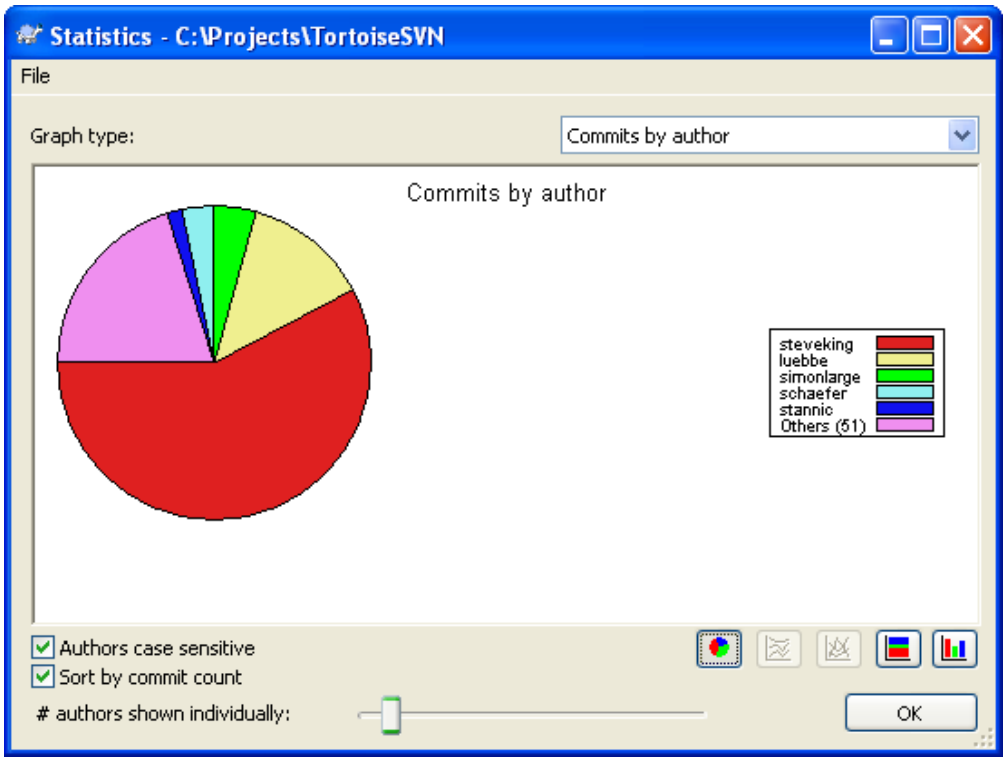
作者提交次数统计页

图 5.20. 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。

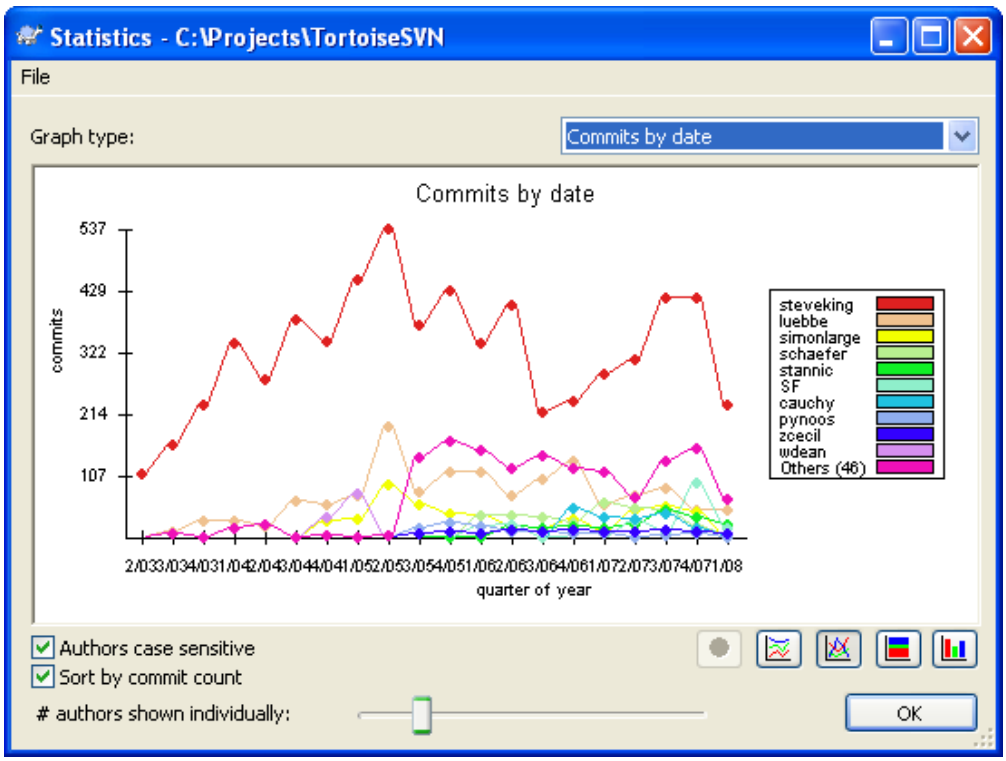
图 5.21. 作者提交次数统计饼图



Where there are a few major authors and many minor contributors, the number of tiny segments can make the graph more difficult to read. The slider at the bottom allows you to set a threshold (as a percentage of total commits) below which any activity is grouped into an Others category.

按日期提交统计页

图 5.22. 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。

By default the analysis is case-sensitive, so users `PeterEgan` and `PeteRegan` are treated as different authors. However, in many cases user names are not case-sensitive, and are sometimes entered inconsistently, so you may want `DavidMorgan` and `davidmorgan` to be treated as the same person. Use the Authors case insensitive checkbox to control how this is handled.

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。

Offline Mode

If the server is not reachable, and you have log caching enabled you can use the log dialog and revision graph in offline mode. This uses data from the cache, which allows you to continue working although the information may not be up-to-date or even complete.

Refreshing the View

If you want to check the server again for newer log messages, 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 messages 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 can use Shift-F5 or Ctrl-F5 to re-fetch the displayed messages from the server and update the log cache. Note that this only affects messages currently shown and does not invalidate the entire cache for that repository.

查看差异

One of the commonest requirements in project development is to see what has changed. You might want to look at the differences between two revisions of the same file, or the differences between two separate files. TortoiseSVN provides a built-in tool named TortoiseMerge for viewing differences of text files. For viewing differences of image files, TortoiseSVN also has a tool named TortoiseIDiff. Of course, you can use your own favourite diff program if you like.

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。邮件菜单允许你使用比较版本来比较它们。阅读"比较文件夹"一节以便获得更多信息。

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

  If you want to see the difference between the last committed revision and your working copy, assuming that the working copy hasn't been modified, just right click on the file. Then select TortoiseSVN → Diff with previous version. This will perform a diff between the revision before the last-commit-date (as recorded in your working copy) and the working BASE. This shows you the last change made to that file to bring it to the state you now see in your working copy. It will not show changes newer than your working copy.

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"比较文件夹"一节以便获得更多信息。

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
: If you want to see the differences between a file in your working copy, and a file in any Subversion repository, you can do that directly in explorer by selecting the file then holding down the Shift key whilst right clicking to obtain the context menu. Select TortoiseSVN → Diff with URL. You can do the same thing for a working copy folder. TortoiseMerge shows these differences in the same way as it shows a patch file - a list of changed files which you can view one at a time.

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谴责信息。这里有更多详细介绍"追溯不同点"一节。

比较文件夹差异

The built-in tools supplied with TortoiseSVN do not support viewing differences between directory hierarchies. But if you have an external tool which does support that feature, you can use that instead. In "其他的比较/合并工具"一节 we tell you about some tools which we have used.

If you have configured a third party diff tool, you can use Shift when selecting the Diff command to use the alternate tool. Read "外部程序设置"一节 to find out about configuring other diff tools.

Line-end and Whitespace Options

Sometimes in the life of a project you might change the line endings from CRLF to LF, or you may change the indentation of a section. Unfortunately this will mark a large number of lines as changed, even though there is no change to the meaning of the code. The options here will help to manage these changes when it comes to comparing and applying differences. You will see these settings in the Merge and Blame dialogs, as well as in the settings for TortoiseMerge.

Ignore line endings excludes changes which are due solely to difference in line-end style.

Compare whitespaces includes all changes in indentation and inline whitespace as added/removed lines.

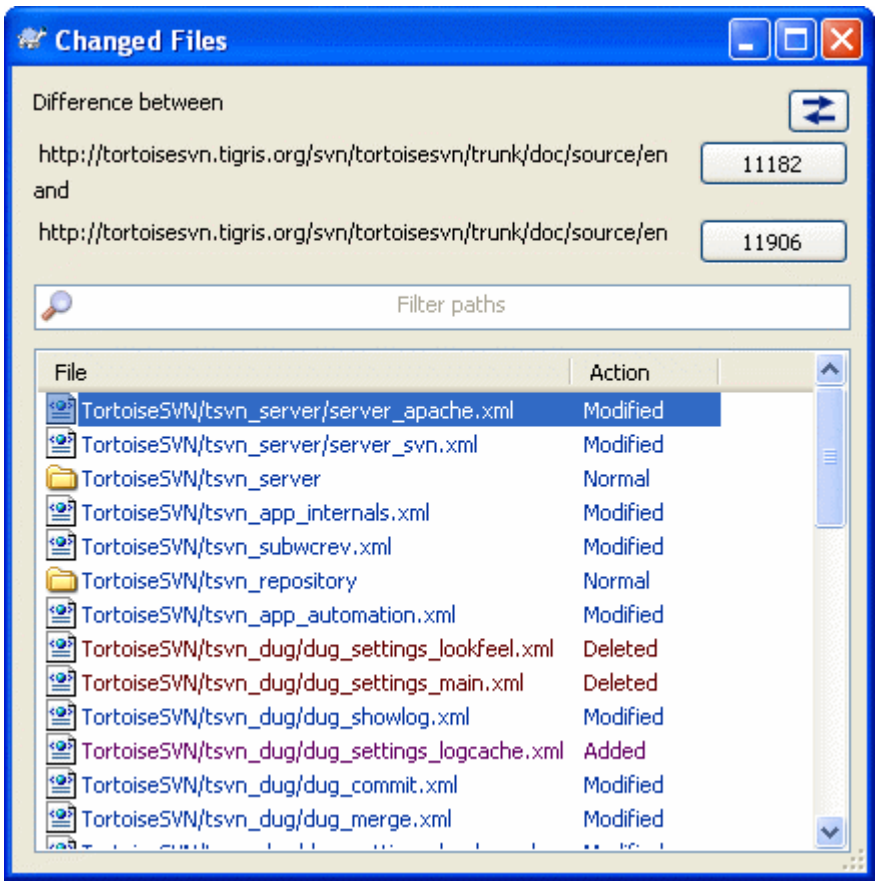
Ignore whitespace changes excludes changes which are due solely to a change in the amount or type of whitespace, eg. changing the indentation or changing tabs to spaces. Adding whitespace where there was none before, or removing a whitespace completely is still shown as a change.

Ignore all whitespaces excludes all whitespace-only changes.

Naturally, any line with changed content is always included in the diff.

比较文件夹

图 5.23. 修订版本版本比较对话框



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，就可以使用上下文菜单 → 比较版本。

这个对话框显示一个所有已经修改的文件列表，允许你使用邮件菜单单独的比较或回溯它们。

你也可以将已经修改的文件列表导出到一个文本文件中，或者将修改的文件导出到一个目录。这个操作只在选择的文件上工作，所以你需要选择感兴趣的文件 - 通常是所有文件。

If you want to export the list of files and the actions (modified, added, deleted) as well, you can do that using the keyboard shortcuts Ctrl-A to select all entries and Ctrl-C to copy the detailed list to the clipboard.

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。

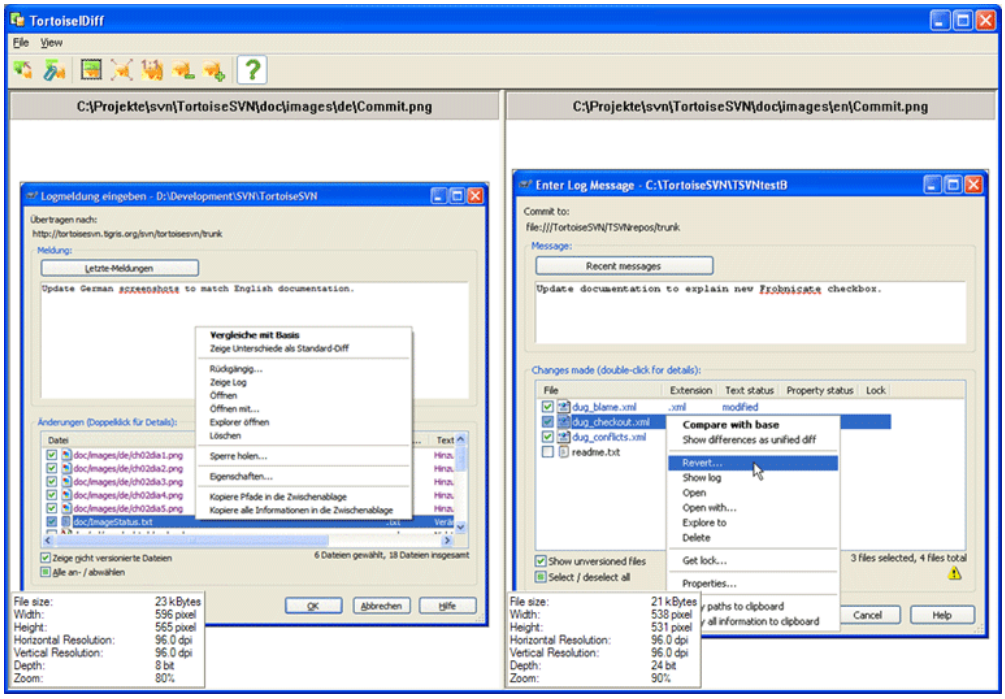
有版本数字的按钮可以用来改变版本范围。当你改变范围时，两个版本不同的项目列表会自动更新。

If the list of filenames is very long, you can use the search box to reduce the list to filenames containing specific text. Note that a simple text search is used, so if you want to restrict the list to C source files you should enter `.c` rather than `*.c`.

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 5.24. 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。

Naturally you can also zoom in and out and pan around the image. You can also pan the image simply by left-dragging it. If you select the Link images together option, then the pan controls (scrollbars, mousewheel) on both images are linked.

An image info box shows details about the image file, such as the size in pixels, resolution and colour depth. If this box gets in the way, use View → Image Info to hide it. You can get the same information in a tooltip if you hover the mouse over the image title bar.

When the images are overlaid, the relative intensity of the images (alpha blend) is controlled by a slider control at the left side. You can click anywhere in the slider to set the blend directly, or you can drag the slider to change the blend interactively. Ctrl+Shift-Wheel to change the blend.

滑杆上部的按钮在两个混合展现之间切换，它标记了混合滑杆控制哪边。默认一个在顶部，另一个在底部，所以切换按钮只是在一个图像与另一个之间转换。你可以移动标记选择切换按钮使用哪个混合值。

Sometimes you want to see a difference rather than a blend. You might have the image files for two revisions of a printed circuit board and want to see which tracks have changed. If you disable alpha blend mode, the difference will be shown as an XOR of the pixel colour values. Unchanged areas will be plain white and changes will be coloured.

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 is a useful commercial tool for diff and merging both files and folders. It does three-way comparison in merges and has synchronization links to use if you've changed the order of functions. Download it from Araxis [http://www.araxis.com/merge/index.html].

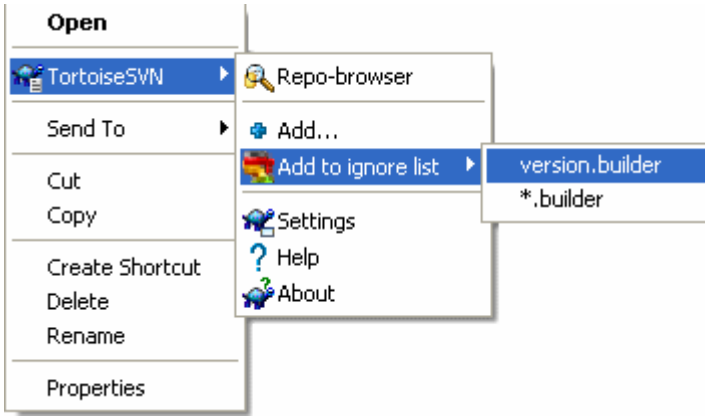
SciTE
: 这款文本编译器在统一比较时提供语法显示，读起来更加容易。可以从这里下载Scintilla [http://www.scintilla.org/SciTEDownload.html]。

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]。

"外部程序设置"一节这里可以了解到怎样起用TortoiseSVN来使用这些工具。

添加新文件和目录

图 5.25. 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

You can also use the Add command on already versioned folders. In that case, the add dialog will show you all unversioned files inside that versioned folder. This helps if you have many new files and need to add them all at once.

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. 拖拽(right-drag)他们到新的工作副本下，
3. 松开鼠标右键
4. 选择上下文菜单 → 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。

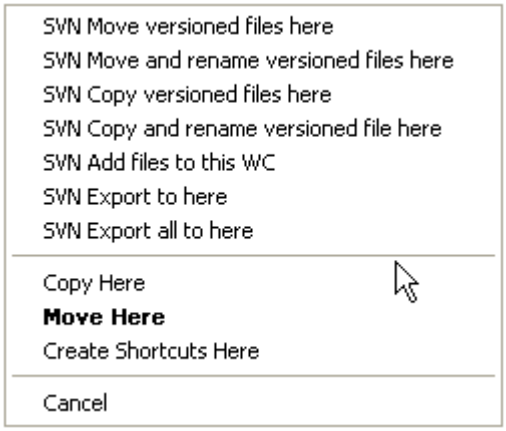
If you add a file or folder by mistake, you can undo the addition before you commit using TortoiseSVN → Undo add....

