TortoiseSVN

Um aplicativo do Subversion para Windows

Version 1.12

Stefan Küng
Lübbe Onken
Simon Large

TortoiseSVN: Um aplicativo do Subversion para Windows: Version 1.12

por Stefan Küng, Lübbe Onken, e Simon Large
Tradução: tradudor-créditos

data de publicação 2019/03/11 21:25:21 (r28524)

Índice

Prefácio xi
1. O que é o TortoiseSVN? xi
2. Caracterísiticas do TortoiseSVN xi
3. Licença xii
4. Desenvolvimento xii
4.1. História do TortoiseSVN xii
4.2. Agradecimentos xiii
5. Guia de leitura xiii
6. Termos usados neste documento xiv
1. Começando 1
1.1. Instalando TortoiseSVN 1
1.1.1. Requerimentos do sistema 1
1.1.2. Instalação 1
1.2. Conceitos Básicos 1
1.3. Vá em frente para um Teste 2
1.3.1. Criando um Repositório 2
1.3.2. Importando um Projeto 2
1.3.3. Obtendo Uma Cópia de Trabalho 3
1.3.4. Alterando 4
1.3.5. Adicionando Mais Arquivos 4
1.3.6. Vendo o Histórico do Projeto 5
1.3.7. Desfazendo Alterações 5
1.4. Continuando... 6
2. Conceitos básicos do Controle de Versão 7
2.1. O Repositório 7
2.2. Modelo de controles 7
2.2.1. O problema do compartilhamento de arquivo 8
2.2.2. A solução Alocar-Modificar-Desalocar 8
2.2.3. A solução Copiar-Modificar-Unificar 9
2.2.4. O que o Subversion faz? 12
2.3. Subversion em Ação 12
2.3.1. Cópias de Trabalho 12
2.3.2. URLs do Repositório 14
2.3.3. Revisões 14
2.3.4. Como Cópia de Trabalho Acompanham o Repositório 16
2.4. Resumo 17
3. O Repositório 18
3.1. Criação do Repositório 18
3.1.1. Criando um Repositório com a linha de comando do cliente 18
3.1.2. Criando um Repositório com o TortoiseSVN 18
3.1.3. Acesso Local para o Repositório 19
3.1.4. Acessando um Repositório em uma Rede Compartilhada 19
3.1.5. Leiaute do Repositório 19
3.2. Cópia de Segurança do Repositório 21
3.3. Rotinas de eventos no servidor 21
3.4. Vínculos externos 22
3.5. Acessando o Repositório 22
4. Guia do Uso Diário 24
4.1. Características Gerais 24
4.1.1. Sobreposição dos Ícones 24
4.1.2. Menus do Contexto 24
4.1.3. Arrastar e Soltar 26
4.1.4. Atalhos Comuns 27
4.1.5. Autenticação 27
4.1.6. Maximizando Janelas 28

4.2. Importando Dados Para Um Repositório 29
4.2.1. Importar 29
4.2.2. Importando na Pasta 30
4.2.3. Arquivos Especiais 30
4.3. Obtendo Uma Cópia de Trabalho 30
4.3.1. Profundidade da Obtenção 31
4.4. Submetendo Suas Alterações Para o Repositório 33
4.4.1. A Janela de Submissão 33
4.4.2. Lista de Alterações 36
4.4.3. Enviar somente partes dos arquivos 36
4.4.4. Excluindo Itens de uma Lista de Submissões 37
4.4.5. Registro de Mensagens de Submissão 37
4.4.6. Progresso da Submissão 39
4.5. Atualizar sua Cópia de Trabalho com mudanças feitas por outros 40
4.6. Resolvendo Conflitos 42
4.6.1. Conflitos de Arquivo 42
4.6.2. Conflitos de Propriedade 43
4.6.3. Conflitos de Estrutura 43
4.7. Obtendo Informações de Estado 46
4.7.1. Sobreposição dos Ícones 46
4.7.2. Situação Detalhada 47
4.7.3. Estado Local e Remoto 49
4.7.4. Visualizar diferenças 51
4.8. Lista de Alterações 51
4.9. Shelving 53
4.10. Janela de Revisão de Registro 55
4.10.1. Invocando a Janela de Histórico de Revisão 56
4.10.2. Histórico de Ações de Revisão 57
4.10.3. Recuperando Informações Adicionais 58
4.10.4. Obtendo mais mensagens de log 64
4.10.5. Revisão da Cópia de Trabalho Atual 65
4.10.6. Combinar Recursos Monitorados 65
4.10.7. Alterando a Mensagem de Auditoria e o Autor 66
4.10.8. Filtrando Mensagens de Log 66
4.10.9. Informação Estatística 68
4.10.10. Modo desconectado 71
4.10.11. Atualizando a Visualização 71
4.11. Visualizando as Diferenças 72
4.11.1. Diferenças do Arquivo 72
4.11.2. Opções de Quebra de Linha e Espaços em Branco 73
4.11.3. Comparando Diretórios 73
4.11.4. Comparando Imagens Utilizando o TortoiseIDiff 75
4.11.5. Diffing Office Documents 76
4.11.6. Ferramentas de Diferenciação/Combinação Externas 76
4.12. Adicionando Novos Arquivos e Diretórios 77
4.13. Copiando/Movendo/Renomeando Arquivos e Diretórios 77
4.14. Ignorando Arquivos e Diretórios 78
4.14.1. Padrões de Filtro na Lista de Arquivos Ignorados 79
4.15. Apagando, Movendo e Renomeando 80
4.15.1. Apagando arquivos e diretórios 81
4.15.2. Movendo arquivos e diretórios 82
4.15.3. Procedimento em caso de conflito com o nome do arquivo 83
4.15.4. Reparando Renomeação de Arquivos 83
4.15.5. Apagando Arquivos não Controlados 83
4.16. Desfazendo Alterações 83
4.17. Limpar 85
4.18. Configurações do Projeto 86
4.18.1. Propriedades do Subversion 86

4.18.2. Propriedades do Projeto TortoiseSVN 90
4.18.3. Editores de Propriedades 96
4.19. Itens Externos 102
4.19.1. Diretórios Externos 102
4.19.2. Arquivos Externos 104
4.19.3. Criando externos via drag e drop 105
4.20. Ramificando / Rotulando 105
4.20.1. Criando um Ramo ou Rótulo 105
4.20.2. Outras maneiras de criar um ramo ou destino 108
4.20.3. Para Obter ou Alternar 108
4.21. Unificando 109
4.21.1. Unificar um Intervalo de Revisões 110
4.21.2. Combinando Duas Árvores Diferentes 112
4.21.3. Opções de Combinação 113
4.21.4. Revisando os Resultados da Combinação 114
4.21.5. Histórico de combinações 115
4.21.6. Handling Conflicts after Merge 115
4.21.7. Funcionalidade de Manutenção de Ramo 116
4.22. Bloqueando 117
4.22.1. Como o Bloqueio Funcionada no Subversion 118
4.22.2. Obtendo uma trava 119
4.22.3. Liberando uma trava 119
4.22.4. Verificando a Situação do Bloqueio 120
4.22.5. Tornando arquivos não-bloqueados somente leitura 120
4.22.6. Os Scripts de Gancho de Bloqueio 121
4.23. Criando e Aplicando Correções 121
4.23.1. Cirando um Arquivo de Correções 121
4.23.2. Aplicando um Arquivo de Correções 123
4.24. Quem Mudou Qual Linha? 123
4.24.1. Autoria por Arquivos 124
4.24.2. Diferenças de Autoria 126
4.25. O Navegador de Repositório 126
4.26. Gráfico de Revisões 129
4.26.1. Nós do Gráfico de Revisões 130
4.26.2. Mudando a Visão 130
4.26.3. Usando o Gráfico 132
4.26.4. Atualizando a Visualização 133
4.26.5. Pruning Trees 133
4.27. Exportando um Cópia de Trabalho do Subversion 133
4.27.1. Removendo uma cópia de trabalho do controle de versão 135
4.28. Realocando uma cópia de trabalho 135
4.29. Integração com Sistemas de Rastreamento de Erros / Rastreadores de Reportes 136
4.29.1. Adicionando Identificadores de Reporte para as Mensagens de Auditoria 137
4.29.2. Obtendo Informações do Rastreador de Reportes 140
4.30. Integração com visualizadores de repositórios locados na web 141
4.31. Configurações do TortoiseSVN 142
4.31.1. Configurações Gerais 142
4.31.2. Configurações do Gráfico de Revisões 151
4.31.3. Configurações de Sobreposição dos Ícones 153
4.31.4. Configurações de rede 157
4.31.5. Configuração de Programa Externo 159
4.31.6. Dados de configuração salvos 164
4.31.7. Cache de Log 165
4.31.8. Client Side Hook Scripts 168
4.31.9. Configurações TortoiseBlame 173
4.31.10. TortoiseUDiff Settings 174
4.31.11. Exporting TSVN Settings 175
4.31.12. Configurações Avançadas 175

4.32. Passo Final 180
5. Project Monitor 181
5.1. Adding projects to monitor 181
5.2. Monitor dialog 182
5.2.1. Principais operações 182
6. SubWCRev 184
6.1. The SubWCRev Command Line 184
6.2. Alteração da palavra-chave 186
6.3. Exemplo de Palavra-chave 187
6.4. COM interface 189
7. IBugtraqProvider interface 192
7.1. Naming conventions 192
7.2. The IBugtraqProvider interface 192
7.3. The IBugtraqProvider2 interface 193
A. Frequently Asked Questions (FAQ) 197
B. Como eu faço... 198
B.1. Mover/copiar vários arquivos de uma só vez 198
B.2. Usuario forçando para entrar uma messagem de log 198
B.2.1. Rotinas de eventos no servidor 198
B.2.2. Propriedades do projeto 198
B.3. Atualizar os arquivos selecionados a partir do repositório 198
B.4. Reverter (Desfazer) revisões no repositório 198
B.4.1. Usar a janela de auditoria de revisão 199
B.4.2. Use a janela de combinação 199
B.4.3. Use `svndumpfilter` 199
B.5. Compare duas revisões de um arquivo ou pasta 199
B.6. Inclua um subprojeto em comum 200
B.6.1. Use svn:externals 200
B.6.2. Use uma cópia de trabalho aninhada 200
B.6.3. Use um local relativo 200
B.6.4. Adicione o projeto para o repositório 201
B.7. Criar atalho para um repositório 201
B.8. Ignorar arquivos que já estão versionados 201
B.9. Unversion a working copy 202
B.10. Remover uma cópia de trabalho 202
C. Dicas Úteis para Administradores 203
C.1. Deploy TortoiseSVN via group policies 203
C.2. Redirecionar a verificação de atualização 203
C.3. Alternado `SVN_ASP_DOT_NET_HACK` variável do ambiente 204
C.4. Desabilitar opções do menu de contexto 204
D. Automatizando o TortoiseSVN 207
D.1. Comandos do TortoiseSVN 207
D.2. Controlador de URL do Tsvncmd 213
D.3. Comandos do TortoiseIDiff 214
D.4. Comandos TortoiseUDiff 214
E. Command Line Interface Cross Reference 216
E.1. Convenções e regras básicas 216
E.2. Comandos do TortoiseSVN 216
E.2.1. Obter 216
E.2.2. Atualizar 216
E.2.3. Atualizar para Revisão 217
E.2.4. Submeter 217
E.2.5. Diff 217
E.2.6. Log 218
E.2.7. Procurar por Modificações 218
E.2.8. Gráfico de Revisões 218
E.2.9. Navegador de 218
E.2.10. Conflitos 218

E.2.11. Resolvido 219
E.2.12. Renomear 219
E.2.13. Apagar 219
E.2.14. Reverter 219
E.2.15. Limpar 219
E.2.16. Obter bloqueio 219
E.2.17. Liberar bloqueio 219
E.2.18. Ramificar/Rotular... 220
E.2.19. Alternar 220
E.2.20. Combinar 220
E.2.21. Exportar 220
E.2.22. Reposicionar 220
E.2.23. Criar repositó aqui 221
E.2.24. Adicionar 221
E.2.25. Importar 221
E.2.26. Autoria 221
E.2.27. Adicionar à lista de ignorados 221
E.2.28. Criar Correção 221
E.2.29. Aplicar correção 221
F. Detalhes da Implementação 222
F.1. Sobreposição dos Ícones 222
G. Pacotes de Idiomas e Corretores Ortográficos 224
G.1. Pacotes de Idioma 224
G.2. Corretor ortográfico 224
Glossário 226
Índice Remissivo 229

Lista de Figuras

1.1. O menu do TortoiseSVN para diretórios não controlados 2
1.2. A Janela de Importação 3
1.3. Visualizador das Diferenças do Arquivo 4
1.4. A Janela de Log 5
2.1. Um típico sistema Cliente/Servidor 7
2.2. O problema a ser evitado 8
2.3. A solução Alocar-Modificar-Desalocar 9
2.4. A solução Copiar-Modificar-Unificar 10
2.5. ...Continuando com Copiar-Modificar-Unificar 11
2.6. O Sistema de Arquivos do Repositório 13
2.7. O Repositório 15
3.1. O menu do TortoiseSVN para diretórios não controlados 18
4.1. Explorer mostra os ícones sobrepostos 24
4.2. Menu do contexto para diretórios controlados 25
4.3. Atalho no menu arquivo do Explorer em um diretório controlado 26
4.4. Menu de quando se clica com o botão direito e se arrasta um diretório que está sob o controle de versão. 27
4.5. Janela de Autenticação 28
4.6. A janela de Obtenção 31
4.7. A janela de Submissão 34
4.8. A Janela de Submissão Com Corretor Ortográfico 38
4.9. A janela de Progresso mostra uma submissão em progresso 39
4.10. Janela de progresso mostrando atualização terminada 40
4.11. Explorer mostra os ícones sobrepostos 46
4.12. Página de propriedade do explorador, aba do Subversion 48
4.13. Procurar por Modificações 49
4.14. Janela de Submissão com Lista de Alterações 52
4.15. Shelve dialog 54
4.16. Unshelve dialog 55
4.17. A Janela de Histórico de Revisão 56
4.18. O Menu de Contexto do Painel Superior da Janela de Auditoria da Revisão 58
4.19. A Janela de Configurações do Colaborador de Código 61
4.20. Menu de Contexto do Painel Superior para 2 Revisões Selecionadas 61
4.21. O Painel Inferior da Janela de Auditoria com Menu de Contexto 62
4.22. The Log Dialog Bottom Pane with Context Menu When Multiple Files Selected. 63
4.23. A Janela de Auditoria Mostrando as Revisões de Rastreamento de Combinação 65
4.24. Histograma de Submissões-por-autor 68
4.25. Gráfico de Pizza das Submissões-por-Autor 69
4.26. Gráfico de Submissões-por-data 70
4.27. Ir para Janela de Desconectado 71
4.28. A Janela de Comparação de Revisões 74
4.29. O visualizar de diferenças de imagem 75
4.30. Explorar o menu de contexto para arquivos não versionados 77
4.31. Menu de quando se clica com o botão direito e se arrasta um diretório que está sob o controle de versão. 78
4.32. Explorar o menu de contexto para arquivos não versionados 79
4.33. Menu de contexto do Explorerer para arquivos controlados 81
4.34. Janela de Reversão 84
4.35. The Cleanup dialog 85
4.36. página de propriedades do Subversion 86
4.37. Adicionando propriedades 88
4.38. Property dialog for hook scripts 92
4.39. Property dialog boolean user types 93
4.40. Property dialog state user types 93
4.41. página da propriedade svn:externals 96

4.42. página da propriedade svn:keywords 97
4.43. página da propriedade svn:eol-style 97
4.44. página da propriedade tsvn:bugtraq 98
4.45. Página da propriedade do tamanho das mensagens de auditoria 99
4.46. Página da propriedade idioma 99
4.47. página da propriedade svn:mime-type 100
4.48. página da propriedade svn:needs-lock 100
4.49. página da propriedade svn:executable 100
4.50. Property dialog merge log message templates 101
4.51. A Janela de Ramificação/Rotulação 106
4.52. A Janela de Troca 109
4.53. O Assistente de Combinação - Selecionar Faixa de Revisões 111
4.54. O Assistente de Combinação - Árvore de Combinação 113
4.55. The Merge Conflict Dialog 116
4.56. The Merge Tree Conflict Dialog 116
4.57. The Merge-All Dialog 117
4.58. A Janela de Bloqueio 119
4.59. A Janela de Verificação de Modificações 120
4.60. A Janela de Criação de Correções 122
4.61. A Janela de Anotação / Autoria 124
4.62. TortoiseBlame 125
4.63. O Navegador de Repositório 127
4.64. Um Gráfico de Revisões 129
4.65. A Janela Exportar de uma URL 134
4.66. A Janela de Realocação 135
4.67. The Bugtraq Properties Dialog 137
4.68. Janela de exemplo de uma consulta de um rastreador de reportes 141
4.69. Janela de Configurações, Opção Geral 143
4.70. Janela de Configurações, Opção Menu de Contexto 145
4.71. Janela de Configurações, Opção Janelas 1 146
4.72. Janela de Configurações, Opção Janelas 2 147
4.73. Janela de Configurações, Opção Janela 3 149
4.74. Janela de Configurações, Opção Cores 150
4.75. Janela de Configurações, Opção Gráfico de Revisões 151
4.76. Janela de Configurações, Opção Cores do Gráfico de Revisões 152
4.77. Janela de Configurações, Opção Sobreposição de Ícones 153
4.78. Janela de Configurações, Opção Definição de Ícones 156
4.79. Janela de Configurações, Opção Manipuladores de Ícones 157
4.80. Janela de Configurações, Opção Rede 158
4.81. Janela de Configurações, Opção Visualizador de Diferenças 159
4.82. Janela de Configurações, Opção Diferenciação/Combinação Avançado 163
4.83. Janela de Configurações, Opção Dados Salvos 164
4.84. Janela de Configurações, Opção Cache da Auditoria 165
4.85. Janela de Configurações, Opção Estatísticas de Cache da Auditoria 167
4.86. Janela de Configurações, Opção Scripts de Gancho 168
4.87. Janela de Configurações, Opção Configuração de Scripts de Gancho 169
4.88. Janela de Configurações, Opção Integração com Rastreador de Reportes 172
4.89. Janela de Configurações, Opção Tortoise Autoria 173
4.90. The Settings Dialog, TortoiseUDiff Page 174
4.91. The Settings Dialog, Sync Page 175
4.92. Barra de tarefas com agrupamento padrão 177
4.93. Barra de tarefas com agrupamento por repositório 177
4.94. Barra de tarefas com agrupamento por repositório 177
4.95. Taskbar grouping with repository color overlays 178
5.1. The edit project dialog of the project monitor 181
5.2. The main dialog of the project monitor 182
C.1. Dialogo do commit, mostrando a notificação atualizada 203

Lista de Tabelas

2.1. URLs de Acesso ao Repositório 14
4.1. Revisãio Pinado 107
6.1. List of available command line switches 185
6.2. List of SubWCRev error codes 185
6.3. Lista de palavras-chave disponíveis 186
6.4. COM/automation methods supported 189
C.1. Entradas do menu e seus valores 204
D.1. Lista de comandos e opções disponíveis 208
D.2. Lista de opções disponíveis 214
D.3. Lista de opções disponíveis 214

Prefácio

Controle de versão é a arte de administrar as mudanças das informações. Isto é uma ferramenta crítica para programadores, que normalmente gastam horas fazendo pequenas modificações em seus aplicativos e então desfazem ou verificam algumas dessas modificações no dia seguinte. Imagine uma equipe de vários desenvolvedores trabalhando juntos - e talvez simultaneamente em mesmos arquivos! - e você precisa ver porque um bom controle é necessário para *controlar uma possível desordem.*

1. O que é o TortoiseSVN?

O TortoiseSVN é um cliente open-source gratuito para a plataforma Windows voltado para o sistema de controle de versão *Apache™ Subversion®*. Ou seja, o TortoiseSVN administra arquivos e diretórios no decorrer do tempo. Os arquivos são armazenados em um *repositório* central. O repositório parece-se muito com um servidor de arquivos comum, exceto pelo fato que ele consegue se lembrar de toda e qualquer alteração já feita em algum momento nos seus arquivos e diretórios. Isso permite a você recuperar versões antigas de seus arquivos e examinar o histórico de como, quando e por quem seus dados foram modificados. Esse é o motivo pelo qual muitas pessoas pensam no Subversion, e em sistemas de controle de versão em geral, como uma espécie de “máquina do tempo”.

Alguns sistemas de controle de versão também são um aplicativo de gerenciamento de configuração (SCM). Esses sistemas são especificamente adaptados para controlar estruturas de código fonte, e tem muitas características de um aplicativo específico de desenvolvimento - como um aplicativo para uma linguagem de programação específica, or fornecendo ferramentas de construção de software. Subversion, entretanto, não é um desses sistemas; é um sistema genérico que pode ser usado para administrar *qualquer* conjunto de arquivos, incluindo código fonte.

2. Caracterísiticas do TortoiseSVN

O que faz do TortoiseSVN um bom aplicativo cliente para Subversion? Aqui está uma pequena lista de recursos.

Interface integrada

TortoiseSVN integra-se perfeitamente ao shell do Windows (ou seja, o Explorer). Isto significa que você pode continuar trabalhando com as ferramentas com as quais você já está familiarizado. E você não tem que mudar para uma aplicação diferente cada vez que precisar das funções de controle de versão.

E você não está limitado a usar o Windows Explorer; os menus de contexto do TortoiseSVN funcionam em muitos outros gerenciadores de arquivos, e também na caixa de diálogo Arquivo/Abrir que é comum na maioria dos aplicativos padrão Windows. Você deve, entretanto, ter em mente que o TortoiseSVN é intencionalmente desenvolvido como uma extensão para o Windows Explorer. Assim, é possível que em outras aplicações, a integração não seja tão completa e, por exemplo, as sobreposições de ícones podem não ser exibidas.

Sobreposição dos ícones
: A situação de cada arquivo e diretório controlado é indicado por uma pequena sobreposição de ícones. O que permite a você ver rapidamente qual é a situação da sua cópia de trabalho.

Interface Gráfica de Usuário
: Quando você lista as alterações em um arquivo ou pasta, você pode clicar em uma revisão para ver os comentários para aquela submissão. Você também pode ver uma lista de arquivos alterados - basta clicar duas vezes em um arquivo para ver exatamente o que mudou.

  A caixa de diálogo de submissões lista todos os itens que serão incluídos em uma submissão, e cada item tem uma caixa de seleção para que você possa escolher os itens que você deseja incluir. Arquivos sem versão também podem ser listados, no caso de você ter esquecido de adicionar aquele novo arquivo.

Fácil acesso aos comandos do Subversion
: Todos os comandos do Subversion estão disponíveis nos menus do explorer. TortoiseSVN adiciona seu próprio submenu.

Uma vez que TortoiseSVN é um aplicativo cliente do Subversion, também gostaríamos de mostrar a você algumas das funcionalidades do Subversion:

Controle de diretório
: CVS somente mantém o histórico de alterações de arquivos individuais, mas Subversion usa um controle "virtual" de sistema de arquivos que mantém o histórico de toda a estrutura de diretório ao longo do tempo. Arquivos *e* diretórios são controlados. E como resultado, temos verdadeiros comandos para **mover** e **copiar** arquivos e diretórios.

Submissão atômica
: Cada submissão é enviada completamente para o repositório, ou não é enviado nada. Isto permite aos desenvolvedores construir e submeter as alterações em partes coesas.

Metadados controlados
: Cada arquivo e diretório possue um conjunto de "propriedades" invisíveis. Você pode inventar e gravar qualquer conjunto de chave/valor que desejar. Propriedades são controladas ao longo do tempo, exatamente como o conteúdo dos arquivos.

Escolha das camadas da rede
: Subversion tem uma noção abstrata de acesso ao repositório, tornando fácil para as pessoas desenvolverem novos mecanismos de rede. O servidor de rede avançado do Subversion é um módulo para o servidor web Apache, do qual expõe uma variante do HTTP chamada WebDAV/DeltaV. Isto dá ao Subversion uma grande vantagem em estabilidade e interoperabilidade, e provê várias funcionalidades chave de graça: autenticação, autorização, compressão, e navegação no repositório, por exemplo. Uma característica menor, um processo servidor autônomo do Subversion também está disponível. Este servidor exterioriza um protocolo específico que pode ser facilmente encapsulado sobre o protocolo ssh.

Manipulação consiste de dados
: Subversion apresenta as diferenças de arquivos usando um algoritmo de comparação binária, que funciona igualmente para arquivos texto (compreensíveis) and binários (ilegíveis). Ambos os tipos de arquivos são gravados compactados da mesma forma no repositório, e as diferenças são trasmitidas em ambas as direções através da rede.

Ramificação e Rotulação eficiente
: Os recursos necessários para ramificar e rotular não é propocional ao tamanho do projeto. Subversion cria ramos e rótulos simplesmente copiando o projeto, usando um mecanismo parecido ao hard-link. Deste modo estas operações são realizadas rapidamente sem variação de tempo, e consomem muito pouco espaço no repositório.

3. Licença

TortoiseSVN é um projeto Open Source desenvolvido sob a licença GNU General Public License (GPL). É gratuito para baixar e de uso gratuito, pessoalmente ou comercialmente, em qualquer número de computadores.

Embora a maioria das pessoas apenas fazem o download do instalador, você ainda possui acesso ao código fonte deste programa. Você pode explorá-lo neste link: https://sourceforge.net/p/tortoisesvn/code/HEAD/tree/. A versão em desenvolvimento corrente está localizada em `/trunk/`, e as versões já lançadas estão em `/tags/`.

4. Desenvolvimento

TortoiseSVN e Subversion são desenvolvidos por uma comunidade de pessoas que estão trabalhando nestes projetos. Eles vêm de diferentes países pelo mundo, trabalhando juntos para criar um ótimo software.

4.1. História do TortoiseSVN

Em 2002, Tim Kemp achou que o Subversion era um sistema de controle de versão muito bom mas que faltava uma boa interface gráfica GUI. A idéia para um programa cliente do Subversion integrado ao shell do Windows foi

inpirado num programa similar para CVS de nome TortoiseCVS. Tim estudou o código fonte do TortoiseCVS e o usou como base para o TortoiseSVN. Ele então começou o projeto, registrando o domínio `tortoisesvn.org` e colocando o código fonte online.

Por volta daquela época, Stefan Küng estava procurando por um bom e gratuito sistema de controle de versão, encontrando assim o Subversion e o código do TortoiseSVN. Como o TortoiseSVN ainda não estava pronto para o uso, ele se juntou ao projeto e começou a programar. Ele, em pouco tempo, reescreveu grande parte do código existente e começou a adicionar comandos e features, até o ponto em que nada do código original restou.

À medida que o Subversion se tornou mais estável, mais e mais usuários foram atraídos, os quais também começaram a usar o TortoiseSVN como seu programa cliente Subversion. A base de usuários cresceu rapidamente (e ainda está crescendo todos os dias). Foi então que Lübbe Onken ofereceu ajuda com alguns belos ícones e a logomarca para o TortoiseSVN. Ele agora toma conta do website e administra muitas traduções.

Com o tempo, outros sistemas de controle de versão ganharam seus próprios clientes do Tortoise, o que passou a causar um problema com a sobreposição de ícones (icon overlays) no Windows Explorer: o número de sobreposições é limitado e mesmo um único cliente do Tortoise pode facilmente exceder esse limite. É quando Stefan Küng implementou o componente TortoiseOverlays, que permite a todos os clientes Tortoise usarem as mesmas sobreposições de ícones. Agora, todos os clientes open-source do Tortoise e mesmo alguns não-Tortoise usam esse componente compartilhado.

4.2. Agradecimentos

Tim Kemp
: por iniciar o projeto TortoiseSVN

Stefan Küng
: pelo trabalho manual para tornar o TortoiseSVN o que é agora, e sua liderança no projeto

Lübbe Onken
: pelos belos ícones, logomarca, encontrar erros, traduções e gerenciar as traduções

Simon Large
: por atualizar a documentação

Stefan Fuhrmann
: pelo cache de log e o gráfico de revisões

O livro Subversion
: por ensinar sobre o Subversion e pelo capítulo 2 de onde copiamos

O estilo de projeto Tigris
: pelos estilos da documentação de onde copiamos

Nossos Colaboradores
: pelos patches, relatório de erros e novas idéias, e por ajudar outros, respondendo perguntas na nossa lista de discussão

Nossos Contribuidores
: por muitas horas de diversão com as músicas que nos enviaram

5. Guia de leitura

Este livro é escrito para o pessoal letrado em computação que querem usar o Subversion para administrar seus dados, mas preferem usar uma cliente GUI do que usar um cliente por linhas de comando. TortoiseSVN é uma extensão do shell do windows e se supõe que o usuário esteja familiarizado com o windows explorer e saiba usá-lo.

Este Prefácio explica o que TortoiseSVN é, um pouco sobre o projeto TortoiseSVN, a comunidade de pessoas que trabalham nele, as condições de licenciamento para seu uso e sua distribuição.

O Capítulo 1, *Começando*explica como instalar TortoiseSVN no seu computador, e como começar a usá-lo de imediato.

Em Capítulo 2, *Conceitos básicos do Controle de Versão* damos uma introdução sobre o controle de revisão *Subversion* que é a base do TortoiseSVN. Isto é uma cópia da documentação do projeto Subversion e explica as diferentes formas de controlar versão, e como Subversion funciona.

O capítulo Capítulo 3, *O Repositório* explica como configurar um repositório local, útil para testar o Subversion e o TortoiseSVN usando um único PC. Ele também explica um pouco sobre administração de repositório, assunto relevante também para repositórios localizados em um servidor.

O Capítulo 4, *Guia do Uso Diário* é a parte mais importante pois explica todas as principais características do TortoiseSVN e como usá-las. Com o formato de tutorial, inicia explicando a função de obter uma cópia de trabalho, depois como modificar a cópia de trabalho, então submeter as modificações, etc. E então continua com os tópicos avançados.

The Capítulo 5, *Project Monitor* explains how you can monitor your Subversion projects so you don't miss important commits from your other team members.

Capítulo 6, *SubWCRev* é um outro aplicativo que acompanha TortoiseSVN que permite extrair informações da cópia de trabalho e escrever isto em arquivos.. Isto é útil para incluir informações sobre a construção dos seus projetos.

A seção Apêndice B, *Como eu faço...* responde as dúvidas comuns sobre a execução das tarefas que não estão explicitamente descritas em outras seções.

A parte Apêndice D, *Automatizando o TortoiseSVN* mostra como a as janelas do TortoiseSVN podem ser chamadas através da linha de comando. Isso pode ser útil em scripts que precisam da interação do usuário.

O Apêndice E, *Command Line Interface Cross Reference* mostra a relação entre os comandos do TortoiseSVN e seus equivalentes na linha de comando do aplicativo Subversion `svn.exe`.

6. Termos usados neste documento

Para facilitar a leitura do documento, o nome de todas as telas e Menus do TortoiseSVN estão em destaque. A Janela de Log por exemplo.

As opções de menu estão indicadas com uma seta. TortoiseSVN → Mostrar Log significa: selecionar *Mostrar Log* no menu *TortoiseSVN*.

Quando referenciar um menu específico dentro de uma das janelas do TortoiseSVN, será usado algo como: Menu Específico → Salvar como ...

Os Botões da Interface do Usuário são indicados como: Clique em OK para continuar.

Ações de Usuários são indicados por uma fonte em negrito. **Alt+A**: pressione a tecla **Alt** no seu teclado e enquanto a aperta, pressione também a tecla **A**. Right drag: aperte o botão direito do mouse e enquanto segura, *arraste* os itens para o novo local.

Saídas de vídeo e entradas de dados são indicadas com uma fonte `different`.

Importante

Detalhes importantes são indicados com um ícone.

Dica

Dicas tornam sua vida mais fácil.

Cuidado

Lugares onde você deve tomar cuidado com o que faz.

Atenção

Onde é preciso cuidado extremo. Corrupção de dados e outras coisas desagradáveis podem ocorrer se estes avisos forem ignorados.

Capítulo 1. Começando

Esta seção é voltada às pessoas que gostariam de saber do que se trata o TortoiseSVN e fazer um teste. Ela explica como instalar o TortoiseSVN e configurar um repositório local, e dá o passo-a-passo para a maioria das operações mais comuns.

1.1. Instalando TortoiseSVN

1.1.1. Requerimentos do sistema

O TortoiseSVN está disponível para Windows Vista ou superior, nas versões 32 e 64 bits. O instalador da versão 64-bit para Windows também inclui os arquivos necessários para a versão 32 bits. Isso significa que não é preciso instalar esta versão separadamente para poder utilizá-la nas aplicações de 32 bits.

O suporte para Windows 98, Windows ME e Windows NT4 foi descontinuado na versão 1.2.0, ao passo que o suporte para Windows 2000 e XP até o SP2 acabou na versão 1.7.0. O suporte para Windows XP com o SP3 terminou na versão 1.9.0. Entretanto, você ainda pode baixar e instalar essas versões mais antigas caso necessite delas.

1.1.2. Instalação

TortoiseSVN vem com um instalador fácil de usar. Clique duas vezes no arquivo de instalação e siga as instruções. O instalador cuidará de todo o resto. Não se esqueça de reiniciar após a instalação.

Importante

Você necessita de privilégios de Administrador para instalar o TortoiseSVN. O arquivo de instalação pedirá as credenciais de Administrador se necessário.

Pacotes de linguagens estão disponíveis, podendo traduzir a interface do TortoiseSVN para muitas diferentes línguas. Por favor, dê uma olhada em Apêndice G, *Pacotes de Idiomas e Corretores Ortográficos* para mais informações em como as instalar.

Se você tiver qualquer problema durante ou após a instalação do TortoiseSVN, por favor acesse nosso FAQ disponível em http://tortoisesvn.net/faq.html.

1.2. Conceitos Básicos

Antes de nos ocuparmos em trabalhar com arquivos reais, é importante uma visão geral de como o Subversion trabalha e os termos em que ele é usado.

O Repositório
: Subversion usa um banco de dados central que contem todos os arquivos versionados com seu histórico completo. Esse banco de dados é referenciado como *repositório*. O repositório normalmente está alocado num servidor de arquivos rodando o servidor Subversion, abastecendo clientes do Subversion (como o TortoiseSVN) quando requisitado. Se você faz backup de apenas uma coisa, faça o backup do seu repositório pois é a cópia mestre definitiva de todos os seus dados.

Cópia de Trabalho
: Aqui é onde você faz o trabalho de verdade. Todo desenvolvedor tem sua própria cópia de trabalho, algumas vezes conhecido como sandbox, no seu PC local. Você pode baixar a última versão do repositório, trabalhar nele localmente sem afetar os outros e então, quando estiver feliz com as mudanças que fez, enviá-las de volta ao repositório.

  Uma cópia de trabalho do Subversion não possui todo o histórico do projeto, mas mantem uma cópia dos arquivos como estavam no repositório antes de você começar a fazer modificações. Isto quer dizer que é fácil fazer a checagem das modificações que você realizou.

Você também tem que saber onde encontrar o TortoiseSVN, pois não há muito do que se ver no Menu Inicial. Isso é porque o TortoiseSVN é uma extensão Shell, portanto, primeiro de tudo, inicie o Windows Explorer. Clique com o botão direito numa pasta no Explorer e você deverá ver umas entradas novas no menu de contexto como este:

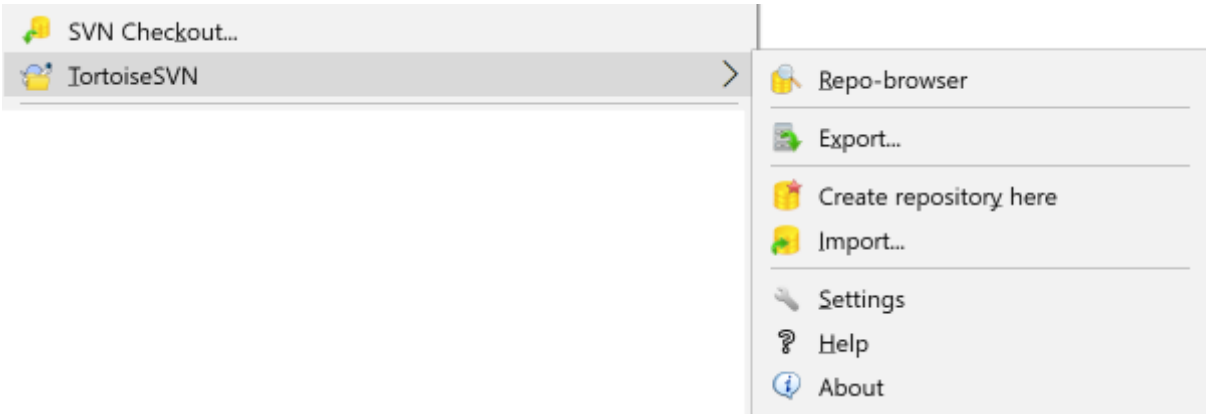


Figura 1.1. O menu do TortoiseSVN para diretórios não controlados

1.3. Vá em frente para um Teste

Esta seção mostra como experimetar as features mais usadas em um repositório pequeno. Naturalmente, não se explica tudo - este é apenas um Guia Rápido apesar de tudo. Uma vez que você estiver com tudo rodando, deverá pegar um tempinho e ler o resto deste guia, que o leva a coisas com muito mais detalhes. Também é explicado mais sobre como configurar um servidor Subversion.

1.3.1. Criando um Repositório

Para um projeto real você terá um repositório configurado em algum lugar seguro e um servidor do Subversion para ser controlado. Para os propósitos deste tutorial nós usaremos um repositório local do Subversion que tem como característica o acesso direto ao repositório criado no seu disco sem a necessidade de um servidor.

Primeiro, crie um diretório novo vazio no seu computador. Pode ser em qualquer lugar e ter qualquer nome, mas neste tutorial nós o chamaremos `C:\svn_repos`. Agora, clique com o botão direito na pasta criada e, no menu de contexto, escolha TortoiseSVN → Criar um repositório aqui.... O repositório é então criado dentro da pasta, pronto para ser usado. Também criaremos a estrutura interna padrão de pastas clicando no botão Criar estrutura de pasta

Importante

A característica do repositório local é muito útil para testes e avaliações mas exceto se você é um desenvolver individual em um único computador você deverá sempre utilizar um servidor do Subversion devidamente configurado. É tentador para uma pequena empresa evitar o trabalho de configurar um servidor e simplesmente acessar seu repositório em uma rede compartilhada. Nunca faça isso. Você perderá dados. Leia Seção 3.1.4, “Acessando um Repositório em uma Rede Compartilhada” para descobrir porque fazer isso é uma má idéia, e como configurar um servidor.

1.3.2. Importando um Projeto

Agora temos um repositório, mas está completamente vazio no momento. Suponhamos que há alguns arquivos em `C:\Projects\Widget1` que eu gostaria de adicionar. Navegue para a pasta `Widget1` pelo Explorer e clique com o botão direto sobre ela. Agora seleiona TortoiseSVN → Importar... que vai mostrar uma janela

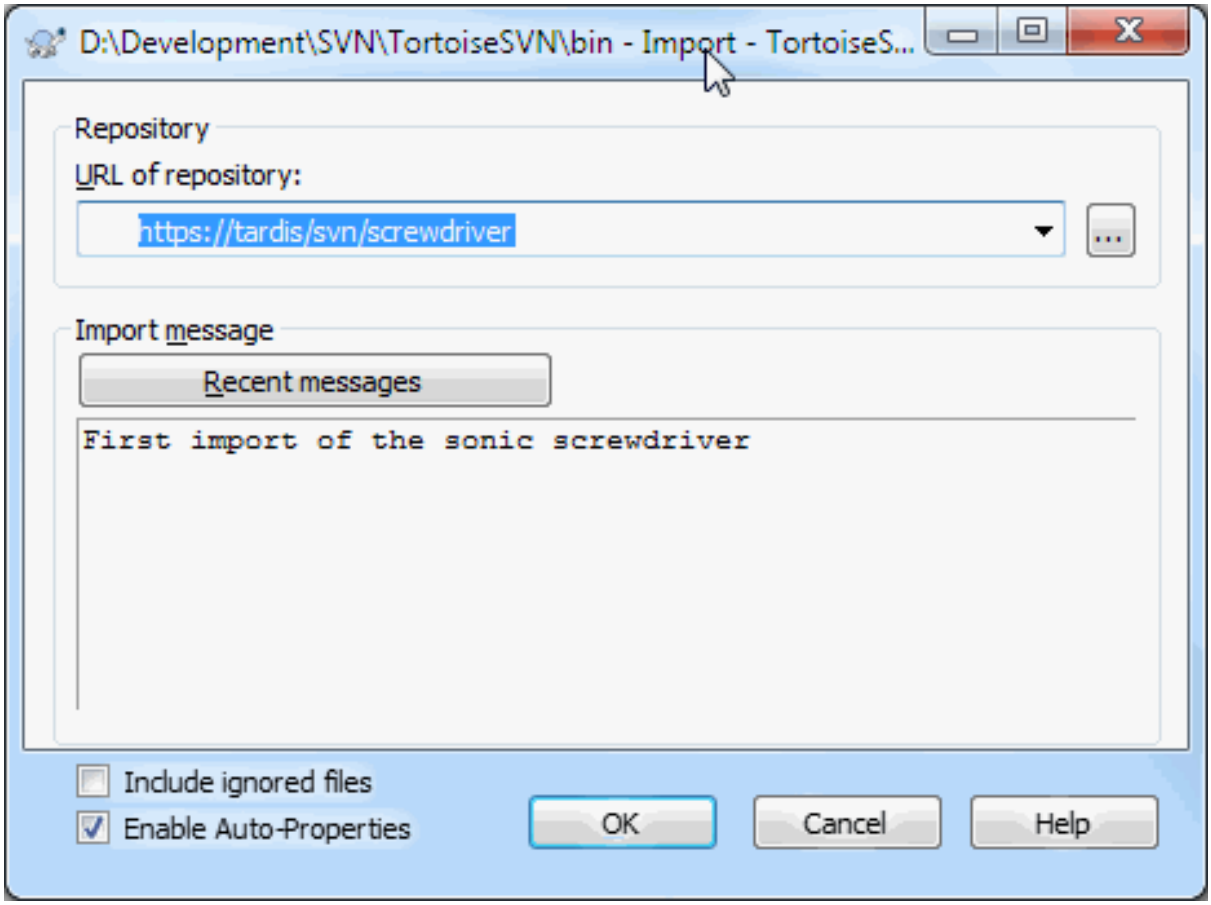


Figura 1.2. A Janela de Importação

Um repositório do Subversion é reconhecido pela URL, que nos permite especificar um repositório em qualquer lugar na Internet. Neste caso, precisamos apontar para nosso próprio repositório local que tem a URL `file:///c:/svn_repos/trunk`, ao qual adicionamos nosso próximo nome para o projeto `Widget1`. Note que há 3 barras depois de `file:` e que eles são mesmo usados.

Outra importante característica desta janela é a caixa Mensagem Importante a qual lhe permite digitar uma mensagem descrevendo o que você está fazendo. Quando você visualizar o histórico do seu projeto, essas mensagens serão um valioso guia para entender as alterações feitas e porque. Neste caso nós podemos dizer algo simples como “Importar o Projeto Componente1”. Clicar no OK e o diretório será adicionado ao seu repositório.

1.3.3. Obtendo Uma Cópia de Trabalho

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

Importante

In the default setting, the checkout menu item is not located in the TortoiseSVN submenu but is shown at the top explorer menu. TortoiseSVN commands that are not in the submenu have SVN prepended: SVN Checkout...

Você notará que a aparência deste diretório será diferente do diretório original. Cada arquivo terá uma marca verde no canto inferior esquerdo. Esses ícones de situação do TortoiseSVN estarão presentes apenas na cópia de trabalho. A situação verde indica que o arquivo não foi alterado em relação à versão do repositório.

1.3.4. Alterando

Time to get to work. In the `Widget1-Dev` folder we start editing files - let's say we make changes to `Widget1.c` and `ReadMe.txt`. Notice that the icon overlays on these files have now changed to red, indicating that changes have been made locally.

Mas o que nós fizemos? Clique com o botão direito em um dos arquivos alterados e selecione TortoiseSVN → Diferenças. A ferramenta de comparação do TortoiseSVN será aberta, mostrando exatamente quais linhas foram modificadas.

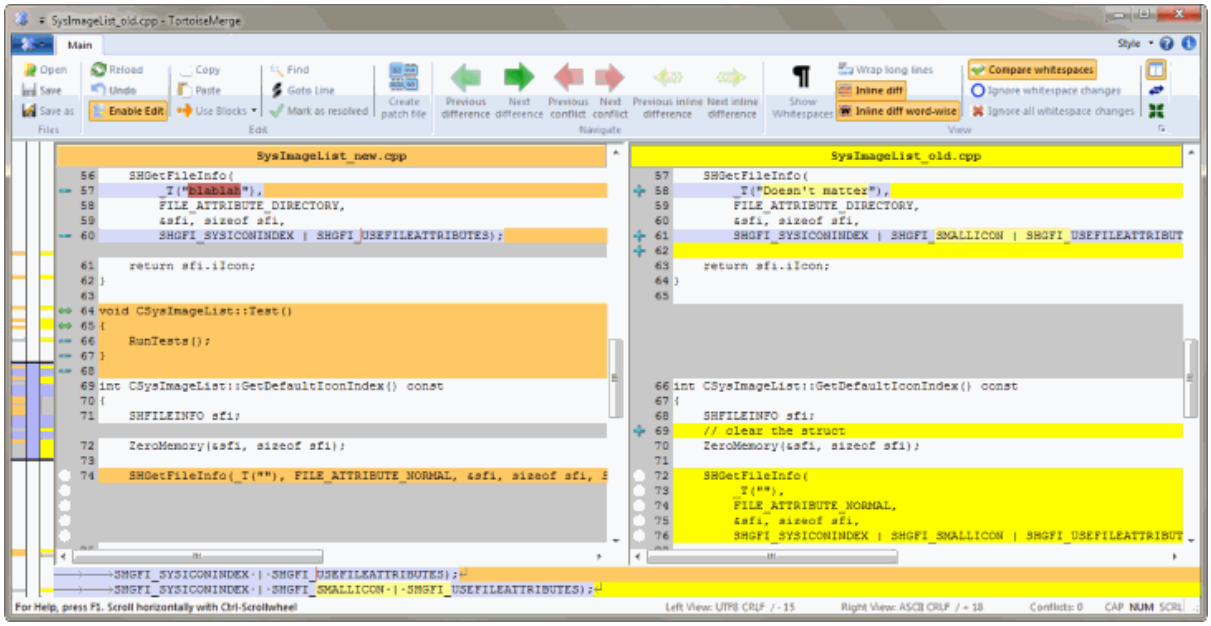


Figura 1.3. Visualizador das Diferenças do Arquivo

Bom, então nós estamos felizes com as alterações, vamos atualizar o repositório. Esta ação é referenciada como `Submissão` das alterações. Clique com o botão direito sobre o diretório `Componente1-Dev` e selecione TortoiseSVN → Submeter. A janela de submissão mostrará os arquivos alterados, cada um com uma caixa de seleção. Você pode querer escolher apenas alguns dos arquivos, mas neste caso nõs vamos submeter as alterações em ambos os arquivos. Digite uma mensagem para descrever sobre o que são as alterações e clique em OK. A janela de progresso mostrará os arquivos sendo enviados para o repositório e então está feito.

1.3.5. Adicionando Mais Arquivos

Como os desenvolvedores precisarão adicionar novos arquivos - vamos dizer que você adicionou alguma nova funcionalidade em `Extras.c` e adicionou um referência no arquivo que já existia `Makefile`. Clique com o botão direito sobre o diretório e selecione TortoiseSVN → Adicionar. A janela de Adição gora mostra para você todos os arquivos não controlados e você poderá selecionar cada um que você quer adicionar. Outra forma de adicionar arquivos poderá ser um clique com o botão direito sobre o próprio arquivo e então selecionar TortoiseSVN → Adicionar.

Agora quando você for submeter o diretório, o novo arquivo aparecerá como *Adicionado* e o arquivo já existente como *Modificado*. Note que você pode dar um duplo clique sobre o arquivo modificado e verificar exatamente as alterações feitas.

1.3.6. Vendo o Histórico do Projeto

Uma das características mais úteis do TortoiseSVN é a janela de Auditoria. A janela mostra a você a lista de todos os commits feitos para um arquivo ou diretório, e mostra detalhadamente as mensagens que você digitou (você *digitou* uma mensagem para o commit como sugerido? Se não, agora você vê porque isso é importante).

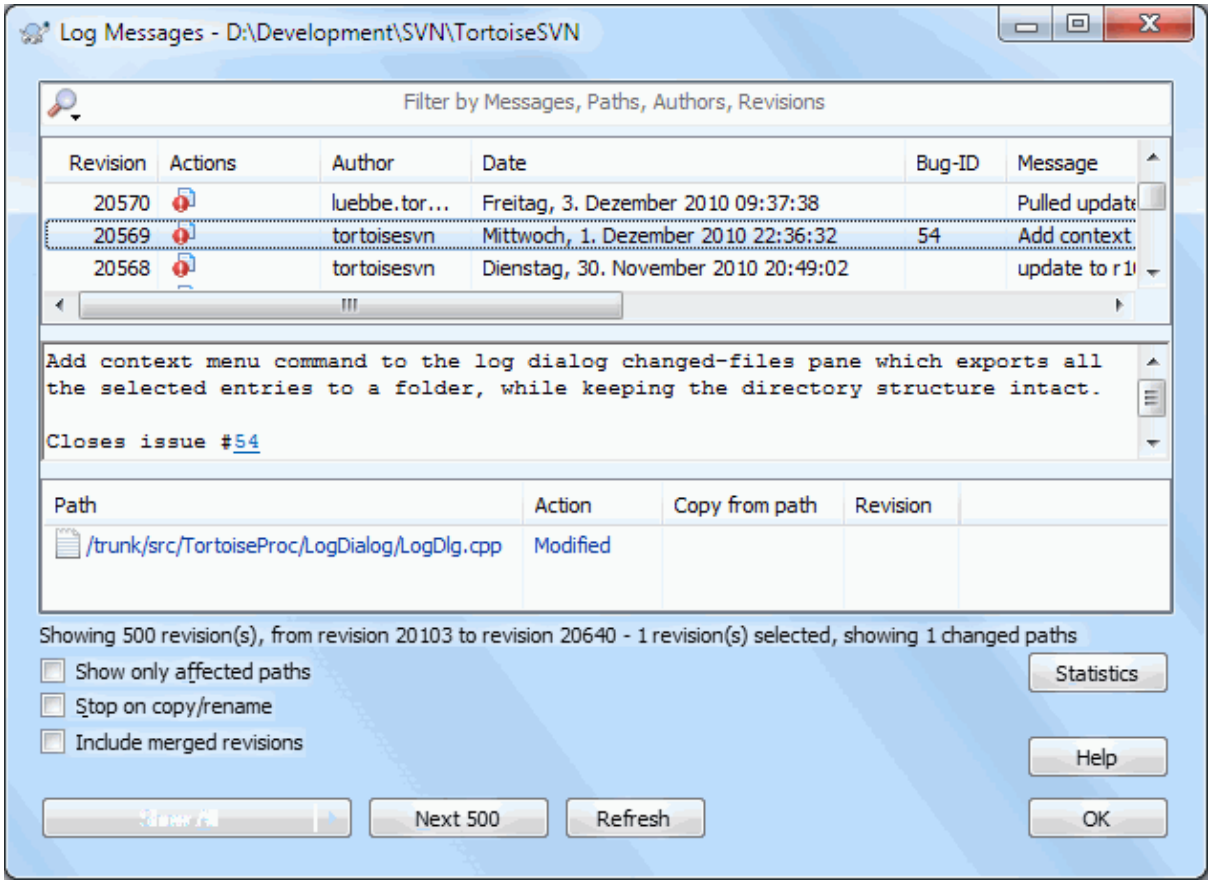


Figura 1.4. A Janela de Log

Bem, então eu manipulei um pouco aqui e usei uma imagem do repositório do TortoiseSVN.

O painel superior mostra a lista das revisões submetidas e o início da mensagem relacionada. Se você selecionar uma dessas revisões, o painel do meio mostrará a mensagem completa para a revisão e o painel de baixo mostrará a lista dos arquivos e diretórios modificados.

Cada um dos painéis possui um menu de contexto que provê a você várias maneiras de usar a informação. No painel inferior você pode dar um duplo clique em um arquivo e ver exatamente as alterações feitas naquela revisão. Leia Seção 4.10, “Janela de Revisão de Registro” para ver o texto completo.

1.3.7. Desfazendo Alterações

Umas das características de todo sistema de controle de revisão é que eles permitem a você desfazer alterações feitas anteriormente. Como você pode esperar, TortoiseSVN faz isso de forma simples.

Se você quer controlar as mudanças que você fez e ainda não submeteu e desfazer essas mudanças voltando para a versão original, TortoiseSVN → Reverter

Se você quer desfazer as alterações de uma revisão em particular, abra a janela de Auditoria e encontre a revisão em questão. Selecione Menu de Contexto → Reverter alterações desta revisão e então as alterações da revisão serão desfeitas.

1.4. Continuando...

Este guia tem mostrado a você uma breve descrição das coisas mais importantes do TortoiseSVN e funcionalidades úteis, mas é claro que há mais que isso que nós não abordamos. Nós fortemente recomendamos que você gaste um tempo para ler todo o manual, especialmente Capítulo 4, *Guia do Uso Diário* que lhe dará mais detalhes sobre o que se faz no do dia-a-dia.

Nós encontramos muitas dificuldades para ter certeza de que este manual é tanto informativo quanto fácil de ler, mas nós reconhecemos que é há muito disso! Fique tranquilo e não tenha medo de tentar as coisas em um repositório de testes a medida que vai lendo. A melhor maneira de aprender é usando.

Capítulo 2. Conceitos básicos do Controle de Versão

Este capítulo é uma versão levemente modificada do mesmo capítulo do livro do Subversion. Há uma versão disponível do livro do Subversion aqui: http://svnbook.red-bean.com/.

Este capítulo é uma pequena, simplória introdução ao Subversion. Se você é novato em controle de versões, este capítulo com certeza é para você. Nós começamos com uma abordagem geral dos conceitos do controle de versão, seguindo nossas idéias sobre o Subversion, e mostrando alguns exemplos simples de uso do Subversion.

Embora os exemplos desse capítulo mostre pessoas compartilhando uma porção de código de fonte, tenha em mente que Subversion pode controlar qualquer conjunto de arquivos - não está limitado a ajudar programadores.

2.1. O Repositório

Subversion é um sistema centralizador de compartilhamento de informação. A essência é um *repositório*, que é um arquivo central de dados. O repositório grava informação no formato de *diretório de arquivo de sistemas* - uma típica hierarquia de arquivos e diretórios. Quantos *clientes* quiser se conectam no repositório, e então leem e escrevem nesses arquivos. Escrevendo dados, um usuário torna a informação disponível para outros; lendo os dados, um usuário recebe a informação de outros.

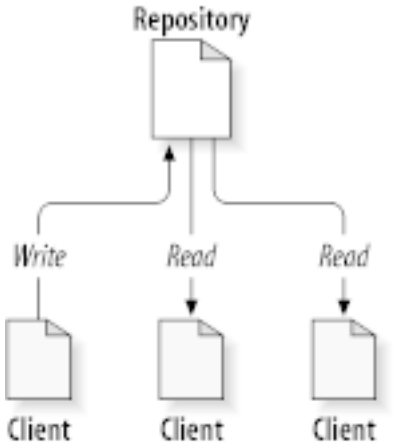


Figura 2.1. Um típico sistema Cliente/Servidor

Então porque isto é interessante? Até agora, isso se parece com a definição de um típico servidor de arquivos. E de verdade, o repositório *é* um tipo de servidor de arquivos, mas não é sua função normal. O que o repositório do Subversion faz de especial é que *ele guarda cada modificação* feita: cada modificação para cada arquivo e mesmo as modificações na pória estrutura de diretórios, como adição, exclusão e reorganização de arquivos e diretórios.

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

2.2. Modelo de controles

Todo sistema de controle de versão precisa resolver algumas problemas fundamentais: como o sistema vai permitir aos usuários compartilhar a informação, mas prevenindo que um não atrapalhe o outro? É muito fácil para os usuários acidentalmente sobreescrever as modificações de outros no repositório.

2.2.1. O problema do compartilhamento de arquivo

Considere o cenário: suponha que nós temos dois usuários, Harry e Sally. Cada um deles decide editar o mesmo arquivo no mesmo repositório ao mesmo tempo. Se Harry salvar suas alterações no repositório primeiro, é possível que (alguns tempo depois) Sally poderia acidentalmente sobreescrever as alterações com sua versão do arquivo. Enquanto a versão do arquivo do Harry se perderia para sempre (porque o sistema guarda cada modificação), qualquer alteração que Harry fez *não estariam* presentes na nova versão do arquivo de Sally, porque ela nunca viu as modificações do Harry. O trabalho do Harry foi perdido - ou no mínimo estaria na versão mais recente - e provavelmente por acidente. Esta é definitivamente um situação que nós queremos evitar!

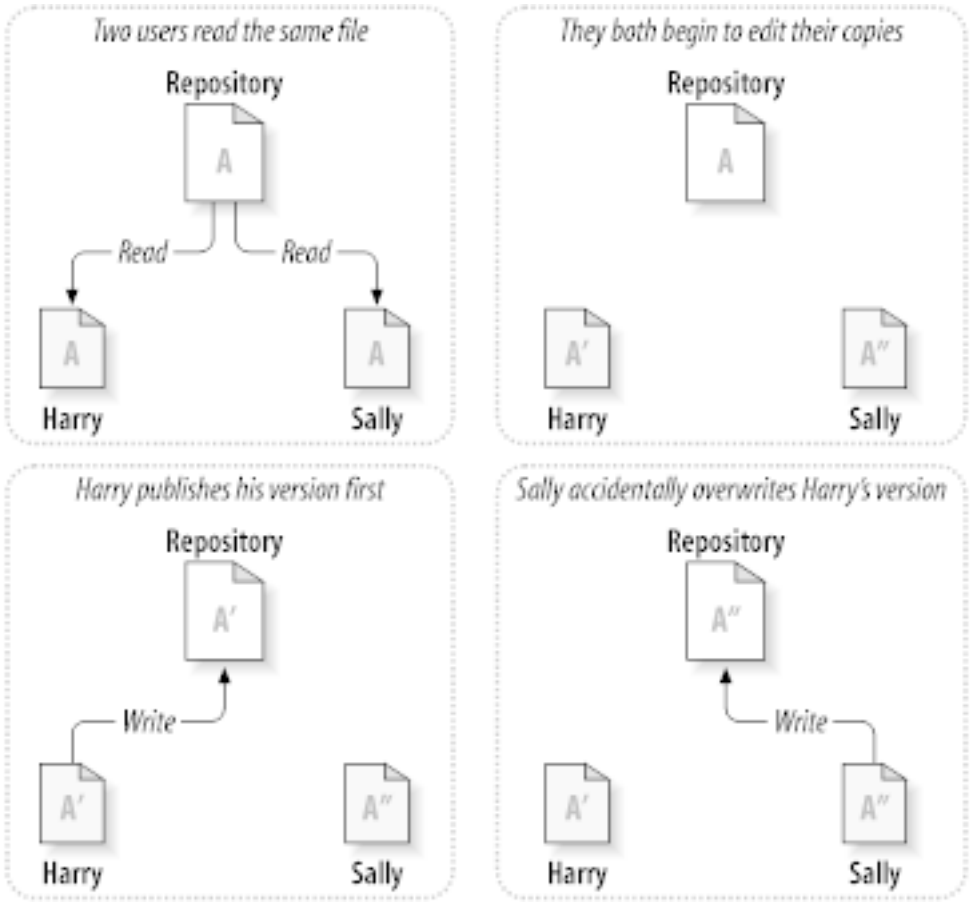


Figura 2.2. O problema a ser evitado

2.2.2. A solução Alocar-Modificar-Desalocar

Muitos sistemas de controle de versão usam o modelo *alocar-modificar-desalocar* para resolver este problema, o qual é uma solução simples. Em cada sistema, o repositório permite somente uma pessoa por vez modificar o arquivo. Primeiro Harry deve *alocar* o arquivo antes que possa fazer as alterações. Alocar um arquivo é como um controle de biblioteca/ se Harry alocou o arquivo, então Sally não pode fazer qualquer alteração nele. Se ela tentar alocar o arquivo, o repositório vai negar essa solicitação. Tudo que ela pode fazer é ler o arquivo, e esperar até que Harry acabe as suas alterações e libere o arquivo. Depois que Harry desalocar o arquivo, sua vez acaba, e então é a vez de Sally alocar o arquivo e fazer suas alterações.

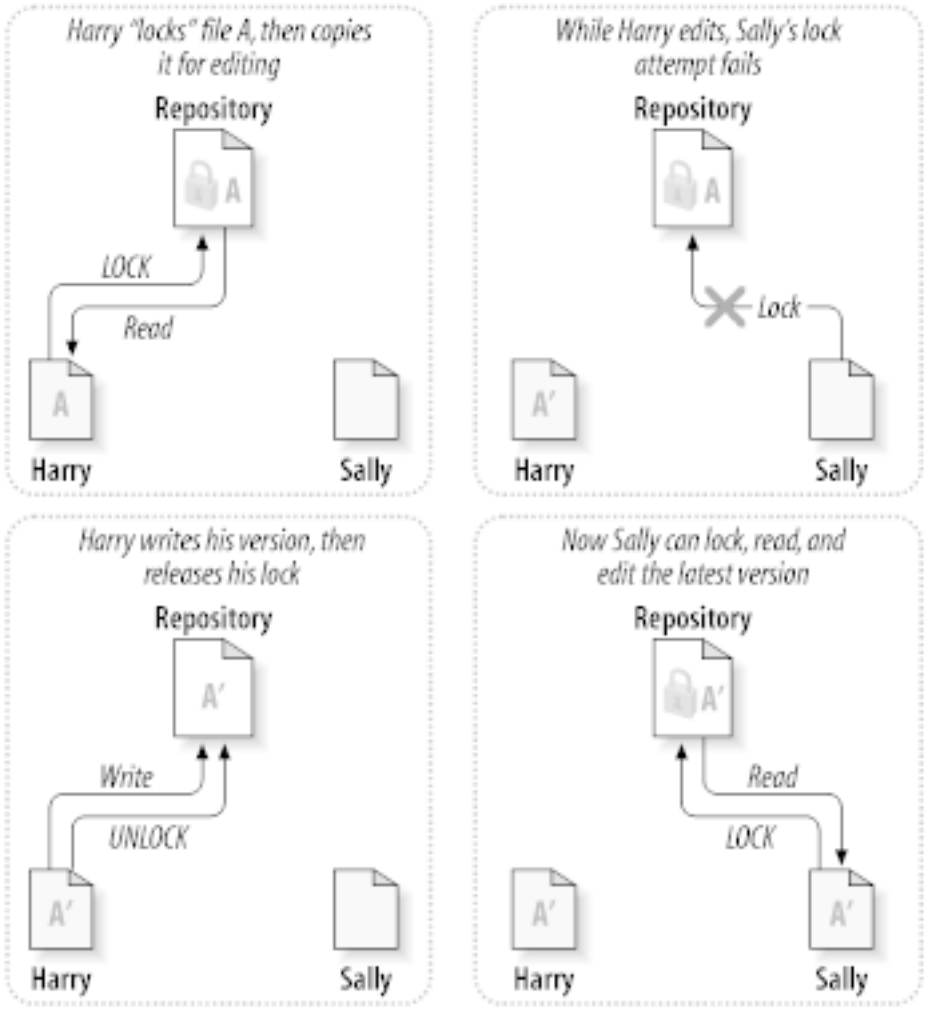


Figura 2.3. A solução Alocar-Modificar-Desalocar

O problema com o modelo alocar-modificar-desalocar é que é muito restritivo, e muitas vezes é um empecilho para os usuários:

- *Alocação pode causar problemas administrativos.* Algumas vezes Harry vai alocar o arquivo e então esquecer dele. Entretanto, porque Sally continua esperando para para editar o arquivo, suas mãos estão atadas. E então Harry sai de férias. Agora Sally precisa pedir a um administrador para liberar o arquivo alocado por Harry. A situação acaba causando atrasos e uma porção de tempo perdido desnecessário.

- *Alocação pode causa uma serialização desnecessária.* O que fazer se Harry estava editando o início de um arquivo texto, e Sally quer editar apenas o fim do arquivo? Estas alterações não se sobrepoem. Eles poderiam facilmente editar o arquivo ao mesmo tempo, e nenhum grande problema ocorreria, assumindo que as modificações seriam corretamente unificadas. Não é necessário travar o arquivo neste caso.

- *Alocar pode criar uma falta noção de segurança.* Suponhamos que Harry aloque e altere o arquivo A, enquanto ao mesmo tempo Sally aloca e edita o arquivo B. Supondo que A e B dependem um do outro, e que as modificações feitas em cada um são semanticamente incompatíveis. Imprevisivelmente A e B não funcionam mais juntos. O sistema de alocação não tem como prever este problema - ainda que esse sistema passe uma falta sensação de segurança. É fácil para Harry e Sally imaginar que alocando arquivos, cada um está seguro, numa tarefa isola, e deste modo evitando discussões precosss sobre as modificações.

2.2.3. A solução Copiar-Modificar-Unificar

Subversion, CVS, and other version control systems use a *copy-modify-merge* model as an alternative to locking. In this model, each user's client reads the repository and creates a personal *working copy* of the file or project.

Users then work in parallel, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a human being is responsible for making it happen correctly.

Aqui vai um exemplo. Digamos que Harry e Sally criam cada um uma cópia de trabalho de um mesmo projeto, copiado do repositório. Eles trabalham ao mesmo tempo, e fazem modificações em um mesmo arquivo A em suas cópias. Sally salva suas alterações no repositório primeiro. Quando Harry tenta salvar suas modificações após Sally, o repositório informa a ele que o arquivo A está *desatualizado*. Em outras palavras, o arquivo A do repositório tem alguma alteração desde a última vez que ele foi copiado. Então Harry é questionado sobre a *unificação* das modificações no repositório serem inseridas em sua cópia do arquivo A. Oportunamente as modicações de Sally não foram sobreescritas pelas dele; uma vez que ele unificou ambas as alterações, ele salva a sua cópia no repositório.

