

# Subversion 使用手册（开发人员）

## 准备工作

- 1、获得个人帐户 告诉管理员你希望得到的用户名以及密码，管理给你添加帐户。
- 2、服务器 IP 地址：  
192.168.0.254
- 3、获得客户端工具：  
从公司文件服务器的Tools 目录获得客户端工具TortoiseSVN 。  
文件服务器的使用可以参照文档《文件服务器使用手册》

## 简单介绍

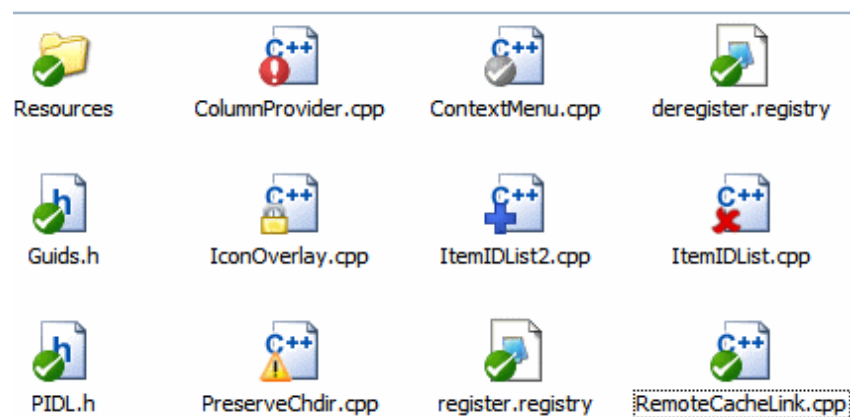
TortoiseSVN是Subversion版本控制系统的一个免费开源客户端，可以超越时间的管理文件和目录。文件保存在中央的Repository，除了能记住文件和目录的每次修改以外，版本库非常像普通的文件服务器。你可以将文件恢复到过去的版本，并且可以通过检查历史 知道数据做了哪些修改，谁做的修改。这就是为什么许多人将Subversion和版本控制系统 看作一种“时间机器”。

## 日常使用指南

### 1. 开始

#### 1.1. 图标重载

图 1.1. 显示重载图标的资源管理器

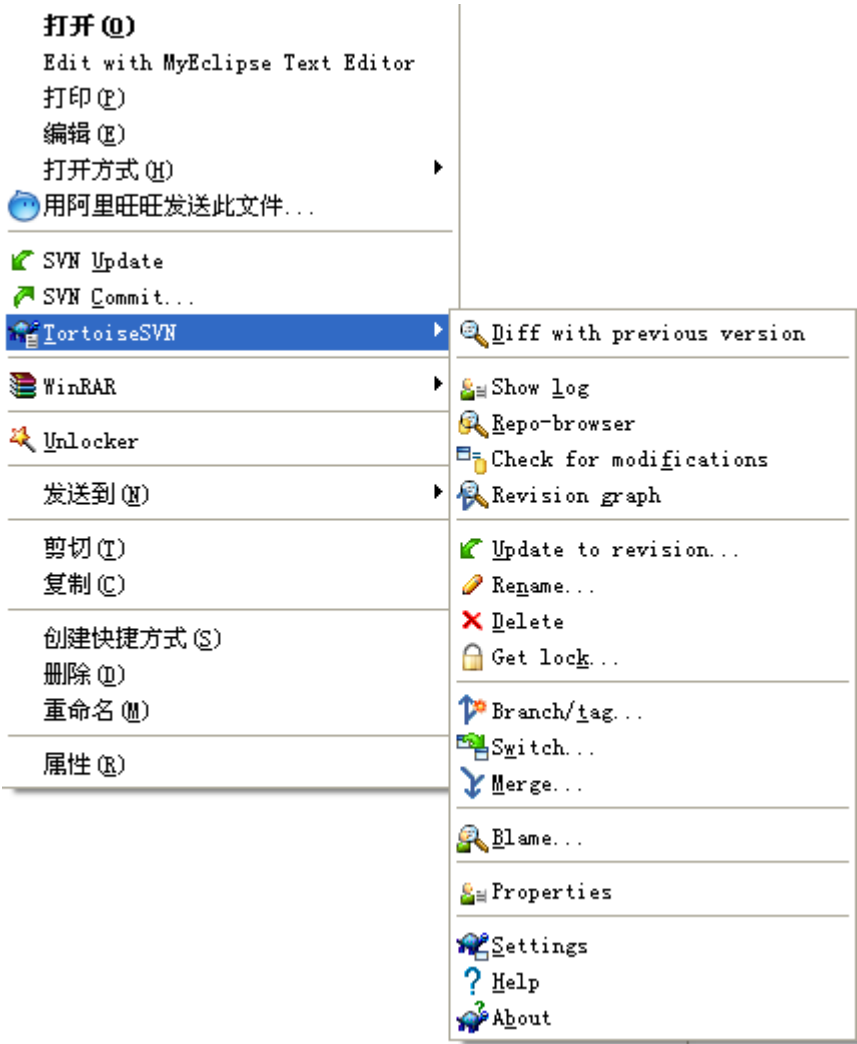


TortoiseSVN 最直观的功能之一就是图标重载，重载的图标显示在你的工作副本文件上。你一眼就可以知道文件被修改过了。后面的部分将详细讲述每个图标所表示的意义。

### 1.1.2. 右键菜单

图 1.2.

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。

图 1.3. 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。

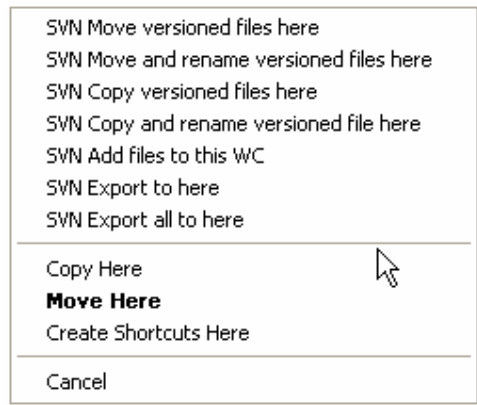


本示例是在一个受控文件夹下的某个未受控的快捷方式， 在资源管理器的文件菜单下有三个 TortoiseSVN 条目。一个是受控文件夹本身的，一个是快捷方式本身的，第三个是快捷方式所指向的对象。为了帮助你区分它们，菜单条目的图标的右下角有标志，表明是文件、快捷方式、文件夹或是选中了多项。

### 1.1.3. 拖放

图 1.4.

版本控制下的一个目录的右键拖拽菜单



在工作拷贝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。

### 1.1.4. 常用快捷方式

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

F1

当然是帮助。

F5

刷新当前视图：这也许是单键命令中唯一一个最常用的：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show log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。

### 1.1.5. 认证 如果要连接的版本库需要口令，一个认证对话框就会显示出来。

图 1.5. 认证对话框



输入你的用户名和口令。那个选择框能让 TortoiseSVN 在 Subversion 的缺省路径下

\$APPDATA\Subversion\auth 的三个子目录里保存认证信息：

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）。

每个文件对应一个要连接的服务器。文件是纯文本格式，因此可以用文本编辑器查看每个文件是应用于哪个服务器的。如果希望 Subversion 和 TortoiseSVN 忘记某个服务器的凭证信息，只需删除这个服务器的对应文件即可。

如果想要清除所有服务器的认证缓存，可以通过 TortoiseSVN 的设置对话框的常规设置页来

实现。那个按钮能够清除 Subversion 的 auth 目录下缓存的所有认证数据。

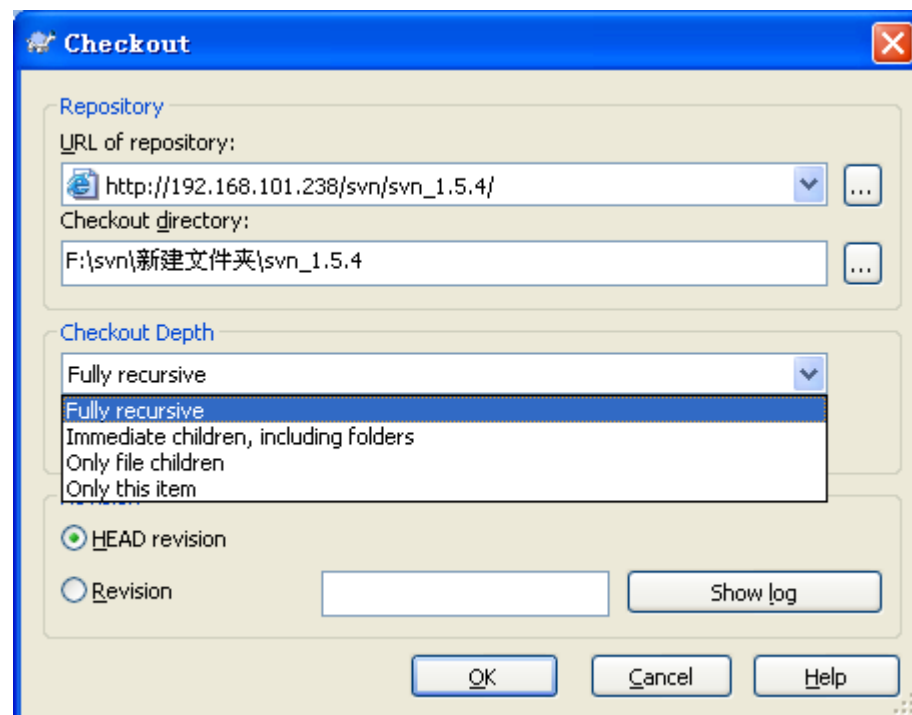
## 1.2. 检出工作拷贝

为了得到一个工作拷贝，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拷贝的目录。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 1.6. 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。



你应该只检出到一个空的目录。如果你要将你的源代码树检出到与你导入它们时相同的目录，Subversion 会给出一个错误信息它不会用已受控的文件覆盖已经存在的但未受控的文件。你必须检出到一个不同的目录或是先将已经存在的源代码树删除。

如果你只希望检出最顶层的文件夹而忽略子文件夹，请选中“only this item”。

如果你选择“immediate children, including folders”，那么你检出的文件夹下的子文件夹就是空的。

如果你选择“Only file children”，那么你检出的文件夹里就没有子文件夹，但是有子文件

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。



如果这两个选项的任何一个选中了，你应该使用 TortoiseSVN → 更新至版本...来更新你的工作副本而不是使用 TortoiseSVN → 更新。标准的更新将会包含所有的子文件夹和 外部引用。

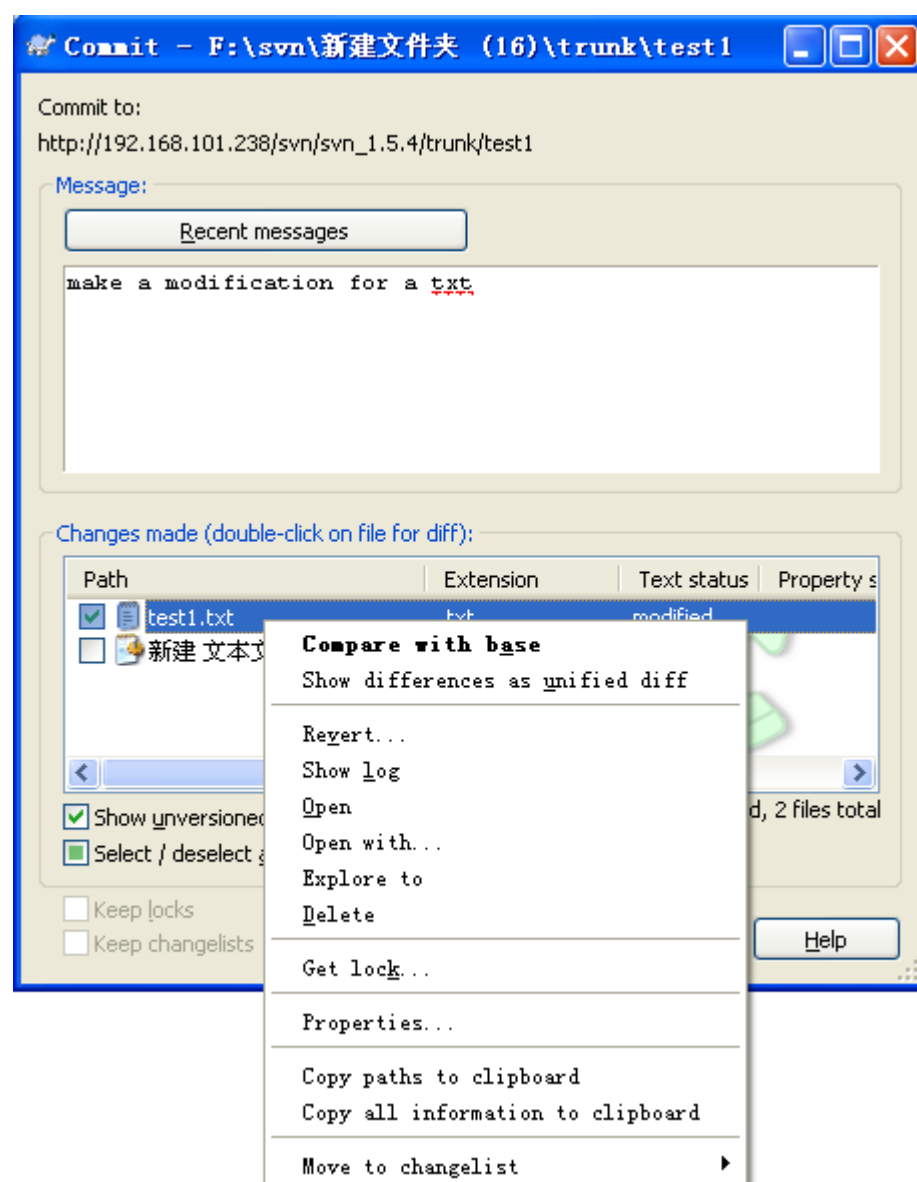
强烈建议你只检出 trunk 的那部分目录树。如果你在 URL 中指定了目录树的父路径，你的硬盘有可能被塞满，因为你将会得到整个版本库树的副本，包括项目所有的分支和标签 (tag)！

## 1.3. 让你的修改进入版本库

将你对工作复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。

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

图 1.7. 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。当你提交文件夹时，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事从而让你的生活更滋润一点。

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



### 在提交对话框中有很多未受控的文件

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

将文件 (或是通配符扩展) 加入到设置页的排除列表中。这对每个工作副本都起作用。

使用 TortoiseSVN → 加入忽略列表，将文件加入 `svn:ignore` 列表。这对你设置了 `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。

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

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柄来改变列的宽度。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。



### 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。



### 修复外部改名

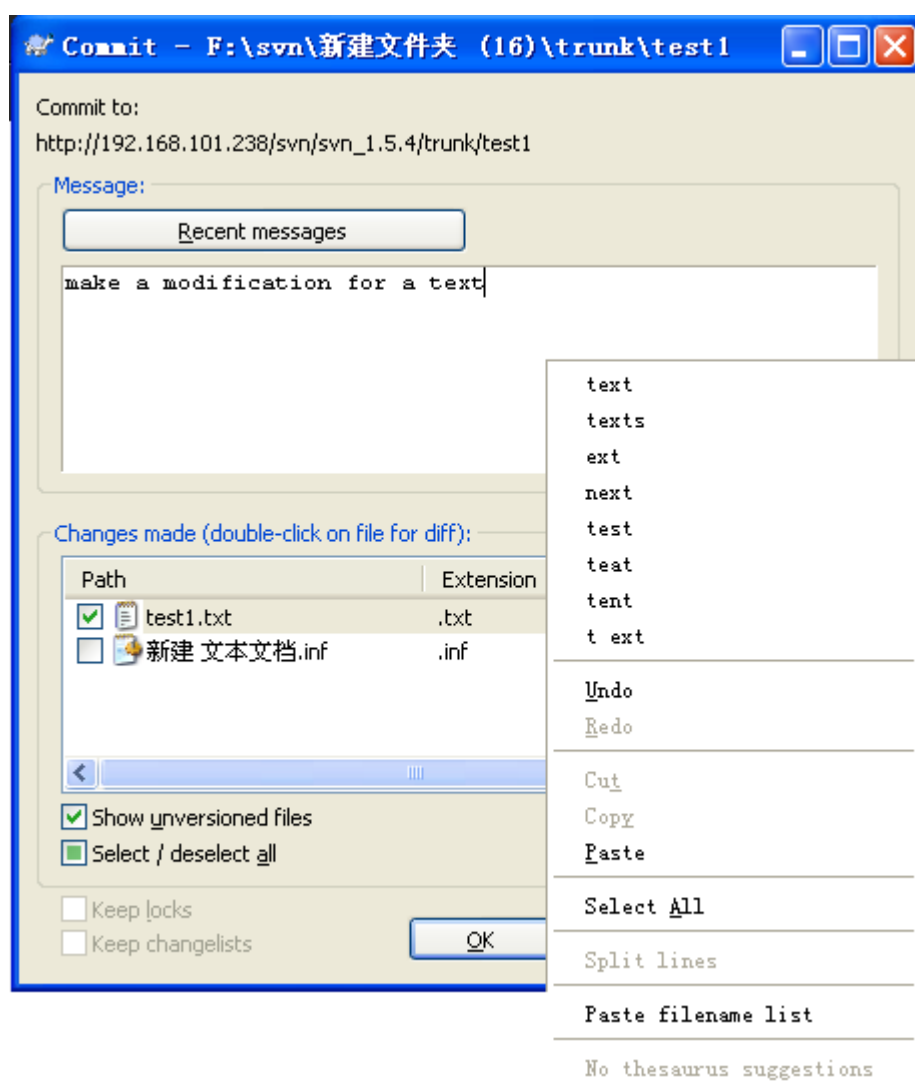


有时候文件不是用 Subversion 改名，于是它们在文件列表中作为丢失和未版本控制的文件出现。为了避免丢失历史，你需要通知 Subversion。简单的选择老名称（丢失）和新名称（未版本控制），然后使用右键菜单 修复移动来指明这两个文件是改名关系。

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

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

图 1.8. 提交对话框的拼写检查器



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

日志信息窗口还包含一个文件名和函数自动完成的功能。这使用了正则表达式来从你提交的(文本)文件中提取类和函数名，当然包括文件名本身。如果你现在敲入的一个词与列表中的

任意一个匹配 (在你输入至少 3 个字符后)，就会出现一个下拉列表，允许你选择完整的名字。与 TortoiseSVN 一起提供的正则表达式位于 TortoiseSVN 安装路径的 bin 文件夹中。你可以 定义自己的正则式并将其存放在%APPDATA%\TortoiseSVN\autolist.txt。当然你自定义的自 动列表不会在升级安装的时候被覆盖。

在按下 OK 之后，会出现一个对话框显示提交的进度。

图 1.9. 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

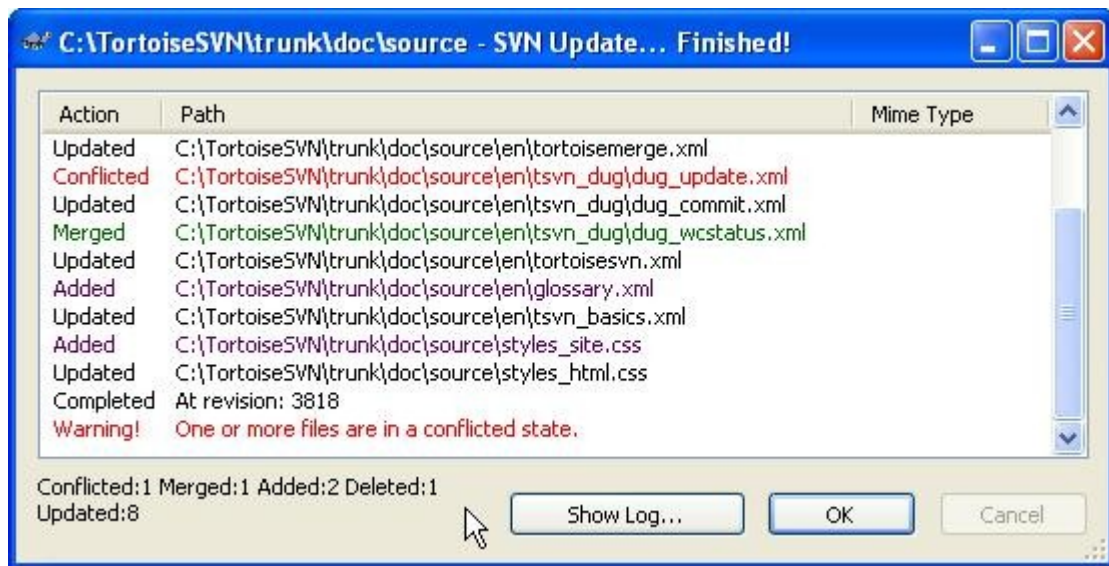
提交一个删除或是替换。

黑色

所有其他项

## 1.4. 用来自别人的修改更新你的工作复本

图 1.10. 已经完成更新的进度对话框



你应该定期地确保别人作的修改与你的工作副本可以整合。从服务器上获取改动到你本地副本的过程称为更新。更新可以针对一个文件、几个选中的文件或是递归整个目录层次。要进行更新操作，请选择要更新文件或路径，右击选择右键菜单中的TortoiseSVN→update。会弹出一个窗口显示更新的进度。别人作的修改将合并到你的文件中，你所做的修改会被保留。版本库受更新操作的影响。

进度对话框使用颜色代码来高亮不同的更新行为

紫色

新项已经增加到你的工作副本中。

深红

你的工作副本中删除了多余项，或是你的工作副本中丢失的项被替换。

绿色

版本库中的修改与你的本地修改成功合并。

亮红

来自版本库的修改在与本地修改合并时出现了冲突，需要你解决。

黑色

你 wc 中的没有改动的项被来自版本库中新版本所更新。

如果你在更新中遇到了冲突 (这是由于别人与你修改了同一个文件的同一行代码，并且两者的修改不匹配)，对话框中将冲突显示为红色，你可以双击这些行启动外部合并工具来解决冲突。

当更新完成后，进度对话框在文件列表下面显示汇总信息，多少项更新，增加，删除，冲突等。汇总信息可以使用 CTRL+C 复制到剪贴板。

标准的更新命令没有选项，仅仅是把你的工作复本更新到版本库中的最新版本，这也是最常用的情况。如果你要对更新过程进行更多的控制，就要使用 TortoiseSVN → 更新到版本...。这个操作允许你更新工作复本到一个指定的版本，不仅仅是最新的。假设你的工作复本是在版本 100，但你要回顾一下版本 50 是什么样的——那你只要简单地更新到版本 50 就可以了。在同一个对话框中你还可以选择不递归更新当前文件夹(就是不更新所有的子文件夹)并且可以选择是否在更新中忽略外部的项目(比如具有属性 svn:externals 的被引用的项目)。



#### 小心

如果你把一个文件或是文件夹更新到某个特定的版本，你不应该对这些文件做修改。你在提交的时候会得到一个已经过期的错误消息!如果你要取消修改，从一个早前的版本重新开始，你可以通过版本show log对话框回滚到之前的版本。