Copying/Moving/Renaming Files and Folders

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 5.26. 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。

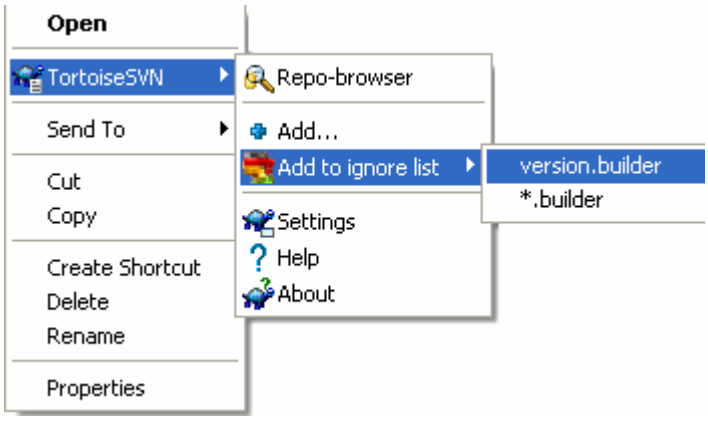
You can copy files and folders from your working copy to another location in the repository using TortoiseSVN → Branch/Tag. Refer to "创建一个分支或标记"一节 to find out more.

You can locate an older version of a file or folder in the log dialog and copy it to a new location in the repository directly from the log dialog using Context menu → Create branch/tag from revision. Refer to "获得更多信息"一节 to find out more.

You can also use the repository browser to locate content you want, and copy it into your working copy directly from the repository, or copy between two locations within the repository. Refer to "版本库浏览器"一节 to find out more.

忽略文件和目录

图 5.27. 未受版本控制的文件之资源管理器上下文菜单



In most projects you will have files and folders that should not be subject to version control. These might include files created by the compiler, `*.obj`, `*.lst`, maybe an output folder used to store the executable. Whenever you commit changes, TortoiseSVN shows your unversioned files, which fills up the file list in the commit dialog. Of course you can turn off this display, but then you might forget to add a new source file.

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"项目设置"一节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"常规设置"一节获得更多信息。

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"忽略已经版本控制的文件"一节介绍怎样"取消版本控制(unversion)"。

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]`匹配任何但个的`A`,`G`,`m`,`n`,`o`或者`p`。

Subversion 在所有内部路径名称中使用 / 作为路径分割符，所有模式匹配都使用这种路径名称风格。如果你想在忽略模式中使用路径分割符，确认使用 /，而不是 Windows 的反斜线符号。

模式匹配是大小写敏感的，这在windows平台下会出问题。你可以要比较的字符硬性的强制忽略大小写。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。

Subversion 对每个路径使用这种匹配模式。这些路径通常相对于执行导入，增加，提交等动作的目录。因此，此匹配模式考虑在文件名称之前可能有，也可能没有路径组件这个事实。

如果目录名称在路径中，匹配算法不会删除它们，所以模式`Fred.*`匹配`Fred.c`，但是不匹配`subdir/Fred.c`。这对于你添加一个包含你想忽略的一些文件的目录来说是非常有意义的，因为在 Subversion 处理忽略模式时，这些文件名的优先级高于目录名。

在模式匹配中，`/` 字符不会做任何特别处理，所以模式 `abc*xyz` 匹配 `abcdxyz`，但是不匹配 `abcdir/subdir/anything/morexyz`。

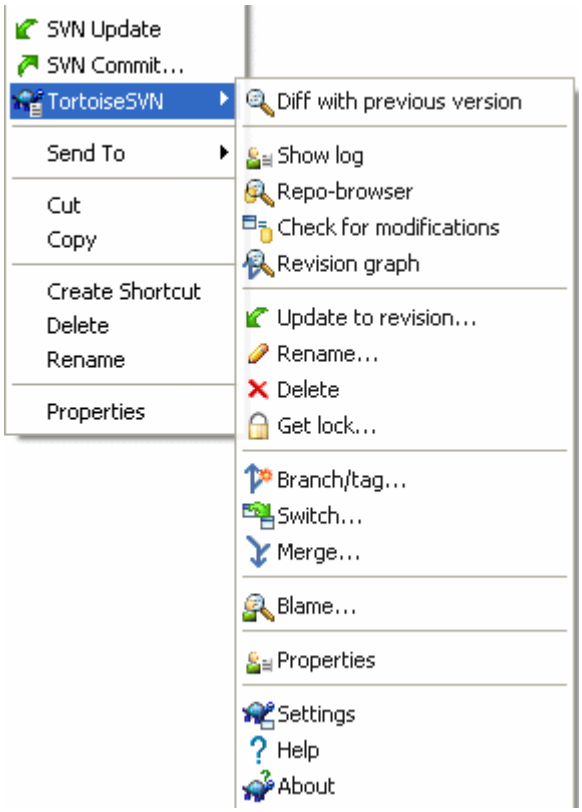
To ignore all `cvs` folders you should either specify a pattern of `*cvs` or better, the pair `cvs */cvs`. The first option works, but would also exclude something called `ThisIsNotCVS`. Using `*/cvs` alone will not work on an immediate child `cvs` folder, and `cvs` alone will not work on sub-folders.

如果你想要定义一个特殊的忽略规则。你可以在关于shell命令行语言的IEEE规范中找到Pattern Matching Notation [http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_13]。

Deleting, Moving and Renaming

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重命名的菜单项。

图 5.28. 版本控制文件的菜单浏览



Deleting files and folders

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

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

When you TortoiseSVN → Delete a folder, it remains in your working copy, but the overlay changes to indicate that it is marked for deletion. Up until you commit the change, you can get the folder back using TortoiseSVN → Revert on the folder itself. This difference in behaviour between files and folders is a part of Subversion, not TortoiseSVN.

If you want to delete an item from the repository, but keep it locally as an unversioned file/folder, use Extended Context Menu → Delete (keep local). You have to hold the Shift key while right clicking on the item in the explorer list pane (right pane) in order to see this in the extended context menu.

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。

Moving files and folders

If you want to do a simple in-place rename of a file or folder, use Context Menu → Rename... Enter the new name for the item and you're done.

If you want to move files around inside your working copy, perhaps to a different sub-folder, use the right-mouse drag-and-drop handler:

1. 选择你要移动的文件或目录
2. 拖拽(right-drag)他们到新的工作副本下，
3. 松开鼠标右键
4. 在弹出菜单选择上下文菜单 → SVN 移动文件。

提交父目录

Since renames and moves are done as a delete followed by an add you must commit the parent folder of the renamed/moved file so that the deleted part of the rename/move will show up in the commit dialog. If you don't commit the removed part of the rename/move, it will stay behind in the repository and when your co-workers update, the old file will not be removed. i.e. they will have both the old and the new copies.

你 必须在重命名目录后而在更改目录下的任何文件前进行提交，不然你的工作副本就回真的混淆。

You can also use the repository browser to move items around. Read "版本库浏览器"一节 to find out more.

不要使用 SVN 移动外部连接

你不应该用TortoiseSVN的移动或重命名命令作用在一个已经用`svn:externals`创建的目录上。因为这个动作可能会导致外部的元素(item)被从他的父版本库中删除，这可能会使其他很多人烦恼。如果你必须移动洋外部的目录，你应该使用一个普通的shell移动，然后调整源文件和目的文件的父目录的`svn:externals`道具。

Changing case in a filename

Making case-only changes to a filename is tricky with Subversion on Windows, because for a short time during a rename, both filenames have to exist. As Windows has a case-insensitive file system, this does not work using the usual Rename command.

Fortunately there are (at least) two possible methods to rename a file without losing its log history. It is important to rename it within subversion. Just renaming in the explorer will corrupt your working copy!

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

Dealing with filename case conflicts

If the repository already contains two files with the same name but differing only in case (e.g. `TEST.TXT` and `test.txt`), you will not be able to update or checkout the parent directory on a Windows client. Whilst Subversion supports case-sensitive filenames, Windows does not.

This sometimes happens when two people commit, from separate working copies, files which happen to have the same name, but with a case difference. It can also happen when files are committed from a system with a case-sensitive file system, like 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。

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。

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。

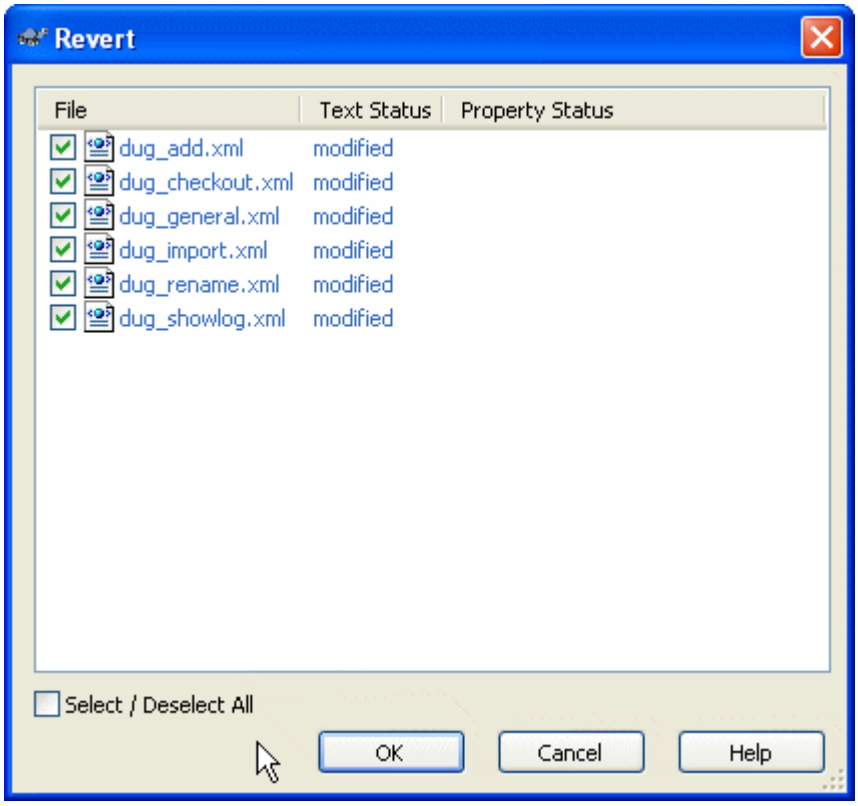
TortoiseSVN 提供了使用扩展上下文菜单 → 删除未版本控制的项目...来清理工作副本。你可以在目录上右键操作时，保持 Shift按下，就可以看到这个上下文菜单。它会出现一个对话框，列出工作副本中的所有未版本控制的文件。你可以选择或取消删除的项目。

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。

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，　　右击弹出快捷菜单，然后选择TortoiseSVN → Revert命令，将会弹出一个显示这个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OK.

图 5.29. 恢复对话框



。

If you want to undo a deletion or a rename, you need to use Revert on the parent folder as the deleted item does not exist for you to right-click on.

If you want to undo the addition of an item, this appears in the context menu as TortoiseSVN → Undo add.... This is really a revert as well, but the name has been changed to make it more obvious.

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"本地与远程状态"一节

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 "版本日志对话框"一节 获得更多信息。

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。

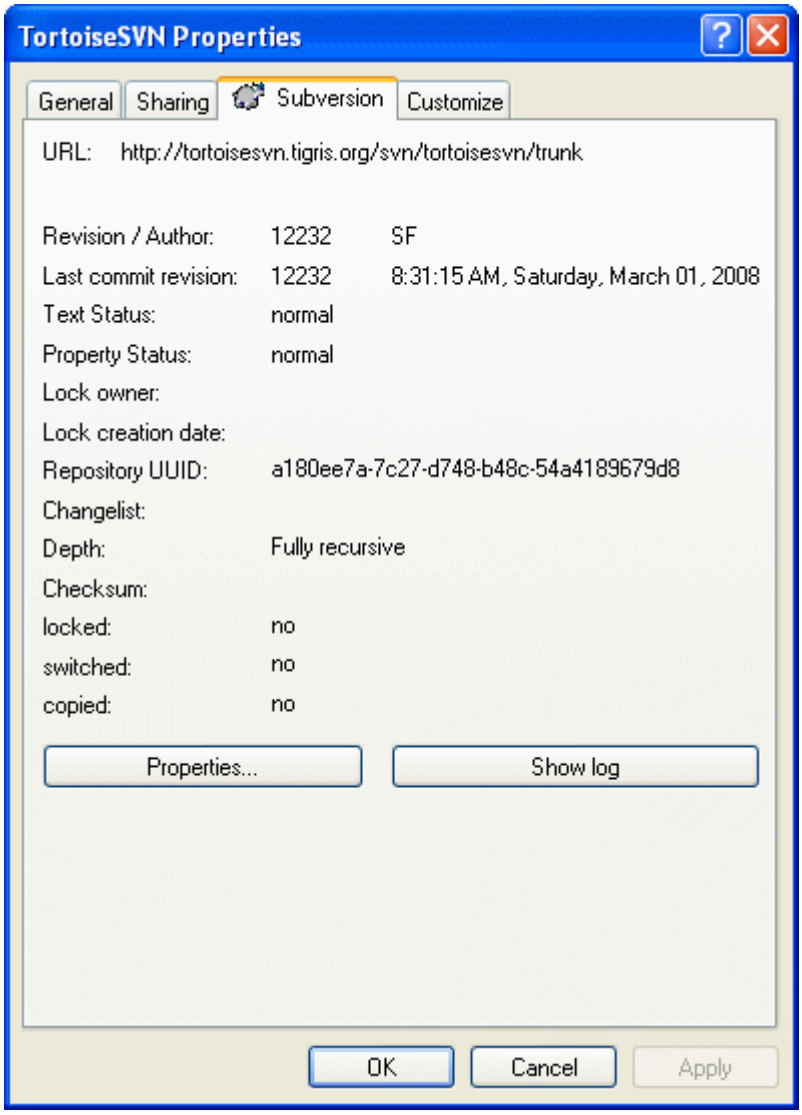
Cleanup has another useful side effect. If a file date changes but its content doesn't, Subversion cannot tell whether it has really changed except by doing a byte-by-byte comparison with the pristine copy. If you have a lot of files in this state it makes acquiring status very slow, which will make many dialogs slow to respond. Executing a Cleanup on your working copy will repair these "broken" timestamps and restore status checks to full speed.

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。

项目设置

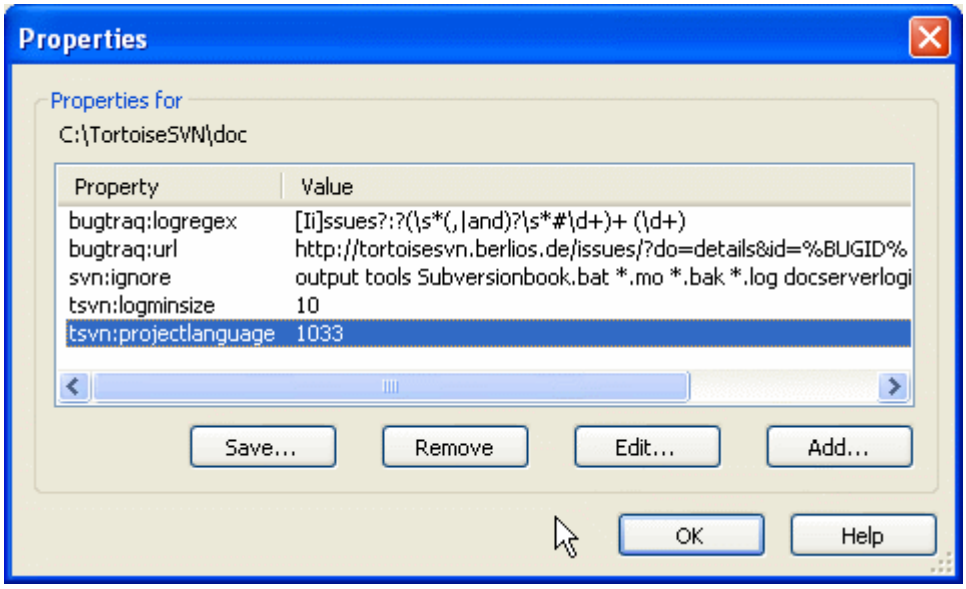
图 5.30. 资源管理器属性页，Subversion 页面



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覆盖的标志。 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 Menu → properties(注意: 这是浏览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。

Subversion 属性

图 5.31. Subversion 属性页



You can read and set the Subversion properties from the Windows properties dialog, but also from TortoiseSVN → properties and within TortoiseSVN's status lists, from Context menu → properties.

You can add your own properties, or some properties with a special meaning in Subversion. These begin with `svn:`. `svn:externals` is such a property; see how to handle externals in "引用的工程"一节.

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。

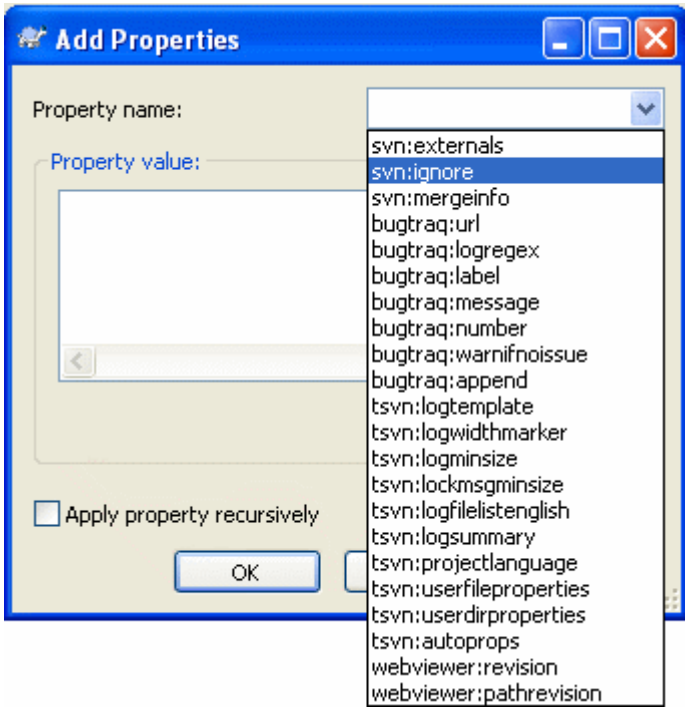
Id
前述四个关键字的压缩组合。

To find out how to use these keywords, look at the svn:keywords section [http://svnbook.red-bean.com/en/1.4/svn.advanced.props.special.keywords.html] in the Subversion book, which gives a full description of these keywords and how to enable and use them.

For more information about properties in Subversion see the Special Properties [http://svnbook.red-bean.com/en/1.4/svn.advanced.props.html].

Adding and Editing Properties

图 5.32. 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"引用的工程"一节以获得更多信息。

Exporting and Importing Properties

Often you will find yourself applying the same set of properties many times, for example `bugtraq:logregex`. To simplify the process of copying properties from one project to another, you can use the Export/Import feature.