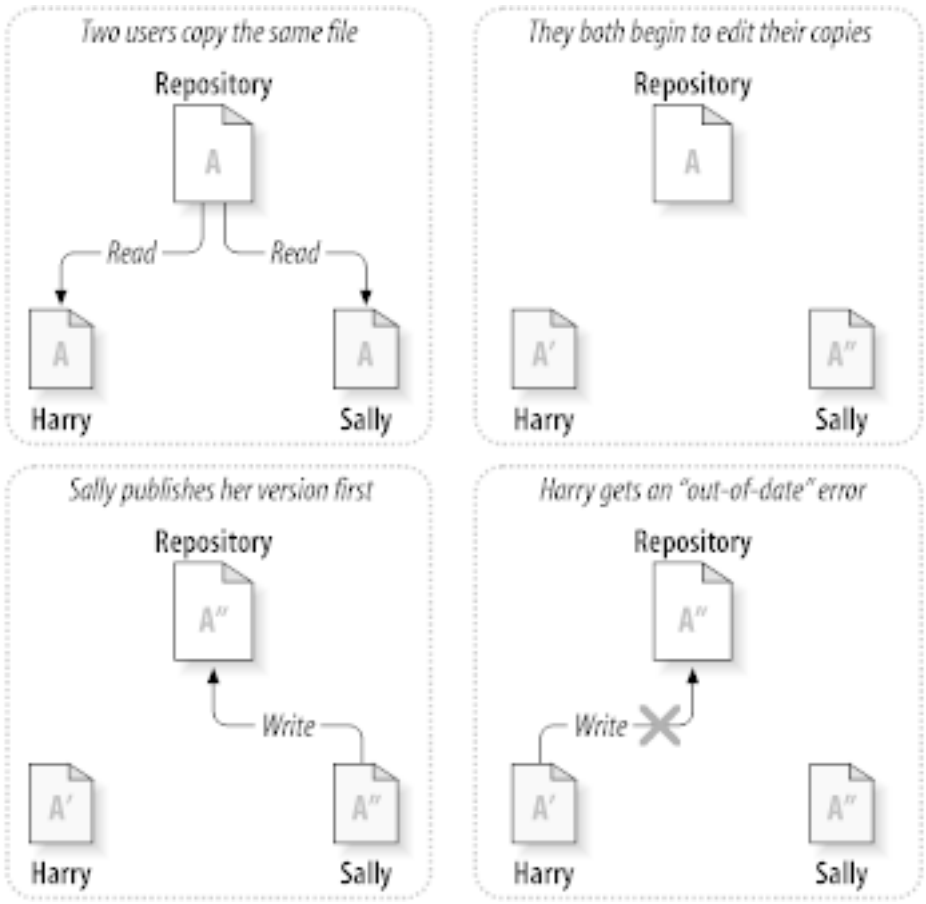


Figura 2.4. A solução Copiar-Modificar-Unificar

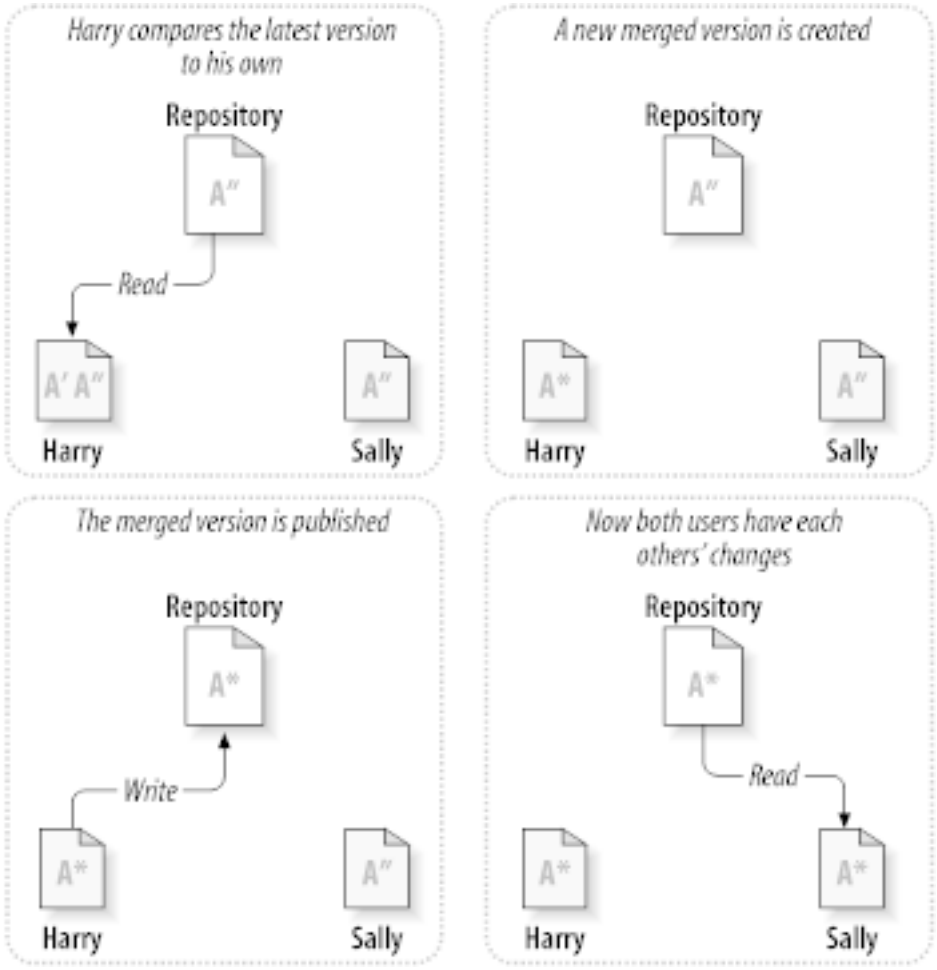


Figura 2.5. ...Continuando com Copiar-Modificar-Unificar

Mas o que acontece se as alterações de Sally se *sobrepoem* sobre as alterações de Harry? O que então deve ser feito? Esta situação é chamada de *conflito*, e em geral isto não é um problema. Quando Harry for questionado sobre a unificação das últimas alterações no repositório em sua cópia de trabalho, sua cópia do arquivo A de qualquer maneira é marcada como conflitante: ele verá todas as modificações em conflito, e manualmente resolverá. Entenda que o software não pode resolver automaticamente os conflitos; somente humanos são capazes de entender quais são as escolhas lógicas a serem tomadas. Uma vez que Harry manualmente resolveu o que estava se sobreponde (talvez precise discutir o conflito com Sally!), ele pode seguramente salvar o arquivo unificado de volta no repositório.

O modelo copiar-modificar-unificar pode parecer bagunçado, mas na prática, isto funciona grandemente. Usuários podem trabalhar paralelamente, sem esperar por outros. Quando eles trabalham num mesmo arquivo, a maioria das alterações não se sobrepoem; conflitos são pouco frequentes. E na maioria das vezes o tempo que leva para resolver os conflitos é muito menor que o tempo perdido em um sistema de alocação.

No final das contas, tudo isso acaba em um ponto crítico: comunicação entre os usuários. Quando os usuários não se comunicam direito, aumenta-se os conflitos sintáticos e semênticos. Nenhum sistema pode forçar uma perfeita comunicação, nenhum sistema pode detectar conflitos de semântica. Então não há porque se entusiasmar com falsas promessas de que um sistema de alocação evitará conflitos; na prática, alocações parecem restringir a produtividade mais que qualquer outra cosa.

Existe uma situação em comum onde o modelo alocar-modificar-desalocar se torna melhor, e é com arquivos que não são unificáveis. Por exemplo, se seu repositório contém alguns arquivos de imagens, e duas pessoas mudam a imagem ao mesmo tempo, não há nenhuma maneira de combinar as alterações. Ou Harry ou Sally perderá sua alteração.

2.2.4. O que o Subversion faz?

Subversion usa a solução copiar-modificar-unificar como padrão, e na maioria dos casos é o suficiente. Contudo, a partir da versão 1.2, Subversion também permite alocar um arquivo, e se existem arquivos que não são unificáveis, você pode adotar uma política de alocação, que o Subversion está preparado para o que você precisa.

2.3. Subversion em Ação

2.3.1. Cópias de Trabalho

Você está pronto para ler sobre cópias de trabalho; agora vamos demonstrar como o aplicativo do Subversion cria e usa as cópias.

Uma cópia de trabalho do Subversion é uma estrutura de diretórios como outra qualquer em seu sistema local, contendo um conjunto de arquivos. Você pode editar esses arquivos como desejar, e se os arquivos são códigos fonte, você pode compilar seu programa a partir desta cópia como sempre fez. Sua cópia de trabalho é sua própria área pessoal. Subversion nunca vai incorporar alterações de outras pessoas, nem disponibilizar suas alterações para outros, até que você mesmo o faça.

Depois que você fez algumas modificações nos seus arquivos em sua cópia de trabalho e verificou que estão funcionando corretamente, Subversion provê a você comandos para *publicar* suas alterações para outras pessoas que trabalham com você em seu projeto (através da escrita no repositório). Se outras pessoas publicarem suas próprias modificações, Subversion provê comandos para unificar estas modificações na sua cópia de trabalho (através da leitura do repositório).

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

Um repositório típico do Subversion geralmente guarda os arquivos (ou código fonte) para diversos projetos; normalmente, cada projeto é um subdiretório na estrutura de arquivos do repositório. Desta forma, uma cópia de trabalho de um usuário normalmente corresponde a uma específica subestrutura do repositório.

Por exemplo, suponha que você tenha um repositório que contem dois projetos de software.

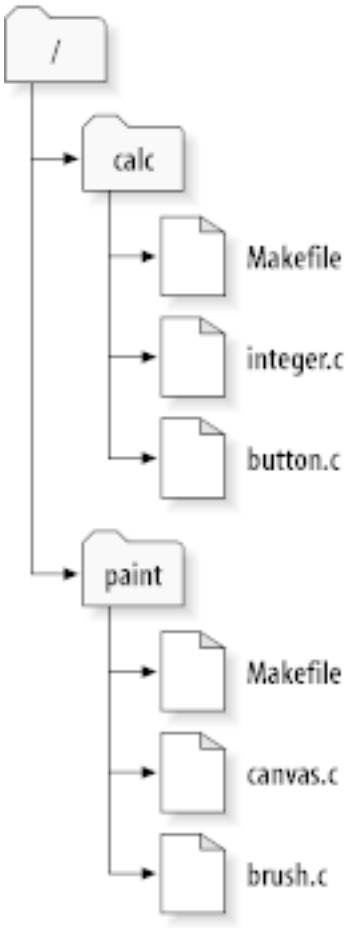


Figura 2.6. O Sistema de Arquivos do Repositório

Em outras palavras, o diretório principal do repositório tem dois subdiretórios: `paint` e `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.)

Suppose you make changes to `button.c`. Since the `.svn` directory remembers the file's modification date and original contents, Subversion can tell that you've changed the file. However, Subversion does not make your changes public until you explicitly tell it to. The act of publishing your changes is more commonly known as *committing* (or *checking in*) changes to the repository.

Para publicar suas modificações para outras pessoas, você pode usar o commando **submeter** do Subversion.

Agora suas modificações para `button.c` foram enviadas para o repositório; se outro usuário obter uma cópia de `/calc`, eles verão suas modificações na última versão do arquivo.

Suponha que você tenha uma colaboradora, Sally, que obteve uma cópia de `/calc no mesmo momento que você. Quando você submeter suas modificações de button.c, a cópia de trabalho de Sally não será modificada; Subversion somente modificará as cópias dos usuários que solicitarem a atualização.`

Para atualizar seu projeto, Sally deve pedir ao Subversion para *atualizar* sua cópia de trabalho, através do comando **atualizar** do Subversion. Isto incorporará sua modificação na cópia de trabalho dela, assim como qualquer outra modificação que tenha sido submetido desde que ela obteve sua cópia de trabalho.

Note que Sally não precisou especificar quais arquivos atualizar; Subversion usa a informação do diretório `.svn`, e além disso informações do repositório, para decidir quais arquivos precisam ser atualizados.

2.3.2. URLs do Repositório

Repositórios do Subversion podem ser acessados através de vários métodos direfentes - como disco local, ou através de vários protocolos de rede. Um local de repositório, contudo, é sempre uma URL. O esquema de URL indica o método de acesso:

Esquema	Método de Acesso
`file://`	Acesso direto ao repositório em um local ou dispositivo de rede.
`http://`	Acesso através do protocolo WebDAV para o módulo Subversion do servidor Apache.
`https://`	O mesmo que `http://`, mas com criptografia SSL
`svn://`	Acesso TCP/IP não autenticado através do protocolo customizado para um servidor `svnserve`.
`svn:ssh://`	acesso TCP/IP autenticado e criptografado através de um protocolo customizado para um servidor `svnserve`

Tabela 2.1. URLs de Acesso ao Repositório

Para a maioria, URLs do Subversion usam a sintaxe padrão, permitido nomes de servidores e números de porta em uma parte específica da URL. O método de acesso `file://` é normalmente usado para acesso local, apesar de que isto pode ser usado com caminhos UNC em uma máquina da rede. A URL portanto usa o formato `file://hostname/path/to/repos`. Para a máquina local, a parte do `hostname` da URL deve ser suprimida ou deve ser `localhost`. Por esta razão, caminhos locais normalmente aparecem com três barras, `file:///path/to/repos`.

Também, usuários do esquema `file://` em plataformas Windows vão precisar usar uma sintaxe “padrão” não oficial para acessar repositórios que estão na mesmo máquina, mas em um dispositivo diferente que o atual dispositivo de trabalho da máquina. Qualquer uma das duas seguintes sintaxes funcionarão onde `X` é o dispositivo no qual o repositório está:

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

Note que a URL usa barras normais embora a forma nativa (não URL) de um caminho no Windows seja barra invertida.

Você pode acessar um repositorio FSFS via uma rede compartilhada, mas isso *não* é recomendado por várias razões:

- Você está concedendo permissão de escrita para todos os usuários, assim, eles podem acidentalmente excluir arquivos ou corromper o sistema de arquivos do repositório.
- Not all network file sharing protocols support the locking that Subversion requires. One day you will find your repository has been subtly *corrupted*.
- Você precisa definir as permissões de acesso da maneira correta. SAMBA é particularmente difícil neste quesito.
- Se uma pessoa instala uma versão mais recente do cliente que atualiza o formato do repositório, todos os demais serão impossibilitados de acessar o repositório até que também tenham atualizado para a nova versão do cliente.

2.3.3. Revisões

Um comando **svn submeter** pode publicar modificações de qualquer quantidade de arquivos e diretórios como uma única transação atômica. Em sua cópia de trabalho, você pode mudar o conteudo de um arquivo, criar,

excluir, renomear e copiar arquivos e diretórios, e depois enviar todas as modificações como uma única e completa alteração.

Em um repositório, cada submissão é tratada como uma transação atômica: ou todas as mudanças são enviadas, ou nenhuma delas. Subversion mantém essa atomicidade por causa de falhas de programas, falhas de sistema, falhas de rede, e ações de outros usuários.

Cada vez que o repositório aceita uma submissão, ele cria um novo estado da estrutura de arquivos, chamada de *revisão*. Cada revisão é associada a um número natural, maior que o número da revisão anterior. A revisão inicial de um repositório recém criado é zero, e não contém nada mais que um diretório principal vazio.

Uma maneira interessante de enxergar o repositório é como uma série de estruturas. Imagine uma matriz de números de revisões, começando em 0, e se estendendo da esquerda para a direita. Cada número de revisão com uma estrutura de arquivos abaixo, e cada divisão já em seguida da maneira como o repositório realizou cada submissão.

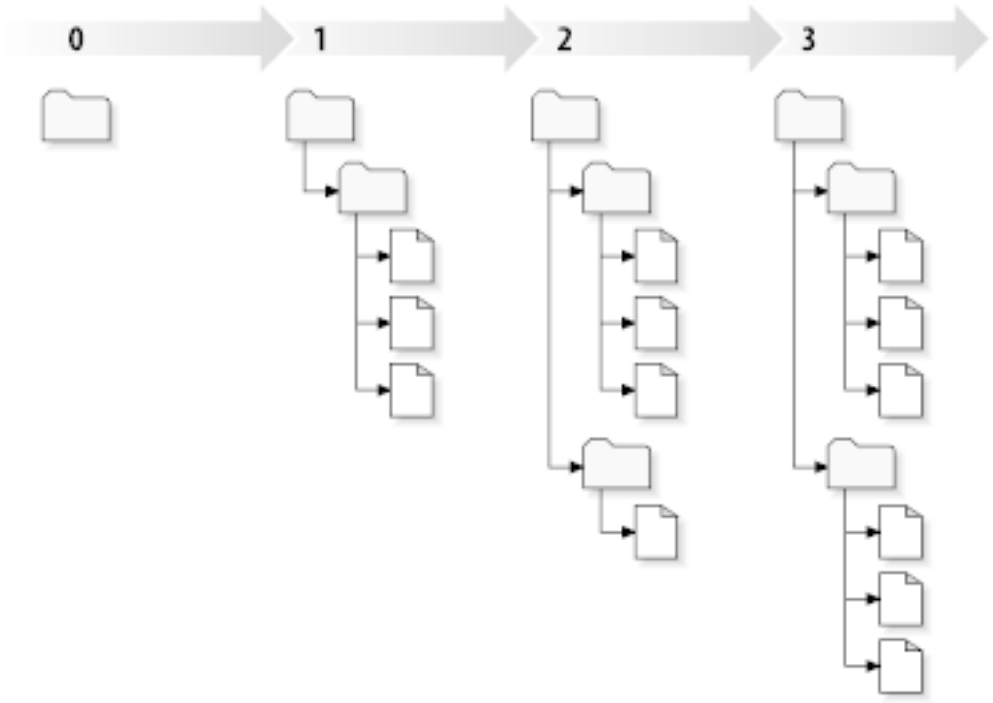


Figura 2.7. O Repositório

> **Números Globais de Revisão**
>
> Unlike those of many other version control systems, Subversion's revision numbers apply to *entire trees* , not individual files. Each revision number selects an entire tree, a particular state of the repository after some committed change. Another way to think about it is that revision N represents the state of the repository filesystem after the Nth commit. When a Subversion user talks about ``revision 5 of `foo.c`", they really mean ```foo.c` as it appears in revision 5." Notice that in general, revisions N and M of a file do *not* necessarily differ!

É importante notar que cópias de trabalho nem sempre correspondem a uma única revisão qualquer no repositório; eles podem conter arquivos de diferentes revisões. Por exemplo, suponha que você obteve uma cópia de trabalho de um repositório do qual a revisão mais recente é a 4:

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

```
button.c:4
```

Neste momento, esta cópia de trabalho corresponde exatamente à revisão 4 no repositório. Entretanto, suponha que você fez modificações em `button.c`, e submeteu essas alterações. Assumindo que não houve nenhuma outra submissão, o envio vai criar a revisão 5 no repositório, e sua cópia de trabalho vai se parecer com isto:

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

Suponha que, neste momento, Sally envia uma alteração do `integer.c`, criando a revisão 6. Se você usar o **svn atualizar** para atualizar sua cópia de trabalho, então você terá algo como isto:

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

As modificações de Sally para `integer.c` vão aparecer em sua cópia de trabalho, e sua alteração vai continuar presente em `button.c`. Neste exemplo, o texto de `Makefile` é identifico nas revisões 4, 5 e 6, mas Subversion vai marcar sua cópia de trabalho do `Makefile` com a revisão 6 indicando que ainda é a versão atual. Então, depois que você completou todas as atualizações na sua cópia de trabalho, isto geralmente corresponderá a uma exata revisão do repositório.

2.3.4. Como Cópia de Trabalho Acompanham o Repositório

Para cada arquivo do diretório de trabalho, Subversion grava duas partes essenciais da informação na área administrativa `.svn/`:

- what revision your working file is based on (this is called the file's *working revision*), and
- a informação de data/hora de quando foi a última atualização da cópia do repositório.

Com estas informação, através de troca de informações com o repositório, Subversion pode identificar qual dos quatro estados seguintes é o estado de um arquivo:

Não modificado, e atualizado
: O arquivo não foi modificado na cópia de trabalho, e nenhuma nova modificação foi submetida no repostório considerando a revisão de trabalho. Um **submetero> do arquivo natualizar do arquivo tamb**

Localmente alterado, e atualizado
: O arquivo possui modificações no diretório de trabalho, e nenhuma modificação para o arquivo foi enviada para o repositório considerando a revisão atual. Existem alterações que não foram submetidas para o repositório, deste modo um **submeter** do arquivo vai obter sucesso ao publicar suas alterações, e um **atualizar** do arquivo não fará nada.

Não modificado, e desatualizado
: O arquivo não possui alterações no diretório de trabalho, mas há modificações no repositório. O arquivo deverá eventualmente ser atualizado, para sincronizar com a revisão publica. Um **submeter** do arquivo não fará nada, e um **atualizar** do arquivo trará as últimas alterações para a cópia local.

Localmente modificado, e desatualizado
: O arquivo possui modificações tanto no diretório de trabalho, como no repositório. Um **submeter** do arquivo vai falhar com o erro *desatualizado*. O arquivo deverá ser atualizado primeiro; o comando **atualizar** vai tentar unificar as alterações do repositório com as alterações locais. Se o Subversion não conseguir completar a unificação de forma plausível automaticamente, ele deixará para o usuário resolver o conflito.

2.4. Resumo

Nós apresentamos vários conceitos fundamentais do Subversion neste capítulo:

- Nós introduzimos as noções de um repositório central, a cópia de trabalho do usuário, e uma porção da estrutura de revisões do repositório.
- Nós mostramos alguns exemplos simples de como dois colaboradores podem usar o Subversion para publicar e receber alterações um do outro, usando o modelo 'copiar-modificar-unificar'.
- Nós falamos um pouco sobre a forma como o Subversion acompanha e controla as informações em uma cópia de trabalho.

Capítulo 3. O Repositório

Não importa qual protocolo você use para acessar seus repositórios, você sempre precisará criar pelo menos um repositório. Isto pode ser feito igualmente com a linha de comando do Subversion ou com o TortoiseSVN.

Se você não criou ainda um repositório do Subversion, a hora é agora.

3.1. Criação do Repositório

3.1.1. Criando um Repositório com a linha de comando do cliente

1. Cria um diretório vazio com o nome SVN (ex: `D:\SVN\`), o qual é usado como diretório principal para todos os seus repositórios.
2. Cria outro diretório `MyNewRepository` dentro de `D:\SVN\`
3. Open the command prompt (or DOS-Box), change into `D:\SVN\` and type

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

Agora você tem uma novo repositório localizado em `D:\SVN\MyNewRepository`.

3.1.2. Criando um Repositório com o TortoiseSVN

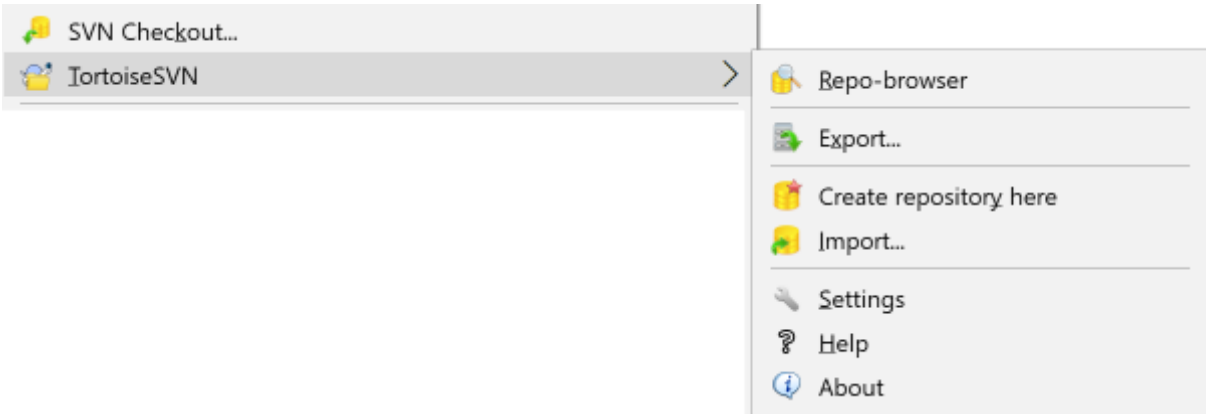


Figura 3.1. O menu do TortoiseSVN para diretórios não controlados

1. Abra o windows explorer
2. Crie um novo diretório e chame de, por exemplo `SVNRepository`
3. clique direito sobre o novo diretório criado e selecione TortoiseSVN → Criar repositório aqui....

   Um repositório é então criado dentro do novo diretório. *Não edite esses arquivos você mesmo!!!*. Se apresentar algum erro tenha certeza que o diretório está vazio e sem proteção de escrita.

   Você será também questionado se você quer criar uma estrutura de diretórios dentro do repositório. Saiba sobre as opções de leiaute em Seção 3.1.5, “Leiaute do Repositório”.

   TortoiseSVN irá configurar um ícone de pasta personalizado quando criar um repositório. Assim você poderá identificar os repositórios locais mais facilmente. Se você criar um repositório usando a linha de comando do cliente oficial o ícone para o diretório não será associado.

Dica

Também recomendamos que você não use `file://` por qualquer razão que não sejam apenas testes locais. Usar um servidor é mais seguro e mais confiável para todos, a não ser para o uso de um único desenvolvedor.

3.1.3. Acesso Local para o Repositório

Para acessar seu repositório local você precisa do caminho para a pasta. Apenas lembre-se que Subversion esperar todos os caminhos dos repositórios no formato `file:///C:/SVNRepository/`. Note o uso de barras em todo o caminho.

Para acessar o repositório localizado em uma rede compartilhada você pode também usar uma unidade mapeada, ou você pode usar um caminho UNC. Para caminhos UNC, o formato é `file://NomeServidor/caminho/para/repos/`. No que há apenas 2 barras aqui.

Antes do SVN 1.2, caminhos UNC tinham que ser informados num formato mais complicado `file:///\NomeServidor/caminho/para/repos`. Este formato ainda é suportado, mas não é recomendado.

3.1.4. Acessando um Repositório em uma Rede Compartilhada

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 for various reasons:

- Primeiramente você está dando para todos os usuários dreito de escrita no repositório, então qualquer usuário pode acidentalmente apagar o repositório inteiro ou corrompe-lo de alguma maneira.
- Em segundo lugar nem todo protocolo de compartilhamento de arquivo suporta as travas necessárias para o Subversion, então você pode encontrar seu repositório corrompido. Isto pode não acontecer logo, mas um dia dois usuários vão tentar acessar o repositório ao mesmo tempo.
- Em terceiro lugar as permissões do arquivo precisam ser bem definidas. Você pode achar fácil num compartilhamento Windows, mas SAMBA é relativamente difícil.
- Se uma pessoa instala uma versão mais recente do cliente que atualiza o formato do repositório, todos os demais serão impossibilitados de acessar o repositório até que também tenham atualizado para a nova versão do cliente.

O acesso através de `file://` é destinado para local, um único usuário, e particularmente para testes e depurações. Quando você quer compartilhar o repositório você *realmente* deve configurar um servidor, e isto não é tão difícil como você deve estar pensando. Leia Seção 3.5, “Acessando o Repositório” para orientações sobre escolhas e configurações de um servidor.

3.1.5. Leiaute do Repositório

Antes de você importar seus dados para o repositório você deve primeiro pensar sobre como você quer organizar os dados. Se você usar um dos formatos recomendados depois você o terá muito mais organizado.

Existem alguns padrões e recomendadas maneiras de organizar um repositório. Muitas pessoas criam um diretório `trunk` para guardar a “linha principal” de desenvolvimento, um diretório `branches` para guardar as ramificações, e um diretório `tags` para guardar as versões concluidas. Se o repositório guarda somente um único projeto, então frequentemente as pessoas criam estes diretórios no nível mais alto:

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

Porque este formato é tão comumente usado, quando você criar um novo repositório usando o TortoiseSVN, ele também oferecerá a criação da estrutura de diretórios para você.

Se o repositório contém vários projetos, as pessoas normalmente organizam sua estrutura por ramificação:

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

... ou por projeto:

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

Organizar por projeto faz sentido se os projetos não são fortemente relacionados e cada um deles é controlado individualmente. Para projetos relacionados onde você pode querer controlar todos os projeto de uma vez, ou onde os projetos são todos amarrados uns aos outros em um único pacote de distribuição, é então mais comum organizar por ramificação. Desta maneira você tem somente um trunk para controlar, e os relacionamentos entre os subprojetos são vislumbrados mais facilmente.

Se você adotar como nível mais alto algo como`/trunk /tags /branches`, não há nada para dizer que você tenha que copiar o trunk inteiro para cada ramificação e liberação, e de alguma forma esta estrutura oferece mais flexibilidade.

Para projetos não relacionados você deve dar preferência para repositórios separados. Quando você submeter alterações, o novo número de revisão será do repositório inteiro, não o apenas a revisão do projeto. Tendo 2 projetos não relacionados compartilhando um mesmo repositório pode significar grandes lacunas no número da revisão. Os projetos Subversion e o TortoiseSVN aparecem no mesmo endereço principal, mas são repositórios completamente separados permitindo o desenvolvimento independente, e sem confusões no número da liberação.

Claro, você é livre para ignorar estes leiautes padrões. Você pode criar qualquer tipo de variação, seja o que funcionar melhor para você e sua equipe. Lembre-se que qualquer que seja a sua escolha, isto não é uma decisão permanente. Você pode reorganizar seu repositório a qualquer momento. Pelos diretórios branches e tags serem diretórios comuns, TortoiseSVN pode mover ou renomeá-los do jeito que você quiser.

Mudar de um leiaute para outro é apenas questão de mover uns diretórios no servidor; se você não gosta da maneira que as coisas estão organizadas no repositório, basta manipular os diretórios no repositório.

Então se você ainda não tem criada uma estrutura base de diretórios dentro do seu repositório você deverá fazer agora. Existem duas maneiras de fazer isso. Se você simplesmente quer criar uma estrutura `/trunk /tags /branches`, você pode usar o navegador de repositórios para criar as três pastas (em três separadas submissões). Se você quer criar um hierarquia mais estruturada então crie a estrutura primeiro no disco local e importe-a em uma única submissão, como isto:

1. criar uma nova pasta vazia em seu disco rígido
2. criar seu diretório da estrutura principal dentro da pasta - não coloque nenhum arquivo dentro ainda!
3. importar esta estrutura em um repositório pelo clique direito em uma pasta que contem esta estrutura de diretório e selecionando TortoiseSVN → Importar.... Na janela de importação digite a URL para o repositório e clique

em OK. Isto irá importar seu diretório temporário dentro da raiz do repositório para criar o leiaute básico do repositório.

Note that the name of the folder you are importing does not appear in the repository, only its contents. For example, create the following folder structure:

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

Import `C:\Temp\New` into the repository root, which will then look like this:

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

3.2. Cópia de Segurança do Repositório

Não importante o tipo de repositório que você usa, é vitalmente importante que você mantenha cópias de segurança regulares, e que você verifique a cópia de segurança. Se o servidor falhar, você terá condições de acessar uma versão recente de seus arquivos, mas sem o repositório todo o histórico estará perdido para sempre.

The simplest (but not recommended) way is just to copy the repository folder onto the backup medium. However, you have to be absolutely sure that no process is accessing the data. In this context, access means *any* access at all. If your repository is accessed at all during the copy, (web browser left open, WebSVN, etc.) the backup will be worthless.

The recommended method is to run

```
svnadmin hotcopy path/to/repository path/to/backup
```

to create a copy of your repository in a safe manner. Then backup the copy.

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

3.3. Rotinas de eventos no servidor

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

Estas rotinas para eventos são executadas pelo servidor que hospeda o repositório. TortoiseSVN também permite que você configura no cliente rotinas para eventos que são executando localmente em consequência de certos acontecimentos. Veja Seção 4.31.8, “Client Side Hook Scripts” para mais informações.

Sample hook scripts can be found in the `hooks` directory of the repository. These sample scripts are suitable for Unix/Linux servers but need to be modified if your server is Windows based. The hook can be a batch file or an executable. The sample below shows a batch file which might be used to implement a pre-revprop-change hook.

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

Note that anything sent to stdout is discarded. If you want a message to appear in the Commit Reject dialog you must send it to stderr. In a batch file this is achieved using `>&2`.

Sobreescrevendo ganchos

Se uma rotina para evento rejeitar sua submissão, então sua decisão é final. Mas você pode construir um mecanismo de desvio na própria rotina usando a técnica de *Palavra Mágica*. Se a rotina quiser rejeitar a operação ela primeiro vasculha a mensagem de log por uma senha específica, frase fixa ou talvez o nome do arquivos com um prefixo. Se ela encontrar a palavra mágica, então ela permite que a submissão continue. Se a frase não for encontrada, então ela bloqueia a submissão com uma mensagem como “Você não disse a palavra mágica”. :-)

3.4. Vínculos externos

Se você quer disponibilizar seu repositório Subversion para outras pessoas você pode incluir um vínculo no seu website. Uma forma de tornar isso mais fácil é incluir um *vínculo externo* para outros usuários do TortoiseSVN.

Quando você instala o TortoiseSVN, ele registra um novo protocolo `tsvn:`. Quando um usuário do TortoiseSVN clica num vínculo como este, a janela de obtenção será aberta automaticamente com a URL do repositório já preenchida.

Para incluir tal vínculo em sua própria página HTML, você precisa adicionar um código parecido com este:

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

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

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

Você pode também fazer o vínculo apontar para uma revisão específica, por exemplo

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

3.5. Acessando o Repositório

Para usar o TortoiseSVN (ou qualquer outro cliente do Subversion), você precisa de um lugar para guardar seus repositórios. Você pode também guardar seus repositórios e acessá-los usando o protocolo `file://` ou você pode colocá-los em um servidor e acessá-los usando os protocolos `http://` ou `svn://`. Os dois protocolos podem também ser criptografados. Você pode usar `https://` ou `svn+ssh://`, ou você pode usar `svn://` com SASL.

If you are using a public hosting service such as *SourceForge* [https://sourceforge.net] or your server has already been setup by someone else then there is nothing else you need to do. Move along to Capítulo 4, *Guia do Uso Diário*.

Se você não tem um servidor e vai trabalhar sozinho, ou se você está apenas avaliando o Subversion e o TortoiseSVN isoladamente, então repositórios locais são provavelmente a melhor opção. Apenas crie um repositório em seu próprio computador como descrito anteriormente em Capítulo 3, *O Repositório*. Você pode pular o resto deste capítulo e ir diretamente para Capítulo 4, *Guia do Uso Diário* para saber como começar a usá-los.

Se você está pensando em configurar um repositório para vários usuários em uma pasta compartilhada, pense novamente. Leia Seção 3.1.4, “Acessando um Repositório em uma Rede Compartilhada” para descobrir porque isto é uma má idéia. Configurando um servidor não é tão difícil quanto parece, e isto vai ter mais segurança e provavelmente mais velocidade.

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

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

- *VisualSVN* [https://www.visualsvn.com/server/]
- *CollabNet* [https://www.collab.net/products/subversion]

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

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

Capítulo 4. Guia do Uso Diário

Este documento descreve o uso diário do cliente TortoiseSVN. Este capítulo *não* é uma introdução aos sistemas de controle de versão, e *não* é uma introdução ao Subversion (SVN). Este capítulo é como um documento auxiliar para olhar quando você sabe mais ou menos o que quer fazer, mas não está absolutamente certo de como fazer.

Se você precisa aprender sobre o controle de versão usando Subversion, então nós recomendamos que você leia o fantástico livro: *Controle de Versão com Subversion* [http://svnbook.red-bean.com/].

Este documento é também um trabalho em progresso, assim como TortoiseSVN e Subversion também são. Se você encontrar qualquer erro, por favor reporte para a lista de discussão para que possamos atualizar a documentação. Algumas imagens do Guia do Uso Diário (GUD) pode não refletir a última versão do aplicativo. Por favor, perdoe-nos. Nós trabalhamos no TortoiseSVN no nosso tempo livre.

A fim de obter o máximo do Guia de Uso Diário:

- Você já deverá ter instalado o TortoiseSVN.
- Você deverá estar familiarizado com um sistema de controle de versões.
- Você deverá ter conhecimentos básicos do Subversion.
- Você deve ter configurado um servidor e/ou ter acesso a um repositório Subversion.

4.1. Características Gerais

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

4.1.1. Sobreposição dos Ícones

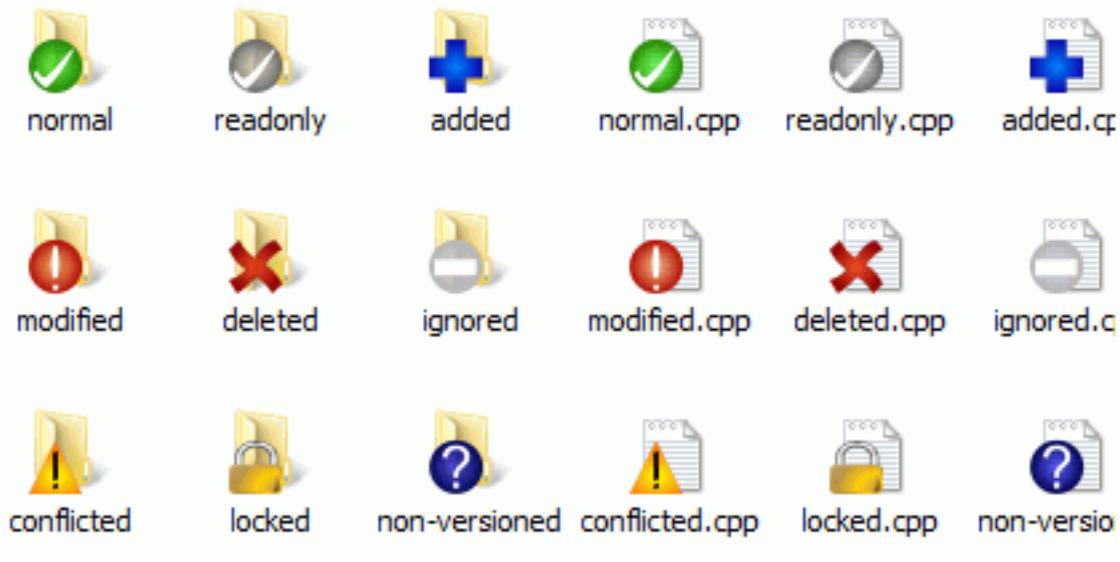


Figura 4.1. Explorer mostra os ícones sobrepostos

Um dos recursos mais visíveis do TortoiseSVN é o de sobreposição de ícones que aparece em arquivos em sua cópia de trabalho. Eles te mostram em um instante quais de seus arquivos foram modificados. Consulte Seção 4.7.1, “Sobreposição dos Ícones” para descobrir o que as diferentes sobreposições representam.

4.1.2. Menus do Contexto

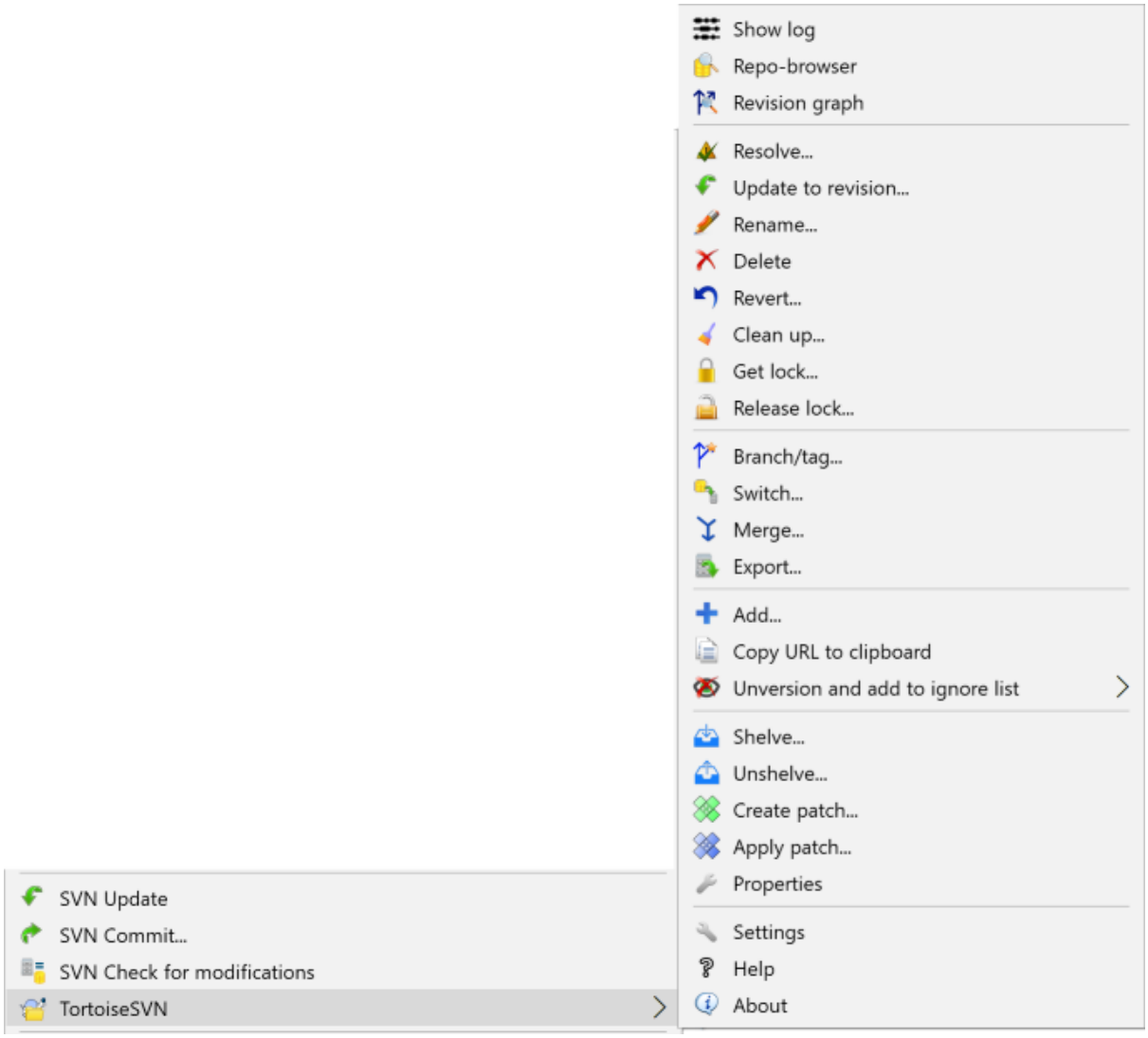


Figura 4.2. Menu do contexto para diretórios controlados

Todos os comandos do TortoiseSVN são acessados através do menu de contexto do Windows Explorer. A maioria são diretamente visíveis quando se clica com o botão direito em um arquivo ou diretório. Os comandos estarão disponíveis se o arquivo ou o diretório que o contém está no controle de versão. Você também pode acessar o menu do TortoiseSVN como parte do menu Arquivo, no Windows Explorer.

Dica

Alguns comandos que são raramente usados estão disponíveis apenas em no menu de contexto extendido. Para ver o menu de contexto extendido, segure a tecla **Shift** quando você clicar com o botão direito.

Em alguns casos você poderá ver várias entradas do TortoiseSVN. Isto não é um bug!

Figura 4.3. Atalho no menu arquivo do Explorer em um diretório controlado

Este exemplo é de um atalho não-versionado dentro de um diretório versionado, e no menu Arquivo do Windows Explorer há *três* entradas para o TortoiseSVN. Uma para o diretório, uma para o atalho, e uma terceira para o objeto para o qual o atalho aponta. Para ajudar a distinguir entre elas, os ícones um indicador no canto inferior direito para mostrar se a entrada se refere a um arquivo, um diretório, um atalho ou para múltiplos ítens selecionados.

4.1.3. Arrastar e Soltar

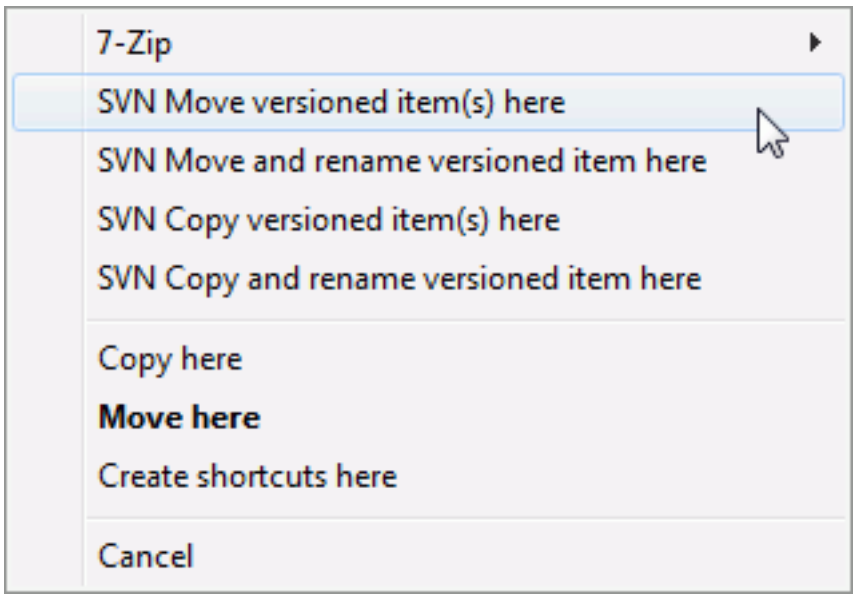


Figura 4.4. Menu de quando se clica com o botão direito e se arrasta um diretório que está sob o controle de versão.

Outros comandos estão disponíveis ao se clicar e arrastar, quando você clica com o botão direito e arrasta arquivos ou diretórios para um novo endereço dentro da cópia de trabalho ou quando você clica com o botão direito e arrasta um arquivo ou diretório não versionado para um diretório sob o controle de versão.

4.1.4. Atalhos Comuns

Algumas operações comuns do Windows tem atalhos bem conhecidos, mas não aparecem em botões e menus. Se você não sabe como fazer algo óbvio como atualizar uma visualização, cheque aqui.

F1
: Ajuda, claro.

F5
: Atualiza a visualização atual. Este talvez seja o mais útil comando de uma única tecla. Por exemplo... No Explorer ele irá atualizar os ícones sobrepostos na sua cópia de trabalho. Na janela de Commit irá reexplorar a Cópia de Trabalho para ver o que precisa ser enviado. Na janela Revision Log irá verificar o repositório novamente a procura de novas modificações recentes.

Ctrl-A
: Seleciona tudo. Pode ser usado se você tiver uma mensagem de erro e precisar copiar e colar em um email. Use Ctrl-A para selecionar a mensagem e então...

Ctrl-C
: Copy the selected text. In case no text is selected but e.g. a list entry or a message box, then the content of that list entry or the message box is copied to the clipboard.

4.1.5. Autenticação

Se o repositório que você está tentando acessar é protegido por senha, uma janela de autenticação aparecerá.

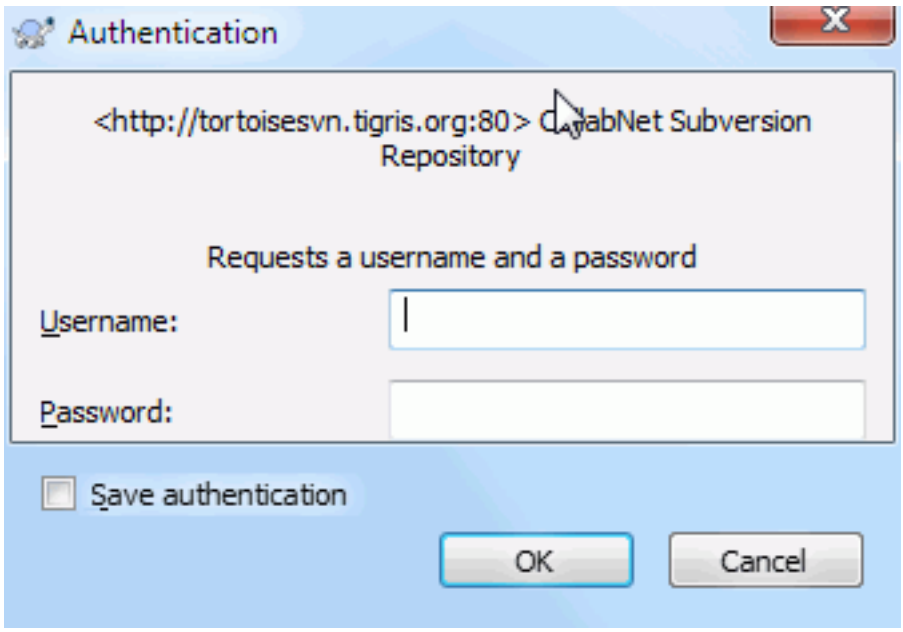


Figura 4.5. Janela de Autenticação

Entre seu nome de usuário e senha. A caixa de seleção (checkbox) fará com que o TortoiseSVN guarde suas credenciais no diretório padrão do Subversion. `%APPDATA%\Subversion\auth` na árvore de subdiretórios:

- `svn.simple` contém as credenciais básicas de autenticação (nome do usuário/senha). Observe que as senhas são armazenadas e encriptadas com o uso do WinCrypt API, não são armazenadas no formato "texto".
- `svn.ssl.server` contém os certificados do servidor para SSL.
- `svn.username` contém credenciais para autenticações que necessitem apenas do nome de usuário (não é necessário senha).

If you want to clear the authentication cache, you can do so from the Saved Data page of TortoiseSVN's settings dialog. The button Clear all will clear the cached authentication data for all repositories. The button Clear... however will show a dialog where you can chose which cached authentication data should be deleted. Refer to Seção 4.31.6, “Dados de configuração salvos”.

Some people like to have the authentication data deleted when they log off Windows, or on shutdown. The way to do that is to use a shutdown script to delete the `%APPDATA%\Subversion\auth` directory, e.g.

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

You can find a description of how to install such scripts at http://www.windows-help-central.com/windows-shutdown-script.html.

Para mais informações sobre como configurr seu servidor para autenticação e controle de acesso, verifique Seção 3.5, “Acessando o Repositório”.

4.1.6. Maximizando Janelas

Muitas das janelas do TortoiseSVN contém muita informação para mostrar, mas é frequentemente útil maximizar apenas a altura ou apenas a largura da janela, ao invés de maximizar na tela toda. Por conveniência, há atalhos

para isto no botão Maximizar. Use o botão do meio do mouse para maximizar verticalmente, e o botão direito do mouse para maximizar horizontalmente.

4.2. Importando Dados Para Um Repositório

4.2.1. Importar

Se você está importando dados para um repositório existente que já contém algum projeto, a estrutura desse repositório já deverá estar definida. Se você está importando dados para um novo repositório, recomenda-se que dispense algum tempo para definir como esse repositório será organizado. Leia Seção 3.1.5, “Leiaute do Repositório” para recomendações adicionais.

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

- Não há qualquer maneira de selecionar os arquivos e pastas que deverão ser incluídos, em compensação poderá utilizar as configurações globais para ignorar alguns arquivos e pastas.
- A pasta importada não se torna uma cópia de trabalho. Você tem que fazer um checkout para copiar os arquivos desde o servidor.
- É fácil importar para uma pasta equivocada no repositório.

For these reasons we recommend that you do not use the import command at all but rather follow the two-step method described in Seção 4.2.2, “Importando na Pasta”, unless you are performing the simple step of creating an initial `/trunk /tags /branches` structure in your repository. Since you are here, this is how the basic import works ...

Antes de importar seu projeto para um repositório, você deveria:

1. Remova todos os arquivos que não são necessários para construir o projeto (arquivos temporários, arquivos que são gerados por um compilador como por exemplo *. obj, binários compilados, ...)
2. Organize os arquivos em pastas e sub-pastas. Embora posteriormente seja possível renomear/mover arquivos, é altamente recomendado que obtenha a estrutura correta do seu projeto antes de realizar a importação!

Agora selecione a pasta de nível superior da sua estrutura de diretório do projeto no Windows Explorer e com o botão direito clique para abrir o menu de contexto. Selecione o comando TortoiseSVN → Importar ...

In this dialog you have to enter the URL of the repository location where you want to import your project. It is very important to realise that the local folder you are importing does not itself appear in the repository, only its content. For example if you have a structure:

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

and you import `C:\Projects\Widget` into `http://mydomain.com/svn/trunk` then you may be surprised to find that your subdirectories go straight into `trunk` rather than being in a `Widget` subdirectory. You need to specify the subdirectory as part of the URL, `http://mydomain.com/svn/trunk/Widget-X`. Note that the import command will automatically create subdirectories within the repository if they do not exist.

A mensagem de importação é usada como uma mensagem para o histórico.

Por padrão, arquivos e pastas que correspondem aos padrões globais de pastas/arquivos ignorados, *não* são importados. Para contornar esse comportamento você pode utilizar o recurso Incluir arquivos ignorados.

Consulte Seção 4.31.1, “Configurações Gerais” para obter mais informações sobre como definir um padrão global para ignorar pastas/arquivos.

Assim que clicar em OK TortoiseSVN inicia a importação da "árvore" de diretórios, incluindo todos os arquivos. O projeto estará armazenado no repositório sob controle de versão. Por favor note que a pasta que você importou *NÃO* está sob controle de versão! Para obter uma *cópia de trabalho* com versão controlada, você deverá fazer um Checkout da versão que você acabou de importar. Ou continue lendo sobre como proceder em "Importando na Pasta".

4.2.2. Importando na Pasta

Assumindo que você já possui um repositório e que deseja adicionar uma nova pasta na estrutura desse repositório, apenas siga estes passos:

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

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

3. Use TortoiseSVN → Adicionar... na pasta sob controle de versão para adicionar algum ou todo conteúdo. Você pode adicionar e remover arquivos, defina as configurações em `svn:ignore` para as pastas e faça outras alterações que você necessita.

4. Ao submeter a pasta de nível superior, você terá uma nova "árvore" sob controle de versão e uma cópia local de trabalho, criada a partir de sua pasta existente.

4.2.3. Arquivos Especiais

Às vezes, você precisa ter um arquivo sob controle de versão que contém dados específicos de usuário. Isso significa que você terá um arquivo que todo desenvolvedor/usuário precisará modificar para adequá-lo a sua configuração local. Portanto, é difícil manter um arquivo sob controle de versão porque cada usuário poderá submeter alterações no repositório constantemente.

Nesses casos, sugere-se a utilização de um arquivo *modelo*. Crie um arquivo que contém todos os dados que seus desenvolvedores necessitam, adicione e mantenha esse arquivo sob controle de versão e permita que os desenvolvedores façam o Checkout desse arquivo. Em seguida, cada desenvolvedor deverá fazer *uma cópiaemphasis> desse arquivo e renomear essa c*

As an example, you can have a look at TortoiseSVN's build script. It calls a file named `default.build.user` which doesn't exist in the repository. Only the file `default.build.user.tmpl`. `default.build.user.tmpl` is the template file which every developer has to create a copy from and rename that file to `default.build.user`. Inside that file, we added comments so that the users will see which lines they have to edit and change according to their local setup to get it working.

So as not to disturb the users, we also added the file `default.build.user` to the ignore list of its parent folder, i.e. we've set the Subversion property `svn:ignore` to include that filename. That way it won't show up as unversioned on every commit.

4.3. Obtendo Uma Cópia de Trabalho

Para obter uma cópia de trabalho você precisa realizar um *obter* do repositório.

Selecione um diretório no windows explorer onde você quer deixar sua cópia de trabalho. Clique direito para abrir o menu de contexto e então selecione o comando TortoiseSVN → Obter..., o qual abrir a seguinte janela:

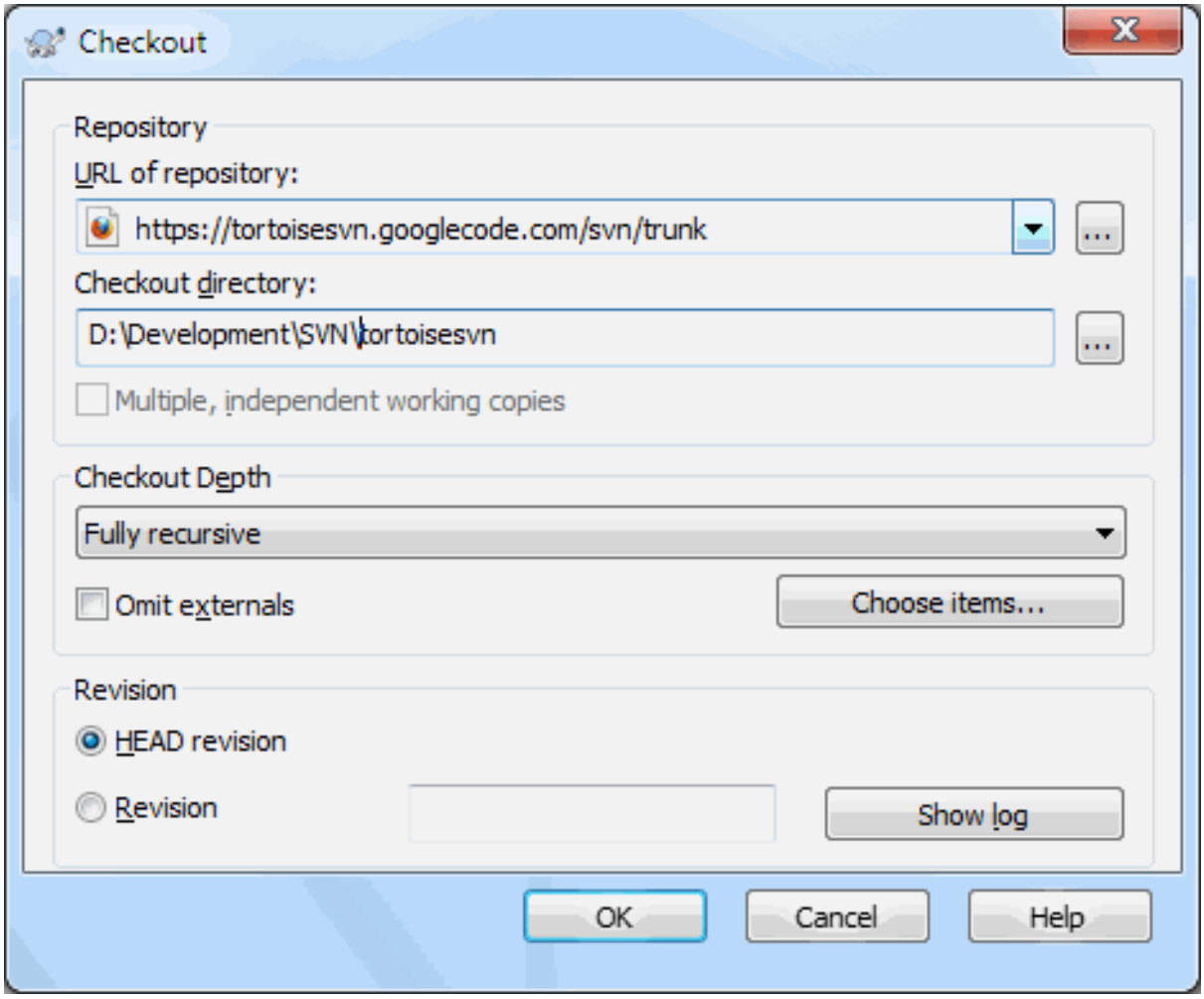


Figura 4.6. A janela de Obtenção

. Se você digitar o nome de um diretório que não existe ainda, então um diretório com este nome será criado.

Importante

In the default setting, the checkout menu item is not located in the TortoiseSVN submenu but is shown at the top explorer menu. TortoiseSVN commands that are not in the submenu have SVN prepended: SVN Checkout...

4.3.1. Profundidade da Obtenção

Você pode escolher a *profundidade* que você quer obter, o qual permitirá a você especificar a profundidade da recursão nos sub diretórios. Se você quer apenas algumas seções de uma grande hierarquia, você pode obter o nível mais alto apenas, e depois atualizar pastas específicas recursivamente.

Totalmente recursiva
: Obter estrutura completa, incluindo todos os níveis de sub diretórios.

Filhos imediatos, incluindo pastas
: Obter o diretório especificado, incluindo todos os arquivos e sub diretórios, mas não carregar os sub diretórios.

Somente os arquivos filhos
: Ober o diretório especificado, incluindo todos os arquivos mas não obter qualquer sub diretório.

Somente este item
: Obter apenas o diretório. Não carregar com arquivos ou sub diretórios.

Cópia de trabalho
: Retêm a profundidade especificada na cópia de trabalho. Esta opção não é usada na janela de obtenção, mas é o padrão em todas as outras janelas que tem a configuração de profundidade.

Excluir
: Usado para reduzir a profundidade da cópia de trabalho depois de uma pasta já ter sido carregada. Esta opção está somente disponível na janela Atualizar para a revisão.

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

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

4.3.1.1. Sparse Update using Update to Revision

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

4.3.1.2. Update esparso utilizando Repo Browser

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

4.3.1.3. Sparse Update using Check for Modifications

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

Esta funcionalidade é muito útil quando você apenas quer obter partes de uma grande estrutura, mas você quer a conveniência de atualizar uma só cópia de trabalho. Supondo que você tenha uma grande estrutura a qual possui sub diretórios `Projeto01` até `Projeto99`, e você quer apenas obter o `Projeto03`, `Projeto25` e `Projeto76`. Use esses passos:

1. Obter o diretório pai com profundidade “Somente este item”. Você agora tem um diretório do nível mais alto.

2. Selectionar um novo diretório e usar TortoiseSVN → Navegador de repositório para mostrar o conteúdo do repositório.

3. Clique direito sobre `Projeto03` e Menu de Contexto → Atualizar item para a revisão.... Deixe as configurações padrões e clique em OK. Você agora terá um diretório carregado.

   Repetir o mesmo processo para `Projeto25`.

4. Navegu para `Projeto76` e faça a mesma coisa. Agora note que o diretório `Projeto76` não tem nada exceto o `SubProj`, o qual ele mesmo está carregado. Subversion criou os diretórios intermediários para você sem carregá-los.

Mudando a profundidade da cópia de trabalho

Uma vez que você obteve uma cópia de trabalho para uma profundidade particular você pode mudar a profundidade posteriormente para obter mais ou menos arquivos usando Menu de Contexto → Atualizar item para a revisão.... Nesta janela, tenha certeza que a opção Deixar a profundidade colada marcada.

Usando um servidor antigo

Servidores pré-1.5 não entendem a requisição de profundidade de uma cópia de trabalho, então eles nem sempre responderão esse pedido de forma eficiente. O comando continuará funcionando, mas servidores mais antigos poderão enviar todos os dados, deixando o cliente para filtrar o que não é requerido, o que significa mais tráfego de rede. Se possível você deverá atualizar seu servidor para uma versão mais nova que 1.5.

Se o projeto contém referências para projetos externos os quais você *não* quer obter no mesmo momento, use a caixa de seleção Omitir externos.

Importante

Se Omitir externos está mrcado, ou se você quer incrementar a profundidade, você precisa executar a atualização da sua cópia de trabalho usando TortoiseSVN → Atualizar para a revisão...ao invés de TortoiseSVN → Atualizar. A atualização padrão incluirá todos os externos e manterá a profundidade configurada.

É recomendado que você obtenha apenas uma parte da estrutura de diretório do `trunk`, ou mais abaixo. Se você especificar o diretório principal da estrutura de diretório da URL você deverá acabar com o disco cheio já que uma vez que você obterá uma cópia da estrutura do repositório inteiro incluindo cada ramo e rótulo do seu projeto.

Exportando

Algumas vezes você vai querer criar uma cópia local sem nenhum dos diretórios `.svn`, ex. criar um arquivo compactado do código fonte. Leia Seção 4.27, “Exportando um Cópia de Trabalho do Subversion” para saber como fazer isto.

4.4. Submetendo Suas Alterações Para o Repositório

Enviar as alterações que você fez em sua cópia de trabalho é conhecido como *Submeter* as alterações. Mas antes de você submeter você deve ter certeza que sua cópia de trabalho está atualizada. Você pode usar tanto TortoiseSVN → Update diretamente. Ou você pode usar TortoiseSVN → Check for Modifications primeiro, para ver quais arquivos foram alterados localmente ou no servidor.

4.4.1. A Janela de Submissão

Se sua cópia de trabalho estiver atualizada e não houver conflitos, você está pronto para submeter suas alterações. Selecione qualquer arquivo e/ou pastas que você quiser submeter, então TortoiseSVN → Commit....

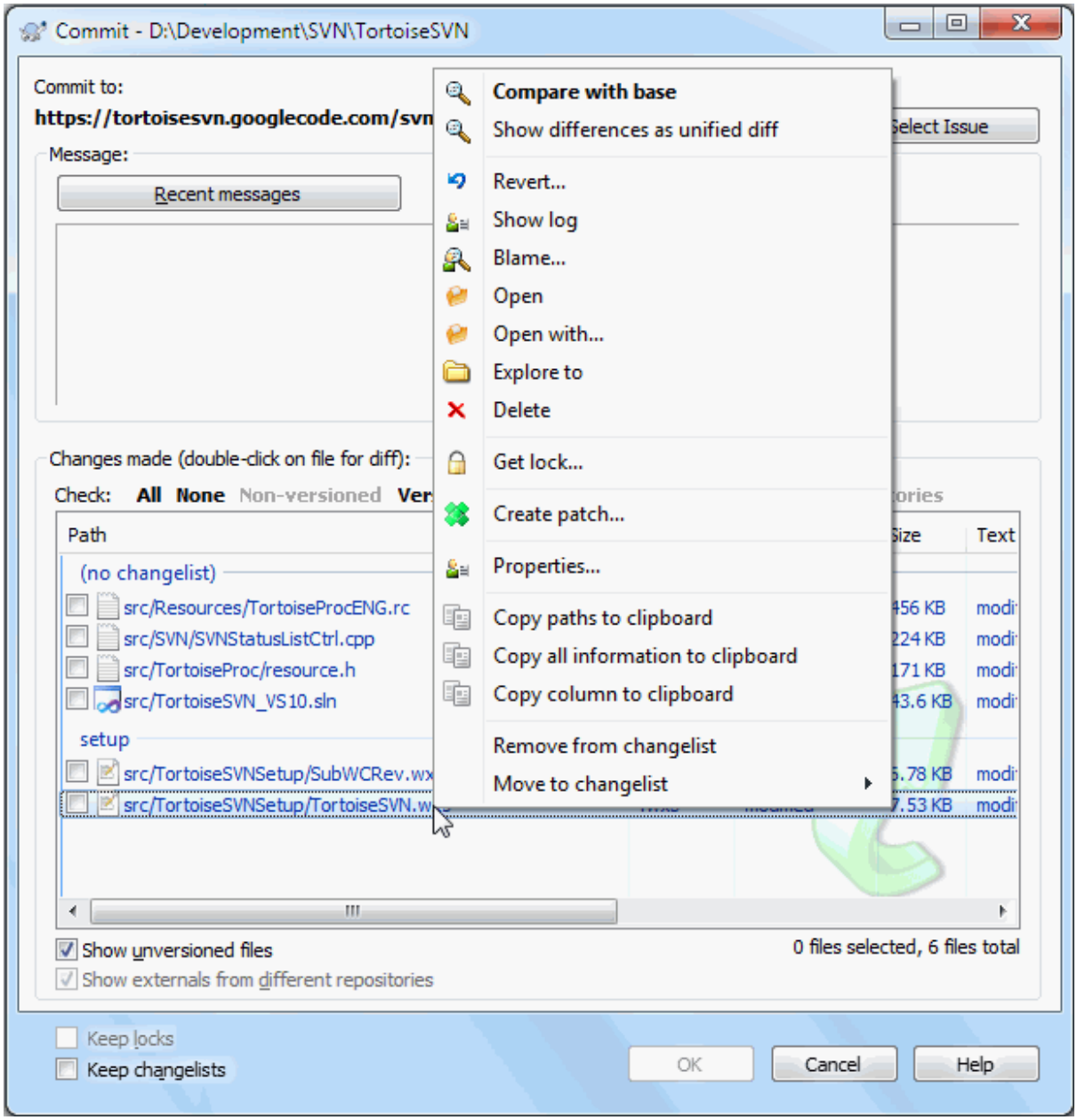


Figura 4.7. A janela de Submissão

A janela de submissão irá exibir todos os arquivos modificados, incluindo arquivos adicionados, deletados e não versionados. Se você não quiser que um arquivo alterado seja submetido, apenas desmarque o arquivo. Se você quiser incluir um arquivo não versionado, apenas marque o arquivo para adiciona-lo a submissão

To quickly check or uncheck types of files like all versioned files or all modified files, click the link items just above the list of shown items.

For information on the coloring and overlays of the items according to their status, please see Seção 4.7.3, “Estado Local e Remoto”.

Items que foram movidos para um caminho de repositório diferente são também indicados usando o marcador `(s)`. Você pode ter movido algo enquanto trabalhava em uma ramificação (branch) e se esqueceu de mover de volta para a versão principal (trunk). Esté é seu sinal de aviso!

Submeter arquivos ou diretórios?

Quando você submete arquivos, a janela de submissão exibe apenas os arquivos que você selecionou. Quando você submete uma pasta, a janela de submissão irá exibir selecionado os arquivos modificados automaticamente. Se você se esqueceu de um novo arquivo que criou, submeter a pasta irá encontra-lo de qualquer forma. Submeter uma pasta *não* significa que todos os arquivos serão marcados como modificados; Isso apenas fará sua vida mais fácil.

Muitos arquivos não versionados na janela de submissão

Se você acha que a janela de submissão exibe muitos arquivos não versionados (arquivos gerados pelo compilador ou backups), existem muitas formas de lidar com isso. Você pode

- adicionar o arquivo (ou um caracter especial) para listar os arquivos para excluir nas configurações da página. Isto afetará toda a cópia de trabalho que você tem.
- adicionar o arquivo para a lista `svn:ignore` usando TortoiseSVN → Adicionar à lista de ignorados. Isto somente afetará o diretório no qual você configurou a propriedade `svn:ignore`. Usando a janela Propriedade do SVN, você pode alterar a propriedades `svn:ignore` para um diretório.
- add the file to the `svn:global-ignores` list using TortoiseSVN → Add to ignore list (recursively) This will affect the directory on which you set the `svn:global-ignores` property and all subfolders as well.