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時候很有用。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复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复本中消失了，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复本，最好是在该文件的show log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/文件夹会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

又是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，却碰到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。

## 1.5. 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  
  
    你的修改  
  
=====
```

来自版本库中的代码

```
>>>>>> 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，在你更新你的工作复本之前存在于你的的工作复本中——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复本之前的基础版本 (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复本时，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。

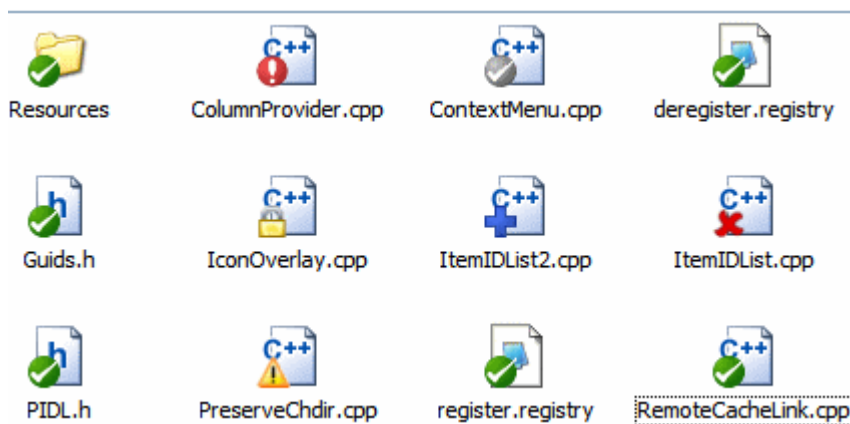
你可以右击父文件夹，选择 TortoiseSVN → 已解决...，使用“已解决”命令来解决多个文件。这个操作会出现一个对话框，列出文件夹下所有有冲突的文件，你可以选择将哪些标记成已解决。

## 1.6. 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。

### 1.6.1. 图标重载

图 1.11. 显示重载图标的资源管理器



现在你已经从 Subversion 版本库中检出了一份工作副本，你可以在资源管理器中看一下这些文件的图标有什么变化。这也正是 TortoiseSVN 这么流行的原因之一。TortoiseSVN 加入

了被称为重载图标的功能重载了原始的文件图标。根据文件的 Subversion 状态的不同，重载的图标也不同。



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



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



如果在提交的过程中出现了冲突图标变成黄色感叹号。



如果你给一个文件设置了 `svn:needs-lock` 属性，Subversion 会让此文件只读，直到你获得 文件锁。只读文件具有这个重载图标来表示你必须在编辑之前先得到一个锁。



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



这个图标表示当前文件夹下的某些文件或文件夹已经被计划从版本控制中删除，或是该文件夹下某个受控的文件丢失了。



加号告诉你有一个文件或是目录已经被计划加入版本控制。

与 TortoiseCVS (一个集成的 CVS shell) 不同，对于未受控的文件没有图标重载。这么做是因为图标重载的数量受到系统的限制，应该要节约使用。

事实上，你会发现并不是所有的图标被使用在你的系统上。这是由于 Windows 限制图标重载 不能超过 15 个。Windows 自己用了 4 个，剩下 11 个可被别的应用程序使用。如果你同时使用了 TortoiseCVS，就不有足够的空位了，所以 TortoiseSVN 希望成为一个“良好市民(TM)”，限制自身的使用，为别的应用留下机会。

正常，已修改和冲突总是被载入，并可见。已删除只要有可能的就载入，但如果没有足够的空位，就使用已修改来代替。只读只要有可能就载入，但如果没有足够的空位就使用正常来代替。已锁定只在少于 13 个重载已经载入的情况下才加载，如果不满足这个条件就使用正常来代替。

已增加只在少于 14 个重载已经载入的情况下才加载，如果不满足这个条件就使用已修改来代替

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

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

右键点击列头，从出现的右键菜单中选择其他...。出现一个对话框，你可以指定在“详细信息视图”中要显示的列及其顺序。滚动对话框中的条目直到 SVN 开头的条目出现。在你想要显示的条目上打勾，然后点击 OK 按钮关闭对话框。你选择的列就会出现在当前显示的列的 右边。你可以通过拖放它们来达到重新排序或是修改列宽度的目的。



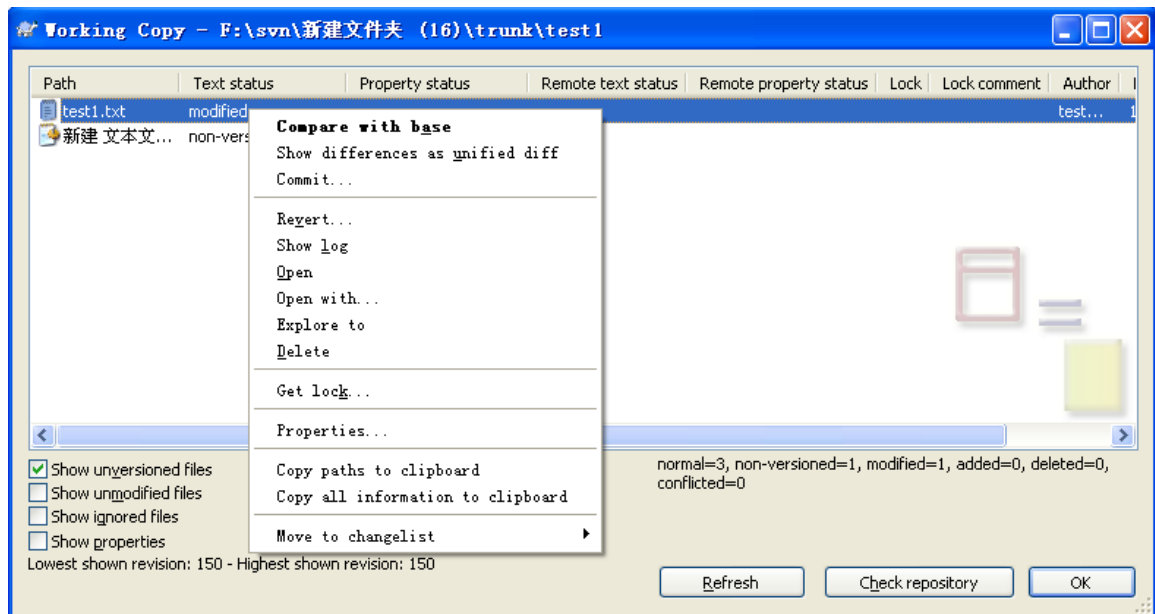


提示

如果你想要当前的布局对你所有的工作副本都有效，你可以考虑把他设成默认视图

### 1.6.3. 本地与远程状态

图 1.12. 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别人修改并提交是很有用的。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。

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。

你还可以对单个文件进行还原 (revert)。如果你不小心删除了一个文件，在对话框中会显示为丢失你可以使用还原来恢复它。



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。如果你向彻底删除(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。



#### 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。

### 1.6.4. 查看差别

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



#### 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。

### 1.7. 版本show log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 show log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）。

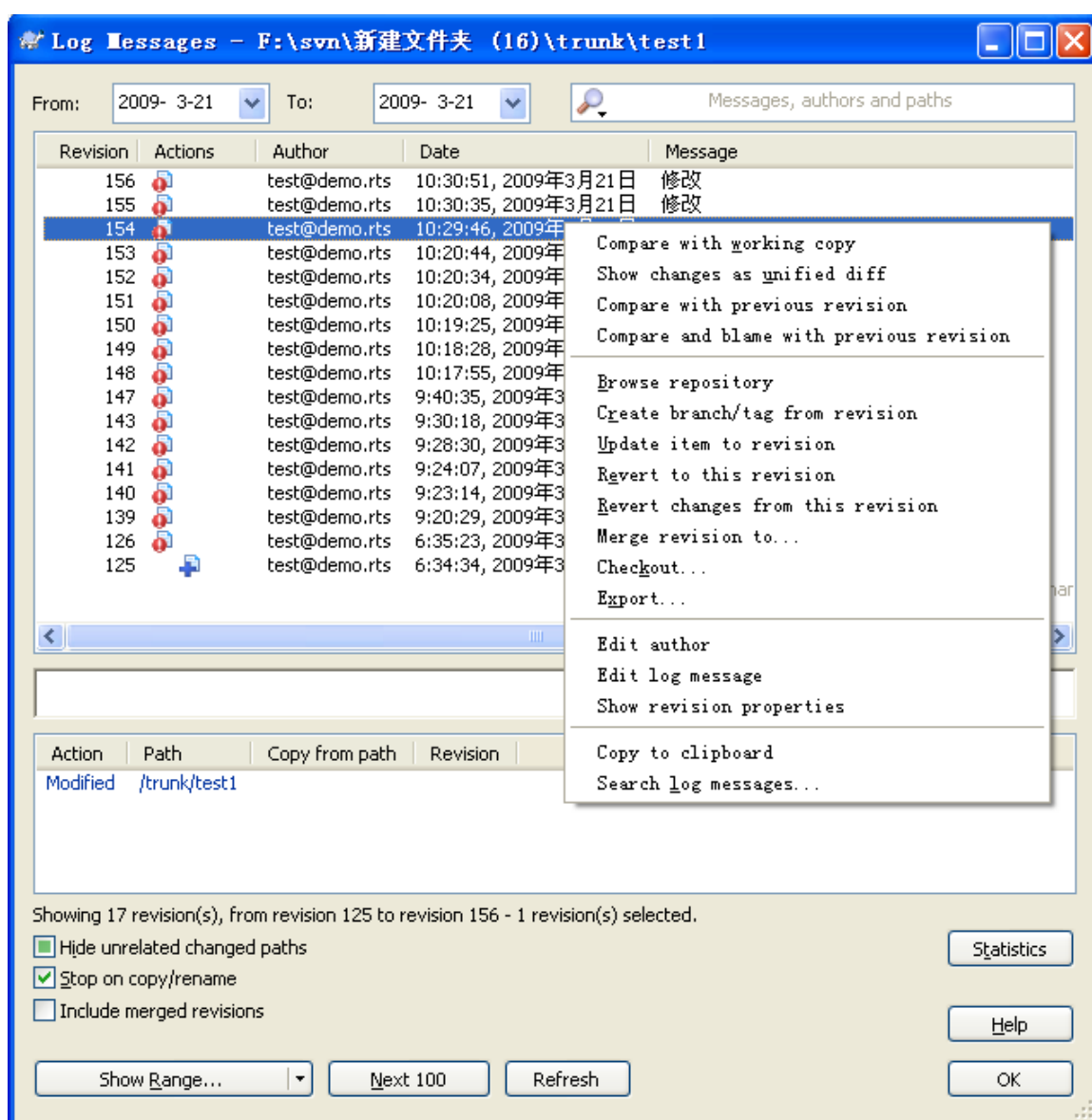
中间的面板显示了被选中的版本的完整的日志信息。

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。

## 17.1. 调用版本show log对话框

图 1.13. 版本show log对话框



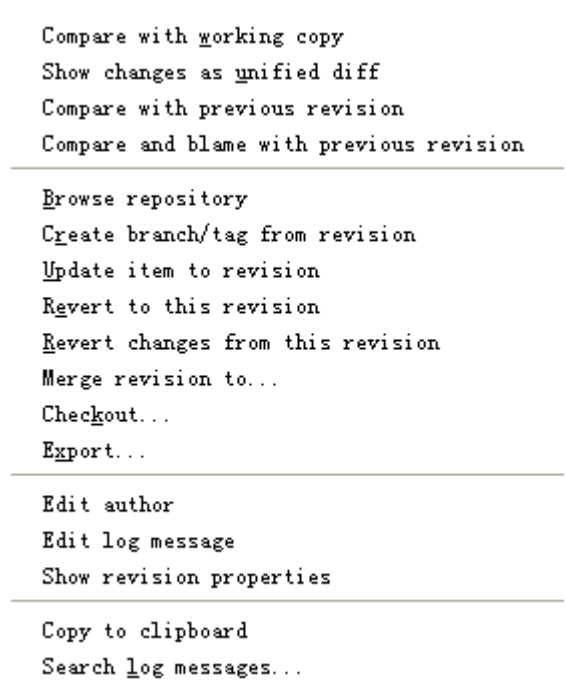
有几种途径可以调出show log对话框：

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，show log对话框只显示你上一次更新以来的版本变化。

## 1.7.2. 获得更多信息

图 1.14. 版本show log对话框的顶部面板的右键菜单



show log对话框的顶底面板右键菜单功能如下：

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show log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。 评审选中的版本，和你的工作基础文件，使用可视化差异工具显示结果。

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，但它所有的变化显示在一个格式更为紧凑的文件中。 将选中的版本保存成文件，这样你就获得了该文件的一个旧的版本。这个选项只在 你查看一个文件的日志时才可用，它只保存那一个文件的一个版本。 打开版本库浏览器，可以查看选中的目录。这个选项只在你查看一个目录的日志时 可用。

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。 将你的工作复本更新到选中的版本。如果你想要你的工作复本折返到过去的某个时 间，那这个功能就很好用。你最好是更新工作复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，你的工作复本就可能不一致，从而导致你无法提交任何 修改。

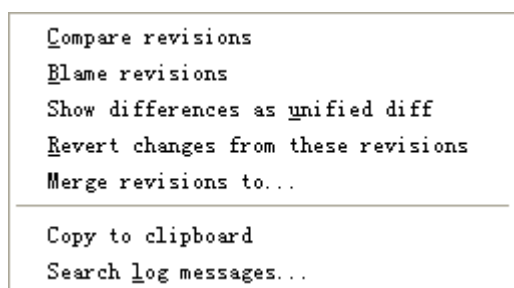
还原选中版本中所做的变更。还原的内容只在你的工作复本中，这个操作完全不会影响版本库！要注意的是，这个操作仅仅恢复该版本中的修改 (译注：就是将你选中

的那个版本中的修改还原，而在那之后的修改，包括你在工作复本中的新改动依然会被保留。不是将整个文件替换成选中的那个版本。(译注：本段对菜单中的“复原 自此版本以来的变更(Revert changes from this revision)”，按实际操作的结果 来看，中文菜单应翻译成“复原此版本的变更”比较合适。

还原到某个早前的版本。如果你进行了多个修改，然后决定要返回到版本 N 中的模 样，你就可以使用这个命令。当然，返回的东西都只在你的工作复本中，在你提交 之前，并不会影响版本库。注意，这将会还原从那个版本以来的所有变更，使用选 中的版本来替换文件/文件夹。(译注：在实际使用中发现并不是简单的替 换，如果 你在当前工作复本中所做的修改不涉及到被选中版本以来的变更内容， 那么你的修 改会与版本 N 进行合并，否则这个操作会产生一个冲突。这段对应菜 单中的“复原 到此版本(Revert to this revision)”。

编辑之前提交时的日志信息或是作者。 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 1.15. 选择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

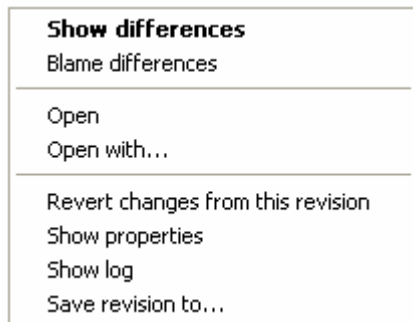
较工作比较两个选中的版本。默认的比较工作是与 TortoiseSVN 一起提供的 TortoiseMerge。

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。

评审两个版本，并使用差异比较工作比较结果。 使用单一差异文件显示差异。这对文件和文件夹都有效。 如前所述可修改日志消息或作者 如前所述可以搜索日志消息。

如果你用 **Ctrl** 或 **Shift** 组合键选择了多个连续的版本，右键菜单将有一个选项，可以让你还原这些版本中的修改。这是一次性还原多个版本中修改的最简方法。

图 1.16. show log对话框的底部面板的右键菜单



show log对话框的底部面板也有右键菜单，你可以：

显示选中版本中的选中文件的差异。这个操作只对显示为已修改的文件有效。评审选中文件的选中版本与前一个版本，使用可视化差异工具显示差异。详情请参阅第 5.20.2 节 “追溯不同点” 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。

### 1.7.3. 获取更多的日志信息

show log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长的时间。

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show log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。

## 1.7.4. 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 允许你在任何时候修改日志消息和作者。但这种改变不可还原 (不在版本控制之列)，正因如此，这些功能默认是不可用的，如果要开启它，必须设置一个 `pre-revprop-change` 钩子。一旦你按需要为服务器设置了钩子，你就可以使用 `show log` 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`）提交后的信息属性将永久覆盖该属性之前的值。

## 1.7.5. 过滤日志信息

如果你只想要显示上千条日志中所感兴趣的日志，你可以使用 `show log` 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。

要注意的是，这些过滤器只对已经获取的信息有效。它们并不从版本库中下载信息。你还

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

## 1.7.6. 统计信息

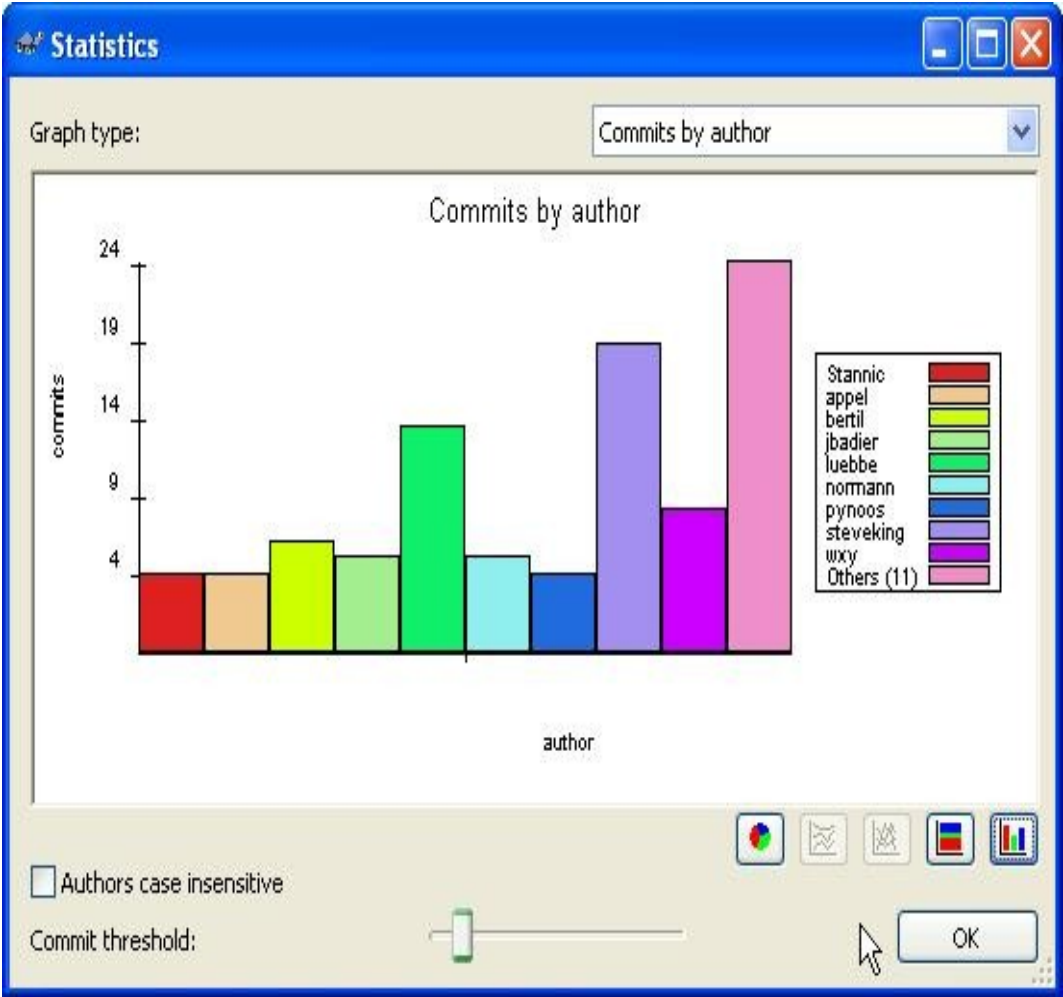
统计按钮，可以显示一些你感兴趣的关于 `show log` 对话框中版本的信息。可以显示已经有几个作者做了工作，他们各提交了几次，按周的统计，等等。现在，你可以发现一个大概情况：谁最勤快，谁偷懒。

### 1.7.6.1. 统计页

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

### 1.7.6.2. 作者提交次数统计页

图 1.17. 作者提交次数统计柱状图



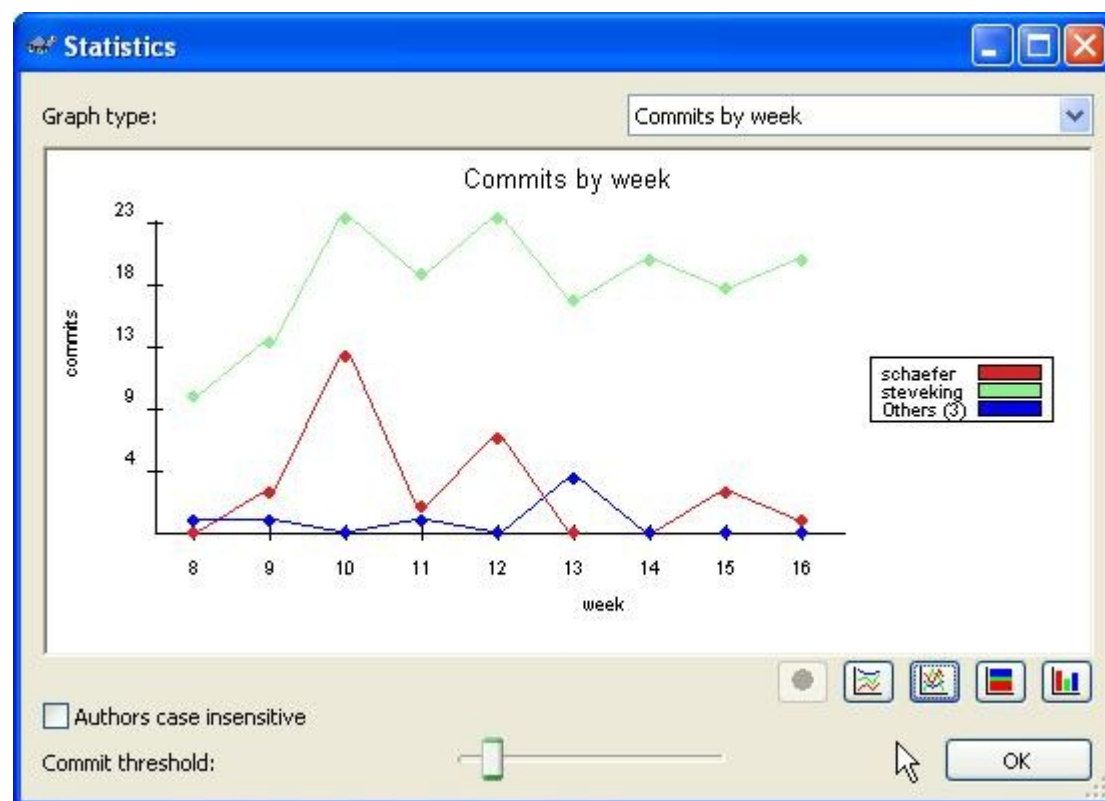
此图用简单柱状图、叠加柱状图或饼图显示了哪些作者已经在项目中活跃了。

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



### 1.7.6.3. 按周提交次数统计页

图 1.18. 按周提交次数统计



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

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

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

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

## 1.9. 查看差异

在项目开发中，有一个很常用的要求就是查看更改。可能是你要求查看同一文件的两个版本之间的差异，或者是查看两个独立的文件的差异。TortoiseSVN 自带了一个工具叫