From the file or folder where the properties are already set, use TortoiseSVN → properties, select the properties you wish to export and click on Export.... You will be prompted for a filename where the property names and values will be saved.

From the folder(s) where you wish to apply these properties, use TortoiseSVN → properties and click on Import.... You will be prompted for a filename to import from, so navigate to the place you saved the export file previously and select it. The properties will be added to the folders non-recursively.

If you want to add properties to a tree recursively, follow the steps above, then in the property dialog select each property in turn, click on Edit..., check the Apply property recursively box and click on OK.

The Import file format is binary and proprietary to TortoiseSVN. Its only purpose is to transfer properties using Import and Export, so there is no need to edit these files.

Binary Properties

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。

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。

自动属性设置

你可以设置当文件和文件夹加入版本库时，自动设置属性。阅读"TortoiseSVN的设置"一节以获得更多信息。

TortoiseSVN Project Properties

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 allows you to set "autoprops" which will be applied to newly added or imported files, based on the file extension. This depends on every client having set appropriate autoprops in their subversion configuration file. `tsvn:autoprops` can be set on folders and these will be merged with the user's local autoprops when importing or adding files.

  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)。

- In the Commit dialog you have the option to paste in the list of changed files, including the status of each file (added, modified, etc). `tsvn:logfilelistenglish` defines whether the file status is inserted in English or in the localized language. If the property is not set, the default is `true`.

- TortoiseSVN can use spell checker modules which are also used by OpenOffice and Mozilla. If you have those installed this property will determine which spell checker to use, i.e. in which language the log messages for your project should be written. `tsvn:projectlanguage` sets the language module the spell checking engine should use when you enter a log message. You can find the values for

your language on this page: MSDN: Language Identifiers [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`。

- The property `tsvn:logsummary` is used to extract a portion of the log message which is then shown in the log dialog as the log message summary.

  The value of the `tsvn:logsummary` property must be set to a one line regex string which contains one regex group. Whatever matches that group is used as the summary.

  An example: `\[SUMMARY\]:\s+(.*)` Will catch everything after "[SUMMARY]" in the log message and use that as the summary.

- 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。

Some `tsvn:` properties require a `true/false` value. TortoiseSVN also understands `yes` as a synonym for `true` and `no` as a synonym for `false`.

TortoiseSVN can integrate with some bug tracking tools. This uses project properties that start with `bugtraq:`. Read "Integration with Bug Tracking Systems / Issue Trackers"一节 for further information.

It can also integrate with some web-based repository browsers, using project properties that start with `webviewer:`. Read "与基于 WEB 的版本库浏览器集成"一节 for further information.

Set the project properties on folders

These special project 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 is sufficient to 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 project 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.

When you add new sub-folders using TortoiseSVN, any project properties present in the parent folder will automatically be added to the new child folder too.

小心

Although TortoiseSVN's project properties are extremely useful, they only work with TortoiseSVN, and some will only work in newer versions of TortoiseSVN. If people working on your project use a variety of Subversion clients, or possibly have old versions of TortoiseSVN, you may want to use repository hooks to enforce project policies. project properties can only help to implement a policy, they cannot enforce it.

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开发品的一个分割线上。这条线被称为分支。分支经常被用来试验新的特性，而不会对开发有编译错误的干扰。当新的特性足够稳定之后，开发品的分支就可以混合回主分支里(主干线).

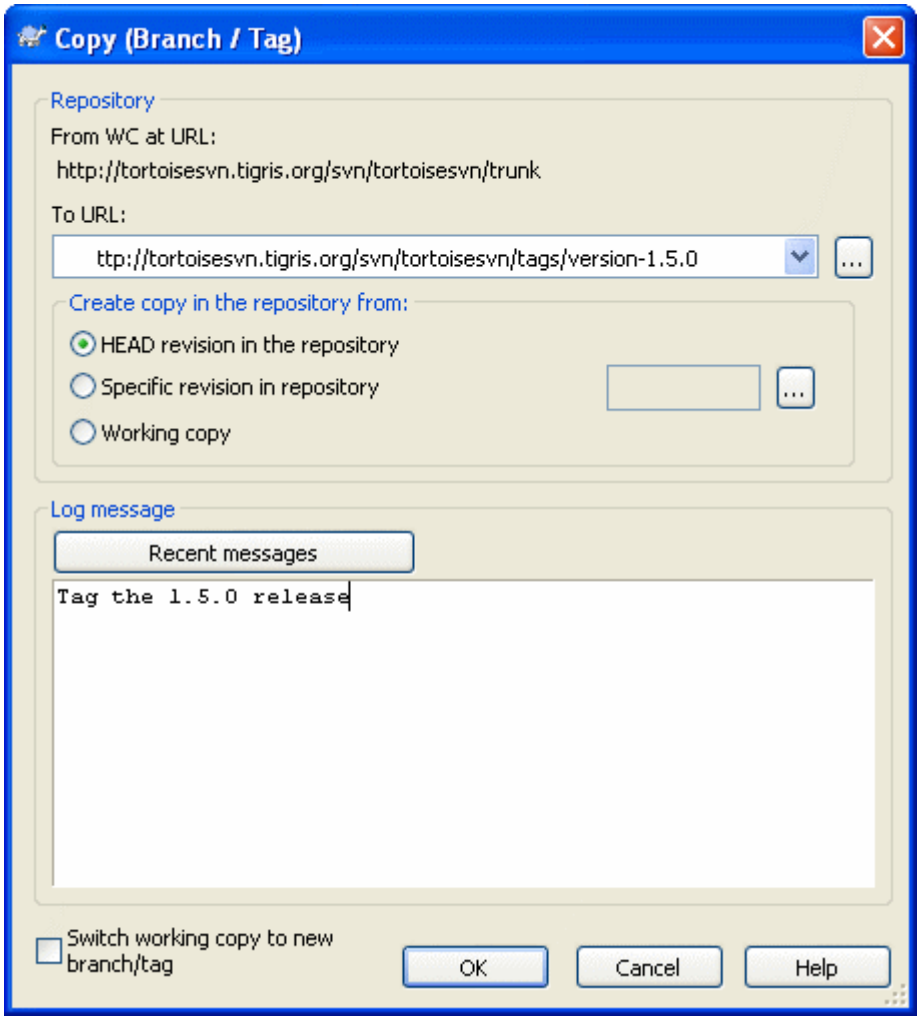
版本控制系统的另一个特性是能够标记特殊的版本(例如某个发布版本)，所以你可以在任何时候重新建立一个特定的构件和环境。这个过程被称作标记。

Subversion does not have special commands for branching or tagging, but uses so-called “cheap copies” instead. Cheap copies are similar to hard links in Unix, which means that instead of making a complete copy in the repository, an internal link is created, pointing to a specific tree/revision. As a result branches and tags are very quick to create, and take up almost no extra space in the repository.

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 5.33. 分支/标记对话框



在你当前的工作副本中给你你想要复制的分支或标记选择一个目录，然后选择命令TortoiseSVN → 分支/标记...。

默认的目标URL将会是你当前工作副本所处的源URL。你必须给你的分支/标记编辑一个新路径。来取代

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

你可以使用这样的设置

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

如果你忘记了你上一次使用的命名约定，可以用鼠标右键打开版本库浏览器来察看已经存在的版本库结构。

现在你必须选择要复制的源位置。在这里你有三个设置选项:

版本库中的最新版本
: 新分支直接从仓库中的最新版本里复制出来。不需要从你的工作副本中传输任何数据，这个分支的建立是非常快的。

在版本库中指定具体的版本
: 在仓库中直接复制建立一个新分支同时你也可以选择一个旧版本。假如在你上周发布了项目时忘记了做标记，这将非常有用。如果你记不起来版本号，通过点击鼠标右键来显示版本日志，同时从这里选取版本号。和上次一样不需要从你的工作副本中传输任何数据，这个分支建立起来是非常快的。

工作副本
: 新的分支是一个完全等同于你的本地工作副本的一个副本。如果你更新了一些文件到你的工作副本的某个旧版本里，或者你在本地做出了修改，这些改变将准确无误的进入副本中。自然而然地这种综合的标记会包含正在从工作副本传输到版本库的数据，如果这些数据还不存在的话。

如果你想把你的工作副本自动切换到最新创建的分支，使用转换工作副本至新分支/标记 选择框.但是如果你打算这么做，首先要确认你的工作副本没有被修改。如果有修改的话，当你转换后这些修改将会混合进你的工作副本分支里。

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。

需要注意建立一个分支或标记不会影响你的工作副本。即使你复制了你的工作副本，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。

检出或者切换

...这是个小问题。当你想从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。这样能减轻你的网络负担，也能减少你的耐心。

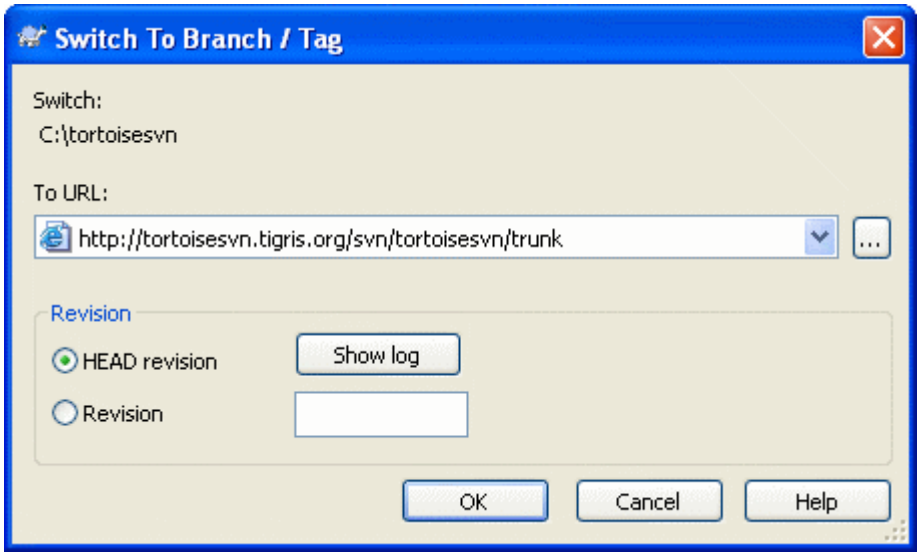
为了能够使用你最新产生的副本你有采用下面几种方法。你可以:

- TortoiseSVN → 检出一个最新的项目在一个空目录下。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副本。
- 在版本库中从你当前的工作副本切换到最新建立的副本。再一次选择你的项目所处的顶级文件夹然后在菜单中使用TortoiseSVN → 切换...。

  在接下来的对话框中蠕蠕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。在工作副本里当你进行切换的时候任何没有提交过的修改都会被混合。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(比如最新版本)。

图 5.34. 切换对话框



尽管Subversion本身不区分标记和分支，它们的使用方法还是有些不同。

- 在某个特殊的阶段标记被用来建立一个项目的静态映像。同样地标记和分支应该被独特地应用于开发品。这就是我们首选推荐 `/trunk /branches /tags`这样的版本库结构的原因。使用标记的版本并不是一个好想法，因为你的本地文件没有写保护，你这样做容易犯错误。不管怎样如果你试着提交(修改)到一个包含`/##/`的版本库路径下，TortoiseSVN 会给你警告。
- 如果你想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。

正在合并

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最新版本，或者将最新版本的修改合并到分支。

It is important to understand how branching and merging works in Subversion before you start using it, as it can become quite complex. It is highly recommended that you read the chapter Branching and Merging [http://svnbook.red-bean.com/en/1.4/svn.branchmerge.html] in the Subversion book, which gives a full description and many examples of how it is used.

The next point to note is that merging always takes place within a working copy. If you want to merge changes into a branch, you have to have a working copy for that branch checked out, and invoke the merge wizard from that working copy using TortoiseSVN → Merge....

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，命令恢复 会丢弃包含你执行合并之前的所有修改。

There are three common use cases for merging which are handled in slightly different ways, as described below. The first page of the merge wizard asks you to select the method you need.

Merge a range of revisions
: 这个方法覆盖了你已经在分支(或者最新版本)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。

  What you are asking Subversion to do is this: “Calculate the changes necessary to get [FROM] revision 1 of branch A [TO] revision 7 of branch A, and apply those changes to my working copy (of trunk or branch B).”

Reintegrate a branch
: This method covers the case when you have made a feature branch as discussed in the Subversion book. All trunk changes have been ported to the feature branch, week by week, and now the feature is complete you want to merge it back into the trunk. Because you have kept the feature branch synchronized with the trunk, the latest versions of branch and trunk will be absolutely identical except for your branch changes.

  This is a special case of the tree merge described below, and it requires only the URL to merge from (normally) your development branch. It uses the merge-tracking features of Subversion to calculate the correct revision ranges to use, and perform additional checks which ensure that the branch has been fully updated with trunk changes. This ensures that you don't accidentally undo work that others have committed to trunk since you last synchronized changes.

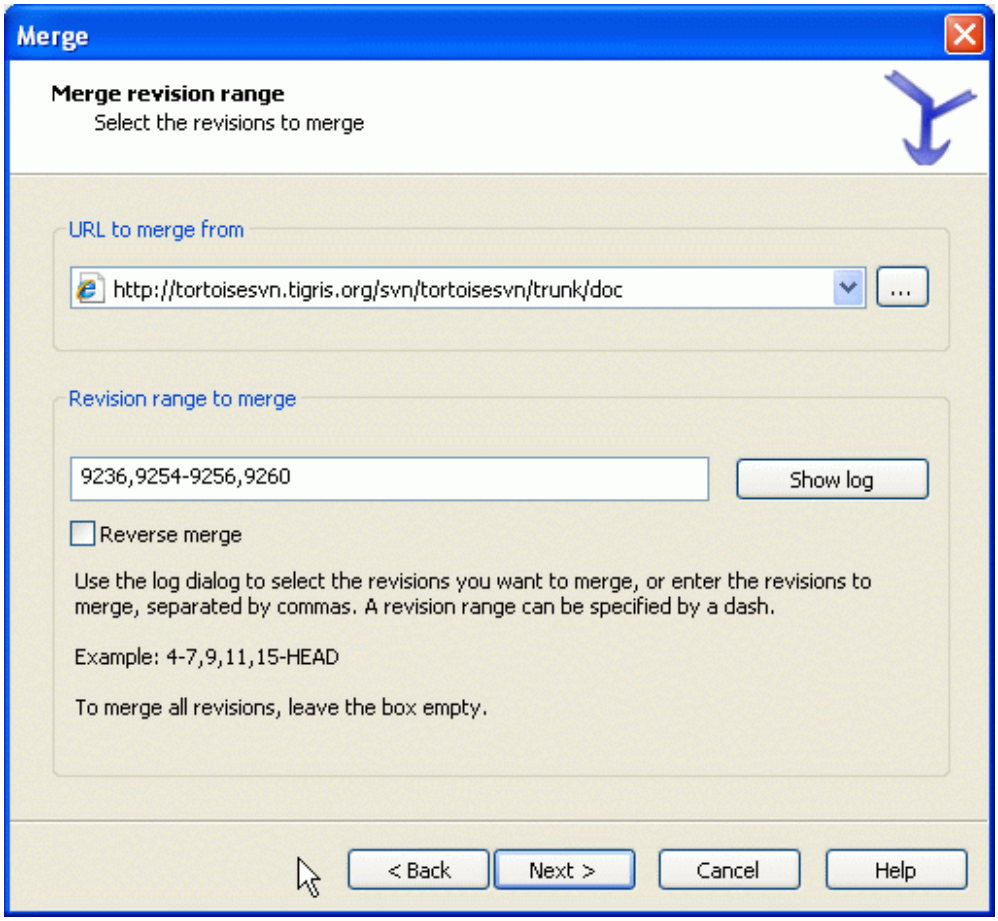
  After the merge, all branch development has been completely merged back into the main development line. The branch is now redundant and can be deleted.

Merge two different trees
: This is a more general case of the reintegrate method. What you are asking Subversion to do is: “Calculate the changes necessary to get [FROM] the head revision of the trunk [TO] the head revision of the branch, and apply those changes to my working copy (of the trunk).” The net result is that trunk now looks exactly like the branch.

If your server/repository does not support merge-tracking then this is the only way to merge a branch back to trunk. Another use case occurs when you are using vendor branches and you need to merge the changes following a new vendor drop into your trunk code. For more information read the chapter on vendor branches [http://svnbook.red-bean.com/en/1.4/svn.advanced.vendorbr.html] in the Subversion Book.

合并指定版本范围

图 5.35. The Merge Wizard - Select Revision Range



In the From: field enter the full folder URL of the branch or tag containing the changes you want to port into your working copy. You may also click ... to browse the repository and find the desired branch. If you have merged from this branch before, then just use the drop down list which shows a history of previously used URLs.

In the Revision range to merge field enter the list of revisions you want to merge. This can be a single revision, a list of specific revisions separated by commas, or a range of revisions separated by a dash, or any combination of these.

The easiest way to select the range of revisions you need is to click on Show Log, as this will list recent changes with their log comments. If you want to merge the changes from a single revision, just select that revision. If you want to merge changes from several revisions, then select that range (using the usual Shift-modifier). Click on OK and the list of revision numbers to merge will be filled in for you.