Read Seção 4.14, “Ignorando Arquivos e Diretórios” para mais informações

Click duplo em qualquer arquivo modificado na janela de submissão irá abrir um programa de diff exibindo suas modificações. O menu de contexto irá dar mais opções, como exibido na screenshot. Você também arrastar arquivos da janela de submissão para outros aplicativos como editores de textos ou IDEs.

Você pode selecionar ou deselecionar items clicando na caixa de seleção a esquerda do item. Para diretórios você pode usar **Shift**-select para fazer a ação recursiva.

As colunas mostradas no painel inferior são personalizadas. Se você clicar com o botão direito em qualquer cabeçalho de coluna você virá um menu de contexto que permitirá a você selecionar quais colunas você quer ver. Você pode também mudar o tamanho da coluna usando a opção de arrasto que aparece quando você mantém o ponteiro do mouse sobre a borda da coluna. Estas personalizações são gravadas, etnão você verá a tela do mesmo jeito quando voltar.

Por padrão quando você submete alterações, qualquer travamento que você determinou em arquivos serão liberados automaticamente após a confirmação da submissão. Se você quer manter as alocações, tenha certeza que a opção Manter bloqueios está selecionada. O padrão desta caixa de seleção é obtido da opção `no_unlock` do arquivo de configuração do Subversion. Leia Seção 4.31.1, “Configurações Gerais” para mais informações sobre como editar o arquivo de configuração do Subversion.

Advertência ao submeter a tag

Usually, commits are done to the trunk or a branch, but not to tags. After all, a tag is considered fixed and should not change.

If a commit is attempted to a tag URL, TortoiseSVN shows a confirmation dialog first to ensure whether this is really what is intended. Because most of the time such a commit is done by accident.

However, this check only works if the repository layout is one of the recommended ones, meaning it uses the names `trunk`, `branches` and `tags` to mark the three main areas. In case the

setup is different, the detection of what is a tag/branch/trunk (also known as `classification patterns`), can be configured in the settings dialog: Seção 4.31.2, “Configurações do Gráfico de Revisões”

Arrastar e Soltar

Você pode arrastar arquivos para dentro da janela de submissão de qualquer lugar, desde que as cópias de trabalho tenham sido obtidas do mesmo repositório. Por exemplo, você pode ter uma grande cópia de trabalho com inúmeras janelas do windows explorer abertas para olhar em pastas em diferentes lugares. Se você quiser evitar submeter do nível mais alto da estrutura de pastas (com uma longa lista de modificações) você pode abrir a janela de submissão para uma pasta e então arrastar os itens de outras janelas para incluir na mesma submissão atômica.

Você pode arrastar arquivos não versionados que residem em uma cópia de trabalho para uma janela de submissão e eles irão ser adicionados ao SVN automaticamente.

Arrastando arquivos da lista na parte de baixo da janela de submissão para o campo para edição da mensagem de log irá inserir os caminhos como texto dentro da caixa de edição. Isto é muito útil se você quer escrever mensagens de log da submissão que incluem os caminhos que foram afetados pela submissão.

Reparando Referência Externas Renomeadas

Algumas vezes arquivos são renomeados fora do Subversion, e eles aparecem na lista de arquivos como arquivos perdidos e como arquivos não versionados. Para evitar a perca do histório você precisa notificar o Subversion sobre a conexão entre os arquivos. Simplesmente selecione o arquivo antigo (perdido) e o arquivo nome (não versionado) e use a opção Menu de Contexto → Reparar movimento para identificar os dois arquivos como uma renomeação.

Reparando Cópias de Referências Externas

Se você fez uma cópia de um arquivo e esqueceu de usar o comando do SVN na cópia, você pode reparar a cópia e assim o novo arquivo não perderá seu histórico. Simplesmente selecione o arquivo antigo (normal ou modificado) e o nome arquivo (não versionado) e use Menu de Contexto → Reparar Cópia para identificar os dois arquivos como um cópia.

4.4.2. Lista de Alterações

A janela de submissão suporta a funcionalidade lista de alterações do Subversion para ajudar a agrupar arquivos com modificações relacionadas. Saiba mais sobre essa funcionalidade em Seção 4.8, “Lista de Alterações”.

4.4.3. Enviar somente partes dos arquivos

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

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

Usando TortoiseMerge

If you use TortoiseMerge to edit the file, you can either edit the changes as you're used to, or mark all the changes that you want to include. right click on a modified block and use Context Menu

> → Mark this change to include that change. Finally right click and use Context Menu → Leave only marked changes which will change the right view to only include the changes you've marked before and undo the changes you have not marked.

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

4.4.4. Excluindo Itens de uma Lista de Submissões

Agumas vezes você tem arquivos versionados que são alterados frequentemente mas que você de verdade não quer controlar. Algumas vezes isto indica uma falha no seu processo de desenvolvimento - por que esses arquivos são controlados? deveriam ser arquivos modelos? Mas ocasionalmente isto é inevitável. Uma razão clássica é que seu IDE muda a data/hora do arquivo em seu projeto cada vez que você compila. O arquivo do projeto tem que ser controlado já que ele inclui todas as configurações de compilação, mas você não precisa submeter apenas porque mudou a data/hora do arquivo.

Para ajudar nessas situações incomuns, nós temos uma lista de alterações específica chamada `ignore-on-commit`. Qualquer arquivo adicionado neste lista de alterações ficará automaticamente desmarcado na janela de submissão. Você pode ainda submeter as alterações, mas você terá que selecionar manualmente na janela de submissão.

4.4.5. Registro de Mensagens de Submissão

Tenha a certeza de digitar uma mensagem de log que descreva as aterações que você está submetendo. Isto ajudará você a ver o que aconteceu e quando, ao navegador pelas mensagens de auditoria do projeto dias mais tarde. A mensagem pode ser tanto longa quanto curta; muitos projetos tem diretivas para o que será incluido, o idioma a ser usado, e algumas vezes até a formatação exigida.

Você pode formatar de forma simples sua mensagem de log usando a convenção similar ao que é usado nos e-mails. Para aplicar estilos para `texto`, use `*texto*` para negrito, `_texto_` para sublinhado, e `^texto^` para itálico.

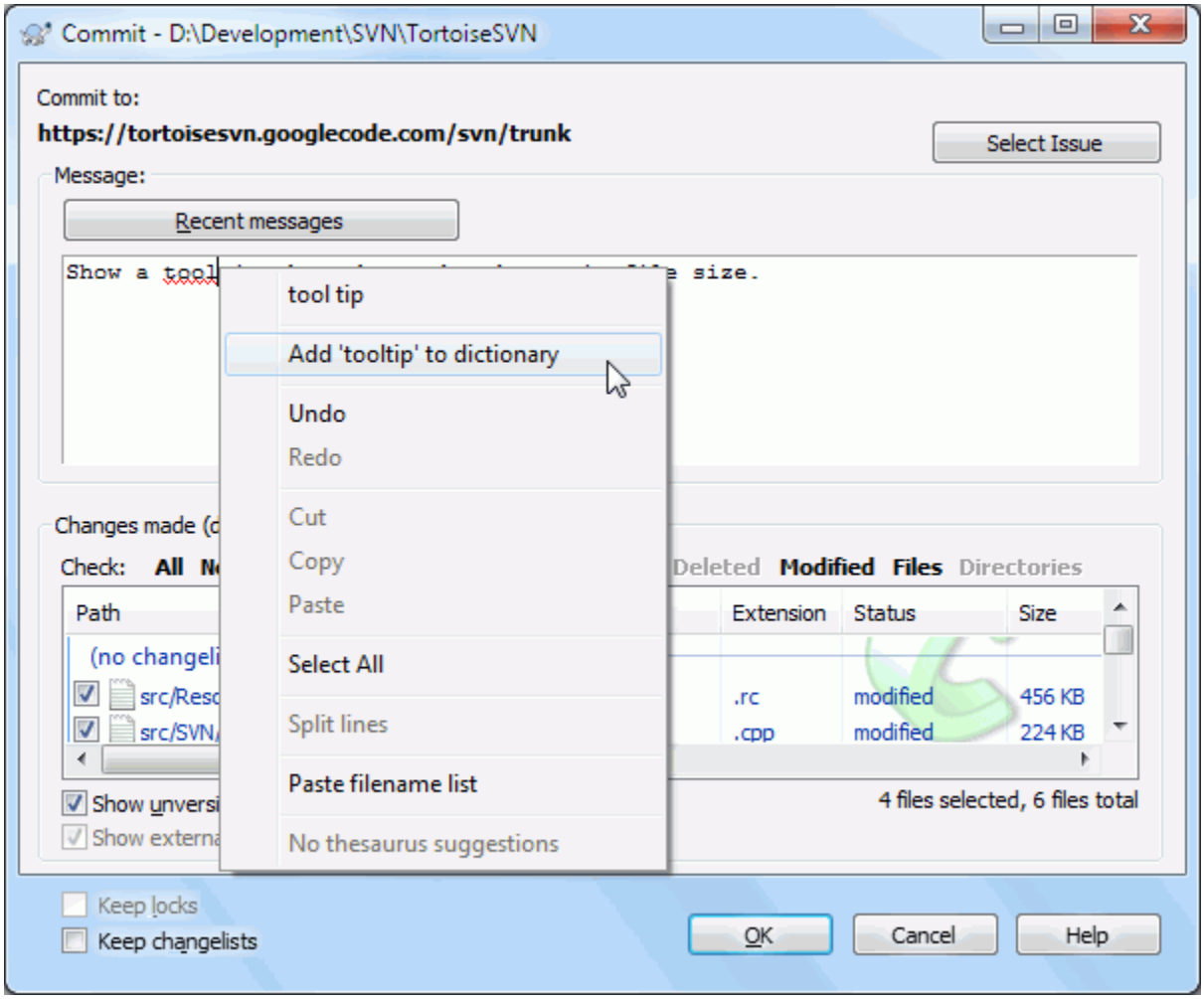


Figura 4.8. A Janela de Submissão Com Corretor Ortográfico

TortoiseSVN inclui um corretor ortográfico para lhe ajudar a escrever a sua mensagem de log corretamente. Ele irá descatar palavras erradas. Use o menu de contexto para acessar as sugestões de correção. É claro, ele não conhece *todos* os termos técnicos que você conhece, então palavras digitadas corretamente irão algumas vezes aparecer como erro. Mas você não deve se preocupar. Você pode simplesmente adicionar a palavra ao seu dicionário pessoal usando o menu de contexto.

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 https://en.wikipedia.org/wiki/Regular_expression, and the online documentation and tutorial at http://www.regular-expressions.info/.

Conseguir a regex certa pode ser confuso, mas para ajudar você a criar uma expressão que funcione, há uma janela de testes que permite que você entre uma expressão e então digite nomes de arquivo para testar. Inicie ela da janela de comando usando o comando `TortoiseProc.exe /command:autotexttest`.

The log message window also includes a commit message snippet facility. These snippets are shown in the autocomplete dropdown once you type a snippet shortcut, and selecting the snippet in the autocomplete dropdown then inserts the full text of the snippet. The snippets supplied with TortoiseSVN are held in the TortoiseSVN installation `bin` folder. You can also define your own snippets and store them in `%APPDATA%\TortoiseSVN`

`\snippet.txt`. # is the comment character. Newlines can be inserted by escaping them like this: `\n` and `\r`. To insert a backslash, escape it like this: `\\`.

Voce pode reutilizar mensagens já digitados. Apenas clique sobre Mensagens recentes para ver a lista das últimas mensagens digitas para a cópia de trabalho. O número de mensagens gravadas pode ser configurado na janela de configurações do TortoiseSVN.

Você pode apagar as mensagens de submissão gravadas usando a tela Dados salvos das configurações do TortoiseSVN, ou você pode apagar as mensagens uma a uma usnado a janela Mensagens recentes usando a tecla **Delete**.

Se você quer incluir os caminhos verificados em sua mensagem de log, você pode usar o comando Menu de Contexto → Colar lista de nomes de arquivos no campo da mensagem.

Outra forma de inserir caminhos em uma mensagem de auditoria é simplesmente arrastas os arquivos da lista de arquivos para dentro do campo da mensagem.

Propriedades de Pastas Especiais

Existem muitas propriedades especiais nos diretórios que podem ser usados para dar mais controle sobre a formatação das mensagens de log na submissão e o idioma usado pelo módulo de correção ortográfica. Leia Seção 4.18, “Configurações do Projeto” para mais informações.

Integração com Ferramentas de Rastreamento de Erros

se você ativou o sistema de rastreamento de erros, você pode configurar um ou mais Controles no campo do texto Bug-ID / Issue-Nr:. Multiplos controles devem ser separados com virgula. Uma alternativa, se você está usando um controle de rastreamento de erros baseado em expressões regulares, você pode adicionar a referência ao seu controle como parte da mensagem de log. Aprenda mais sobre isso em Seção 4.29, “Integração com Sistemas de Rastreamento de Erros / Rastreadores de Reportes”.

4.4.6. Progresso da Submissão

Depois de pressionar OK, a janela aparece mostrando o progresso da submissão.

Figura 4.9. A janela de Progresso mostra uma submissão em progresso

A janela de progresso usa cores por código para destacar diferentes ações de submissão

Azul
: Submetendo uma modificação.

Roxo
: Submetendo uma nova adição.

Vermelho escuro
: Submetendo um exclusão ou substituição.

Preto
: Todos os outros itens.

Este é o esquema padrão de cores, mas você pode personalizar as cores usando a janela de configurações. Acesse Seção 4.31.1.5, “Configurações de Cores do TortoiseSVN” para mais informações.

4.5. Atualizar sua Cópia de Trabalho com mudanças feitas por outros

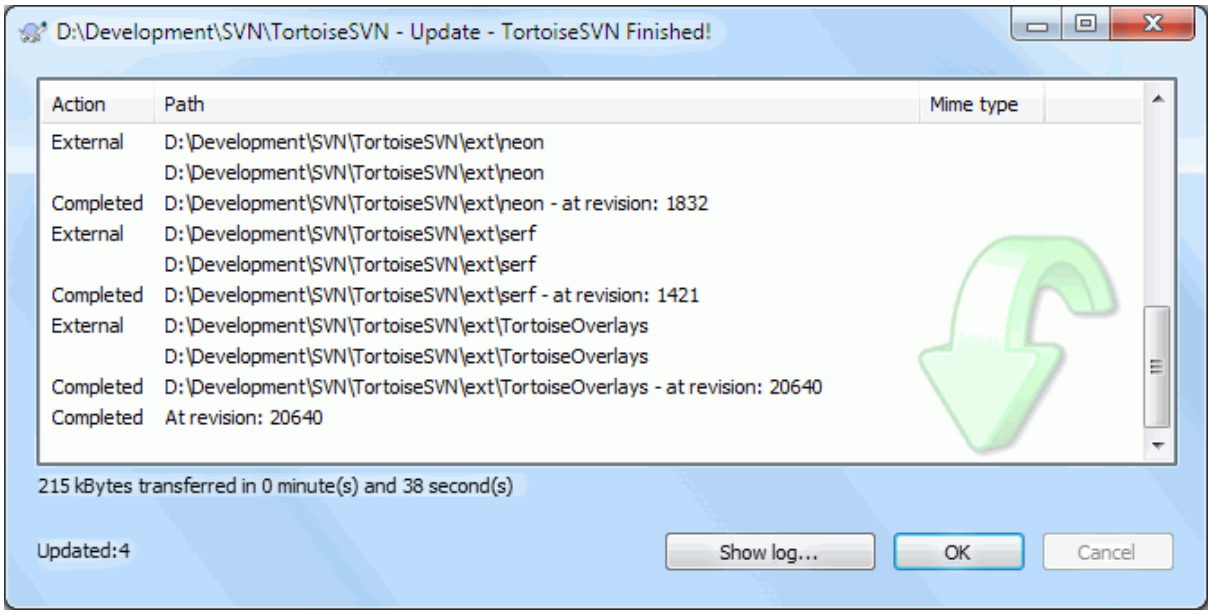


Figura 4.10. Janela de progresso mostrando atualização terminada

Periodicamente, você deve garantir que mudanças feitas por outros sejam incorporadas na sua cópia de trabalho. O processo de pegar as mudanças do servidor para sua cópia local é conhecido como *atualização*. A atualização pode ser feita em arquivos, uma seleção de arquivos, ou recursivamente na herarquia inteira de diretórios. Para atualizar, selecione os arquivos e/ou diretórios que quiser, clique com o botão direito e selecione TortoiseSVN → Atualizar no menu de contexto do explorer. Uma janela aparecerá mostrando o progresso da atualização em tempo real. As mudanças feitas por outros serão consolidadas com seus arquivos, mantendo quaisquer mudanças que você possa ter feito nos mesmos arquivos. O repositório *não* é afetado por uma atualização.

A janela de progresso faz uso de código de cores para sublinhar diferentes ações de atualização

Roxo
: Novo item adicionado à CT.

Vermelho escuro
: Item redundante apagado da sua CT, ou item substituído na sua CT.

Verde
: Mudanças no repositório consolidadas com suas mudanças locais com sucesso

Vermelho vivo
: Mudanças no repositório consolidadas com mudanças locais, resultando em conflitos que você precisa resolver.

Preto
: Item inalterado em sua CT atualizado com nova versão do repositório.

Este é o esquema padrão de cores, mas você pode personalizar as cores usando a janela de configurações. Acesse Seção 4.31.1.5, “Configurações de Cores do TortoiseSVN” para mais informações.

Se você receber quaisquer *conflitos* durante uma atualização (isso pode acontecer se outras pessoas alteraram as mesmas linhas no mesmo arquivo que você e essas mudanças não são compatíveis) então a janela mostra esses conflitos em vermelho. Você pode efetuar um clique duplo nessas linhas para começar a ferramenta consolidação externa para resolver esses conflitos.

Quando a atualização estiver completa, a janela de progresso mostrará um sumário do número de itens atualizados, adicionados, removidos, em conflito, etc. abaixo da lista de arquivos. Esta informação pode ser copiada para a área de transferência usando **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 Seção 4.3.1, “Profundidade da Obtenção”. The default depth is Working copy, which preserves the existing depth setting. You can also set the depth `sticky` which means subsequent updates will use that new depth, i.e. that depth is then used as the default depth.

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

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

Cuidado

Se você atualizar um arquivo ou pasta para uma versõa específica, você não deve fazer mudanças nesses arquivos. Você receberá mensagens de erro do tipo “desatualizados” quando tentar submetê-las! Se você quiser desfazer mudanças feitas em um arquivo e começar novamente de uma versão anterior, você pode reverter para uma versão anterior a partir da janela de log de revisões. Dê uma olhada em Seção B.4, “Reverter (Desfazer) revisões no repositório” para mais instruções, e métodos alternativos.

Atualizar para Revisão pode ser ocasionalmente útil para ver como seu projeto parecia em um momento anterior de sua história. Mas de modo geral, atualizar arquivos individualmente para uma versão anterior não é uma boa idéia porque deixa sua cópia de trabalho em estado inconsistente. Se o arquivo que você está atualizando mudou de nome, você pode até descobrir que o arquivo simplesmente desaparece da sua cópia de trabalho porque não havia arquivo com aquele nome na revisão anterior. Você também deve notar que o item terá umrevestimento verde normal, para que seja indistinguível dos arquivos que estão atualizados com a última versão.

Se você simplesmente quer uma cópia local de uma versão anterior de um arquivo é melhor usar o comando Menu de Contexto → Salvar revisão para... a partir da janela de log daquele arquivo.

Múltiplos Arquivos/Diretórios

Se você selecionar múltiplos arquivos e pastas no explorer e depois selecionar Atualizar, todos estes arquivos/pastas serão atualizados um a um. O TortoiseSVN garantirá que todos arquivos/pastas que

> são do mesmo repositório serão atualizados para a mesma revisão! Mesmo se entre estas atualizações uma outra submissão tenha ocorrido.

4.6. Resolvendo Conflitos

Em algum momento você poderá se deparar com um *conflito* quando for atualizar/unificar seus arquivos desde o repositório ou quando você for modificar sua cópia local para uma URL diferente. Existem dois tipos de conflitos:

conflitos de arquivo
: Um conflito de arquivo ocorre se dois (ou mais) desenvolvedores modificam as mesmas linhas de um arquivo.

conflitos de estrutura
: Um conflito de estrutura ocorre quando um desenvolvedor move/renomeia/elimina um arquivo ou uma pasta que outro desenvolvedor tenha também movido/renomeado/eliminado ou ainda apenas modificado.

4.6.1. Conflitos de Arquivo

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

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

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

filename.ext.mine
: Este é seu arquivo tal como existia em sua cópia local antes de tê-la atualizado - ou seja, sem indicações de conflito. Este arquivo contém as modificações mais recentes e nada mais.

filename.ext.rREVANT
: Este é o arquivo que serviu de revisão BASE antes de atualizar sua cópia local. Ou seja, é o arquivo que você assinalou para controlar antes de efetuar as modificações.

filename.ext.rNOVAREV
: Este é o arquivo que o cliente Subversion acabou de receber do servidor quando você atualizou sua cópia local. Este arquivo corresponde à revisão HEAD do repositório.

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

Em seguida execute o comando TortoiseSVN → Resolvido e submeta suas modificações no repositório. Por favor, note que o comando Resolver não resolve o conflito. Ele apenas remove os arquivos `filename.ext.mine` e `filename.ext.r*`, para permitir que você submeta suas modificações.

Se tiver conflitos com arquivos binários, Subversion não irá incluir esses arquivos (propriamente dito). A cópia local permanece inalterada (exatamente como a modificação mais recente) e você terá arquivos `filename.ext.r*`. Se desejar descartar as modificações e manter a versão do repositório, apenas utilize o comando Reverter. Se desejar manter sua versão e sobreescrever o repositório, utilize o comando Resolvido e submeta sua versão.

Você pode utilizar o comando Resolvido para múltiplos arquivos se clicar com o botão direito do mouse na pasta que contém os arquivos e selecionar TortoiseSVN → Resolvido... Isto fará aparecer uma caixa de diálogo de todos os arquivos em conflito naquela pasta e então poderá selecionar quais deverão ser assinalados como resolvido.

4.6.2. Conflitos de Propriedade

Um conflito de propriedade acontece quando dois ou mais desenvolvedores mudaram a mesma propriedade. Assim como para o conteúdo, resolver conflitos pode ser feito apenas pelos desenvolvedores.

Se uma das alterações deve sobreescrever outra então escolha a opção Resolver usando a propriedade local ou a opção Resolver usando a propriedade remota. Se as alterações devem ser combinadas então selecione Editar a propriedade manualmente, altere o valor da propridade como ela deve ser e marque como resolvido.

4.6.3. Conflitos de Estrutura

Um conflito de estrutura ocorre quando um desenvolvedor move/renomeia/elimina um arquivo ou uma pasta que outro desenvolvedor tenha também movido/renomeado/eliminado ou ainda apenas modificado. Existem diversas situações que podem resultar num conflito de estrutura e cada situação exigirá uma ação diferente para a solução do conflito.

Quando um arquivo é eliminado localmente na Subversão, o arquivo também é eliminado do sistema local de arquivos ou seja, mesmo que seja parte do conflito de estrutura isso não comprova ser uma sobreposição de conflito e você não poderá clicar nele com o botão direito do mouse, para resolver o conflito. Utilize a caixa de diálogo Verificar por Modificações ao invés de acessar a opção Editar conflitos.

TortoiseSVN pode auxiliar para encontrar o lugar correto para aplicar modificações mas poderão existir tarefas adicionais e requeridas para a solução de conflitos. Lembre-se que após uma atualização, a BASE de trabalho sempre conterá a revisão de cada item tal como era no repositório no momento da atualização. Se você reverter uma modificação após uma atualização, o repositório não retornará ao estado original, tal como era antes das modificações que você iniciou localmente.

4.6.3.1. Exclusão local, alteração na atualização

1. Desenvolvedor A modifica `Foo.c` e o submete para o repositório.

2. Desenvolvedor B simultaneamente moveu `Foo.c` para `Bar.c` em sua cópia local ou simplesmente eliminou `Foo.c` ou a pasta que o continha.

Uma atualização da cópia de trabalho do desenvolvedor B resulta num conflito de estrutura:

- `Foo.c` foi eliminado da cópia de trabalho mas está marcado com um conflito de estrutura.

- Se o conflito resulta por ter renomeado ao invés de ter eliminado então `Bar.c` estará marcado como adicionado mas não irá conter as modificações do desenvolvedor A.

Desenvolvedor B agora deverá escolher se mantém ou não as modificações do desenvolvedor A. No caso de um arquivo ter sido renomeado, ele poderá submeter as modificações de `Foo.c` para o arquivo renomeado `Bar.c`. No caso de simples arquivo ou pasta deletada, ele poderá escolher entre manter o item com as modificações do desenvolvedor A e descartar o arquivo eliminado. Ou então, poderá marcar o conflito como resolvido sem tomar qualquer atitude e efeticamente irá descartar as modificações do desenvolvedor A.

The conflict edit dialog offers to merge changes if it can find the original file of the renamed `Bar.c`. If there are multiple files that are possible move sources, then a button for each of these files is shown which allow you to chose the correct file.

4.6.3.2. Alteração local, exclusão na atualização

1. Desenvolvedor A move `Foo.c` para `Bar.c` e submete para o repositório.

2. Desenvolvedor B modifica `Foo.c` na sua cópia de trabalho.

Ou no caso de um diretório movido ...

1. Desenvolvedor A move a pasta que o continha de `FooFolder` para `BarFolder` e submete para o repositório.

2. Desenvolvedor B modifica `Foo.c` na sua cópia de trabalho.

Uma atualização na cópia de trabalho do desenvolvedor B resulta num conflito de estrutura. Para um simples conflito de arquivo:

- `Bar.c` é adicionado na cópia de trabalho como um arquivo normal.
- `Foo.c` é assinalado como adicionado (com histórico) e indica um conflito de estrutura.

Para um conflito de pasta:

- `BarFolder` é adicionada para a cópia de trabalho como uma pasta normal.
- `FooFolder` é assinalada como adicionada (com histórico) e indica um conflito de estrutura.

  `Foo.c` é assinalado como modificado.

Desenvolvedor B agora terá que decidir o que fazer com a reorganização do desenvolvedor A e submeter suas modificações para o arquivo correspondente na nova estrutura ou simplesmente reverter as modificações do desenvolvedor A e manter o arquivo local.

To merge her local changes with the reshuffle, Developer B must first find out to what filename the conflicted file `Foo.c` was renamed/moved in the repository. This can be done by using the log dialog. Then use the button which shows the correct source file to resolve the conflict.

If Developer B decides that A's changes were wrong then she must choose the Mark as resolved button in the conflict editor dialog. This marks the conflicted file/folder as resolved, but Developer A's changes need to be removed by hand. Again the log dialog helps to track down what was moved.

4.6.3.3. Exclusão local, exclusão na atualização

1. Desenvolvedor A move `Foo.c` para `Bar.c` e submete para o repositório.

2. Desenvolvedor B move o arquivo `Foo.c` para `Bix.c`.

Uma atualização da cópia de trabalho do desenvolvedor B resulta num conflito de estrutura:

- `Bix.c` é marcado como adicionado com histórico.
- `Bar.c` é adicionado na cópia local com status "normal".
- `Foo.c` é marcado como eliminado e possui um conflito de estrutura.

Para resolver este conflito, Desenvolvedor B terá que descobrir para qual nome o arquivo em conflito `Foo.c` foi renomeado/movido no repositório. Isto pode ser feito usando a caixa de diálogo do log.

Em seguida, o Desenvolvedor B terá que decidir se o novo nome do arquivo `Foo.c` será mantido - se aquele dado pelo Desenvolvedor A ou se aquele renomeado por ele mesmo.

Depois que o Desenvolvedor B tenha manualmente solucionado o conflito, o conflito de estrutura deve ser marcado como "resolvido", usando o botão apropriado na caixa de diálogo do Editor de Conflitos.

4.6.3.4. Inexistência local, alteração na atualização

1. Desenvolvedor A trabalhando no trunk, modifica o arquivo `Foo.c` e submete o arquivo para o repositório

2. Desenvolvedor B trabalhando com o branch, move e renomeia o arquivo `Foo.c` para `Bar.c` e submete a alteração para o repositório

A atualização do Desenvolvedor A no trunk e a do Desenvolvedor B no branch, resulta num conflito de estrutura:

- `Bar.c` já está na cópia de trabalho com status "norma"'.
- `Foo.c` está assinalado como "desaparecido" e com um conflito de estrutura.

Para solucionar este conflito, o Desenvolvedor B terá que marcar esse arquivo como "resolvido" na caixa de diálogo do Editor de Conflitos - o qual irá removê-lo da lista de conflitos. O Desenvolvedor B terá então que decidir se deve copiar o arquivo `Foo.c` "desaparecido" do repositório para a cópia de trabalho ou então se deve atualizar as modificações do Desenvolvedor A no arquivo `Foo.c` para o arquivo renomeado como `Bar.c` ou então se deve ignorar as modificações marcando o conflito como resolvido e fazer nada mais.

Note que se você copiar o arquivo "desaparecido" do repositório e então marcá-lo como resolvido, esse arquivo copiado será removido novamente. Você deve primeiro, resolver o conflito.

4.6.3.5. Alteração local, exclusão na unificação

1. Desenvolvedor A trabalhando em trunk move `Foo.c` para `Bar.c` e o submete para o repositório.
2. Desenvolvedor B trabalhando no branch modifica o arquivo `Foo.c` e o submete para o repositório.

1. Desenvolvedor A trabalhando no trunk move a pasta `FooFolder` para `BarFolder` e submete a alteração para o repositório.
2. Desenvolvedor B trabalhando no branch modifica o arquivo `Foo.c` em sua cópia de trabalho.

A atualização do Desenvolvedor A no trunk e a do Desenvolvedor B no branch, resulta num conflito de estrutura:

- `Bar.c` é marcado como adicionado.
- `Foo.c` é marcado como modificado com um conflito de estrutura.

Desenvolvedor B agora terá que decidir o que fazer com a reorganização do desenvolvedor A e submeter suas modificações para o arquivo correspondente na nova estrutura ou simplesmente reverter as modificações do desenvolvedor A e manter o arquivo local.

To merge her local changes with the reshuffle, Developer B must first find out to what filename the conflicted file `Foo.c` was renamed/moved in the repository. This can be done by using the log dialog for the merge source. The conflict editor only shows the log for the working copy as it does not know which path was used in the merge, so you will have to find that yourself. The changes must then be merged by hand as there is currently no way to automate or even simplify this process. Once the changes have been ported across, the conflicted path is redundant and can be deleted.

If Developer B decides that A's changes were wrong then she must choose the Mark as resolved button in the conflict editor dialog. This marks the conflicted file/folder as resolved, but Developer A's changes need to be removed by hand. Again the log dialog for the merge source helps to track down what was moved.

4.6.3.6. Exclusão local, exclusão na unificação

1. Desenvolvedor A trabalhando em trunk move `Foo.c` para `Bar.c` e o submete para o repositório.
2. Desenvolvedor B trabalhando em um ramo, move e renomeia o arquivo `Foo.c` para `Bix.c` e submete a alteração para o repositório.

A atualização do Desenvolvedor A no trunk e a do Desenvolvedor B no branch, resulta num conflito de estrutura:

- `Bix.c` é marcado com Status de normal (não modificado).
- `Bar.c` é marcado como adicionado com histórico.

- `Foo.c` é marcado como inexistente e como tendo um conflito de estrutura.

To resolve this conflict, Developer B has to find out to what filename the conflicted file `Foo.c` was renamed/moved in the repository. This can be done by using the log dialog for the merge source.

Em seguida, o Desenvolvedor B terá que decidir se o novo nome do arquivo `Foo.c` será mantido - se aquele dado pelo Desenvolvedor A ou se aquele renomeado por ele mesmo.

Depois que o Desenvolvedor B tenha manualmente solucionado o conflito, o conflito de estrutura deve ser marcado como "resolvido", usando o botão apropriado na caixa de diálogo do Editor de Conflitos.

4.6.3.7. Outros conflitos de estrutura

Há outros casos em que são identificados como conflitos de estrutura simplesmente porque o conflito envolve um diretório ao invés de um arquivo. Por exemplo, se você adicionar um diretório com o mesmo nome no tronco e no ramo e então tentar combiná-los você terá um conflito de estrutura.

4.7. Obtendo Informações de Estado

Enquanto você estiver trabalhando na sua cópia de trabalho você frequentemente precisa saber quais arquivos foram alterados/adicionados/removidos ou renomeados, ou ainda quais arquivos foram alterados por outros.

4.7.1. Sobreposição dos Ícones

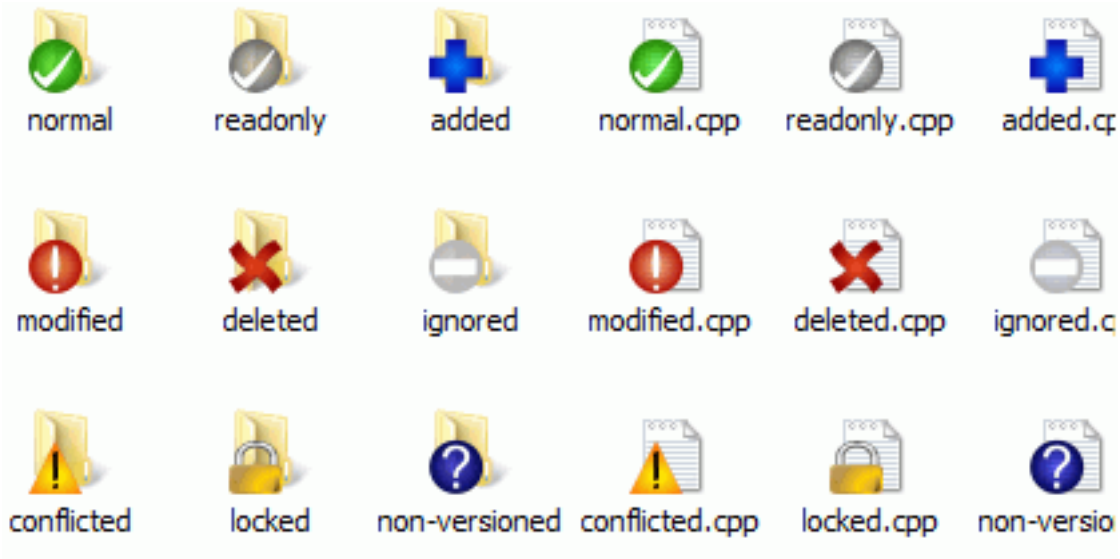


Figura 4.11. Explorer mostra os ícones sobrepostos

Agora que você obteve uma cópia de trabalho do repositório Subversion você pode ver seus arquivos no windows explorer com ícones modificados. Esta é uma das razões para que o TortoiseSVN seja tão popular. O TortoiseSVN adiciona um ícone de revestimento sobre o ícone de cada arquivo. Dependendo do status do arquivo no Subversion o revestimento é diferente.

Uma cópia de trabalho recém obtida possui uma camada contendo uma marca verde. Isto significa que o estado é *normal*.

Assim que você começar a editar um arquivo, o estado muda para *modificado* e a camada de revestimento então muda para um ponto de exclamação vermelho. Dessa maneira você pode facilmente ver quais arquivos foram alterados desde que você atualizou sua cópia de trabalho e que precisam ser submetidos.

Se durante uma atualização ocorrer um *conflito* então o ícone muda para um ponto de exclamação amarelo.

Se você marcou a propriedade `svn:needs-lock` em um arquivo, o Subversion faz com que aquele arquivo se torne somente-leitura até que você obtenha uma trava naquele arquivo. Tais arquivos tem este revestimento para indicar que você precisa obter uma trava antes que você possa editar aquele arquivo.

Se você mantém uma trava em um arquivo, e o estado do Subversion é *normal*, o revestimento do ícone lembra você de que você deve liberar a trava se não estiver usando-o para permitir que outros possam submeter suas mudanças para quele arquivo.

Este ícone mostra a você que alguns arquivos ou pastas dentro da pasta atual foram agendados para serem *deletados* do controle de versionamento ou um arquivo dentro do controle de versionamento está faltando em uma pasta.

O sinal de mais significa que um arquivo ou pasta foi agendado para ser *addicionado* ao controle de versões.

O sinal de barra lhe diz que um arquivo ou pasta é *ignorado* do controle de versões. Esse revestimento é opcional.

Este ícone mostra arquivos e pastas que não estão sob o controle de versionamento, mas não foram ignorados. Este revestimento é opcional.

De fato, você pode descobrir que nem todos estes ícones são usados em seu sistema. Isto é porque o número de camadas permitidas pelo Windows é bem limitado e se você também estiver usando uma versão antiga do TortoiseCVS, então não haverá espaço suficiente. O TortoiseSVN tenta ser um “Good Citizen (TM) - Bom Cidadão” e limita seu uso de camadas para dar outros aplicativos uma chance também.

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 :-)

Para uma descrição de como as camadas de ícones correspondem ao estado do Subversion e outros detalhes técnicos, leia Seção F.1, “Sobreposição dos Ícones”.

4.7.2. Situação Detalhada

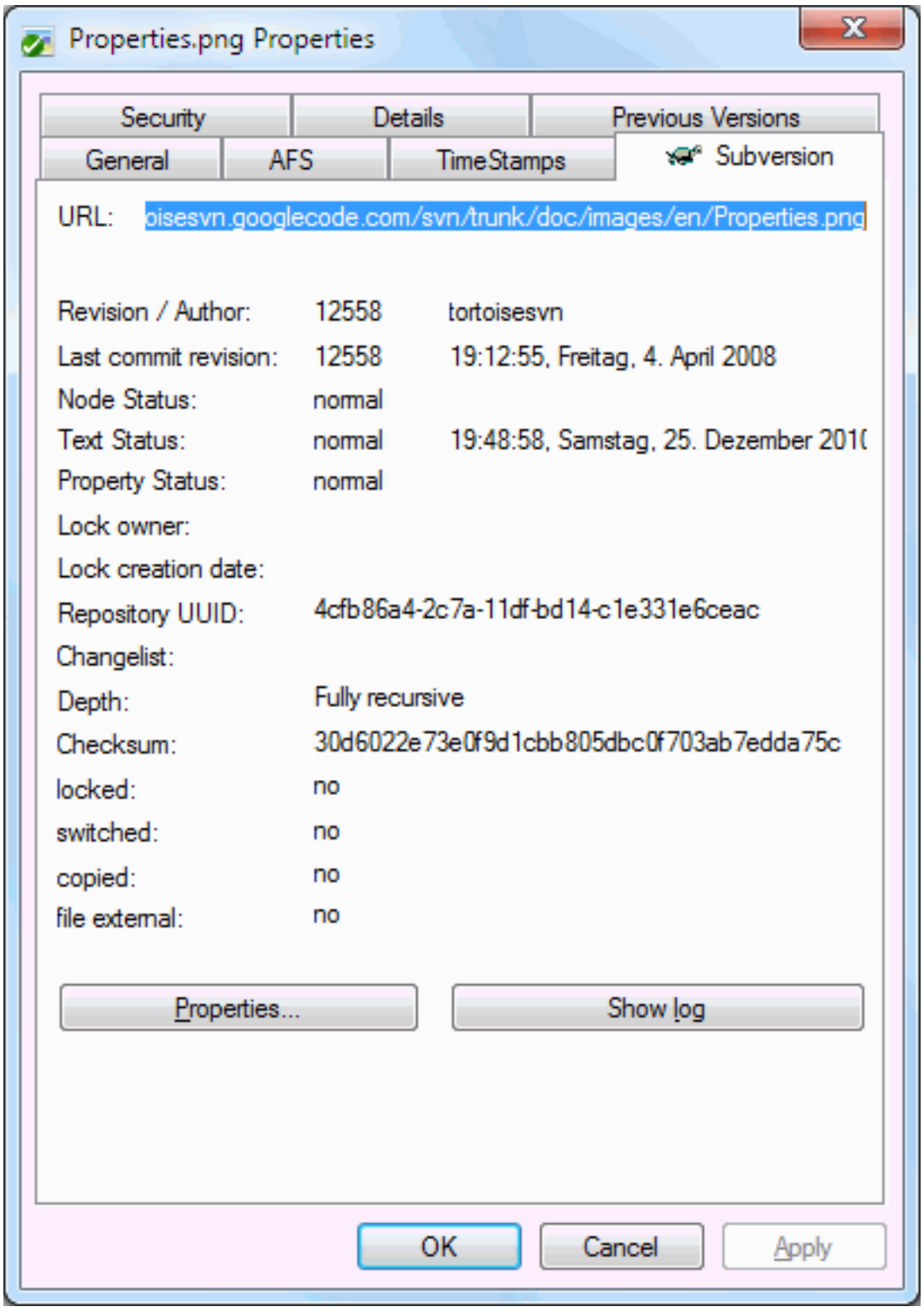


Figura 4.12. Página de propriedade do explorador, aba do Subversion

Algumas vezes você quer ter informações mais detalhadas sobre um arquivo/diretório do que a sobreposição de ícones. Você pode obter toda a informação que Subversion provê na janela de propriedades no explorer. Apenas selecione o arquivo ou diretório e selecione Menu do Windows → propriedades no menu de contexto (nota: esta é a entrada de menu de propriedades do explorer, não o submenu do TortoiseSVN!). No caixa de propriedades da janela, o TortoiseSVN adicionou uma nova página de propriedades para arquivos/pastas sobre o controle do Subersion, onde você pode ver todas as informações relevantes sobre os arquivos/diretórios selecionados.

4.7.3. Estado Local e Remoto

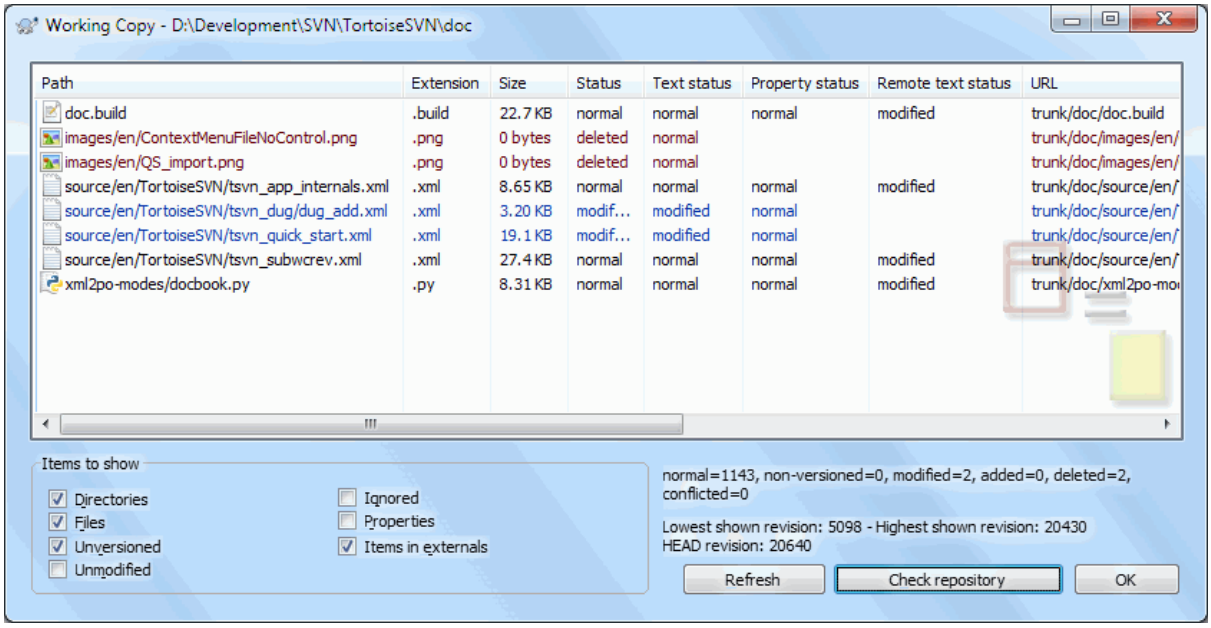


Figura 4.13. Procurar por Modificações

É frequentemente muito útil saber quais arquivos você alterou e também quais arquivos foram alterados e submetidos por outros. É aí que o comando TortoiseSVN → Verificar por Modificações... se torna útil. Esta janela lhe mostrará cada arquivo que foi alterado de qualquer maneira dentro da sua cópia de tabalho, assim como quaisquer arquivos não-versionados que você possa possuir.

Se você clicar no botão Verificar Repositório então você também pode olhar as mudanças no repositório. Dessa maneira você pode checar antes de uma atualização se pode haver um possível conflito. Você também pode atualizar arquivos selecionados do repositório sem atualizar toda a pasta. Por padrão, o botão Verificar Repositório somente busca o estado remoto com a profundidade de obtenção da sua cópia de trabalho. Se você deseja ver todos os arquivos e pastas no repositório, mesmo aquelas que você não obteve, então você deve manter pressionada a tecla **Shift** quando clicar no botão Verificar Repositório.

A janela usa codificação de cores para demarcar o estado.

Azul
: Itens modificados localmente.

Roxo
: Itens adicionados. Itens que foram adicionados com histórico tem um sinal + na coluna de Estado de Texto, e uma janela mostra de onde o item foi copiado.

Vermelho escuro
: Itens excluídos ou perdidos.

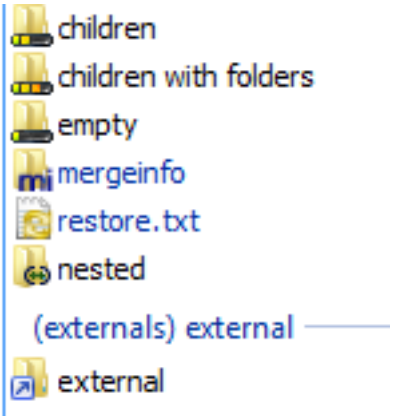
Verde
: Itens modificados localmente e no repositório. As mudanças serão juntadas ao atualizar. Isto *pode* produzir conflitos ao atualizar.

Vermelho vivo
: Itens modificados localmente e apagados no repositório, ou modificados no repositório e apagados localmente. Isto *irá* produzir conflitos durante a atualização.

Preto
: Itens não modificados ou não controlados.

Este é o esquema padrão de cores, mas você pode personalizar as cores usando a janela de configurações. Acesse Seção 4.31.1.5, “Configurações de Cores do TortoiseSVN” para mais informações.

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



Overlays are shown for the following states:

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

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

Do menu de contexto da janela você pode ver um diff de mudanças. Marque as mudanças localis que *você* fez usando Menu de Contexto → Comparar com Base. Marque as mudanças no repositório feitas por outros usando Menu de Contexto → Mostrar Diferenças como Diff Unificado.

Você também pode reverter as mudanças em arquivos individuais. Se você apagou um arquivo acidentalmente, ele aparecerá como *Faltando* e você pode usar *Reverter* para recuperá-lo.

Arquivos não-versionados e ignorados podem ser enviados para a lixeira daqui usando Menu de Contexto → Apagar. Se você deseja apagar arquivos permanentemente (ignorando a lixeira) matenha pressionada a tecla **Shift** enquanto cliclar em Apagar.

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

As colunas são personalizáveis. Se você clicar com o botão direito em qualquer cabeçalho de coluna você verá um menu de contexto permitindo-o selecionar quais colunas serão exibidas. Você pode também alterar a largura das colunas usando o ícone de arrastar que aparece quando você passa o mouse em cima de uma borda de coluna. Essas personalizações são preservadas, então você verá os mesmos cabeçalhos da próxima vez.

Se você está trabalhando em diversas tarefas independentes ao mesmo tempo, você pode também agrupar arquivos em changelists. Leia Seção 4.4.2, “Lista de Alterações” para maiores informações.

Na parte inferior da janela você pode ver um sumário da variação das revisões do repositório em uso na sua cópia de trabalho. Essas são as revisões *submetidas*, não as revisões *atualizadas*; elas representam a variação das revisões onde estes arquivos foram submetidos pela última vez, não as revisões nas quais foram atualizados. Note que a variação de revisões mostrada aplica-se somente aos itens exibidos, não à cópia de trabalho inteira. Se você deseja ver informações sobre a cópia de trabalho inteira você deve marcar a caixa Mostrar arquivos não modificados.

Dica

Se você deseja uma exibição simples da sua cópia de tabalho, p. ex. mostrando todos os arquivos e pastas em cada nível da hierarquia de pastas, então a janela Verificar por Modificações é a maneira mais fácil de conseguir isso. Simplesmente marque a caixa Mostrar arquivos não modificados para mostrar todos os arquivos na sua cópia de trabalho.

Reparando Referência Externas Renomeadas

Algumas vezes arquivos são renomeados fora do Subversion, e eles aparecem na lista de arquivos como arquivos perdidos e como arquivos não versionados. Para evitar a perca do histório você precisa notificar o Subversion sobre a conexão entre os arquivos. Simplesmente selecione o arquivo antigo (perdido) e o arquivo nome (não versionado) e use a opção Menu de Contexto → Reparar movimento para identificar os dois arquivos como uma renomeação.

Reparando Cópias de Referências Externas

Se você fez uma cópia de um arquivo e esqueceu de usar o comando do SVN na cópia, você pode reparar a cópia e assim o novo arquivo não perderá seu histórico. Simplesmente selecione o arquivo antigo (normal ou modificado) e o nome arquivo (não versionado) e use Menu de Contexto → Reparar Cópia para identificar os dois arquivos como um cópia.

4.7.4. Visualizar diferenças

Frequentemente você deseja olhar dentro de seus arquivos, para observar o que foi alterado. Você pode fazê-lo selecionando o arquivo que foi alterado, e selecionando Diff do menu de contexto do TortoiseSVN. Isto inicia o visualizador-diff externo, que irá então comparar o arquivo atual com a cópia pristina (revisão BASE), que foi armazenada após a última submissão ou atualização.

Dica

Mesmo quando não não se está dentro de uma cópia de trabalho ou quando você tiver múltiplas versões do arquivo em disco, você ainda pode visualizar diffs:

Selecione os dois arquivos que você deseja comparar no explorer (p. ex. usando **Ctrl** e o mouse) e escolha Diff do menu de contexto do TortoiseSVN. O arquivo clicado por último (o que possui foco, ou seja, um retângulo pontilhado) será compreendido como o último.

4.8. Lista de Alterações

Em um mundo ideal, você sempre estará tabalhando em uma coisa de cada vez, e sua cópia de trabalho contém apenas um conjunto de alterações lógicas. OK, devolta a realidade. Frequentemente acontece que você precisa trabalhar em diversas tarefas não relacionadas ao mesmo tempo, e quando você olhar na janela de submissão, todas as mudanças estão misturadas. A funcionalidade das *changelists* ajuda-o a agrupar os arquivos, tornando mais fácil de ver o que você está fazendo. É claro que isto só pode acontecer se as mudanças não se sobrepuserem. Se duas tarefas diferentes afetarem o mesmo arquivo, não há outra maneira de separar as mudanças.

Você pode ver listas de alterações em diversos lugares, mas os mais importantes são na janela de submissão e na janela de verificação-de-modificações. Vamos começar na janela de verificação-de-modificações após você

tiver trabalhado em diversas funcionalidades e muitos arquivos. Quando você abrir esta janela pela primeira vez, todos os arquivos alterados estarão listados juntos. Suponha que agora você queira organizar as coisas e agrupar os arquivos de acordo com a funcionalidade.

Selecione um ou mais arquivos e use Menu de Contexto → Mover para a changelist para adicionar um item à lista de alterações. Inicialmente não haverão listas de alterações, então a primeira vez que você fizer isto você criará uma nova changelist. Dê o nome que melhor descreve para que à está usando, e clique OK. A janela irá agora mudar para mostrar os grupos de itens.

Uma vez que você tiver criado a lista de alterações você pode arrastar e soltar itens nela, tanto de outra lista de alterações, como do Windows Explorer. Arrastar do Explorer pode ser útil já que permite-o adicionar itens para a lista de modificações antes que o arquivo seja modificado. Você pode fazê-lo da janela de verificar-por-modificações, mas somente através da exibição de todos os arquivos não modificados.

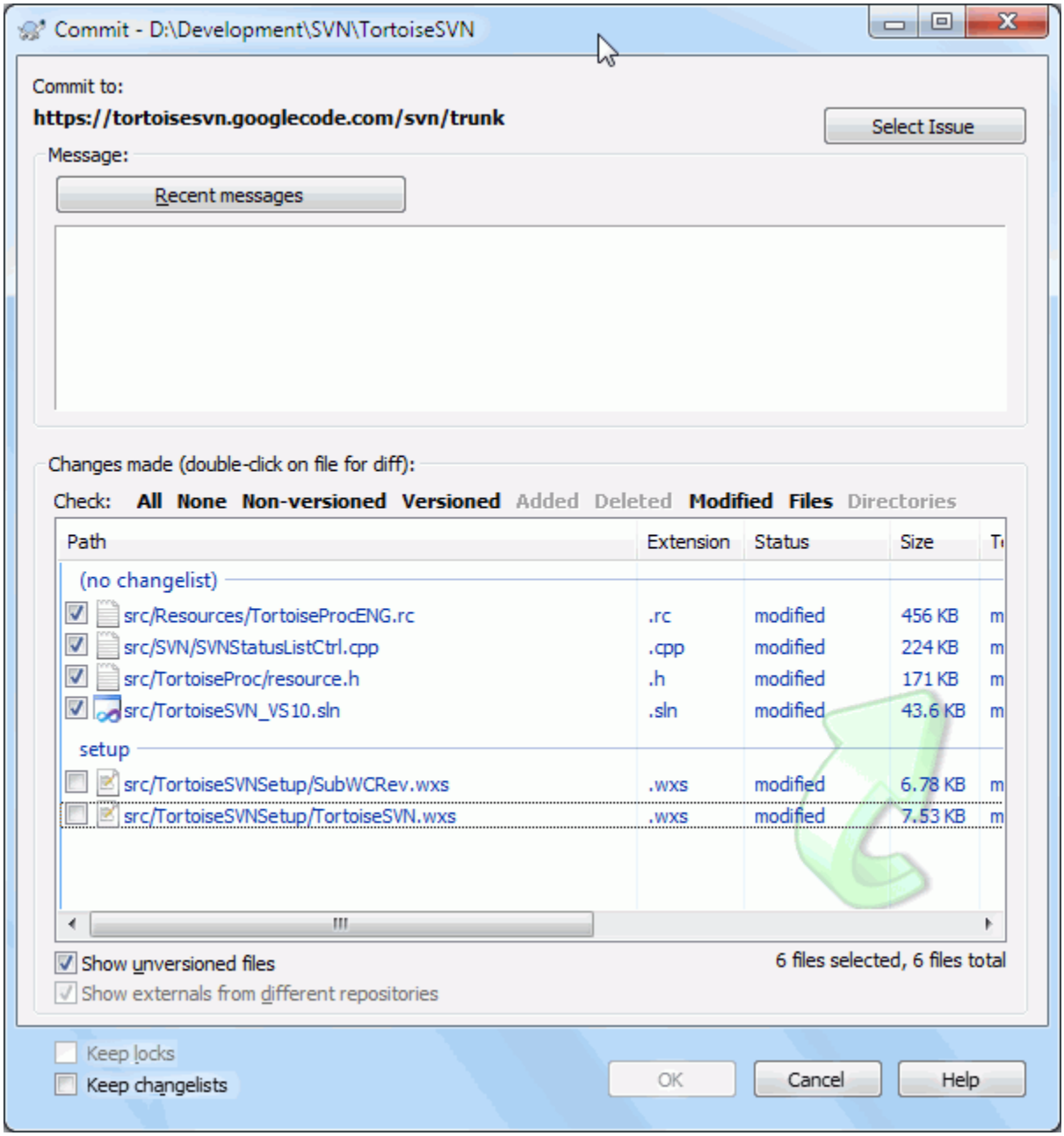


Figura 4.14. Janela de Submissão com Lista de Alterações

Na janela de submissão você pode ver os mesmos arquivos, agrupados pela lista de alterações. Além de visualizar de forma imediata a indicação visual dos agrupamentos, você também pode usar os grupos para selecionar quais arquivos submeter.

O TortoiseSVN reserva um nome de lista de alterações para uso pessoal, chamado de `ignore-on-commit`. Isto é usado para marcar arquivos versionados que você geralmente não deseja submeter mesmo que tenham mudanças locais. Esta funcionalidade está desrita em Seção 4.4.4, “Excluindo Itens de uma Lista de Submissões”.

Quando você submete arquivos pertencendo à uma lista de mudanças então normalmente você espera que a pertinência da lista de mudanças não seja mais necessária. Então por padrão, arquivos são removidos das listas automaticamente na submissão. Se você deseja reter o arquivo em sua lista, marque a caixa Guardar changelists na parte inferior da janela de submissão.

Dica

Listas de Alterações são puramente uma funcionalidade local do cliente. Criar e remover Listas de Alterações não afetará o repositório, nem a cópia de tabalho de mais ninguém. Elas são simplesmente uma maneira conveniente para vocÇê organizar seus arquivos.

Atenção

Note that if you use changelists, externals will no longer show up in their own groups anymore. Once there are changelists, files and folders are grouped by changelist, not by external anymore.

4.9. Shelving

More often than wanted, it's necessary to stop what you were working on and work on something else. For example a serious problem needs immediate dealing with and you have to stop working on the new feature. If possible, you should commit the changes you have done so far and then start working on the urgent issue, but often those changes would break the build or are just not ready for committing yet.

So if you can't commit your local changes yet, you have to put them aside while you're working on the urgent issue. The *shelving* feature helps you do exactly that: you can store your local changes on a shelve, get your working copy in a clean state again and work on the issue. After you're finished with the urgent issue and you've committed those changes, you can *unshelve* your shelved work and continue working on your previous task again.

Two new commands are implemented for this. One for shelving and one for unshelving.

To shelve your local changes, select your working copy and use Context Menu → Shelve The following dialog allows you to select the files you want to shelve and give a name under which you want to store them.

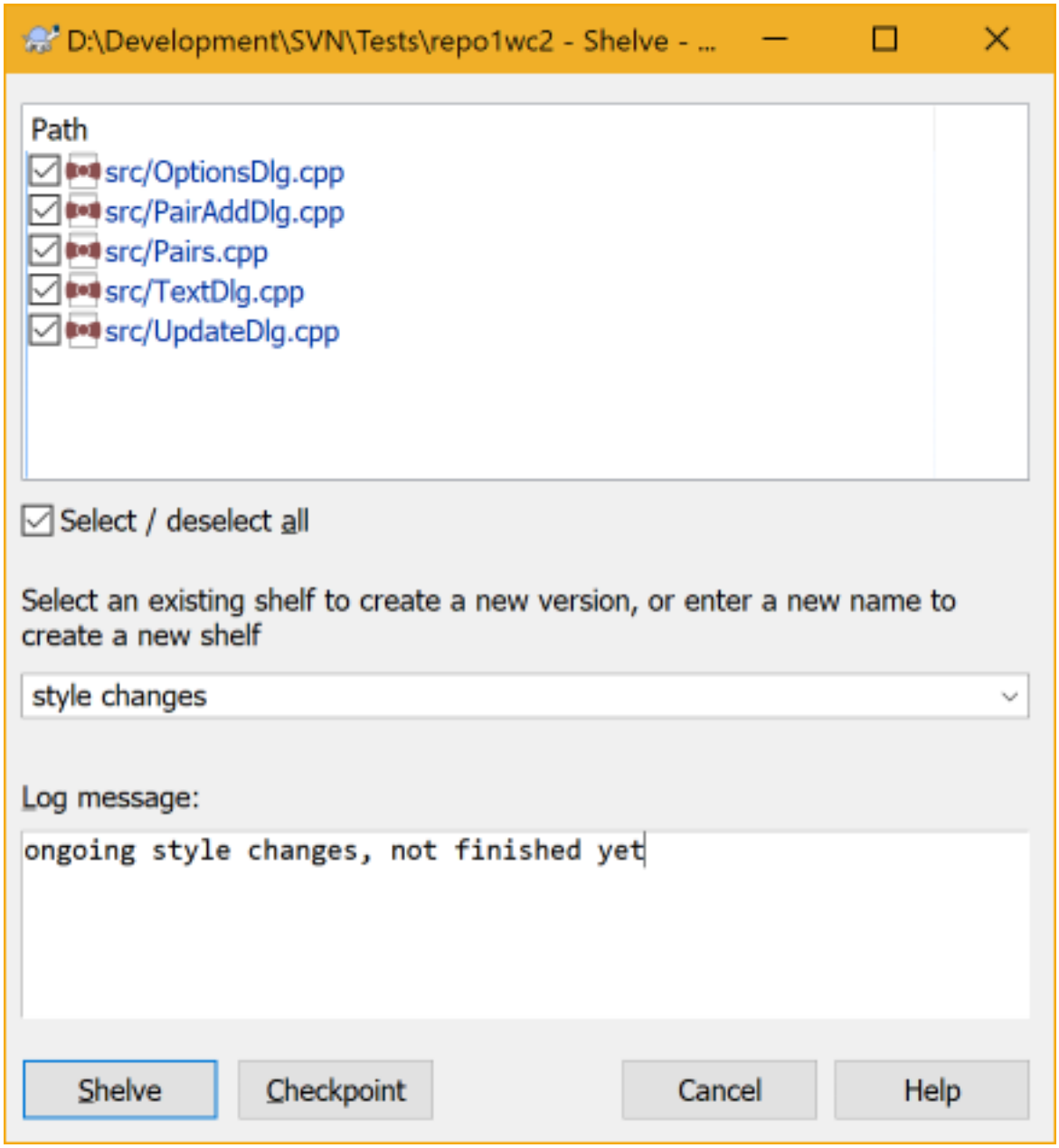


Figura 4.15. Shelve dialog

If you select an existing shelf, then a new version is created for that shelf. If you provide a new name, a new shelf is created for the selected files.

If you click the Shelve button, the shelf is created and your working copy files are reset to a clean state. If you click the Checkpoint button, the shelf is created but your local modifications are kept.

To unshelve your changes, use Context Menu → Unshelve to get the unshelve dialog. This dialog shows you a list of all shelved items. Select the shelved item you want and the version to apply back to your working copy and click Apply.

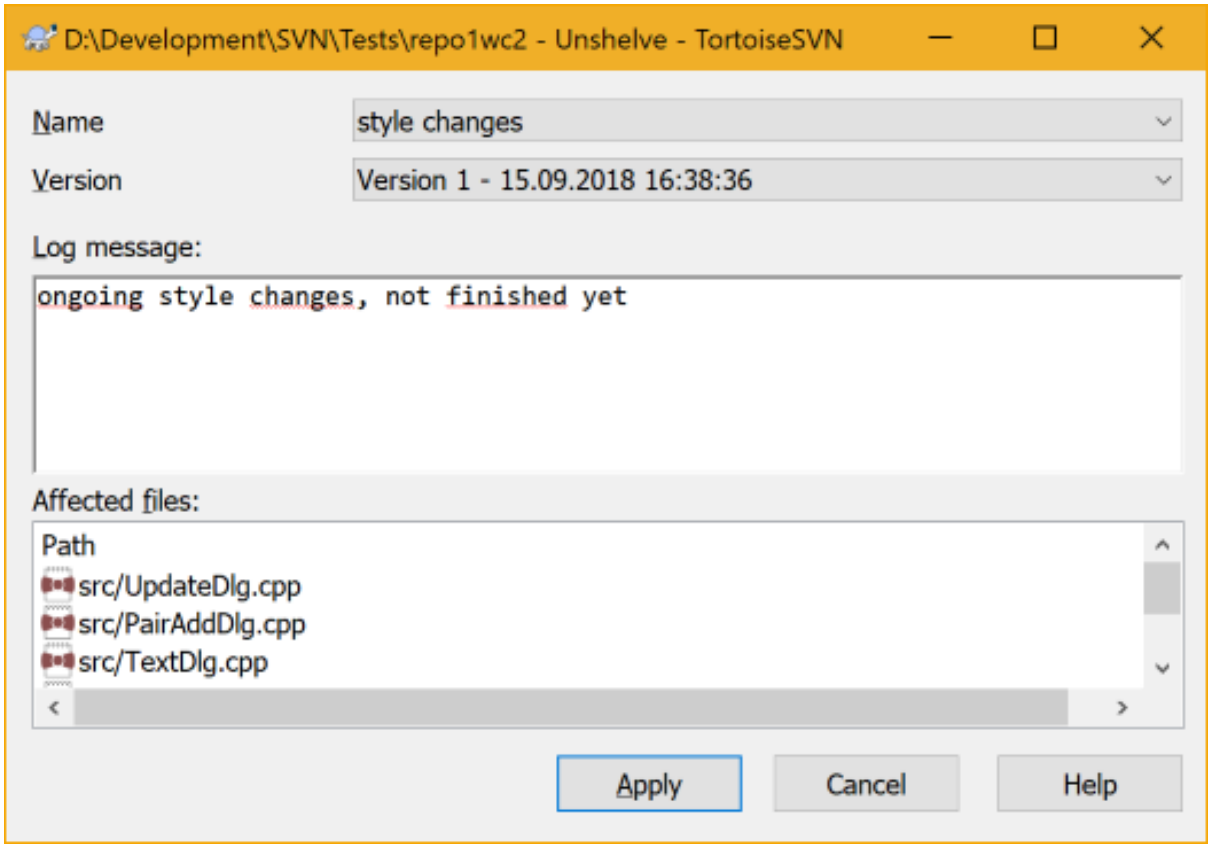


Figura 4.16. Unshelve dialog

Dica

Shelves are purely a local client feature. Creating and removing Shelves will not affect the repository, nor anyone else's working copy.

Experimental

The shelving feature is still marked as `experimental`.

That means that while shelving works as advertised, it is still in a stage where it's heavily improved and worked on. That also means that there's no guarantee that the shelves you create are upwards compatible and future versions might not be able to use them. And of course the UI might change as well in future versions to accommodate new features and behaviors.

4.10. Janela de Revisão de Registro

Para cada mudança feita e submetida, você deverá prover uma mensagem de auditoria para cada mudança. Desta maneira você pode mais tarde encontrar o que foi feito e porque, e você terá um log detalhado para cada processo de desenvolvimento.

A Janela de Auditoria da Revisão recupera todas as mensagens de log e as mostra para você. A visualização é dividida em 3 painéis.

- O painel superior mostra uma lista de revisões onde alterações para o arquivo/diretório foi submetido. Este resumo inclui a data e a hora, a pessoa que submeteu a revisão e os primeiros caracteres da mensagem de auditoria.

Linhas mostradas em azul indicam que alguma coisa foi copiada para esta linha de desenvolvimento (talvez de um ramo).

- O painel do meio mostra a mensagem de auditoria completa da revisão selecionada.
- O painel inferior mostra a lista de todos os arquivos e diretórios que foram modificados para a revisão selecionada.

Mas isto é muito mais que isso - a janela provê um menu de contexto com comandos que podem ser usados para obter ainda mais informações sobre o histórico do projeto.