TortoiseMerge 用来查看文本文件的差异。也有一个叫 TortoiseIDiff 的工具来比较图像文件的差异。当然，你可以根据你自己的喜好来选择比较差异的工具。

## 1.9.1. 文件差异

### 本地更改

如果你想看到你的本地副本有哪些更改，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。

### 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拷贝之间的差异，使用显示 show log 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拷贝比较差异

### 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 show log 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。

### 提交所有修改

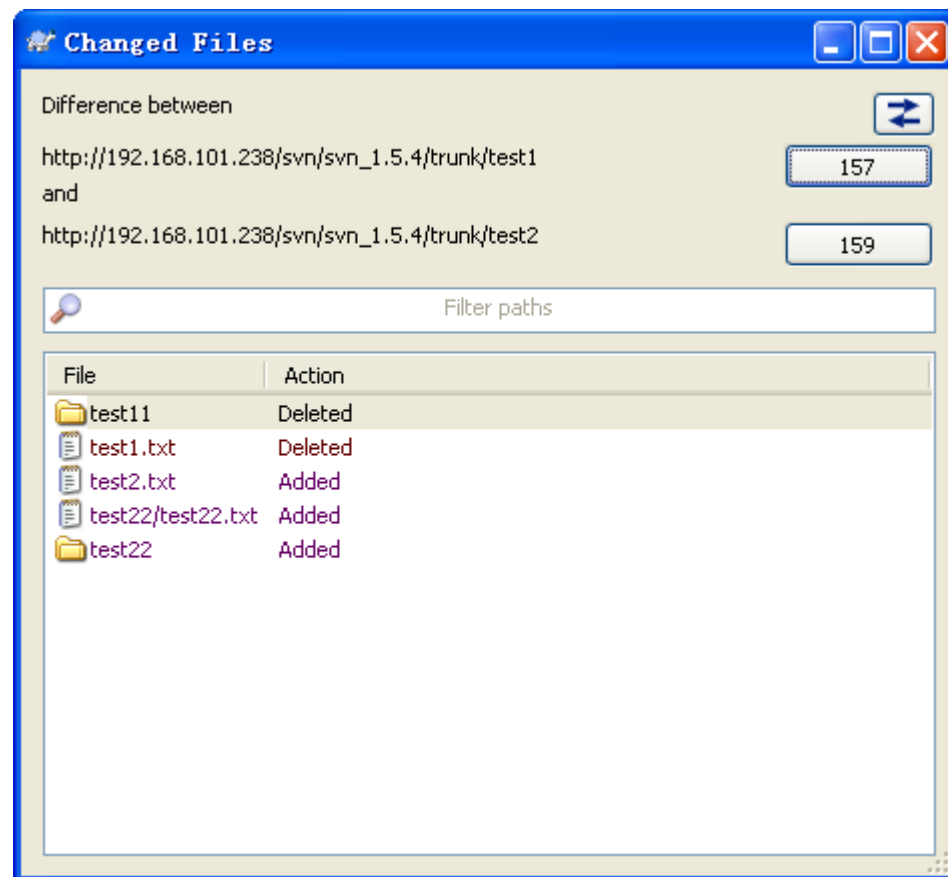
如果你要在一个视窗中查看某一版本的所有更改，你可以使用统一显示所有比较 (GNU 片段整理)。它将显示所有修改中的部分内容。它很难显示一个全面清晰的比较，但是会将所有更改都集中显示出来。在版本 show log 对话框中选择某一版本，然后 在右键菜单中选择统一显示所有比较。

### 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 → Diff。

## 1.9.2. 比较文件夹

图 1.20. 修订版本版本比较对话框



当你在版本库浏览器中选择了一个树，并右键 `mark for comparison`，然后选择另一个树，右键 `show differences as unified Diff`，可打开上面的窗口。

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。

如果你需要导出文件列表和动作 (修改，增加，删除)，你可以使用快捷键 Ctrl-A 选择所有项，用 Ctrl-C 将详细列表复制到剪贴板。

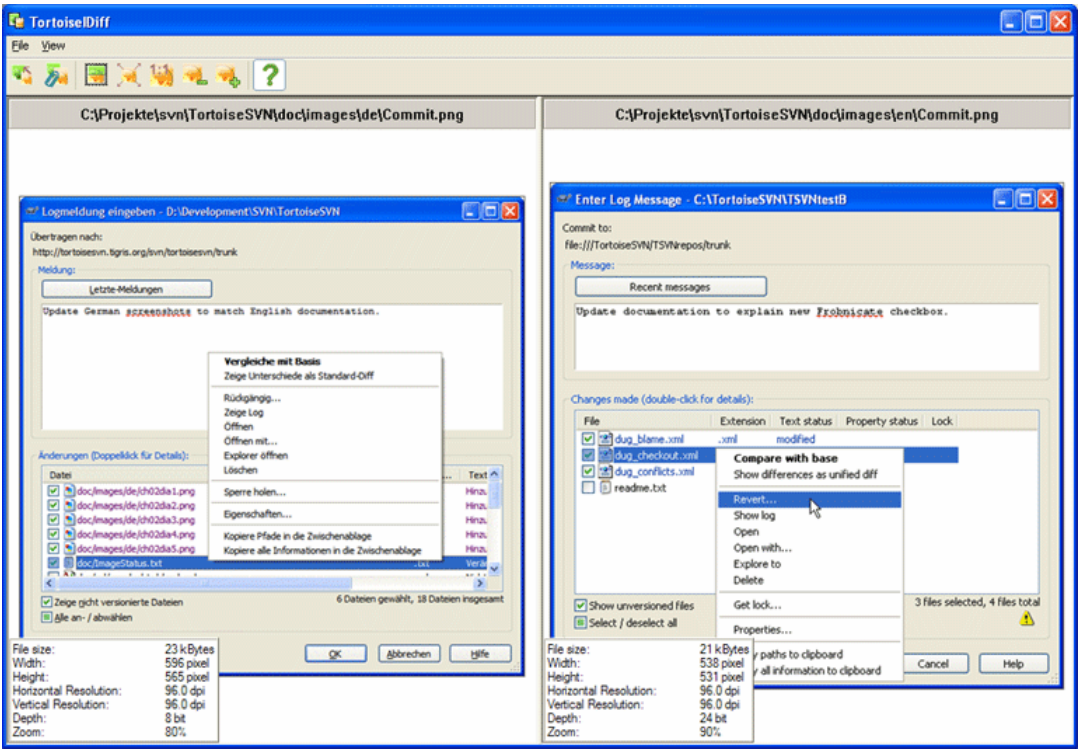
顶部的按钮允许你改变比较的方向。你可以显示从 A 到 B 的修改，或者如果你喜欢，显示从 B 到 A 的修改。

两个标有数字的按钮允许你改变当前路径版本库的版本号，只要你修改的版本号存在并且有效。

### 1.8.3. 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，但是我們也需要查看图像文件的更改。这就是我们设计 TortoiseIDiff 的原因。

图 1.20. 差异察看器截图



TortoiseSVN → 比较差异 TortoiseIDiff 可以显示同种格式的图像差异。一般情况下是左右对称地显示两个图像，但你也可以通过调整视图滑动条转变为上下显示的模式，如果你愿意，这里支持使用透明框进行图像覆盖的方式，在顶端的滑动条可以调整图像之间的吻合程度 (alpha blend)，你也可以使用 Ctrl-Shift-Wheel 来调整这种程度。

当然你也可以围绕在图像周围上上下下灵活变动地比较。如果你选择联合图像项，则被选择 (滑动条，鼠标) 的两个图像就被关联起来。

在图像信息框中显示了图像的基本信息，比如像素的大小，颜色的深度。如果觉得这个框碍眼可以选择视图 → 图像信息来隐藏它

## 1.8.4. 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 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](#) 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](#) 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](#) 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](#)。

Araxis Merge

Araxis Merge 是一款能对文件和文件夹进行比较和合并的商业软件。它从三条比较路径进行合并，而且在你修改的同时进行及时有效的链接。可以从这里下载 [Araxis](#)。

SciTE

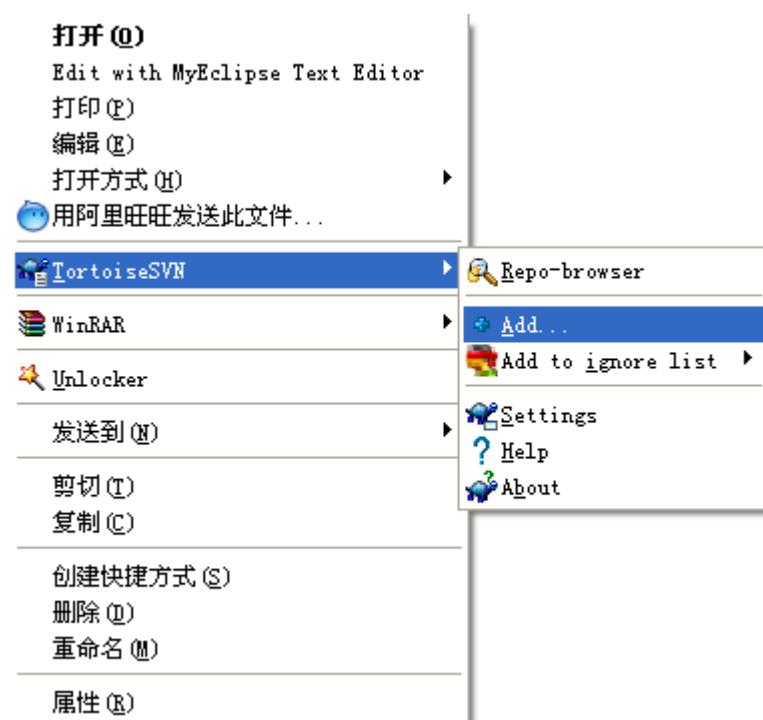
这款文本编译器在统一比较时提供语法显示，读起来更加容易。可以从这里下载 [Scintilla](#)。

Notepad2

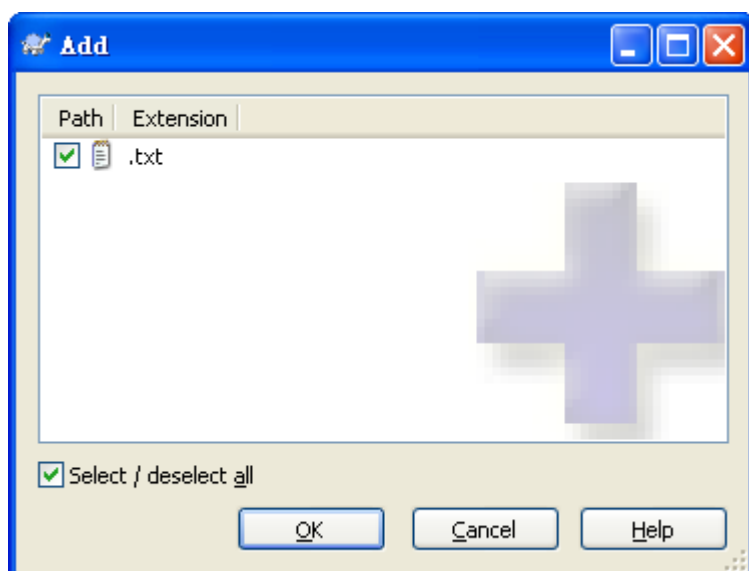
Notepad2 的设计旨在替代 Windows 自带的记事本的功能，它以开源编译控制为基础。在查看统一比较时，它能实现比 Windows 自带的记事本更多功能。免费下载 [here](#)。

## 1.9. 添加新文件和目录

图 5.21. 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 影响版本库



更多

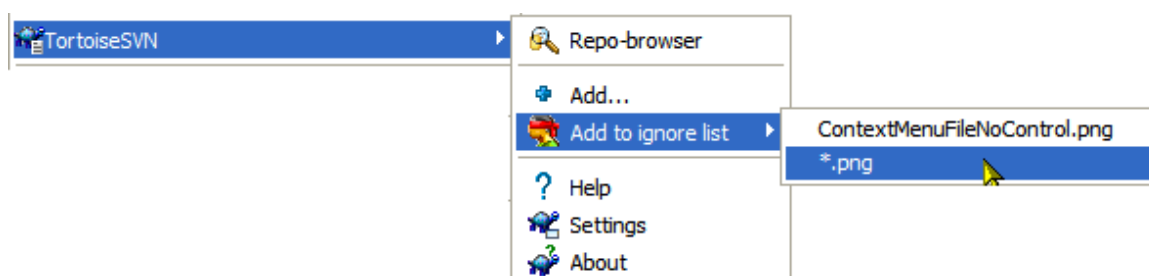
你也可以在已经版本化的目录上使用Add命令。那样的话，添加对话框会显示该版本化目录下所有未版本化的文件。如果你有许多新文件需要一起添加的话，这是很有帮助的。 你可以使用鼠标拖拽的方式从你的工作副本外部

添加进文件。

1. 选择你要添加的文件
2. 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。

## 1.10. 忽略文件和目录

图 1.22. 未受版本控制的文件之资源管理器上下文菜单



5. 在多数项目中你总会有文件和目录不需要进行版本控制。这可能包括一些由编译器生成的文件, \*.obj, \*.lst。或许是一个外部的用于存放可执行程序的目录。只要你提交变更, TSVN 就会在提交对话框的文件列表中列表显示出你的未版本化文件。当然你可以关闭这个显示, 不过你可能会忘记添加新的版本文件。
6. 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, 而真正的未版本化文件则仍然列出。
7. 如果你右键一个单独的未版本化文件, 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。
8. 如果你想从忽略列表中移除一个或多个条目, right click 右键该条目并选择 TortoiseSVN → Remove from Ignore List 你也可以直接通过目录的 svn:ignore 特性。他允许你指定多个文件名段的通用样式, 来部分的描述。阅



### 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。



### 忽略已版本化的条目

已版本化的文件或目录不能够忽略, 这是Subversion的一个特性。

## 1.10.1. 忽略(Ignore)列表中的文件簇

Subversion 的忽略模式使用了文件簇, 一种起初在 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:

星号

匹配任何字符串, 包括空串 (没有字符)

问号(?)

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 执行这样的簇, 那么定界符一直为/, 而不是 windows 下的反斜线。

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。

如果当前路径的目录名作为模式出现在匹配中，这个模式 `Fred.*` 将匹配 `Fred.c` 但不匹配 `subdir/Fred.c`。这对于你添加了一个包含许多文件而又想忽略的目录来说是非常有意义的，因为这个目录名的优先级高于这些文件名。

你应该指定一个 `*cvs` 或者更好的 `cvs */cvs` 模式中的任意一个来忽略所有的 `cvs` 目录。执行第一个选择也将会排斥一些像 `ThisIsNotCvs` 这样的命令。而单独使用 `*/cvs` 又不能作用在一个紧跟着 `cvs` 的子目录上，而且单独的 `cvs` 不能作用在子目录上。

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](#)。

## 1.11. 删除、重命名和移动

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 1.23. 版本控制文件的菜单浏览





如果你通过 TSVN 删除了一个文件/目录，那么这个文件被移出你的工作副本并标记为删除。该文件的父目录会覆盖上一个“删除”标记。你随时可以通过在父目录调用 TortoiseSVN → Revert 命令来恢复该文件。

如果你想在工作副本中移动文件，那么可以这样使用鼠标拖拽：

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。



### 不要 SVN Move Externals

你不应该用TortoiseSVN的移动或重命名作用在一个已经用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。

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。



### 提交父目录

既然重命名和移动都可以像添加之后又删除一样被执行，你必须提交该重命名/移动文件的父目录，所以重命名/移动的删除部分将出现在提交对话框中。如果你不提交重命名/移动的已删除部分，他将保留在仓库中并且你的同组人将更新该未移除的旧文件。例如，他们将有二个一老一新的副本。

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。



### 找回已删除的文件或目录

如果你删除了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show log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。

### 1.11.1. 仅在单一实例中重命名文件

万一在你的版本库中有两个名字相同但大小拼写不同（例如，TEST.TXT和test.txt）的文件，你是不能更新或者检出该包含该文件的目录的。

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

这里（至少）有两种可能的解决方案来重命名文件而不丢失他的日志记录。在Subversion里重命名它是很重要的。仅在浏览器中重命名将会损坏你的工作副本。

解决方案A（推荐）

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

解决方案B

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



#### 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。

### 1.11.2. 修复文件改名

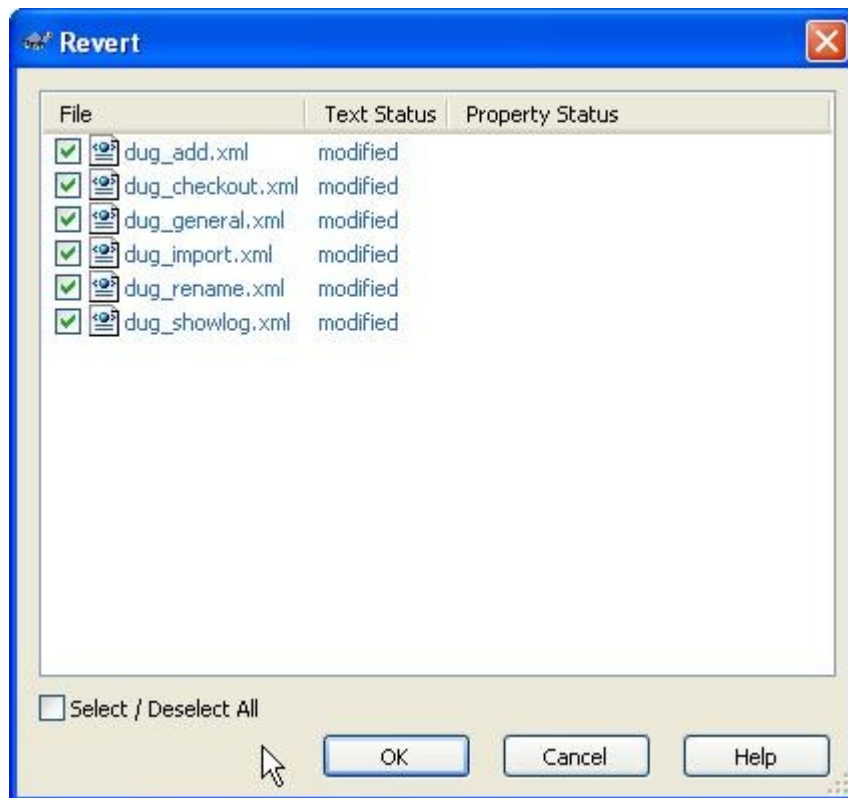
有时候你的 IDE 会因为执行反射操作，改名文件，当然它不能告诉 Subversion。如果你尝试提交修改，Subversion 会发现丢失了老文件，新增了未版本控制的新文件。你可以简单的增加新文件，但是你将丢失历史记录，因为 Subversion 不知道这些文件的关系。

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。

## 1.12. 撤消更改

如果你想要撤销一个文件自上次更新后的所有变更，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 1.24. 恢复对话框



。

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。



### 取消已经提交的改变

Revert 仅能撤销你本地的变更，踏步能撤销已经提交的变更。如果你想撤销所有的包括已经提交到一个特定版本的变更，

## 1.13. 清除

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 → Cleanup 命令。在工作副本的全局使用它是一个好主意。

Cleanup 有另外的一个有用的副作用。如果一个文件日期变化了但是它的内容没变，Subversion 除了采用 byte-by-byte 将该文件和原副本进行对照，不能分清它是否真的变更。如果你有很多这种状态下的文件，将会使获得状态非常慢，还会导致许多会话响应变慢。在你的工作副本上执行一个 Cleanup （清除）命令将会修正这些“坏掉的”时间戳并全速核对他们的状态。

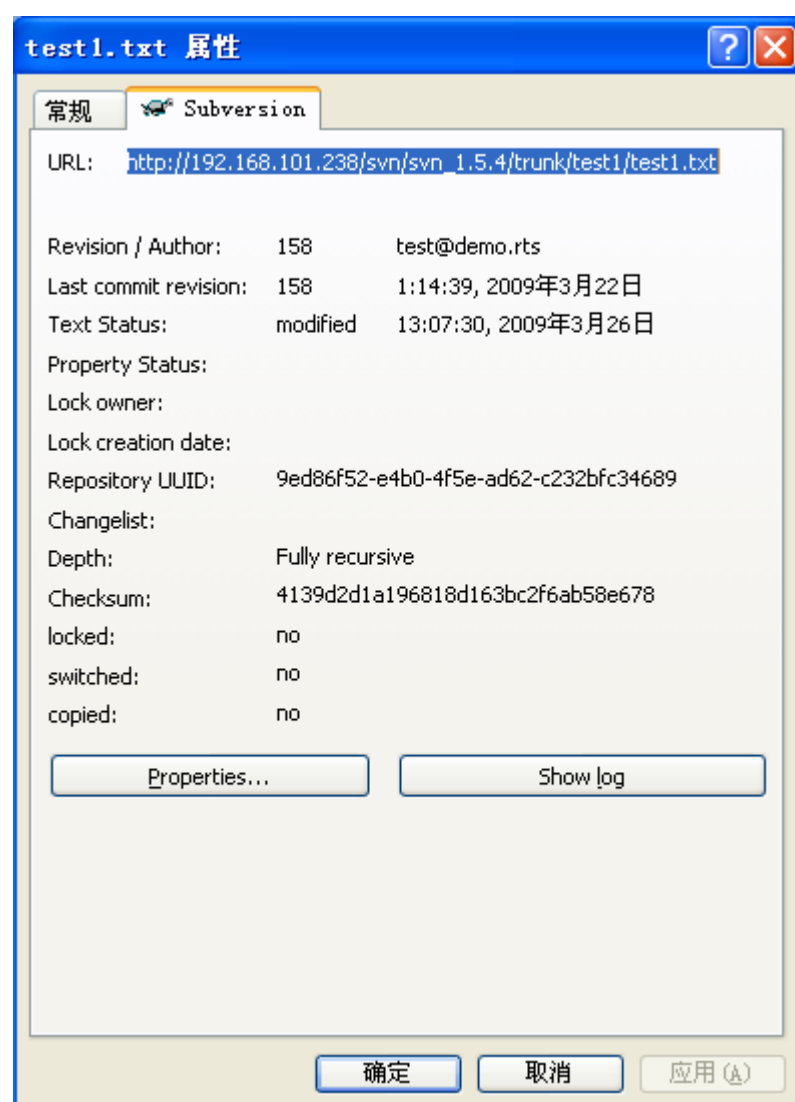


### 提交时间戳

Subversion的一些早期发布中存在一个bug，当你使用Use commit timestamps检查选项check out（检出）的时候会造成时间戳混乱。使用命令Cleanup加速这些工作副本更新。