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，你可以在工作服本上使用显示日志对话框跟踪日志。使用最后合并的版本号作为本次合并的开始版本。例如，你已经合并了版本37到39，那么本次合并你应该从版本39开始。

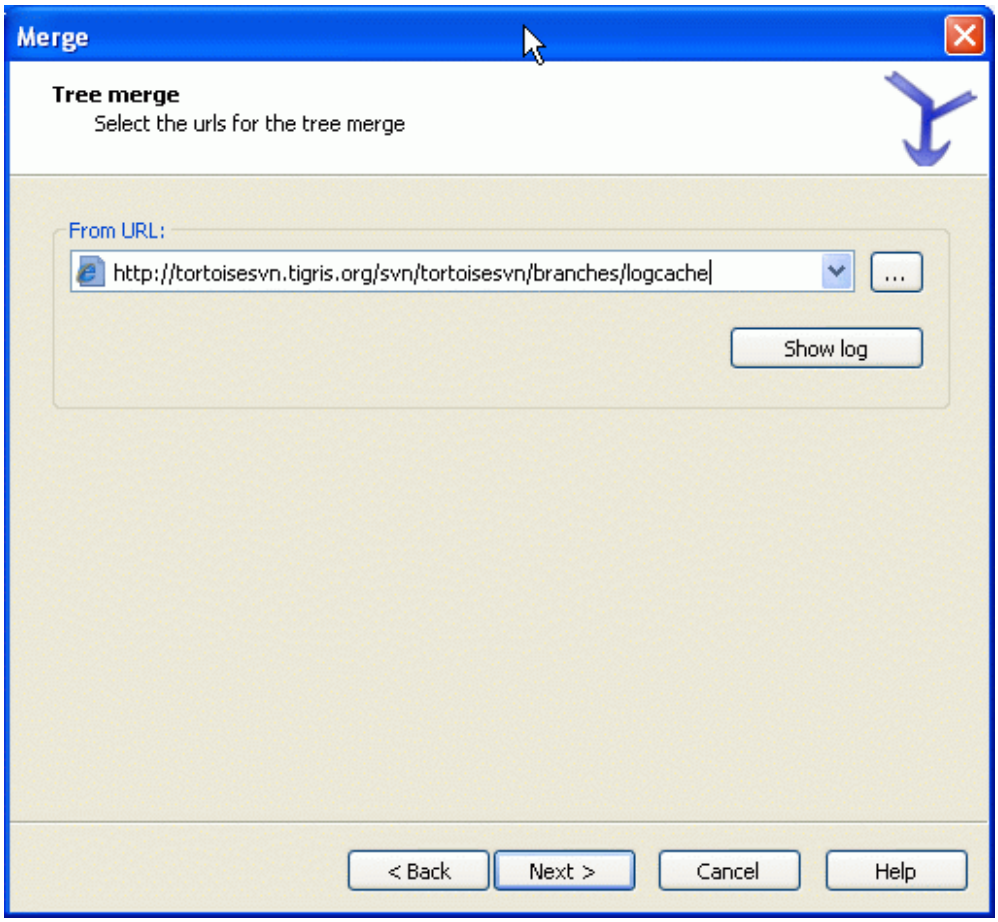
If you are using the merge tracking features of Subversion, you do not need to remember which revisions have already been merged - Subversion will record that for you. If you leave the revision range blank, all revisions which have not yet been merged will be included. Read “Merge Tracking”一节 to find out more.

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。

Click Next and go to “Merge Options”一节

Reintegrate a branch

图 5.36. The Merge Wizard - Reintegrate Merge



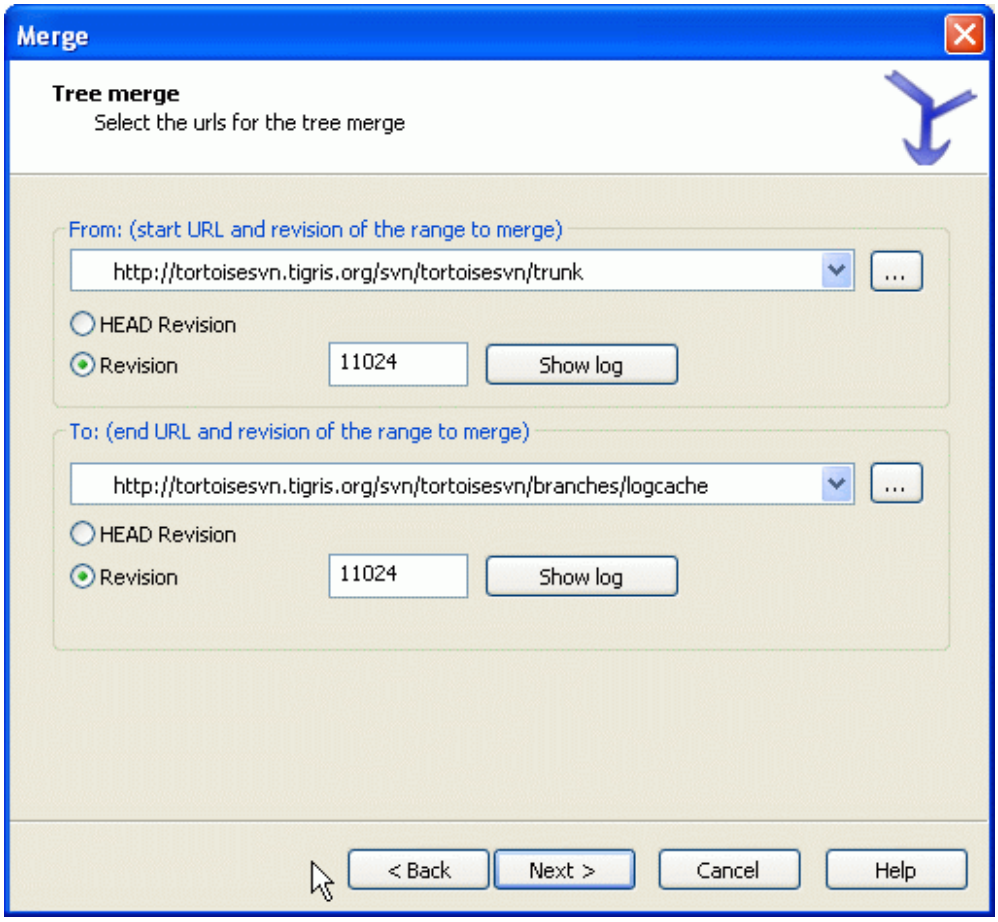
To merge a feature branch back into the trunk you must start the merge wizard from within a working copy of the trunk.

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.

合并两个不同的目录树

图 5.37. The Merge Wizard - Tree Merge



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.

In the To: field enter the full folder URL of the feature branch.

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。

You can also use Show Log to select the revision.

Merge Options

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 “Checkout Depth”一节. 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/1.4/svn.branchmerge.copychanges.html#svn.branchmerge.copychanges.bestprac.ancestry]

You can specify the way that line ending and whitespace changes are handled. These options are described in "Line-end and Whitespace Options"一节. 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.

Reviewing the Merge Results

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.

If your server and all clients are running Subversion 1.5 or higher, the merge tracking facility will record the revisions merged and avoid a revision being merged more than once. This makes your life much simpler as you can simply merge the entire revision range each time and know that only new revisions will actually be merged.

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。

提示

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.

重要

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.

Merge Tracking

Subversion 1.5 introduced facilities for merge tracking. When you merge changes from one tree into another, the revision numbers merged are stored and this information can be used for several different purposes.

- 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 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.

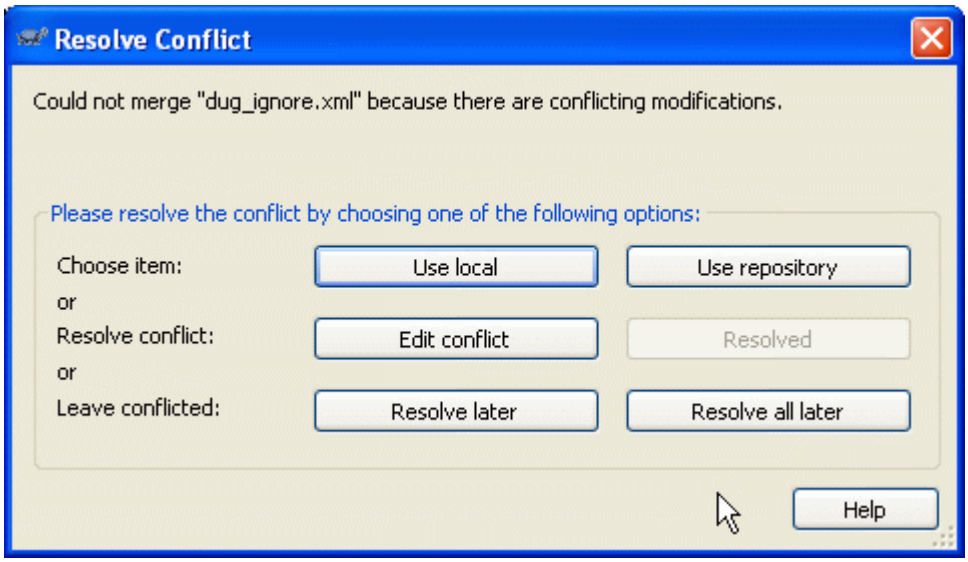
Merge tracking information is stored in the `svn:mergeinfo` property by the client when it performs a merge. When the merge is committed the server stores that information in a database, and when you request merge, log or blame information, the server can respond appropriately. For the system to work 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].

Handling Conflicts during Merge

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 5.38. The Merge Conflict Callback Dialog



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

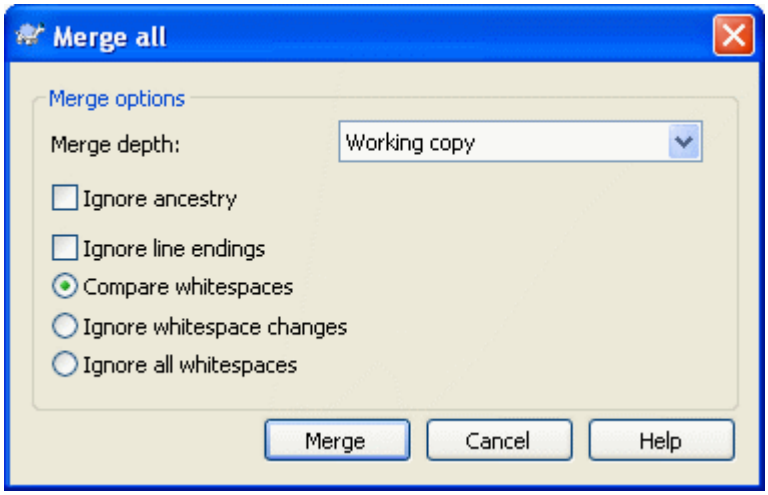
1. You may decide that your local changes are much more important, so you want to discard the version from the repository and keep your local version. Or you might discard your local changes in favour of the repository version. Either way, no attempt is made to merge the changes - you choose one or the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

Merge All Changes

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

图 5.39. The Merge all Dialog



This dialog is very easy. All you have to do is set the options for the merge, as described in "Merge Options"一节. The rest is done by TortoiseSVN automatically using merge tracking.

If you use this feature you must be sure that all clients are using merge tracking, otherwise the merge history may be incomplete. Also, if the server does not have a merge tracking database, gathering the information will be very slow.

锁

使用之前在"复制-修改-合并　方案"一节中描述的"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".

首先，你需要确保你的Subversion服务器更新到至少1.2版本。早期的版本不支持锁定。如果你使用`file:///`进行访问，那么当然只要更新你的客户端就可以了。

锁定在Subverion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。

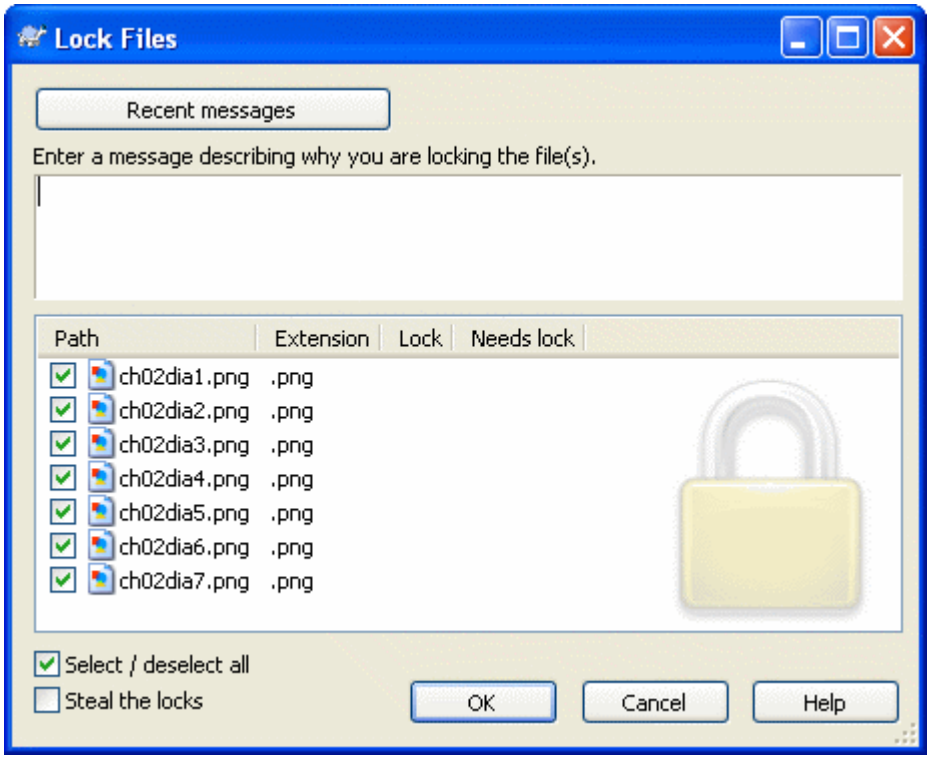
如果你的合作者之一请求一个锁，但却外出旅游去了，你怎么办？Subversion提供了一种强制锁。释放别人拥有的锁被称为破坏锁定，强制获得别人拥有的锁称为窃取锁定。当然，如果你想要与你的合作者保持良好的关系，轻易不要这么做。

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。

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 5.40. 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 ...

释放锁定

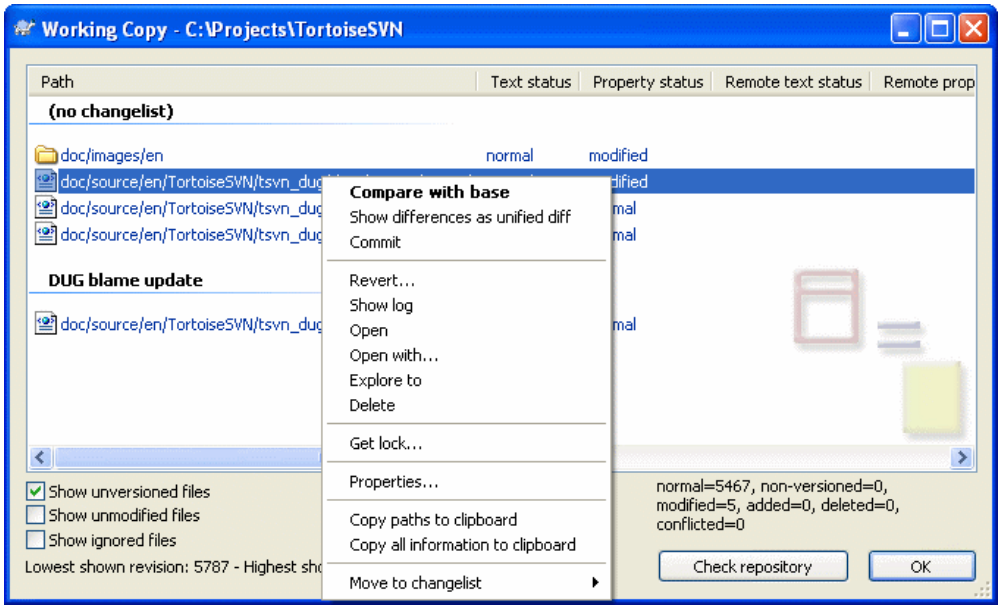
为了确保你不会忘记释放锁，你不需要做别的事，在提交对话框中，总是会显示锁定的文件，并总是默认被选中。如果你继续提交，选中的文件中的锁就被移除了，就算你从没有修改过。如果你不希望释放某文件的锁，你

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。

检查锁定状态

图 5.41. 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 → 检查更新...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。

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"项目设置"一节如何设置属性。带有此属性的文件将总被按只读的方式检出和更新(除非你的工作副本拥有锁定)。

作为提醒的方式之一，TortoiseSVN使用一个特殊的图标表示。

如果你管理的策略要求每个文件都必须被锁定，那么你会发现使用Subversion的auto-props功能，在你每次加入新文件的时候，自动设置属性是很方便的。请阅读"TortoiseSVN的设置"一节。

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 “钩子脚本”一节

创建并应用补丁

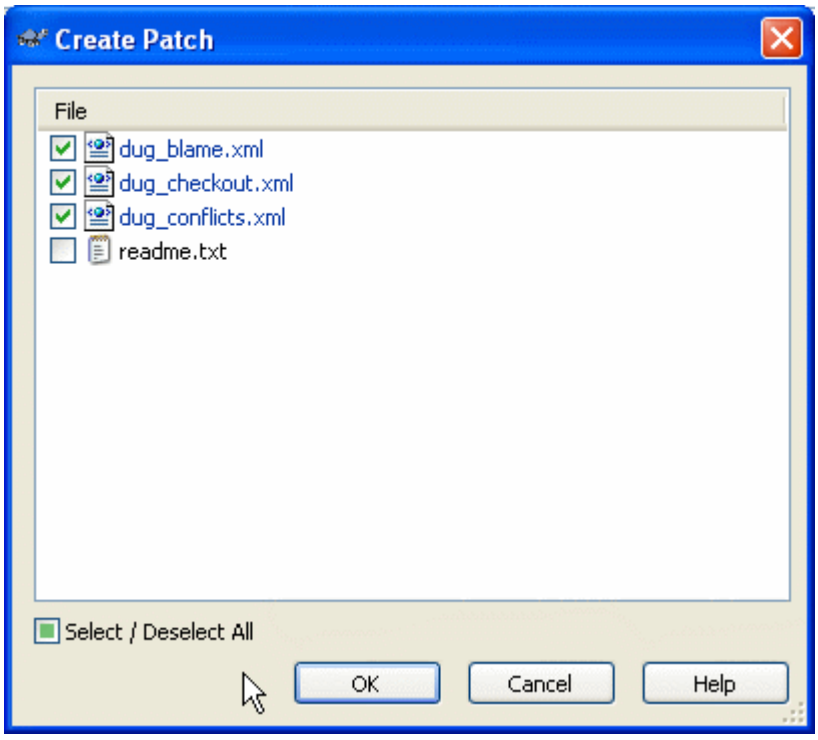
对开源工程(比如本工程)来说，每个人对仓库都有读访问权，并且任何人都可以对该工程做出修改。那么如何控制这些修改呢？如果任何人都可以提交自己的修改，那么这个工程可能永远都会处于不稳定状态，而且很有可能永远的瘫痪下去。在这种情况下，修改需要以补丁文件的形式先递交到有写访问权限的开发组。开发组可以先对该补丁文件进行审查，然后决定将其提交到仓库里或者是退还给作者。

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。

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 5.42. 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“本地与远程状态”一节