4.10.1. Invocando a Janela de Histórico de Revisão

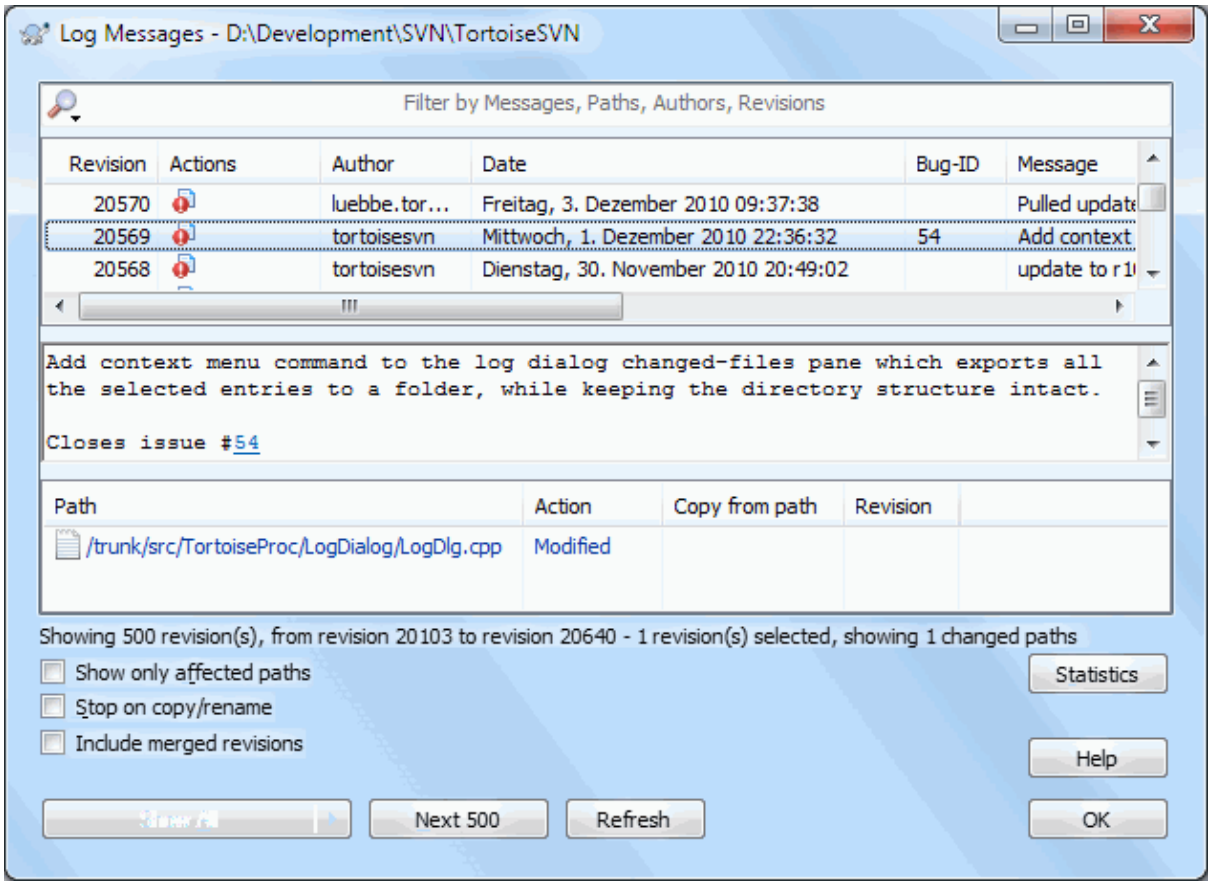


Figura 4.17. A Janela de Histórico de Revisão

There are several places from where you can show the Log dialog:

- Do TortoiseSVN contexto do submenu
- Da página de propriedades
- Da janela de Progresso depois que uma atualização foi encerrada. Então a janela de Auditoria somente mostra tas revisões que foram alteradas desde a sua última atualização.
- A partir do navegador do repositório

Se o repositório está indisponível você verá a janela Deseja continuar desconectado?, descrito em Seção 4.10.10, “Modo desconectado”.

4.10.2. Histórico de Ações de Revisão

o painel superior tem uma coluna Ações contendo ícones que resumem o que foi feito na revisão. Existem quatro diferentes ícones, cada um apresentado na sua própria coluna.

Se uma revisão modificou algum arquivo ou diretório, o ícone *modificado* é apresentado na primeira coluna.

Se uma revisão adicionou algum arquivo ou diretório, o ícone *adicionado* é apresentado na segunda coluna.

Se uma revisão apagou algum arquivo ou diretório, o ícone *apagado* é apresentado na terceira coluna.

Se uma revisão sobrepôs algum arquivo ou diretório, o ícone *substituído* é apresentado na quarta coluna.

If a revision moved or renamed a file or directory, the *moved* icon is shown in the fourth column.

If a revision replaced a file or directory by moving/renaming it, the *move replaced* icon is shown in the fourth column.

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

If a revision reverse merged a file or directory, the *reverse merged* icon is shown in the fourth column.

4.10.3. Recuperando Informações Adicionais

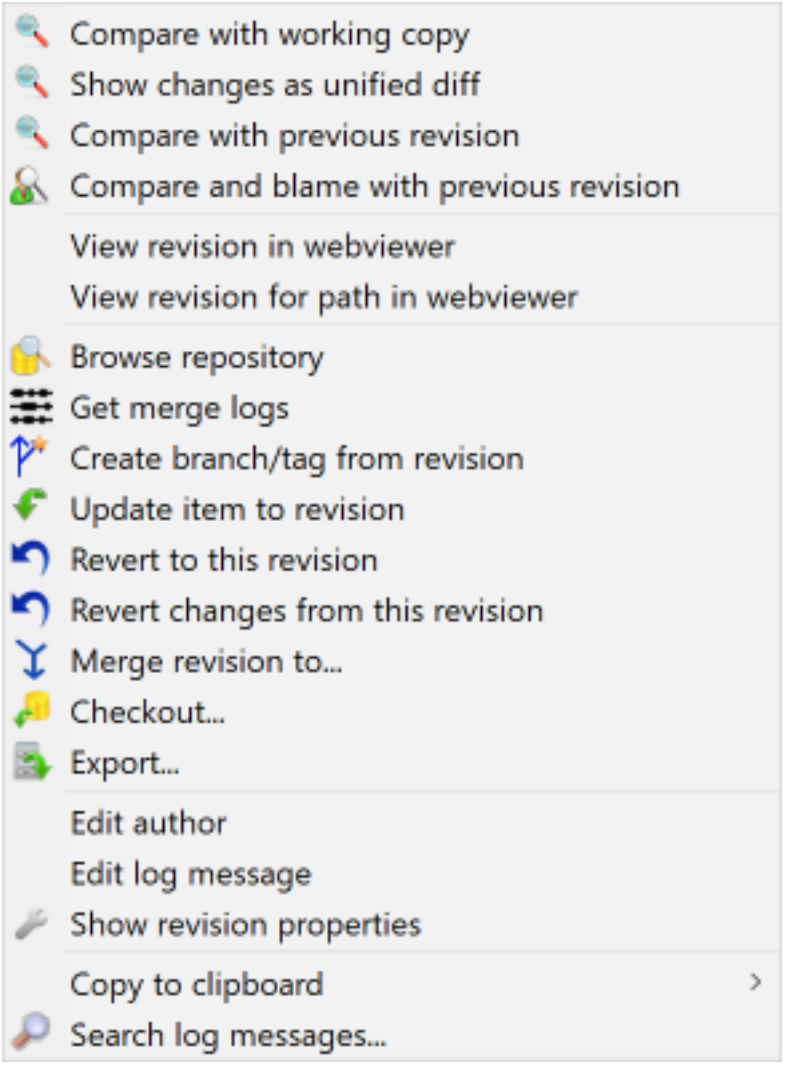


Figura 4.18. O Menu de Contexto do Painel Superior da Janela de Auditoria da Revisão

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.

Comparar com a cópia de trabalho
: Compara a revisão selecionada com a cópia de trabalho. A ferramenta padrão Diff é o TortoiseMerge o qual é um suplemento do TortoiseSVN. Se a janela de auditoria é para um diretório, ela mostrará a você a lista dos arquivos alterados, e permitirá a você rever as alterações feitas em cada arquivo individualmente.

Comparar e ver autoria em relação à BASE de trabalho
: Veja a autori da revisão selecionada, e o arquivo BASE na sua cópia de trabalho e compare o relatório de autoria usando a ferramenta visual diff. Leia Seção 4.24.2, “Diferenças de Autoria” para mais detalhes. (somente arquivos)

Mostrar alterações com as diferenças unificadas
: Visualizar as alterações feitas em uma revisão selecionada como um arquivo Unificado-Diff (formato de correção GNU). Isto mostrará somente as diferenças com algumas linhas do contexto. É algo mais complicado de ler que o arquivo de comparação visual, mas mostrará todas as alterações juntas em um formato compactado.

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

Comparar com a revisão anterior
: Compara a revisão selecionada com a revisão anterior. Isto funciona de maneira similar à comparação com sua cópia de trabalho. Para diretórios esta opção mostrará primeiro os arquivos alterados permitindo a você selecionar os arquivos para comparar.

Comparar e ver autoria em relação à revisão anterior
: Mostra uma janela com os arquivos alterados permitindo a você selecionar os arquivos. Mostra a autoria da revisão selecionada, e a revisão anterior, e compara os resultados usando uma ferramenta visual diff. (somente diretórios)

Save revisão para...
: Salva a revisão selecionada para um arquivo e então você terá uma versão antiga do arquivo. (somente arquivos)

Abrir / Abrir com...
: Abre o arquivo selecionado, tanto com o visualizador padrão para aquele tipo de arquivo, como com o programa que você escolheu. (somente arquivos)

Autoria...
: Autorias do arquivo até a revisão selecionada (apenas arquivos).

Navegar no repositório
: Abre o navegador de repositório para examinar o arquivo selecionado ou diretório dentro do repositório como ele estava na revisão selecionada.

Criar ramificação/tórulo de uma revisão
: Criar um ramo ou um rótulo de uma revisão selecionada. Isto é muito útil, por exemplo, se você esqueceu de criar um rótulo e já submeteu outras alterações que supostamente não deveriam fazer parte da liberação.

Atualizar item para a revisão
: Atualize sua cópia de trabalho para a revisão selecionada. Prático se você quer que sua cópia de trabalho reflita algum momento no passado, ou se você tem submissões adicionais no repositório e você quer atualizar sua cópia um passo por vez. É melhor atualizar o diretório inteiro da sua cópia de trabalho, não apenas um arquivo, senão sua cópia de trabalho poderá ficar inconsistente.

  Se você quer desfazer uma alteração mais recente permanentemente, use alternativamente a opção Reverter para esta revisão.

Reverter para esta revisão
: Reverter para uma revisão mais recente. Se você fez várias alterações, e então decidiu que você quer voltar atrás para como as coisas eram na revisão N, este é comando que você precisa. As alterações serão desfeitas em sua cópia de trabalho, assim esta operação *não* afetará seu repositório até que você submeta as alterações. Note que isto irá desfazer *todas* as alterações feitas depois da revisão selecionada, substituindo o arquivo/diretório com a revisão mais recente.

  Se sua cópia de trabalho está em um estado de não modificada, depois que você realizou a operação sua cópia de trabalho aparecerá como modificada. Se você já tem alterações locais, este comando combinará o *desfazimento* das alterações na sua cópia de trabalho.

  O que está acontecendo internamente é que o Subversion realiza uma combinação reversa das alterações feitas depois da revisão selecionada, desfazendo o que foi feito dessas submissões anteriores.

  Se depois de executar esta ação você decidir que você quer *desfazer o desfazimento* e voltar sua cópia de trabalho de volta para o estado anterior de não modificado, você deve usar TortoiseSVN → Reverter de dentro do Windows Explorr, e isto descartará as modificações locais feitas pela ação da combinação reversa.

  Se você simplesmente quer ver como um arquivo ou diretório era em uma revisão anterior, use Atualizar para a revisão ou então Salvar revisão como....

Reverter alterações de uma revisão
: Desfazer alterações das quais foram feitas em uma revisão selecionada. As alterações são desfeitas em sua cópia de trabalho, assim, esta operação *não* afeta o repositório de jeito algum! Note que isto irá desfazer somes as alterações feitas na revisão; a ação não substituirá sua cópia de trabalho com o arquivo inteiro da revisão anterior. Ela é muito útil por desfazer uma alteração anterior quando outras alterações não relacionadas já foram feitas.

  Se sua cópia de trabalho está em um estado de não modificada, depois que você realizou a operação sua cópia de trabalho aparecerá como modificada. Se você já tem alterações locais, este comando combinará o *desfazimento* das alterações na sua cópia de trabalho.

  O que está acontecendo internamente é que o Subversion executa uma combinação reversa de uma revisão, desfazendo seu efeito desde a submissão anterior.

  Você pode *desfazer o desfazimento* como descrito acima em Reverter para esta revisão

Unificar revisão para...
: Combinas as revisões selecionadas em uma cópia de trabalho diferente. Uma janela de seleção de pastas permite que você escolha as cópias de trabalho para combinar, mas após isto, não há uma janela de confirmação, nem qualquer oportunidade de tentar uma combinação de teste. É uma boa idéia combinar em uma cópia de trabalho não modificada, assim você pode reverter as mudanças se não funcionar! Isto é um recurso útil se você quer combinar revisões selecionadas de um ramo para outro.

Obter...
: Obtenha uma cópia atualização de um diretório selecionado em uma revisão selecionada. Isto abrirá uma janela para você confirmar a URL e a revisão, e selecionar o local para gravar os arquivos.

Exportar...
: Exporta o arquivo/diretório selecionado na revisão selecionada. Isto abrirá uma janela para você confirmar a URL e a revisão, e selecionar o local para exportar.

Editar autor / mensagem de log
: Editar a mensagem de auditoria ou o autor associado à submissão anterior. Leia Seção 4.10.7, “Alterando a Mensagem de Auditoria e o Autor” para saber como isto funciona.

Mostrar propriedades da revisão
: Visualizar e editar qualquer propriedade, não apenas a mensagem de auditoria e o autor. Veja em Seção 4.10.7, “Alterando a Mensagem de Auditoria e o Autor”.

Copiar para área de transferência
: Copiar os detalhes do log das revisões selecionadas para a área de transferência. Isto copiará o número da revisão, o autor, a data, a mensagem de auditoria e a lista dos itens alterados para cada revisão.

Procurar mensagens de log...
: Procura mensagens de auditoria pelo texto que você digitou. Isto procura nas mensagens de log o que você digitou e também a ação resumida criada pelo Subversion (mostrado no painel inferior). A procura não diferencia maiúscula/minúscula.

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

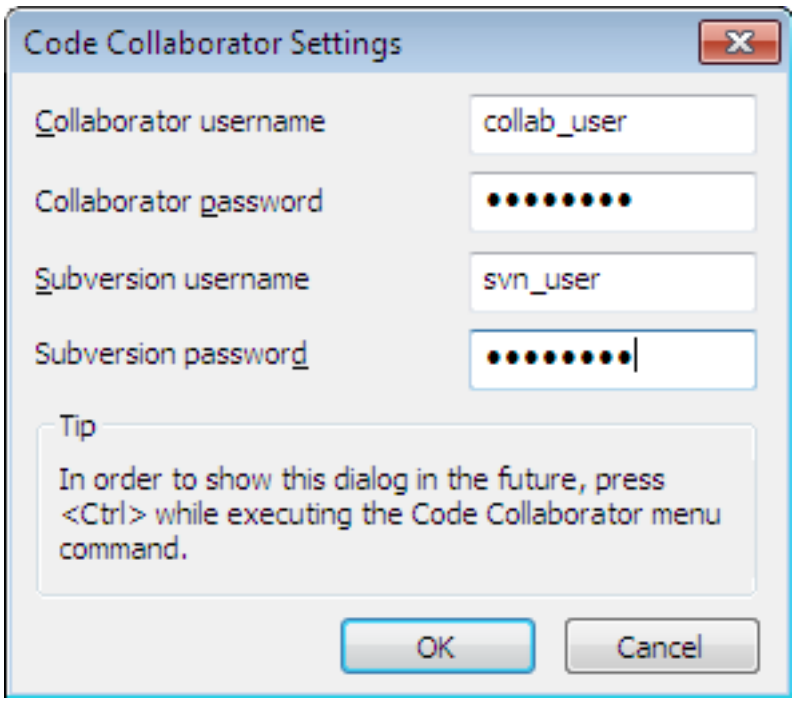


Figura 4.19. A Janela de Configurações do Colaborador de Código

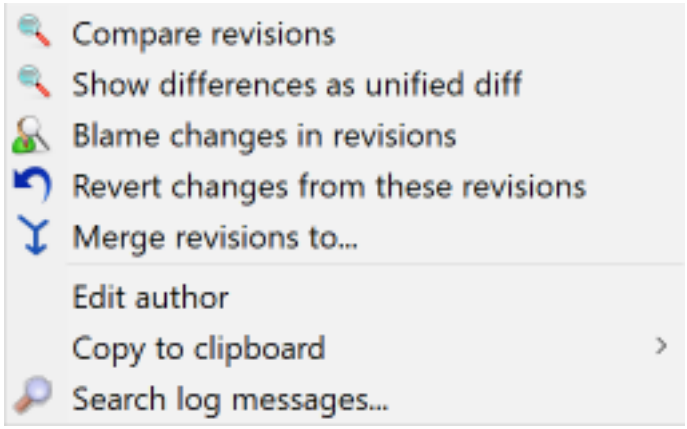


Figura 4.20. Menu de Contexto do Painel Superior para 2 Revisões Selecionadas

Se você selecionar duas revisões de uma vez (usando o comum **Ctrl**-modificador), o menu de contexto mudará e mostrará algumas opções adicionais:

Se você selecionar duas ou mais revisões (usando os modificadores **Ctrl** ou **Shift**), o menu de contexto irá incluir uma entrada para Reverter todas as mudanças que foram reitas pelas revisões selecionadas. Esta é a maneira mais fácil de restaurar um grupo de revisões de uma vez.

Você pode também escolher combinar as revisões selecionadas para outra cópia de trabalho, como descrito acima.

Se todas as revisões selecionadas são do mesmo autor, você pode editar o autor de todas as revisões de uma vez só.

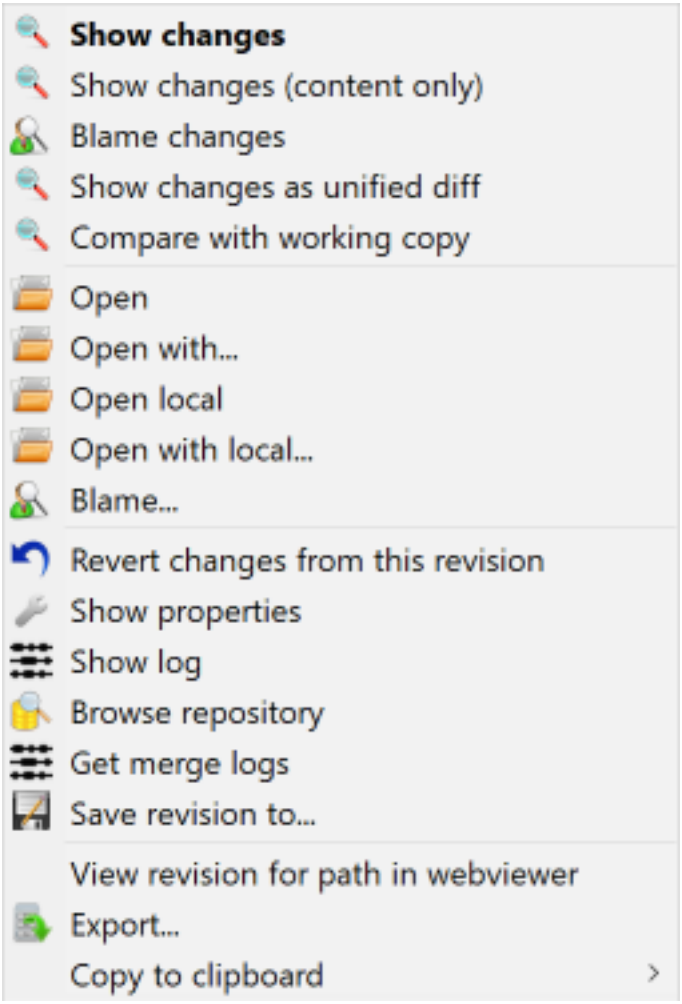


Figura 4.21. O Painel Inferior da Janela de Auditoria com Menu de Contexto

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

Mostrar alterações
: Mostra as alterações feitas na revisão selecionada para o arquivo selecionado.

Autoria das alterações
: Auditar a revisão selecionada e a revisão anterior para um arquivo selecionado, e comparar o relatório de autoria usando uma ferramenta visual de comparação. Leia Seção 4.24.2, “Diferenças de Autoria” para mais detalhes.

Mostra como arquivo de diferenças unificado
: Mostra as alterações do arquivo em um formato unificado diff. Este menu de contexto está disponível apenas para arquivos apresentados como *modificado*.

Abrir / Abrir com...
: Abre o arquivo selecionado, tanto com o visualizar padrão para o tipo de arquivo, ou com o programa que você escolheu.

Autoria...
: Abre a janela de Autoria, permitindo a você verificar a autoria até a revisão selecionada.

Reverter alterações de uma revisão
: Reverter as alterações feitas para o arquivo selecionado na revisão.

Mostrar propriedades
: Visualizar as propriedades do Subversion para o item selecionado.

Mostrar log
: Mostrar o log de revisões para um único arquivo selecionado.

Recuperar log das unificações
: Mostrar o log da revisão para um único arquivo selecionado, incluindo as alterações de combinação. Descubra mais em Seção 4.10.6, “Combinar Recursos Monitorados”.

Save revisão para...
: Salvar a revisão selecionada para um arquivo e então você terá uma versão mais antiga deste arquivo.

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

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

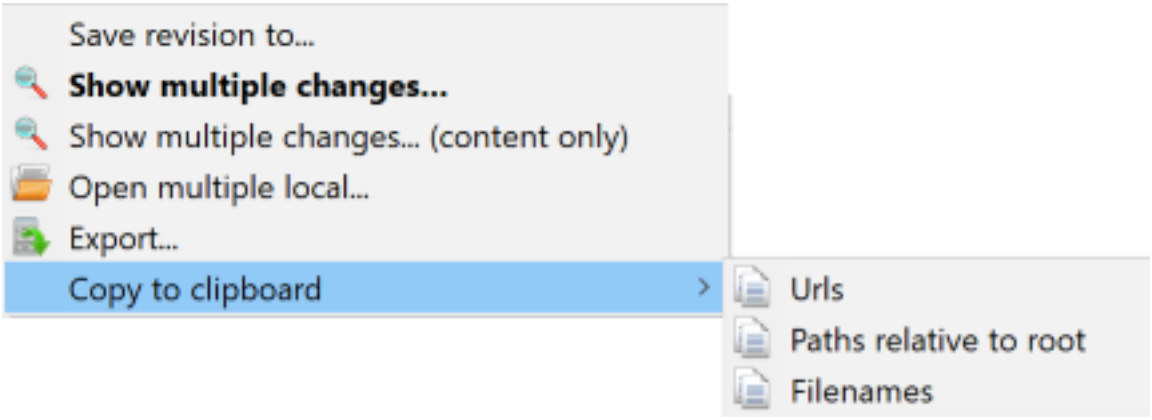


Figura 4.22. The Log Dialog Bottom Pane with Context Menu When Multiple Files Selected.

Save revisão para...
: Salvar a revisão selecionada para um arquivo e então você terá uma versão mais antiga deste arquivo.

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

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

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

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

Dica

Você deve notar que algumas vezes nós nos referimos às mudanças e outras vezes às diferenças. Qual é a diferença?

Subversion usa números de revisão para demonstrar 2 coisas diferentes. Uma revisão geralmente representa um estado do repositório em algum momento no tempo, mas também pode ser usado para representar uma conjunto de mudanças em uma revisão, por exemplo, “Feito em r1234” significa que as alterações submetidas em r1234 implementam a funcionalidade X. Para deixar mais claro a maneira que isto está sendo usando, nós usamos dois termos diferentes.

Se você selecionar duas revisões N e M, o menu de contexto oferecerá mostrar a *diferença* entre essas duas revisões. Nos termos do Subversion isto é `diff -r M:N`.

Se você selecionar uma única revisão N, o menu de contexto oferecerá mostrar as *alterações* feitas nesta revisão. Nos termos do Subversion é `diff -r N-1:N` ou `diff -c N`.

O painel inferior mostra os arquivos alterados em todas as revisões selecionadas, então o menu de contexto sempre oferecerá mostrar as *diferenças*

4.10.4. Obtendo mais mensagens de log

A caixa de diálogo Log nem sempre motra todas as alterações já feitas por uma série de motivos:

- Para um grande repositório talvez haja centenas ou milhares de alterações e recuperar todas eles pode levar um bom tempo. Normalmente você está interessado apenas nas alterações mais recentes. Por padrão, o número de mesagens recuperadas do log é limitado a 100, mas você pode alterar este valor em TortoiseSVN → Preferências (Seção 4.31.1.2, “Janela 1 de Configurações do TortoiseSVN”),

- Quando a caixa Parar para copiar/renomear é marcada, Mostrar Log irá parar no ponto de onde o arquivo selecionado ou a pasta foram copiados de outro lugar no repositório. Isto pode ser útil quando observando ramos (ou etiquetas) porque ele pára na raiz do ramo, e dá uma indicação rápida das mudanças que foram feitas apenas naquele ramo.

  Normalmente você vai querer deixar esta opção desmarcada. TortoiseSVN lembrará o estado da caixa de opção, então ele respeitará sua preferência.

  Quando a janela Mostrar Log é chamada da janela de Combinação, a caixa é sempre marcada por padrão. Isto é porque uma combinação está sempre procurando mudanças nos ramos, e voltar além da raiz de um ramo não faz sentido neste caso.

  Note que o Subversion implementa atualmente a ação de renomear como um par cópia/deleção, portanto renomear um arquivo ou pasta irá fazer com que o display de log pare se esta opção tiver marcada.

Se você quer ver mais mensagens de log, clique no Próximos 100 para recuperar as próximas 100 mensagens de log. Você pode repetir isso quantas vezes for preciso.

Próximo a este botão existe um botão multi-funcional que lembra a última opção que você usou. Clique na seta para ver as outras opções fornecidas.

Use Mostrar Faixa ... se você quiser ver uma faixa de revisões específica. Uma janela irá pedir que você entre as revisões inicial e final.

Use Mostrar Todas se você quiser ver *todas* as mensagens de log de HEAD até a revisão 1.

Para atualizar as últimas revisões no caso de terem havido outros commits enquanto a caixa de diálogo de log esteve aberta, pressione a tecla **F5**.

Para atualizar o chache de log, pressione as teclas **Ctrl-F5**.

4.10.5. Revisão da Cópia de Trabalho Atual

Como a janela de log mostra o log de HEAD, não da revisão da cópia de trabalho atual, é comum que haja mensagens de log para conteúdo que ainda não foi atualizado na cópia de trabalho. Para ajudar a deixar isto mais claro, a mensagem de submissão que corresponde à revisão que você têm na sua cópia de trabalho é mostrada em negrito.

Quando você exibe o log para uma pasta, a revisão destacada é a maior revisão encontrada dentro daquela pasta, o que requer uma pesquisa da cópia de trabalho. A pesquisa é realizada em um processo separado para não demorar para mostrar o log, mas com isso, o destaque de pastas pode não aparecer imediatamente.

4.10.6. Combinar Recursos Monitorados

Subversion 1.5 e posteriores mantém um registro das combinações usando propriedades. Isto permite que nós tenhamos um histórico mais detalhado das mudanças combinadas. Por exemplo, se você deselvolve um novo recurso em um ramo e então combina este ramo de volta com o tronco, o desenvolvimento do recurso será mostrado no log do tronco como uma única submissão para a combinação, mesmo que possa haver 1000 submissões durante o desenvolvimento no ramo.

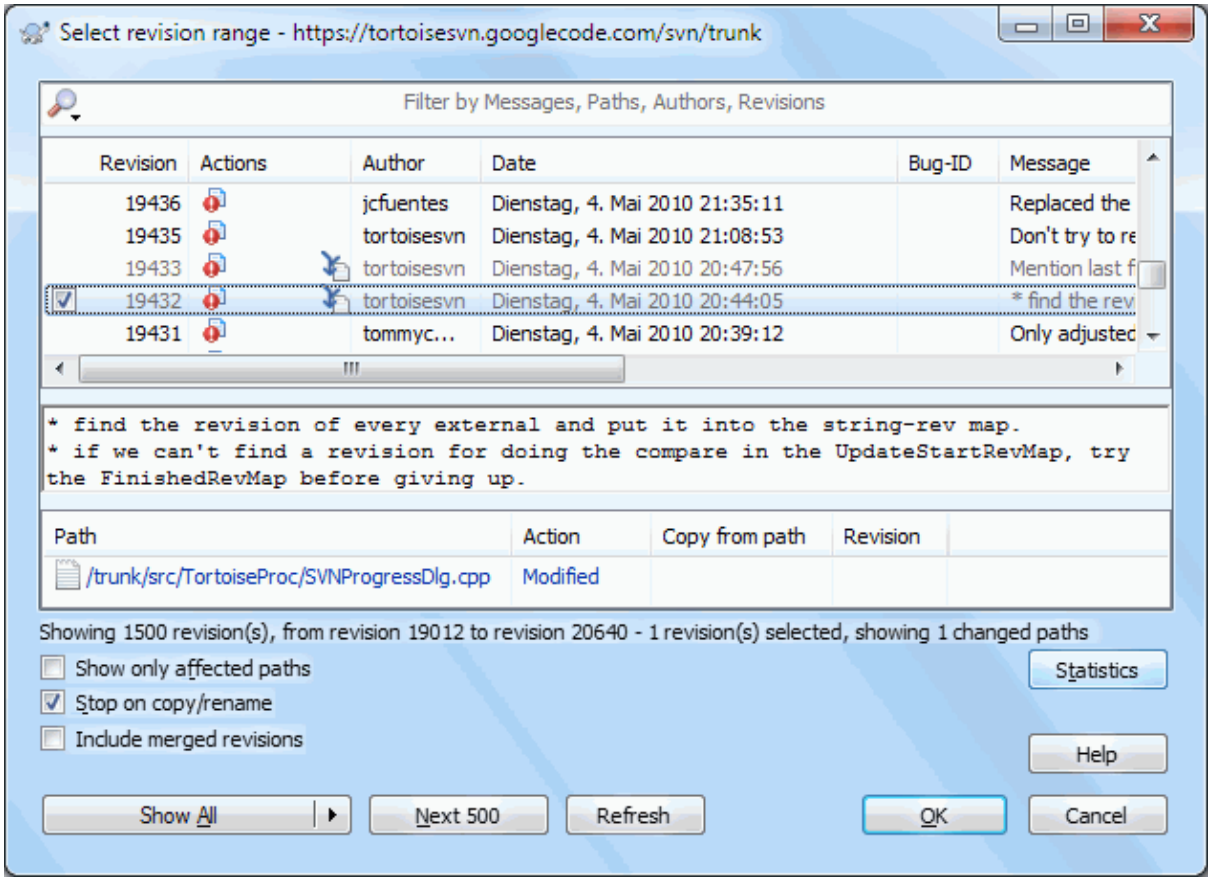


Figura 4.23. A Janela de Auditoria Mostrando as Revisões de Rastreamento de Combinação

Se você quiser ver os detalhes de quais revisões foram combinadas como parte da submissão, use a caixa de marcar Incluir revisões combinadas. Isto irá obter as mensagens de log novamente, mas também irá intercalar as mensagens de log das revisões que foram combinadas. revisões combinadas são mostradas em cinza porquê representam mudanças feiras em partes diferentes da árvore.

É claro que combinar nunca é simples! Durante o desenvolvimento de um recurso no ramo, provavelmente irão ocorrer combinações com o tronco para manter o ramo sincronizado com a linha de código principal. Então o histórico de combinações do ramo também irá mostrar outra camada de histórico de combinações. Estas camadas diferentes são mostradas na janela de log usando níveis de indentação.

4.10.7. Alterando a Mensagem de Auditoria e o Autor

Propriedades de revisão são completamente diferentes das propriedades da Subversion de cada item. Propriedades de revisão são itens descritivos que são associados com um número de revisão específico no repositório, como uma mensagem de log, data de submissão e nome do autor.

Algumas vezes você pode querer mudar uma mensagem de log que você entrou, talvez porque há erros ou porque você quer melhorar a mensagem ou mudá-la por outras razões. Ou você quer mudar o autor da submissão porque você esqueceu de configurar a autenticação ou...

Subversion lets you change revision properties any time you want. But since such changes can't be undone (those changes are not versioned) this feature is disabled by default. To make this work, you must set up a pre-revprop-change hook. Please refer to the chapter on *Hook Scripts* [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.create.html#svn.reposadmin.create.hooks] in the Subversion Book for details about how to do that. Read Seção 3.3, “Rotinas de eventos no servidor” to find some further notes on implementing hooks on a Windows machine.

Uma vêz que você já configurou seu servidor com os hooks necessários, você pode mudar o autor e a mensagem de log (ou qualquer outra propriedade de revisão) de qulquer revisão, usando o menu de contexto do painel superior da janela de Log. Você também pode editar uma mensagem de log usando o menu de contexto do painel do meio.

Atenção

Como as propriedades de revisão da Subversion não são versionadas, fazer modificações em tais propriedades (por exemplo, a propriedade mensagem de submissão `svn:log`) irá sobrescrever os valores anteriores da propriedade *para sempre*.

Importante

Since TortoiseSVN keeps a cache of all the log information, edits made for author and log messages will only show up on your local installation. Other users using TortoiseSVN will still see the cached (old) authors and log messages until they refresh the log cache. Refer to Seção 4.10.11, “Atualizando a Visualização”

4.10.8. Filtrando Mensagens de Log

Se você quiser restringir as mensagen de log para mostrar apenas às que você está interessado ao invés de rolar por toda a lista de milhares, você pode usar os controles de filtro no topo da Janela de Log. Os controles de datas inicial e final permitem que você restrinja a saída à uma faixa de datas conhecida. A caixa de busca permite que você mostre apenas as mensagens que contenham uma frase particular.

Clique no ícone de busca para selecionar a informação na qual você quer pesquisar e escolha o modo *regex*. Normalmente você irá precisar apenas de uma busca simples de palavras, mas se você precisar de termos de busca mais flexíveis, você pode usar expressões regulares. Se você colocar o curosor sobre a caixa, uma dica de tela irá dar dicas de como usar funções de expressões regulares. O filtro funciona checando quando o texto do filtro combina com as entradas de log, e então apenas estas entradas que *combinam* com o filtro são exibidas.

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

```
Alice Bob -Eve
```

searches for strings containing both Alice and Bob but not Eve

```
Alice -Bob +Eve
```

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

```
-Case +SpecialCase
```

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

```
!Alice Bob
```

searches for strings which do not contain both Alice and Bob

```
!-Alice -Bob
```

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

```
"Alice and Bob"
```

searches for the literal expression “Alice and Bob”

```
""
```

searches for a double-quote anywhere in the text

```
"Alice says ""hi"" to Bob"
```

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

Descrever o uso de expressões regulares está além do escopo deste manual, mas você pode encontrar documentações online e um tutorial em http://www.regular-expressions.info/.

Note que estes filtros atuam nas mensagens já recuperadas. Eles não controlam a recuperação das mensagens do repositório.

Você também pode filtrar os nomes de caminho no painel de baixo usando a caixa Mostrar apenas caminhos afetados. Caminhos afetados são os que contém o caminho usado para mostrar o log. Se você obter o log para uma pasta, isto significa qualquer coisa na pasta ou abaixo dela. Para um arquivo, isto significa apenas aquele arquivo. Normalmente a lista de caminhos mostra quaisquer outros caminhos que são afetados pela mesma submissão, mas em cinza. Se a caixa é marcada, estes caminhos são ocultos.

Algumas vezes suas práticas de trabalho irão precisar que as mensagens de log sigam um formato particular, o que significa que o texto descrevendo as mudanças não é visível no resumo abreviado que é exibido no painel superior. A propriedade `tsvn:logsummary` pode ser usada para extrair uma parte da mensagem de log para ser exibida no painel superior. Leia Seção 4.18.2, “Propriedades do Projeto TortoiseSVN” para saber como usar esta propriedade.

Nenhuma Formatação de Auditoria do Navegador do Repositório

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

4.10.9. Informação Estatística

O botão Estatísticas tráz uma caixa exibindo algumas informações interessantes sobre as revisões mostradas na janela de Log. Isto mostra quantos autores trabalharam, quantas submissões eles fizeram, progresso por semana, e muito mais. Agora você pode ter um relance de quem tem trabalhado duro e quem está enrolando ;-)

4.10.9.1. Página de Estatísticas

Esta página dará a você todos os números que você pode imaginar sobre, em um período específico e em uma faixa de revisões, e alguns valores mín/máx/médios.

4.10.9.2. Submissões por Página do Autor

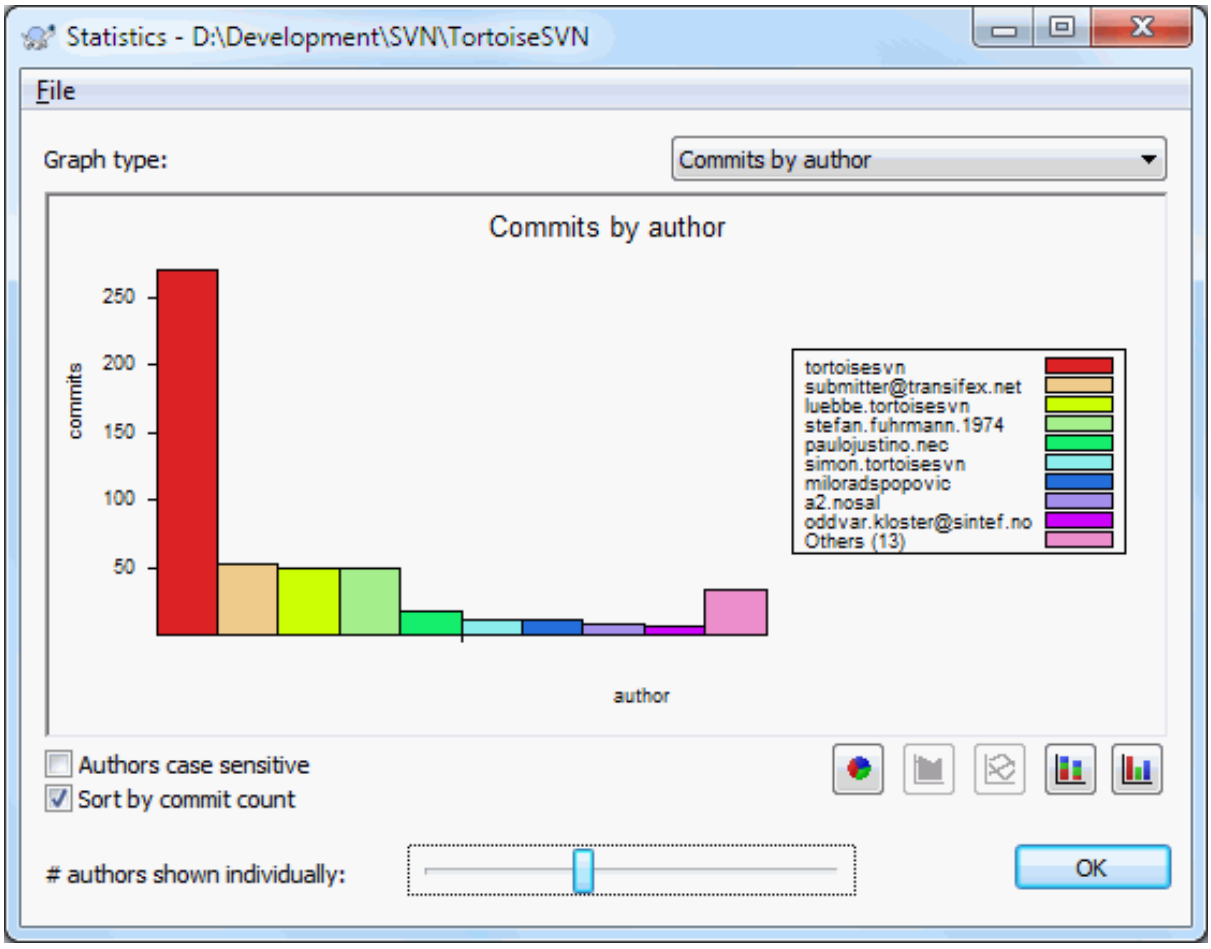


Figura 4.24. Histograma de Submissões-por-autor

Este gráfico mostra a você quais autores têm estado ativos no projeto com um histograma simples, histograma de barras ou gráfico de pizza.

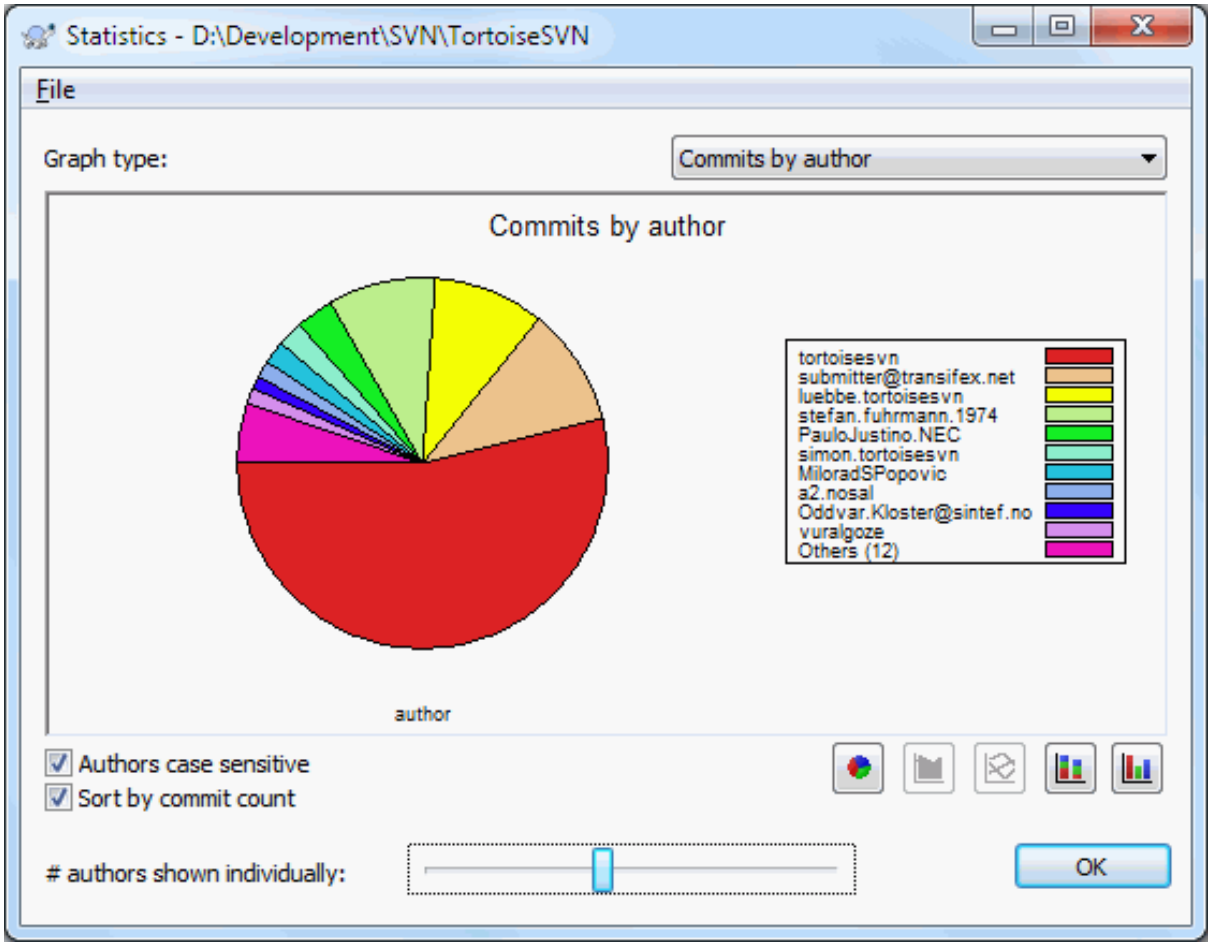


Figura 4.25. Gráfico de Pizza das Submissões-por-Autor

Quando houver poucos autores principais e muitos contribuintes menores, o número de segmentos pequenos pode fazer o gráfico mais difícil de ler. O controle deslizante abaixo permite que você ajuste um limite (como uma porcentagem do total de submissões) abaixo do qual qualquer atividade é agrupada na categoria *Outros*.

4.10.9.3. Submissões por Página de Data

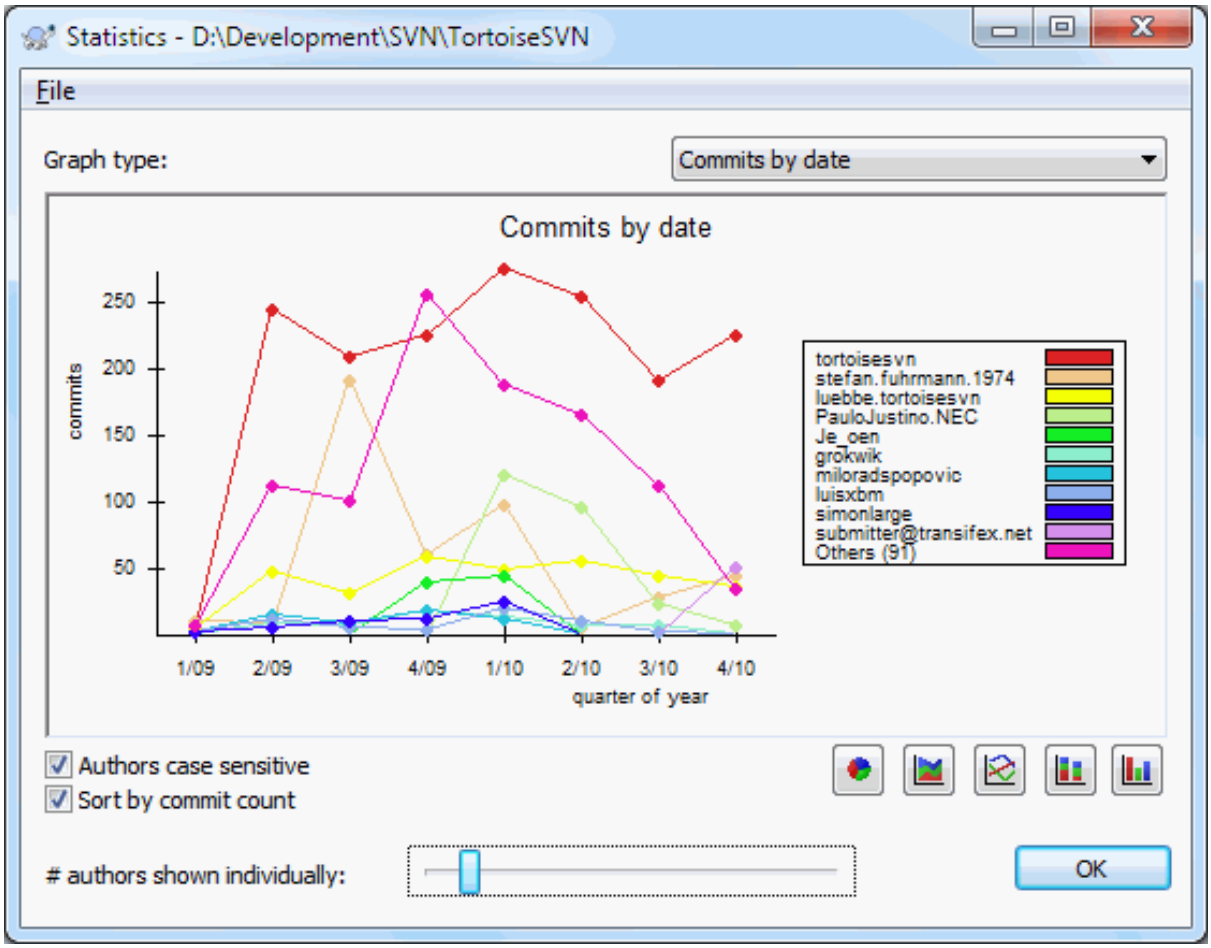


Figura 4.26. Gráfico de Submissões-por-data

Esta página dá a você uma representação gráfica da atividade do projeto em termos de número de submissões *e* autor. Isso dá alguma ideia de quando um projeto está sendo trabalhado e quem estava trabalhando em que tempo.

Quando existem vários autores, você verá muitas linhas no gráfico. Existem duas visões disponíveis aqui: *normal*, onde a atividade de cada autor é relativa à linha de base, e *compactado*, onde a atividade de cada autor é relativa à linha de baixo. A última opção evita linhas cruzadas, a qual permite uma gráfico mais fácil de ser lido, porém mais difícil de visualizar as informações de um único autor.

Por padrão a analise é case-sensitive, portanto usuários como `PeterEgan` e `PeteRegan` são considerados autores diferentes. Entretanto, em muitos casos os nomes de usuários não são case-sensitive, e às vezes são inseridos de forma inconsistente, então você pode querer que `DavidMorgan` e `davidmorgan` sejam considerados a mesma pessoa. Use a seleção Authors case insensitive para escolher como lidar com isso.

Observe que as estatísticas cobrem o mesmo período que o diálogo de Log. Se este está mostrando apenas uma revisão então as estatísticas não irão informar muito.

4.10.10. Modo desconectado

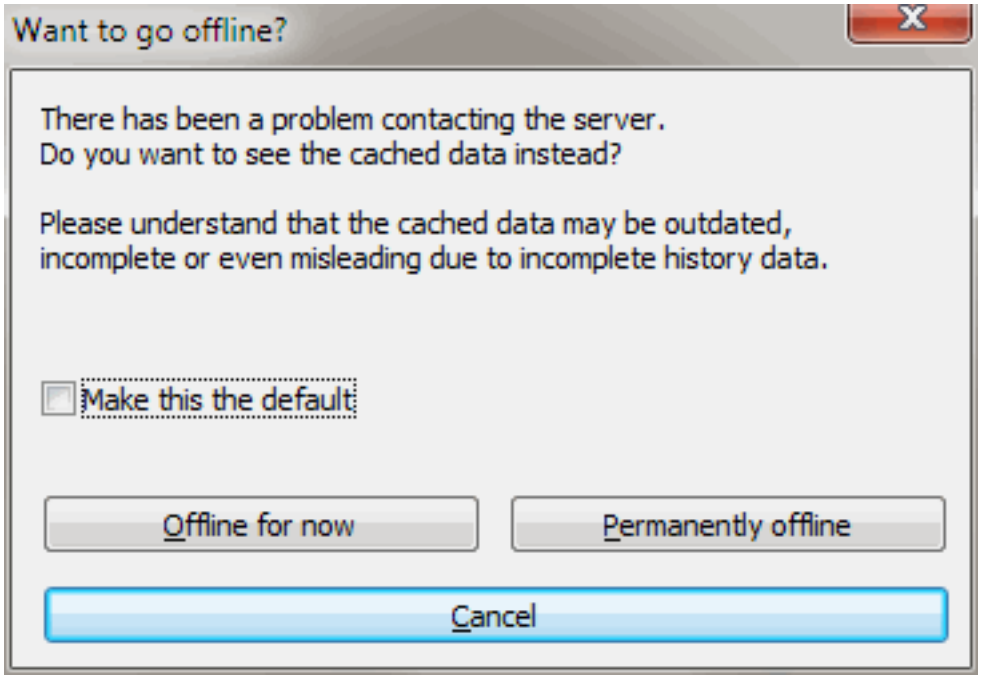


Figura 4.27. Ir para Janela de Desconectado

Se o servidor não está acessível, e você tem o habilitar o cache de log, você pode usar a caixa de diálogo e gráfico de revisão no modo offline. Este usa dados do cache, o que lhe permite continuar a trabalhar, embora a informação não possa ser atualizada ou mesmo completada.

Aqui você tem três opções:

Desconectado agora
: Complete a operação corrente em modo offline, mas obtenha o repositório da próxima vez que os dados de log forem solicitados.

Permanentemente desconectado
: Permaneça em modo offline até que uma verificação de repositório for especificamente requisitada. Veja Seção 4.10.11, “Atualizando a Visualização”.

Cancelar
: Se você não quiser continuar a operação com dados possivelmente obsoletos, basta cancelar.

A caixa de opção Torne isto o padrão previne que este diálogo reapareça e sempre seleciona a próxima opção que você escolher. Você ainda pode mudar (ou remover) o padrão depois de fazer isso em TortoiseSVN → Configurações.

4.10.11. Atualizando a Visualização

Se você quiser verificar novamente o servidor por novas mensagens de auditoria, você pode simplesmente atualizar a visualização usando **F5**. Se você estiver usando o cache de auditoria (habilitado por padrão), isto irá verificar o repositório por mensagens mais novas e buscar apenas as novas. Se o cache de auditoria estiver em modo desativado, isto também tentará colocá-lo em modo disponível.

Se você estiver usando o cache de auditoria e achar que o conteúdo ou o autor da mensagem mudou, você pode usar **Shift-F5** ou **Ctrl-F5** para buscar novamente do servidor as mensagens apresentadas e atualizar o cache de

auditoria. Note que isto afeta apenas as mensagens atualmente mostradas e não invalida inteiramente o cache deste repositório.

4.11. Visualizando as Diferenças

Um dos requisitos mais comuns no desenvolvimento de projetos é ver o que foi modificado. Você pode querer ver as diferenças entre duas revisões de um mesmo arquivo, ou as diferenças entre dois arquivos separados. TortoiseSVN fornece uma ferramenta interna chamada TortoiseMerge para visualizar diferenças em arquivos de texto. Para visualizar diferenças em arquivos de imagem, TortoiseSVN também possui uma ferramenta chamada TortoiseIDiff. Claro, você pode usar seu programa favorito de diferenciação se quiser.

4.11.1. Diferenças do Arquivo

Alterações Locais
: Se você quiser ver as modificações que *você* fez na sua cópia de trabalho, apenas use o menu de contexto do explorer e selecione TortoiseSVN → Diferenciar.

Diferença para outro ramo/destino
: Se você quiser ver o que foi modificado no tronco (se você estiver trabalhando em um ramo) ou em um ramo específico (se você estiver trabalhando no tronco), você pode usar o menu de contexto do explorer. Apenas mantenha pressionado a tecla **Shift** enquanto clica com o botão direito no arquivo. Então selecione TortoiseSVN → Diferenciar com URL. No diálogo seguinte, especifique a URL do repositório com o qual você quer comparar com seu arquivo local.

  Você também pode usar o navegador de repositório e selecionar duas árvores para diferenciar, talvez duas etiquetas, ou um ramo/etiqueta e o tronco. O menu de contexto apresentado possibilita a você compará-los usando Comparar revisões. Leia mais em Seção 4.11.3, “Comparando Diretórios”.

Diferenças da revisão anterior
: Se você quer ver a diferença entre uma revisão particular e sua cópia de trabalho, utilize o diálogo Revisão de Log, selecione a revisão do seu interesse, então selecione Comparar com cópia de trabalho do menu de contexto.

  Se você quiser ver a diferença entre a última revisão submetida e sua cópia de trabalho, assumindo que a cópia de trabalho não foi modificada, apenas clique com o botão direito no arquivo. Então selecione TortoiseSVN → Diferenciar com a versão anterior. Isto irá realizar a diferenciação entre a revisão anterior à data da última submissão (como gravada na sua cópia de trabalho) e a base de trabalho. Isto mostra a você a última modificação feita naquele arquivo para trazê-lo ao estado que você agora vê em sua cópia de trabalho. Não serão mostradas modificações mais recentes que sua cópia de trabalho.

Diferença entre as duas últimas revisões
: Se você quiser ver a diferença entre duas revisões que já foram submetidas, use o diálogo de Auditoria de Revisões e selecione as duas revisões que você quer comparar (usando o habitual modificador **Ctrl**). Então selecione Comparar revisões no menu de contexto.

  Se você fez isso da auditoria de revisão para um diretório, uma janela de Comparação de Revisões aparecerá, mostrando a lista de arquivos alterados no diretório. Leia mais em Seção 4.11.3, “Comparando Diretórios”.

Todas as alterações feitas em uma submissão
: Se você quiser ver as mudanças feitas a todos os arquivos de uma revisão particular em uma visão, você pode usar a saída Comparação-Unificada (formato de correção GNU). Isso mostra apenas as diferenças com poucas linhas de contexto. É mais difícil de ler do que em uma comparação de arquivo visual, mas irá mostrar todas as mudanças juntas. A partir da janela de Auditoria de Revisão selecione a revisão de interesse, então selecione Mostrar Diferenças como Comparação-Unificada no menu de contexto.

Diferença entre arquivos
: Se você quiser ver as diferenças entre dois arquivos diferentes, você pode fazer isso diretamente no Explorer selecionando ambos os arquivos (usando o modificador usual **Ctrl**). Então, a partir do menu de contexto do Explorer, selecionar TortoiseSVN

If the files to compare are not located in the same folder, use the command TortoiseSVN → Diff later to mark the first file for diffing, then browse to the second file and use TortoiseSVN → Diff with "path/of/marked/file". To remove the marked file, use the command TortoiseSVN → Diff later again, but hold down the **Ctrl**-modifier while clicking on it.

Diferença entre o arquivo/diretório CT e uma URL
: Se você quiser ver as diferenças entre um arquivo na sua cópia de trabalho e um arquivo em um repositório Subversion, você pode fazer isso diretamente no Explorer selecionando o arquivo e então mantendo pressionada a tecla **Shift** enquanto clica com o botão direito do mouse para obter o menu de contexto. Selecione TortoiseSVN → Diferenciar com URL. Você pode fazer o mesmo para uma pasta na cópia de trabalho. TortoiseMerge mostra estas diferenças da mesma maneira que ele mostra um arquivo de correção - uma lista de arquivos modificados que você pode ver um de cada vez.

Diferença com a informação de autoria
: Se você quiser ver não apenas as diferenças mas também o autor, a revisão e a data em que as mudanças foram feitas, você pode combinar os relatórios de diferenças e de responsabilizações a partir da janela de auditoria de revisões. Leia Seção 4.24.2, “Diferenças de Autoria” para mais detalhes.

Diferença entre diretórios
: 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 Seção 4.11.6, “Ferramentas de Diferenciação/Combinação Externas” 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 Seção 4.31.5, “Configuração de Programa Externo” to find out about configuring other diff tools.

4.11.2. Opções de Quebra de Linha e Espaços em Branco

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.

Ignorar fim de linha exclui alterações que são exclusivamente ara diferenciar o identificador de fim de linha.

Comparar espaços em branco inlcui todas as alterações de indentação e espaços em branco como linhas adicionadas/removidas.

Ignore whitespace changes excludes changes which are due solely to a change in the amount or type of whitespace, e.g. 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.

Ignorar todos os espaços em branco exclui todas as mudanças que são apenas espaços em branco.

Naturalmente, qualquer linha com conteúdo alterado é sempre incluída na diferenciação.

4.11.3. Comparando Diretórios

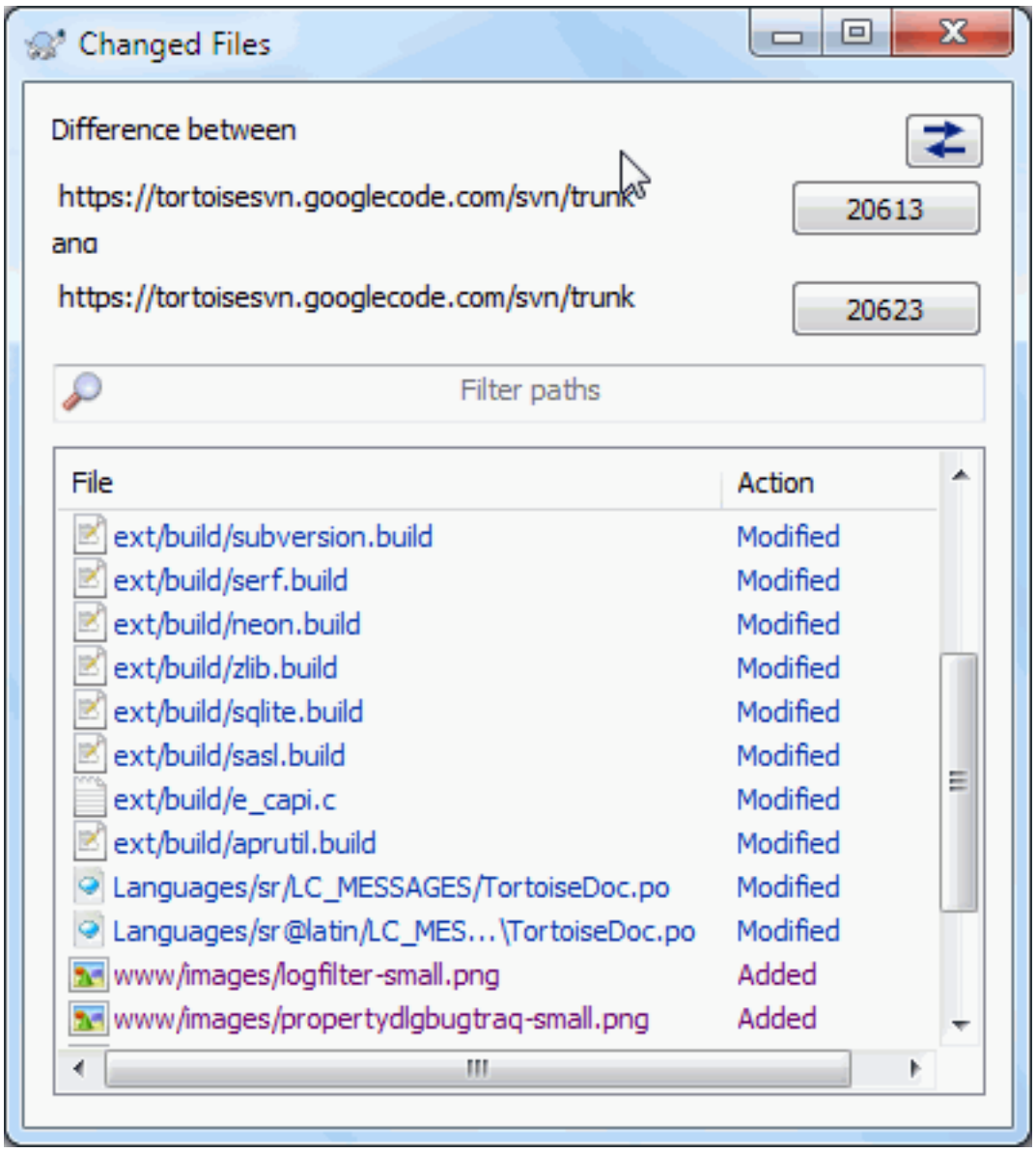


Figura 4.28. A Janela de Comparação de Revisões

When you select two trees within the repository browser, or when you select two revisions of a folder in the log dialog, you can Context menu → Compare revisions.

Esta janela mostra a lista de todos os arquivos que sofreram alteração e permite a você comparar ou identificar seu autor individualmente usando o menu de contexto.

You can export a *change tree*, which is useful if you need to send someone else your project tree structure, but containing only the files which have changed. This operation works on the selected files only, so you need to select the files of interest - usually that means all of them - and then Context menu → Export selection to.... You will be prompted for a location to save the change tree.

Você pode também exportar a *lista* de arquivos alterados para um arquivo texto usando Menu de contexto → Salvar lista de arquivos selecionados para....

Se você quer exportar a lista de arquivos *e* também as ações (modificado, adicionado, apagado), você pode fazê-lo usando Menu de contexto → Copiar seleção para área de transferência.

O botão no topo permite a você mudar a direção da comparação. Você pode mostrar as alterações realizadas de A para B, ou se você preferir, de B para A.

Os botões contendo os números de revisão podem ser usados para mudar para uma faixa de revisão diferente. Quando você mudar a faixa, a lista de itens os quais diferem em duas revisões serão automaticamente atualizados.

Se a lista de arquivos é muito extensa, você pode usar a caixa de busca para reduzir a lista de nomes de arquivos para apenas os que contém o texto específico. Note que uma busca por texto simples é usada, então se você quer restringir a lista de arquivos para código em C você deve digitar `.c` ao invés de `*.c`.

4.11.4. Comparando Imagens Utilizando o TortoiseIDiff

Existem muitas ferramentas disponíveis para comparar arquivos texto, incluindo nosso próprio TortoiseMerge, porém nós frequentemente nos pegamos querendo ver as modificações em uma imagem além de somente textos. Por isso nós criamos o TortoiseIDiff.

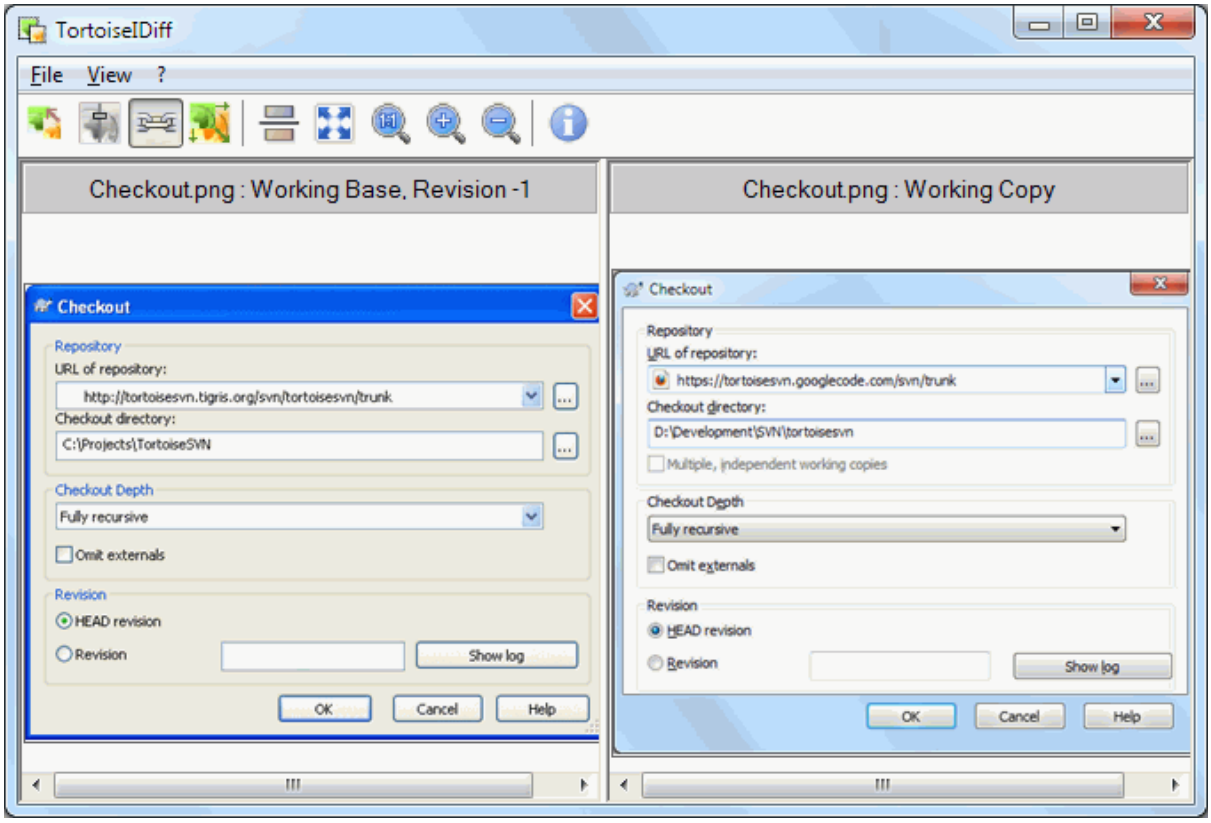


Figura 4.29. O visualizar de diferenças de imagem

TortoiseSVN → Diff for any of the common image file formats will start TortoiseIDiff to show image differences. By default the images are displayed side-by-side but you can use the View menu or toolbar to switch to a top-bottom view instead, or if you prefer, you can overlay the images and pretend you are using a lightbox.

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.

The button above the slider toggles between 0% and 100% blends, and if you double click the button, the blend toggles automatically every second until you click the button again. This can be useful when looking for multiple small changes.

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.

4.11.5. Diffing Office Documents

When you want to diff non-text documents you normally have to use the software used to create the document as it understands the file format. For the commonly used Microsoft Office and Open Office suites there is indeed some support for viewing differences and TortoiseSVN includes scripts to invoke these with the right settings when you diff files with the well-known file extensions. You can check which file extensions are supported and add your own by going to TortoiseSVN → Settings and clicking Advanced in the External Programs section.

Problemas com o Office 2010

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

4.11.6. Ferramentas de Diferenciação/Combinação Externas

Se as ferramentas providas por nós não faz o que você precisa, tente uma das muitas disponíveis de código aberto ou comerciais. Todo mundo tem suas favoritas, e esta lista não tem o propósito de estar completa, mas aqui estão algumas que nós devemos consideramos:

WinMerge
: *WinMerge* [https://winmerge.sourceforge.net/] is a great open-source diff tool which can also handle directories.