## 1.14. 项目设置

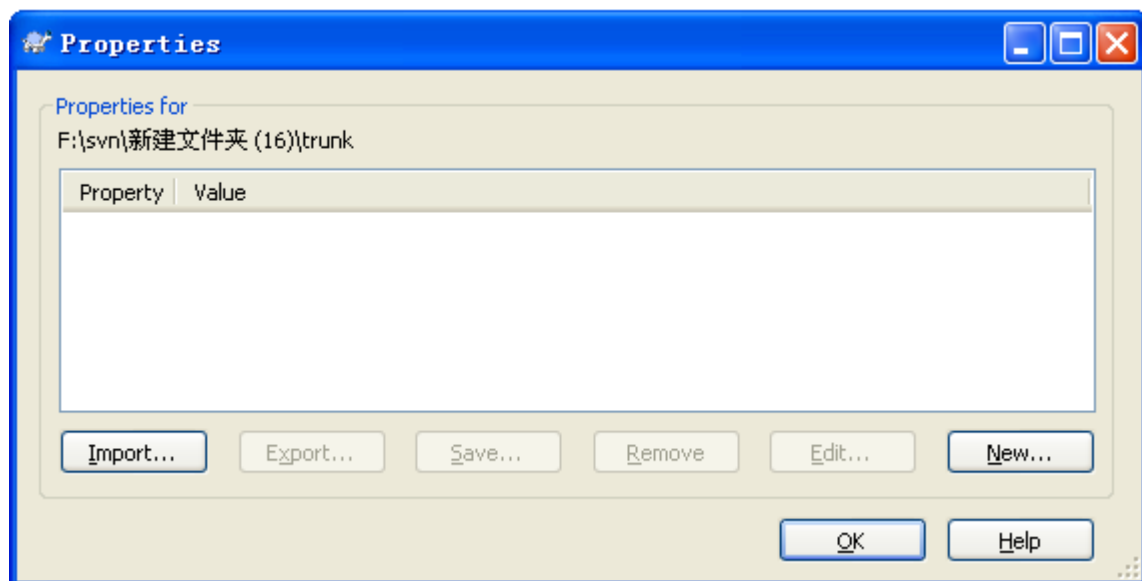
图 1.25. 资源管理器属性页，Subversion 页面



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Window 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。

### 1.14.1. Subversion 属性

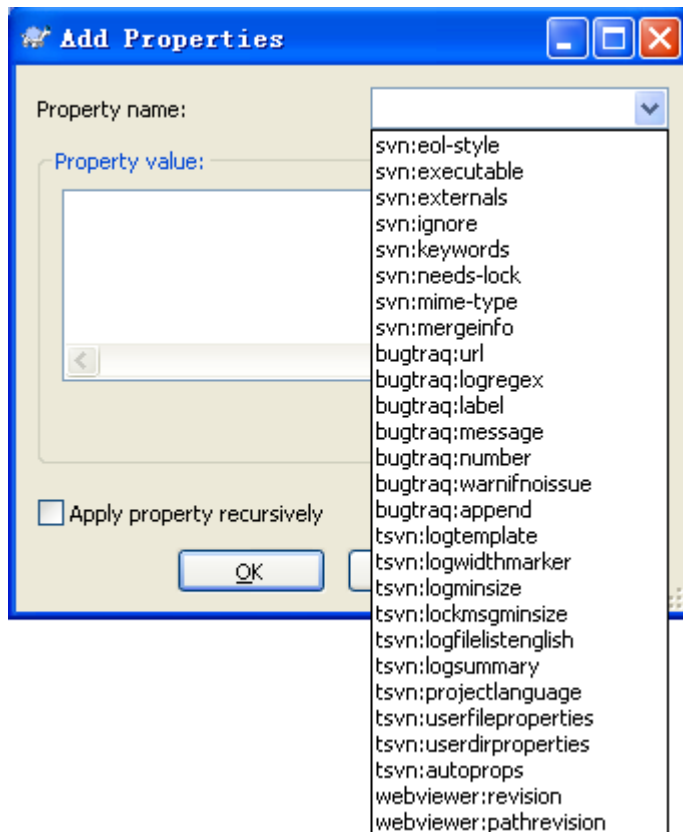
图 1.26. Subversion 属性页



你可以在 Windows 属性对话框读写Subversion 属性；也可以从 TortoiseSVN → 属性，或者 上下文菜单 → 属性，来读写 Subversion 属性。

从版本库里删除数据的唯一方法就是使用 `svnadmin` 这个 Subversion 命令行工具。

图 1.27. 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。单击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。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。

## 1.14.2. TortoiseSVN 属性

TortoiseSVN 有自己专用的几个属性，它们都有 tsvn:前缀。

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

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

tsvn:logwidthmarker 用在要求日志信息被格式化为在最大宽度(典型是 80 字符)处换行非常有用。设置此属性为大于 0 的值会在日志消息对话框中做两件事：放置一个标记指示最大宽度，和禁止自动换行，于是你可以看到输入的信息是否太长。注意：这个特性仅在你选择的消息使用固定宽度字体时才能正确工作。tsvn:logtemplate 在需要定义日志消息格式化规则的工程中使用。在你开始提交时，这个属性的多行消息会被插入日志消息编辑框。你可以编辑它以便包含需要的信息。注意：如果你使用了 tsvn:logminsize 属性，请确认这个长度大于模板的长度，不然就会失去其保护作用。在提交对话框，你可以复制修改的文件列表，包含每个文件的状态(增加，修改等)。tsvn:logfilelistenglish 定义了文件状态用英文插入，还是用本地消息插入。此属性的默认值是真。

TortoiseSVN 可以使用 OpenOffice 和 Mozilla 使用的拼写检查模块。如果你安装了这些模块，那么这个属性将检测使用哪个拼写检查模块。也就是，你的工程的日志信息用的语言。tsvn:projectlanguage 设置拼写检查引擎应该使用什么语言模块来检查日志信息。你可以在这个页面找到你的语言的取值：MSDN：语言标示符。

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。

一些 tsvn: 属性需要 true/false 值。它也理解 yes 是 true 的同义词，no 是 false 的同义词。



## 设置文件夹的 `tsvn:` 属性

属性 `tsvn:` 只能在文件夹设置。当你提交文件或文件夹时，这些属性从文件夹读取。如果没有发现这些属性，TortoiseSVN 会向上级目录搜索，直到未版本控制的文件夹，或到达根目录(例如 `C:\`)。如果你确认每个用户都是从同一个目录检出，例如 `trunk/`，而不是其它子目录，那么只在 `trunk/` 设置属性就足够了。如果你不能确定，那么应当递归的设置每个子目录。深层的设置覆盖高层的设置（靠近 `trunk/`）。

对于 `tsvn:` 属性，你只能对于所有子目录使用递归检查框设置属性，不能设置文件的属性。

TortoiseSVN 可以与一些问题跟踪工具集成。它使用 `bugtraq:` 开始的属性。

它也与一些基于 WEB 的版本库浏览器集成。

## 1.15. 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。

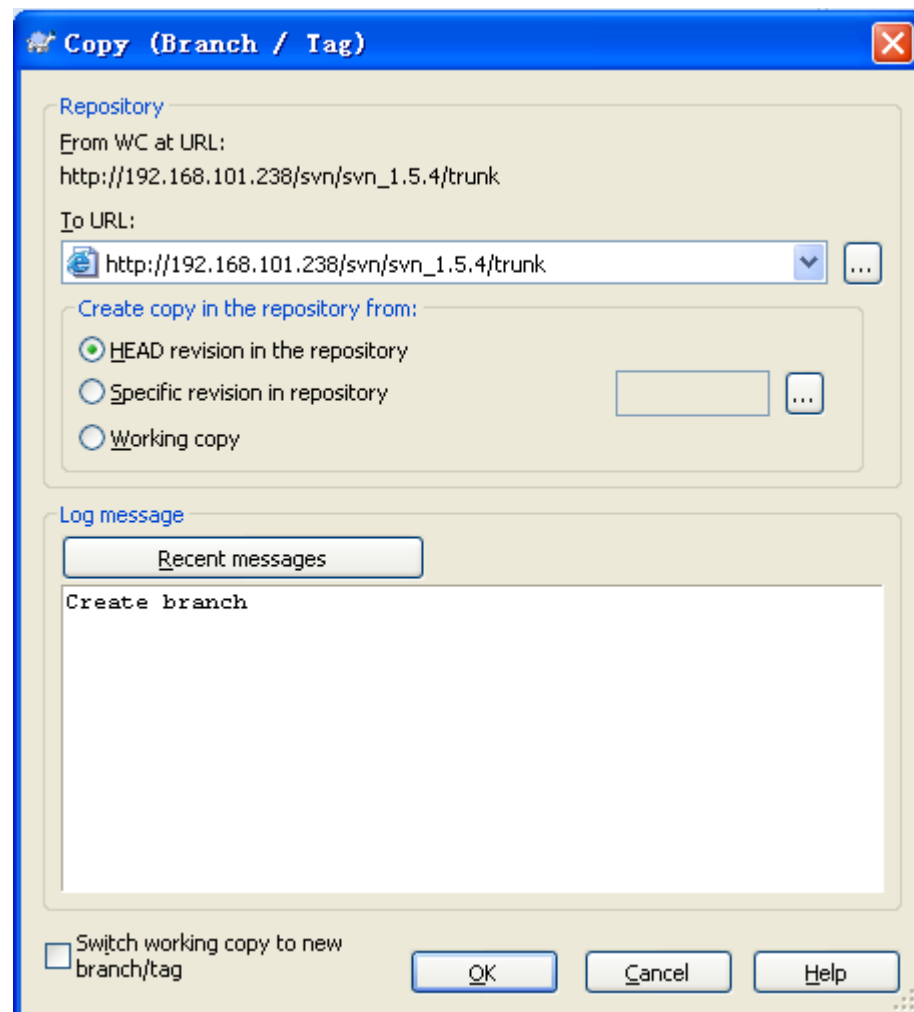
Subversion 没有用于建立分支和标记的特殊命令，但是使用所谓的便宜拷贝来代替。便宜拷贝类似于 Unix 里的硬连接，它意思是代替一个版本库里的完整的拷贝，创建一个内部的连接，指向一个具体的版本树。结果分支和标记就迅速被创建，并且没有在版本库里占据任何额外的空间。



### 1.15.1. 创建一个分支或标记

如果你用推荐的目录结构导入了一个工程，那么创建分支或标记就非常简单：

图 1.28. 分支/标记对话框



在你当前的工作拷贝中给你你想要拷贝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 将会是你当前工作拷贝所处的源 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。现在你必须选择要拷贝的源位置。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拷贝出来。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 在仓库中直接拷贝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拷贝。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拷贝中。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，使用转换工作拷贝至新分支/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。即使你拷贝了你的工作副本，这些修改也会提交到新分支里，而不是到主干里，所以你的工作拷贝可能仍然标记为已修改状态。

## 1.15.2. 检出或者切换

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。

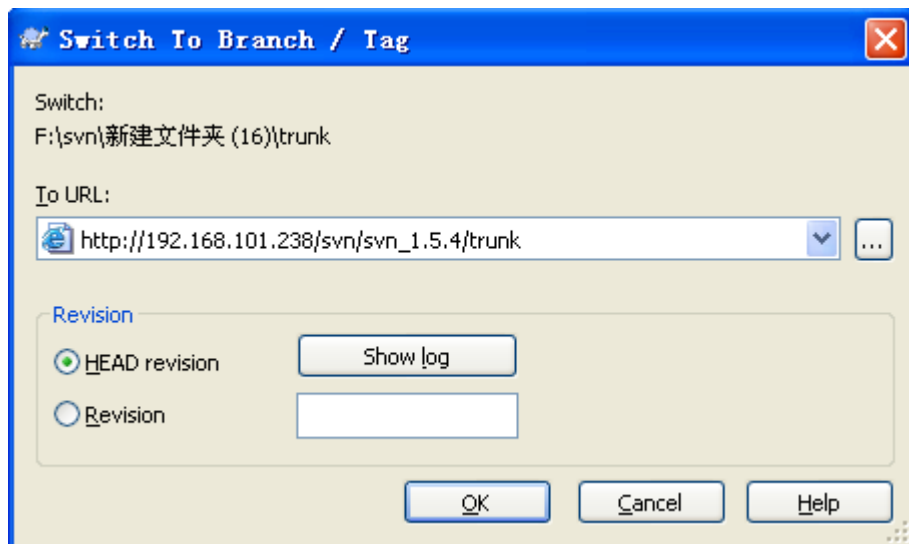
为了能够使用你最新产生的副本你有采用下面几种方法。你可以：

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 1.29. 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/标记/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拷贝。

## 1.16. 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。在你开始使用之前，请先理解分支

与合并是怎么工作的，因为它很复杂。

十分关键之处是 Subversion 的合并与差异关系很密切。合并工作的基础是在版本库对两个分支产生一个差异列表，然后对你的工作副本应用这些差异。举例说明，如果你需要合并版本 N 的修改，那么你要与版本 N-1 比较。新手经常问“为什么我需从开始版本减一”。考虑底层的差异处理，这个问题就会清楚了。为了容易操作，当你使用显示日志选择一个版本范围合并的时候，TortoiseSVN 自动为你做这个调。

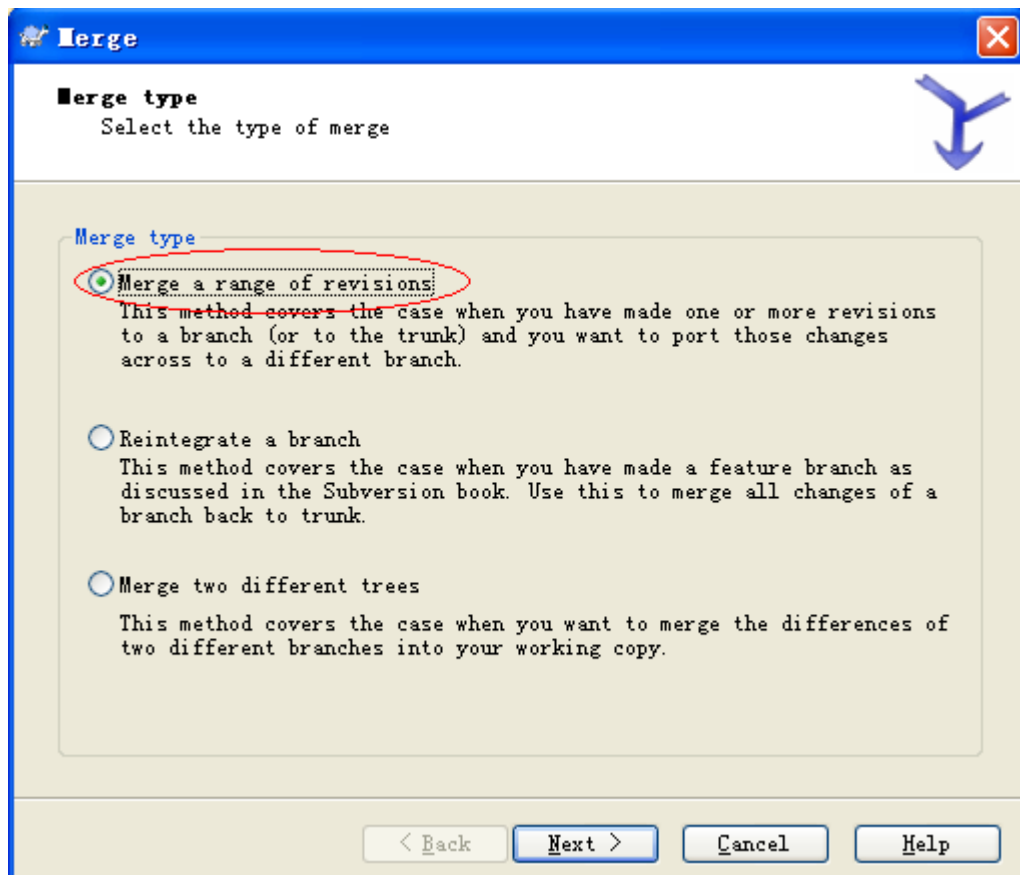
通常来说，在没有修改的工作副本上执行合并是一个好想法。如果你在工作副本上做了修改，请先提交。如果合并没有按照你的想法执行，你可能需要撤销这些修改，命令恢复 会丢弃包含你执行合并之前的所有修改。

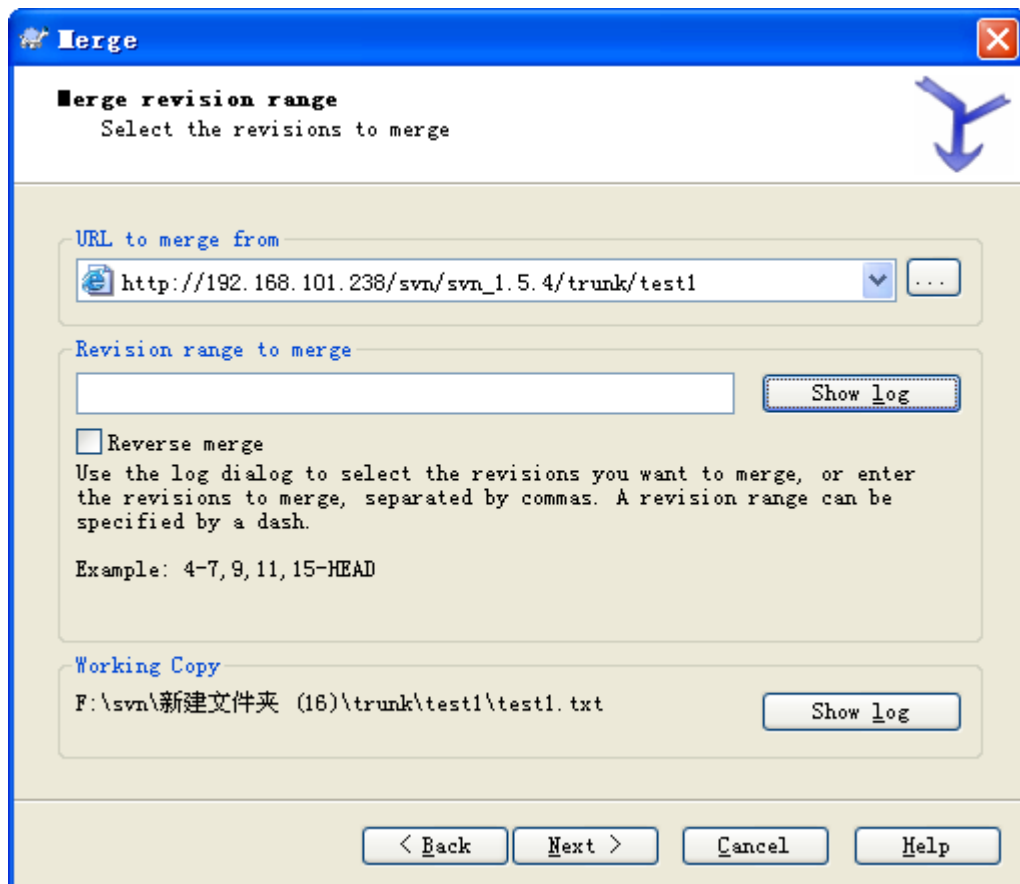
这里有两个处理稍微不同的用例，如下所述。

### 1.16.1. 合并指定版本范围

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。

图 1.30. 合并对话框





为了合并版本，你需要进入接收修改的分支的工作副本，经常是 trunk。选择右键菜单 TortoiseSVN → 合并...

1. 它包含了你想应用到工作副本的修改。你也可以点击...浏览版本库，找到渴望的分支。如果你以前已经从这个分支合并过，可以直接从包含历史的下拉列表选择以前使用的 URL。
2. 在从:域输入文件夹在分支或标记中的完整 URL，因为你要将同一分支的版本范围合并到工作副本，所以要确保使用 "从:" URL 检查框选中。
3. 在从版本域输入开始版本号。它是在你要执行合并的修改之前的版本号。切记为了合并，Subversion 将会创建一个差异，所以开始点务必准确。例如，你的日志象这样：

4. 版本 注释
5. 39. Working on MyBranch
6. 38. Working on trunk
7. 37. Working on MyBranch
8. 36. Create branch MyBranch

```
9.      35. Working on trunk
10.     34. Working on trunk
11.     ...
```

如果你要将 MyBranch 的修改合并到 trunk，应该选择 36 作为开始版本，而不是象

你想的是 37。如果你选择 37 作为开始点，那么差异引擎将会比较结束点与版本 37

比较，这就丢失了版本 37 做的修改。如果这听起来很复杂，不要担心，在 TortoiseSVN 中有更简单的方法 ...