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。

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。

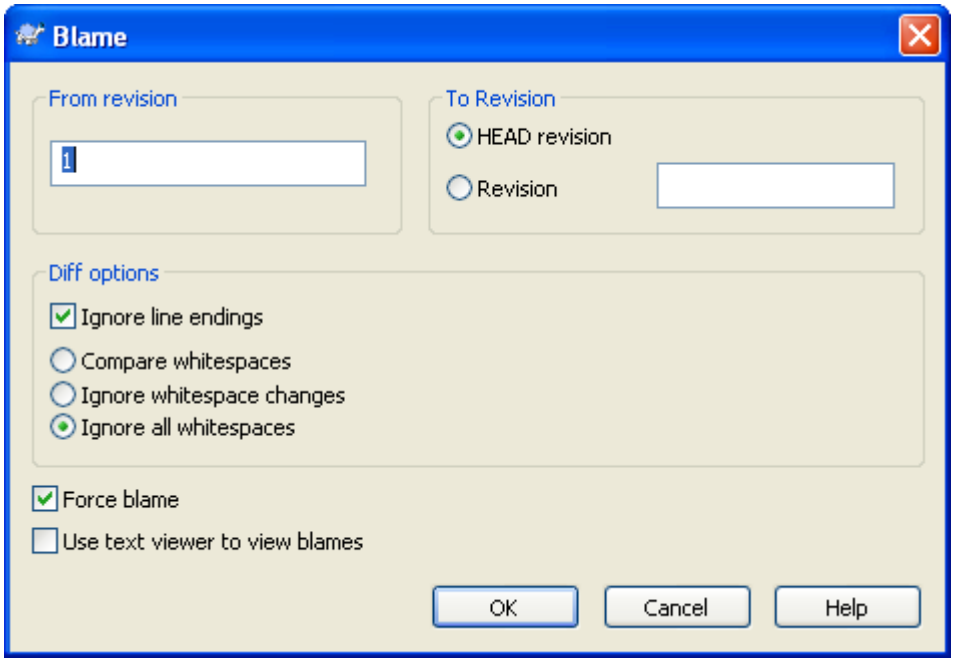
谁修改了哪一行？

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。

追溯文件

图 5.43. 评注/追溯对话框



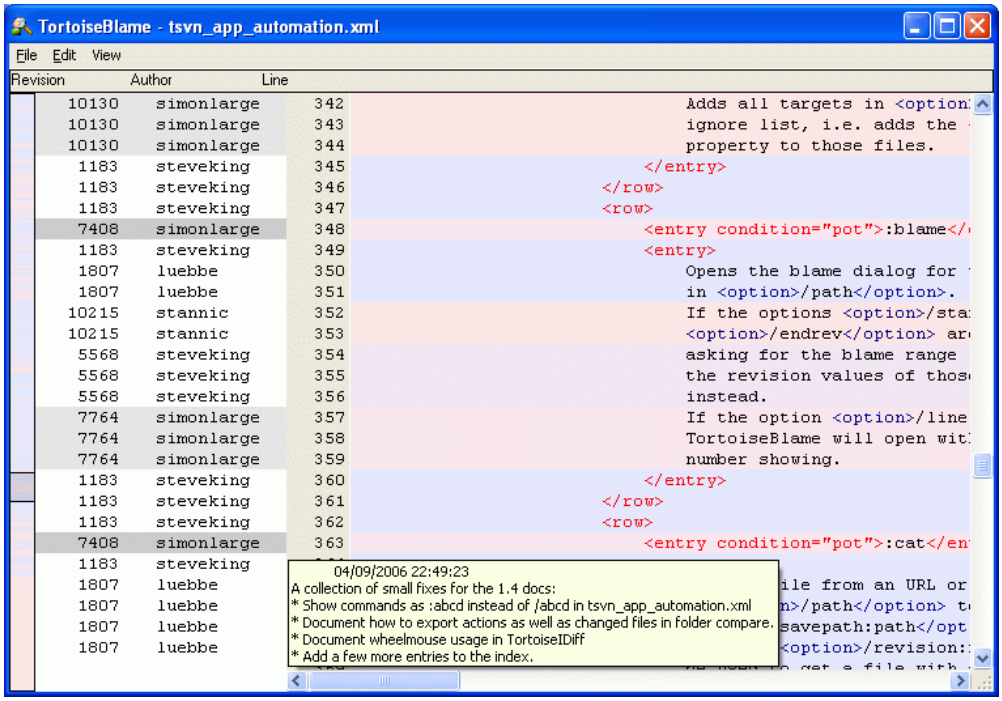
如果对早期版本的修改不感兴趣，你可以设置从哪个版本开始追溯。如果你想追溯每一个版本，你可以把那个数值设置为1。

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。

You can specify the way that line ending and whitespace changes are handled. These options are described in "Line-end and Whitespace Options"一节. The default behaviour is to treat all whitespace and line-end differences as real changes, but if you want to ignore an indentation change and find the original author, you can choose an appropriate option here.

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 5.44. 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 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 “TortoiseBlame 的设置”一节.

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。

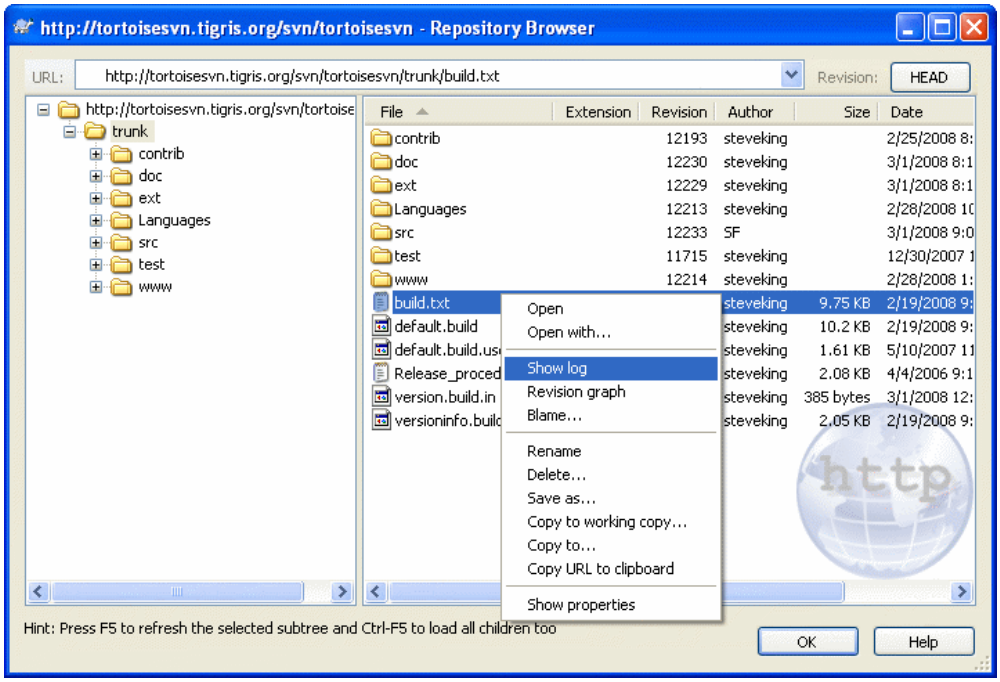
Blame Changes
: Select one revision in the top pane, then pick one file in the bottom pane and select Context menu → Blame changes. This will fetch the blame data for the selected revision and the previous revision, then use the diff viewer to compare the two blame files.

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。

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 5.45. 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。
- View/Edit the folder's properties.
- 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/##/)，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.

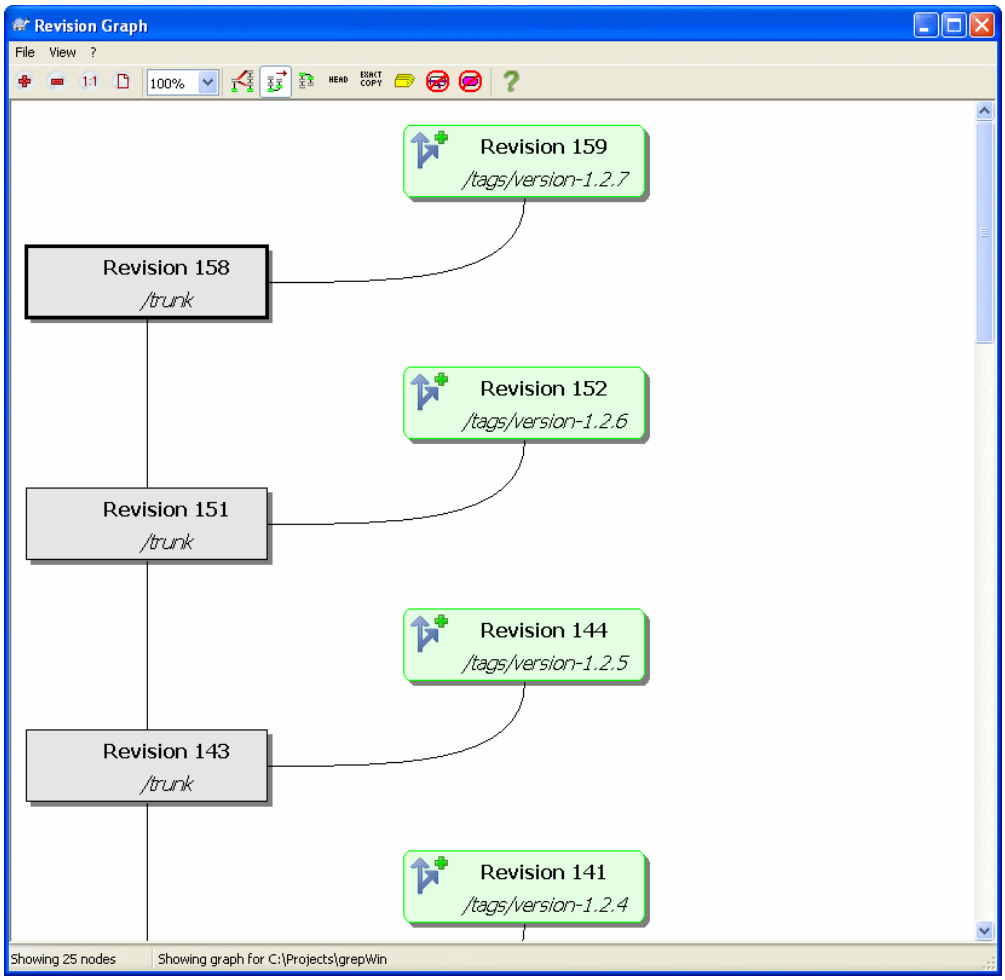
If you want to copy/move a file or folder to another location and also give it a new name at the same time, you can right drag or Ctrl-right drag the item instead of using left drag. In that case, a rename dialog is shown where you can enter a new name for the file or folder.

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.

版本分支图

图 5.46. 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。

重要

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.

Revision Graph Nodes

The revision graph shows several types of node:

增加文件/文件夹
已增加或通过复制生成的文件/文件夹以圆圈的方式标记。

已删除文件/文件夹
已删除文件，比如说一个不在需要的分支，将显示octagon(rectangle with corners cut off)。

分支最新版本
Where a branch (or trunk or tag) has been modified since the last branch node, this is shown using an ellipse. Shown when the Show HEAD revisions option is selected.

一般的文件/文件夹
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 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.

Changing the View

Because a revision graph is often quite complex, there are a number of features which can be used to tailor the view the way you want it. These are available in the View menu and from the toolbar.

Group branches
The default behavior (grouping off) will use one row per revision and all rows are sorted strictly by revision. As a result, long-living branches 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.

Oldest on top
Normally the graph shows the oldest revision at the bottom, and the tree grows upwards. Use this option to grow down from the top instead.

Show HEAD revisions
This ensures that the latest revision on every branch is always shown on the graph.

Exact copy sources
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.

Fold tags
If you want to see a graph of software development, tagged releases may be of little interest to you. 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.

Reduce cross lines
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 it may take a little time to optimise.

Filter
: 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.

Using the Graph

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 dry run. 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.

> 小心
>
> 因为Subversion不能支持提供所有请求的信息，在有大量请求信息时，有时它能提供强大的返回信息。但无论如何，主干都会产生有用的输出。

Refreshing the View

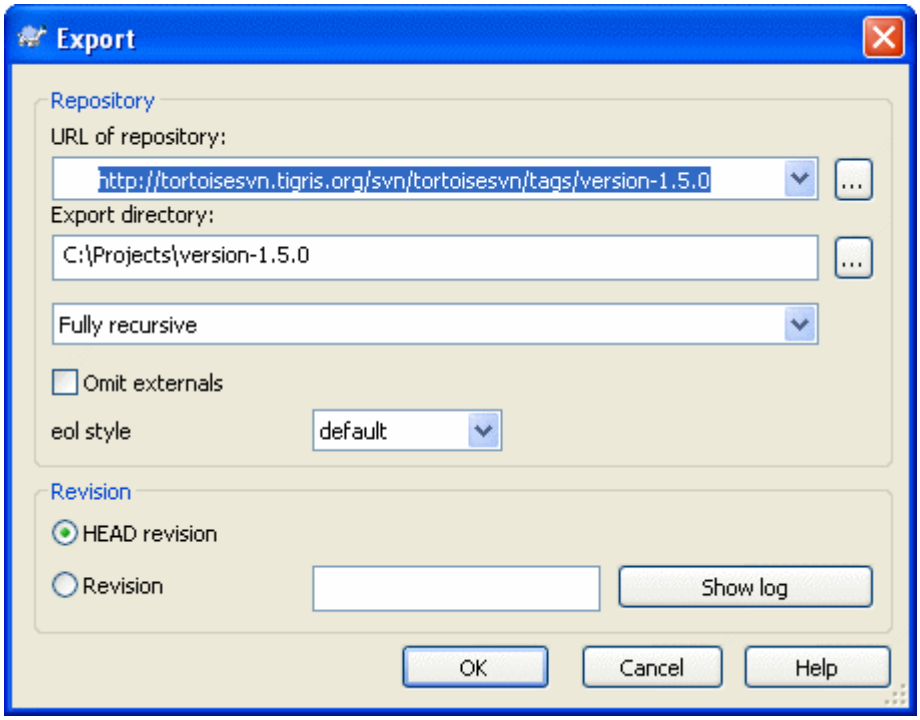
If you want to check the server again for newer information, you can simply refresh the view using F5. If you are using the log cache (enabled by default), this will check the repository for newer commits and fetch only the new ones. If the log cache was in offline mode, this will also attempt to go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

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 5.47. 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`## (1)`。

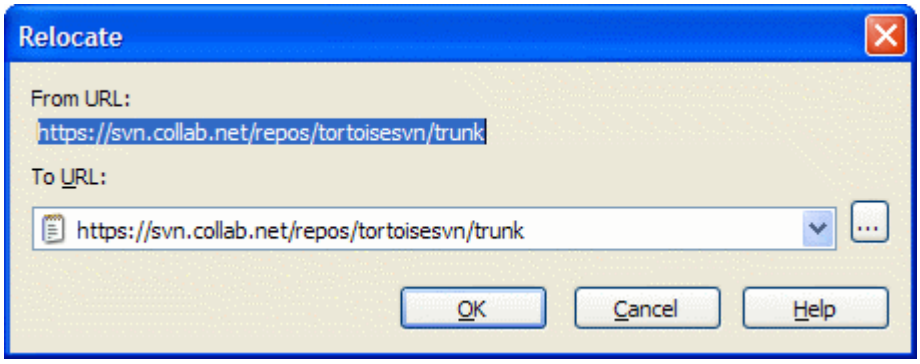
Removing a working copy from version control

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.

重新定位工作副本

图 5.48. 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.

警告

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。

Integration with Bug Tracking Systems / Issue Trackers

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。

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: (“项目设置”一节)

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%`

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.

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. e.g the TortoiseSVN project is using `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.

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 one or two regular expressions, separated by a newline.

  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+)`

  If two expressions are set, then the first expression is used to find a string which relates to the bug ID but may contain more than just the bug ID (e.g. “Issue #123” or “resolves issue 123”). The second

expression is then used to extract the bare bug ID from the string extracted with the first expression. An example:

If you want to catch every pattern "issue #XXX" and "issue #890, #789" inside a log message you could use the following regex strings: `[Ii]ssue #?(\d+)(,? ?#(\d+))*` and the second expression as `(\d+)`

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，`#######`会优先使用。

提示

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！

Some `tsvn:` properties require a `true/false` value. TortoiseSVN also understands `yes` as a synonym for `true` and `no` as a synonym for `false`.

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.

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.tigris.org/svn/tortoisesvn/trunk/doc/issuetrackers.txt].

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: ("项目设置"一节)

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`.

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/`).

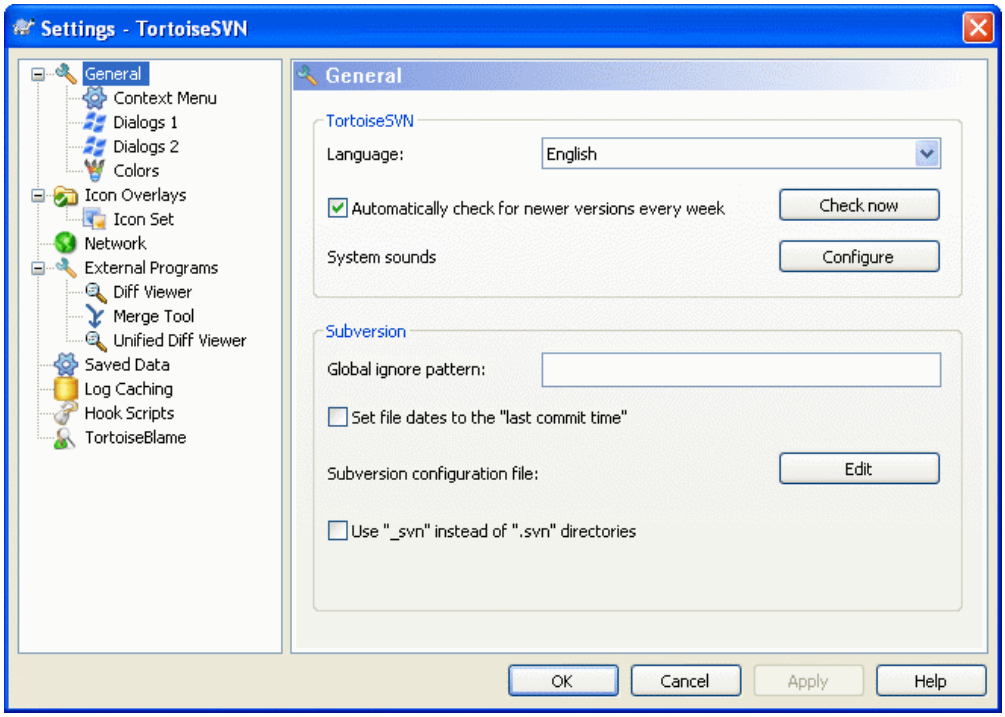
For `tsvn:` properties only you can use the Recursive checkbox to set the property to all sub-folders in the hierarchy, without also setting it on all files.