Forçar Unificação
: Perforce is a commercial RCS, but you can download the diff/merge tool for free. Get more information from *Perforce* [https://www.perforce.com/perforce/products/merge.html].

KDiff3
: KDiff3 é uma ferramenta livre que também suporta diretórios. Você pode baixá-la *aqui* [http://kdiff3.sf.net/].

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

ExaminarDiferenças
: ExamDiff Standard is freeware. It can handle files but not directories. ExamDiff Pro is shareware and adds a number of goodies including directory diff and editing capability. In both flavours, version 3.2 and above can handle unicode. You can download them from *PrestoSoft* [http://www.prestosoft.com/].

Beyond Compare
: Similar to ExamDiff Pro, this is an excellent shareware diff tool which can handle directory diffs and unicode. Download it from *Scooter Software* [https://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* [https://www.araxis.com/merge/index.html].

Leia Seção 4.31.5, “Configuração de Programa Externo” para mais informações sobre como configurar o TortoiseSVN para usar essas ferramentas.

4.12. Adicionando Novos Arquivos e Diretórios

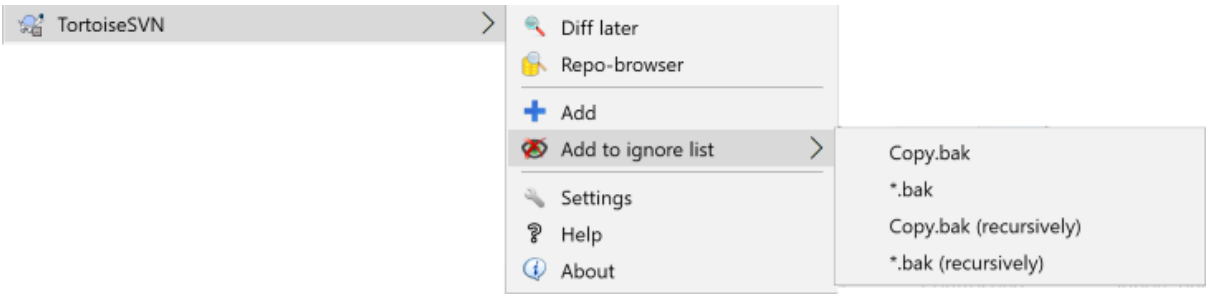


Figura 4.30. Explorar o menu de contexto para arquivos não versionados

If you created new files and/or directories during your development process then you need to add them to source control too. Select the file(s) and/or directory and use TortoiseSVN → Add.

After you added the files/directories to source control the file appears with a `added` icon overlay which means you first have to commit your working copy to make those files/directories available to other developers. Adding a file/directory does *not* affect the repository!

Muitas Adições

Você pode usar também o comando Adicionar em diretório já versionados. Neste caso a janela de adição mostrar a você todos os arquivos não versionados contido no diretório já controlado. Esta função ajuda se você tem muitos novos arquivos e precisa adicioná-los todos de uma vez.

Para adicionar arquivos de fora da sua cópia de trabalho você pode usar o controle de arrastar-e-soltar:

1. selecione os arquivos que você quer adicionar
2. arrasto direito neles para a nova localização dentro da cópia de trabalho
3. solte o botão direito do mouse
4. selecione Menu de Contexto → SVN Adicionar arquivos para está CT. Os arquivos serão então copiados para a cópia de trabalho e adicionados ao controle de versão.

Você pode também adicionar arquivos dentro da sua cópia de trabalho apenas arrastando com o botão esquerdo e soltando eles na janela de submissão.

Se você adicionar um arquivo ou diretório por engano, você pode desfazer a adição antes de submeter usando TortoiseSVN → Desfazer Adição....

4.13. Copiando/Movendo/Renomeando Arquivos e Diretórios

It often happens that you already have the files you need in another project in your repository, and you simply want to copy them across. You could simply copy the files and add them, but that would not give you any history. And if you subsequently fix a bug in the original files, you can only merge the fix automatically if the new copy is related to the original in Subversion.

The easiest way to copy files and folders from within a working copy is to use the right drag menu. When you right drag a file or folder from one working copy to another, or even within the same folder, a context menu appears when you release the mouse.

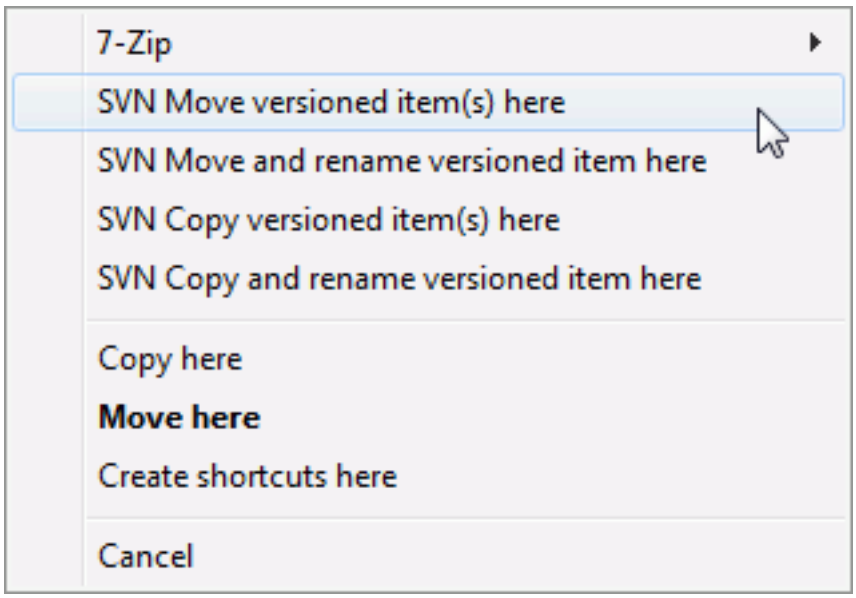


Figura 4.31. Menu de quando se clica com o botão direito e se arrasta um diretório que está sob o controle de versão.

Now you can copy existing versioned content to a new location, possibly renaming it at the same time.

You can also copy or move versioned files within a working copy, or between two working copies, using the familiar cut-and-paste method. Use the standard Windows Copy or Cut to copy one or more versioned items to the clipboard. If the clipboard contains such versioned items, you can then use TortoiseSVN → Paste (note: not the standard Windows Paste) to copy or move those items to the new working copy location.

You can copy files and folders from your working copy to another location in the repository using TortoiseSVN → Branch/Tag. Refer to Seção 4.20.1, “Criando um Ramo ou Rótulo” 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 Seção 4.10.3, “Recuperando Informações Adicionais” to find out more.

Você também pode usar o navegador de repositório para localizar o conteúdo que procura e copiá-lo para a sua cópia de trabalho diretamente do repositório, ou copiar para outro local dentro do próprio repositório. Veja mais em Seção 4.25, “O Navegador de Repositório”.

Não pode copiar entre repositórios

Whilst you can copy or move files and folders *within* a repository, you *cannot* copy or move from one repository to another while preserving history using TortoiseSVN. Not even if the repositories live on the same server. All you can do is copy the content in its current state and add it as new content to the second repository.

If you are uncertain whether two URLs on the same server refer to the same or different repositories, use the repo browser to open one URL and find out where the repository root is. If you can see both locations in one repo browser window then they are in the same repository.

4.14. Ignorando Arquivos e Diretórios

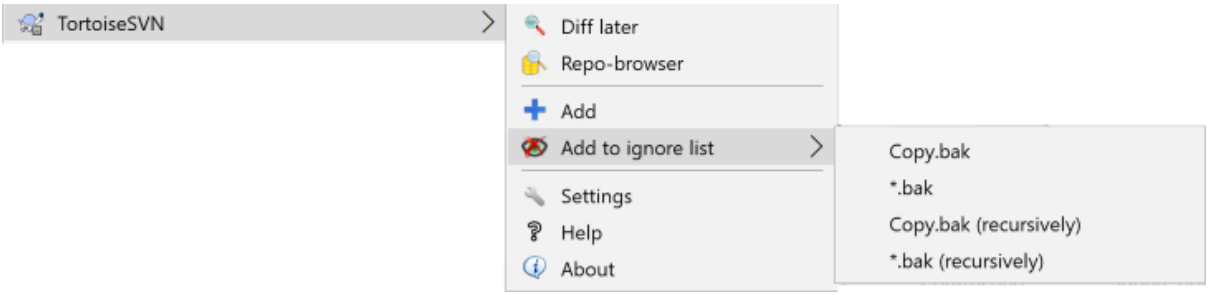


Figura 4.32. Explorar o menu de contexto para arquivos não versionados

Na maioria dos projetos você terá arquivos e diretórios que não deverão ser controlados. Entre eles estão arquivos criados pelo compilador, <nome_do_arquivo>*.obj, *.lst</nome_do_arquivo>, talvez um diretório de saída usado para gravar o executável. Não importa que você submeta alterações, TortoiseSVN mostrará seus arquivos não controlados, os quais serão apresentados na lista da janela de submissão. Claro, você pode optar por não mostrá-los, mas então você poderá esquecer de adicionar alguma novo arquivo de código.

A melhor maneira de se livrar desses problemas é adicionar esses arquivos na lista de arquivos ignorados do projeto. Dessa forma eles nunca serão mostrados na lista da janela de submissão, mas arquivos genuinos de código não controlados aparecerão para serem adicionados.

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

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

Se você quer remover um ou mais itens da lista de arquivos ignorados, clique com o botão direito sobre os itens e selecione TortoiseSVN → Remover da lista de ignorados. Você pode também acessar a propriedade `svn:ignore` do diretório diretamente. Isto lhe permite especificar regras gerais usando o nome dos arquivos globais, descrito na próxima seção. Leia Seção 4.18, “Configurações do Projeto” para mais informações sobre como definir as propriedades diretamente. Por favor tenha o cuidado de colocar cada regra em uma linha separada. Separá-las por espaço não funciona.

A Lista Global de Arquivos Ignorados

Outra forma de ignorar arquivos é adicionar eles para a *lista global de arquivo*. A grande diferença é que a lista global de arquivos ignorados é uma propriedade no cliente. Isto é aplicado parar *todos* projetos no Subversion, but somente no PC cliente. Em geral o melhor é usar a propriedade `svn:ignore` onde possível, porque pode ser aplicado para áreas específicas do projeto, e isto funciona para todos que obterem o projeto. Leia Seção 4.31.1, “Configurações Gerais” para mais informações.

Ignorando Arquivos Controlados

Arquivos e diretórios controlados nunca devem ser ignorados - esta é uma característica do Subversion. Se você está controlando um arquivo mas não deveria, leia Seção B.8, “Ignorar arquivos que já estão versionados” para instruções de como “não controlar” o arquivo.

4.14.1. Padrões de Filtro na Lista de Arquivos Ignorados

Os padrões para ingorar arquivos do Subversion fazem uso do nome de arquivo global, uma técnica originalmente usada no Unix para especificar arquivos usando meta-caracteres como coringas. Os caracteres a seguir tem um significado especial:

*
Filtra por qualquer texto de caracteres, incluindo um texto vazio (nenhum caracter).

?
Filtra por qualquer caracter simples.

[...]
Filtra por qualquer um dos caracteres dentro dos colchetes. Dentro de colchetes, um par de caracteres separados por "-" filtra qualquer caracter léxico entre os dois. Por exemplo `[AGm-p]` flitra por qualquer um dos caracteres `A`, `G`, `m`, `n`, `o` or `p`.

Pattern matching is case sensitive, which can cause problems on Windows. You can force case insensitivity the hard way by pairing characters, e.g. to ignore `*.tmp` regardless of case, you could use a pattern like `*.[Tt][Mm][Pp]`.

Se você quer uma definição oficial para nomes de arquivos globais, você pode encontrar nas especificações IEEE para linguagem de comando de integração *Pattern Matching Notation* [http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_13].

Nenhum Caminho na Lista Global de Arquivos Ignorados

Você não deverá incluir caminhos no seu padrão. O padrão de filtro tem a intensão de ser usado apenas para nomes de arquivos e diretórios. Se você quer ignorar todos os diretórios `CVS`, apenas adicione o diretório `CVS` na lista de arquivos ignorados. Não é necessário especificar `CVS */CVS` como você faria em versões anteriores. Se você quer ignorar todos os diretórios `tmp` quando existirem dentro de um diretório `prog` mas não quer quando estiverem dentro de um diretório

4.15. Apagando, Movendo e Renomeando

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

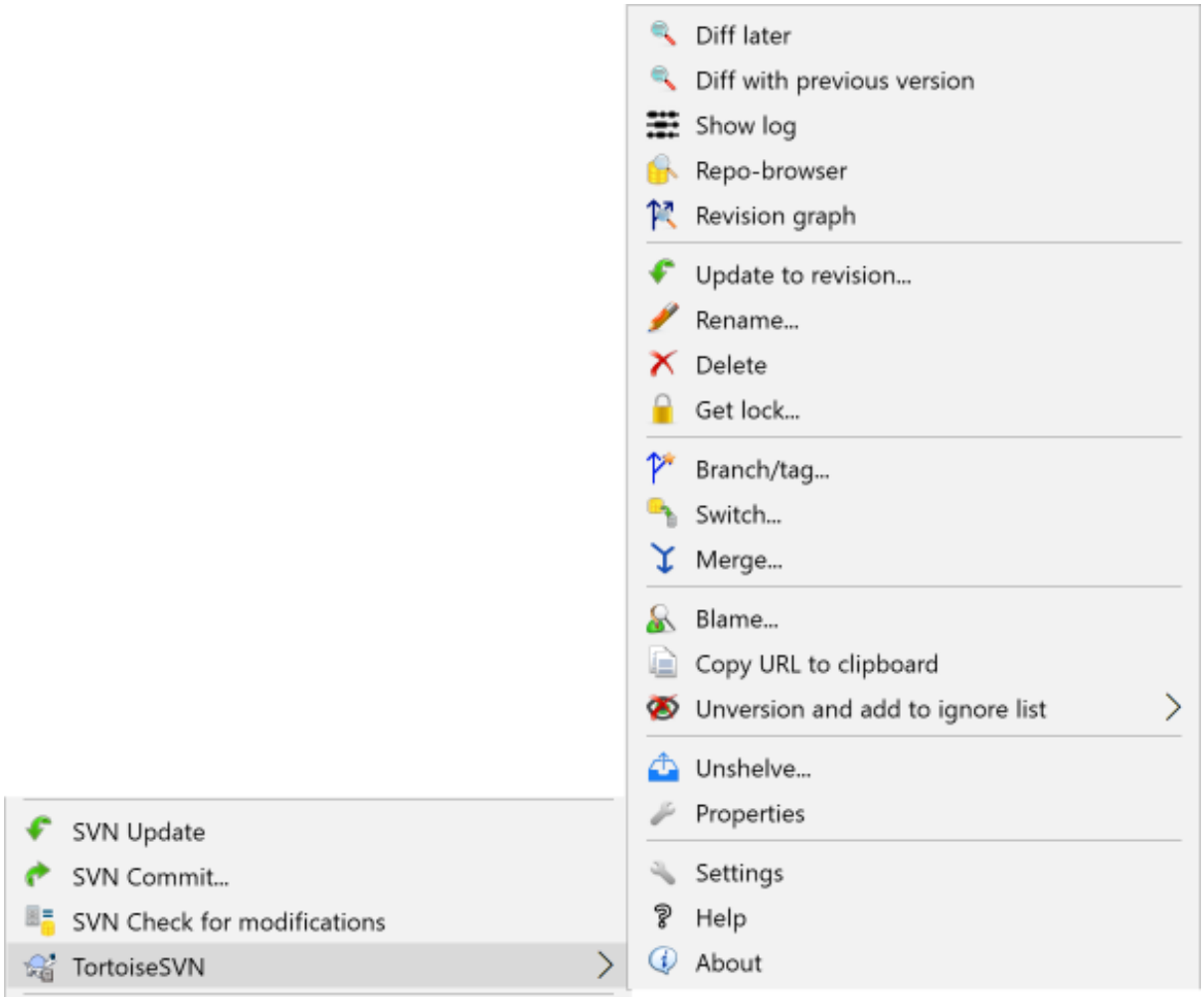


Figura 4.33. Menu de contexto do Explorerer para arquivos controlados

4.15.1. Apagando arquivos e diretórios

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

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

Se você quer apagar um item do repositório, mas manter uma versão local não versionada do arquivo/diretório, use Menu de Contexto Estendido → Excluir (manter local). Você deve manter a tecla **Shift** pressionada enquando clica com o botão direito no item desejado no painel direito do "explorer" para ver este menu de contexto estendido.

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

Recuperando arquivos e diretórios excluidos

If you have deleted a file or a folder and already committed that delete operation to the repository, then a normal TortoiseSVN → Revert can't bring it back anymore. But the file or folder is not lost at all. If you know the revision the file or folder got deleted (if you don't, use the log dialog to find out) open the repository browser and switch to that revision. Then select the file or folder you deleted, right click and select Context Menu → Copy to... as the target for that copy operation select the path to your working copy.

4.15.2. Movendo arquivos e diretórios

Se você quer fazer uma simples renomeação de arquivo ou diretório, use Menu de Contexto → Renomear... e entre com o novo nome para o item e estará feito.

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. selecione os arquivos ou diretórios que você quer mover
2. arrasto direito neles para a nova localização dentro da cópia de trabalho
3. solte o botão direito do mouse
4. no menu que se abre escolha Menu de Contexto → SVN Mover itens sob controle de versão para cá

Submetendo o diretório pai

Já que renomear e mover arquivos está feito assim como a exclusão como complemento você deve submeter o diretório pai dos arquivos renomeados/movidos assim como os excluídos/movidos que aparecerão na janela de submissão. Se você não submeteu a parte removida da ação de renomear/mover, ela continuará no repositório e quando algum outro participante do projeto atualizar sua cópia, os velhos arquivos não serão removidos. Por exemplo eles *ambas* as cópias, velhas e novas.

Você *deve* submeter um diretório renomeado antes de alterar qualquer arquivo dentro dele, caso contrário sua cópia de trabalho pode realmente se corromper.

Another way of moving or copying files is to use the Windows copy/cut commands. Select the files you want to copy, right click and choose Context Menu → Copy from the explorer context menu. Then browse to the target folder, right click and choose TortoiseSVN → Paste. For moving files, choose Context Menu → Cut instead of Context Menu → Copy.

Você pode também usar o navegador de repositório para mover itens. Leia Seção 4.25, “O Navegador de Repositório” para descobrir mais.

Não Mova Arquivos Externos

Você *não* deverá usar os comandos do TortoiseSVN Mover ou Renomear em um diretório que tenha sido criado usando `svn:externals`. Esta ação poderá causar a exclusão de um item externo em seu repositório, provavelmente chateando muitas outras pessoas. Se você precisa mover um diretório externo você deverá usar uma outra ferramenta para mover, e então ajustar as propriedades `svn:externals` do diretório de origem e de destino.

4.15.3. Procedimento em caso de conflito com o nome do arquivo

Se o repositório atualmente já contém dois arquivos com o mesmo nome diferenciados apenas pela caixa (exemplo: `TEST.TXT` e `test.txt`, você não poderá atualizar ou obter o diretório em um cliente Windows.Embora o Subversion suporte diferenciação na caixa no nome dos arquivos, o Windows não suporta.

Isto algumas vezes acontece quando duas pessoas submetem, de diferentes cópias de trabalho, arquivos salvos com o mesmo nome, mas com caixa diferente. Isto pode também acontecer quando os arquivos forem submetidos de um sistema de arquivos que trata a caixa do nome dos arquivos, como Linux.

Nestes casos, você precisa decidir qual dos arquivos você quer manter e então apagar (ou renomear) o outro arquivo no repositório.

Prevenindo dois arquivos com o mesmo nome

There is a server hook script available at: https://svn.apache.org/repos/asf/subversion/trunk/contrib/hook-scripts/ that will prevent checkins which result in case conflicts.

4.15.4. Reparando Renomeação de Arquivos

Algumas vezes seu amigável IDE vai renomear arquivos para você como parte de uma refatoração, e é claro isto não é o Subversion. Se você tentar submeter suas alterações, Subversion verá que há velhos arquivos perdidos e que há novos arquivos não controlados. Você poderá apenas verificar o novo nome para adicioná-lo, mas você então perderá o histórico, já que o Subversion não conhece a relação dos arquivos.

Uma maneira melhor de notificar o Subversion é identificar a alteração como uma renomeação, e você pode fazer isso usando as janelas de Submeter e Verificar alterações. Apenas selecione ambos os arquivos (perdido e não controlado) e use Menu de Contexto → Reparar movimento para identificar os dois arquivos como uma renomeação.

4.15.5. Apagando Arquivos não Controlados

Normalmente você define sua lista de arquivos ignorados para que todos os arquivos criados sejam ignorados pelo Subversion. Mas e se você quer remover todos os arquivos ignorados para produzir uma compilação limpa? Normalmente você definie isso no arquivo makefile, mas e se você está depurando o arquivo makefile, ou alterando o sistema de compilação é útil ter uma forma de limpar tudo.

TortoiseSVN provê justamente uma opção usando Menu de Contexto Estendido → Apagar arquivos não controlados.... VocShift enquando clica o bot

Quando tais arquivos são removidos, a lixeira é usada, então se você cometeu algum erro ao excluir um arquivo que não estava sendo controlado, você ainda poderá restaurar o arquivo.

4.16. Desfazendo Alterações

Se você quer desfazer todas as alterações feitas em um arquivo desde a última atualização você precisa selecionar o arquivo, clique com o botão direito para abrir o menu de contexto e então selecione a opção TortoiseSVN → Reverter. Uma janela será apresentada com os arquivos que você alterou e que podem ser revertidos. Selecione os arquivos que você reverter e clique em Ok.

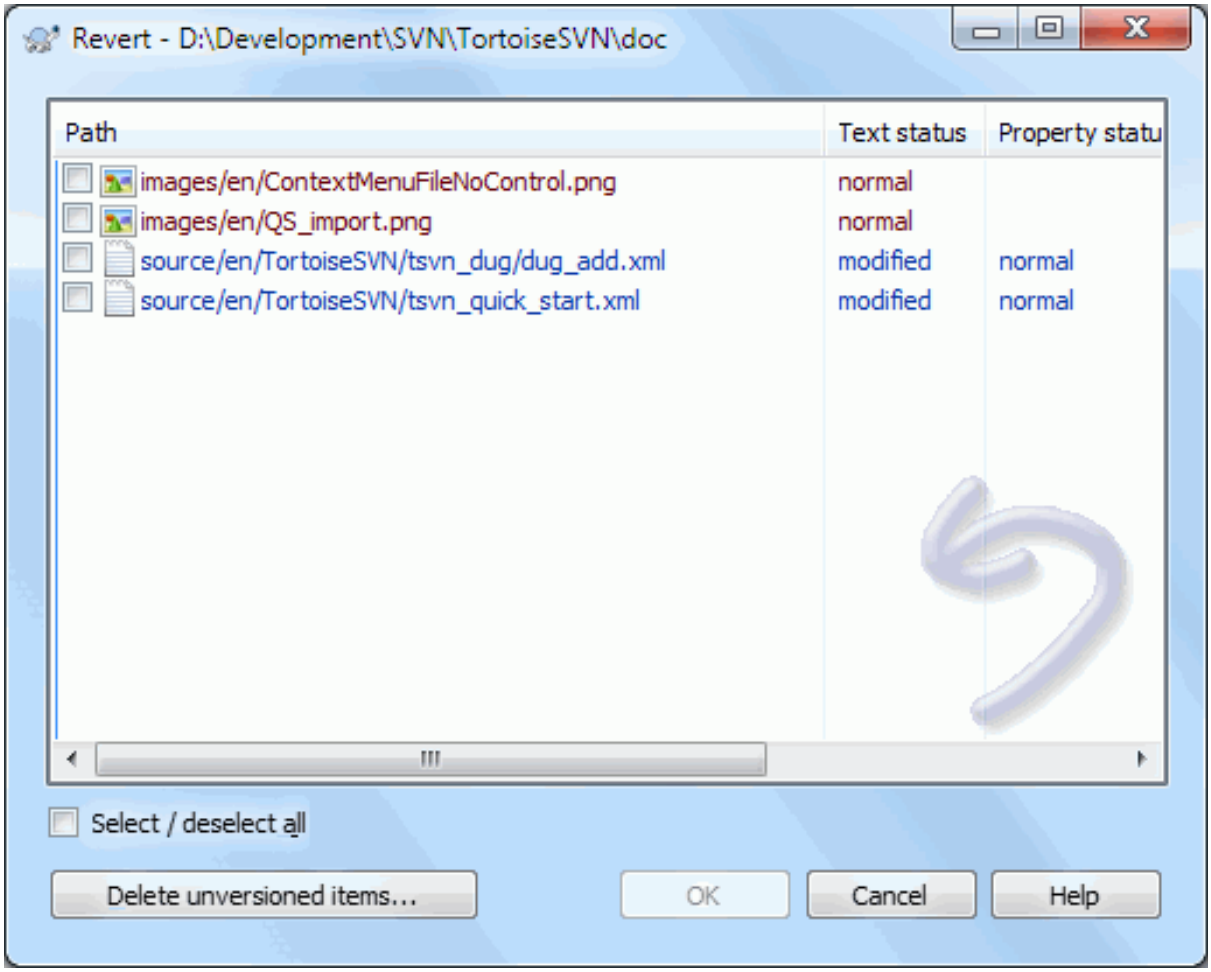


Figura 4.34. Janela de Reversão

If you also want to clear all the changelists that are set, check the box at the bottom of the dialog.

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.

Se você quer desfazer a adição de um item, ele aparece no menu de contexto como TortoiseSVN → Desfazer adição.... Esta op

As colunas nesta janela podem ser personalizadas da mesma forma como as colunas da janela Verificar alterações. Leia Seção 4.7.3, “Estado Local e Remoto” para mais detalhes.

Já que algumas vezes a opção reverter é usada para limpar uma cópia de trabalho, existe uma botão extra o qual também permite a você excluir arquivos não controlados. Quando você clicar neste botão outra janela será mostrada listando todos os arquivos não controlados, os quais você poderá selecionar para exclusão.

Desfazendo Alterações que já foram Submetidas

Reverter irá somente desfazer alterações locais. A opção reverter *não* desfará alterações que já foram submetidas. Se você quer desfazer todas as alterações que foram submetidas em uma revisão em particular, leia Seção 4.10, “Janela de Revisão de Registro” para informações adicionais.

Reversão está Lenta

Quando você reverter alteracões você pode achar que a operação levou mais tempo que o esperado. Isto acontece porque a versão da modificação do arquivo foi enviado para a lixeira, então você pode recuperar suas alterações se você reverter algo por engano. Entretanto, se sua lixeira estiver cheia, Windows vai levar um longo tempo para encontrar um lugar para colocar o seu arquivo. A solução é simples: mantenha a lixeira vazia ou desative a opção Usar lixeira para fazer reversão nas configurações do TortoiseSVN.

4.17. Limpar

If a Subversion command cannot complete successfully, perhaps due to server problems, your working copy can be left in an inconsistent state. In that case you need to use TortoiseSVN → Cleanup on the folder. It is a good idea to do this at the top level of the working copy.

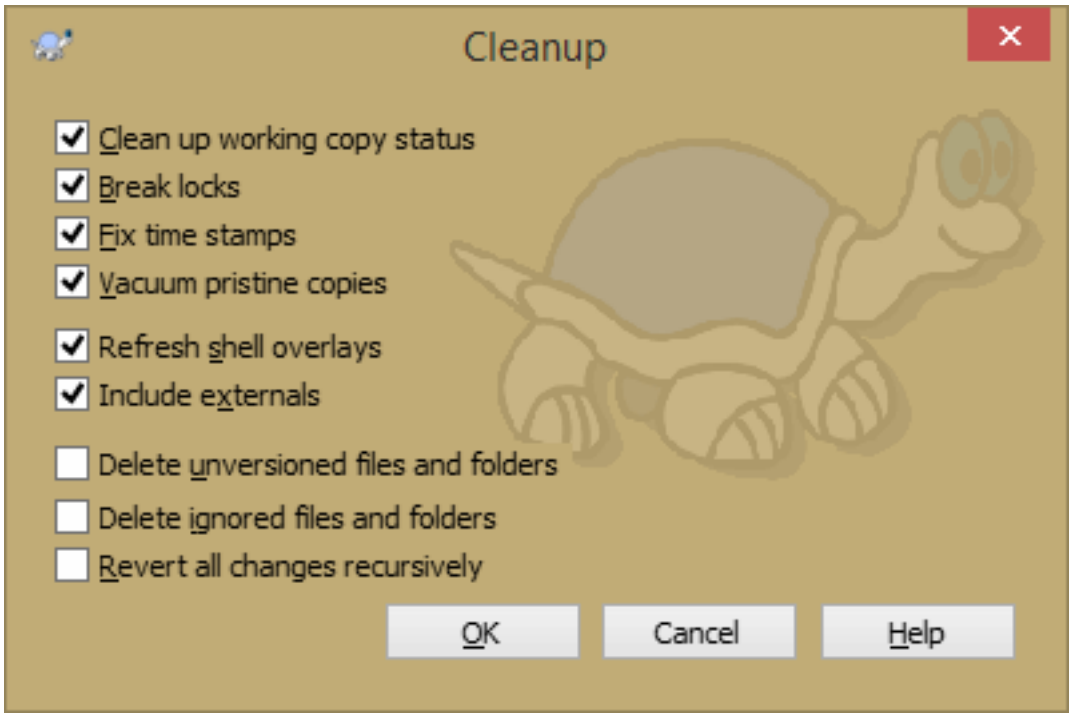


Figura 4.35. The Cleanup dialog

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

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

Break write locks
: If checked, all write locks are removed from the working copy database. For most situations, this is required for the cleanup to work!

  Only uncheck this option if the working copy is used by other users/clients at the time. But if the cleanup then fails, you have to check this option for the cleanup to succeed.

Fix time stamps
: Adjusts the recorded time stamps of all files, speeding up future status checks. This can speed up all dialogs that show working copy file lists, for example the Commit dialog.

Vacuum pristine copies
: Removes unused pristine copies and compresses all remaining pristine copies of working copy files.

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

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

Apagar arquivos e diretórios não versionados, Apagar arquivos e diretórios ignorados
: This is a fast and easy way to remove all generated files in your working copy. All files and folders that are not versioned are moved to the trash bin.

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

Reverter todas as modificações recursivamente
: Este comando reverte todas as suas modificações que não foram comitadas ainda.

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

4.18. Configurações do Projeto

4.18.1. Propriedades do Subversion

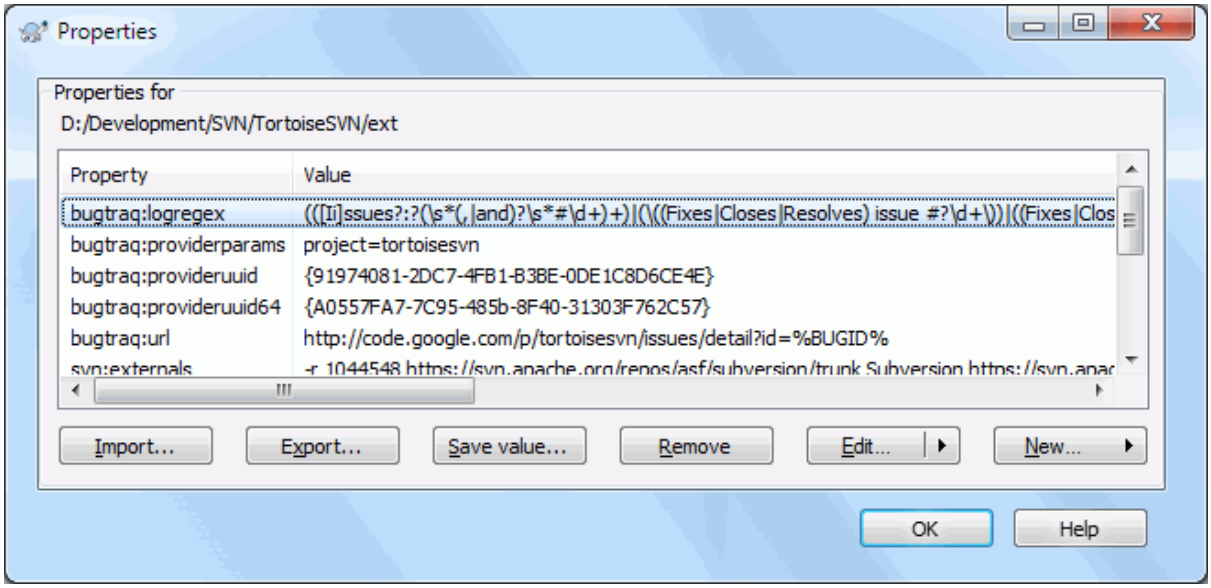


Figura 4.36. página de propriedades do Subversion

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 Seção 4.19, “Itens Externos”.

4.18.1.1. svn:keywords

Subversion supports CVS-like keyword expansion which can be used to embed filename and revision information within the file itself. Keywords currently supported are:

$Date$
: Date of last known commit. This is based on information obtained when you update your working copy. It does *not* check the repository to find more recent changes.

$Revision$
: Revisão da última submissão conhecida

$Author$
: Autor que fez a última submissão conhecida.

$HeadURL$
: A URL completa deste arquivo no repositório

Id
: A combinação comprimida das quatro palavras-chave anteriores.

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

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

4.18.1.2. Adicionando e Editando Propriedades

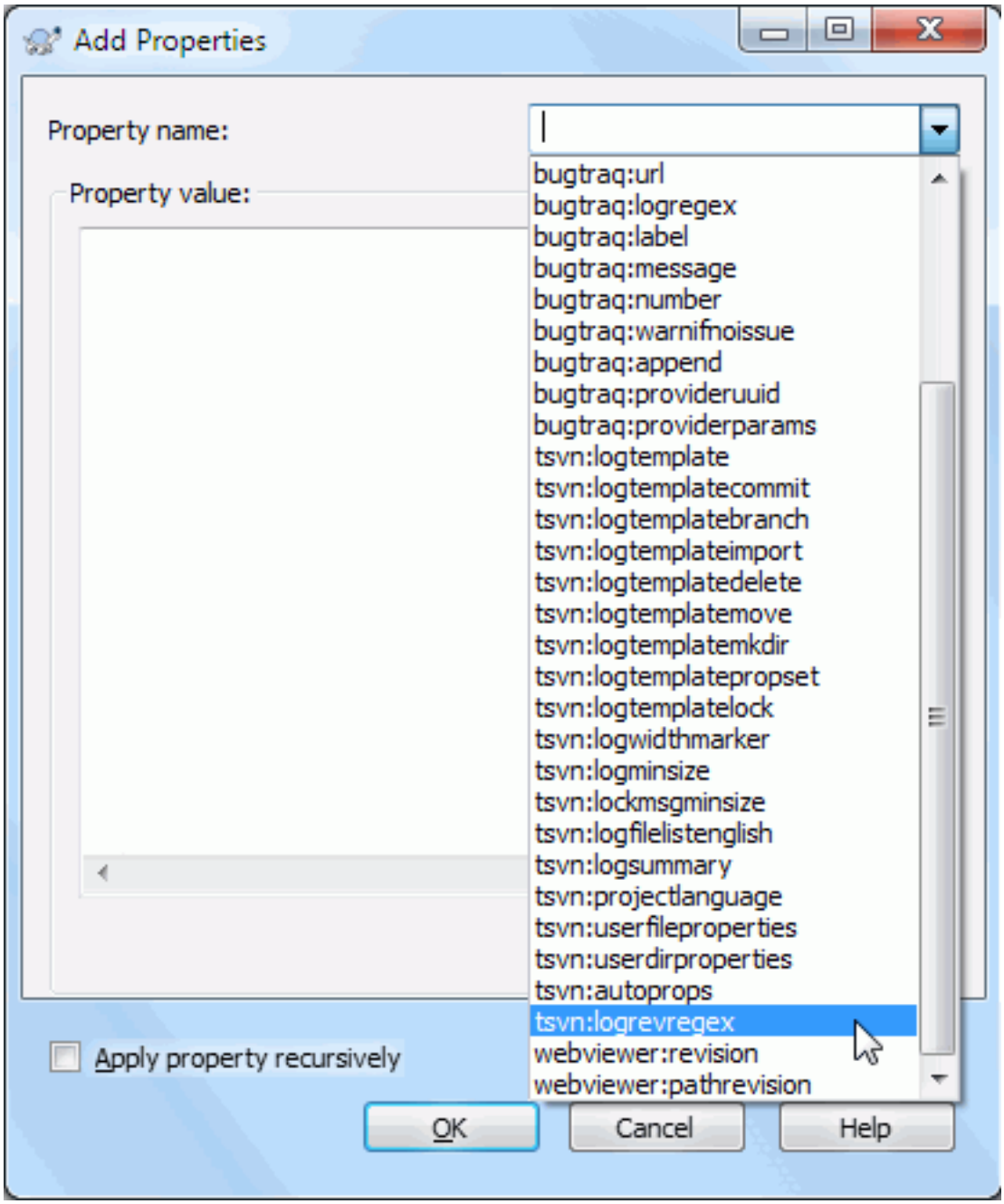


Figura 4.37. Adicionando propriedades

To add a new property, first click on New.... Select the required property name from the menu, and then fill in the required information in the specific property dialog. These specific property dialogs are described in more detail in Seção 4.18.3, “Editores de Propriedades”.

To add a property that doesn't have its own dialog, choose Advanced from the New... menu. Then either select an existing property in the combo box or enter a custom property name.

If you want to apply a property to many items at once, select the files/folders in explorer, then select Context menu → properties.

Se deseja aplicar a propriedade para *todos* os arquivos e diretórios na hierarquia abaixo do diretório atual, marque a opção Recursivo .

Se deseja editar uma propriedade existente, selecione a propriedade da lista de propriedades existentes e então clique em Editar....

Se deseja editar uma propriedade existente, selecione a propriedade da lista de propriedades existentes e então clique em Remover.

A propriedade `svn:externals` pode ser usada para enviar outros projetos para o mesmo repositório ou para um repositório completamente diferente. Para maiores informações, leia Seção 4.19, “Itens Externos”.

Editar propriedades na revisão atual

Pelo fato das propriedades serem versionadas, não é possível editar as revisões anteriores. Se verificar as propriedades pela caixa de dialogo do log, ou por uma revisão no navegador do repositório que não seja a ultima, poderá visualizar a lista de propriedades e valores, porem não terá nenhum controle de edição disponível.

4.18.1.3. Exportando e Importando Propriedades

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.

4.18.1.4. Propriedades Binárias

TortoiseSVN can handle binary property values using files. To read a binary property value, Save... to a file. To set a binary value, use a hex editor or other appropriate tool to create a file with the content you require, then Load... from that file.

Although binary properties are not often used, they can be useful in some applications. For example if you are storing huge graphics files, or if the application used to load the file is huge, you might want to store a thumbnail as a property so you can obtain a preview quickly.

4.18.1.5. Definição automática de propriedade

You can configure Subversion and TortoiseSVN to set properties automatically on files and folders when they are added to the repository. There are two ways of doing this.

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

An alternative method is to set the `tsvn:autoprops` property on folders, as described in the next section. This method only works for TortoiseSVN clients, but it does get propagated to all working copies on update.

As of Subversion 1.8, you can also set the property `svn:auto-props` on the root folder. The property value is automatically inherited by all child items.

Whichever method you choose, you should note that auto-props are only applied to files at the time they are added to the working copy. Auto-props will never change the properties of files which are already versioned.

If you want to be absolutely sure that new files have the correct properties applied, you should set up a repository pre-commit hook to reject commits where the required properties are not set.

Submeter propriedades

Propriedades do Subversion são versionadas. Depois que você mudar ou adicionar uma propriedade você precisa submeter suas alterações.

Conflitos em propriedades

Se existe um conflito ao submeter as alterações, porque outro usuário alterou a mesma propriedade, o Subversion gerará um arquivo `.prej`. Apague este arquivo depois que você resolver o conflito.

4.18.2. Propriedades do Projeto TortoiseSVN

TortoiseSVN tem algumas propriedades especiais próprias, que começam com `tsvn:`.

- `tsvn:logminsize` sets the minimum length of a log message for a commit. If you enter a shorter message than specified here, the commit is disabled. This feature is very useful for reminding you to supply a proper descriptive message for every commit. If this property is not set, or the value is zero, empty log messages are allowed.

  `tsvn:lockmsgminsize` sets the minimum length of a lock message. If you enter a shorter message than specified here, the lock is disabled. This feature is very useful for reminding you to supply a proper descriptive message for every lock you get. If this property is not set, or the value is zero, empty lock messages are allowed.

- `tsvn:logwidthmarker` is used with projects which require log messages to be formatted with some maximum width (typically 80 characters) before a line break. Setting this property to a non-zero will do 2 things in the log message entry dialog: it places a marker to indicate the maximum width, and it disables word wrap in the display, so that you can see whether the text you entered is too long. Note: this feature will only work correctly if you have a fixed-width font selected for log messages.

- `tsvn:logtemplate` is used with projects which have rules about log message formatting. The property holds a multi-line text string which will be inserted in the commit message box when you start a commit. You can then edit it to include the required information. Note: if you are also using `tsvn:logminsize`, be sure to set the length longer than the template or you will lose the protection mechanism.

  There are also action specific templates which you can use instead of `tsvn:logtemplate`. The action specific templates are used if set, but `tsvn:logtemplate` will be used if no action specific template is set.

  Os modelos específicos de ação são:

  - `tsvn:logtemplatecommit` é usado para para todas as submissões da cópia de trabalho.
  - `tsvn:logtemplatebranch` é usado quando você cria um ramo/marca, ou quando você copia arquivos ou diretórios dentro do navegador de repositório.
  - `tsvn:logtemplateimport` é usado para importação.
  - `tsvn:logtemplatedelete` é usado quando itens são apagados dentro do navegador de repositório.
  - `tsvn:logtemplatemove` é usado quando arquivos são renomeados ou movidos dentro do navegador de repositório.

  - `tsvn:logtemplatemkdir` é usado quando diretórios são criados dentro do navegador de repositório.
  - `tsvn:logtemplatepropset` é usado quando propriedades são modificadas dentro do navegador de repositório.
  - `tsvn:logtemplatelock` é usado quando é obtido um bloqueio.

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

  Se existe um conflito para a propriedade `tsvn:autoprops` entre a versão remota e local, as configurações do projeto são precedentes porque elas são específicas do projeto.

  As of Subversion 1.8, you should use the property `svn:auto-props` instead of `tsvn:autoprops` since this has the very same functionality but works with all svn clients and is not specific to TortoiseSVN.

- Na janela de Submissão você tem a opção de colar a lista de arquivos modificados, incluindo a situação de cada arquivo (adicionado, modificado, etc). `tsvn:logfilelistenglish` define se a situação do arquivo é inserida em Inglês ou no idioma configurado. Se a propriedade não está configurada, o padrão é `true`.

- TortoiseSVN can use a spell checker. On Windows 10, the spell checker of the OS is used. On earlier Windows versions, it 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* [https://docs.microsoft.com/en-us/windows/desktop/intl/language-identifier-constants-and-strings].

  Você pode tanto digitar o valor em decimal, ou em hexadecimal se iniciado com `0x`. Por exemplo Inglês (US) pode ser informado como `0x0409` ou `1033`.

- A propriedade `tsvn:logsummary` é usada para extrair uma porção da mensagem de auditoria que é mostrada na janela de auditoria como o resumo da mensagem.

  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.

  Por exemplo: `\[RESUMO\]:\s+(.*)` Vai pegar tudo depois de “[RESUMO]” na mensagem de auditoria e usar como resumo.

- The property `tsvn:logrevregex` defines a regular expression which matches references to revisions in a log message. This is used in the log dialog to turn such references into links which when clicked will either scroll to that revision (if the revision is already shown in the log dialog, or if it's available from the log cache) or open a new log dialog showing that revision.

  A expressão regular deve ser bater com a referência completa, não apenas com o número da revisão. O número da revisão é extraído do texto de referência compatível automaticamente.

  Se esta propriedade não estiver definida, uma expressão regular padrão é usada para relacionar as referências para as revisões.

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

  The available properties/hook-scripts are

  - tsvn:startcommithook

- tsvn:precommithook
- tsvn:postcommithook
- tsvn:startupdatehook
- tsvn:preupdatehook
- tsvn:postupdatehook
- tsvn:prelockhook
- tsvn:postlockhook

The parameters are the same as if you would configure the hook scripts in the settings dialog. See Seção 4.31.8, “Client Side Hook Scripts” for the details.

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

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

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

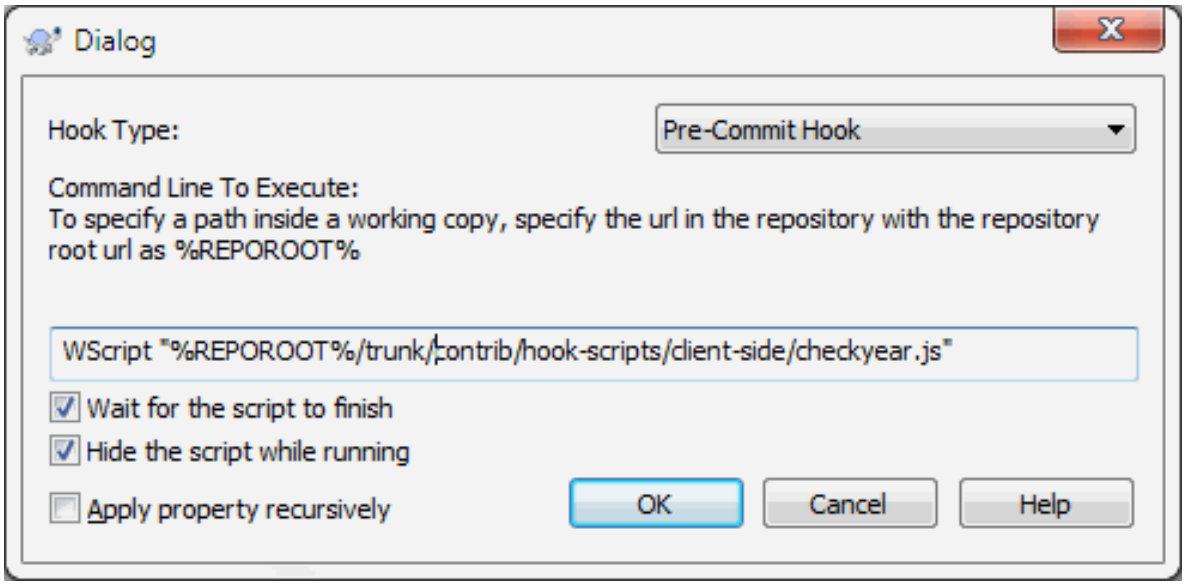


Figura 4.38. Property dialog for hook scripts

- When you want to add a new property, you can either pick one from the list in the combo box, or you can enter any property name you like. If your project uses some custom properties, and you want those properties to appear in the list in the combo box (to avoid typos when you enter a property name), you can create a list of

your custom properties using `tsvn:userfileproperties` and `tsvn:userdirproperties`. Apply these properties to a folder. When you go to edit the properties of any child item, your custom properties will appear in the list of pre-defined property names.

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

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

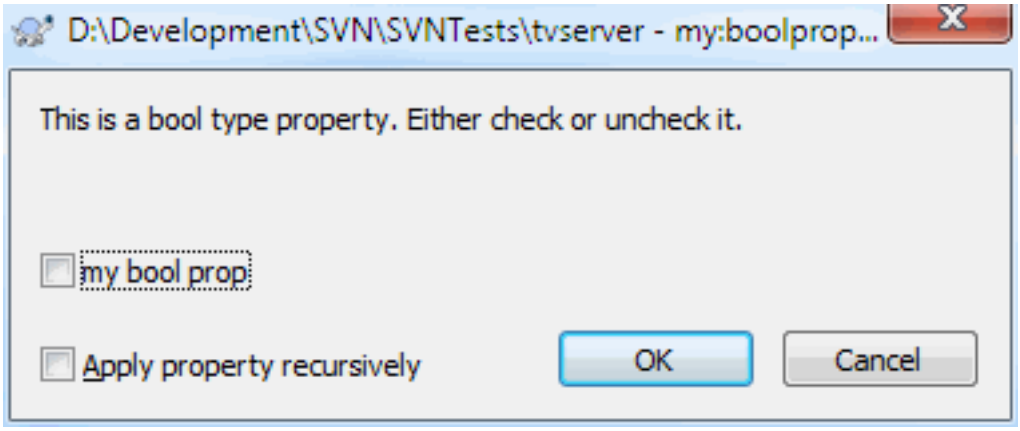


Figura 4.39. Property dialog boolean user types

Specify your property like this:

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

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

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

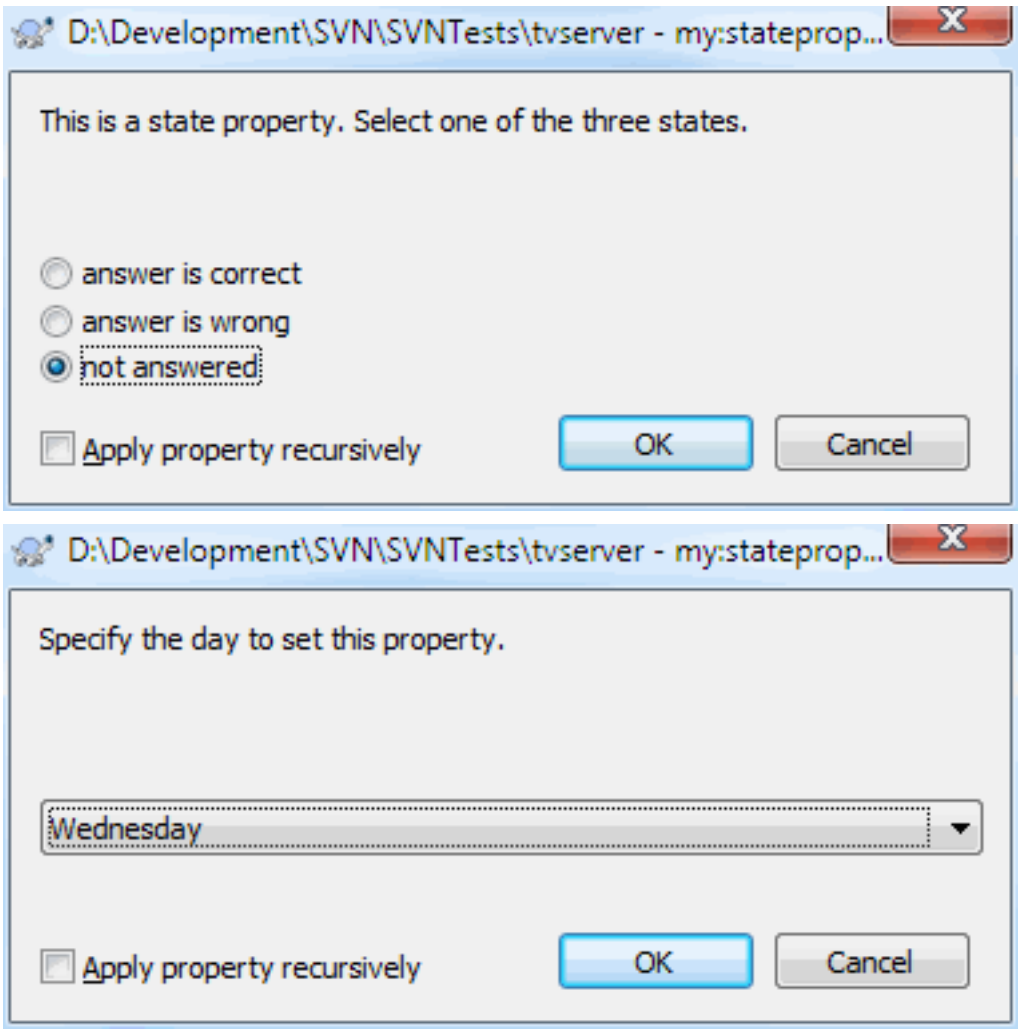


Figura 4.40. Property dialog state user types

property like this:

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

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

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

singleline

Para propriedades que consistem de uma linha de texto, use a o tipo de propriedade `linha simples`:/>nomepropriedade=linhasimples;identiftexto(exreg)the regex specifies a regular expression which is used to validate (match) the text the user entered. If the text does not match the regex, then the user is shown an error and the property isn't set.multilineFor properties that consist of multiple lines of text, use the multiline property type: Property dialog multi-line user typesnomepropriedade=multilinha;identiftexto(exreg)the regex specifies a regular expression which is used to validate (match) the text the user entered. Don't forget to include the newline (\n) character in the regex! The screenshots above were made with the following tsvn:userdirproperties: my:boolprop=bool;This is a bool type property. Either check or uncheck it.(true;false;my bool prop) my:stateprop1=state;This is a state property. Select one of the two states.(true;true;true value;false;false value) my:stateprop2=state;This is a state property. Select one of the three states.(maybe;true;answer is correct;false;answer is wrong;maybe;not answered) my:stateprop3=state;Specify the day

to set this property.(1;1;Monday;2;Tuesday;3;Wednesday;4;Thursday;5;Friday;6;Saturday;7;Sunday) my:singlelineprop=singleline;enter a small comment(.*) my:multilineprop=multiline;copy and paste a full chapter here(.*)

TortoiseSVN can integrate with some bug tracking tools. This uses project properties that start with `bugtraq:`. Read Seção 4.29, “Integração com Sistemas de Rastreamento de Erros / Rastreadores de Reportes” for further information.

It can also integrate with some web-based repository browsers, using project properties that start with `webviewer:`. Read Seção 4.30, “Integração com visualizadores de repositórios locados na web” for further information.

Define as propriedades do projeto em diretórios

These special project properties must be set on *folders* for the system to work. When you use a TortoiseSVN command which uses these properties, the properties are read from the folder you clicked on. 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 (e.g. `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. If you set the same property but you use different values at different depths in your project hierarchy then you will get different results depending on where you click in the folder structure.

For project properties *only*, i.e. `tsvn:`, `bugtraq:` and `webviewer:` 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 to a working copy using TortoiseSVN, any project properties present in the parent folder will automatically be added to the new child folder too.

Limitações Usando o Navegador de Repositório

Fetching properties remotely is a slow operation, so some of the features described above will not work in the repository browser as they do in a working copy.

- When you add a property using the repo browser, only the standard `svn:` properties are offered in the pre-defined list. Any other property name must be entered manually.
- Propriedades não podem ser definidas ou apagadas recursivamente usando o navegador de repositório.
- As propriedade do projeto *não* serão propagadas automaticamente quando um sub diretório é adicionado pelo navegador de repositório.
- `tsvn:autoprops` *não* definirá as propriedades em arquivos que são adicionados pelo navegador de repositório.

Cuidado

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.

4.18.3. Editores de Propriedades

Some properties have to use specific values, or be formatted in a specific way in order to be used for automation. To help get the formatting correct, TortoiseSVN presents edit dialogs for some particular properties which show the possible values or break the property into its individual components.

4.18.3.1. Conteúdo Externo

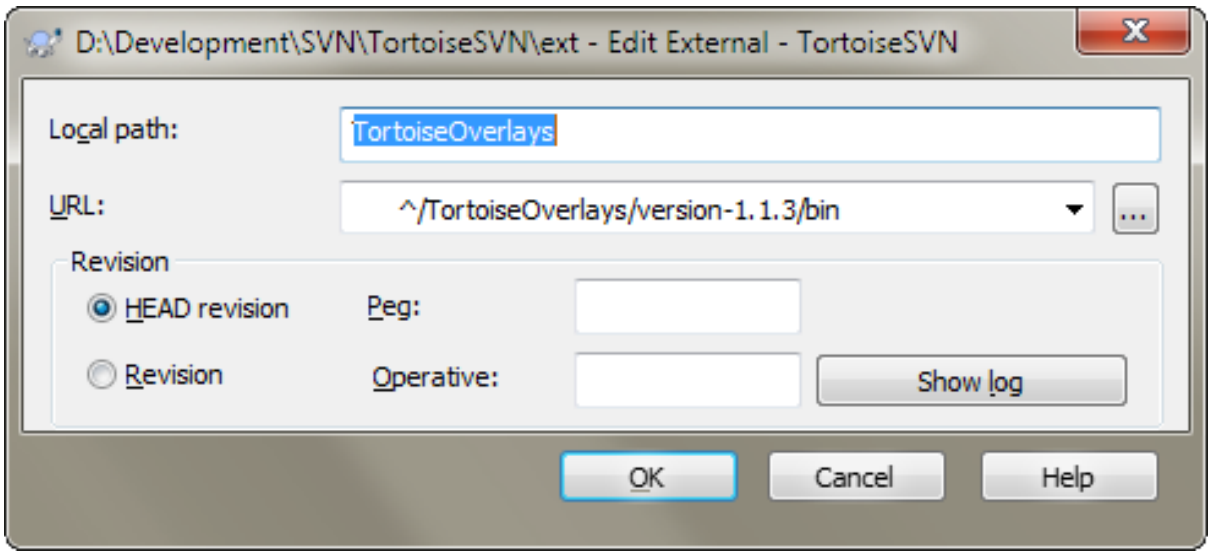


Figura 4.41. página da propriedade svn:externals

The `svn:externals` property can be used to pull in other projects from the same repository or a completely different repository as described in Seção 4.19, “Itens Externos”.

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

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

4.18.3.2. Palavras-chave do SVN

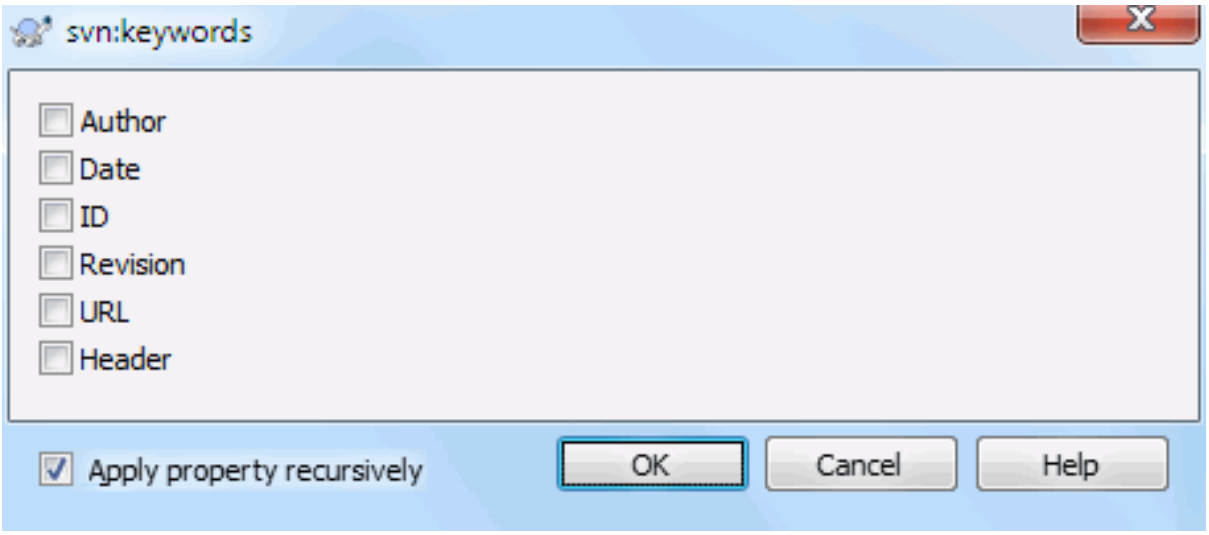


Figura 4.42. página da propriedade svn:keywords

Seleciona as palavras-chave que você gostaria que fosse expandida em seu arquivo.

4.18.3.3. Estilo FDL

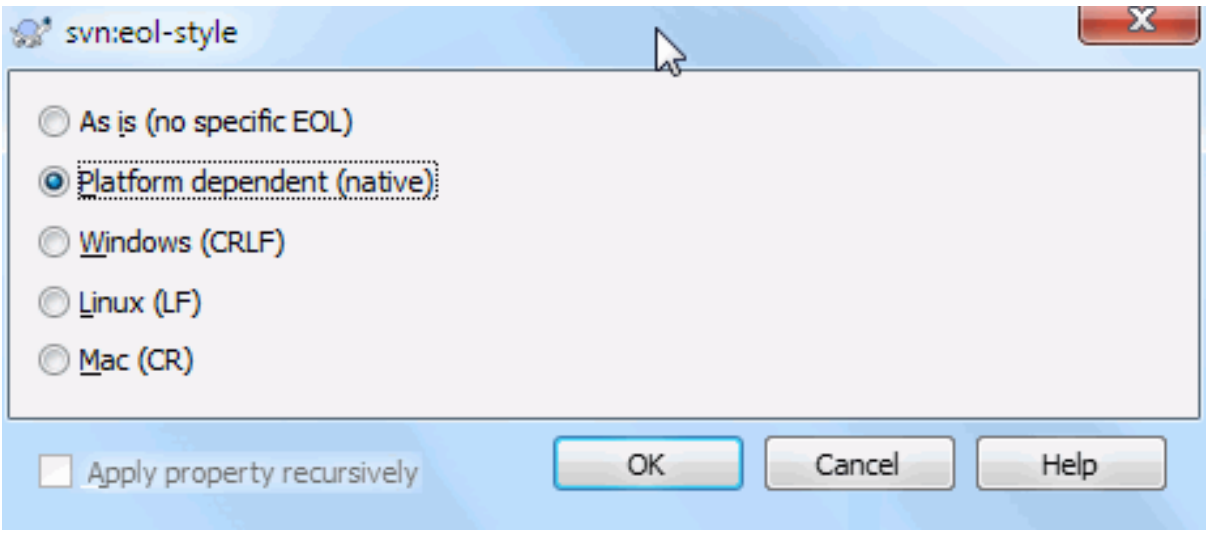


Figura 4.43. página da propriedade svn:eol-style

Seleciona o estilo de fim de linha que você deseja usar e o TortoiseSVN usará o valor da propriedade correto.

4.18.3.4. Integração para Rastreador de Reportes

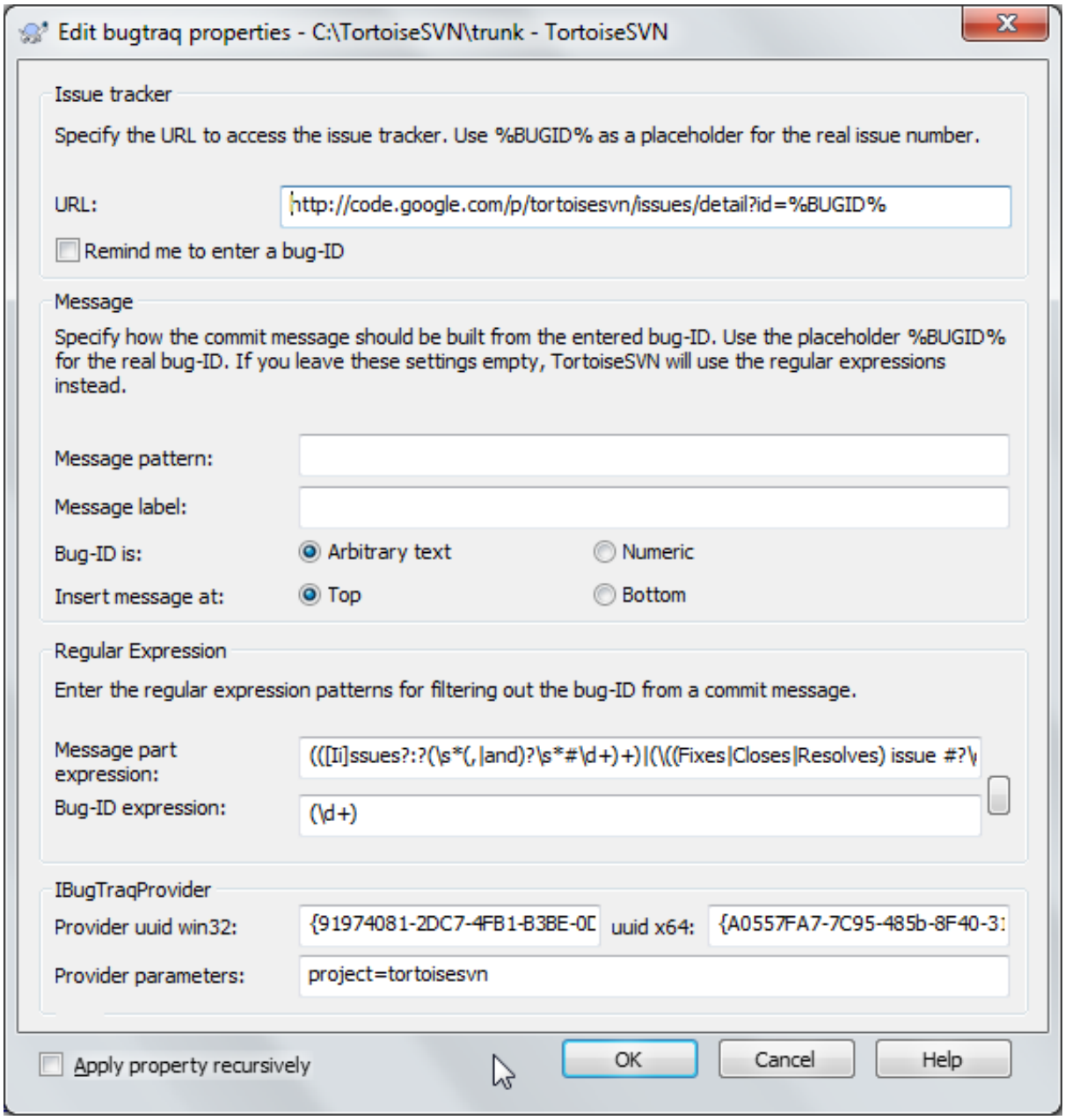


Figura 4.44. página da propriedade tsvn:bugtraq

4.18.3.5. Tamanhos das Mensagens de Auditoria

Figura 4.45. Página da propriedade do tamanho das mensagens de auditoria

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

4.18.3.6. Idioma do Projeto

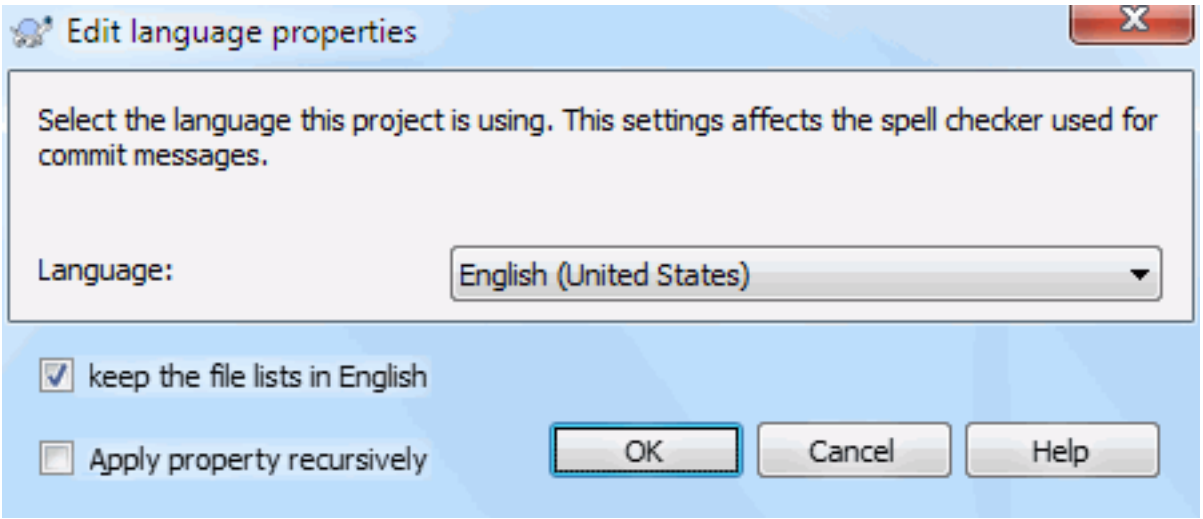


Figura 4.46. Página da propriedade idioma

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

4.18.3.7. MIME-type

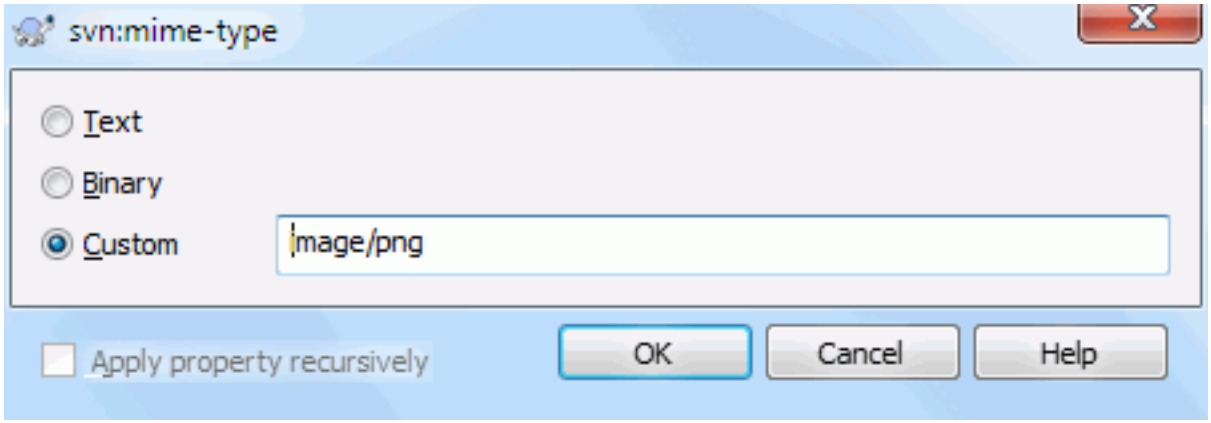


Figura 4.47. página da propriedade svn:mime-type

4.18.3.8. svn:needs-lock

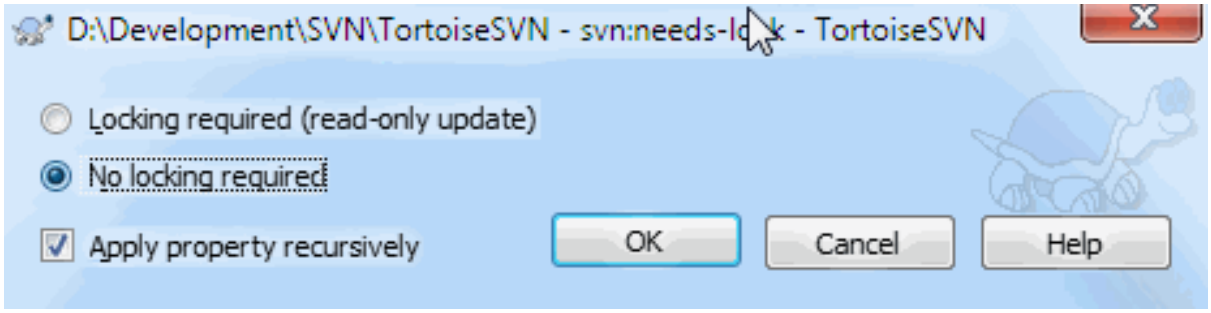


Figura 4.48. página da propriedade svn:needs-lock

This property simply controls whether a file will be checked out as read-only if there is no lock held for it in the working copy.