选择版本范围最简单的方法是，点击显示日志，列出最近的修改和日志。如果你要 合并单个版本的修改，直接选取那个版本。如果你要合并多个版本，就选择范围(使用通常的 **Shift**-键)。点击确认，就会为你填写合并对话框的全部域，开始版本和 结束版本。

当选择了检查框 使用 "开始:" URL，只有按钮显示日志可用。这是因为显示日志对话框设置了全部开始:和结束:版本。所以你只用上面说的多项选择方法即可。

如果你已经从这个分支合并了一些修改，希望你在提交日志中注明最后一个合并的版本号。这时，你可以在工作副本上使用显示show log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。

12. 如果你没有使用显示show log对话框显示版本范围，那么你需要手工设置结束版本。在 范围中输入你想合并的最后一个版本号。这经常是最新版本，尽管它不必是 - 你只 想合并单个版本。

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。

13. 点击合并按钮完成合并。

现在合并结束。察看合并，看看它是否如预期那样工作，是个好想法。合并通常很复杂，如 果分支与最新版本差别很大，合并经常会出现冲突。

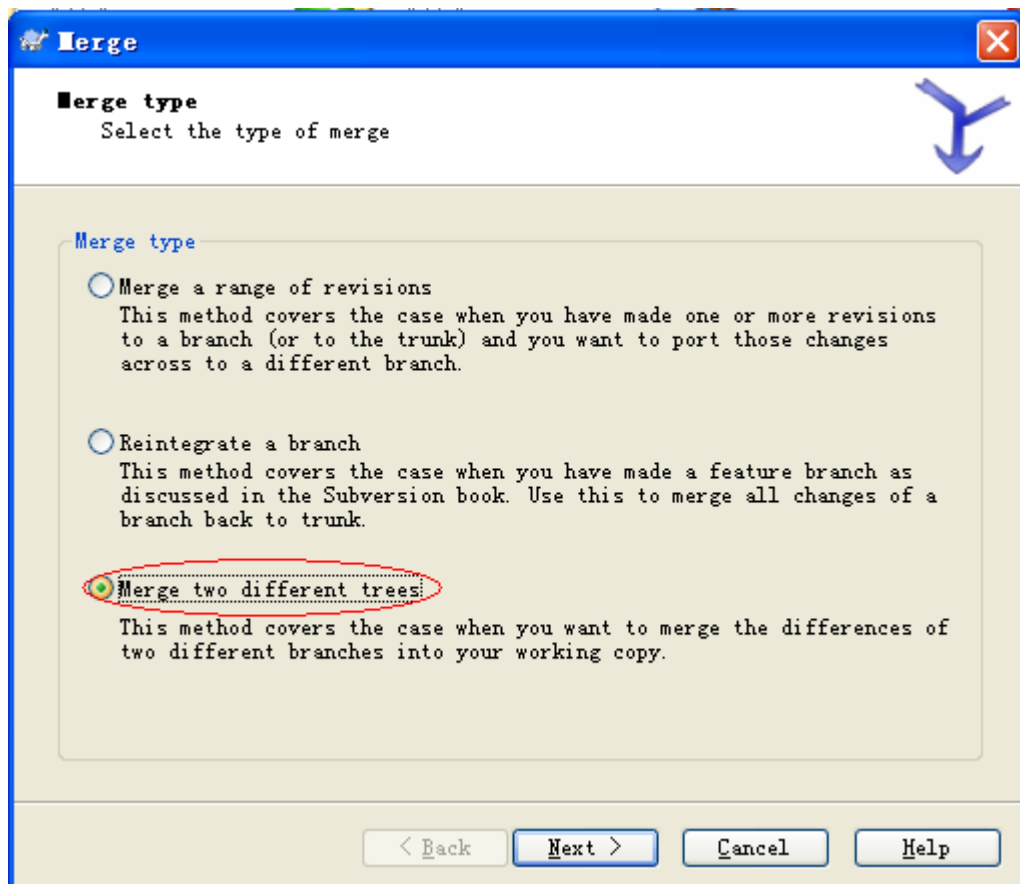
当你已经测试了修改，准备提交时，日志信息应当总是包含这次合并的版本信息。如果你以 后需要执行合并，就需要知道已经合并了什么，因为你不希望多次合并同一修改。不幸的是， 合并信息不会存储在 Subversion 版本库中。阅读 Subversion Book 中的手工跟踪合并以获 得更多信息。

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。



Subversion 不能进行文件与文件夹的合并，反之亦然 – 只能文件夹对文件夹，文件对文件。如果选择了文件，打开合并对话框，那么你必须要在对话框中给出文件的路径。如果你选择了文件夹，打开合并对话框，那么你必须给出文件夹的对话框。

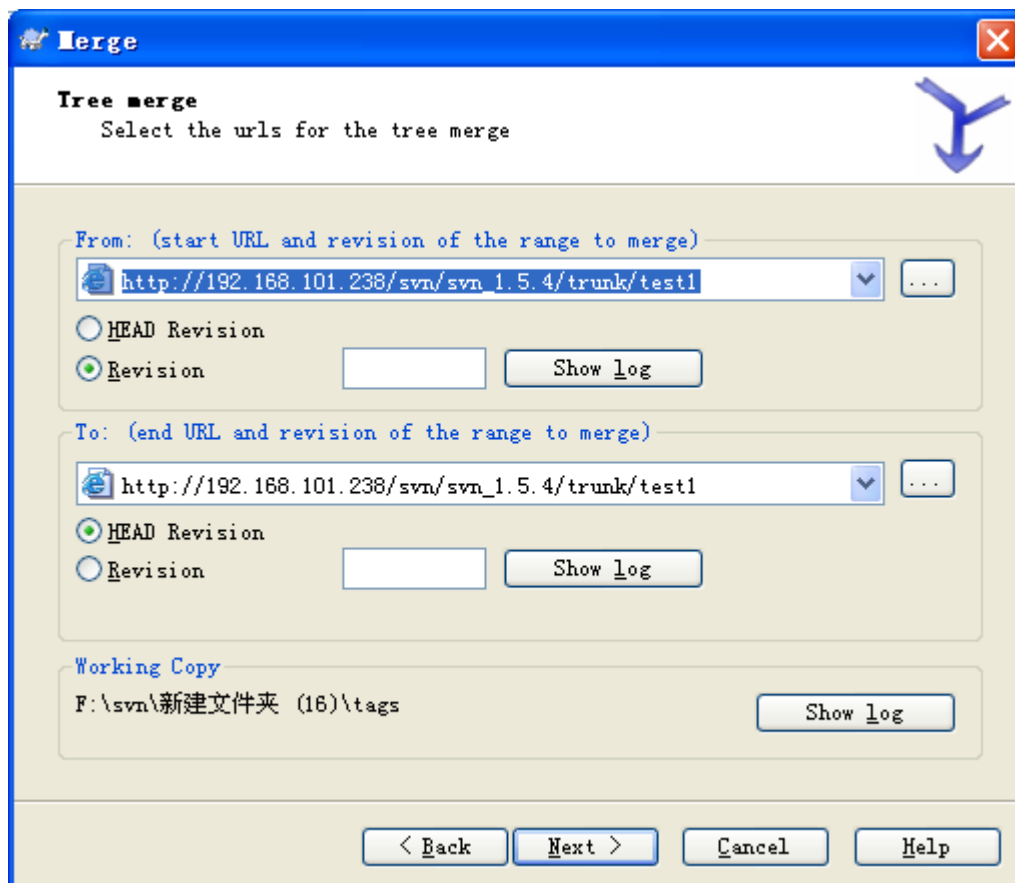
### 1.16.2. 合并两个不同的目录树



这个用例覆盖了这种情况，当你象 Subversion 手册里讨论的那样，创建了一个新特性分支。

所有最新版本的修改都要每周一次合并到新特性分支，等新特性完成后，你向将它合并到最新版本。因为你已经保持了新特性分支和最新版本同步，所以除了你在分支做的修改，其它部分在分支和最新版本应该是相同的。于是在这种情况下，你应当用比较分支和最新版本的方法来合并。

为了将新特性从分支合并到最新版本，你需要进入最新版本的工作副本。在右键菜单选择 TortoiseSVN → 合并...



1. 在开始:域在开始输入 trunk 文件夹的全路径。这听起来好像是错误的,但是切记 trunk 是 你想增加分支修改的开始点。也可以点击 ...浏览版本库。
2. 因为你在比较两个不同的树,确认没有选择使用 "开始:" 路径检查框。
3. 在结束:域输入关注的分支中文件夹的全路径。
4. 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。

你也可以使用显示日志选择版本。注意这种情况下,你不能选择版本范围,所以此时你选择的版本会实际出现在版本域。

5. 点击合并按钮完成合并。

这种情况下,因为新特性已经集成到最新版本,你不再需要这个新特性分支。新特性分支变成多余的,如果需要可以从版本库删除它。

### 1.16.3. 预览合并结果

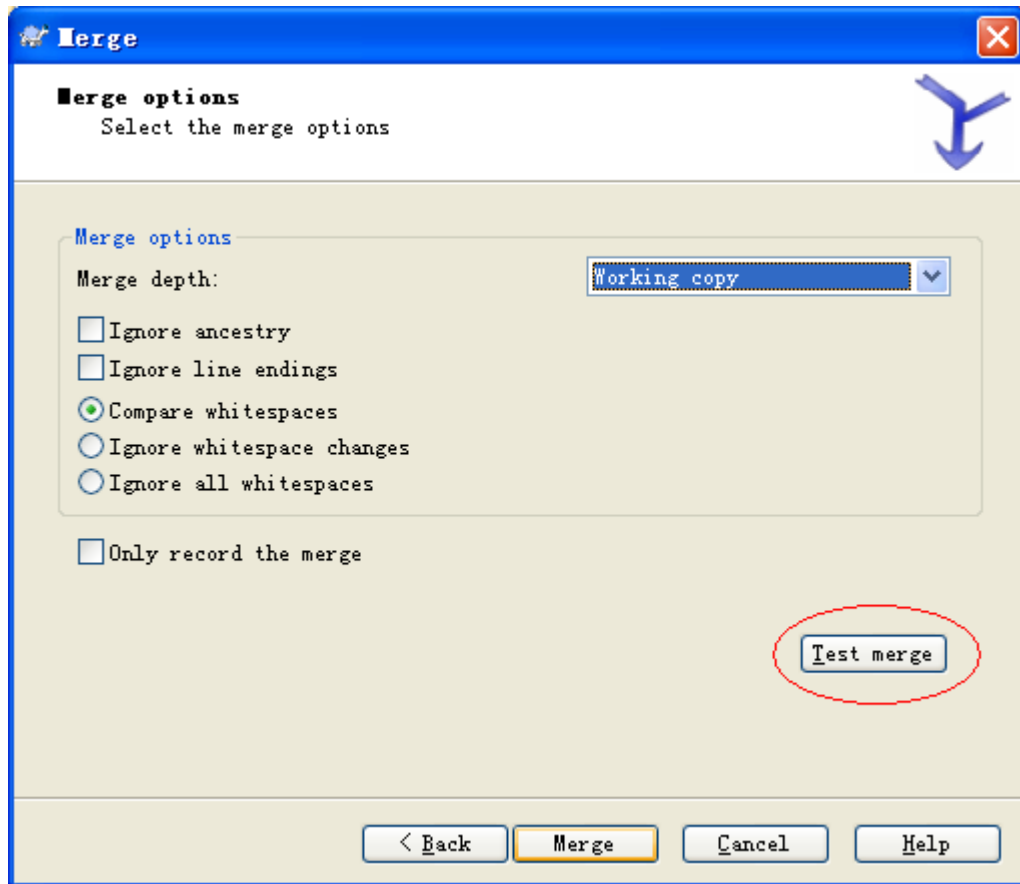
如果你不信任合并操作,可以在允许它修改你的工作副本之前预览效果。有三个额外的按钮可以帮着你预览。

统一差异创建差异文件(切记合并基于差异),显示你的工作目录哪些行将要被修改。因为这是统一差异(补丁)文件,所以离开上下文,它经常很难读。但是对于小的修改,由于它将所有修改在一起显示,因此很有用。



差异显示修改文件列表。双击任一文件启动差异察看器。不像统一差异，它显示具有前后关系的详细修改信息。象统一差异那样，你看到的是开始版本：和 结束版本：之间的差异。它不显示应用此改变之后，你的工作版本如何改变。

演习运行执行合并操作，但是根本不 修改工作副本。它显示在真实的合并中要修改的文件列表，还告诉你哪里会出现冲突。



#### 1.16.4. 忽略祖先

大部分时候，你要合并文件的历史，于是相对于公共祖先合并。有时你需要合并或许是有关系的文件，但是不在你的版本库中。例如，你已经将一个第三方库的版本 1 和 2 导入到两个 单独目录。尽管它们逻辑相关，因为 Subversion 只看到你导入的文件，所以它不知道这些 关系。如果你试图合并这两个树的修改，将会看但完全的删除和增加。让 Subversion 只使用路径差异，不关心历史差异，选中忽略祖先检查框。

### 1.17. 锁

“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。

你的公司过去经常使用 vcs 锁定，这是个管理决定，“锁定是最好的”。

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。

### 1.17.1. 锁定在 Subversion 中是如何工作的

默认情况下，所有的东西都没有锁定，只要有提交权限的人都可以在任何时候提交任何的文件。其他人会定时更新他们的工作复本，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，本地的复本都被设成只读，除非该工作复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，还在工作复本中记录。如果你有多个工作复本（在家，在单位），那么在这些工作复本中，只允许对其中一份拥有锁。

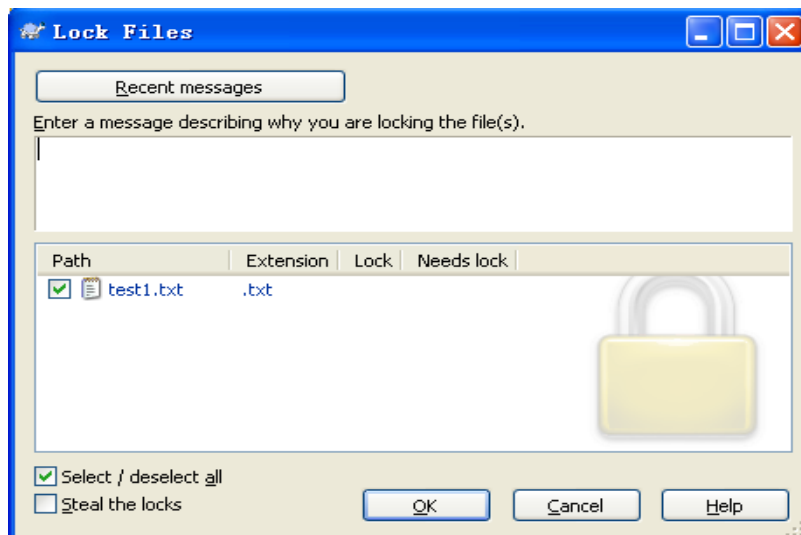
如果你的合作者之一请求一个锁，但却外出旅游去了，你怎么办？Subversion 提供了一种强制锁。释放别人拥有的锁被称为破坏锁定，强制获得别人拥有的锁称为窃取锁定。当然，如果你想要与你的合作者保持良好的关系，轻易不要这么做。

锁在库中进行记录，一个锁定令牌建立在你的本地工作复本中。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。

### 1.17.2. 取得锁定

选择工作复本中你想要获取锁定的文件，然后选择命令 TortoiseSVN → 取得锁定....

图 5.32. 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。

如果你选择一个文件夹，使用 TortoiseSVN → 获取锁定... 锁定对话框将显示所有子文件夹中的所有文件。如果你真的要锁定整个目录，就这么做，但如果你这么做，可能会很不受你的合作者的欢迎。小心使用...

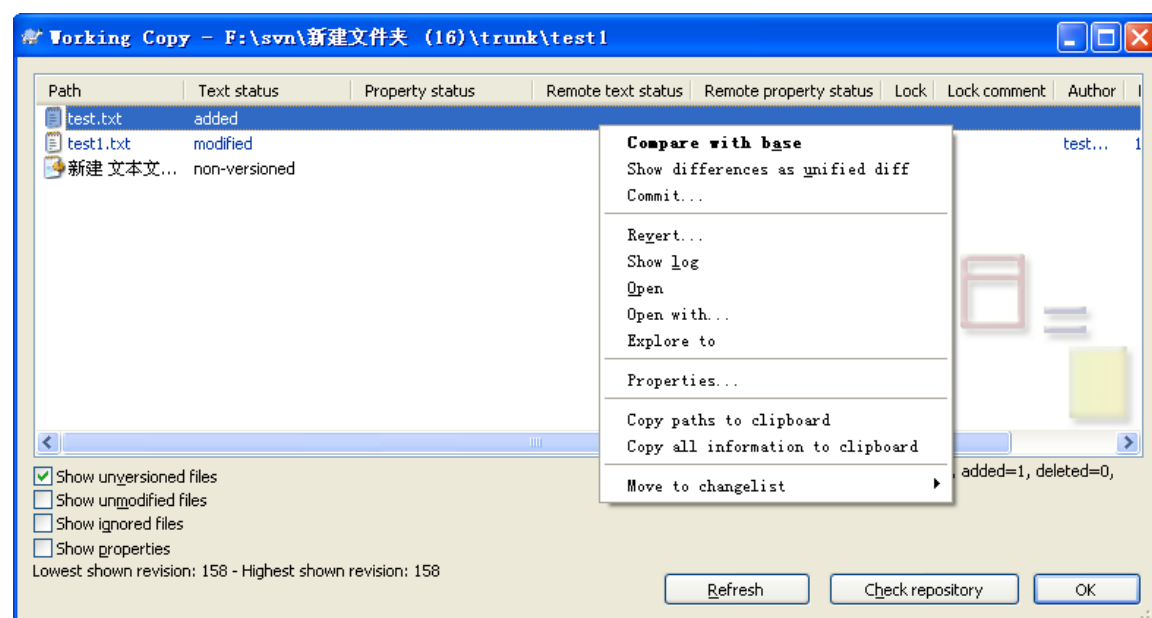
### 1.17.3. 释放锁定

为了确保你不会忘记释放锁，你不需要做别的事，在提交对话框中，总是会显示锁定的文件，并总是默认被选中。如果你继续提交，选中的文件中的锁就被移除了，就算你从没有修改过。如果你不希望释放某文件的锁，你可以取消选中它（如果你没有修改过）。如果你希望保持一个修改过的文件的锁，你需要在提交之前选中保持锁定复选框。

要手动释放锁定，选中工作副本中要释放的文件，选择命令 TortoiseSVN → 释放锁定。不需要输入什么 TortoiseSVN 会联系版本库并释放锁。你可以对一个文件夹来使用这个命令 释放其中的所有锁定项。

### 1.17.4. 检查锁定状态

图 5.32 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 → check for modification。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。

## 1.17.5. 让非锁定的文件变成只读

正如上面提到的，最有效的使用锁的方式是对一个文件设置 `svn:needs-lock` 属性。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。

## 1.17.6. 锁定钩子脚本

当你使用 Subversion 1.2 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。

## 1.18. 创建并应用补丁

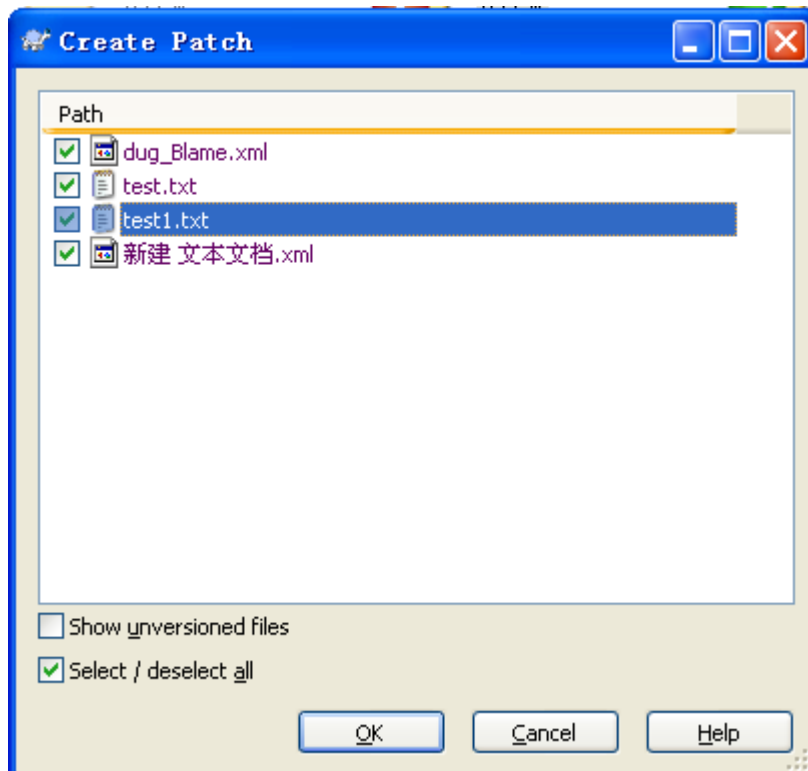
对开源工程(比如本工程)来说，每个人对仓库都有读访问权，并且任何人都可以对该工程做出修改。那么如何控制这些修改呢？如果任何人都可以提交自己的修改，那么这个工程可能永远都会处于不稳定状态，而且很有可能永远的瘫痪下去。在这种情况下，修改需要以补丁文件的形式先递交到有写访问权限的开发组。开发组可以先对该补丁文件进行审查，然后决定将其提交到仓库里或者是退还给作者。

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拷贝和基础版本的不同点。

### 1.18.1. 创建一个补丁文件

首先你需要做出修改并测试这个修改的内容。然后在父目录上使用 TortoiseSVN → 创建补丁...代替 TortoiseSVN → 提交....

图 1.33. 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。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。

### 1.18.2. 应用一个补丁文件

当你对你的工作拷贝打补丁的时候，你应当在与创建补丁文件时相同的目录层次上。如果你不能确定在那个目录层次上，就看一下补丁文件的第一行。例如，如果补丁第一个要处

理的文件是 `doc/source/english/chapter1.xml` 并且补丁文件的第一行是 `Index: english/chapter1.xml` 说明你必须要对 `english` 目录应用这个补丁文件。尽管如此，倘若你在正确的工作拷贝上工作，如果你选择了一个错误的目录层次，TSVN 会察觉到这个错误，并给出正确的建议。

为了给你的工作拷贝打补丁，你至少需要对代码库的读权限。因为合并程序必须要参考修订版本中其他开发人员做的修改。

从那个目录的上下文菜单，点击 `TortoiseSVN → 应用补丁...` 系统会弹出一个打开文件的对话框，让你选择要应用的补丁文件。默认情况下只显示 `.patch` 或者 `.diff` 文件，但是你可以选择“所有文件”。

如果补丁文件以 `.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拷贝的位置。

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，你先选择工作拷贝，然后浏览

补丁文件。第二种方法，你先选择补丁文件，然后浏览工作拷贝。

一旦你选定了补丁文件和工作拷贝的位置，`TortoiseMerge` 就会把补丁文件合并到你的工作拷贝中。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拷贝上，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。

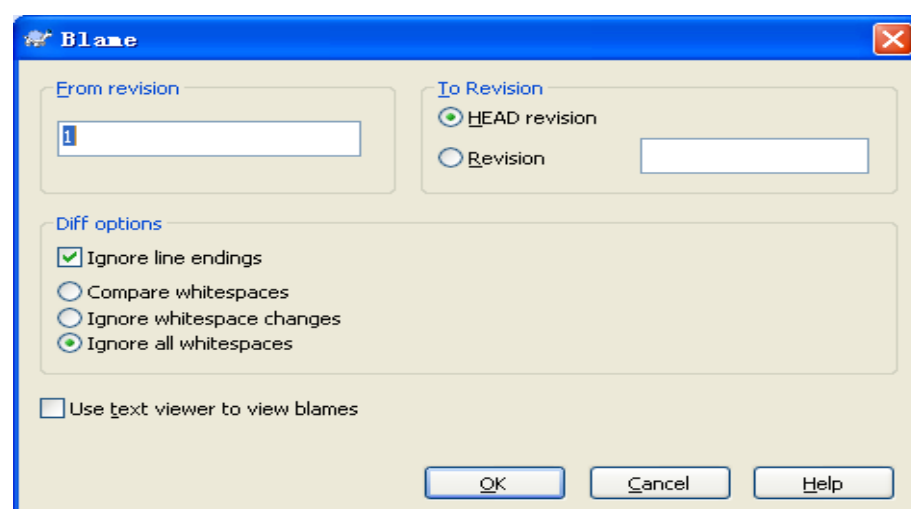
远程开发者的补丁现在已经应用到了你的工作拷贝上，你需要提交它以使每一个人都可以从代码库访问到这些修改。