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。

常规设置

图 5.49. 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

全局忽略模式被用来防止非版本控制的文件在例如提交时的对话框中被列出来。那些符合模式的文件，在执行导入操作时同样被忽略。通过在模式框中输入文件名或扩展名来忽略文件或文件夹。不同的模式之间以空格分隔，例如 `*/bin */obj *.bak *.~?? *.jar *.[Tt]mp`。请记得这些模式可以用来处理含有 N 级父目录的路径。设置忽略模式并非象刚才显示的那么简单，所以请务必阅读 "忽略列表中的模式匹配"一节 以获得更多模式匹配语法信息，和如何检查路径。

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。

> 小心
>
> 如果你象下面段落那样使用Subversion配置文件来设置一个 `#####` 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 "忽略文件和目录"一节 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/1.4/svn.advanced.confarea.html]. The section on Automatic Property Setting [http://svnbook.red-bean.com/en/1.4/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/1.4/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] to find out more.

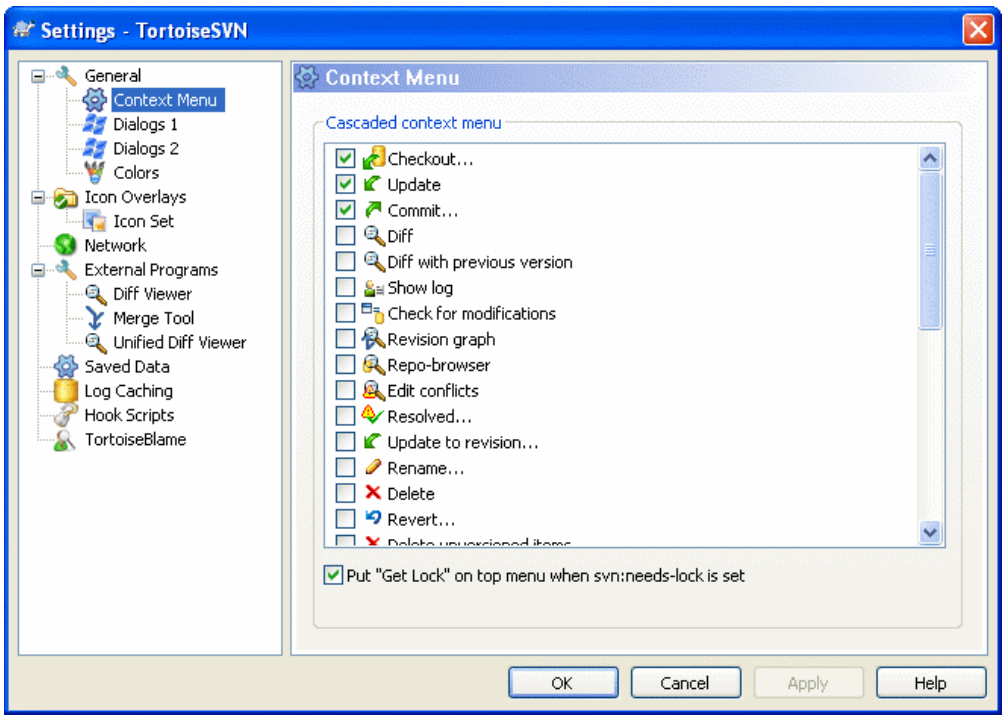
Use `_svn` instead of `.svn` directories
在使用VS.NET环境做web工程时，将无法处理 `.svn` 文件夹，但Subversion是要用这些文件夹来储存自己的内部信息的。这可不是Subversion的bug，这bug是VS.NET和它使用的frontpage扩展带来的。阅读 "Subversion 的工作文件夹"一节 来获得有关此问题的更多信息。

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。

Context Menu Settings

图 5.50. The Settings Dialog, Context Menu Page

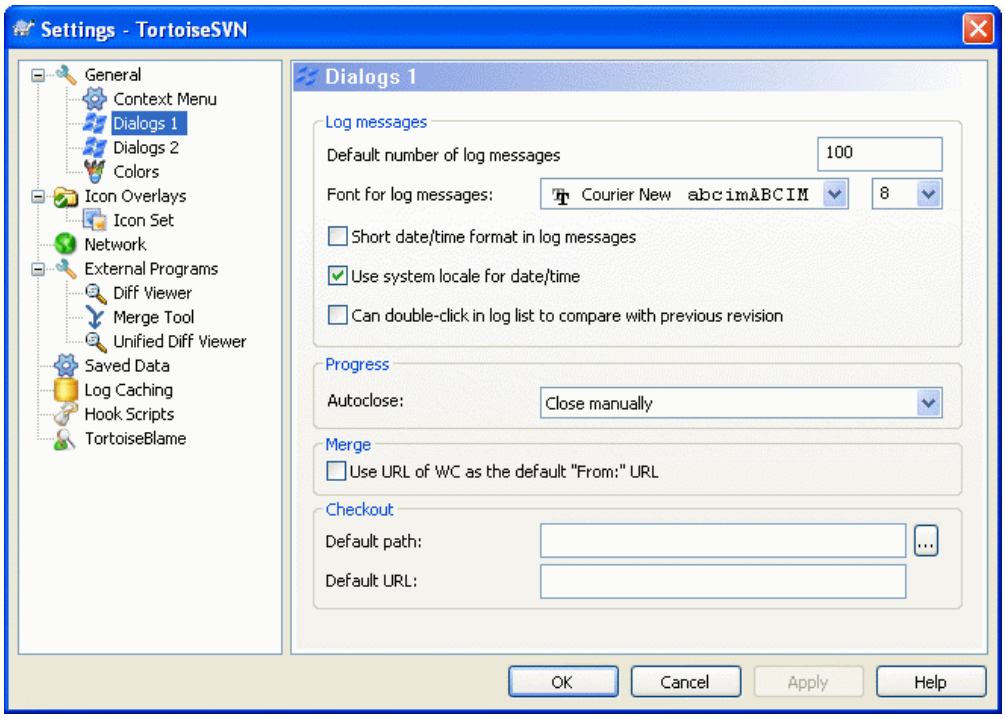


该页面允许你指定: 在TortoiseSVN的主上下文菜单中哪些条目可以直接在鼠标右键菜单显示，哪些在TortoiseSVN子菜单显示。默认情况下很多项未被勾选，只在子菜单显示。

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。

TSVN对话框设置一

图 5.51. 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。

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。

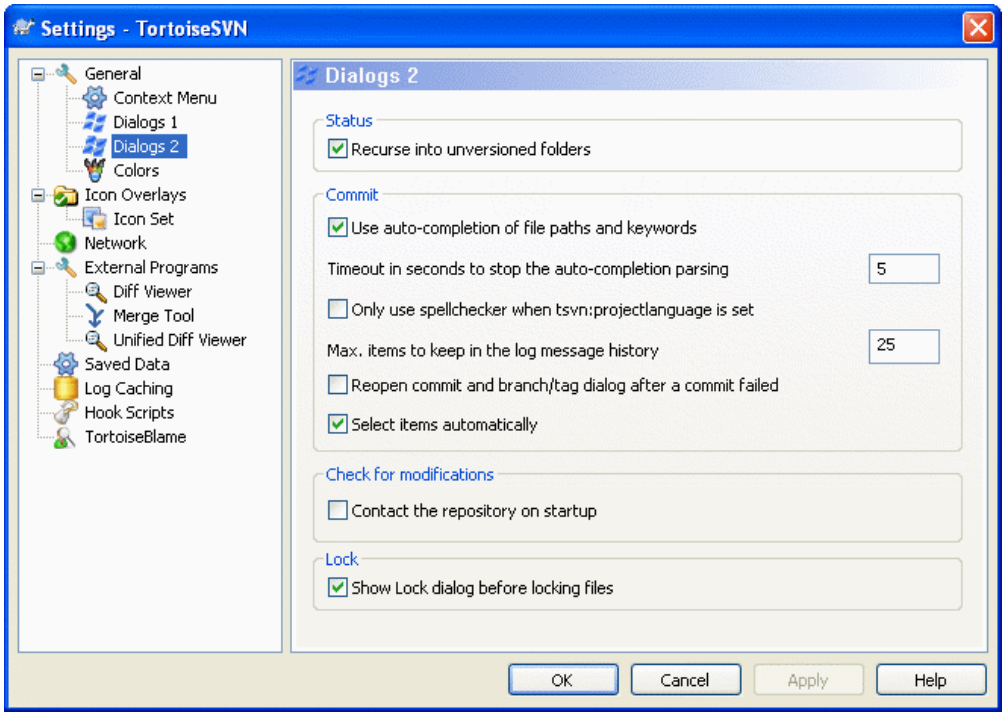
Use URL of WC as the default “From:” URL
: 在合并对话框里，默认行为是在每次合并中记忆 起始: 的URL。无论如何，都有某些人喜欢在他们的版本进化树中从很多不同的位置执行合并操作，他们发现从当前工作副本的URL开始更方便些。该URL可以随后被编辑来指向一个同级路径或另一个分支。

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。

TSVN对话框设置二

图 5.52. 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 检查更新 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。

Use auto-completion of file paths and keywords
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.

Timeout in seconds to stop the auto-completion parsing
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.

Only use spellchecker when `tsvn:projectlanguage` is set
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
TSVN可以为每个版本库保存你访问时所输入的最后25条日志信息。你可以自定义该数目。若你有很多不同的版本库，你可能会希望减少该数目以防止向注册表中填入过多信息。

Re-open commit and branch/tag dialog after a commit failed
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。

Select items automatically
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
"检查更新"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。

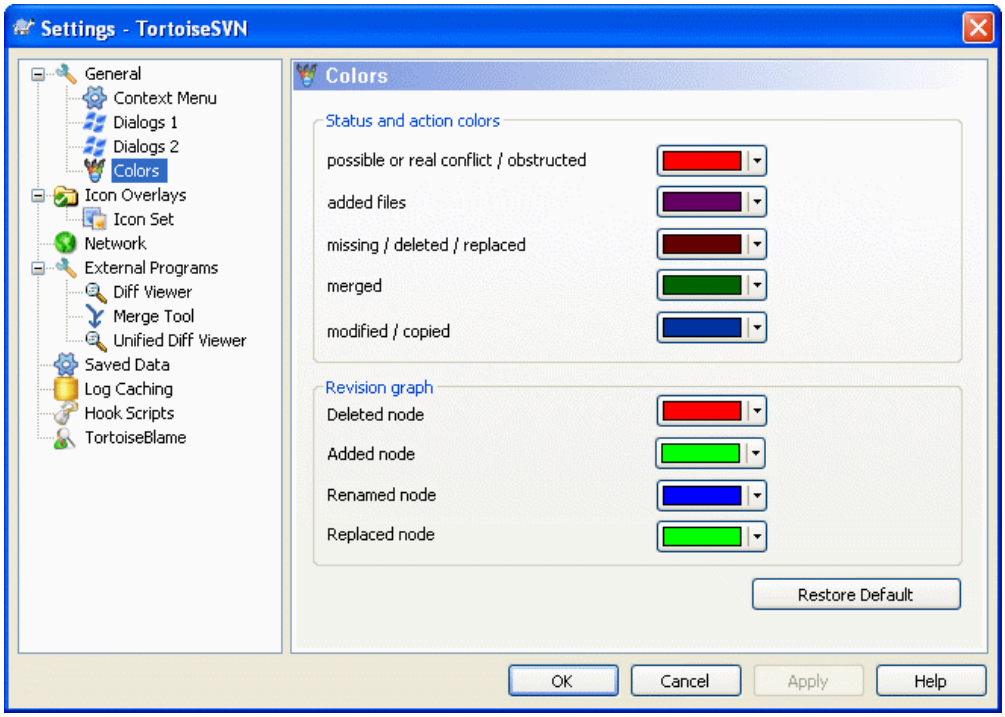
在锁定文件之前显示加锁对话框
当你选择一个或多个文件，然后选择 TortoiseSVN → 加锁 后，一些项目的惯例是写加锁信息，解释你为什么锁定这些文件。如果你不使用加锁信息，可以取消此选择框，从而略过对话框，直接锁定文件。

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。

TortoiseSVN 颜色设置

图 5.53. 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。

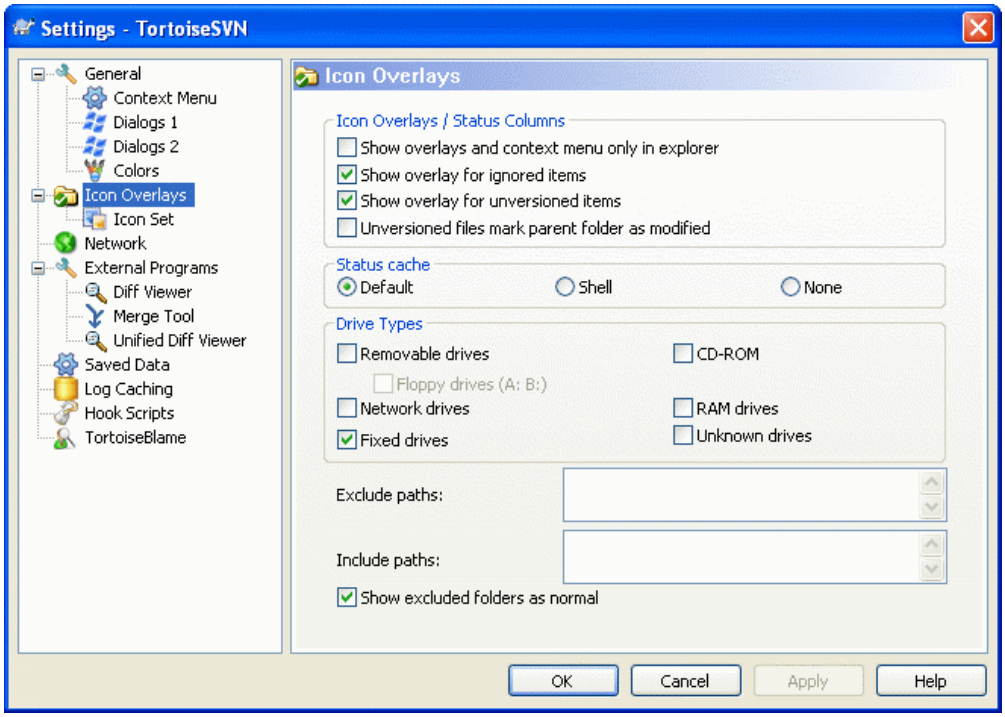
添加的节点
一个通过添加、复制或移动操作，已经被添加到版本库的条目。

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。

图标叠加设置

图 5.54. 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。

Advantage: needs only very little memory (around 1 MB of RAM) and can show the status in real time.

缺点:　因为仅有一个文件夹被缓存，图标覆盖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覆盖。文件夹也仅仅有个"正常"状态的图标覆盖，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覆盖和状态列。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覆盖，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、文件夹上看到图标覆盖。

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覆盖。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覆盖了。

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覆盖显示的特定驱动类型下，或是处在上面的排除路径之下， 也依然会显示图标覆盖。

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.

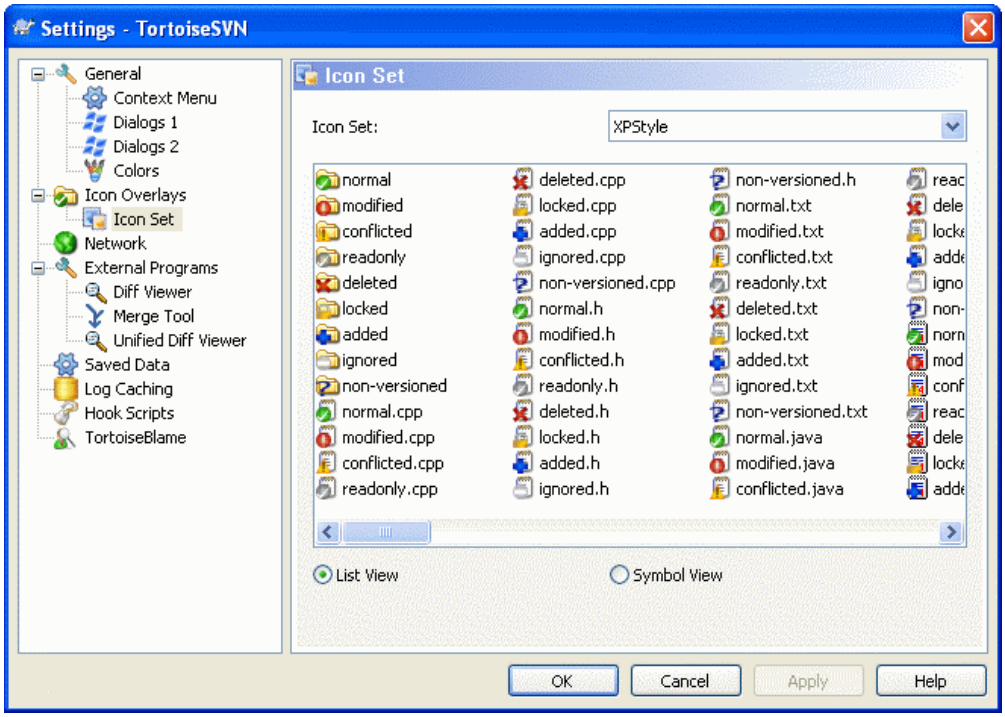
TSVNCache.exe 同样使用这些路径来限制它的扫描。如果你想让它仅仅在某些特定文件夹里监视，就取消所有的驱动器类型，并仅仅包含你允许被扫描的文件夹。

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.

图标集选择

图 5.55. 设置对话框，图标集页面



你可以选择你最喜欢的覆盖图标集。要注意的是，倘若改变了覆盖图标集，你可能需要重启计算机使更改生效。

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/1.4/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.