4.18.3.9. svn:executable

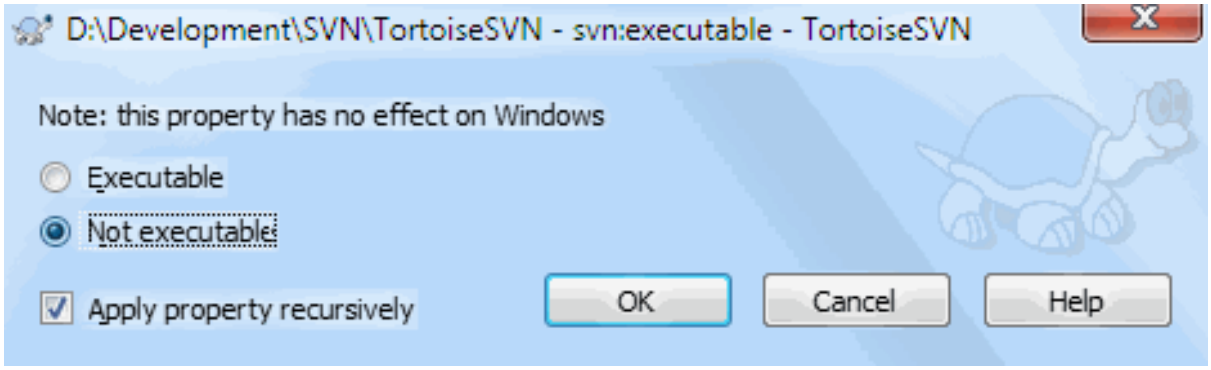


Figura 4.49. página da propriedade svn:executable

This property controls whether a file will be given executable status when checked out on a Unix/Linux system. It has no effect on a Windows checkout.

4.18.3.10. Merge log message templates

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

Você pode customizar esta mensagem gerada com as seguintes propriedades:

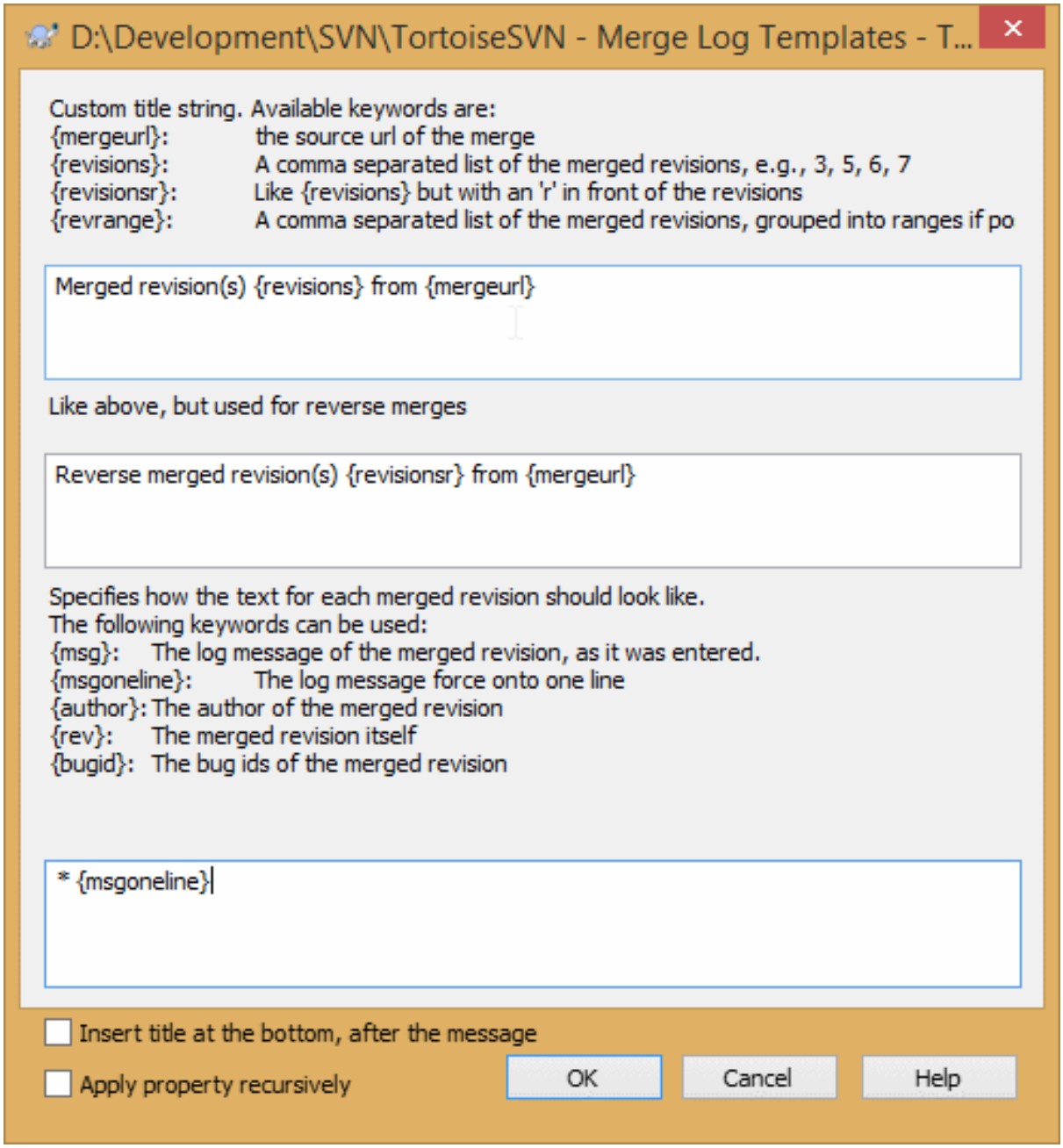


Figura 4.50. Property dialog merge log message templates

tsvn:mergelogtemplatetitle, tsvn:mergelogtemplatereversetitle
: Esta propriedade especifica a primeira parte da mensagem de auditoria gerada. As seguintes palavras-chave podem ser usadas:

  {revisions}
  : Uma lista de revisões combinadas separadas por vírgula, p.ex., `3, 5, 6, 7`

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

  {revrange}
  : Uma lista de revisões combinadas separadas por vírgula, se possível agrupadas em intervalos, p.ex., `3, 5-7`

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

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

tsvn:mergelogtemplatemsg
Esta propriedade especifica como o texto para cada revisão combinada deve se parecer. As seguintes palavras-chaves podem ser usadas:

{msg}
A mensagem de auditoria da revisão combinada, como ela foi entrada.

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

{author}
O autor da revisão mesclada

{rev}
A própria revisão combinada.

{bugids}
Os IDs dos erros da revisão combinada, se existirem.

tsvn:mergelogtemplatemsgtitlebottom
This property specifies the position of the title string specified with the `tsvn:mergelogtemplatetitle` or `tsvn:mergelogtemplatereversetitle`. If the property is set to `yes` or `true`, then the title string is appended at the bottom instead of the top.

Importante

Isto só funciona se as revisões combinadas já estiverem no cache de auditoria. Se você tiver desabilitado o cache de auditoria ou não tiver mostrado a auditoria logo antes da combinação, a mensagem gerada não conterá quaisquer informações sobre as revisões combinadas.

4.19. Itens Externos

Por vezes é útil construir uma cópia de trabalho feita a partir de diferentes checkouts. Por exemplo, você poderá querer diferentes arquivos ou sub-diretórios de diferentes locais de um repositório ou talvez, de diferentes repositórios. Se você quiser que todo usuário tenha o mesmo layout, você pode definir as propriedades `svn:externals` para incluir o recurso especificado nos locais onde são necessários.

4.19.1. Diretórios Externos

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 Properties.... In the properties dialog, either double click on the `svn:externals` if it already exists, or click on the New... button and select `externals` from the menu. To add a new external, click the New... and then fill in the required information in the shown dialog.

Cuidado

URLs must be properly escaped or they will not work, e.g. you must replace each space with `%20`.

Se você quiser que o caminho local inclua "espaços" ou outro caracter especial, terá que incluí-lo entre aspas ou usar o caracter `\` (barra invertida) como um caracter de escape, ao estilo da linha de comandos do Linux, precedendo cada caracter especial. É claro que isto também significa que terá que necessariamente usar `/` (barra inclinada) como delimitador de caminho. Note que este comportamento é inusitado no Subversion 1.6 e não funcionará em versões mais antigas.

Usar números explícitos de revisão

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 the 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.

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

Se o projeto externo está no mesmo repositório, quaisquer alterações feitas serão incluídas na lista de submissão quando submeter seu projeto principal.

If the external project is in a different repository, any changes you make to the external project will be shown or indicated when you commit the main project, but you have to commit those external changes separately.

Se usar URLs absolutas nas definições `svn:externals` e tiver que realocar sua cópia de trabalho (ou seja, a URL do seu repositório é alterada), então seus "externos" não serão alterados e poderão deixar de funcionar.

Para evitar tais problemas, a versão 1.5 ou superior do cliente Subversion, suporta URLs externas relativas. Quatro diferentes métodos de especificar uma URL relativa são suportados. Nos exemplos seguintes assume-se que temos dois repositórios: um em `http://example.com/svn/repos-1` e outro em `http://example.com/svn/repos-2`. Temos um SVN exportado do `http://example.com/svn/repos-1/project/trunk` em `C:\Working` e a propriedade `svn:externals` está configurada no trunk.

Relativo ao diretório pai
: These URLs always begin with the string `../` for example:

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

This will extract `http://example.com/svn/repos-1/widgets/foo` into `C:\Working\common\foo-widget`.

Note que a URL é relativa a URL do diretório com a propriedade `svn:externals` e não a URL do diretório onde o arquivo externo está armazenado.

Relativo ao diretório principal do repositório
: These URLs always begin with the string `^/` for example:

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

This will extract `http://example.com/svn/repos-1/widgets/foo` into `C:\Working\common\foo-widget`.

You can easily refer to other repositories with the same `SVNParentPath` (a common directory holding several repositories). For example:

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

This will extract `http://example.com/svn/repos-2/hammers/claw` into `C:\Working\common\claw-hammer`.

Relativo ao esquema
: URLs beginning with the string `//` copy only the scheme part of the URL. This is useful when the same hostname must the accessed with different schemes depending upon network location; e.g. clients in the intranet use `http://` while external clients use `svn+ssh://`. For example:

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

This will extract `http://example.com/svn/repos-1/widgets/foo` or `svn+ssh://example.com/svn/repos-1/widgets/foo` depending on which method was used to checkout `C:\Working`.

Relativo ao nome do servidor
: URLs beginning with the string `/` copy the scheme and the hostname part of the URL, for example:

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

This will extract `http://example.com/svn/repos-1/widgets/foo` into `C:\Working\common\foo-widget`. But if you checkout your working copy from another server at `svn+ssh://another.mirror.net/svn/repos-1/project1/trunk` then the external reference will extract `svn+ssh://another.mirror.net/svn/repos-1/widgets/foo`.

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

Importante

If you specify the target folder for the external as a subfolder like in the examples above, make sure that *all* folders in between are versioned as well. So for the examples above, the folder `common` should be versioned!

While the external will work in most situations properly if folders in between are not versioned, there are some operations that won't work as you expect. And the status overlay icons in explorer will also not show the correct status.

Caso necessite mais informações sobre como TortoiseSVN lida com "Propriedades", consulte Seção 4.18, “Configurações do Projeto”.

Para saber mais sobre diferentes métodos de acesso a subprojetos comuns, consulte Seção B.6, “Inclua um subprojeto em comum”.

4.19.2. Arquivos Externos

Desde a versão 1.6 do Subversion você pode incluir um único arquivo externo em sua cópia de trabalho usando a mesma sintaxe que usaria para diretórios. Entretanto existem algumas restrições.

- The path to the file external must be a direct child of the folder where you set the `svn:externals` property.
- A URL para um arquivo externo deverá estar no mesmo repositório que a URL na qual o arquivo externo será incluído; arquivos externos inter-repositórios não são suportados.

Em vários aspectos um arquivo externo comporta-se como qualquer outro arquivo versionado, mas eles não podem ser movidos ou eliminados usando os comandos normais; é obrigatório que a propriedade `svn:externals` seja modificada.

4.19.3. Criando externos via drag e drop

If you already have a working copy of the files or folders you want to include as externals in another working copy, you can simply add those via drag and drop from the windows explorer.

Simply right drag the file or folder from one working copy to where you want those to be included as externals. A context menu appears when you release the mouse button: SVN Add as externals here if you click on that context menu entry, the `svn:externals` property is automatically added. All you have to do after that is commit the property changes and update to get those externals properly included in your working copy.

4.20. Ramificando / Rotulando

One of the features of version control systems is the ability to isolate changes onto a separate line of development. This line is known as a *branch*. Branches are often used to try out new features without disturbing the main line of development with compiler errors and bugs. As soon as the new feature is stable enough then the development branch is *merged* back into the main branch (trunk).

Another feature of version control systems is the ability to mark particular revisions (e.g. a release version), so you can at any time recreate a certain build or environment. This process is known as *tagging*.

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.

4.20.1. Criando um Ramo ou Rótulo

If you have imported your project with the recommended directory structure, creating a branch or tag version is very simple:

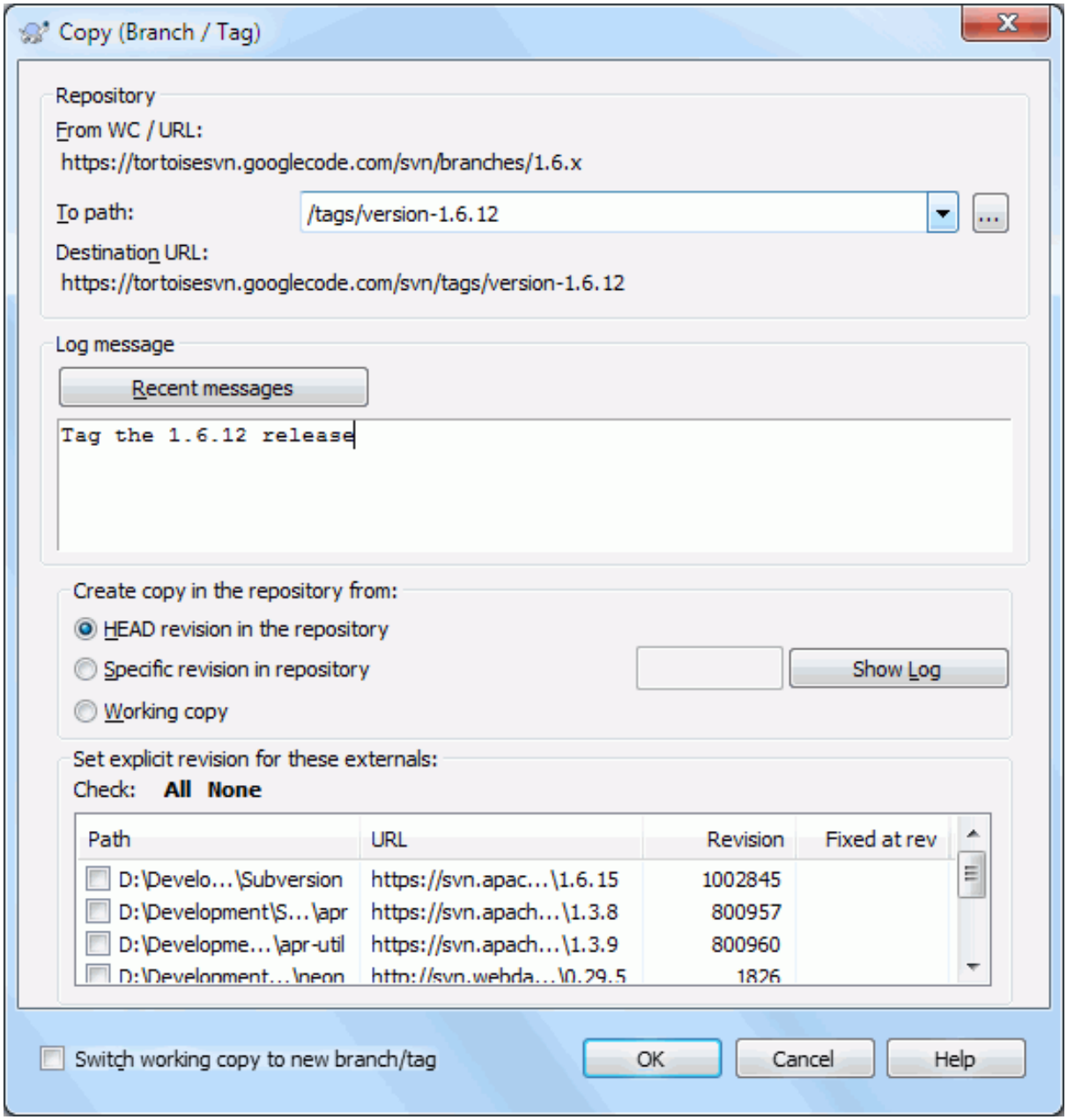


Figura 4.51. A Janela de Ramificação/Rotulação

Select the folder in your working copy which you want to copy to a branch or tag, then select the command TortoiseSVN → Branch/Tag....

The default destination URL for the new branch will be the source URL on which your working copy is based. You will need to edit that URL to the new path for your branch/tag. So instead of

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

you might now use something like

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

If you can't remember the naming convention you used last time, click the button on the right to open the repository browser so you can view the existing repository structure.

Pastas intermediárias

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

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

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

Agora você tem que selecionar a fonte da cópia. Aqui você tem três opções:

Última revisão no repositório
: The new branch is copied directly in the repository from the HEAD revision. No data needs to be transferred from your working copy, and the branch is created very quickly.

Revisão específica no repositório
: The new branch is copied directly in the repository but you can choose an older revision. This is useful if you forgot to make a tag when you released your project last week. If you can't remember the revision number, click the button on the right to show the revision log, and select the revision number from there. Again no data is transferred from your working copy, and the branch is created very quickly.

Cópia de trabalho
: The new branch is an identical copy of your local working copy. If you have updated some files to an older revision in your WC, or if you have made local changes, that is exactly what goes into the copy. Naturally this sort of complex tag may involve transferring data from your WC back to the repository if it does not exist there already.

If you want your working copy to be switched to the newly created branch automatically, use the Switch working copy to new branch/tag checkbox. But if you do that, first make sure that your working copy does not contain modifications. If it does, those changes will be merged into the branch WC when you switch.

If your working copy has other projects included with `svn:externals` properties, those externals will be listed at the bottom of the branch/tag dialog. For each external, the target path and the source URL is shown.

If you want to make sure that the new tag always is in a consistent state, check all the externals to have their revisions pinned. If you don't check the externals and those externals point to a HEAD revision which might change in the future, checking out the new tag will check out that HEAD revision of the external and your tag might not compile anymore. So it's always a good idea to set the externals to an explicit revision when creating a tag.

The externals are automatically pinned to either the current HEAD revision or the working copy BASE revision, depending on the source of the branch/tag:

Copiar Fonte	Revisãio Pinado
Última revisão no repositório	external's repos HEAD revision

Copiar Fonte	Revisão Pinado
especexternal's repos HEAD revisionCexternal's WC BASE revision	

Tabela 4.1. Revisão Pinado

externals within externals

If a project that is included as an external has itself included externals, then those will not be tagged! Only externals that are direct children can be tagged.

Press OK to commit the new copy to the repository. Don't forget to supply a log message. Note that the copy is created *inside the repository*.

Note that unless you opted to switch your working copy to the newly created branch, creating a Branch or Tag does *not* affect your working copy. Even if you create the branch from your WC, those changes are committed to the new branch, not to the trunk, so your WC may still be marked as modified with respect to the trunk.

4.20.2. Outras maneiras de criar um ramo ou destino

You can also create a branch or tag without having a working copy. To do that, open the repository browser. You can there drag folders to a new location. You have to hold down the **Ctrl** key while you drag to create a copy, otherwise the folder gets moved, not copied.

You can also drag a folder with the right mouse button. Once you release the mouse button you can choose from the context menu whether you want the folder to be moved or copied. Of course to create a branch or tag you must copy the folder, not move it.

Yet another way is from the log dialog. You can show the log dialog for e.g. trunk, select a revision (either the HEAD revision at the very top or an earlier revision), right click and choose create branch/tag from revision....

4.20.3. Para Obter ou Alternar

...that is (not really) the question. While a checkout downloads everything from the desired branch in the repository to your working directory, TortoiseSVN → Switch... only transfers the changed data to your working copy. Good for the network load, good for your patience. :-)

To be able to work with your freshly generated branch or tag you have several ways to handle it. You can:

- TortoiseSVN → Checkout to make a fresh checkout in an empty folder. You can check out to any location on your local disk and you can create as many working copies from your repository as you like.
- Switch your current working copy to the newly created copy in the repository. Again select the top level folder of your project and use TortoiseSVN → Switch... from the context menu.

  In the next dialog enter the URL of the branch you just created. Select the Head Revision radio button and click on OK. Your working copy is switched to the new branch/tag.

  Switch works just like Update in that it never discards your local changes. Any changes you have made to your working copy which have not yet been committed will be merged when you do the Switch. If you do not want this to happen then you must either commit the changes before switching, or revert your working copy to an already-committed revision (typically HEAD).
- If you want to work on trunk and branch, but don't want the expense of a fresh checkout, you can use Windows Explorer to make a copy of your trunk checkout in another folder, then TortoiseSVN → Switch... that copy to your new branch.

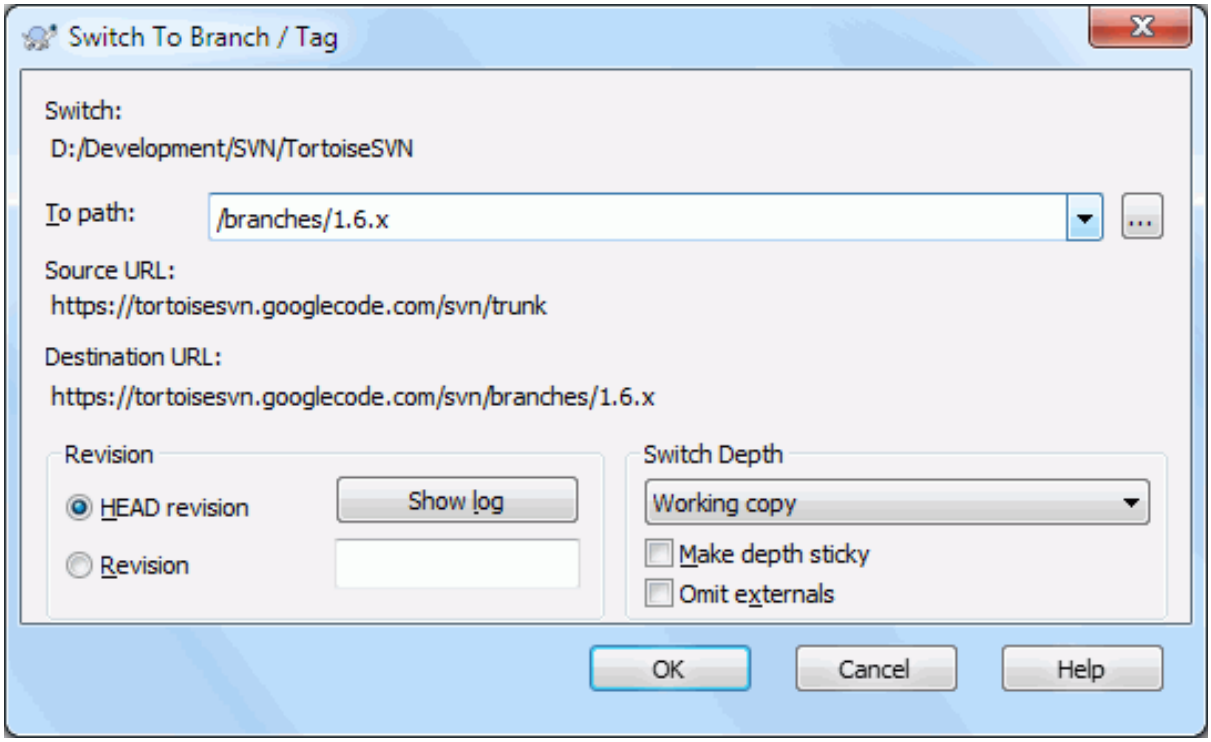


Figura 4.52. A Janela de Troca

Although Subversion itself makes no distinction between tags and branches, the way they are typically used differs a bit.

- Tags are typically used to create a static snapshot of the project at a particular stage. As such they are not normally used for development - that's what branches are for, which is the reason we recommended the `/trunk /branches /tags` repository structure in the first place. Working on a tag revision is *not a good idea*, but because your local files are not write protected there is nothing to stop you doing this by mistake. However, if you try to commit to a path in the repository which contains `/tags/`, TortoiseSVN will warn you.
- It may be that you need to make further changes to a release which you have already tagged. The correct way to handle this is to create a new branch from the tag first and commit the branch. Do your Changes on this branch and then create a new tag from this new branch, e.g. `Version_1.0.1`.
- If you modify a working copy created from a branch and commit, then all changes go to the new branch and *not* the trunk. Only the modifications are stored. The rest remains a cheap copy.

4.21. Unificando

Where branches are used to maintain separate lines of development, at some stage you will want to merge the changes made on one branch back into the trunk, or vice versa.

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

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....

In general it is a good idea to perform a merge into an unmodified working copy. If you have made other changes in your WC, commit those first. If the merge does not go as you expect, you may want to revert the changes, and the Revert command will discard *all* changes including any you made before the merge.

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.

Combinar uma faixa de revisões
: Esse método cobre o caso em que foi feita uma ou mais revisões em um ramo (ou ao tronco) e se deseja transportar essas mudanças para um ramo diferente.

  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). ”

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

Unificar duas árvores diferentes
: 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/1.8/svn.advanced.vendorbr.html] in the Subversion Book.

4.21.1. Unificar um Intervalo de Revisões

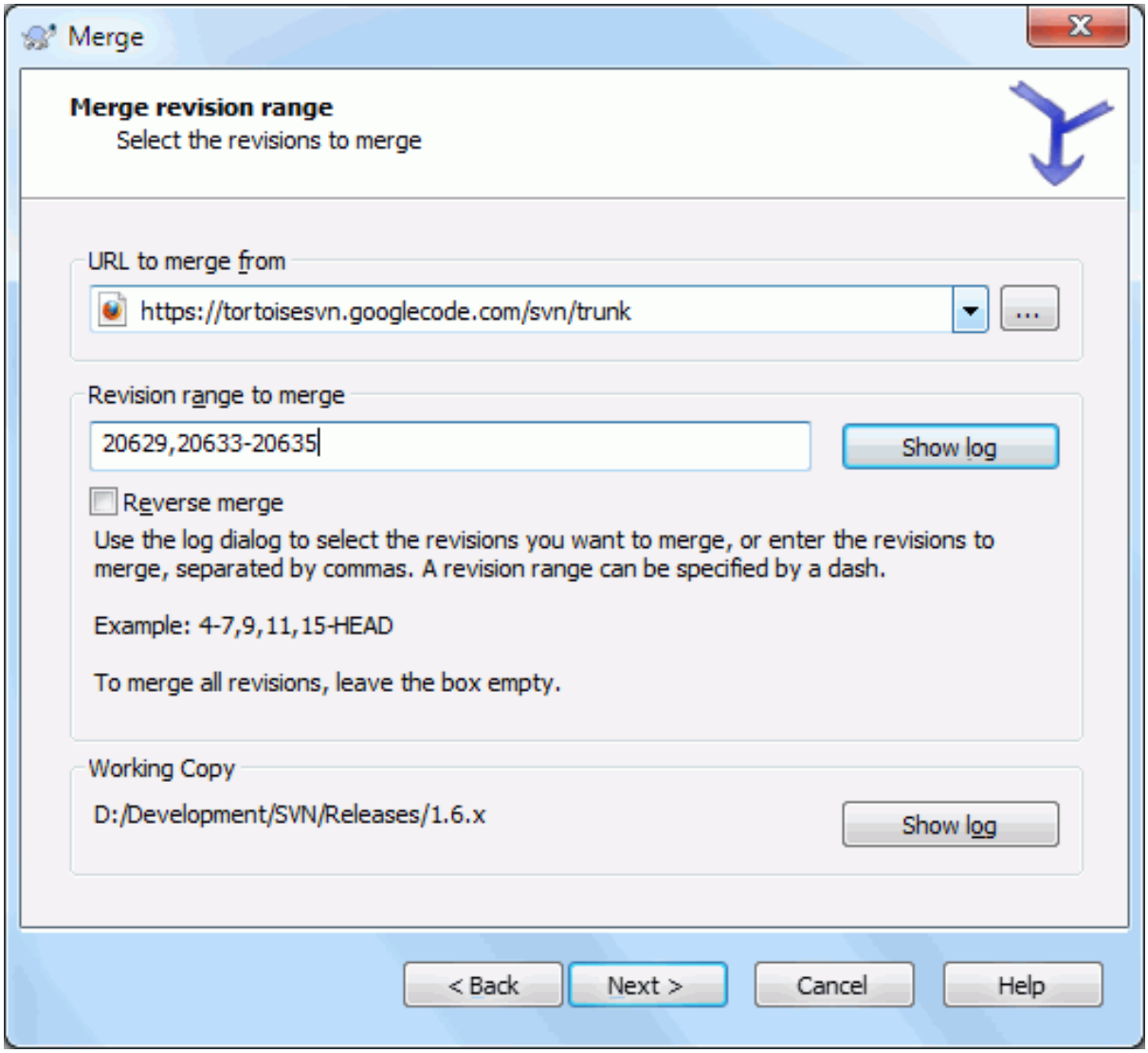


Figura 4.53. O Assistente de Combinação - Selecionar Faixa de Revisões

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.

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

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.

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

Importante

There is an important difference in the way a revision range is specified with TortoiseSVN compared to the command line client. The easiest way to visualise it is to think of a fence with posts and fence panels.

> With the command line client you specify the changes to merge using two “fence post” revisions which specify the *before* and *after* points.
>
> With TortoiseSVN you specify the changeset to merge using “fence panels”. The reason for this becomes clear when you use the log dialog to specify revisions to merge, where each revision appears as a changeset.
>
> If you are merging revisions in chunks, the method shown in the Subversion book will have you merge 100-200 this time and 200-300 next time. With TortoiseSVN you would merge 100-200 this time and 201-300 next time.
>
> This difference has generated a lot of heat on the mailing lists. We acknowledge that there is a difference from the command line client, but we believe that for the majority of GUI users it is easier to understand the method we have implemented.

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.

If you have already merged some changes from this branch, hopefully you will have made a note of the last revision merged in the log message when you committed the change. In that case, you can use Show Log for the Working Copy to trace that log message. Remembering that we are thinking of revisions as changesets, you should Use the revision after the end point of the last merge as the start point for this merge. For example, if you have merged revisions 37 to 39 last time, then the start point for this merge should be revision 40.

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 Seção 4.21.5, “Histórico de combinações” to find out more.

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

If other people may be committing changes then be careful about using the HEAD revision. It may not refer to the revision you think it does if someone else made a commit after your last update.

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

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.

Click Next and go to Seção 4.21.3, “Opções de Combinação”.

4.21.2. Combinando Duas Árvores Diferentes

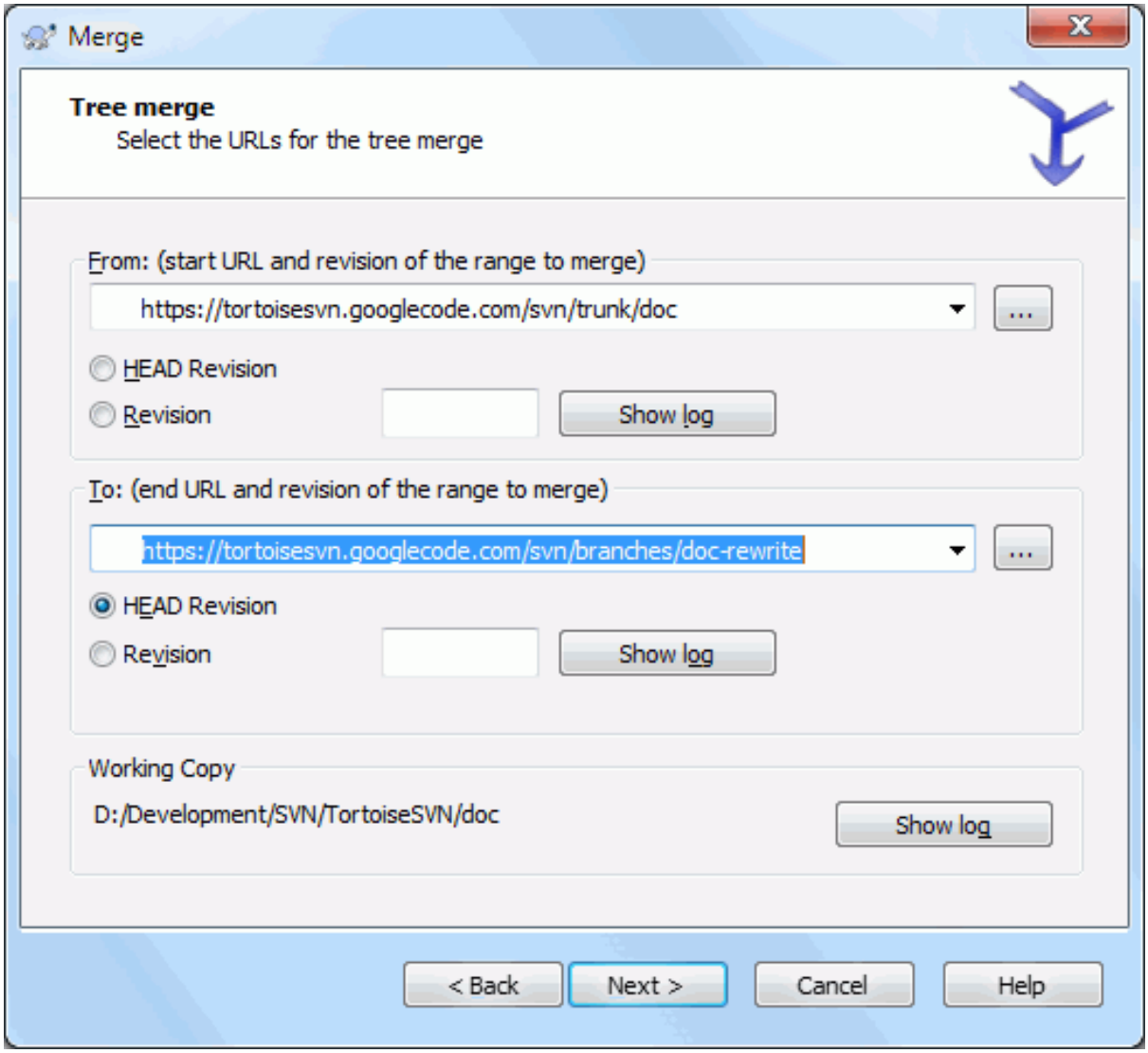


Figura 4.54. O Assistente de Combinação - Árvore de Combinação

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.

In both the From Revision field and the To Revision field, enter the last revision number at which the two trees were synchronized. If you are sure no-one else is making commits you can use the HEAD revision in both cases. If there is a chance that someone else may have made a commit since that synchronization, use the specific revision number to avoid losing more recent commits.

Você também pode utilizar o Visualizar Histórico para selecionar a revisão.

4.21.3. Opções de Combinação

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 Seção 4.3.1, “Profundidade da Obtenção”. 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/1.8/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry].

You can specify the way that line ending and whitespace changes are handled. These options are described in Seção 4.11.2, “Opções de Quebra de Linha e Espaços em Branco”. The default behaviour is to treat all whitespace and line-end differences as real changes to be merged.

The checkbox marked Force the merge is used to avoid a tree conflict where an incoming delete affects a file that is either modified locally or not versioned at all. If the file is deleted then there is no way to recover it, which is why that option is not checked by default.

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 simulates 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 files where conflicts *may* occur. Because merge tracking makes the merge process a lot more complicated, there is no guaranteed way to find out in advance whether the merge will complete without conflicts, so files marked as conflicted in a test merge may in fact merge without any problem.

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

4.21.4. Revisando os Resultados da Combinação

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.

Dica

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

Para customizar a mensagem gerada, determine as propriedades correspondentes do projeto na sua cópia de trabalho. Veja Seção 4.18.3.10, “Merge log message templates”

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.

Branch management is important. If you want to keep this branch up to date with the trunk, you should be sure to merge often so that the branch and trunk do not drift too far apart. Of course, you should still avoid repeated merging of changes, as explained above.

Dica

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.

Importante

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

4.21.5. Histórico de combinações

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.
- Quando você mostra a janela de auditoria a partir de uma janela de combinação, revisões já combinadas são apresentadas em cinza.
- Quando você está mostrando a informação de autoria para um arquivo você pode escolher mostrar o autor original das revisões combinadas, ao invés da pessoa que fez a combinação.
- Você pode marcar revisões como *não combinadas* incluindo as revisões na lista de revisões combinadas sem realmente fazer a combinação.

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://svn.apache.org/repos/asf/Subversion/trunk/notes/merge-tracking/index.html].

4.21.6. Handling Conflicts after Merge

Importante

The text in the conflict resolver dialogs are provided by the SVN library and might therefore not (yet) be translated as the TortoiseSVN dialogs are. Sorry for that.

Merging does not always go smoothly. Sometimes there is a conflict. TortoiseSVN helps you through this process by showing the *merge conflict* dialog.

Figura 4.55. The Merge Conflict Dialog

It is likely that some of the changes will have merged smoothly, while other local changes conflict with changes already committed to the repository. All changes which can be merged are merged. The Merge Conflict dialog gives you different ways of handling the lines which are in conflict.

For normal conflicts that happen due to changes in the file content or its properties, the dialog shows buttons which allow you to chose which of the conflicting parts to keep or reject.

Postpone
: Don't deal with the conflict now. Let the merge continue and resolve the conflicts after the merge is done.

Accept base
: This leaves the file as it was, without neither the changes coming from the merge nor the changes you've made in your working copy.

Accept incoming
: This discards all your local changes and uses the file as it arrives from the merge source.

Reject incoming
: This discards all the changes from the merge source and leaves the file with your local edits.

Accept incoming for conflicts
: This discards your local changes where they conflict with the changes from the merge source. But it leaves all your local changes which don't conflict.

Reject conflicts
: This discards changes from the merge source which conflict with your local changes. But it keeps all changes that don't conflict with your local changes.

Mark as resolved
: Marks the conflicts as resolved. This button is disabled until you use the button Edit to edit the conflict manually and save those changes back to the file. Once the changes are saved, the button becomes enabled.

Editar
: Starts the merge editor so you can resolve the conflicts manually. Don't forget to save the file so the button Mark as resolved becomes enabled.

If there's a tree conflict, please first see Seção 4.6.3, “Conflitos de Estrutura” about the various types of tree conflicts and how and why they can happen.

To resolve tree conflicts after a merge, a dialog is shown with various options on how to resolve the conflict:

Figura 4.56. The Merge Tree Conflict Dialog

Since there are various possible tree conflict situations, the dialog will show buttons to resolve those depending on the specific conflict. The button texts and labels explain what the option to resolve the conflict does. If you're not sure, either cancel the dialog or use the Postpone button to resolve the conflict later.

4.21.7. Funcionalidade de Manutenção de Ramo

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

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

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

Another (fast) way to merge all changes from trunk to the feature branch is to use the TortoiseSVN → Merge all... from the extended context menu (hold down the **Shift** key while you right click on the file).

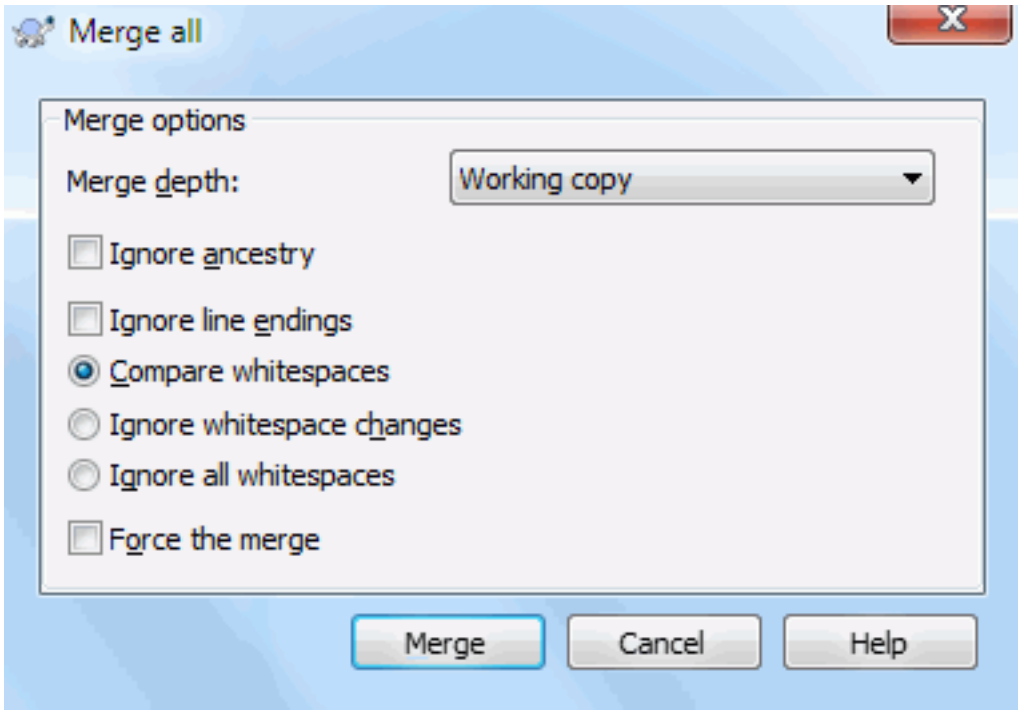


Figura 4.57. The Merge-All Dialog

This dialog is very easy. All you have to do is set the options for the merge, as described in Seção 4.21.3, “Opções de Combinação”. The rest is done by TortoiseSVN automatically using merge tracking.

4.22. Bloqueando

Subversion generally works best without locking, using the “Copy-Modify-Merge” methods described earlier in Seção 2.2.3, “A solução Copiar-Modificar-Unificar”. However there are a few instances when you may need to implement some form of locking policy.

- Você está usando arquivos “não combináveis”, por exemplo, arquivos de imagens. Se duas pessoas mudarem o mesmo arquivo, a combinação não será possível, assim as alterações de alguém serão perdidas.

- Sua empresa desde sempre usou um sistema de controle de versão por bloqueio no passado e decidiu que “bloquear é o melhor”.

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

The Three Meanings of “Lock”

In this section, and almost everywhere in this book, the words “lock” and “locking” describe a mechanism for mutual exclusion between users to avoid clashing commits. Unfortunately, there are two other sorts of “lock” with which Subversion, and therefore this book, sometimes needs to be concerned.

The second is `working copy locks`, used internally by Subversion to prevent clashes between multiple Subversion clients operating on the same working copy. Usually you get these locks whenever a command like update/commit/... is interrupted due to an error. These locks can be removed by running the cleanup command on the working copy, as described in Seção 4.17, “Limpar”.

And third, files and folders can get locked if they're in use by another process, for example if you have a word document opened in Word, that file is locked and can not be accessed by TortoiseSVN.

You can generally forget about these other kinds of locks until something goes wrong that requires you to care about them. In this book, “lock” means the first sort unless the contrary is either clear from context or explicitly stated.

4.22.1. Como o Bloqueio Funcionada no Subversion

Por padrão, nada é bloqueado e qualquer um que tenha acesso para submissão pode alterar qualquer arquivo a qualquer momento. Outros atualizarão suas cópias de trabalho periodicamente e as mudanças no repositório serão combinadas com as alterações locais.

If you *Get a Lock* on a file, then only you can commit that file. Commits by all other users will be blocked until you release the lock. A locked file cannot be modified in any way in the repository, so it cannot be deleted or renamed either, except by the lock owner.

Importante

Um bloquei não está associado a um usuário específico, mas para um usuário específico e sua cópia de trabalho. Existindo um bloqueio em uma cópia de trabalho também prevê que o mesmo usuário submete o arquivo bloqueado de outra cópia de trabalho.

As an example, imagine that user Jon has a working copy on his office PC. There he starts working on an image, and therefore acquires a lock on that file. When he leaves his office he's not finished yet with that file, so he doesn't release that lock. Back at home Jon also has a working copy and decides to work a little more on the project. But he can't modify or commit that same image file, because the lock for that file resides in his working copy in the office.

However, other users will not necessarily know that you have taken out a lock. Unless they check the lock status regularly, the first they will know about it is when their commit fails, which in most cases is not very useful. To make it easier to manage locks, there is a new Subversion property `svn:needs-lock`. When this property is set (to any value) on a file, whenever the file is checked out or updated, the local copy is made read-only *unless* that working copy holds a lock for the file. This acts as a warning that you should not edit that file unless you have first acquired a lock. Files which are versioned and read-only are marked with a special overlay in TortoiseSVN to indicate that you need to acquire a lock before editing.

Locks are recorded by working copy location as well as by owner. If you have several working copies (at home, at work) then you can only hold a lock in *one* of those working copies.

If one of your co-workers acquires a lock and then goes on holiday without releasing it, what do you do? Subversion provides a means to force locks. Releasing a lock held by someone else is referred to as *Breaking* the lock, and forcibly acquiring a lock which someone else already holds is referred to as *Stealing* the lock. Naturally these are not things you should do lightly if you want to remain friends with your co-workers.

Locks are recorded in the repository, and a lock token is created in your local working copy. If there is a discrepancy, for example if someone else has broken the lock, the local lock token becomes invalid. The repository is always the definitive reference.

4.22.2. Obtendo uma trava

Select the file(s) in your working copy for which you want to acquire a lock, then select the command TortoiseSVN → Get Lock....

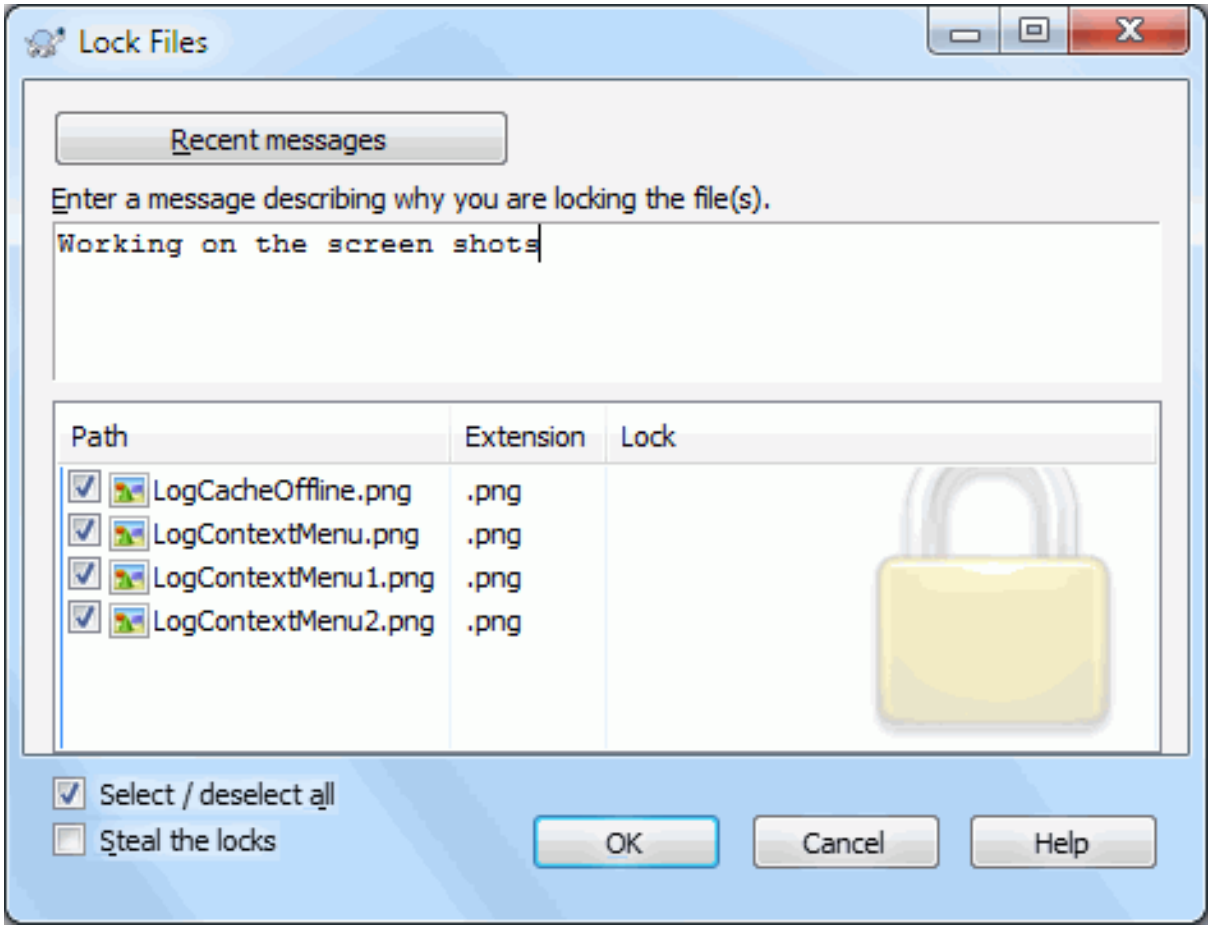


Figura 4.58. A Janela de Bloqueio

A dialog appears, allowing you to enter a comment, so others can see why you have locked the file. The comment is optional and currently only used with Svnserve based repositories. If (and *only* if) you need to steal the lock from someone else, check the Steal lock box, then click on OK.

You can set the project property `tsvn:logtemplatelock` to provide a message template for users to fill in as the lock message. Refer to Seção 4.18, “Configurações do Projeto” for instructions on how to set properties.

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

4.22.3. Liberando uma trava

To make sure you don't forget to release a lock you don't need any more, locked files are shown in the commit dialog and selected by default. If you continue with the commit, locks you hold on the selected files are removed, even if the files haven't been modified. If you don't want to release a lock on certain files, you can uncheck them (if they're not modified). If you want to keep a lock on a file you've modified, you have to enable the Keep locks checkbox before you commit your changes.

To release a lock manually, select the file(s) in your working copy for which you want to release the lock, then select the command TortoiseSVN → Release Lock There is nothing further to enter so TortoiseSVN will contact the repository and release the locks. You can also use this command on a folder to release all locks recursively.

4.22.4. Verificando a Situação do Bloqueio

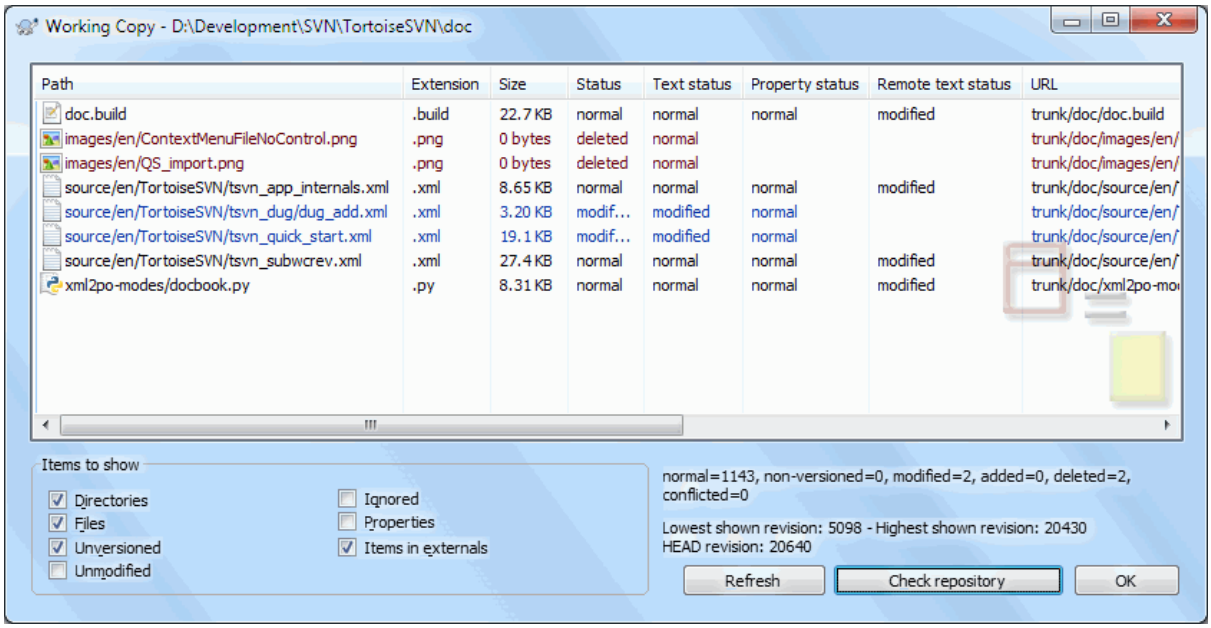


Figura 4.59. A Janela de Verificação de Modificações

To see what locks you and others hold, you can use TortoiseSVN → Check for Modifications.... Locally held lock tokens show up immediately. To check for locks held by others (and to see if any of your locks are broken or stolen) you need to click on Check Repository.

A partir deste menu de contexto você pode também obter ou liberar bloqueios, assim como quebrar e roubar os bloqueios mantidos por outros.

Evite a quebra e Roubos de Bloqueios

If you break or steal someone else's lock without telling them, you could potentially cause loss of work. If you are working with unmergeable file types and you steal someone else's lock, once you release the lock they are free to check in their changes and overwrite yours. Subversion doesn't lose data, but you have lost the team-working protection that locking gave you.

4.22.5. Tornando arquivos não-bloqueados somente leitura

As mentioned above, the most effective way to use locking is to set the `svn:needs-lock` property on files. Refer to Seção 4.18, “Configurações do Projeto” for instructions on how to set properties. Files with this property set will always be checked out and updated with the read-only flag set unless your working copy holds a lock.

As a reminder, TortoiseSVN uses a special overlay to indicate this.

If you operate a policy where every file has to be locked then you may find it easier to use Subversion's auto-props feature to set the property automatically every time you add new files. Read Seção 4.18.1.5, “Definição automática de propriedade” for further information.

4.22.6. Os Scripts de Gancho de Bloqueio

Quando você criar um novo repositório com o Subversion 1.2 ou superior, quatro modelos de gancho são criados no diretório `hooks` no repositório. Eles são chamados antes e depois de efetuar um bloqueio, e antes e depois de liberar um bloqueio.

It is a good idea to install a `post-lock` and `post-unlock` hook script on the server which sends out an email indicating the file which has been locked. With such a script in place, all your users can be notified if someone locks/unlocks a file. You can find an example hook script `hooks/post-lock.tmpl` in your repository folder.

Você também pode usar ganchos para proibir a quebra ou o roubo de travas, ou talvez limitá-lo a um administrador nomeado. Ou talvez você queira enviar um e-mail para o proprietário quando um de seus bloqueios é quebrado ou roubado.

Leia Seção 3.3, “Rotinas de eventos no servidor” para saber mais.

4.23. Criando e Aplicando Correções

For open source projects (like this one) everyone has read access to the repository, and anyone can make a contribution to the project. So how are those contributions controlled? If just anyone could commit changes, the project would be permanently unstable and probably permanently broken. In this situation the change is managed by submitting a *patch* file to the development team, who do have write access. They can review the patch first, and then either submit it to the repository or reject it back to the author.

Patch files are simply Unified-Diff files showing the differences between your working copy and the base revision.

4.23.1. Cirando um Arquivo de Correções

First you need to make *and test* your changes. Then instead of using TortoiseSVN → Commit... on the parent folder, you select TortoiseSVN → Create Patch...

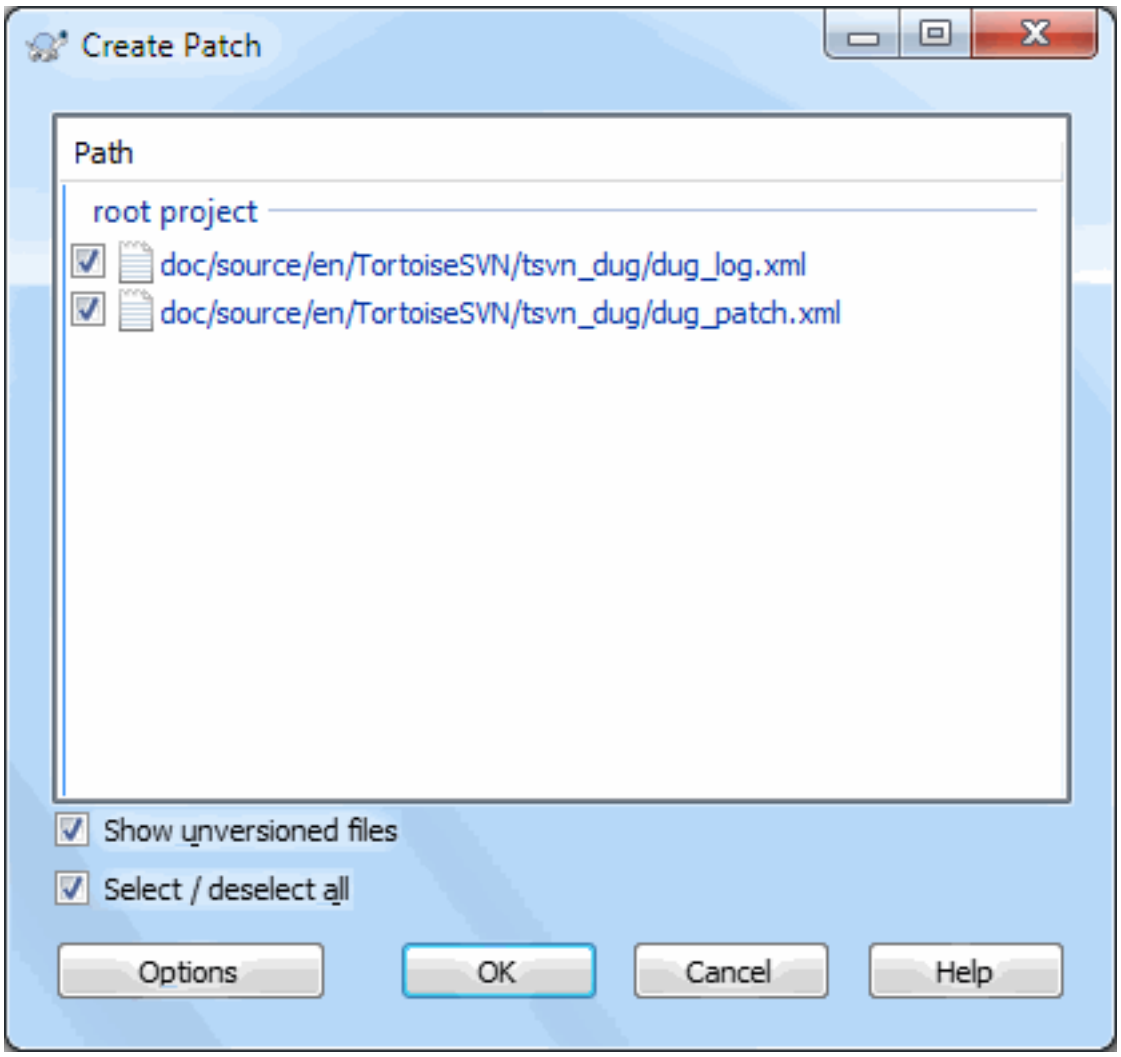


Figura 4.60. A Janela de Criação de Correções

you can now select the files you want included in the patch, just as you would with a full commit. This will produce a single file containing a summary of all the changes you have made to the selected files since the last update from the repository.

As colunas nesta janela podem ser personalizadas da mesma forma como as colunas da janela Verificar alterações. Leia Seção 4.7.3, “Estado Local e Remoto” para mais detalhes.

By clicking on the Options button you can specify how the patch is created. For example you can specify that changes in line endings or whitespaces are not included in the final patch file.

You can produce separate patches containing changes to different sets of files. Of course, if you create a patch file, make some more changes to the *same* files and then create another patch, the second patch file will include *both* sets of changes.

Just save the file using a filename of your choice. Patch files can have any extension you like, but by convention they should use the `.patch` or `.diff` extension. You are now ready to submit your patch file.

Dica

Do not save the patch file with a `.txt` extension if you intend to send it via email to someone else. Plain text files are often mangled with by the email software and it often happens that whitespaces and newline chars are automatically converted and compressed. If that happens, the patch won't apply smoothly. So use `.patch` or `.diff` as the extension when you save the patch file.

You can also save the patch to the clipboard instead of to a file. You might want to do this so that you can paste it into an email for review by others. Or if you have two working copies on one machine and you want to transfer changes from one to the other, a patch on the clipboard is a convenient way of doing this.

If you prefer, you can create a patch file from within the Commit or Check for Modifications dialogs. Just select the files and use the context menu item to create a patch from those files. If you want to see the Options dialog you have to hold **shift** when you right click.

4.23.2. Aplicando um Arquivo de Correções

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.

In order to apply a patch file to your working copy, you need to have at least read access to the repository. The reason for this is that the merge program must reference the changes back to the revision against which they were made by the remote developer.

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. Note that this option only appears if you saved the patch to the clipboard using TortoiseSVN → Create Patch.... Copying a patch to the clipboard from another app will not make the button appear.

Alternatively, if the patch file has a `.patch` or `.diff` extension, you can right click on it directly and select TortoiseSVN → Apply Patch.... In this case you will be prompted to enter a working copy location.

These two methods just offer different ways of doing the same thing. With the first method you select the WC and browse to the patch file. With the second you select the patch file and browse to the WC.

Once you have selected the patch file and working copy location, TortoiseMerge runs to merge the changes from the patch file with your working copy. A small window lists the files which have been changed. Double click on each one in turn, review the changes and save the merged files.

The remote developer's patch has now been applied to your working copy, so you need to commit to allow everyone else to access the changes from the repository.

4.24. Quem Mudou Qual Linha?

Sometimes you need to know not only what lines have changed, but also who exactly changed specific lines in a file. That's when the TortoiseSVN → Blame... command, sometimes also referred to as *annotate* command comes in handy.

Estas listas de comandos, para cada linha no arquivo, o autor e a revisão da alteração da linha.

4.24.1. Autoria por Arquivos

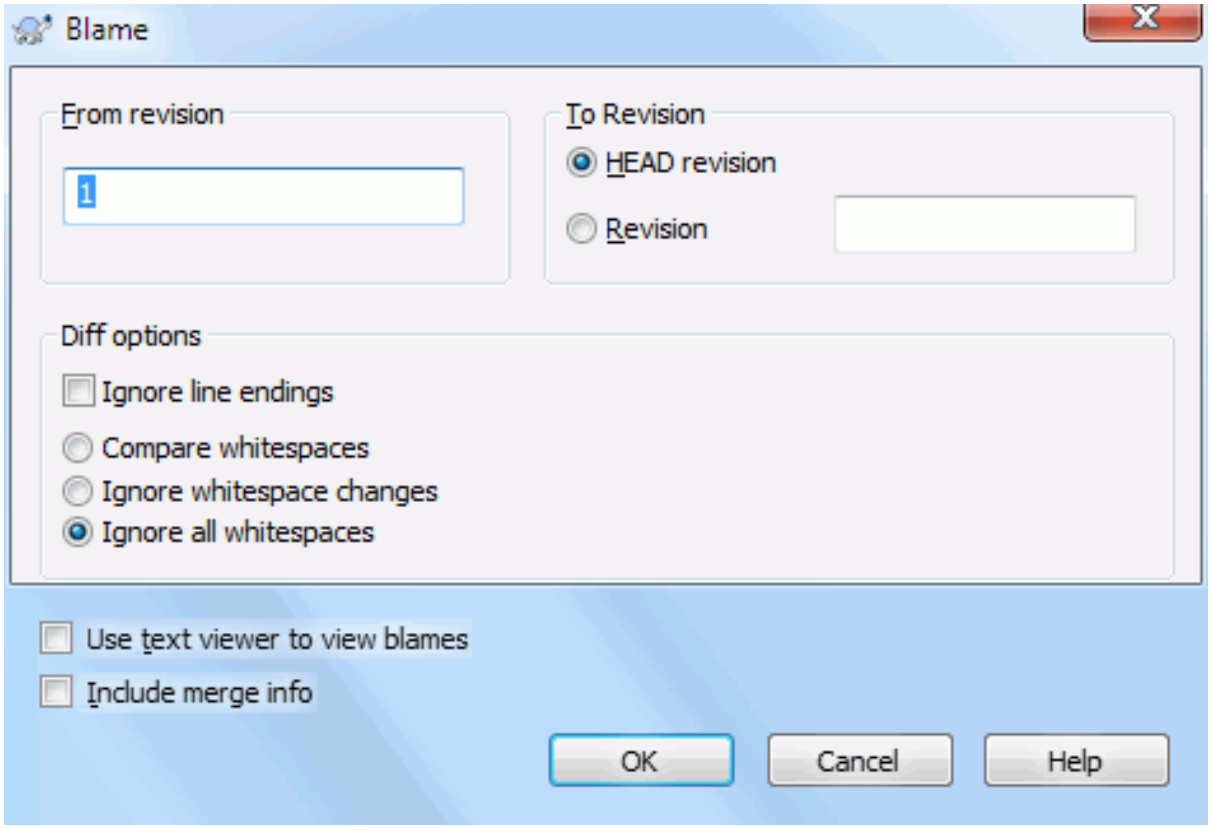


Figura 4.61. A Janela de Anotação / Autoria

If you're not interested in changes from earlier revisions you can set the revision from which the blame should start. Set this to `1`, if you want the blame for *every* revision.