## 1.18. 谁修改了哪一行？

### 1.19.1. 追溯文件

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。

图 1.34. 评注/追溯对话框





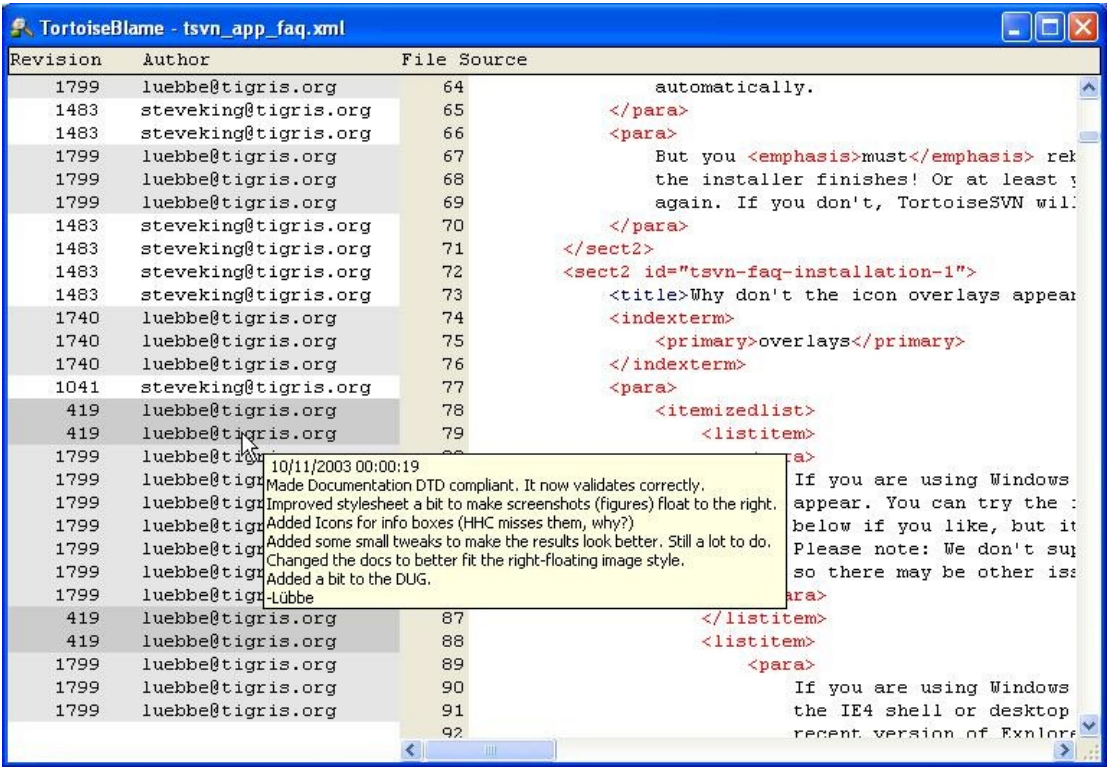
如果对早期版本的修改不感兴趣，你可以设置从哪个版本开始追溯。如果你想追溯每一个版本，你可以把那个数值设置为 1。TortoiseSVN → 追溯... 命令，有时候也叫做 评注命令派上用场的时候了。

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。

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。

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 1.35. 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，所有和该行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show log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。

## 1.19.2. 追溯不同点

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

在版本show log对话框里包含了以下几个选项支持你做这样的操作。

### 追溯版本

在顶部窗口，选择两个版本，然后选择上下文菜单 → 追溯版本。它将取出两个版本的追溯数据，然后使用差异察看器比较这两个追溯文件。

### 追溯不同点

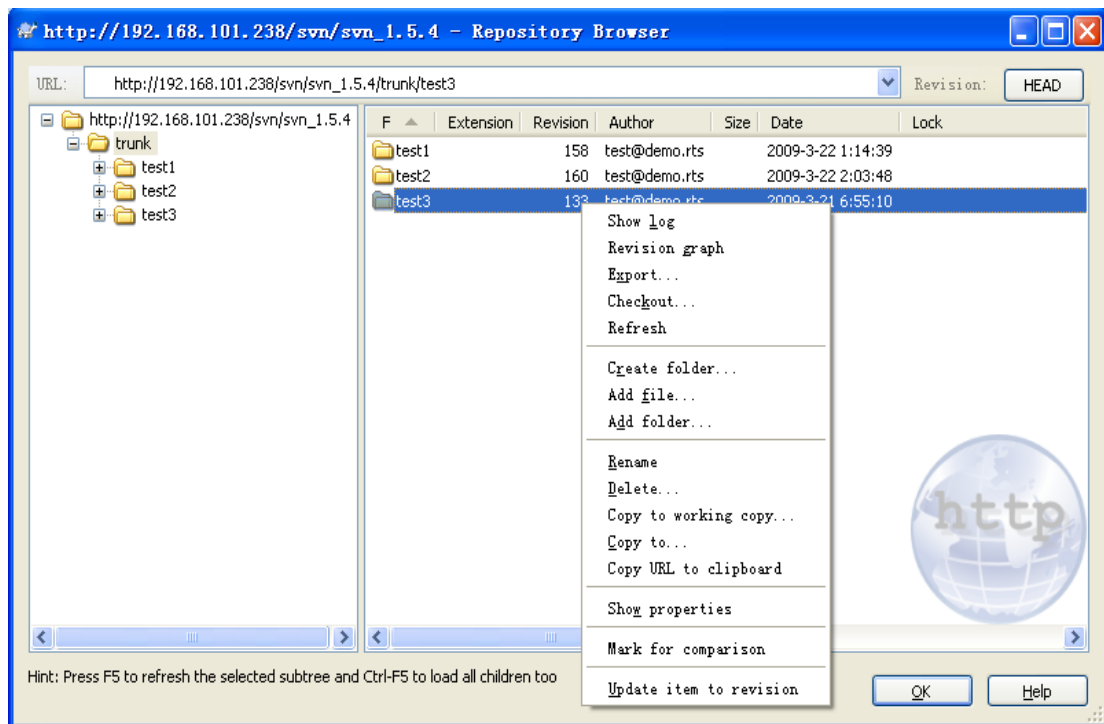
在上面的面板里选择一个版本，然后在下面的面板选择一个文件然后选择上下文菜单 → 追溯差异。这会为选定版本及其上一个版本获取追溯数据，然后用差异阅读器比较两份追溯文件。

与工作拷贝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 → 与工作拷贝比较并追溯。这会为文件的选择版本和工作拷贝获取追溯数据，然后使用差异阅读器比较两份追溯文件。

## 1.20. 版本库浏览器

有时候我们需要在版本库中直接进行操作，而不是在工作拷贝中。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拷贝一样，版本库浏览器允许你浏览版本库的结构和状态。

图 1.36. 版本库浏览器





在版本库浏览器中你可以执行比如拷贝，转移，重命名。直接操作在版本库上。

在版本库浏览器窗口的顶端输入版本库的 URL 地址和你要浏览的版本号。浏览一个历史版本 对一些操作比如覆盖以前删除的文件是很有用的。用右键菜单 → 拷贝到...输入你要覆盖 的被删除文件的工作地址就能做到。

主浏览窗和其他任意目录树浏览器一样。你可以点击顶部的列表头来调整排列顺序。如果你 右击文件 置前转换按数字顺序排序可以实现。这个要比无格式二进制处理数字排序要智能 多了，对在队列中获得版本标签也很有用。这种默认状态在设置对话框中可以设置。

如果你选择两个版本，你可以以单一差异文件显示修改，或者可以使用自带的比较工具清晰地显示整个列表的差异。这个对比较两个日志信息的修改很适用。

如果你选择两个拥有相同历史的标签（特别是/主干/），你可以使用右键菜单 → 显示日志...来查看。

你也可以在版本库浏览器中使用拖拽的操作。如果你把一个文件夹从资源管理器中拖拽到版本库浏览器中，那就实现该文件夹的导入功能。有一点要注意，在导入多个文件时，必须分开一个一个一个地拖进去。

如果要在版本库浏览器中移动一个文件，只用左键拖到它要去的位置。如果你更愿意去拷贝该文件的话，使用这个按住左键拖 。操作就像在资源管理器中一样，拷贝时指针“加”来标记。

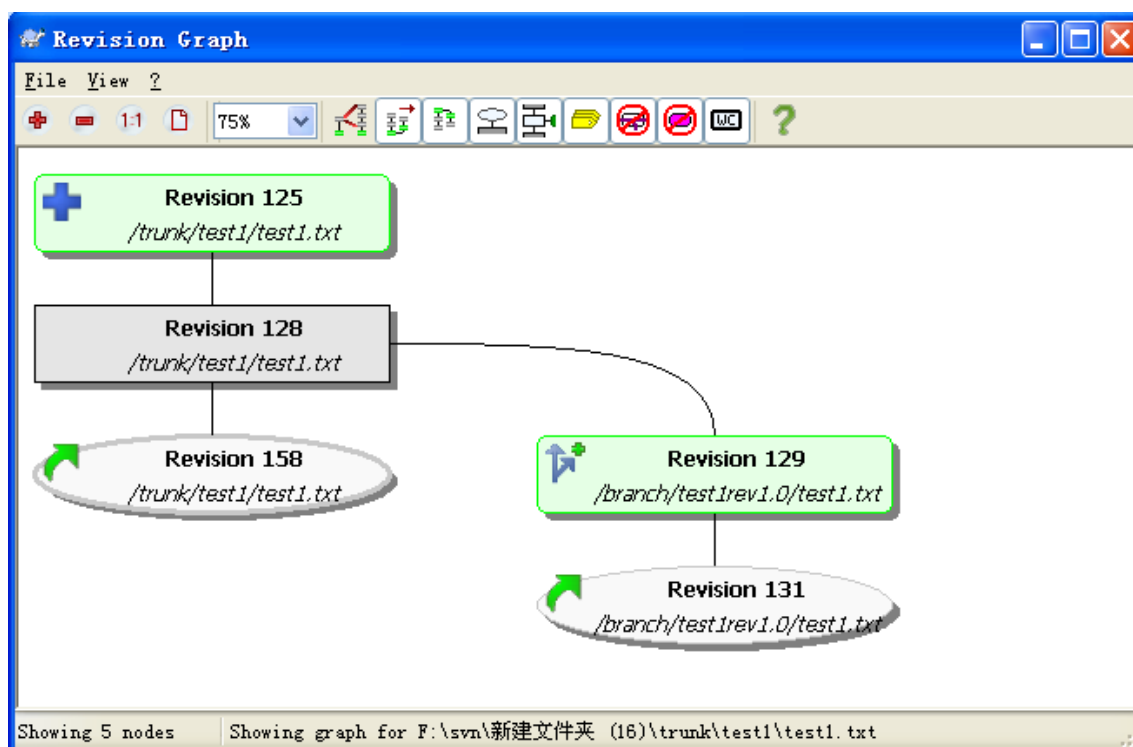
如果你要拷贝/移动一个文件或文件夹到一个新的位置并重命名，你可以右键拖或者按住右键拖这个文件，而不用左键拖。这样，就会显示对话框来为该文件或文件夹重命名。

无论什么时候对版本库的任意操作，都要求加入它的操作日志。这为你操作失误提供了返回的机会。

有时候当你访问一个路径时，会跳出一个错误信息。也许是不可用的 URL，也许是你没有访问权限，或者其他的一些原因，如果你要把这些信息发送出去，只用点击它然后右键 → 复制错误信息到剪贴板，或者简单地用 CTRL+C。

## 1.21. 版本分支图

图 1.38. 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

要实现版本分支图，TortoiseSVN 必须从版本库中获取所有的日志信息。一个有上千版本号的版本库会需要几分钟的时间，这个主要有服务器速度，网络带宽等因素决定。如果你使用的是 Apache 的方式而且版本号超过 300,000 的话估计要等上比较长的时间了。

版本分支图将显示以下内容：

增加文件/文件夹

已增加或通过拷贝生成的文件/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

当一个分支（或者主干或者标签）从最新的节点开始以来有了新的更改，将有一个插件显示出来。也就是说，任意分支的最新版本都会显示在分支图上。

一般的文件/文件夹

其他一般的文件用 plain rectangle 标示。

默认的分支信息只会显示那些有增加或删除文件的节点。要显示一个项目的所有版本将会产生一个庞大的毫无价值的版本图。如果你呀看到所有的版本，在视图工具栏里可以找到实现该功能的选项。

这也有一个功能来排列版本分支旁支。它根据日志来对分支排序。如果分支从拷贝创建后都没有修改过一般被认为是一个标签而独立保存起来。每个分支（创建后有过修改的）会有它们自己的列，所以你可以看到分支延展到什么情况了。

任何时候只要鼠标在版本框上滑过，都会显示该版本的日期、作者和备注信息。

选择两个版本（用 `ctrl+左击`），你可以在右键菜单下查看这两个版本的差异。你可以在分支的起始点查看差异，但一般你会在一个分支的结束点来查看，比如，在最新版本。

你可以用查看单一差异文件的方式来查看差异，它在单一的文件中显示差异。如果你选右键菜单 → 比较版本所有修改过的文件都会呈现在列表中。双击一个文件名可以返回文件版本号和他们的差异。

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。



#### 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。

## 1.22. 导出一个 Subversion 工作拷贝

有时候你需要拷贝出一个没有 `.svn` 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。不用像一般的先把所有源文件拷贝出来然后将所有 `.svn` 文件删除。TortoiseSVN 提供这样的功能 TortoiseSVN → 导出...。如果你要用这个功能操作拥有一份工作拷贝，将会在要求你保存一份干净的文件。一般，要导出被版本控制了的文件，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拷贝中存在的文件导出来。另外可以使用 `svn:externals` 来忽略请求。

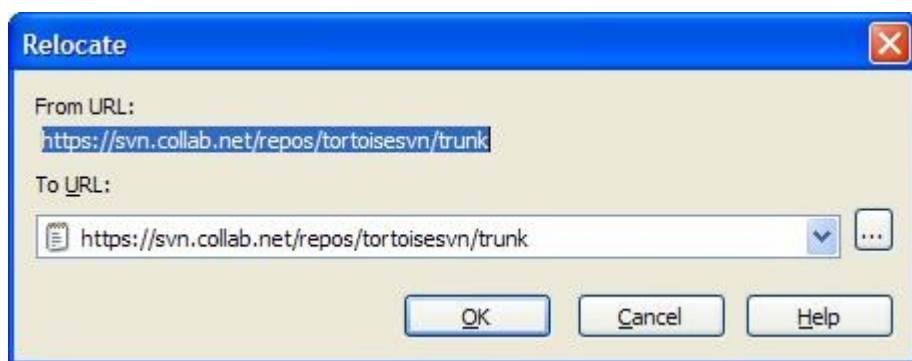
另外一个导出的方法是 右键拖工作拷贝到另外的保存位置，然后选择右键菜单 → SVN 导出到这或者右键菜单 → SVN 全导出到这。后者可以把未受控制的文件也导出。

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

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，然后使用上下文菜单 → 导出。

## 1.23. 重新定位工作拷贝

图 1.38. 重定位对话框



如果你的版本库因为某些原因而重定位了(IP/URL). 也许你不能进行操作, 不能提交, 不能从新的定位检出拷贝, 也不能将修改的数据提交到新位置的版本库中, TortoiseSVN → 重新定位也许正是你需要的。它的做法很简单: 查找所有 .svn 文件的接口然后把 URL 值改为新的版本库地址。



#### 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, 只有当你的工作拷贝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, 它将破坏你的工作拷贝, 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。

## 1.24. 与 BUG 跟踪系统/问题跟踪集成

在软件开发中, 修改依赖于一个 bug 或问题编号是很常见的。bug 跟踪系统的用户(问题跟踪者)喜欢在问题跟踪中将 Subversion 的修改与一个指定编号联系起来。因此很多问题跟踪者提供了一个预提交钩子脚本, 分析日志, 查找提交相关的 bug 编号。这稍微有些不可靠, 因为它依赖于用户写完全的日志, 预提交钩子才能正确分析。

TortoiseSVN 可以在两个方面帮助用户:

1. 当用户输入日志信息时, 一个定义良好, 包含问题编号, 与此提交相关的的行, 会自动增加。这样减少了用户输入的问题编号不能被 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。

你可以在 TortoiseSVN 中集成 bug 跟踪工具。为了使用这个特性，你要定义一些以 bugtraq: 开始的属性，它们只能在文件夹上设置。

有两个方法集成 TortoiseSVN 和问题跟踪。一个基于简单字符串，另一个基于正则表达式。它们的用法是：

bugtraq:url

将这个属性设置为你的 bug 跟踪工具的地址。它必须编码并且包含 %BUGID%。%BUGID% 用你输入的问题编号替换。它允许 TortoiseSVN 在 show log 对话框中显示链接，于是你 可以在察看版本日志时直接进入 bug 跟踪工具。你可以不提供这个属性，但是这样 TortoiseSVN 就不能显示链接了，只能显示问题编号。例如 TortoiseSVN 使用 <http://issues.tortoisesvn.net/?do=details&id=%BUGID%>。

bugtraq:warnifnoissue

如果你想 TortoiseSVN 给出空问题编号的警告，就设置为 真。有效取值是 真/假。如果没有定义，那么假定为 假。

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

这个属性将问题追踪系统激活为输入框模式。 如果设置了这个属性，在拟提交时，TortoiseSVN 会提示你输入问题单号码。它通常会在日志信息后面添加一行。必须 包含 %BUGID%，在提交时会被替换为问题单号。这确保了你的提交日志包含了问题单 号，保证了单号可以被问题追踪工具解析，从而与提交关联。例如 TortoiseSVN 项 目使用 Issue : %BUGID%，但是这依赖于你的工具。

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

是 TortoiseSVN 的提交对话框中用来输入问题单号码的输入项，如果没有设置，将 会显示 Bug-ID / Issue-Nr:，要记住窗口不会为适应标签而改变大小，所以请保持 标签的小于 20-25 个字符。

bugtraq:number

如果设置为 true，问题单号文本框只能输入数字，一个例外是逗号，所以你可以使用逗号分割输入的多个号码。合法的值包括 true/false，如果没有设置，默认是 true。

在使用正则表达式的方法中，TortoiseSVN 不会显示一个单独的输入框，而是标记用户输入的日志信息，认为这些标志可以被问题追踪工具识别。这是在用户编写日志信息的时候完成的，这也意味着 bug ID 可以出现在日志信息的任何位置！这种方法非常灵活，也是 TortoiseSVN 项目本身使用的方法。

如果同时设置了 bugtraq:message 和 bugtraq:logregex 属性，日志正则表达式会优先使用。



#### 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！

一些 tsvn: 属性需要 true/false 值。它也理解 yes 是 true 的同义词，no 是 false 的同义词。



#### 设置文件夹的属性

为了系统能够工作，这个属性必须设置到文件夹上。当你提交文件或文件夹，属性会从文件夹上读取。如果没有发现属性，TortoiseSVN 会向上级查找，直到发现一个没有版本化的文件夹或根目录（例如 C:\）才会停止，如果你能够确定每个用户只从 trunk/检出，而不是其他目录，你可以直接在 trunk/上使用这个属性，而不必每个子目录都设置。如果你不能确定，你必须为每个子目录设置这些属性。一个深级目录的设置会覆盖高级目录（离 trunk/更近的）。

对于 tsvn:属性，你只能对于所有子目录使用递归检查框设置属性，不能设置文件的属性。

问题追踪集成并没有限制在 TortoiseSVN，可以用于所有的 Subversion 客户端。

## 1.25. 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 and WebSVN，TortoiseSVN 提供了链接这些浏览器的方法。

你可以通过 TortoiseSVN 选择集成一种版本库浏览器，为此，你需要定义一些链接的属性，  
必须设置在文件夹：

```
webviewer:revision
```

将这个属性设置为版本库浏览器显示所有本版本修改内容的 url，它必须是 URI 编码的，也必须包含%REVISION%。在问题单中%REVISION%将会被替换成版本号。这

允许 TortoiseSVN 在 show log 对话框的增加这样的条目右键菜单 → 在版浏览器中查看  
此修订

webviewer:pathrevision

设置。



### 设置文件夹的属性

为了系统能够工作，这个属性必须设置到文件夹上。当你提交文件或文件夹，属性会从文件夹上读取。如果没有发现属性，TortoiseSVN 会向上级查找，直到发现一个没有版本化的文件夹或根目录（例如 C:\）才会停止，如果你能够确定每个用户只从 trunk/检出，而不是其他目录，你可以直接在 trunk/上使用这个属性，而不必每个子目录都设置。如果你不能确定，你必须为每个子目录设置这些属性。一个深级目录的设置会覆盖高级目录（离 trunk/更近的）。

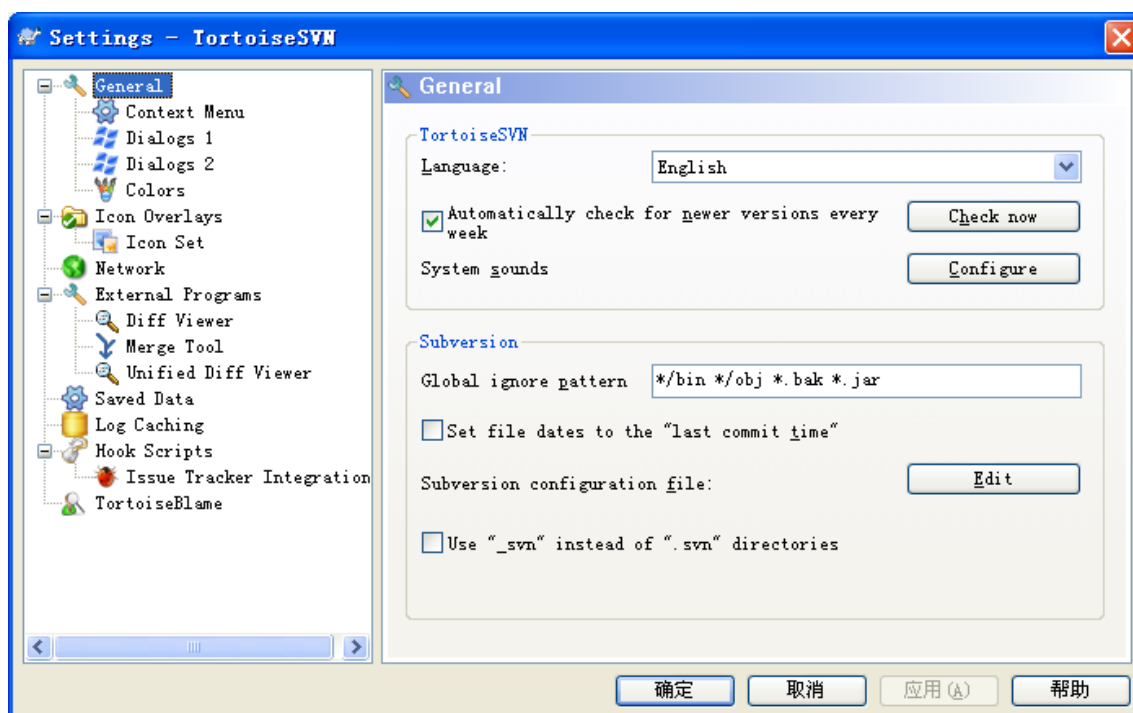
对于 tsvn:属性，你只能对于所有子目录使用递归检查框设置属性，不能设置文件的属性。