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。

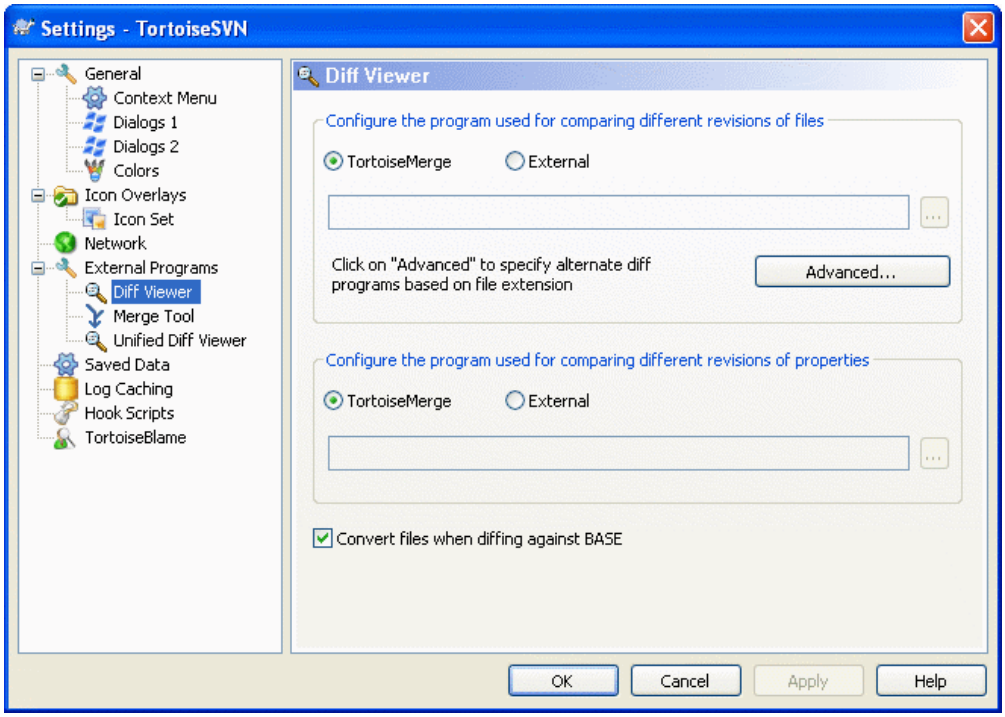
TortoisePlink does not have any documentation of its own because it is just a minor variant of Plink. Find out about command line parameters from the PuTTY website [http://www.chiark.greenend.org.uk/~sgtatham/putty/]

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].

外部程序设置

图 5.56. 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 "其他的比较/合并工具"一节 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。

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 `### : ## 123` 和 `### : ####`

例如，使用 ExamDiff Pro：

```
C:\Path-To\ExamDiff.exe %base %mine
```

或者使用 KDiff3：

```
C:\Path-To\kdiff3.exe %base %mine --L1 %bname --L2 %yname
```

或者使用 WinMerge：

```
C:\Path-To\WinMerge.exe -e -ub -dl %bname -dr %yname %base %mine
```

或者使用 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.

合并工具

外部合并程序被用来解决文件冲突。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

例如，使用 Perforce Merge：

```
C:\Path-To\P4Merge.exe %base %theirs %mine %merged
```

或者使用 KDiff3：

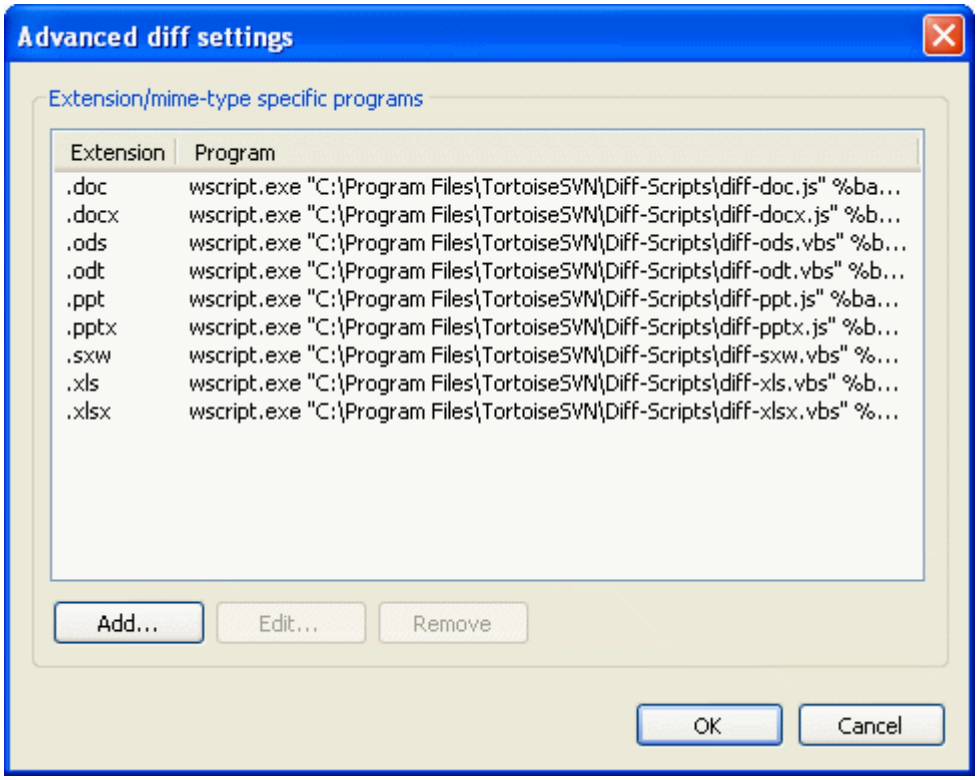
```
C:\Path-To\kdiff3.exe %base %mine %theirs -o %merged
    --L1 %bname --L2 %yname --L3 %tname
```

或者使用 Araxis：

```
C:\Path-To\compare.exe /max /wait /3 /title1:%tname /title2:%bname
    /title3:%yname %theirs %base %mine %merged /a2
```

差异查看/合并工具的高级设置

图 5.57. 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`。

统一的差异查看器

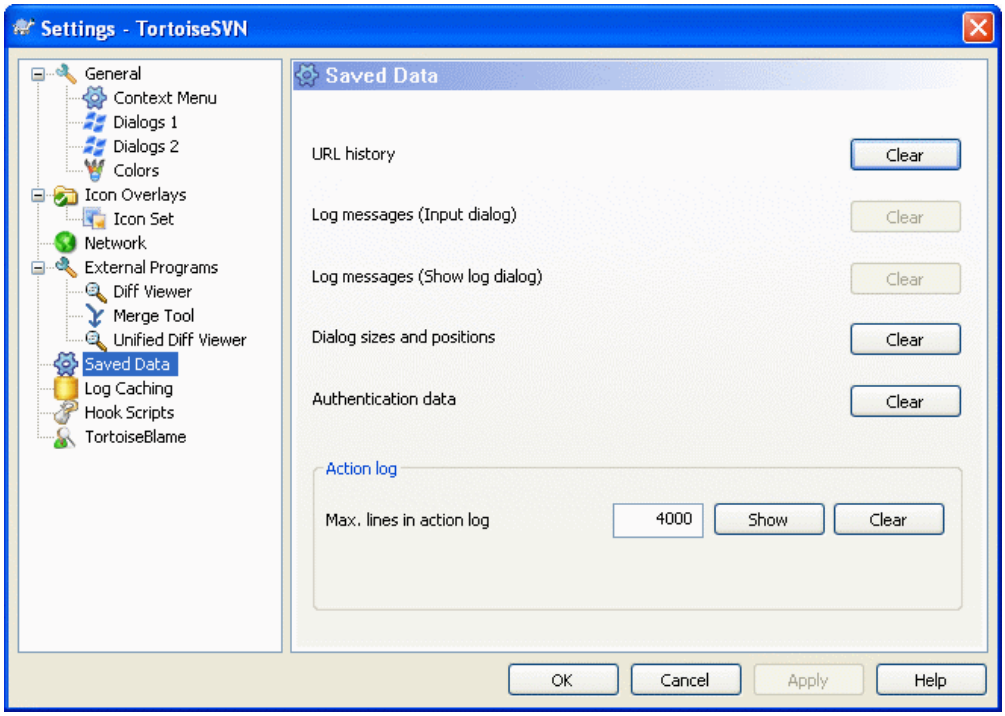
一个统一差异文件(补丁文件)的查看程序。不需要任何参数。默认 选项遵循先检查 `.diff` 文件，再检查 `.txt` 文件的顺序。如果你没有 `.diff` 文件的查看器，就需要用记事本来查看了。

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。

已保存数据的设置

图 5.58. 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.

Log messages (Input dialog)
: 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。

Log messages (Show log dialog)
: 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 Dialogs 1 tab.

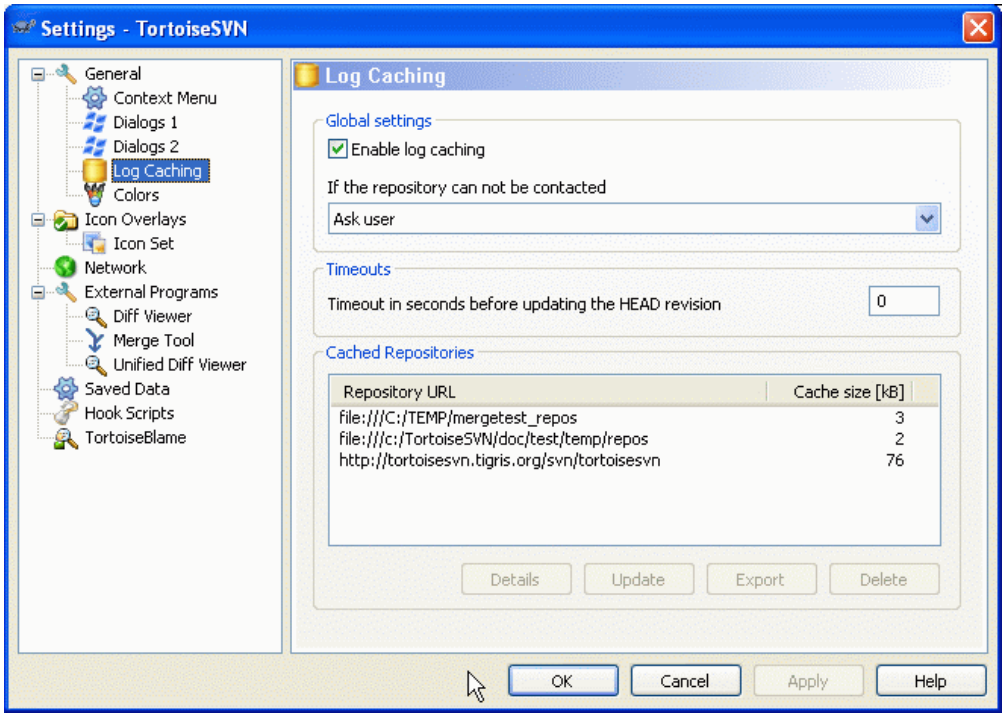
窗口大小及位置
: 许多对话框都可以记录你最后一次使用时的窗口大小和位置。

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 "认证"一节 for instructions on how to find the cached data.

Log Caching

图 5.59. The Settings Dialog, Log Cache Page



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.

Enable log caching
: 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.

If the repository cannot be contacted
: 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.

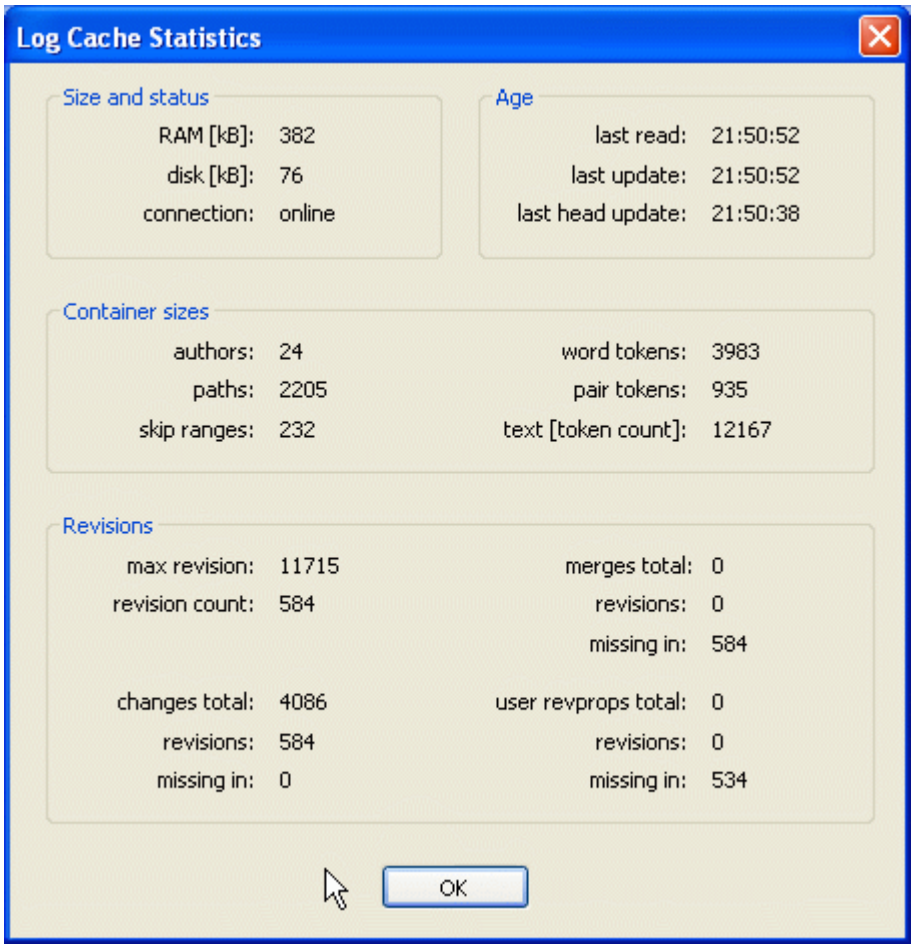
  When log data is being taken from the cache without contacting the server, the dialog using those message will show the offline state in its title bar.

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.

Below the settings 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.

Log Cache Statistics

图 5.60. The Settings Dialog, Log Cache Statistics



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.

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

Connection
: Shows whether the repository was available last time the cache was used.

Last update
: The last time the cache content was changed.

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

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

Paths
: 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.

Max revision
: The highest revision number stored in the cache.

Revision count
: The number of revisions stored in the cache. This is another measure of cache completeness.

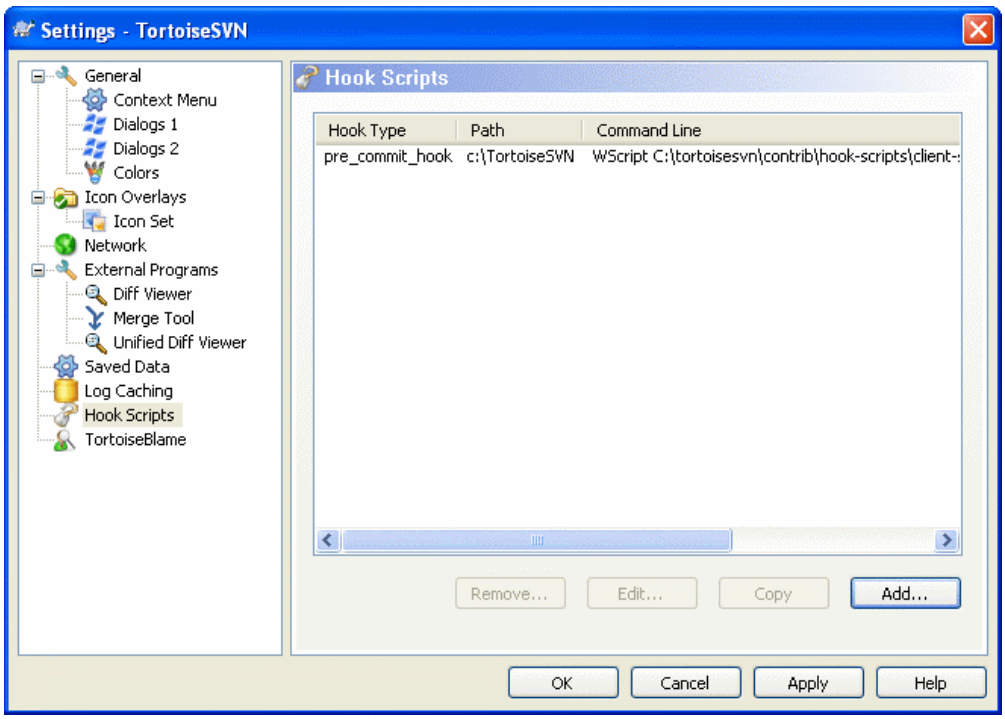
Click on the Update to completely refresh the cache and fill in any holes. For a large repository this could be very time consuming, but useful if you are about to go offline and want the best available cache.

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.

客户端钩子脚本

图 5.61. 设置对话框，钩子脚本页

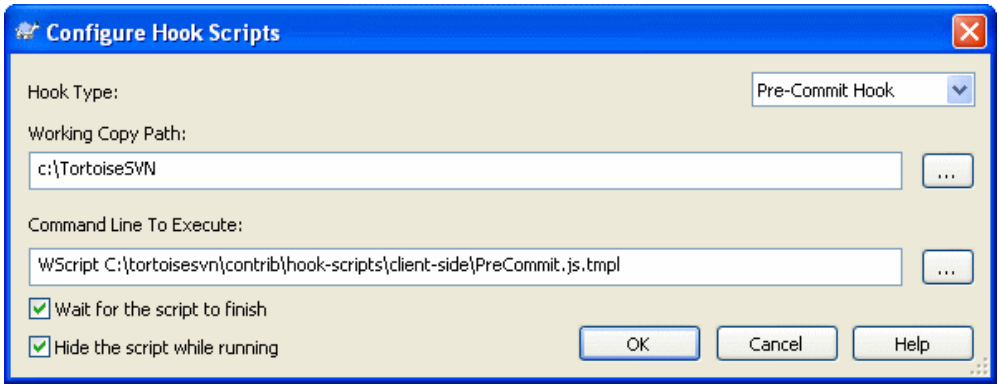


This dialog allows you to set up hook scripts which will be executed automatically when certain Subversion actions are performed. As opposed to the hook scripts explained in "钩子脚本"一节, these scripts are executed locally on the client.