By default the blame file is viewed using *TortoiseBlame*, which highlights the different revisions to make it easier to read. If you wish to print or edit the blame file, select Use Text viewer to view blames.

You can specify the way that line ending and whitespace changes are handled. These options are described in Seção 4.11.2, “Opções de Quebra de Linha e Espaços em Branco”. 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.

You can include merge information as well if you wish, although this option can take considerably longer to retrieve from the server. When lines are merged from another source, the blame information shows the revision the change was made in the original source as well as the revision when it was merged into this file.

Once you press OK TortoiseSVN starts retrieving the data to create the blame file. Once the blame process has finished the result is written into a temporary file and you can view the results.

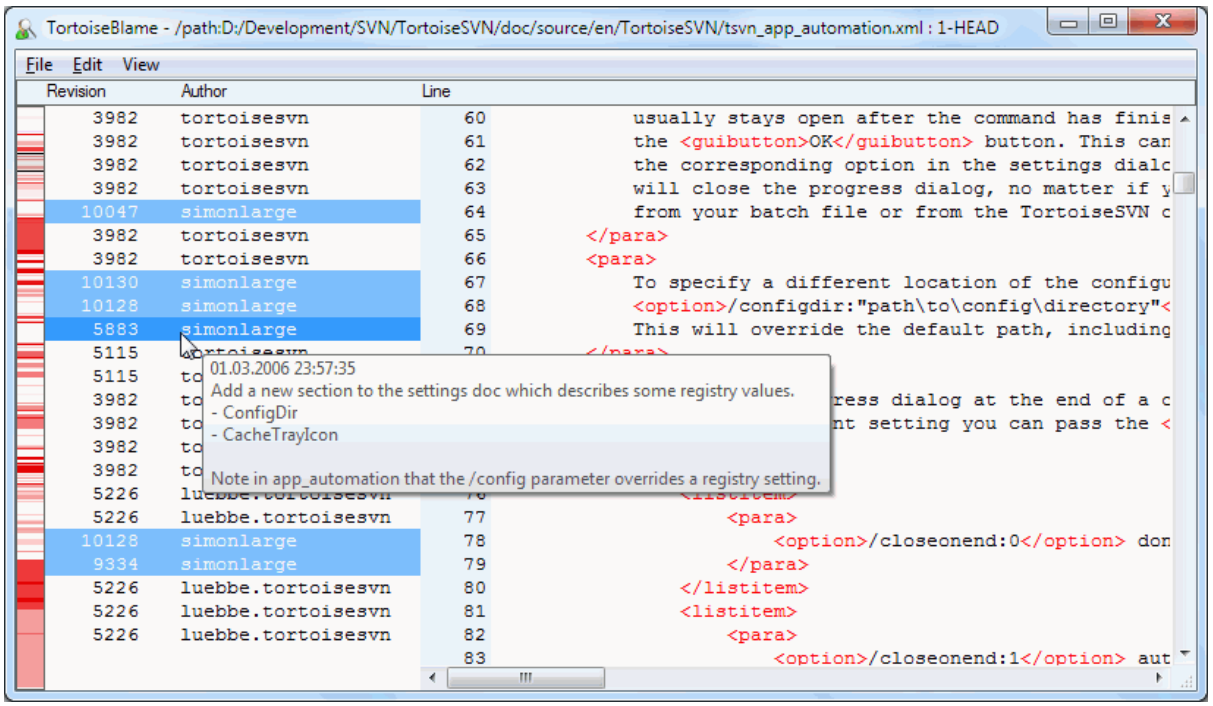


Figura 4.62. TortoiseBlame

TortoiseBlame, which is included with TortoiseSVN, makes the blame file easier to read. When you hover the mouse over a line in the blame info column, all lines with the same revision are shown with a darker background. Lines from other revisions which were changed by the same author are shown with a light background. The colouring may not work as clearly if you have your display set to 256 colour mode.

If you left click on a line, all lines with the same revision are highlighted, and lines from other revisions by the same author are highlighted in a lighter colour. This highlighting is sticky, allowing you to move the mouse without losing the highlights. Click on that revision again to turn off highlighting.

The revision comments (log message) are shown in a hint box whenever the mouse hovers over the blame info column. If you want to copy the log message for that revision, use the context menu which appears when you right click on the blame info column.

You can search within the Blame report using Edit → Find.... This allows you to search for revision numbers, authors and the content of the file itself. Log messages are not included in the search - you should use the Log Dialog to search those.

You can also jump to a specific line number using Edit → Go To Line....

When the mouse is over the blame info columns, a context menu is available which helps with comparing revisions and examining history, using the revision number of the line under the mouse as a reference. Context menu → Blame previous revision generates a blame report for the same file, but using the previous revision as the upper limit. This gives you the blame report for the state of the file just before the line you are looking at was last changed. Context menu → Show changes starts your diff viewer, showing you what changed in the referenced revision. Context menu → Show log displays the revision log dialog starting with the referenced revision.

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 and you requested merge info when starting the blame, merged lines are shown slightly differently. Where a line has changed as a result of merging from another path, TortoiseBlame will show

the revision and author of the last change in the original file rather than the revision where the merge took place. These lines are indicated by showing the revision and author in italics. The revision where the merge took place is shown separately in the tooltip when you hover the mouse over the blame info columns. If you do not want merged lines shown in this way, uncheck the Include merge info checkbox when starting the blame.

If you want to see the paths involved in the merge, select View → Merge paths. This shows the path where the line was last changed, excluding changes resulting from a merge.

The revision shown in the blame information represents the last revision where the content of that line changed. If the file was created by copying another file, then until you change a line, its blame revision will show the last change in the original source file, not the revision where the copy was made. This also applies to the paths shown with merge info. The path shows the repository location where the last change was made to that line.

The settings for TortoiseBlame can be accessed using TortoiseSVN → Settings... on the TortoiseBlame tab. Refer to Seção 4.31.9, “Configurações TortoiseBlame”.

4.24.2. Diferenças de Autoria

One of the limitations of the Blame report is that it only shows the file as it was in a particular revision, and the last person to change each line. Sometimes you want to know what change was made, as well as who made it. If you right click on a line in TortoiseBlame you have a context menu item to show the changes made in that revision. But if you want to see the changes *and* the blame information simultaneously then you need a combination of the diff and blame reports.

A janela de auditoria de revisão inclui diversas opções que lhe permitem fazer isto.

Revisões de Autoria

In the top pane, select 2 revisions, then select Context menu → Blame revisions. This will fetch the blame data for the 2 revisions, then use the diff viewer to compare the two blame files.

Alterações de Autoria

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.

Comparar e Responsabilizar com a Referência BASE

Show the log for a single file, and in the top pane, select a single revision, then select Context menu → Compare and Blame with Working BASE. This will fetch the blame data for the selected revision, and for the file in the working BASE, then use the diff viewer to compare the two blame files.

4.25. O Navegador de Repositório

Sometimes you need to work directly on the repository, without having a working copy. That's what the *Repository Browser* is for. Just as the explorer and the icon overlays allow you to view your working copy, so the Repository Browser allows you to view the structure and status of the repository.

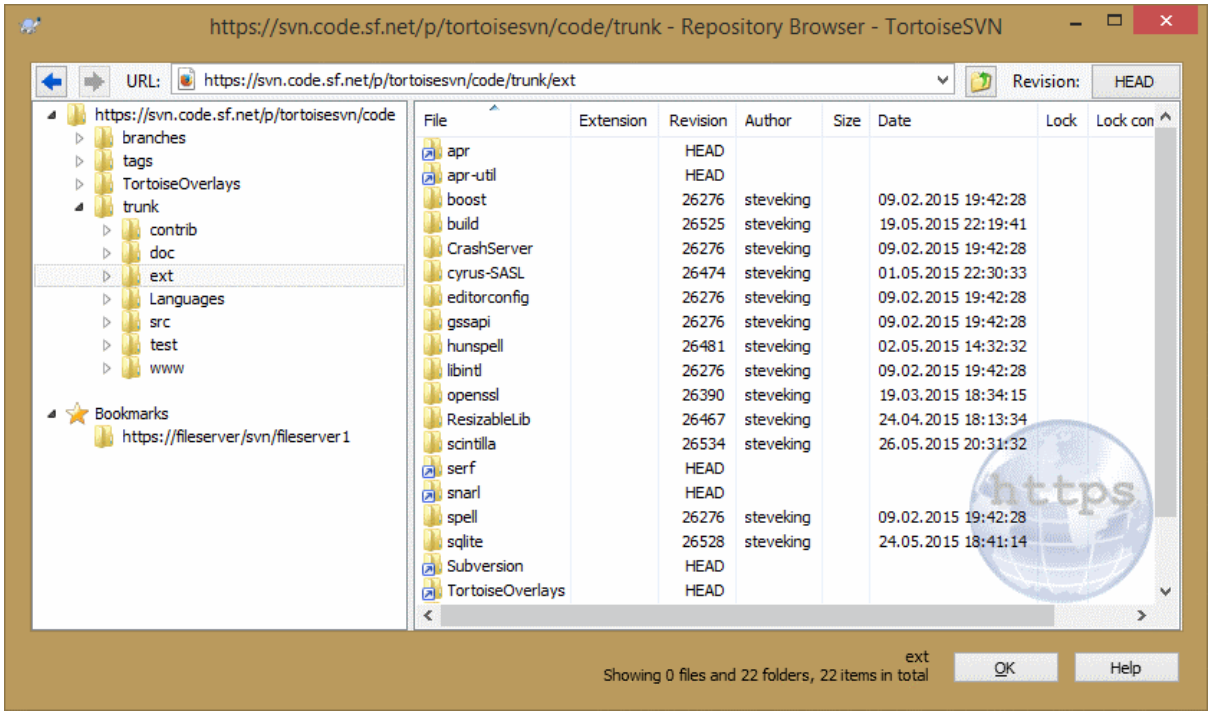


Figura 4.63. O Navegador de Repositório

With the Repository Browser you can execute commands like copy, move, rename, ... directly on the repository.

The repository browser looks very similar to the Windows explorer, except that it is showing the content of the repository at a particular revision rather than files on your computer. In the left pane you can see a directory tree, and in the right pane are the contents of the selected directory. At the top of the Repository Browser Window you can enter the URL of the repository and the revision you want to browse.

Folders included with the `svn:externals` property are also shown in the repository browser. Those folders are shown with a small arrow on them to indicate that they are not part of the repository structure, just links.

Just like Windows explorer, you can click on the column headings in the right pane if you want to set the sort order. And as in explorer there are context menus available in both panes.

O menu de contexto para um arquivo lhe permite:

- Abre o arquivo selecionado, tanto com o visualizar padrão para o tipo de arquivo, ou com o programa que você escolheu.
- Edit the selected file. This will checkout a temporary working copy and start the default editor for that file type. When you close the editor program, if changes were saved then a commit dialog appears, allowing you to enter a comment and commit the change.
- Show the revision log for that file, or show a graph of all revisions so you can see where the file came from.
- Responsabilizar o arquivo, para ver quem mudou cada linha e quando.
- Checkout a single file. This creates a “sparse” working copy which contains just this one file.
- Apagar ou renomear o arquivo.
- Salvar uma cópia não versionada do arquivo para o seu disco.
- Copie o URL mostrado na barra de endereços para a área de transferência.

- Fazer uma cópia do arquivo, tanto para uma parte diferente do repositório quanto para uma cópia de trabalho roteado no mesmo repositório.
- Visualizar/Editar as propriedades do arquivo.
- Create a shortcut so that you can quickly start repo browser again, opened directly at this location.

O menu de contexto lhe permite:

- Mostrar a auditoria de revisão para o diretório, ou mostrar o gráfico de todas as revisões para que você possa ver a origem do diretório.
- Exportar o diretório para uma cópia local não versionada em seu disco.
- Obter o diretório para produzir uma cópia de trabalho local em seu disco.
- Criar um novo diretório no repositório.
- Add unversioned files or folders directly to the repository. This is effectively the Subversion Import operation.
- Apagar ou renomear o diretório.
- Fazer uma cópia do diretório, tanto para uma parte diferente do repositório, ou para uma cópia de trabalho roteada no mesmo diretório. Isto pode também ser usado para criar um ramo/versão sem que seja necessário obter uma cópia de trabalho.
- Visualizar/Editar as propriedades do diretório.
- Marcar o diretório para comparação. O diretório marcado é mostrado em negrito.
- Compare the folder with a previously marked folder, either as a unified diff, or as a list of changed files which can then be visually diffed using the default diff tool. This can be particularly useful for comparing two tags, or trunk and branch to see what changed.

Se você selecionar dois diretório no painel direito, você poderá ver as diferenças tanto como um arquivo unificado de diferenças, ou também como uma lista de arquivos a qual pode ser visualmente comparados usando a ferramenta padrão de diferenciação.

Se você selecionar vários diretórios no painel direito, você poderá obter todos eles de uma vez em um diretório acima em comum.

If you select 2 tags which are copied from the same root (typically `/trunk/`), you can use Context Menu → Show Log... to view the list of revisions between the two tag points.

External items (referenced using `svn:externals` are also shown in the repository browser, and you can even drill down into the folder contents. External items are marked with a red arrow over the item.

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.

Você também pode usar o navegador de repositório para operações de arrastar-e-soltar. Se você arrastar um diretório do Windows Explorer para dentro do navegador de repositório, ele será importado para o repositório. Note que se você arrastar múltiplos itens, eles serão importados em submissões individuais.

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.

Whenever you make changes in the repository using one of these methods, you will be presented with a log message entry dialog. If you dragged something by mistake, this is also your chance to cancel the action.

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**.

Bookmarked urls/repositories are shown below the current repository folders in the left tree view. You can add entries there by right clicking on any file or folder and select Context Menu → Add to Bookmarks. Clicking on a bookmark will browse to that repository and file/folder.

4.26. Gráfico de Revisões

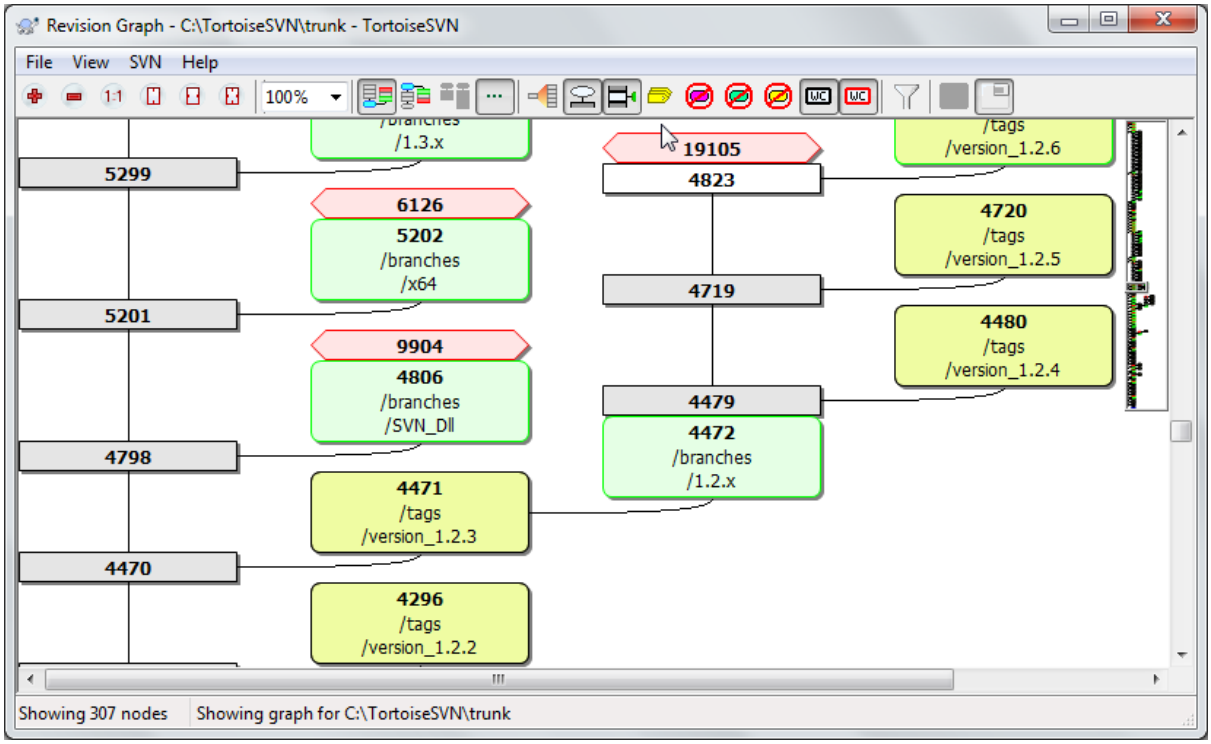


Figura 4.64. Um Gráfico de Revisões

Sometimes you need to know where branches and tags were taken from the trunk, and the ideal way to view this sort of information is as a graph or tree structure. That's when you need to use TortoiseSVN → Revision Graph...

This command analyses the revision history and attempts to create a tree showing the points at which copies were taken, and when branches/tags were deleted.

Importante

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

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

4.26.1. Nós do Gráfico de Revisões

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

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

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

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

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

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

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

Item normal
: Todos os outros itens são apresentados usando um retângulo plano.

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

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

4.26.2. Mudando a Visão

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.

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

This mode groups changes by branch, so that there is no global revision ordering: Consecutive revisions on a branch will be shown in (often) consecutive lines. Sub-branches, however, are arranged in such a way that later branches will be shown in the same column above earlier branches to keep the graph slim. As a result, a given row may contain changes from different revisions.

Mais antigo primeiro
: 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.

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

Reduzir linhas cruzadas
: This option is normally enabled and avoids showing the graph with a lot of confused crossing lines. However this may also make the layout columns appear in less logical places, for example in a diagonal line rather than a column, and the graph may require a larger area to draw. If this is a problem you can disable the option from the View menu.

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

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

Mostrar revisões ATUAIS
: This ensures that the latest revision on every branch is always shown on the graph.

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

Rótulo duplicado
: When a project has many tags, showing every tag as a separate node on the graph takes a lot of space and obscures the more interesting development branch structure. At the same time you may need to be able to access the tag content easily so that you can compare revisions. This option hides the nodes for tags and shows them instead in the tooltip for the node that they were copied from. A tag icon on the right side of the source node indicates that tags were made. This greatly simplifies the view.

  Note that if a tag is itself used as the source for a copy, perhaps a new branch based on a tag, then that tag will be shown as a separate node rather than folded.

Ocultar caminhos apagados
: Hides paths which are no longer present at the HEAD revision of the repository, e.g. deleted branches.

  If you have selected the Fold tags option then a deleted branch from which tags were taken will still be shown, otherwise the tags would disappear too. The last revision that was tagged will be shown in the colour used for deleted nodes instead of showing a separate deletion revision.

  If you select the Hide tags option then these branches will disappear again as they are not needed to show the tags.

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

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

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

Filtro
: 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.

  If you hide a particular path and that node has child nodes, the children will be shown as a separate tree. If you want to hide all children as well, use the Remove the whole subtree(s) checkbox.

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

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

4.26.3. Usando o Gráfico

To make it easier to navigate a large graph, use the overview window. This shows the entire graph in a small window, with the currently displayed portion highlighted. You can drag the highlighted area to change the displayed region.

The revision date, author and comments are shown in a hint box whenever the mouse hovers over a revision box.

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.

You can view the differences as a Unified-Diff file, which shows all differences in a single file with minimal context. If you opt to Context Menu → Compare Revisions you will be presented with a list of changed files. Double click on a file name to fetch both revisions of the file and compare them using the visual difference tool.

If you right click on a revision you can use Context Menu → Show Log to view the history.

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

Aprenda a Ler o Gráfico de Revisões

First-time users may be surprised by the fact that the revision graph shows something that does not match the user's mental model. If a revision changes multiple copies or branches of a file or folder,

for instance, then there will be multiple nodes for that single revision. It is a good practice to start with the leftmost options in the toolbar and customize the graph step-by-step until it comes close to your mental model.

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

4.26.4. Atualizando a Visualização

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

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

4.26.5. Pruning Trees

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

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

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

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

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

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

4.27. Exportando um Cópia de Trabalho do Subversion

Sometimes you may want a clean copy of your working tree without the `.svn` directory, e.g. to create a zipped tarball of your source, or to export to a web server. Instead of making a copy and then deleting the `.svn` directory manually, TortoiseSVN offers the command TortoiseSVN → Export.... Exporting from a URL and exporting from a working copy are treated slightly differently.

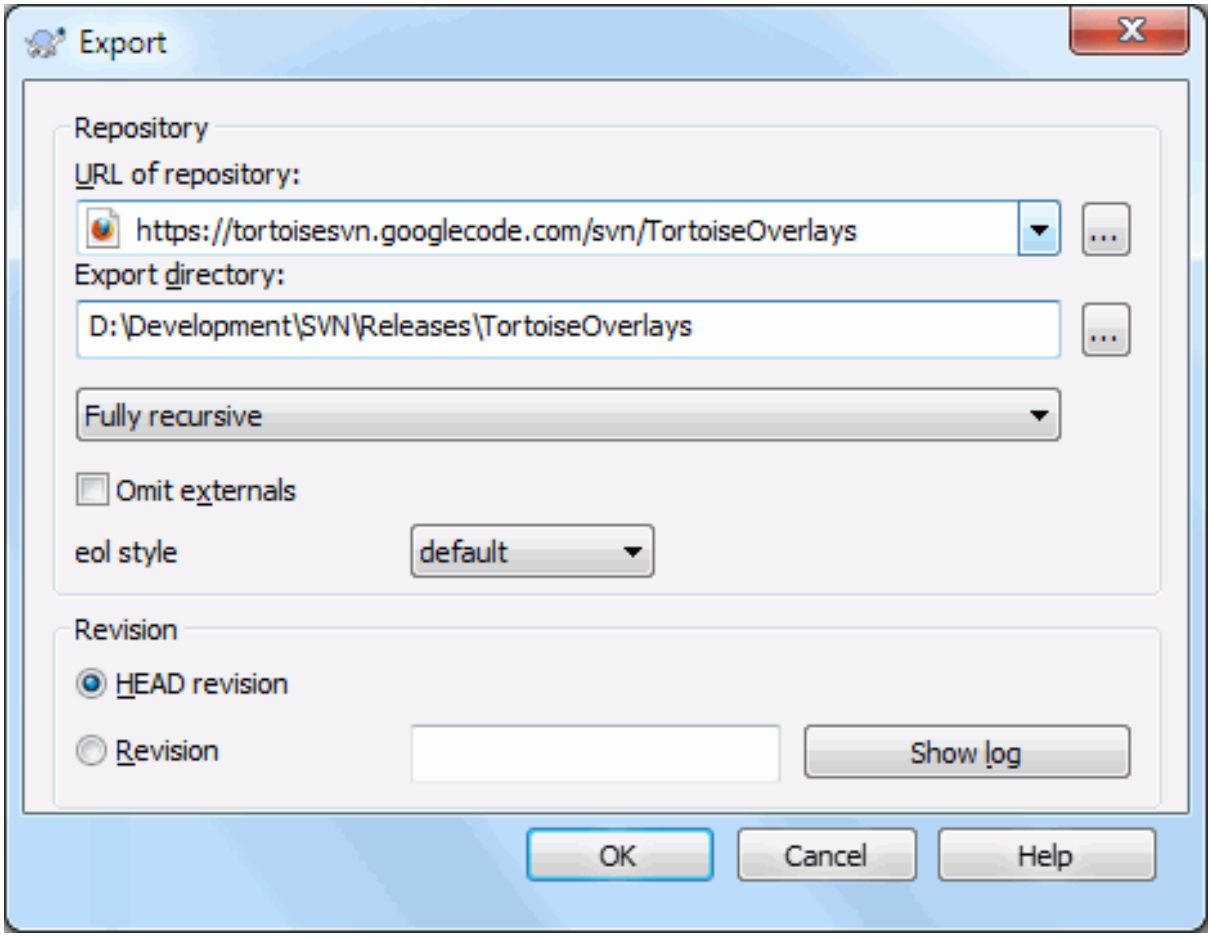


Figura 4.65. A Janela Exportar de uma URL

If you execute this command on an unversioned folder, TortoiseSVN will assume that the selected folder is the target, and open a dialog for you to enter the URL and revision to export from. This dialog has options to export only the top level folder, to omit external references, and to override the line end style for files which have the `svn:eol-style` property set.

Of course you can export directly from the repository too. Use the Repository Browser to navigate to the relevant subtree in your repository, then use Context Menu → Export. You will get the Export from URL dialog described above.

If you execute this command on your working copy you'll be asked for a place to save the *clean* working copy without the `.svn` folder. By default, only the versioned files are exported, but you can use the Export unversioned files too checkbox to include any other unversioned files which exist in your WC and not in the repository. External references using `svn:externals` can be omitted if required.

Another way to export from a working copy is to right drag the working copy folder to another location and choose Context Menu → SVN Export versioned items here or Context Menu → SVN Export all items here or Context Menu → SVN Export changed items here. The second option includes the unversioned files as well. The third option exports only modified items, but maintains the folder structure.

When exporting from a working copy, if the target folder already contains a folder of the same name as the one you are exporting, you will be given the option to overwrite the existing content, or to create a new folder with an automatically generated name, e.g. `Target (1)`.

Exportando um arquivo apenas

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

To export single files with TortoiseSVN, you have to use the repository browser (Seção 4.25, “O Navegador de Repositório”). Simply drag the file(s) you want to export from the repository browser to where you want them in the explorer, or use the context menu in the repository browser to export the files.

Exportando um Conjunto de Alterações

If you want to export a copy of your project tree structure but containing only the files which have changed in a particular revision, or between any two revisions, use the compare revisions feature described in Seção 4.11.3, “Comparando Diretórios”.

If you want to export your working copy tree structure but containing only the files which are locally modified, refer to SVN Export changed items here above.

4.27.1. Removendo uma cópia de trabalho do controle de versão

Sometimes you have a working copy which you want to convert back to a normal folder without the `.svn` directory. All you need to do is delete the `.svn` directory from the working copy root.

Alternatively you can export the folder to itself. In Windows Explorer right drag the working copy root folder from the file pane onto itself in the folder pane. TortoiseSVN detects this special case and asks if you want to make the working copy unversioned. If you answer *yes* the control directory will be removed and you will have a plain, unversioned directory tree.

4.28. Realocando uma cópia de trabalho

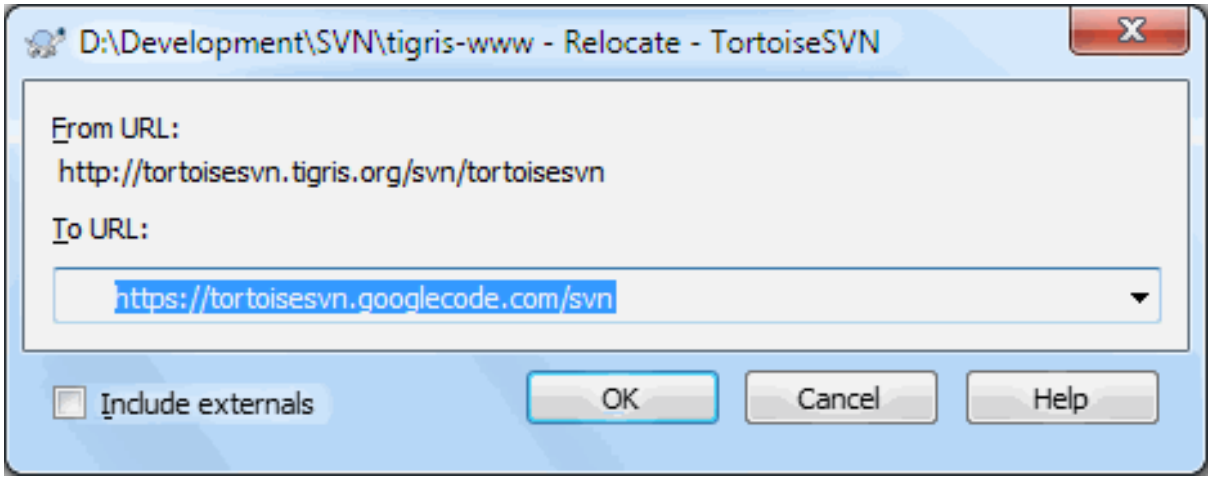


Figura 4.66. A Janela de Realocação

If your repository has for some reason changed it's location (IP/URL). Maybe you're even stuck and can't commit and you don't want to checkout your working copy again from the new location and to move all your changed data back into the new working copy, TortoiseSVN → Relocate is the command you are looking for. It basically does very little: it rewrites all URLs that are associated with each file and folder with the new URL.

Nota

This operation only works on working copy *roots*. So the context menu entry is only shown for working copy roots.

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

Atenção

This is a very infrequently used operation. The relocate command is *only* used if the URL of the repository root has changed. Possible reasons are:

- O endereço IP do servidor foi modificado.
- O protocolo foi modificado (e.g. http:// to https://).
- O diretório raiz do repositório no servidor configurado foi modificado.

Put another way, you need to relocate when your working copy is referring to the same location in the same repository, but the repository itself has moved.

Isto não se aplica se:

- You want to move to a different Subversion repository. In that case you should perform a clean checkout from the new repository location.
- You want to switch to a different branch or directory within the same repository. To do that you should use TortoiseSVN → Switch.... Read Seção 4.20.3, “Para Obter ou Alternar” for more information.

If you use relocate in either of the cases above, it *will corrupt your working copy* and you will get many unexplainable error messages while updating, committing, etc. Once that has happened, the only fix is a fresh checkout.

4.29. Integração com Sistemas de Rastreamento de Erros / Rastreadores de Reportes

It is very common in Software Development for changes to be related to a specific bug or issue ID. Users of bug tracking systems (issue trackers) would like to associate the changes they make in Subversion with a specific ID in their issue tracker. Most issue trackers therefore provide a pre-commit hook script which parses the log message to find the bug ID with which the commit is associated. This is somewhat error prone since it relies on the user to write the log message properly so that the pre-commit hook script can parse it correctly.

TortoiseSVN pode ajudar o usuário de duas maneiras:

1. When the user enters a log message, a well defined line including the issue number associated with the commit can be added automatically. This reduces the risk that the user enters the issue number in a way the bug tracking tools can't parse correctly.

   Or TortoiseSVN can highlight the part of the entered log message which is recognized by the issue tracker. That way the user knows that the log message can be parsed correctly.

2. When the user browses the log messages, TortoiseSVN creates a link out of each bug ID in the log message which fires up the browser to the issue mentioned.

4.29.1. Adicionando Identificadores de Reporte para as Mensagens de Auditoria

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: (Seção 4.18, “Configurações do Projeto”)

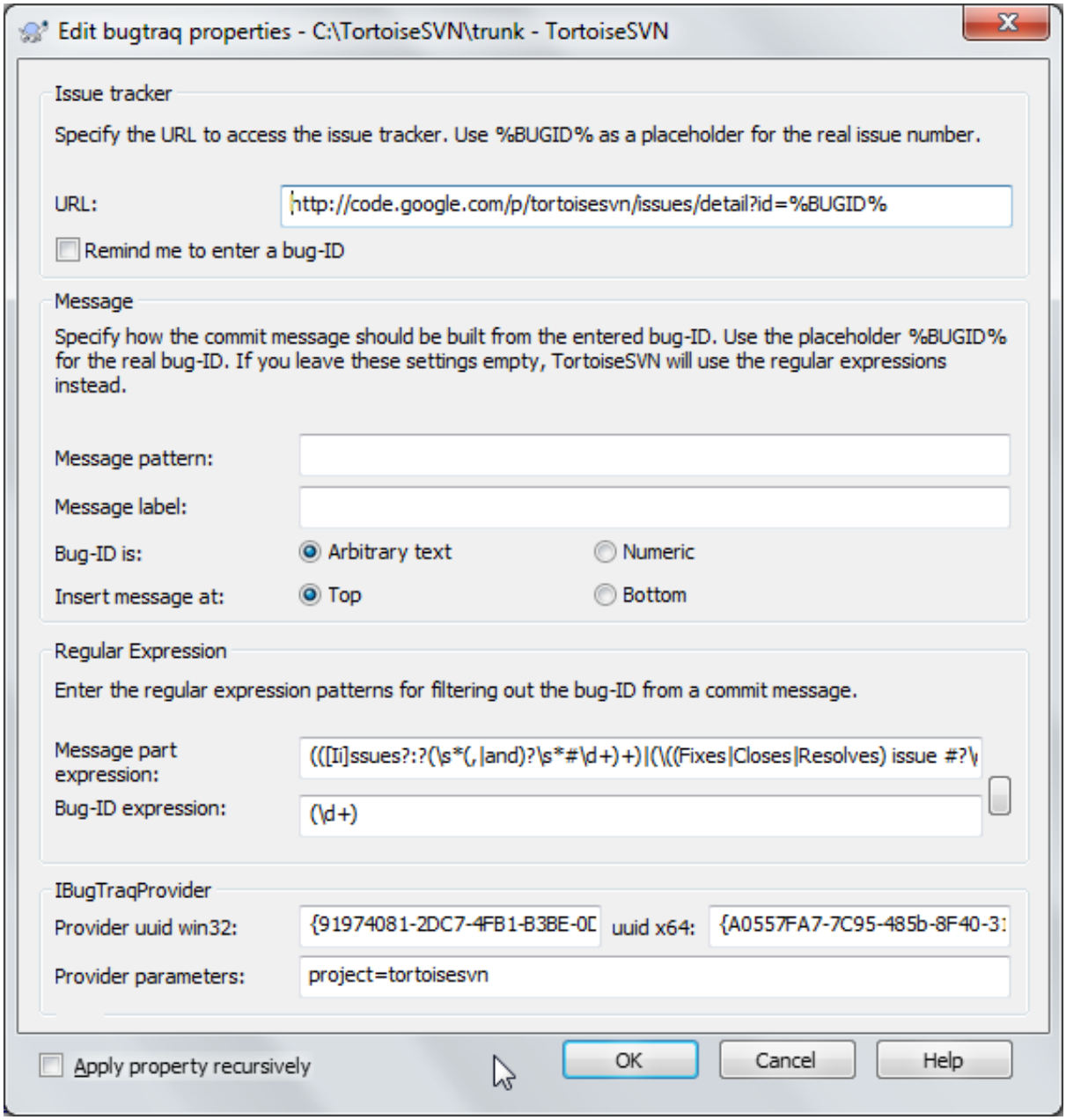


Figura 4.67. The Bugtraq Properties Dialog

When you edit any of the bugtraq properties a special property editor is used to make it easier to set appropriate values.

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

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

and not the link to it. e.g the TortoiseSVN project is using `http://issues.tortoisesvn.net/?do=details&id=%BUGID%`.

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

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

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

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

4.29.1.1. Número de Reporte em uma Caixa de Texto

In the simple approach, TortoiseSVN shows the user a separate input field where a bug ID can be entered. Then a separate line is appended/prepended to the log message the user entered.

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

bugtraq: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.*

bugtraq:append
: This property defines if the bug-ID is appended (true) to the end of the log message or inserted (false) at the start of the log message. Valid values are `true/false`. *If not defined, `true` is assumed, so that existing projects don't break.*

4.29.1.2. Números de Reporte Usando Expressões Regulares

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

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

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

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

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

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

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

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

It's not always easy to get the regex right so to help out there is a test dialog built into the bugtraq properties dialog. Click on the button to the right of the edit boxes to bring it up. Here you can enter some test text, and change each regex to see the results. If the regex is invalid the edit box background changes to red.

If both the `bugtraq:message` and `bugtraq:logregex` properties are set, `logregex` takes precedence.

Dica

Even if you don't have an issue tracker with a pre-commit hook parsing your log messages, you still can use this to turn the issues mentioned in your log messages into links!

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

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

Definir as Propriedades em um Diretório

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 (e.g. `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/`).

As of version 1.8, TortoiseSVN and Subversion use so called `inherited properties`, which means a property that is set on a folder is automatically also implicitly set on all subfolders. So there's no need to set the properties on all folders anymore but only on the root folder.

For project properties *only*, i.e. `tsvn:`, `bugtraq:` and `webviewer:` 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 to a working copy using TortoiseSVN, any project properties present in the parent folder will automatically be added to the new child folder too.

Nenhuma Informação do Rastreador de Reportes no Navegador de Repositório

Because the issue tracker integration depends upon accessing Subversion properties, you will only see the results when using a checked out working copy. Fetching properties remotely is a slow operation, so you will not see this feature in action from the repo browser unless you started the repo browser from your working copy. If you started the repo browser by entering the URL of the repository you won't see this feature.

Pela mesma razão, as propriedades do projeto não se propagarão automaticamente quando são adicionadas pastas filhas através do navegador da repo

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* [https://svn.code.sf.net/p/tortoisesvn/code/trunk/doc/notes/issuetrackers.txt] in the TortoiseSVN source repository. (Seção 3, “Licença” explains how to access the repository.)

4.29.2. Obtendo Informações do Rastreador de Reportes

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

Any such interface is of course highly specific to your issue tracker system, so we cannot provide this part, and describing how to create such a program is beyond the scope of this manual. The interface definition and sample plugins in C# and C++/ATL can be obtained from the `contrib` folder in the *TortoiseSVN repository* [https://svn.code.sf.net/p/tortoisesvn/code/trunk/contrib/issue-tracker-plugins]. (Seção 3, “Licença” explains how to access the repository.) A summary of the API is also given in Capítulo 7, *IBugtraqProvider interface*. Another (working) example plugin in C# is *Gurtle* [http://code.google.com/p/gurtle/] which implements the required COM interface to interact with the *Google Code* [http://code.google.com/hosting/] issue tracker.

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

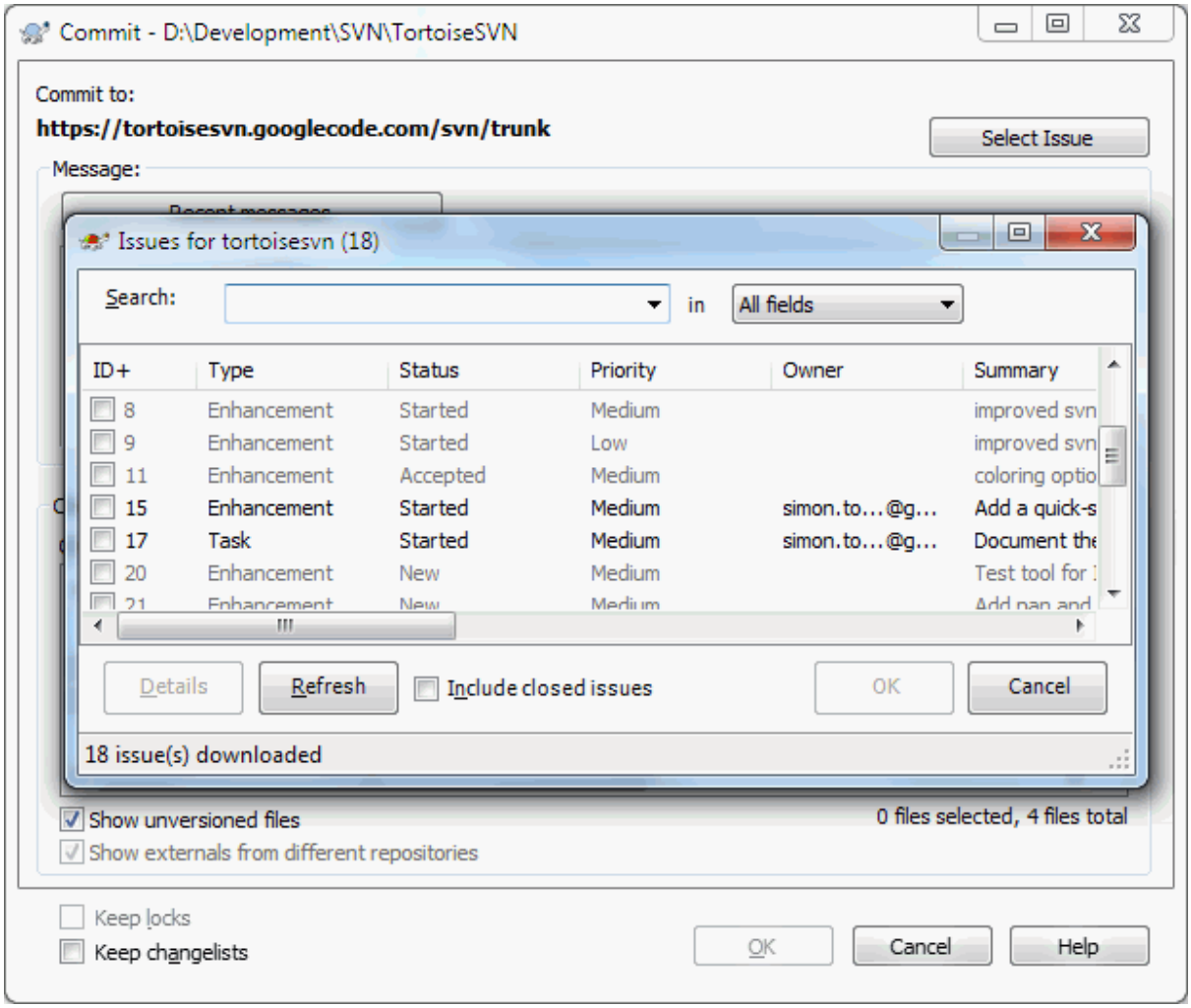


Figura 4.68. Janela de exemplo de uma consulta de um rastreador de reportes

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

4.30. Integração com visualizadores de repositórios locados na web

There are several web-based repository viewers available for use with Subversion such as *ViewVC* [http://www.viewvc.org/] and *WebSVN* [http://websvn.tigris.org/]. TortoiseSVN provides a means to link with these viewers.

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: (Seção 4.18, “Configurações do Projeto”)

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 for path in webviewer For example, if you right click in the log dialog bottom pane on a file entry `/trunk/src/file` then the `%PATH%` in the URL will be replaced with `/trunk/src/file`.

You can also use relative URLs instead of absolute ones. This is useful in case your web viewer is on the same domain/server as your source repository. In case the domain name ever changes, you don't have to adjust the `webviewer:revision` and `webviewer:pathrevision` property. The format is the same as for the `bugtraq:url` property. See Seção 4.29, “Integração com Sistemas de Rastreamento de Erros / Rastreadores de Reportes”.

Definir as Propriedades em um Diretório

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 (e.g. `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 project properties *only*, i.e. `tsvn:`, `bugtraq:` and `webviewer:` 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 to a working copy using TortoiseSVN, any project properties present in the parent folder will automatically be added to the new child folder too.

Limitações Usando o Navegador de Repositório

Because the repo viewer integration depends upon accessing Subversion properties, you will only see the results when using a checked out working copy. Fetching properties remotely is a slow operation, so you will not see this feature in action from the repo browser unless you started the repo browser from your working copy. If you started the repo browser by entering the URL of the repository you won't see this feature.

Pela mesma razão, as propriedades do projeto não se propagarão automaticamente quando são adicionadas pastas filhas através do navegador da repo

4.31. Configurações do TortoiseSVN

To find out what the different settings are for, just leave your mouse pointer a second on the editbox/checkbox... and a helpful tooltip will popup.

4.31.1. Configurações Gerais

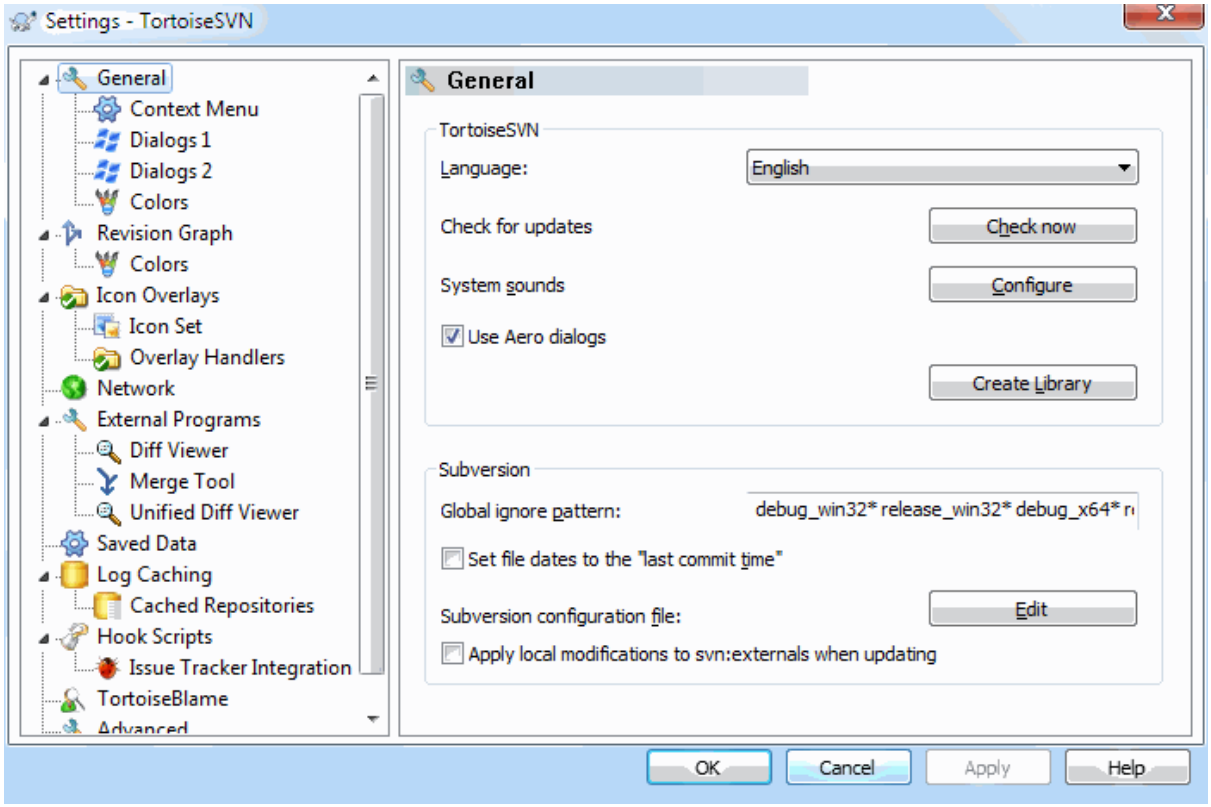


Figura 4.69. Janela de Configurações, Opção Geral

This dialog allows you to specify your preferred language, and the Subversion-specific settings.

Idioma
: Selects your user interface language. Of course, you have to install the corresponding language pack first to get another UI language than the default English one.

Verificar atualizações
: TortoiseSVN will contact its download site periodically to see if there is a newer version of the program available. If there is it will show a notification link in the commit dialog. Use Check now if you want an answer right away. The new version will not be downloaded; you simply receive an information dialog telling you that the new version is available.

Sons do sistema
: TortoiseSVN has three custom sounds which are installed by default.

  - Error
  - Notificação
  - Aviso

  You can select different sounds (or turn these sounds off completely) using the Windows Control Panel. Configure is a shortcut to the Control Panel.

Use Aero Dialogs
: On Windows Vista and later systems this controls whether dialogs use the Aero styling.

Criar Biblioteca
: On Windows 7 you can create a Library in which to group working copies which are scattered in various places on your system.

Padrão global para ignorar

Global ignore patterns are used to prevent unversioned files from showing up e.g. in the commit dialog. Files matching the patterns are also ignored by an import. Ignore files or directories by typing in the names or extensions. Patterns are separated by spaces e.g. `bin obj *.bak *.~?? *.jar *.[Tt]mp`. These patterns should not include any path separators. Note also that there is no way to differentiate between files and directories. Read Seção 4.14.1, “Padrões de Filtro na Lista de Arquivos Ignorados” for more information on the pattern-matching syntax.

Note that the ignore patterns you specify here will also affect other Subversion clients running on your PC, including the command line client.

Cuidado

If you use the Subversion configuration file to set a `global-ignores` pattern, it will override the settings you make here. The Subversion configuration file is accessed using the Edit as described below.

This ignore pattern will affect all your projects. It is not versioned, so it will not affect other users. By contrast you can also use the versioned `svn:ignore` or `svn:global-ignores` property to exclude files or directories from version control. Read Seção 4.14, “Ignorando Arquivos e Diretórios” for more information.

Altera as datas dos arquivos para “hora da última submissão”

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.

Arquivo de configuração do Subversion

Use Edit to edit the Subversion configuration file directly. Some settings cannot be modified directly by TortoiseSVN, and need to be set here instead. For more information about the Subversion `config` file see the *Runtime Configuration Area* [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html]. The section on *Automatic Property Setting* [http://svnbook.red-bean.com/en/1.8/svn.advanced.props.html#svn.advanced.props.auto] is of particular interest, and that is configured here. Note that Subversion can read configuration information from several places, and you need to know which one takes priority. Refer to *Configuration and the Windows Registry* [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] to find out more.

Apply local modifications to svn:externals when updating

This option tells TortoiseSVN to always apply local modifications to the `svn:externals` property when updating the working copy.

4.31.1.1. Configurações do Menu de Contexto

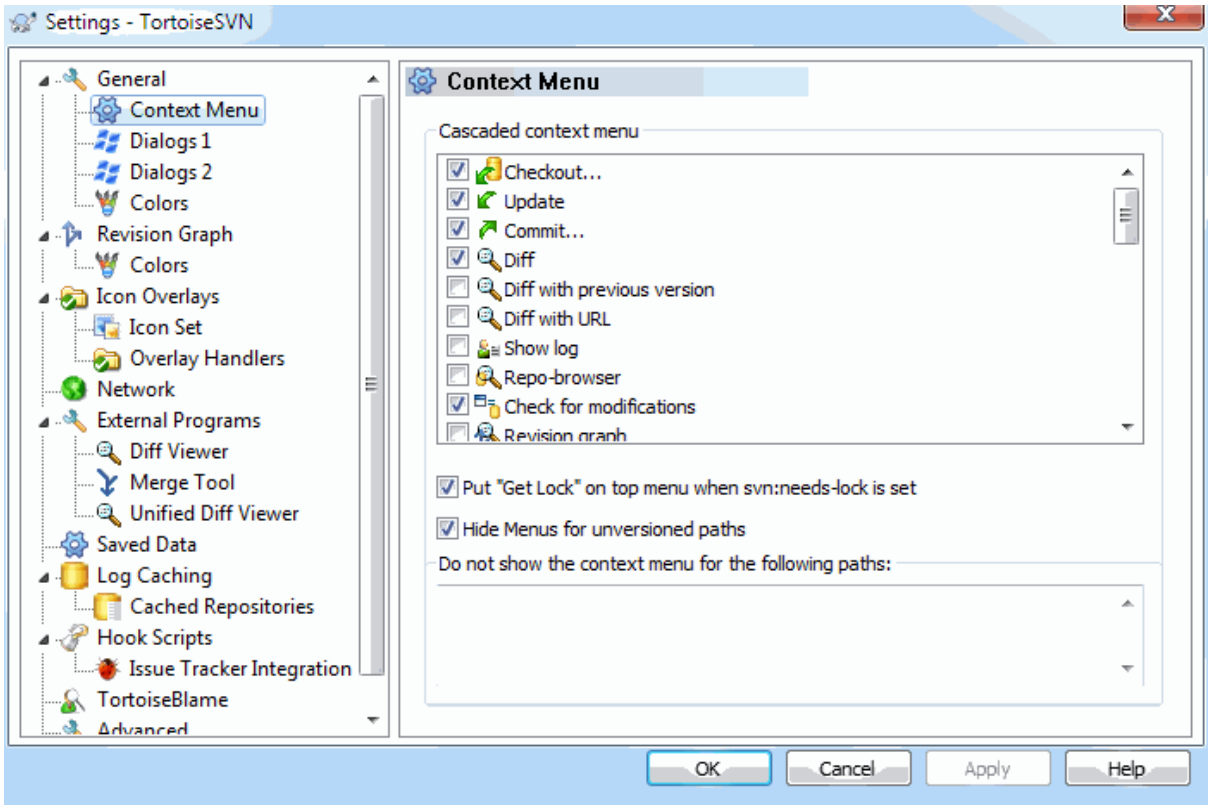


Figura 4.70. Janela de Configurações, Opção Menu de Contexto

This page allows you to specify which of the TortoiseSVN context menu entries will show up in the main context menu, and which will appear in the TortoiseSVN submenu. By default most items are unchecked and appear in the submenu.

There is a special case for Get Lock. You can of course promote it to the top level using the list above, but as most files don't need locking this just adds clutter. However, a file with the `svn:needs-lock` property needs this action every time it is edited, so in that case it is very useful to have at the top level. Checking the box here means that when a file is selected which has the `svn:needs-lock` property set, Get Lock will always appear at the top level.

Most of the time, you won't need the TortoiseSVN context menu, apart for folders that are under version control by Subversion. For non- versioned folders, you only really need the context menu when you want to do a checkout. If you check the option `Hide menus for unversioned paths` , TortoiseSVN will not add its entries to the context menu for unversioned folders. But the entries are added for all items and paths in a versioned folder. And you can get the entries back for unversioned folders by holding the **Shift** key down while showing the context menu.

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

4.31.1.2. Janela 1 de Configurações do TortoiseSVN

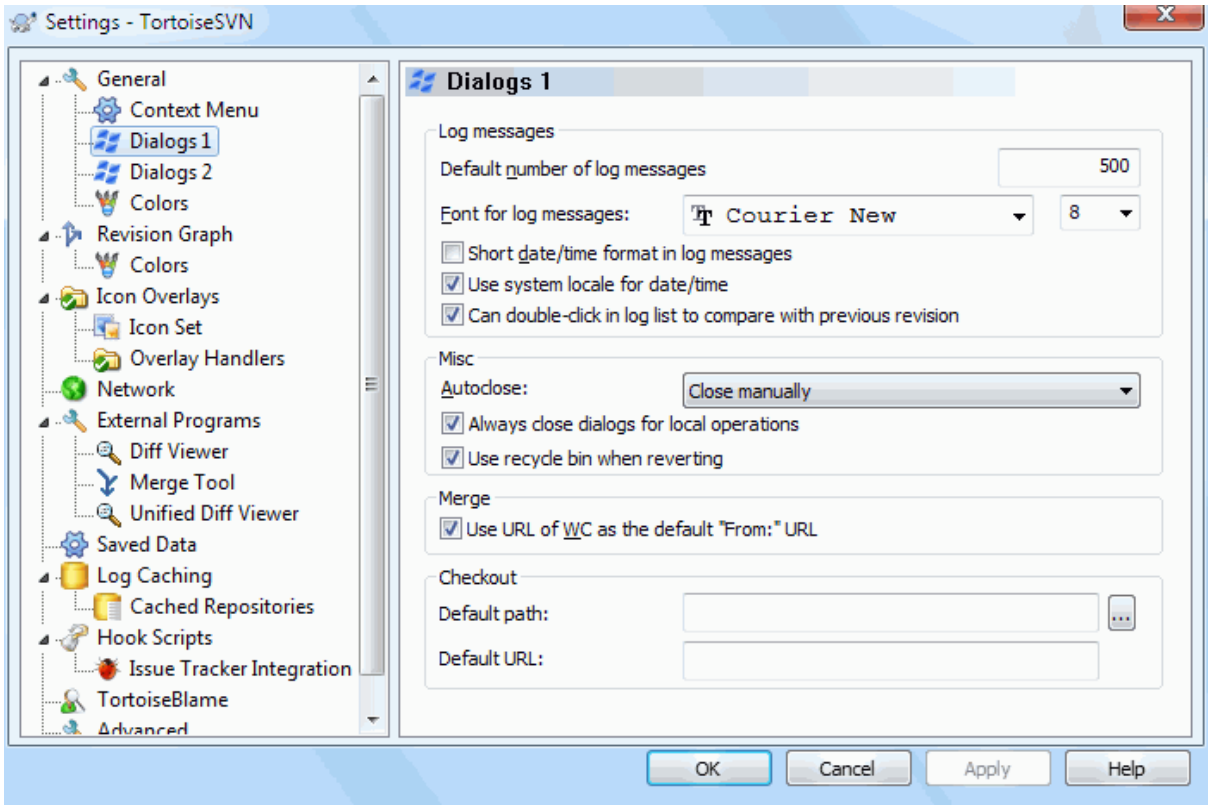


Figura 4.71. Janela de Configurações, Opção Janelas 1

This dialog allows you to configure some of TortoiseSVN's dialogs the way you like them.

Número padrão de mensagens de log

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.

Fonte para mensagens de log

Selects the font face and size used to display the log message itself in the middle pane of the Revision Log dialog, and when composing log messages in the Commit dialog.

Formato curto de data / hora no histórico

If the standard long messages use up too much space on your screen use the short format.

Duplo-clique na lista de comentários para comparar com a revisão anterior

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

Fechamento Automático

TortoiseSVN can automatically close all progress dialogs when the action is finished without error. This setting allows you to select the conditions for closing the dialogs. The default (recommended) setting is Close manually which allows you to review all messages and check what has happened. However, you may decide that you want to ignore some types of message and have the dialog close automatically if there are no critical changes.

Auto-close if no merges, adds or deletes means that the progress dialog will close if there were simple updates, but if changes from the repository were merged with yours, or if any files were added or deleted, the dialog will remain open. It will also stay open if there were any conflicts or errors during the operation.

Auto-close if no conflicts relaxes the criteria further and will close the dialog even if there were merges, adds or deletes. However, if there were any conflicts or errors, the dialog remains open.

Auto-close if no errors always closes the dialog even if there were conflicts. The only condition that keeps the dialog open is an error condition, which occurs when Subversion is unable to complete the task. For example, an update fails because the server is inaccessible, or a commit fails because the working copy is out-of-date.

Sempre fechar as janelas para operações locais
: Local operations like adding files or reverting changes do not need to contact the repository and complete quickly, so the progress dialog is often of little interest. Select this option if you want the progress dialog to close automatically after these operations, unless there are errors.

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

Use URL of WC as the default “From:” URL
: In the merge dialog, the default behaviour is for the From: URL to be remembered between merges. However, some people like to perform merges from many different points in their hierarchy, and find it easier to start out with the URL of the current working copy. This can then be edited to refer to a parallel path on another branch.

Caminho padrão de envio:
: You can specify the default path for checkouts. If you keep all your checkouts in one place, it is useful to have the drive and folder pre-filled so you only have to add the new folder name to the end.

Default checkout URL
: You can also specify the default URL for checkouts. If you often checkout sub-projects of some very large project, it can be useful to have the URL pre-filled so you only have to add the sub-project name to the end.

4.31.1.3. Janela 2 de Configurações do TortoiseSVN

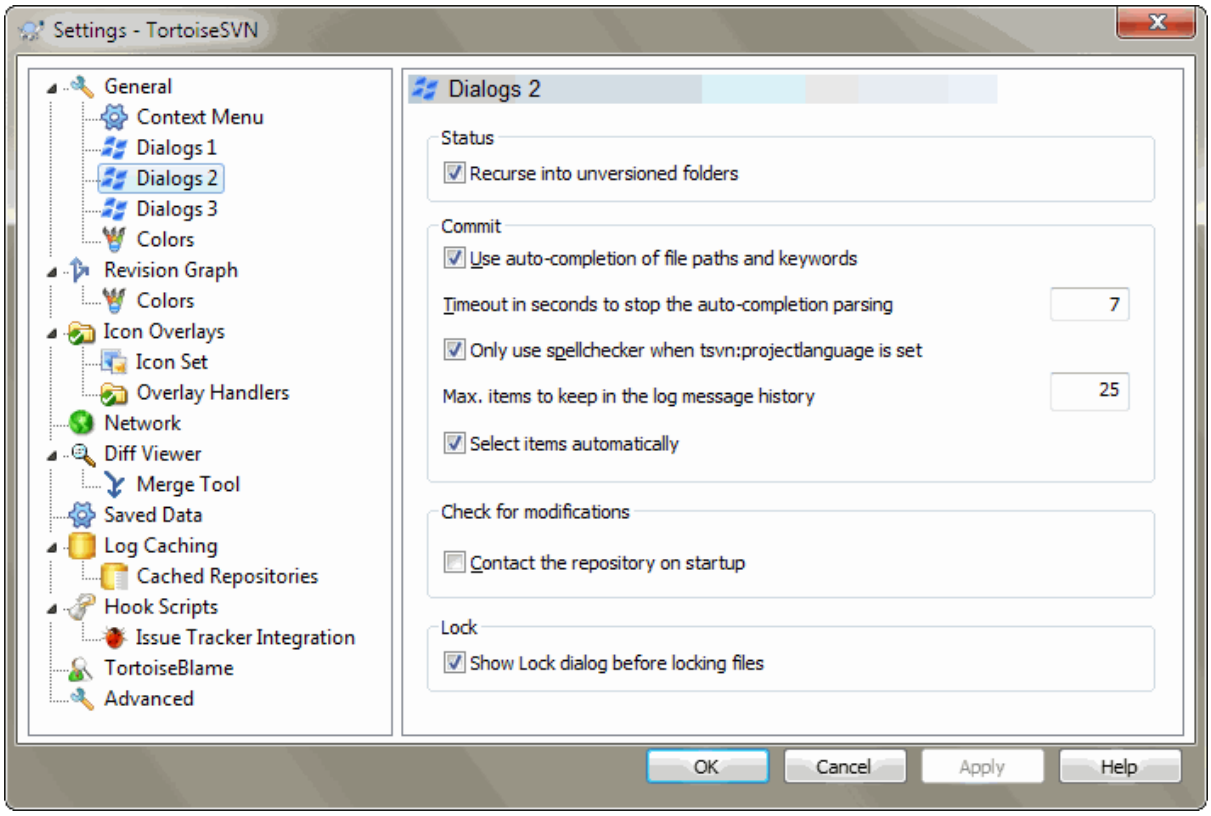


Figura 4.72. Janela de Configurações, Opção Janelas 2

Usar recursividade nas pastas não versionadas
: If this box is checked (default state), then whenever the status of an unversioned folder is shown in the Add, Commit or Check for Modifications dialog, every child file and folder is also shown. If you uncheck this box, only the unversioned parent is shown. Unchecking reduces clutter in these dialogs. In that case if you select an unversioned folder for Add, it is added recursively.

  In the Check for Modifications dialog you can opt to see ignored items. If this box is checked then whenever an ignored folder is found, all child items will be shown as well.

Usar autocompletar para caminhos de arquivo e palavras-chave
: 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.

Tempo limite, em segundos, para parar a interpretação do autocompletar
: 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
: If you don't wish to use the spellchecker for all commits, check this box. The spellchecker will still be enabled where the project properties require it.

Máximo de itens no histórico de mensagens de log
: When you type in a log message in the commit dialog, TortoiseSVN stores it for possible re-use later. By default it will keep the last 25 log messages for each repository, but you can customize that number here. If you have many different repositories, you may wish to reduce this to avoid filling your registry.

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

Selecionar itens automaticamente
: 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.

Abrir novamente a janela depois de submeter se houver itens não submetidos
: This reopens the commit dialog automatically at the same directory after a successful commit. The dialog is reopened only if there still are items left to commit.

Conectar ao repositório ao inicializar
: The Check for Modifications dialog checks the working copy by default, and only contacts the repository when you click Check repository. If you always want to check the repository, you can use this setting to make that action happen automatically.

Mostrar janela de Bloqueio antes de bloquear arquivos
: When you select one or more files and then use TortoiseSVN → Lock to take out a lock on those files, on some projects it is customary to write a lock message explaining why you have locked the files. If you do not use lock messages, you can uncheck this box to skip that dialog and lock the files immediately.

  If you use the lock command on a folder, you are always presented with the lock dialog as that also gives you the option to select files for locking.

  If your project is using the `tsvn:lockmsgminsize` property, you will see the lock dialog regardless of this setting because the project *requires* lock messages.

4.31.1.4. Janela 2 de Configurações do TortoiseSVN

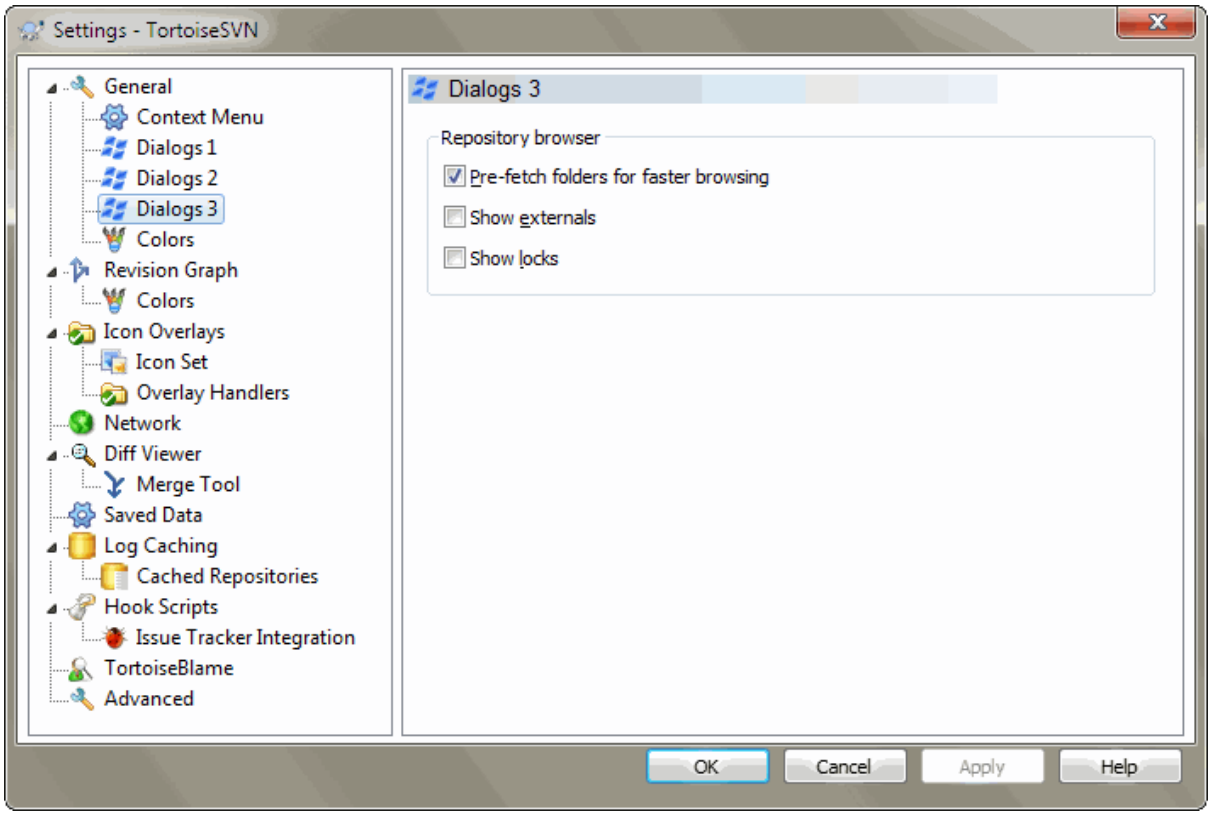


Figura 4.73. Janela de Configurações, Opção Janela 3

Pré-carga dos diretórios por uma navegação mais rápida
: If this box is checked (default state), then the repository browser fetches information about shown folders in the background. That way as soon as you browse into one of those folders, the information is already available.

  Some servers however can't handle the multiple requests this causes or when not configured correctly treat so many requests as something bad and start blocking them. In this case you can disable the pre-fetching here.

Mostrar externos
: If this box is checked (default state), then the repository browser shows files and folders that are included with the `svn:externals` property as normal files and folders, but with an overlay icon to mark them as from an external source.