## 1.26. 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。

### 1.26.1. 常规设置

图 1.39. 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

全局忽略样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。那些符合样式的文件，在执行导入操作时将同样被忽略掉。通过在样式框中输入文件名或扩展名来忽略文件或文件夹。不同的样式之间将以空格分隔，例如 \*/bin \*/obj \*.bak \*.~?? \*.jar \*. [Tt]mp。例子中前两个条目表示文件夹，其他四个表示文件。这些样式将采用文件名模糊匹配的方式来匹配要忽略的文件。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 全局一忽略 样式，那么它将覆盖你在这里做的设置。该 Subversion 配置文件可以象下面段落描述的那样，通过 编辑 按钮来访问。

忽略样式将作用于你所有的项目工程。因为它是非版本控制的，所以它将不会对其 他的用户起作用。相对而言，你也可以使用可版本控制的 svn:ignore 属性来把要 忽略的文件或文件夹排斥在版本控制之外。

将文件日期设置为“最后提交时间”



该选项通知 TSVN 在做检出或更新操作时，把文件日期设置为最后提交的时间。否则 TSVN 将使用当前日期。如果你在做软件开发的话，使用当前日期是总体上最好的选择，因为那些软件构建器通常通过查看时间戳来决定需要编译哪些文件。如果你使用了“最后提交时间”并把代码文件还原到了一个旧版本，你的工程可能就不会像你期望的那样被编译了。

## Subversion 配置文件

使用 编辑 按钮来直接编译 Subversion 配置文件。有些设置不能被 TSVN 直接修改，就需要在这里完成。Subversion 可以从许多不同的位置读取配置信息，因此你也就需要了解哪一个将优先起作用。

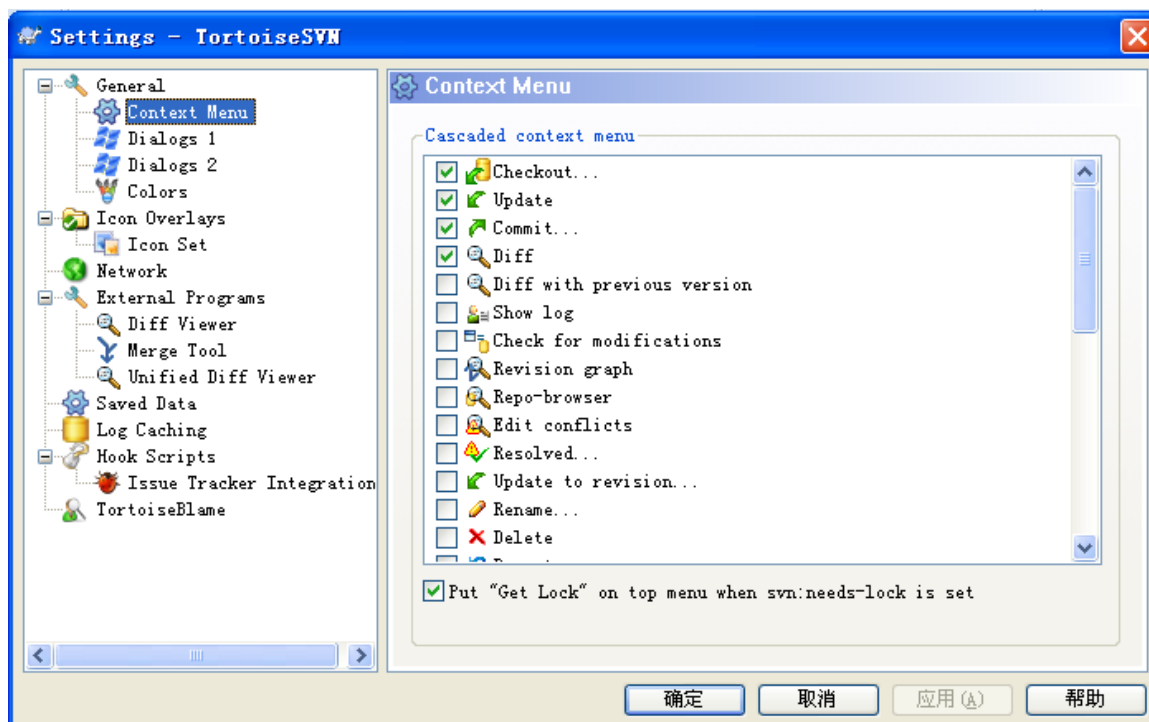
## 使用 “\_svn” 目录替代 “.svn” 目录

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。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拷贝中的管理文件夹从 “\_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拷贝。

## 1.26.2. 外观与样式设置

图 1.40. 设置对话框，外观与样式页面



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

获得锁会有一个特别的情况，你可以将其提升到顶级带但，但是大多数文件不需要锁

定, 这样做只是添加了混乱。然而, 一个标记为 `svn:needs-lock` 属性的文件每次编辑前都需要那个操作, 所以这个菜单会进入顶级菜单会比较方便。选定这个选项, 会使设置 `svn:needs-lock` 属性的文件的 `Get Lock` 出现在顶级菜单中。

如果在你的工作拷贝下有大量的文件, 那么你右键点击该文件夹的时候将花费较长的时间使上下文菜单显示出来。这是因为在你查询文件夹状态的时候, Subversion 要去获取其下所有文件的状态。你可以在这里取消对为右键菜单获取 SVN 状态选项的勾选以避免这种延迟。要注意的是, 文件夹的上下文菜单有时不太正确, 可能会包含一些本来不该在那儿的条目。举个例子, 右键点击一个新添加的文件夹, 你可能会看到 TortoiseSVN → 显示日志的操作选项, 但那个是不会起作用的, 因为该文件夹还没真正提交到版本库里。

选项 使用顶级菜单的快捷键 有三种状态:

未勾选 (默认)

这种状态下, 菜单项全部是由 TSVN 画出的, 不会显示任何快捷键。

已勾选

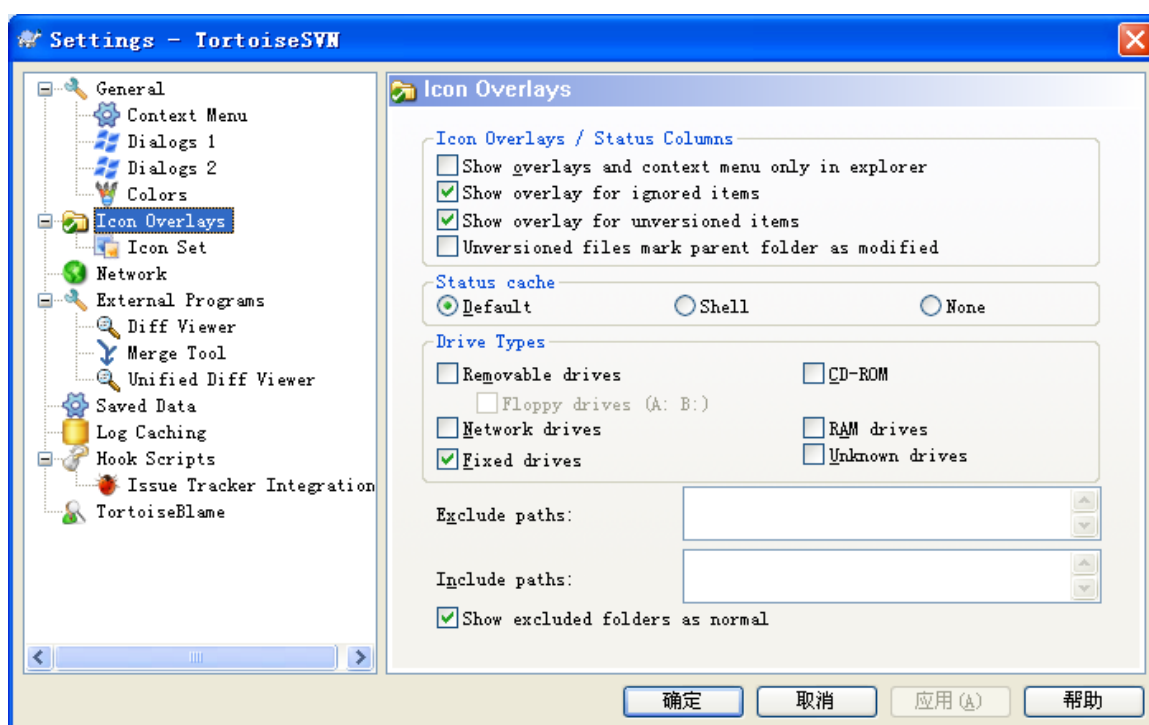
这种状态激活了 TSVN 命令的快捷键, 但这些与浏览器右键菜单中的其他快捷键显然存在起冲突的可能。多次按下快捷键将循环匹配右键菜单中共享同个快捷键的条目。这种状态下, 菜单项是被 Windows 画出来的, 而且看上去丑的要命。

半勾选 (灰色)

这种模式下快捷键被激活, 同时菜单项将以纯文本的形式被画出, 不显示任何图标。

### 1.26.2.1. 图标叠加设置

图 1.41. 设置对话框, 外观与样式页面



此页面允许你选择 TSVN 为哪些条目显示图标覆盖。选择网络磁盘可能会非常慢，所以默认情况下不为定位于网络共享中的工作拷贝显示图标覆盖。你甚至可以取消所有的图标覆盖，但那样做还剩下什么好玩儿的呢？

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。

默认情况下，图标覆盖将不止显示在 Windows 资源浏览器下，同样会显示在所有的打开/保存对话框里。如果你想让它们仅仅显示在 Windows 资源浏览器下，勾选仅在资源管理器中显示图标覆盖选项。

因为它要花费一段时间来获取工作拷贝的状态，TSVN 将使用一个缓存来存储这些状态，从而使浏览器在显示图标覆盖时，资源占用的不太厉害。你可以根据你的系统和工作拷贝的大小来在这里选择让 TSVN 使用哪种类型的缓存：

#### 默认

把所有状态信息缓存在一个独立进程中（TSVNCache.exe）。该进程监视所有驱动器 的更改，并在工作拷贝中的文件被修改时重新获取其状态。该进程以最低优先级运行，所以其他程序不会被它挤兑。这同样意味着状态信息并不是 实时 的，因为它 需要几秒钟时间处理图标覆盖的变化。

优点：图标覆盖递归地显示状态，就是说，如果一个处在工作拷贝深处的文件被修改了，所有途径的文件夹包括工作拷贝的根目录都会显示出修改的图标覆盖。也因为该进程可以向 Windows 外壳发送通知，资源管理器左面的树形图通常也会更改。

缺点：即使你已经不在项目下工作了，该进程仍然持续运行。取决于你工作拷贝的数量和大小，它将占用 10-50 MB 的 RAM 内存空间。

#### Windows 外壳

缓存在外壳扩展 dll 中直接完成，但仅仅是为那些当前可见的文件夹。每次你浏览到其他文件夹，状态信息就会被重新获取。

优点：仅仅需要很少的内存（大约 1 MB），并且可以 实时 显示状态。

缺点：因为仅有一个文件夹被缓存，图标覆盖不会递归地显示状态。在大一些的工作拷贝下，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。

缺点：文件、文件夹的状态信息不会显示在浏览器中。要获知你的工作拷贝是否被修改了，你需要使用“检查更新”对话框。

若你选择了默认选项，将同样决定如下选择：在文件夹包含非版本控制的项目时，把文件夹 图标覆盖标记为已修改。这个有用的设置可以提醒你已经创建了非版本控制的新文件。

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拷贝，而这些工作拷贝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。

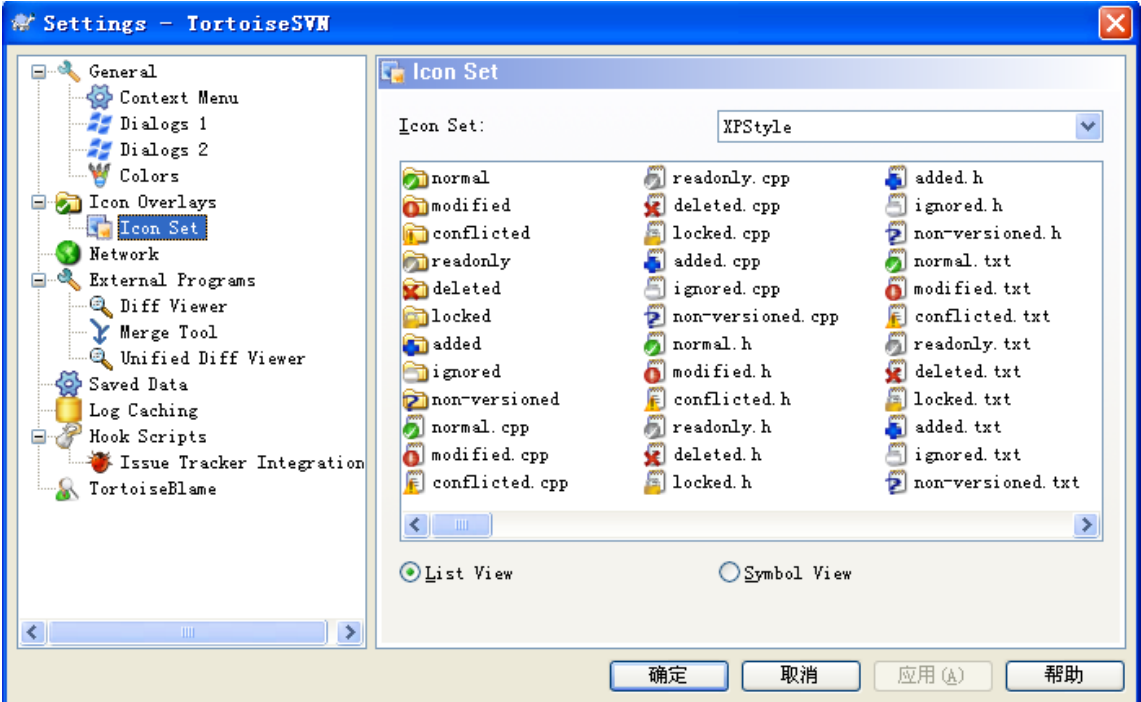
填写 f:\development\SVN\Subversion\* 将在路径以 f:\development\SVN\Subversion 开始的所有文件和文件夹上取消图标覆盖。这意味着你在该路径下的任何文件/文件夹上都将看不到图标覆盖了。

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。

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。

### 1.26.2.2. 图标集选择

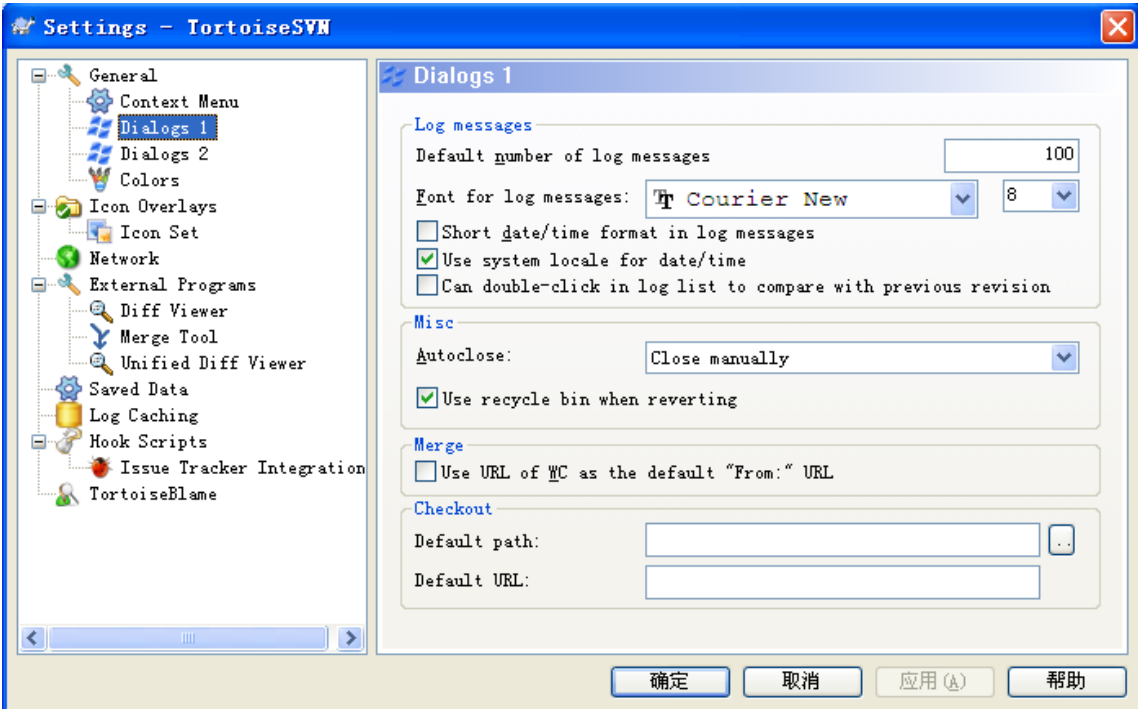
图 1.42. 设置对话框，图标集页面



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。

### 1.26.2.3. TSVN 对话框设置一

图 1.43. 设置对话框，对话框一页面



此对话框允许你按照喜欢的方式去配置一些 TSVN 的对话框。

#### 默认的日志信息数

限制你第一次选择 TortoiseSVN → 显示日志 时，TSVN 向服务器获取的日志信息数。在服务器连接缓慢时很有用。你可以使用 全部显示 或 下 100（条） 来获得更多信息。

#### 日志信息字体

选择日志信息显示的字体样式和大小，作用域为版本show log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。

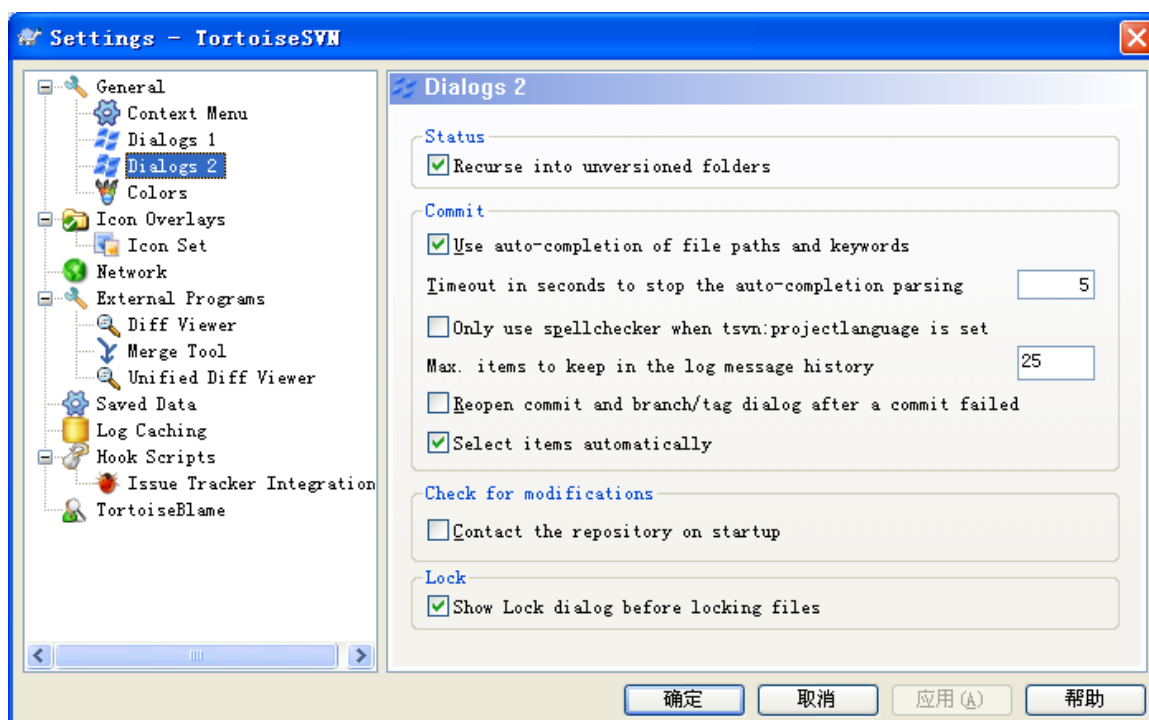
如无错误，自动关闭 即使在有冲突发生时也会关闭。维持对话框打开的唯一条件是发生了错误，使得 Subversion 无法完成任务。举个例子，一个更新操作由于服务器不可达而失败了，或是一个提交操作因为工作拷贝已经过期而失败。

使用工作拷贝的 URL 作为默认的来源 URL

在合并对话框里，默认行为是在每次合并中记忆 起始： 的 URL。无论如何，都有某些人喜欢在他们的版本进化树中从很多不同的位置执行合并操作，他们发现从当前工作拷贝的 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。

## 1.26.2.4. TSVN 对话框设置二

图 1.44. 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。

## 自动完成文件路径和关键词

提交对话框包含了一个功能模块，可以解析被提交的一系列文件名。当你输入一个提交列表中某个文件的前三个字母时，自动完成对话框就会弹出来，使你随后可以点击回车来直接完成这个文件名。选择该选项来使用这个功能特性。

## 对自动完成进行多长时间的分析（秒）

如果有大量文件需要程序检查，自动完成解析器可能会非常慢。该超时时间设置可以防止提交对话框被长时间挂起。若你错过了某些重要的自动完成信息，你可以延长该超时时间。

## 仅在保留了 `tsvn:projectlanguage` 时才进行拼写检查

若你不愿意在所有提交操作时都进行拼写检查，就选择该选项。而后拼写检查功能将在项目属性做出明确要求时才生效。

## 日志中保留的最大条目数量

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。

如果提交失败，自动重新打开提交对话框 当一个提交操作由于某些原因（工作拷贝需要更新、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 更新你的工作拷贝，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。

## 启动时连接版本库

“检查更新”对话框将默认检查工作拷贝，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。

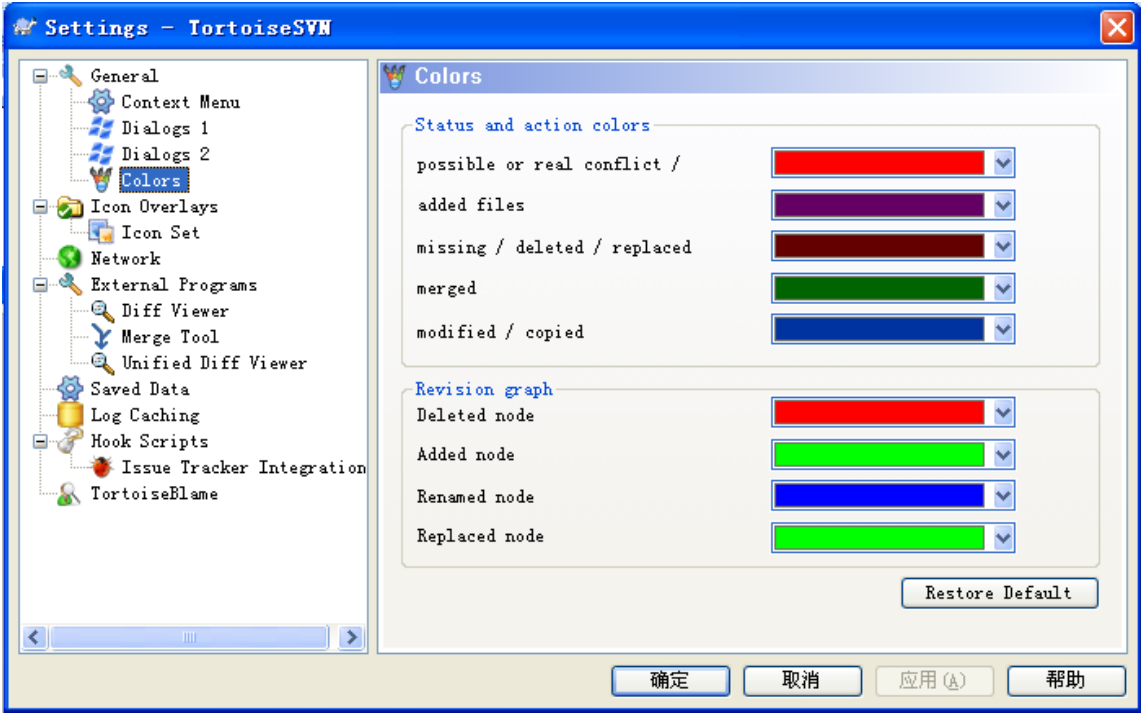
## 按数字顺序排序

版本库浏览器可以使用一个更智能的排序算法，该算法处理包含数字的路径的效果比纯按 ASCII 排序效果要好。这在某些时候会变得很有用，比如用来使版本号标签 排成正确的顺序。该选项可以控制你所使用的默认排序类型。