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 5.62. 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.

提交之前
: 在提交对话框点击确认之后，实际提交之前调用。

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 available 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
The depth with which the commit/update is done.

Possible values are:

-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
The repository revision to which the update should be done or after a commit completes.

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

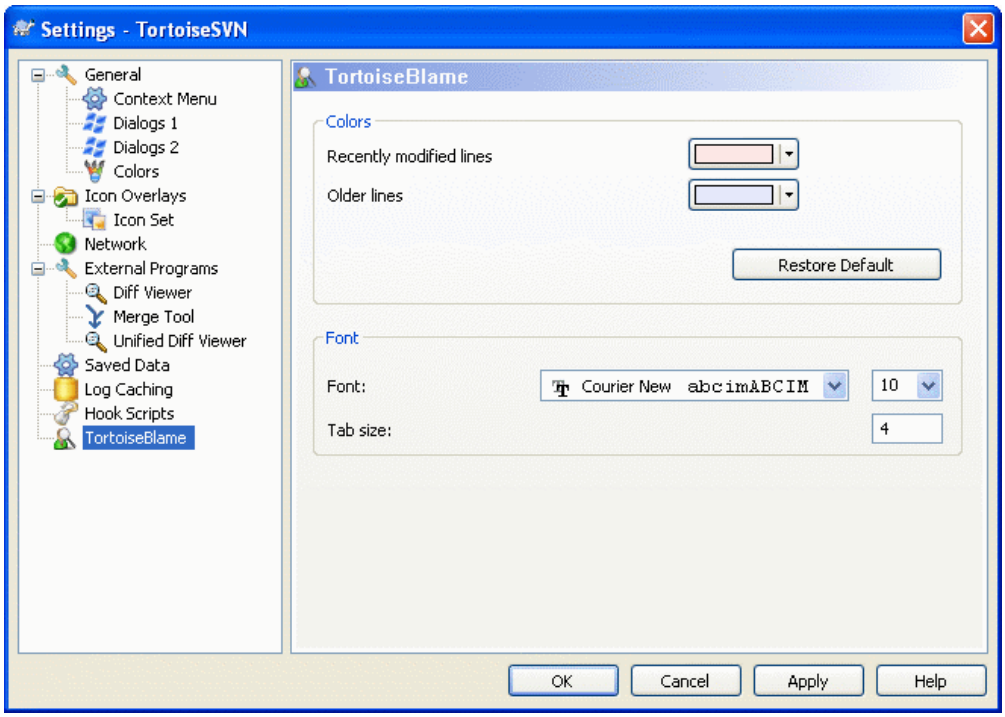
CWD
The current working directory with which the script is run. This is set to the common root directory of all affected paths.

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.

TortoiseBlame 的设置

图 5.63. 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。

注册表设置

一些极不常用的设置只有通过直接修改注册表的方式才能生效。

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。

Cache tray icon
: 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)

Filenames without extensions in auto-completion list
: 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`.

Merge log separator
: 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.

Always blame changes with TortoiseMerge
: TortoiseSVN allows you to assign external diff viewer. Most such viewers, however, are not suited for change blaming ("追溯不同点"一节), 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`.

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 "常规设置"一节.

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].

最后步骤

> **Donate!**
>
> 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 6 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。

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`。

表 6.1. 列出可用的命令行开关

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 '0X' prepended.

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:

表 6.2. 列出可用的命令行开关

关键字	描述
$WCREV$	用工作副本中最高的提交版本来替换
$WCDATE$	Replaced with the commit date/time of the highest commit revision. By default, international format is used: `yyyy-mm-dd hh:mm:ss`. Alternatively, you can specify a custom format which will be used with `strftime()`, for example: `$WCDATE=%a %b %d %I:%M:%S %p$`. For a list of available formatting characters, look at the online reference [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 is as described above.
$WCRANGE$	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。
$WCMIXED$	`$WCMIXED?TText:FText$` is replaced with `TText` if there are mixed update revisions, or `FText` if not.
$WCMODS$	`$WCMODS?TText:FText$` is replaced with `TText` if there are local modifications, or `FText` if not.
$WCURL$	用传递给SubWCRev的工作目录的版本库地址替换。
$WCNOW$	Replaced with the current time and date
$WCNOW=	Replaced with the current time and date in standard format
$WCINSVN$	`$WCINSVN?TText:FText$` 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
$WCLOCKOWNER$	Replaced with the name of the lock owner
$WCLOCKCOMMENT$	Replaced with the comment of the lock

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://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/src/SubWCRev";

#if 1
#error Source is modified
#endif

// End of file
```

提示

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.

COM interface

If you need to access Subversion revision information from other programs, you can use the COM interface of SubWCRev. The object to create is `SubWCRev.object`, and the following methods are supported:

表 6.3. COM/automation methods supported

Method	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	The highest commit revision in the working copy. Equivalent to `$WCREV$`
.Date	The commit date/time of the highest commit revision. Equivalent to `$WCDATE$`
.Author	The author of the highest commit revision, that is, the last person to commit changes to the working copy.
.MinRev	The minimum update revision, as shown in `$WCRANGE$`
.MaxRev	The maximum update revision, as shown in `$WCRANGE$`
.HasModifications	True if there are local modifications
.Url	Replaced with the repository URL of the working copy path used in `GetWCInfo`. Equivalent to `$WCURL$`

The following example shows how the interface might be used.

```
// testCOM.js - javascript file
// test script for the SubWCRev COM/Automation-object

filesystem = new ActiveXObject("Scripting.FileSystemObject");

SubWCRev1 = new ActiveXObject("SubWCRev.object");
SubWCRev2 = new ActiveXObject("SubWCRev.object");
SubWCRev3 = new ActiveXObject("SubWCRev.object");

SubWCRev1.GetWCInfo(filesystem.GetAbsolutePathName("."), 0, 0);
SubWCRev2.GetWCInfo(filesystem.GetAbsolutePathName(".."), 1, 1);
SubWCRev3.GetWCInfo(filesystem.GetAbsolutePathName("SubWCRev.cpp"), 0, 0);

sInfo1 = "Revision = " + SubWCRev1.Revision +
         "\nMin Revision = " + SubWCRev1.MinRev +
         "\nMax Revision = " + SubWCRev1.MaxRev +
```

```
          "\nDate = " + SubWCRev1.Date +
          "\nURL = " + SubWCRev1.Url +
          "\nAuthor = " + SubWCRev1.Author +
          "\nHasMods = " + SubWCRev1.HasModifications;
sInfo2 = "Revision = " + SubWCRev2.Revision +
          "\nMin Revision = " + SubWCRev2.MinRev +
          "\nMax Revision = " + SubWCRev2.MaxRev +
          "\nDate = " + SubWCRev2.Date +
          "\nURL = " + SubWCRev2.Url +
          "\nAuthor = " + SubWCRev2.Author +
          "\nHasMods = " + SubWCRev2.HasModifications;
sInfo3 = "Revision = " + SubWCRev3.Revision +
          "\nMin Revision = " + SubWCRev3.MinRev +
          "\nMax Revision = " + SubWCRev3.MaxRev +
          "\nDate = " + SubWCRev3.Date +
          "\nURL = " + SubWCRev3.Url +
          "\nAuthor = " + SubWCRev3.Author +
          "\nHasMods = " + SubWCRev3.HasModifications;

WScript.Echo(sInfo1);
WScript.Echo(sInfo2);
WScript.Echo(sInfo3);
```

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。

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.

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。

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。

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 “项目设置”一节

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。

Roll back (Undo) revisions in the repository

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. 如果想要把以前的某个版本变成最新版本，右键点击选中的版本(范围)，然后选择右键菜单 → 恢复到此版本。就能够撤销被选中版本后面所有的变更。

工作副本已经恢复到了变更以前的状态。检查恢复后的结果，然后提交变更。

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。

Use `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/1.4/svn.reposadmin.maint.html].

比较一个文件的两个版本

如果希望比较某个文件的两个历史版本，比如同一个文件修订版本100和200，可以用TortoiseSVN → 显示日志列出这个文件的历史版本纪录，选择希望比较的两个版本，然后使用右键菜单 → 比较版本差异。

如果希望比较两个不同目录树下的同一个文件，比如主干和分支，可以使用版本库浏览器打开两个目录树，在两个目录树下选择同一个文件，然后使用 右键菜单 → 比较版本差异。

如果希望比较两个目录树下的所有变化，比如主干和某个发布标签，可以使用TortoiseSVN → 版本分支图。选择两个想要比较的节点，然后使用右键菜单 → 比较最新版本，就会列出一个变更文件列表。在列表上选择单个文件就能够浏览该文件的具体变更内容。另外一个方法是使用右键菜单 → 最新版本的标准差异(Unified diff)显示所有变更的汇总和最少限度的上下文。

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。

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 the 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 "引用的工程"一节.

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项目的修改。

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。

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。

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。

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。

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. Hold the Shift key to get the extended context menu and use TortoiseSVN → Delete (keep local) to mark the file/folder for deletion from the repository without losing the local copy.
2. TortoiseSVN → Commit the parent folder.
3. Add the file/folder to the ignore list so you don't get into the same trouble again.

Unversion a working copy

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 “Removing a working copy from version control”一节 to find out how.

Remove a working copy

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。

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.

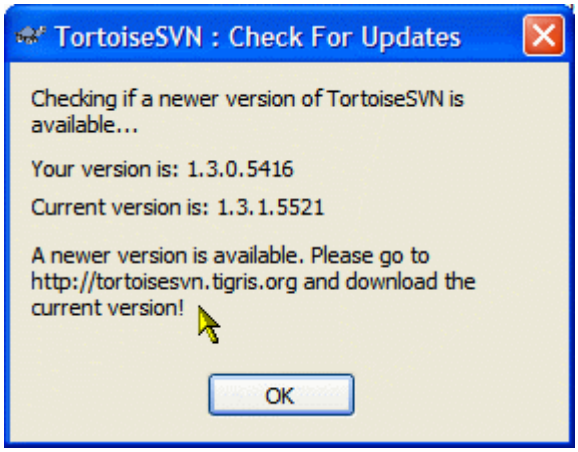
A good walk-through on how to do that can be found in the knowledge base article 314934 from Microsoft: http://support.microsoft.com/?kbid=314934.

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.

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.

Setting the SVN_ASP_DOT_NET_HACK environment variable

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
```

Disable context menu entries

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!

The registry keys which hold the information on which context menus to show are `HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskLow` and `HKEY_CURRENT_USER\Software\TortoiseSVN\ContextMenuEntriesMaskHigh`.

Each of these registry entries is a DWORD value, with each bit corresponding to a specific menu entry. A set bit means the corresponding menu entry is deactivated.

表 C.1. Menu entries and their values

Value	Menu entry
0x0000000000000001	检出
0x0000000000000002	更新
0x0000000000000004	提交
0x0000000000000008	添加
0x0000000000000010	恢复
0x0000000000000020	清理
0x0000000000000040	解决
0x0000000000000080	切换
0x0000000000000100	导入
0x0000000000000200	输出
0x0000000000000400	Create Repository here
0x0000000000000800	分支/标记
0x0000000000001000	合并
0x0000000000002000	删除
0x0000000000004000	改名
0x0000000000008000	Update to revision
0x0000000000010000	差异
0x0000000000020000	显示日志
0x0000000000040000	编辑冲突
0x0000000000080000	重新定位
0x0000000000100000	Check for modifications
0x0000000000200000	Ignore
0x0000000000400000	Repository Browser
0x0000000000800000	追溯
0x0000000001000000	创建补丁
0x0000000002000000	应用补丁(Apply Patch)
0x0000000004000000	Revision graph
0x0000000008000000	锁
0x0000000010000000	Remove Lock
0x0000000020000000	Properties
0x0000000040000000	Diff with URL
0x0000000080000000	Delete unversioned items
0x2000000000000000	Settings
0x4000000000000000	Help
0x8000000000000000	About

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.

To enable the menu entries again, simply delete the two registry keys.

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。

> **重要**
>
> 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。

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。

表 D.1. 有效命令及选项列表

	These pre-fill the relevant fields in the merge dialog.
:mergeall	Opens the merge all dialog. The `/path` specifies the target directory.
复制	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.
设置	打开设置对话框。
删除	从版本控制里移除`/path`中的文件。
改名	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`。
比较	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. The options `url1`, `url2`, `revision1` and `revision2` must be specified. The options `pegrevision`, `ignoreancestry`, `blame` and `unified` are optional.
冲突	打开TortoiseSVN设置的冲突工具，在`/path`中需要设置冲突文件的正确文件。
重新定位	打开重定位对话框，`/path`指定了重定位的工作副本路径。
Windows 外壳	打开帮助文件
状态	Opens the check-for-modifications dialog. The `/path` specifies the working copy directory.
版本库浏览器	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.
忽略	Adds all targets in `/path` to the ignore list, i.e. adds the `svn:ignore` property to those files.
追溯	Opens the blame dialog for the file specified in `/path`. If the options `/startrev` and `/endrev` are set, then the dialog asking for the blame range is not shown but the revision values of those options are used instead. If the option `/line:nnn` is set, TortoiseBlame will open with the specified line number showing. The options `/ignoreeol`, `/ignorespaces` and `/ignoreallspaces` are also supported.
: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。
创建补丁	创建`/path`下的补丁文件。
版本图	显示`/path`目录下的版本变化图。
加锁	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`.
属性	显示 `/path` 给出的路径之属性对话框。

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
```

TortoiseIDiff Commands

The image diff tool has a few command line options which you can use to control how the tool is started. The program is called `TortoiseIDiff.exe`.

The table below lists all the options which can be passed to the image diff tool on the command line.

表 D.2. List of available options

Option	描述
:left	Path to the file shown on the left.
:lefttitle	A title string. This string is used in the image view title instead of the full path to the image file.
:right	Path to the file shown on the right.
:righttitle	A title string. This string is used in the image view title instead of the full path to the image file.
:overlay	If specified, the image diff tool switches to the overlay mode (alpha blend).
:fit	If specified, the image diff tool fits both images together.
:showinfo	Shows the image info box.

Example (which should be entered on one line):

```
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。

> 注意
>
> 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.

约定和基本规则

在这个描述里，版本库位置URL使用`URL`显示，一个例子是`http://tortoisesvn.tigris.org/svn/tortoisesvn/trunk`，工作副本使用`PATH`显示，一个例子是`C:\TortoiseSVN\trunk`。

> 重要
>
> 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。

TortoiseSVN 命令

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。

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。

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。

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...
```

####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。

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。

显示日志

```
svn log -v -r 0:N --limit 100 [--stop-on-copy] PATH
  ##
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。

检查所作的修改

```
svn status -v PATH
  ##
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。

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。

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####。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。

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。

已解决

```
svn resolved PATH
```

改名

```
svn rename CURR_PATH NEW_PATH
```

删除

```
svn delete PATH
```

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...
```

清理

```
svn cleanup PATH
```

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...
```

####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。

释放锁

```
svn unlock PATH
```

分支/标记

```
svn copy -m "LogMessage" URL URL
  #
svn copy -m "LogMessage" URL@rev URL@rev
  #
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。

####是日志编辑框的内容。它可以为空。

切换

```
svn info URL_of_WC
svn switch [-r rev] URL PATH
```

合并

```
svn merge [--dry-run] --force From_URL@revN To_URL@revM PATH
```

Dry run与使用--dry-run选项的merge相同。

```
svn diff From_URL@revN To_URL@revM
```

Unified diff显示了用来合并的区别操作。

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。

重新定位

```
svn switch --relocate From_URL To_URL
```

在当前位置创建版本库

```
svnadmin create --fs-type fsfs PATH
  #
svnadmin create --fs-type bdb PATH
```

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。

导入

```
svn import -m LogMessage PATH URL
```

`####`是日志编辑框的内容。它可以为空。

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.

加入忽略列表

```
svn propget svn:ignore PATH > tempfile
{#########tempfile###}
svn propset svn:ignore -F tempfile PATH
```

因为svn:ignore通常是多行的，这里是通过文件显示，而不是直接使用命令行操作。

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.

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. Implementation Details

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

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.

The Locked overlay is used when the local working copy holds a lock for that file.

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.

An item can only have one Subversion status value. For example a file could be locally modified and it could be marked for deletion at the same time. Subversion returns a single status value - in this case `deleted`. Those priorities are defined within Subversion itself.

When TortoiseSVN displays the status recursively (the default setting), each folder displays an overlay reflecting its own status and the status of all its children. In order to display a single summary overlay, we use the priority order shown above to determine which overlay to use, with the Conflicted overlay taking highest priority.

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. Securing Svnserve using SSH

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!

Setting Up a Linux Server

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.

Setting Up a Windows Server

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
```

SSH Client Tools for use with TortoiseSVN

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`.

Creating OpenSSH Certificates

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.

Create Keys using ssh-keygen

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`.

Create Keys using PuTTYgen

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
```

Test using PuTTY

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 `.PKK` 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
```

Testing SSH with TortoiseSVN

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.

SSH Configuration Variants

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.

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.

术语表

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。
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。
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。
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版以前的缺省版本库格式。
分支	有一个版本控制系统经常使用，来描述在某个时间点的两个人追随不同的路径的术语。你可以创建一个分支离开开发的主线来添加一个新特性，而不必影响主线的稳定，或者你可以创建一个分支用来发布bug修正，而新的开发发生在不稳定的主干。在Subversion，分支是使用"廉价的复制"实现的。
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。
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.
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。
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。
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。
删除	当你删除了一个版本控制的条目(并且提交这个修改)，这个条目将不会存在于版本库以后的修订。但它还是存在于版本库的以前的修订版本里，如果必要，你可以复制一个删除的条目回来，并且保持所有的历史。
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。
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。
FSFS	FS文件系统，版本库可以使用的一种Subversion专用的后台文件系统格式，能够在通过网络共享的文件系统上使用，FSFS是Subversion 1.2版以后的缺省版本库格式。
GPO	组策略对象
最新版本(HEAD revision)	版本库里文件或目录的最新版本。
导入	在一个修订里将整个目录导入到版本库的Subversion命令。
锁	当一个版本控制条目的被你锁定，就是将它在版本库里标示为不可提交，只有作出锁定的工作副本可以提交。
日志	显示一个文件或是文件夹的版本历史。也就是"历史"。
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。
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。
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。
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。
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。
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。
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。
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。
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。
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。
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。
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。
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。

索引

符号

A

B

W