1.26.2.5. TSVN 颜色设置

图 1.45. 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/有问题

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 已从工作拷贝中遗失的条目；已从版本库中删除；或已经从工作拷贝

删除并且被另一个同名文件替换。

合并

从版本库所做的更改被成功地合并到工作拷贝，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show log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、拷贝或移动操作, 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。

### 1.26.3. 网络设置

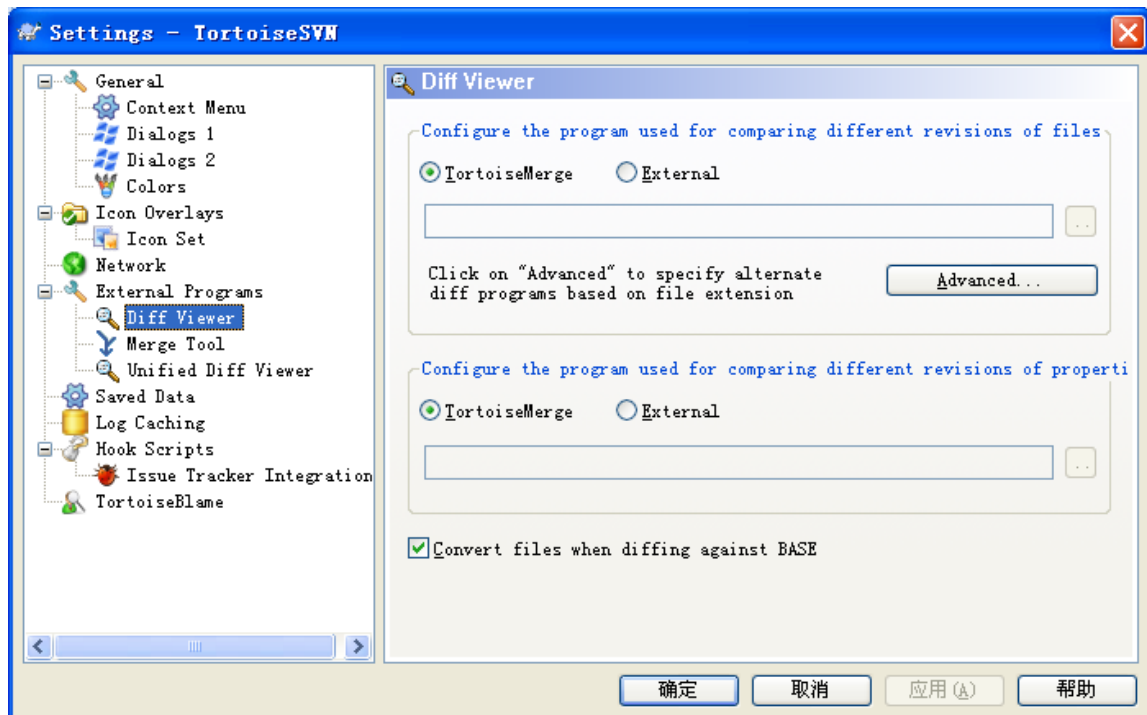
<placeholder-1> 如果需要穿透你公司的防火墙, 在这里可以配置你的代理服务器。  
</placeholder-1>

如果你需要对每个版本库建立一套代理设置, 你必须使用 Subversion 服务器文件来配置。使用编辑来直接访问该配置文件 (servers.txt)。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。

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。

### 1.26.4. 外部程序设置

图 1.46. 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。

#### 1.26.4.1. 差异查看器

有时你可能需要一个外部的差异查看程序来比较不同版本的文件。在为你的命令行填写各种可选参数的同时，要确保这些外部程序从中获得文件名。在 TortoiseSVN 编辑命令行时，使用以 % 开头的替代参数。当外部程序执行至遇到这些替代参数，它将从 TortoiseSVN 那里 获取那些实际的值。参数的填写顺序将依赖于你使用的差异查看程序。

%base

没更改的原始文件

%bname

原始文件的窗口标题

%mine

你更改过的新文件

%yname

你新文件的窗口标题

窗口标题并不一定代表真正的文件名。TortoiseSVN 把它伪装成一个名字用来创建和显示。

因此，倘若你在对比一个版本为 123 的文件和你当前工作拷贝中的文件，名字将显示为 文件名：版本 123 和 文件名：工作拷贝

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
```

如果你使用了 `svn:keywords` 属性来扩展关键词，特别是那些 `revision` 版本关键词，那么在那些纯粹在关键词上取值不同的文件之间对比将有一些不同。同样如果你使用 `svn:eol-style = native`，那么在工作基础版本文件只有纯粹的 LF（换行）结束符的地方，你的文件将有完整的 CR-LF（回车-换行）结束符。TSVN 在做差异对比操作之前，通常会先行对那些扩展关键词和结束符等格式进行解析转换，从而自动隐藏这些差异。无论如何，在遇到大文件时这样做无疑会经过一个很长的处理时间。如果取消对与基础版本比较时转换文件的勾选，那么 TSVN 将忽略这些对文件的预处理。

## 1.26.4.2. 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:

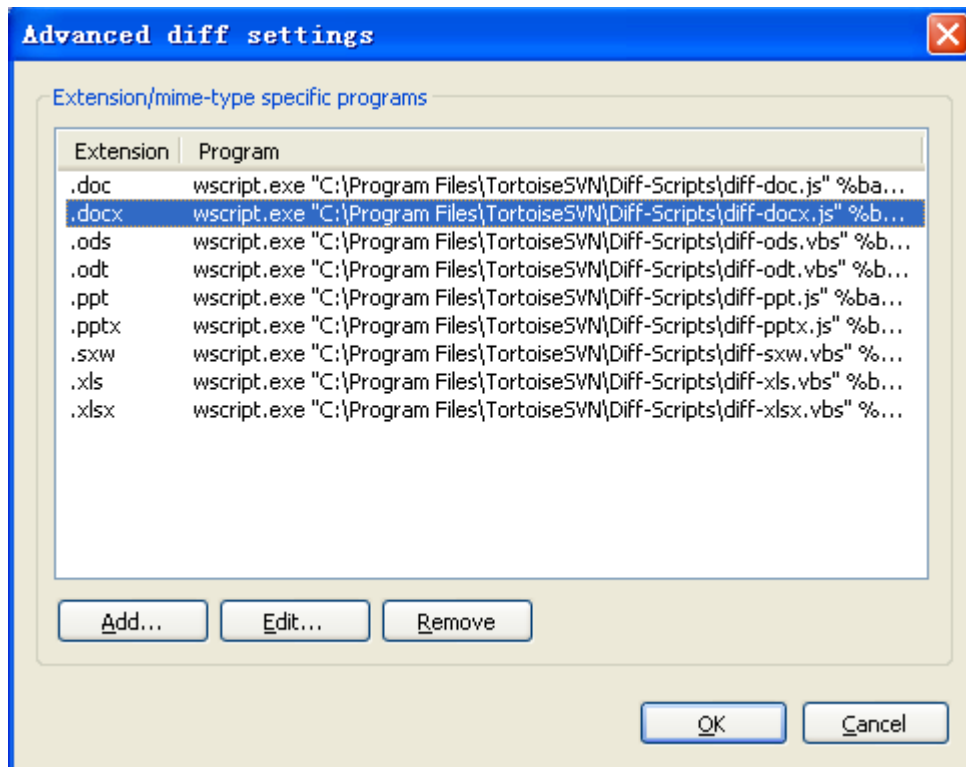
```
C:\Path-To\kdiff3.exe %base %mine %theirs -o %merged  
  
--L1 %bname --L2 %yname --L3 %tname
```

或者使用 Araxis:

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

### 1.26.4.3. 差异查看/合并工具的高级设置

图 1.47. 高级差异比较设置/高级合并设置的对话框



在高级设置中，你可以为每种文件类型都定义一个不同的差异比较/合并程序。例如，你可以指定 Photoshop 作为 .jpg 文件的“比较”程序 :-) 也可以按照 svn:mime-type 属性指定差异/合并程序。

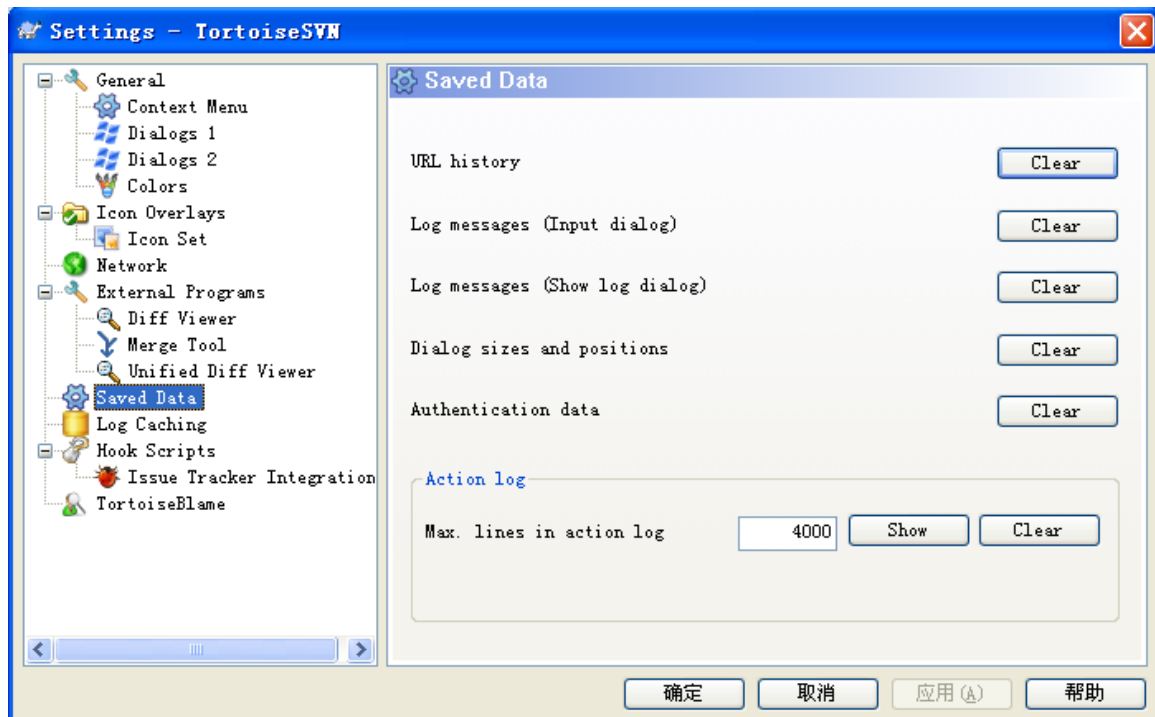
无需任何通配符，你仅需指定文件的扩展名，包含开始的点，但是没有任何通配符。使用 .BMP 来描述 Windows 位图文件，而不是 \*.BMP。如果使用 svn:mime-type 属性，要指定多媒体文件类型，包含斜线，例如 text/xml。

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，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。

## 1.26.5. 已保存数据的设置

图 1.48. 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拷贝，合并那些更改的文件，或仅仅是在使用版本库浏览器时，

TortoiseSVN 都将保存一个记录，记录那些最近使用过的 URL，并在一个下拉列表框 中显示出来。有时列表会被逐渐增多的过期 URL 弄得乱糟糟的，所以有定期清理一 下的必要。

如果你希望从列表中删除单独的条目，可以点击下拉列表，移动鼠标到你要删除的项上，然后按组合键 SHIFT+DELETE。

## 日志信息

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。

## 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）。

### 1.26.6. 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。

缓存托盘图标

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

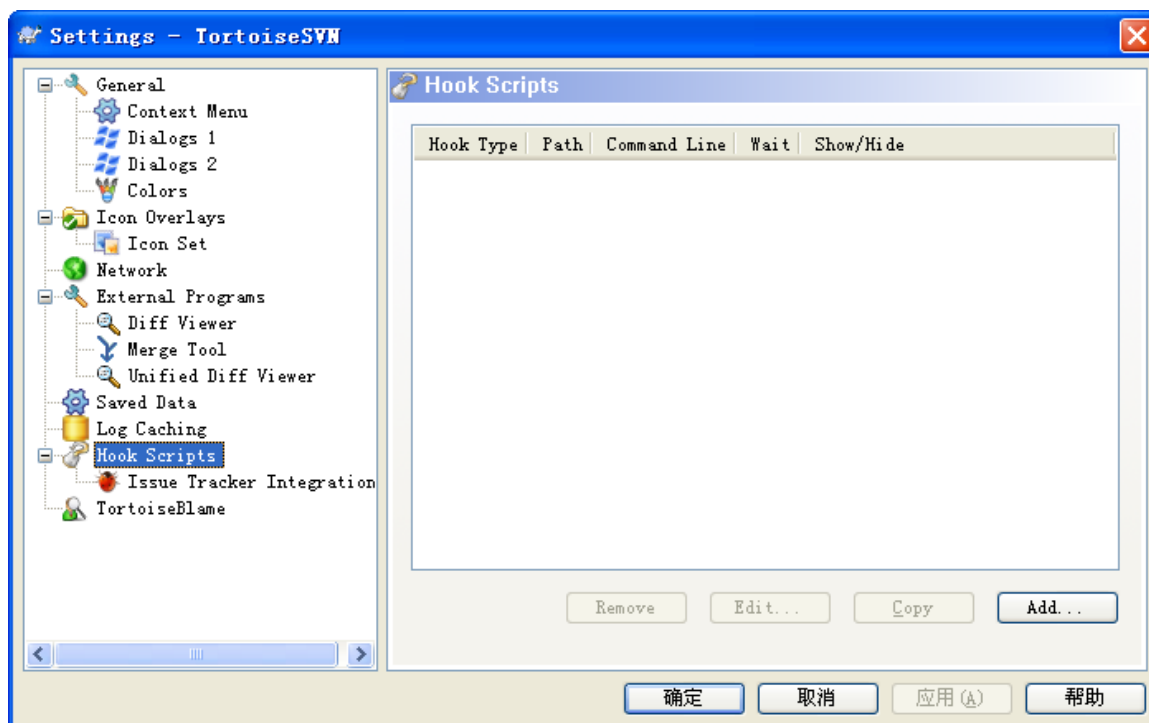
### 1.26.7. Subversion 的工作文件夹

在使用 VS.NET 环境做 web 工程时，遇到.svn 文件夹会出问题，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。

你可以设置环境变量 SVN\_ASP\_DOT\_NET\_HACK，来通知 Subversion 用 \_svn 文件夹替代 .svn 文件夹。你必须重启你系统的外壳程序来使环境变量生效，一般意味着你要重启 PC。这项替代工作就变得非常简单，你只需从常规设置页面选择一个单选框：使用“\_svn”目录替代“.svn”目录。

### 1.26.8. 钩子脚本

图 1.49. 设置对话框，钩子脚本页



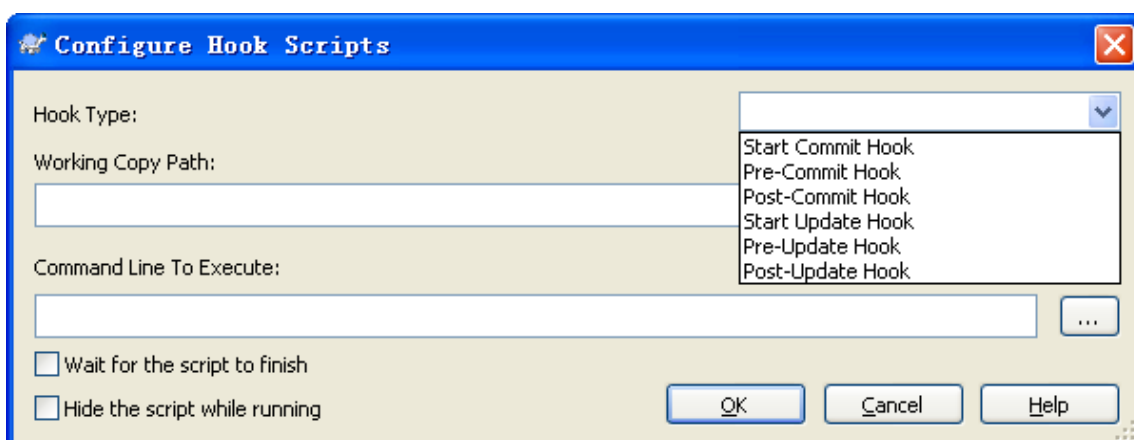
这个对话框允许你指定当特定 Subversion 动作执行时，自动执行的钩子脚本。

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 1.50. 设置对话框，配置钩子脚本页面**

要增加钩子脚本，直接点击 增加 ，然后输入脚本即可。



现在有六种钩子脚本类型可用

开始提交

在提交对话框之前调用。当钩子修改受版本控制的文件，影响了提交的文件列表的 时



候适用。

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。

命令行可以包含被 TortoiseSVN 填写的几个参数。这些参数依赖于调用了什么脚本。

开始提交

%PATHS%

提交之前

%PATHS%%SELECTEDPATHS%

提交之后

%SELECTEDPATHS%%REVISION%%ERROR%

开始更新

%PATHS%

更新之前

%PATHS%

更新之后

%PATHS%%REVISION%%ERROR%

每个变量的含义如下：

%PATHS%

当操作开始时选择的路径。例如当启动提交对话框是在资源管理器中选择的路径。如果选择了多个路径，它们用\*自负隔开。

%SELECTEDPATHS%

我们在提交对话框内选择的路径。如果选择了多个路径，那么它们用\*字符分割。

%REVISION%

在提交完成后的版本库的版本

%ERROR%

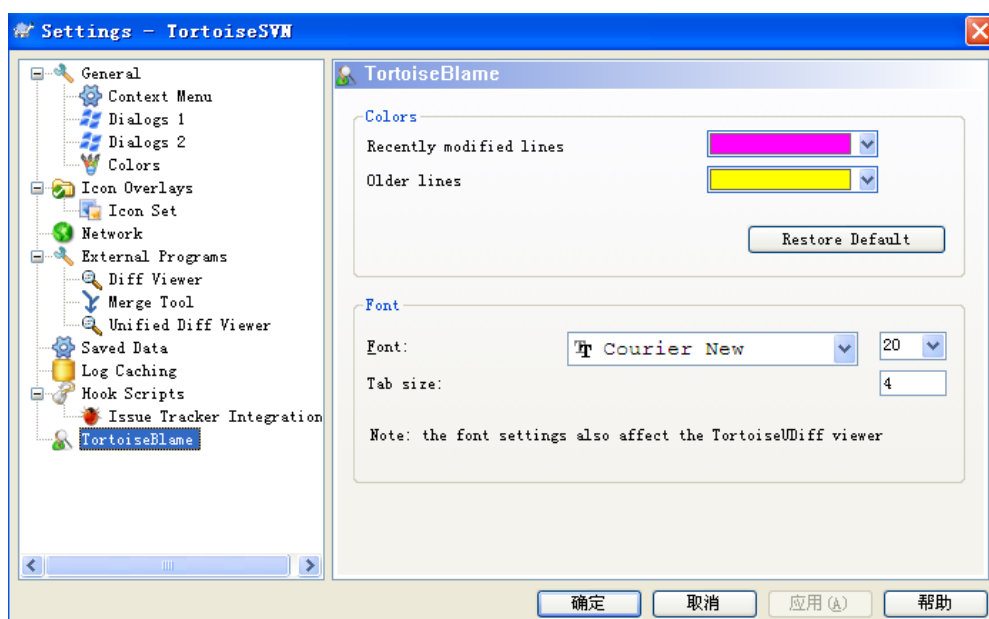
如果操作成功，它是空的，否则操作失败时就显示错误信息。

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。

通常脚本运行时，你会想隐藏丑陋的控制台窗口，所以默认选择运行时隐藏脚本。为了调试，你可能想观察控制台窗口的输出。

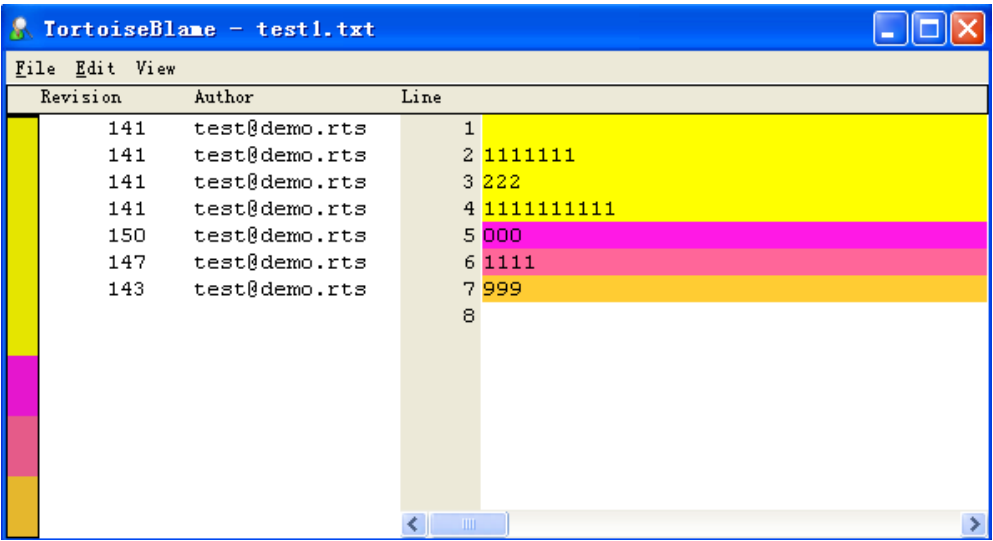
## 1.26.9. 设置TortoiseBlame显示

如图1.51 设置TortoiseBlame的界面



colors一栏是用来设置在TortoiseBlame中显示文字的背景色

如图1.52



font一栏用来设置在TortoiseBlame中显示文字的字体