  As with the pre-fetch feature explained above, this too can put too much stress on weak servers. In this case you can disable this feature here.

4.31.1.5. Configurações de Cores do TortoiseSVN

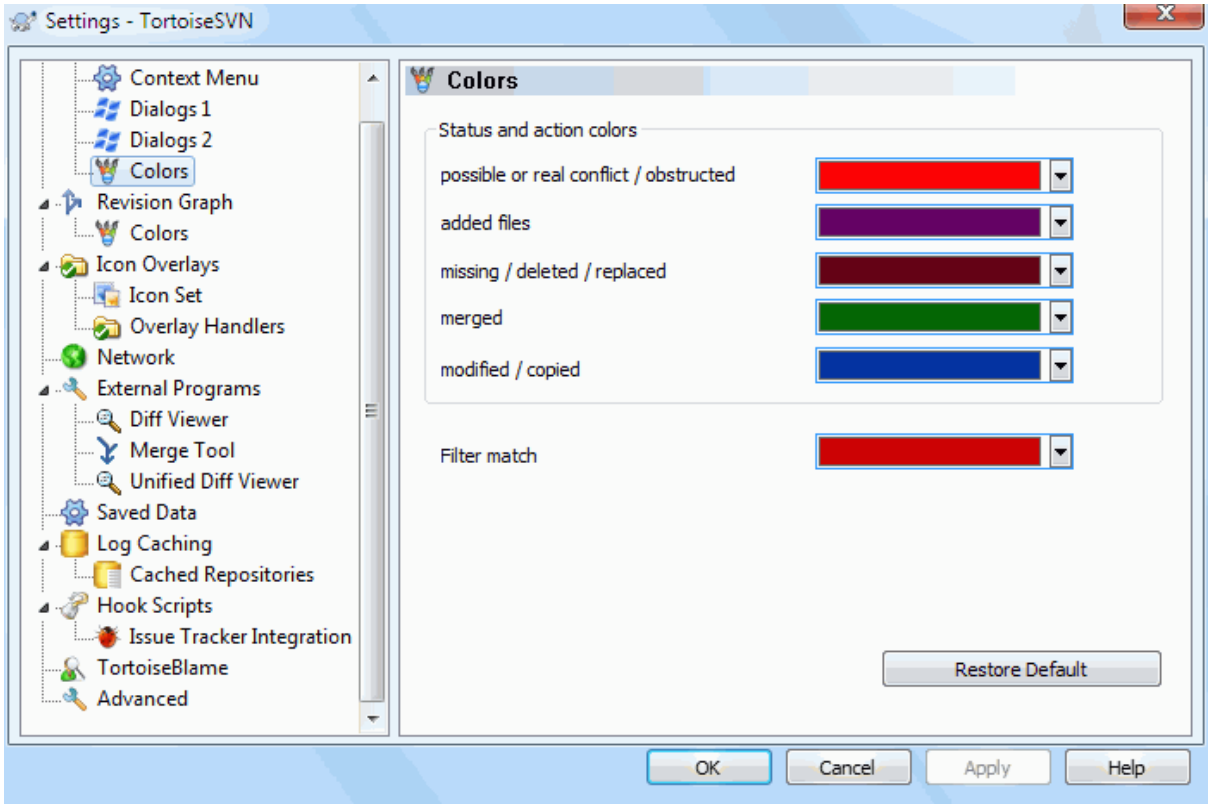


Figura 4.74. Janela de Configurações, Opção Cores

Esta janela permite configurar as cores do texto nas janelas do TortoiseSVN de acordo com a sua preferência.

Possível ou real conflito / obstrução
: A conflict has occurred during update, or may occur during merge. Update is obstructed by an existing unversioned file/folder of the same name as a versioned one.

  Esta cor é também usada para mensagens de erro em janelas de progresso.

Arquivos adicionados
: Itens adicionados ao repositório.

Perdido / apagada / substituído
: Items deleted from the repository, missing from the working copy, or deleted from the working copy and replaced with another file of the same name.

Combinado
: Changes from the repository successfully merged into the WC without creating any conflicts.

Modificado / copiado
: Add with history, or paths copied in the repository. Also used in the log dialog for entries which include copied items.

Nó excluído
: Um item que foi apagado do repositório.

Nó adicionado
: Um item que foi adicionado ao repositório, por ação de adição, cópia ou movimentação.

Nó renomeado
: Um item que foi renomeado dentro do repositório.

Nó substituido
: O item original foi apagado e um novo item com o mesmo nome o substituiu.

Correspondência de filtro
: Quando houver um filtro em uso na janela de auditoria, os termos de busca são destacados nos resultados usando esta cor.

4.31.2. Configurações do Gráfico de Revisões

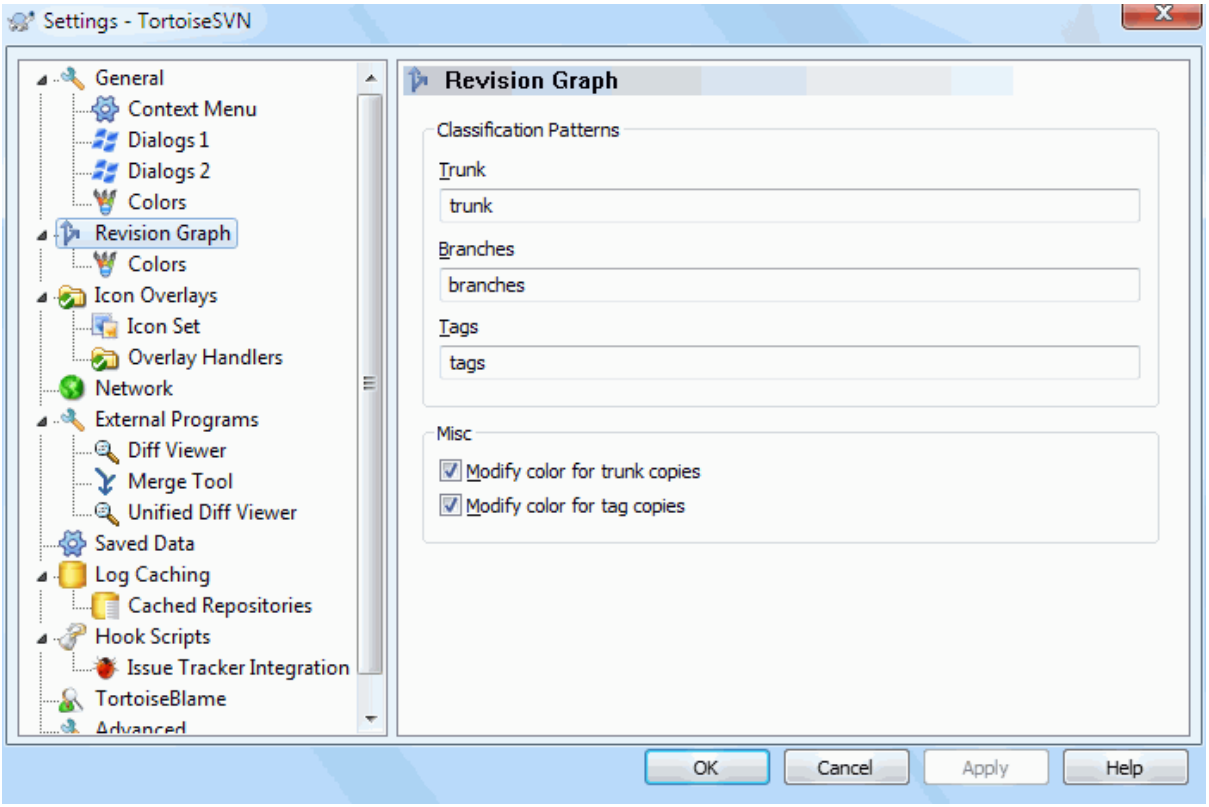


Figura 4.75. Janela de Configurações, Opção Gráfico de Revisões

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

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

Commit tag detection

Please note that these patterns are also used to detect commits to a tag, not just for the revision graph.

Cores Modificadas
: Colors are used in the revision graph to indicate the node type, i.e. whether a node is added, deleted, renamed. In order to help pick out node classifications, you can allow the revision graph to blend colors to give an

indication of both node type and classification. If the box is checked, blending is used. If the box is unchecked, color is used to indicate node type only. Use the color selection dialog to allocate the specific colors used.

4.31.2.1. Cores do Gráfico de Revisões

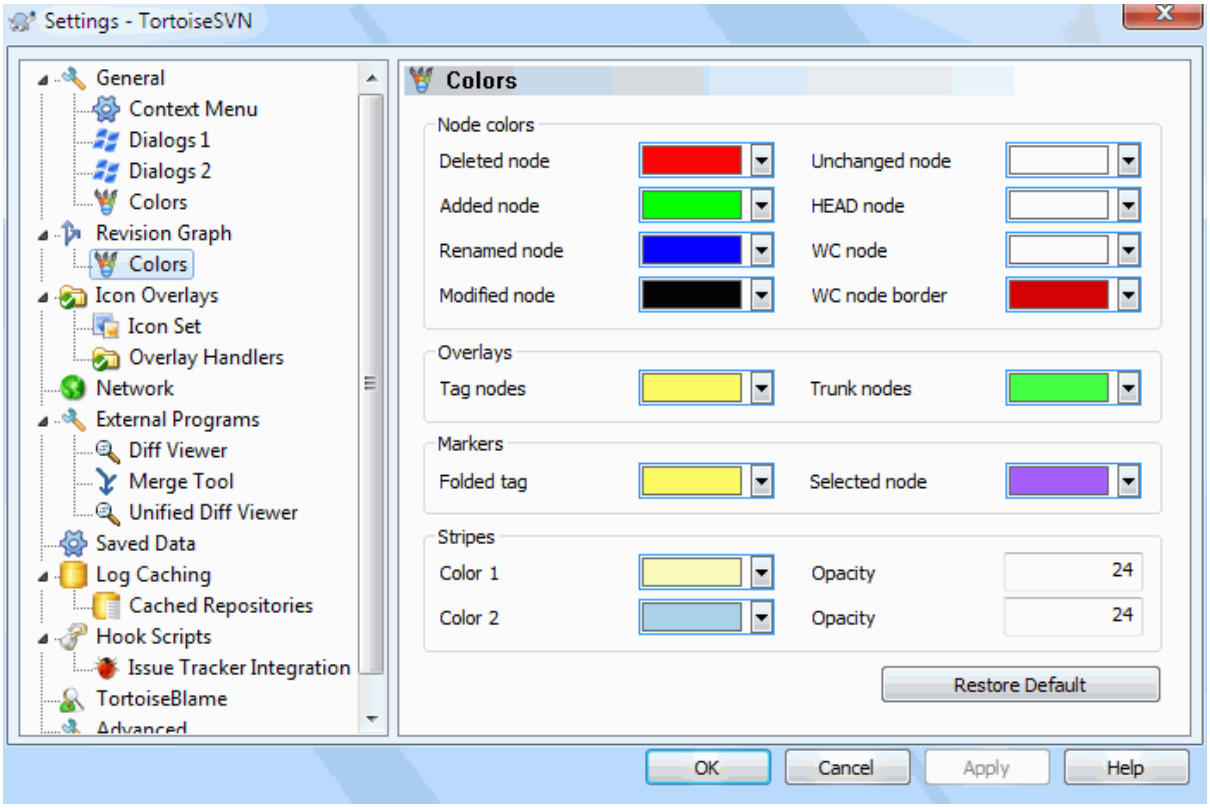


Figura 4.76. Janela de Configurações, Opção Cores do Gráfico de Revisões

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

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

Nó Adicionado
: Itens recém-adicionado, ou copiados (adicionar com histórico).

Nó Renomeado
: Itens excluídos de um local e acrescentou em outro na mesma revisão.

Nó Modificado
: Modificações simples, sem qualquer adição ou exclusão.

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

Nó Principal
: Atual última revisão no repositório

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

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

Nós de rótulos
: Nós classificados como etiquetas podem ser misturados com essa cor

Nós do tronco
: Nodes classificados como tronco podem ser misturados com esta cor.

Marcador dobrado
: If you use tag folding to save space, tags are marked on the copy source using a block in this color.

Marcadores de Nós Selecionados
: When you left click on a node to select it, the marker used to indicate selection is a block in this color.

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

4.31.3. Configurações de Sobreposição dos Ícones

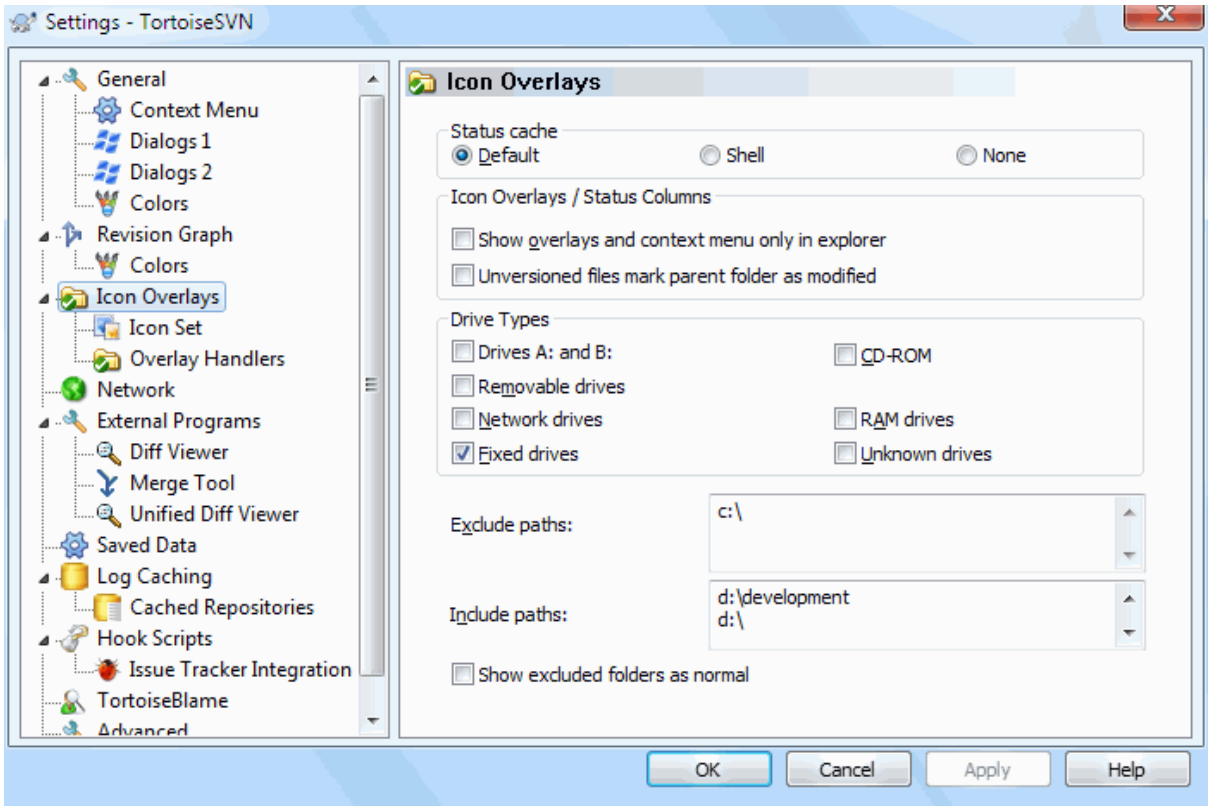


Figura 4.77. Janela de Configurações, Opção Sobreposição de Ícones

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

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:

Padrão
: 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.

Disadvantage: the process runs constantly, even if you're not working on your projects. It also uses around 10-50 MB of RAM depending on number and size of your working copies.

Shell
: Caching is done directly inside the shell extension dll, but only for the currently visible folder. Each time you navigate to another folder, the status information is fetched again.

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

  Disadvantage: Since only one folder is cached, the overlays don't show the status recursively. For big working copies, it can take more time to show a folder in explorer than with the default cache. Also the mime-type column is not available.

Nenhum
: With this setting, the TortoiseSVN does not fetch the status at all in Explorer. Because of that, files don't get an overlay and folders only get a 'normal' overlay if they're versioned. No other overlays are shown, and no extra columns are available either.

  Advantage: uses absolutely no additional memory and does not slow down the Explorer at all while browsing.

  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.

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.

You can force the status cache to *None* for elevated processes by checking the Disable status cache for elevated processes box. This is useful if you want to prevent another `TSVNCache.exe` process getting created with elevated privileges.

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).

If you have files in the `ignore-on-commit` changelist, you can chose to make those files not propagate their status to the parent folder. That way if only files in that changelist are modified, the parent folder still shows the unmodified overlay icon.

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 Flash drives appear to be a special case in that the drive type is identified by the device itself. Some appear as fixed drives, and some as removable drives.

The Exclude Paths are used to tell TortoiseSVN those paths for which it should *not* show icon overlays and status columns. This is useful if you have some very big working copies containing only libraries which you won't change at all and therefore don't need the overlays, or if you only want TortoiseSVN to look in specific folders.

Any path you specify here is assumed to apply recursively, so none of the child folders will show overlays either. If you want to exclude *only* the named folder, append `?` after the path.

The same applies to the Include Paths. Except that for those paths the overlays are shown even if the overlays are disabled for that specific drive type, or by an exclude path specified above.

Users sometimes ask how these three settings interact. For any given path check the include and exclude lists, seeking upwards through the directory structure until a match is found. When the first match is found, obey that include or exclude rule. If there is a conflict, a single directory spec takes precedence over a recursive spec, then inclusion takes precedence over exclusion.

An example will help here:

```
Exclude:
C:
C:\develop\?
C:\develop\tsvn\obj
C:\develop\tsvn\bin

Include:
C:\develop
```

These settings disable icon overlays for the C: drive, except for `c:\develop`. All projects below that directory will show overlays, except the `c:\develop` folder itself, which is specifically ignored. The high-churn binary folders are also excluded.

TSVNCache.exe also uses these paths to restrict its scanning. If you want it to look only in particular folders, disable all drive types and include only the folders you specifically want to be scanned.

Exclude SUBST Drives

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

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

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

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

Sometimes you will exclude areas that contain working copies, which saves TSVNCache from scanning and monitoring for changes, but you still want a visual indication that a folder contains a working copy. The Show excluded root folders as 'normal' checkbox allows you to do this. With this option, working copy root folders in any excluded area (drive type not checked, or specifically excluded) will show up as normal and up-to-date, with a green check mark. This reminds you that you are looking at a working copy, even though the folder overlays may not be correct. Files do not get an overlay at all. Note that the context menus still work, even though the overlays are not shown.

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

4.31.3.1. Seleção de conjuntos de ícones

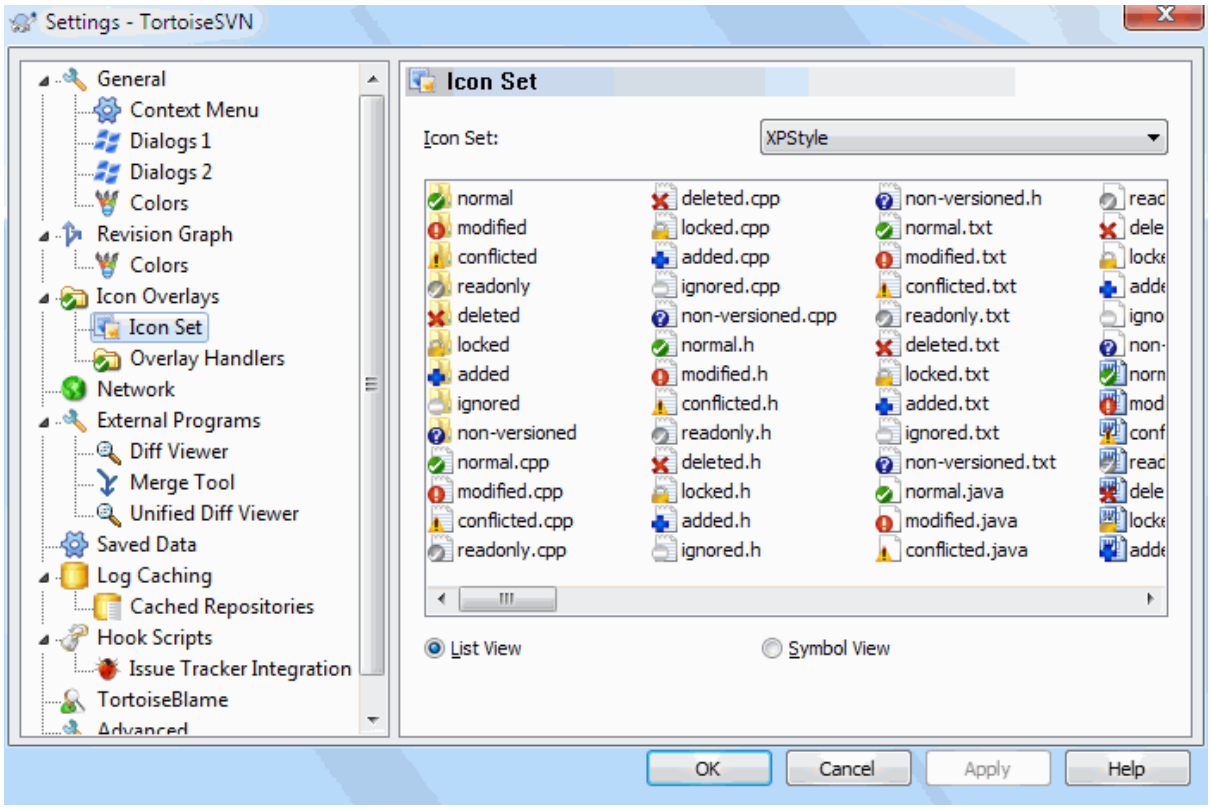


Figura 4.78. Janela de Configurações, Opção Definição de Ícones

You can change the overlay icon set to the one you like best. Note that if you change overlay set, you may have to restart your computer for the changes to take effect.

4.31.3.2. Habilitar Controladores de Sobreposição

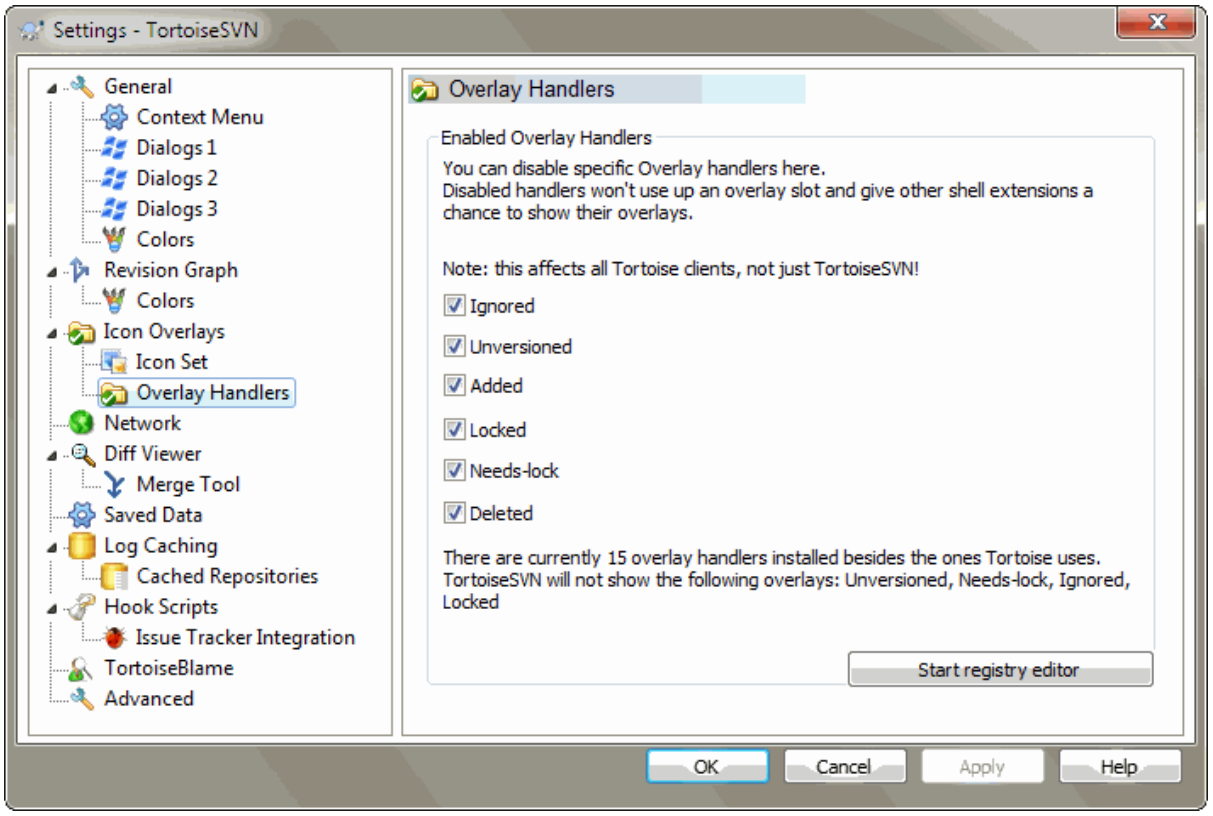


Figura 4.79. Janela de Configurações, Opção Manipuladores de Ícones

Because the number of overlays available is severely restricted, you can choose to disable some handlers to ensure that the ones you want will be loaded. Because TortoiseSVN uses the common TortoiseOverlays component which is shared with other Tortoise clients (e.g. TortoiseCVS, TortoiseHg) this setting will affect those clients too.

4.31.4. Configurações de rede

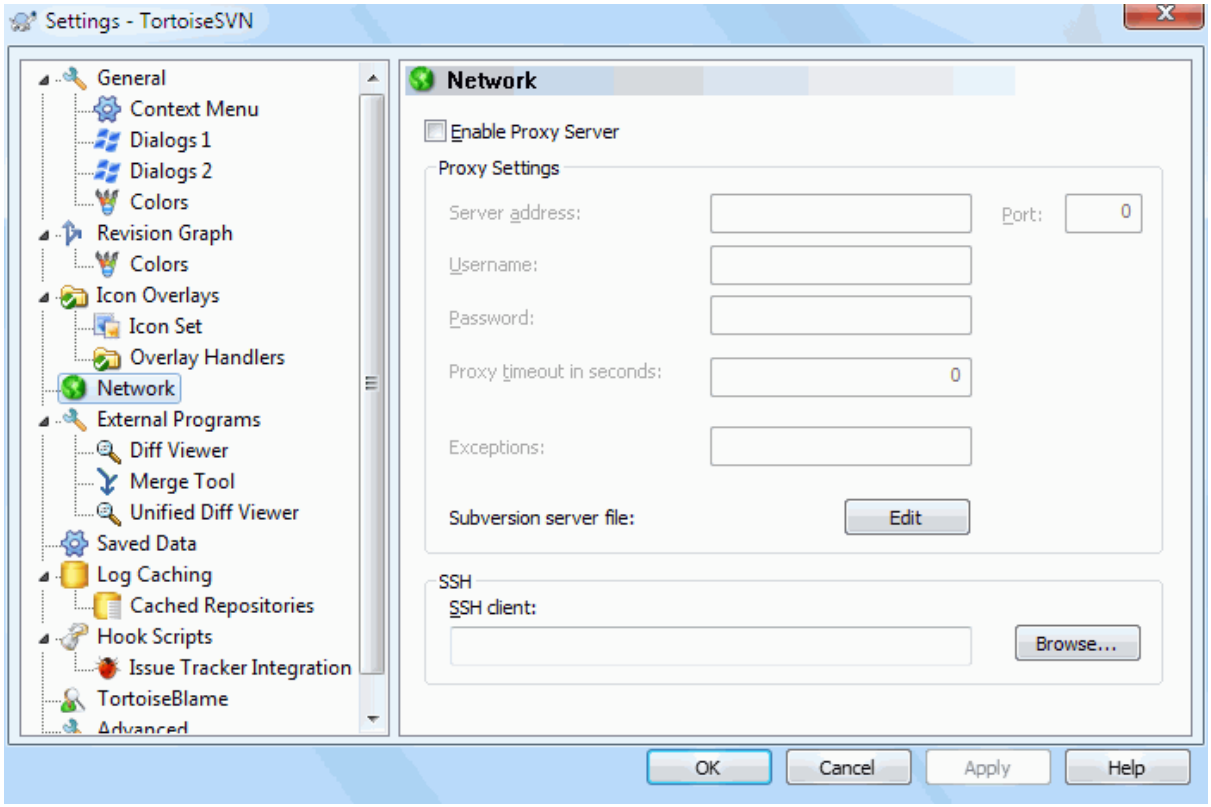


Figura 4.80. Janela de Configurações, Opção Rede

Here you can configure your proxy server, if you need one to get through your company's firewall.

If you need to set up per-repository proxy settings, you will need to use the Subversion `servers` file to configure this. Use Edit to get there directly. Consult the *Runtime Configuration Area* [http://svnbook.red-bean.com/en/1.8/svn.advanced.confarea.html] for details on how to use this file.

You can also specify which program TortoiseSVN should use to establish a secure connection to a svn+ssh repository. We recommend that you use TortoisePlink.exe. This is a version of the popular Plink program, and is included with TortoiseSVN, but it is compiled as a Windowless app, so you don't get a DOS box popping up every time you authenticate.

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

```
"C:\Program Files\TortoiseSVN\bin\TortoisePlink.exe"
```

One side-effect of not having a window is that there is nowhere for any error messages to go, so if authentication fails you will simply get a message saying something like “Unable to write to standard output”. For this reason we recommend that you first set up using standard Plink. When everything is working, you can use TortoisePlink with exactly the same parameters.

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* [https://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* [https://tortoisesvn.net/ssh_howto.html].

4.31.5. Configuração de Programa Externo

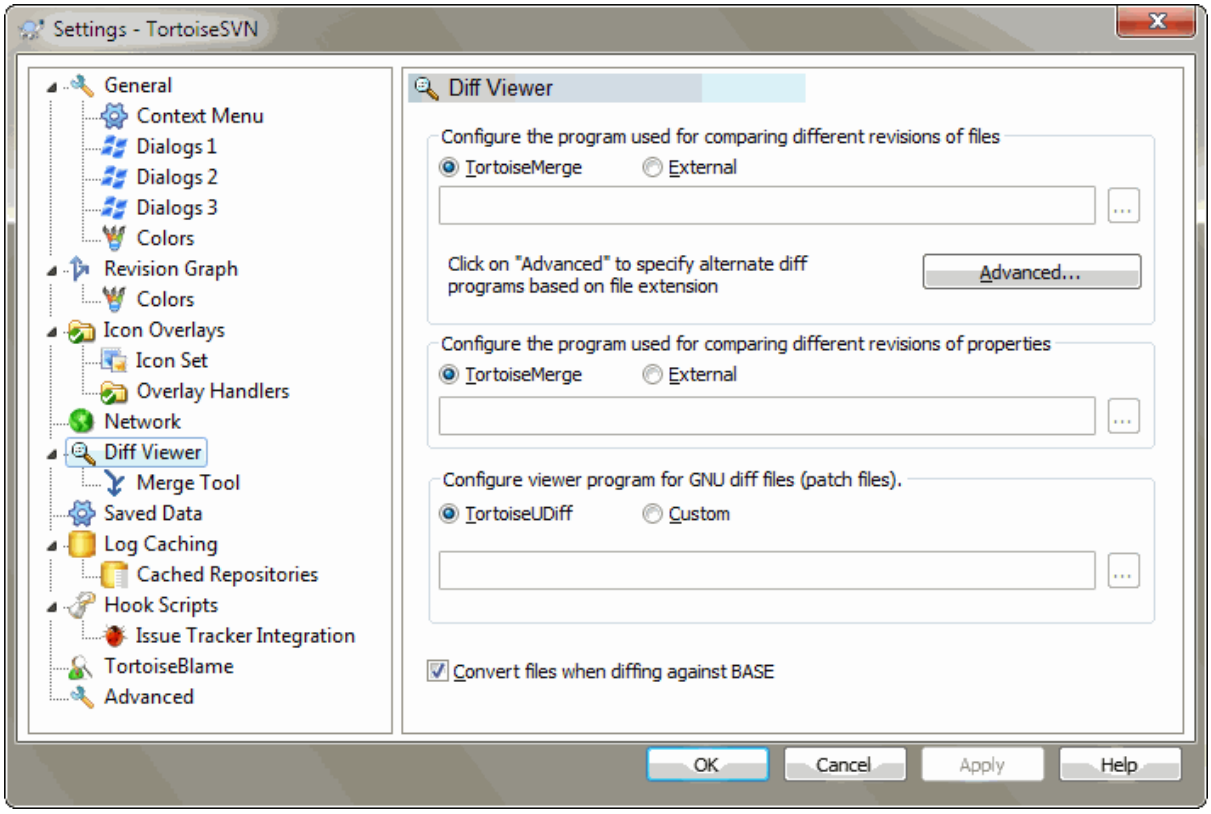


Figura 4.81. Janela de Configurações, Opção Visualizador de Diferenças

Here you can define your own diff/merge programs that TortoiseSVN should use. The default setting is to use TortoiseMerge which is installed alongside TortoiseSVN.

Read Seção 4.11.6, “Ferramentas de Diferenciação/Combinação Externas” for a list of some of the external diff/merge programs that people are using with TortoiseSVN.

4.31.5.1. Visualizador de Diferenças

An external diff program may be used for comparing different revisions of files. The external program will need to obtain the filenames from the command line, along with any other command line options. TortoiseSVN uses substitution parameters prefixed with %. When it encounters one of these it will substitute the appropriate value. The order of the parameters will depend on the Diff program you use.

%base
: O arquivo original sem alterações.

%bname
: O título da janela para o arquivo de base

%nqbname
: The window title for the base file, without quotes

%mine
: Seu próprio arquivo com suas mudanças

%yname
O título da janela para o seu arquivo

%nqyname
The window title for your file, without quotes

%burl
A URL do arquivo original, se disponível

%nqburl
A URL do arquivo original, se disponível, sem citações

%yurl
A URL do segundo arquivo, se disponível

%nqyurl
The URL of the second file, if available, without quotes

%brev
A revisão do arquivo original, se disponível

%nqbrev
A revisão do arquivo original, se diponivel, sem citações

%yrev
A revisão do segundo arquivo, se disponível

%nqyrev
A revisão do segundo arquivo, se disponivel, sem citações

%peg
A revisão marcada, se disponível

%nqpeg
The peg revision, if available, without quotes

%fname
O nome do arquivo. Esta é uma string vazia se dois diferentes arquivos são diffed ao invés de dois estados do mesmo arquivo.

%nqfname
O nome do arquivo, sem citação.

The window titles are not pure filenames. TortoiseSVN treats that as a name to display and creates the names accordingly. So e.g. if you're doing a diff from a file in revision 123 with a file in your working copy, the names will be `filename : revision 123` and `filename : working copy`.

For example, with ExamDiff Pro:

```
C:\Path-To\ExamDiff.exe %base %mine --left_display_name:%bname
                                    --right_display_name:%yname
```

or with KDiff3:

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

or with WinMerge:

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

or with Araxis:

```
C:\Path-To\compare.exe /max /wait /title1:%bname /title2:%yname
                       %base %mine
```

or with UltraCompare:

```
C:\Path-To\uc.exe %base %mine -title1 %bname -title2 %yname
```

or with DiffMerge:

```
C:\Path-To\DiffMerge.exe -nosplash -t1=%bname -t2=%yname %base %mine
```

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.

You can also specify a different diff tool to use on Subversion properties. Since these tend to be short simple text strings, you may want to use a simpler more compact viewer.

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.

At the bottom of the dialog you can configure a viewer program for unified-diff files (patch files). No parameters are required. The Default setting is to use TortoiseUDiff which is installed alongside TortoiseSVN, and colour-codes the added and removed lines.

Since Unified Diff is just a text format, you can use your favourite text editor if you prefer.

4.31.5.2. Ferramente de Unificação

Um programa externo de mesclagem usado para resolver arquivos conflitantes. É utilizado substituição de parâmetro do mesmo modo como seria com um problema comparador.

%base
: o arquivo original sem as suas ou quaisquer mudanças

%bname
: O título da janela para o arquivo de base

%nqbname
: The window title for the base file, without quotes

%mine
: Seu próprio arquivo com suas mudanças.

%yname
: O título da janela para o seu arquivo

%nqyname
: The window title for your file, without quotes

%theirs
: o arquivo como ele é no repositório

%tname
: O título da janela para o arquivo no repositório

%nqtname
: O título da janela para o arquivo dentro do repositório, sem citações

%merged
: O arquivo conflitante, o resultado da operação de mesclagem.

%mname
: O título da janela para o arquivo mesclado

%nqmname
: A janela do título do arquivo fundido, sem citações

%fname
: O nome dos arquivos conflitantes.

%nqfname
: O nome dos arquivos conflitantes, sem citação.

For example, with Perforce Merge:

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

or with KDiff3:

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

or with Araxis:

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

or with WinMerge (2.8 or later):

```
C:\Path-To\WinMerge.exe %merged
```

or with DiffMerge:

```
C:\Path-To\DiffMerge.exe -caption=%mname -result=%merged -merge
      -nosplash -t1=%yname -t2=%bname -t3=%tname %mine %base %theirs
```

4.31.5.3. Configurações Avançadas Comparação/Combinação

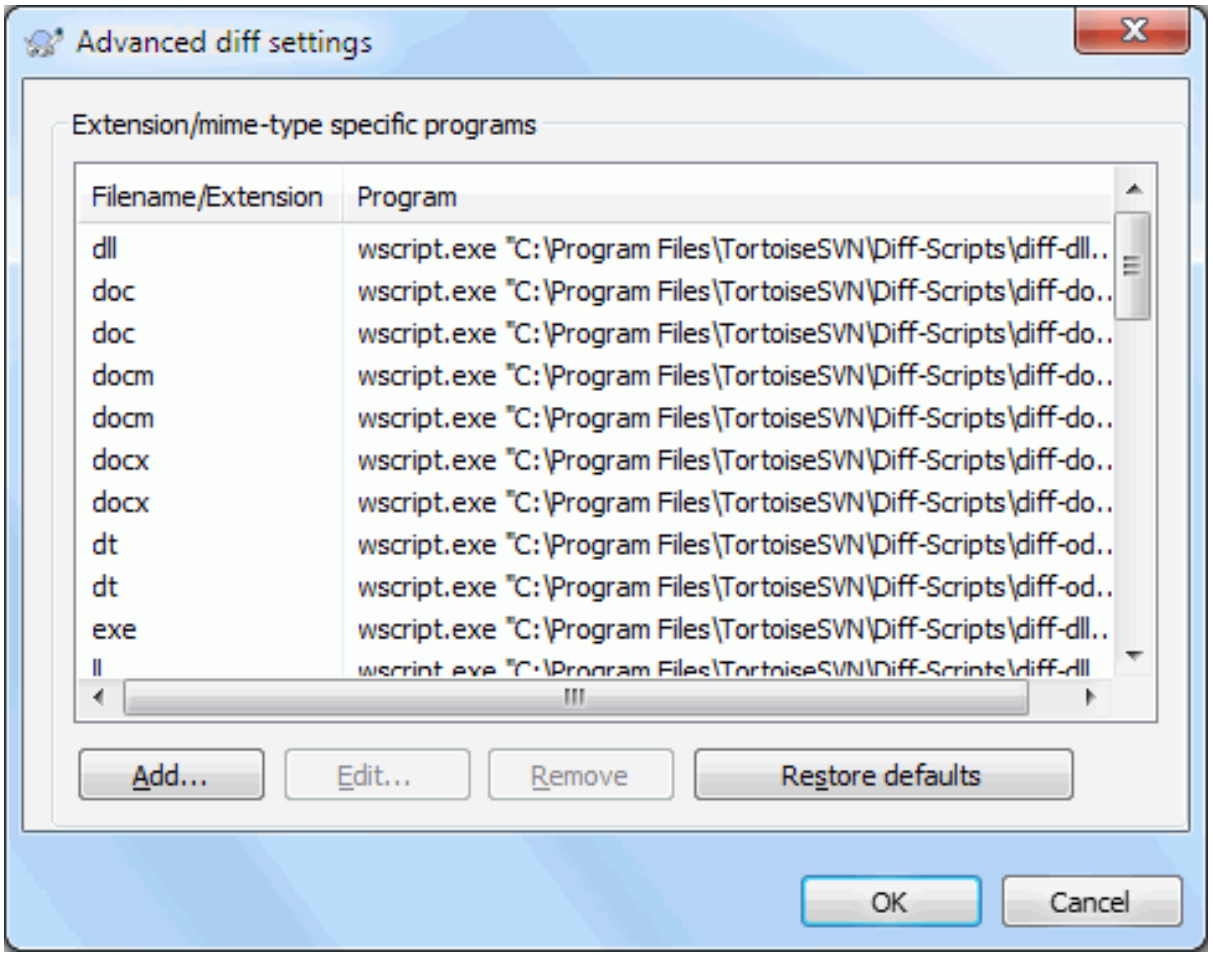


Figura 4.82. Janela de Configurações, Opção Diferenciação/Combinação Avançado

Nas configurações avançadas, é possível definir um comparador e um mesclador diferente para cada extensão de arquivo. Para cada instancia pode-se associar o Photoshop como o programa “Comparador” para arquivos `.jpg` :-) É possível também associar a propriedade `svn:mime-type` como programa comparador ou de mesclagem.

To associate using a file extension, you need to specify the extension. Use `.bmp` to describe Windows bitmap files. To associate using the `svn:mime-type` property, specify the mime type, including a slash, for example `text/xml`.

4.31.6. Dados de configuração salvos

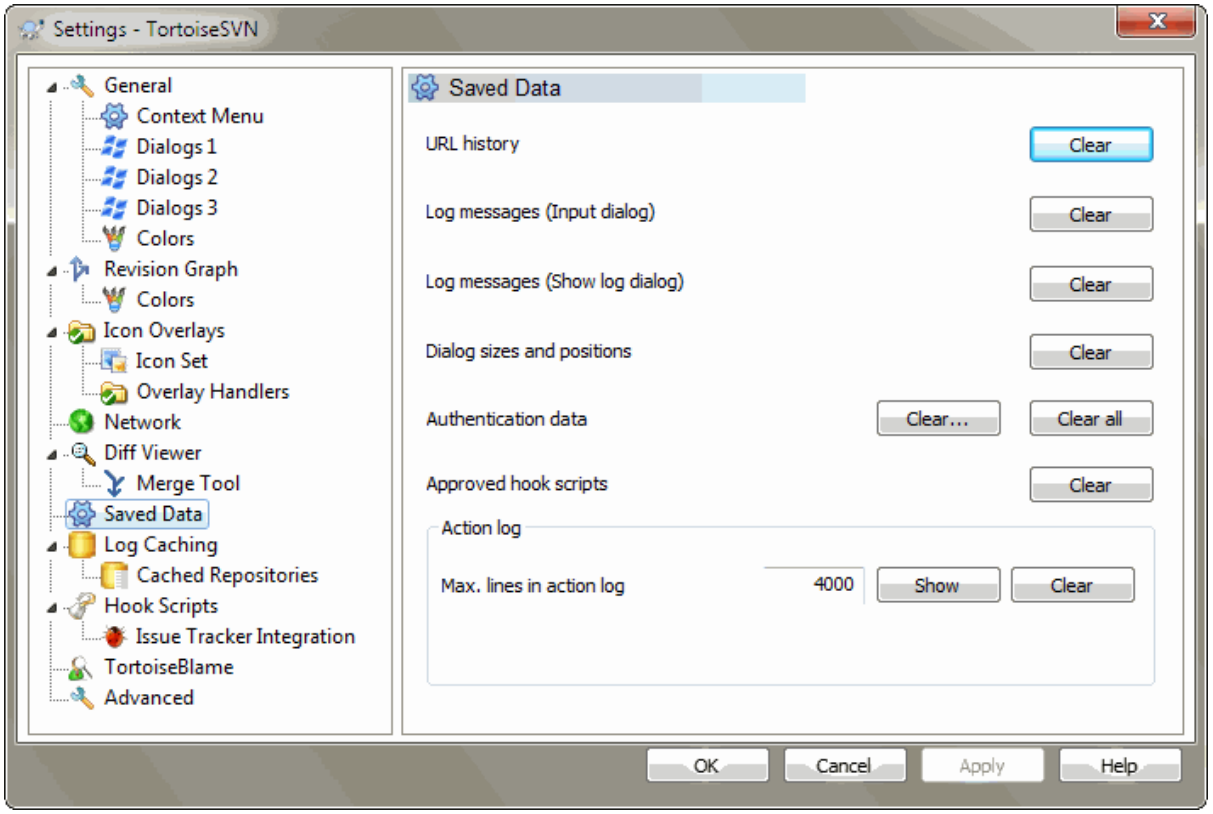


Figura 4.83. Janela de Configurações, Opção Dados Salvos

Para sua comodidade, TortoiseSVN salva muitas das configurações que você utiliza, e lembra onde você esteve anteriormente. Se você desejar limpar os dados de cache, pode ser feito aqui.

Histórico de URLs
: Whenever you checkout a working copy, merge changes or use the repository browser, TortoiseSVN keeps a record of recently used URLs and offers them in a combo box. Sometimes that list gets cluttered with outdated URLs so it is useful to flush it out periodically.

  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**.

Mensagens de log (Janela de entrada)
: TortoiseSVN stores recent commit log messages that you enter. These are stored per repository, so if you access many repositories this list can grow quite large.

Mensagens de log (Mostra janela de log)
: TortoiseSVN caches log messages fetched by the Show Log dialog to save time when you next show the log. If someone else edits a log message and you already have that message cached, you will not see the change until you clear the cache. Log message caching is enabled on the Log Cache tab.

Dimensões e posições das janelas
: Muitos diálogos lembrar o tamanho e tela de posição que você usado pela última vez.

Dados de autenticação
: When you authenticate with a Subversion server, the username and password are cached locally so you don't have to keep entering them. You may want to clear this for security reasons, or because you want to access the repository under a different username ... does John know you are using his PC?

If you want to clear authentication data for one particular server only, use the Clear... instead of the Clear all button.

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

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

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

4.31.7. Cache de Log

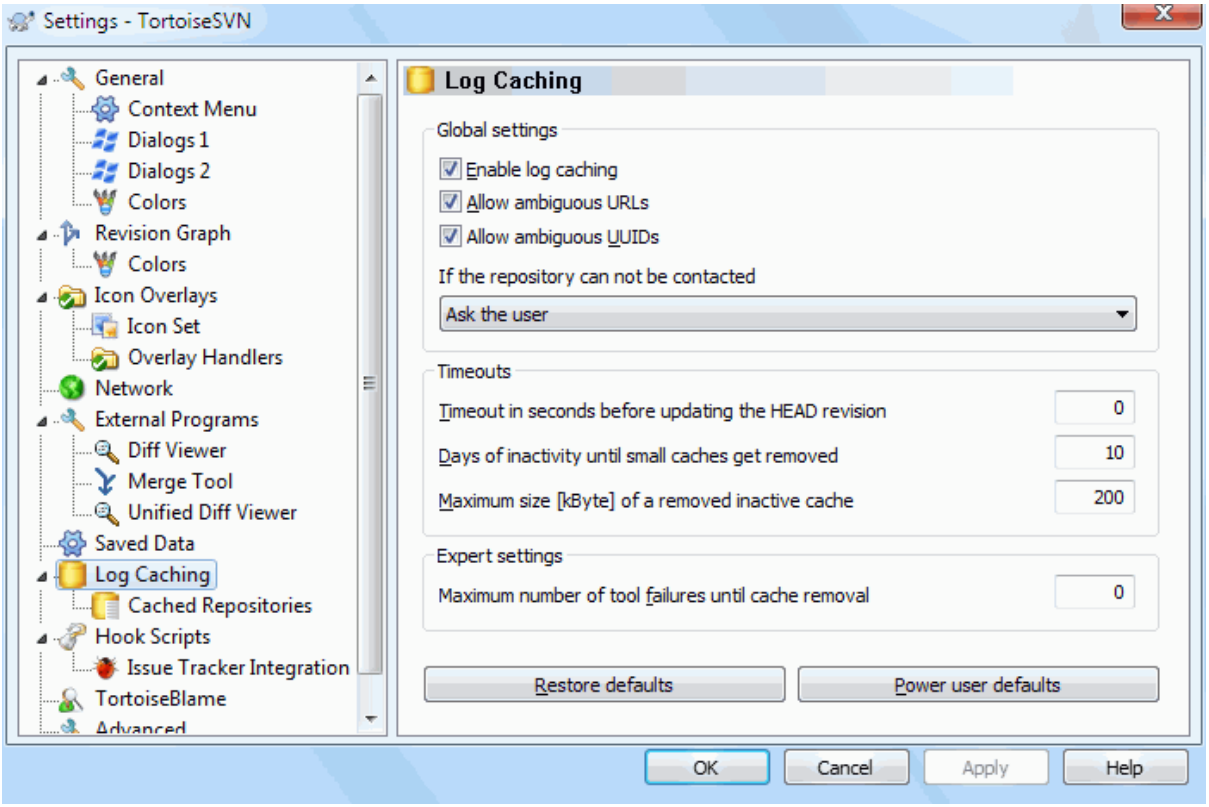


Figura 4.84. Janela de Configurações, Opção Cache da Auditoria

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.

Habilitar cache de log
: 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.

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

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

Se o repositório não pode ser acessado
: 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.

Tempo limite antes de atualizar a última revisão
: 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.

Dias de inatividade até que pequenos caches sejam removidos
: If you browse around a lot of repositories you will accumulate a lot of log caches. If you're not actively using them, the cache will not grow very big, so TortoiseSVN purges them after a set time by default. Use this item to control cache purging.

Tamanho máximo de um cache inativo removido
: Larger caches are more expensive to reacquire, so TortoiseSVN only purges small caches. Fine tune the threshold with this value.

Número máximo de falhas da ferramenta antes da remoção do cache
: Occasionally something goes wrong with the caching and causes a crash. If this happens the cache is normally deleted automatically to prevent a recurrence of the problem. If you use the less stable nightly build you may opt to keep the cache anyway.

4.31.7.1. Repositórios em Cache

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

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

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

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

4.31.7.2. Estatísticas de Cache de Log

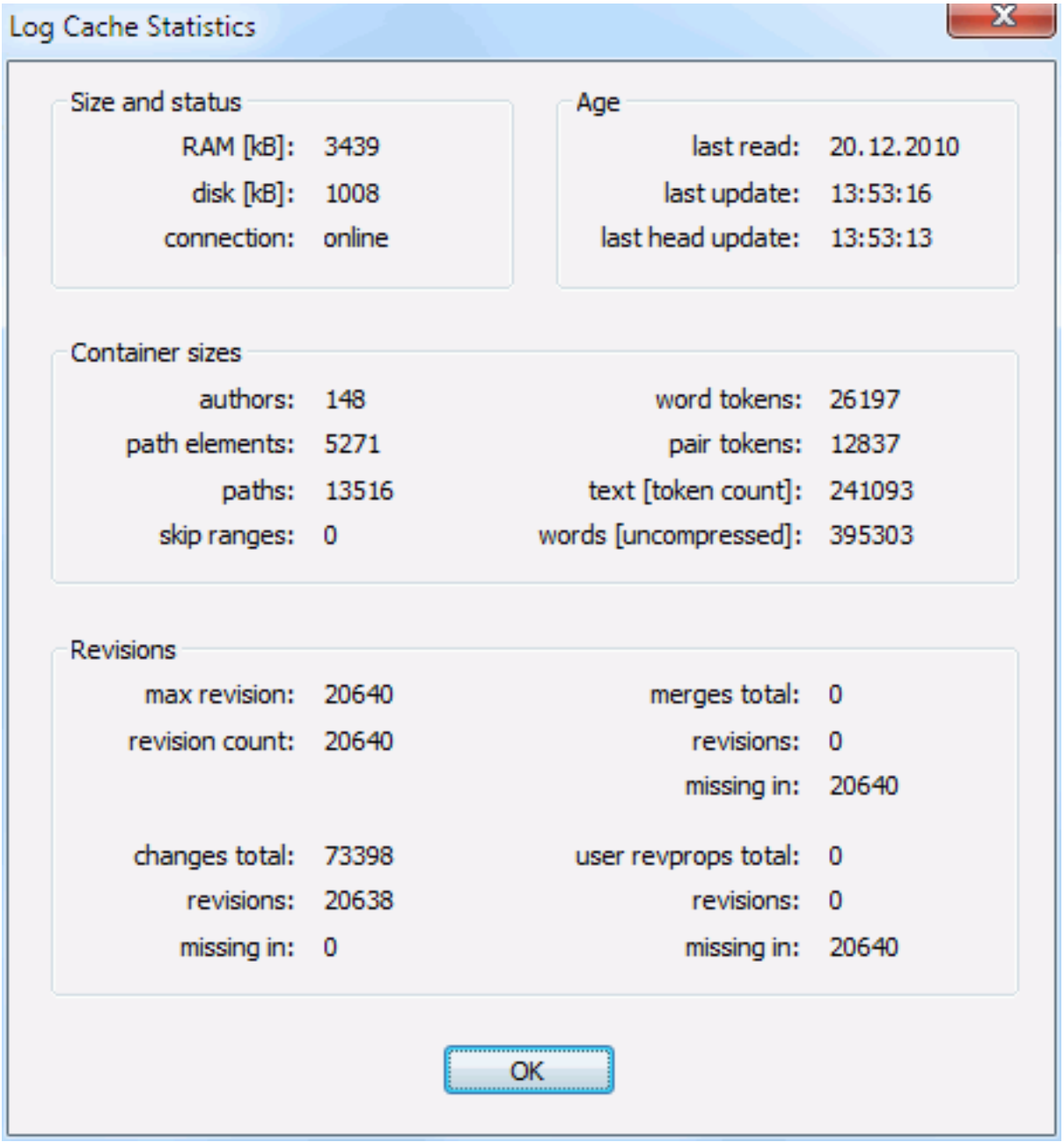


Figura 4.85. Janela de Configurações, Opção Estatísticas de Cache da Auditoria

clique no botão Detalhes para ver statisticas detalhadas de um cache em particular. Muitos dos campos mostrados aqui são de interesse de desenvolvedores do TortoiseSVN, por esse motivo não são exibidos todos os seus detalhes.

RAM
: Quantidade de memória necessária para carregar o cache

Disco
: Quantidade de espaço em disco usado para o cache, Dados são comprimidos, então o uso do disco é geralmente bastante modesto.

Conexão
: Mostra se o repositório estava disponível última vez que o cache foi usado.

Última atualização
: A última vez que o conteúdo do cache foi alterado.

Última atualização principal
: A Última vez que foi requisitada a última revisão do servidor.

Autores
: O número de autores diferentes com mensagens registradas no cache.

Caminhos
: O número de caminhos listados, como você veria usando `svn log -v`.

INtervalos omitidos
: 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.

Revisão máxima
: O maior número de revisão armazenada em cache

Número de revisões
: O número de revisões armazenadas no cache. Esta é outra medida de cache complementar.

4.31.8. Client Side Hook Scripts

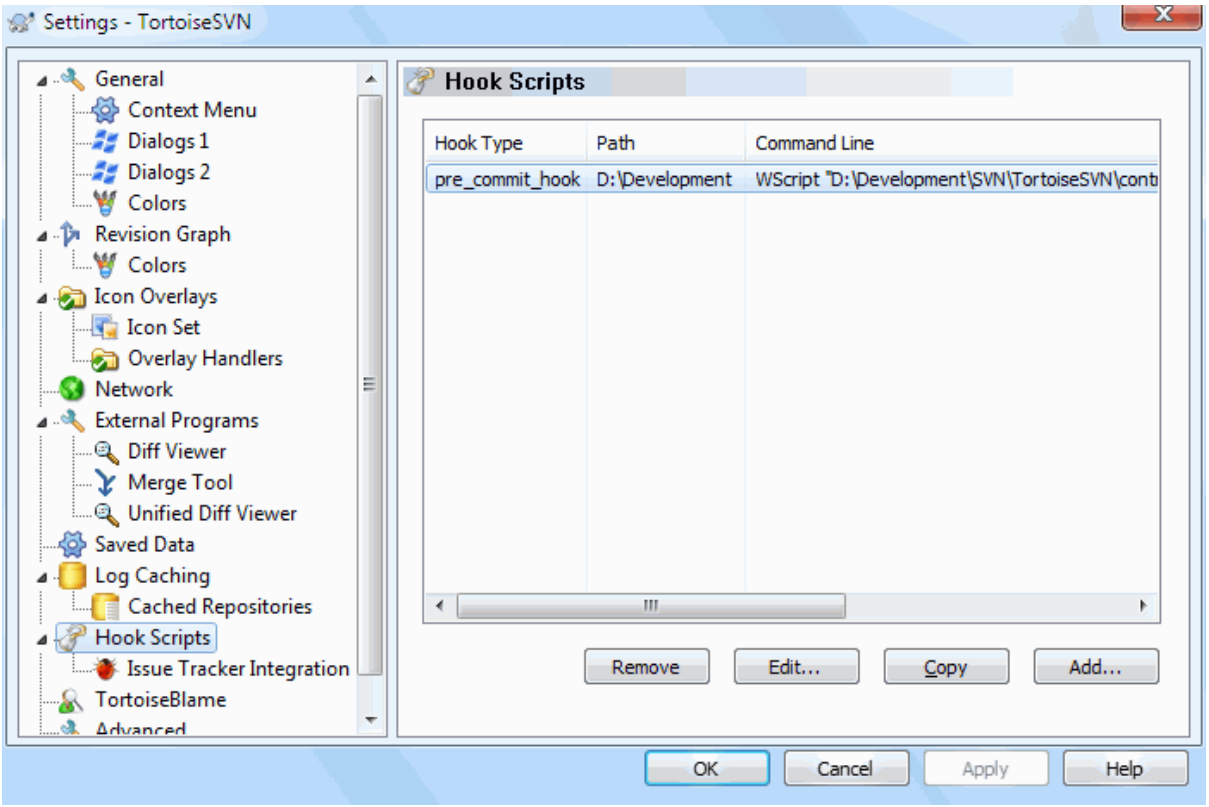


Figura 4.86. Janela de Configurações, Opção Scripts de Gancho

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 Seção 3.3, “Rotinas de eventos no servidor”, these scripts are executed locally on the client.

One application for such hooks might be to call a program like `SubWCRev.exe` to update version numbers after a commit, and perhaps to trigger a rebuild.

Note that you can also specify such hook scripts using special properties on your working copy. See the section Seção 4.18.2, “Propriedades do Projeto TortoiseSVN” for details.

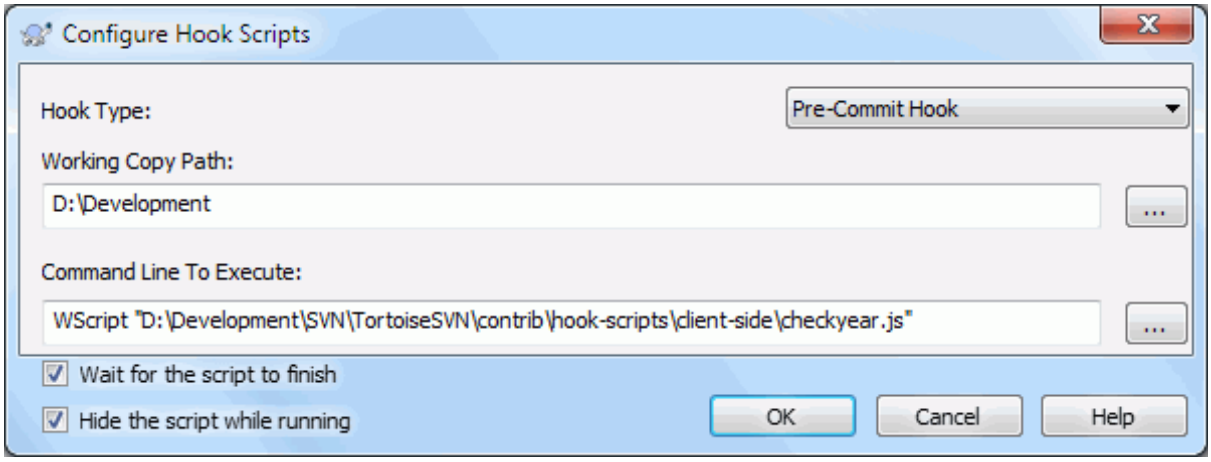


Figura 4.87. Janela de Configurações, Opção Configuração de Scripts de Gancho

To add a new hook script, simply click Add and fill in the details.

There are currently these types of hook script available

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

Manual Pre-commit
: If this is specified, the commit dialog shows a button Run Hook which when clicked runs the specified hook script. The hook script receives a list of all checked files and folders and the commit message if there was one entered.

Check-commit
: Called after the user clicks OK in the commit dialog, and before the commit dialog closes. This hook gets a list of all the checked files. If the hook returns an error, the commit dialog stays open.

  If the returned error message contains paths on newline separated lines, those paths will get selected in the commit dialog after the error message is shown.

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

Post-commit
: Called after the commit finishes successfully.

Start-update
: Chamado antes que a notificação de uptate-to-revision seja exibida.

Pre-update
: Called before the actual Subversion update or switch begins.

Post-update
: Called after the update, switch or checkout finishes (whether successful or not).

Pré-conexão
: Called before an attempt to contact the repository. Called at most once in five minutes.

Pre-lock
: Called before an attempt to lock a file.

Post-lock
: Called after a file has been locked.

A hook is defined for a particular working copy path. You only need to specify the top level path; if you perform an operation in a sub-folder, TortoiseSVN will automatically search upwards for a matching path.

Next you must specify the command line to execute, starting with the path to the hook script or executable. This could be a batch file, an executable file or any other file which has a valid windows file association, e.g. a perl script. Note that the script must not be specified using a UNC path as Windows shell execute will not allow such scripts to run due to security restrictions.

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

Start-commit
    `PATHMESSAGEFILECWD`

Manual Pre-commit
    `PATHMESSAGEFILECWD`

Check-commit
    `PATHMESSAGEFILECWD`

Pre-commit
    `PATHDEPTHMESSAGEFILECWD`

Post-commit
    `PATHDEPTHMESSAGEFILEREVISIONERRORCWD`

Start-update
    `PATHCWD`

Pre-update
    `PATHDEPTHREVISIONCWD`

Post-update
    `PATHDEPTHREVISIONERRORCWDRESULTPATH`

Pré-conexão
    no parameters are passed to this script. You can pass a custom parameter by appending it to the script path.

Pre-lock
    `PATHLOCKFORCEMESSAGEFILEERRORCWD`

Post-lock
    `PATHLOCKFORCEMESSAGEFILEERRORCWD`

The meaning of each of these parameters is described here:

PATH
    A path to a temporary file which contains all the paths for which the operation was started. Each path is on a separate line in the temp file.

    Note that for operations done remotely, e.g. in the repository browser, those paths are not local paths but the urls of the affected items.

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

    Os valores possíveis são:

    -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.

LOCK
Either `true` when locking, or `false` when unlocking.

FORCE
Either `true` or `false`, depending on whether the operation was forced or not.

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.

RESULTPATH
A path to a temporary file which contains all the paths which were somehow touched by the operation. Each path is on a separate line in the temp file.

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

If you want the Subversion operation to hold off until the hook has completed, check Wait for the script to finish.

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

Sample client hook scripts can be found in the `contrib` folder in the *TortoiseSVN repository* [https://svn.code.sf.net/p/tortoisesvn/code/trunk/contrib/hook-scripts]. (Seção 3, “Licença” explains how to access the repository.)

When debugging hook scripts you may want to echo progress lines to the DOS console, or insert a pause to stop the console window disappearing when the script completes. Because I/O is redirected this will not normally work. However you can redirect input and output explicitly to CON to overcome this. e.g.

```
echo Checking Status > con
pause < con > con
```

A small tool is included in the TortoiseSVN installation folder named `ConnectVPN.exe`. You can use this tool configured as a pre-connect hook to connect automatically to your VPN before TortoiseSVN tries to connect to a repository. Just pass the name of the VPN connection as the first parameter to the tool.

4.31.8.1. Integração para Rastreador de Reportes

TortoiseSVN can use a COM plugin to query issue trackers when in the commit dialog. The use of such plugins is described in Seção 4.29.2, “Obtendo Informações do Rastreador de Reportes”. If your system administrator has provided you with a plugin, which you have already installed and registered, this is the place to specify how it integrates with your working copy.

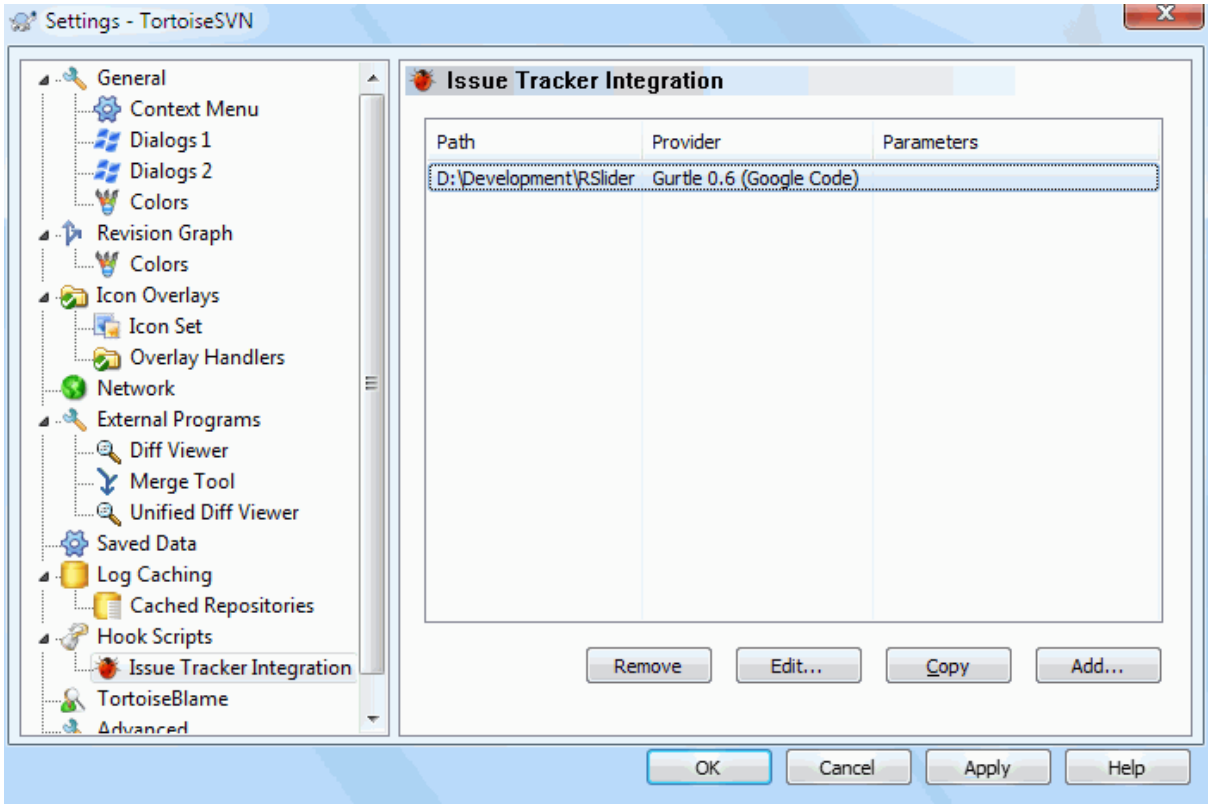


Figura 4.88. Janela de Configurações, Opção Integração com Rastreador de Reportes

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

If you want all users to use the same COM plugin for your project, you can specify the plugin also with the properties `bugtraq:provideruuid`, `bugtraq:provideruuid64` and `bugtraq:providerparams`.

bugtraq:provideruuid
: This property specifies the COM UUID of the IBugtraqProvider, for example `{91974081-2DC7-4FB1-B3BE-0DE1C8D6CE4E}`. (This example is the UUID of the *Gurtle bugtraq provider* [http://code.google.com/p/gurtle/], which is a provider for the *Google Code* [http://code.google.com/hosting/] issue tracker.)

bugtraq:provideruuid64
: This is the same as `bugtraq:provideruuid`, but for the 64-bit version of the IBugtraqProvider.

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

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

4.31.9. Configurações TortoiseBlame

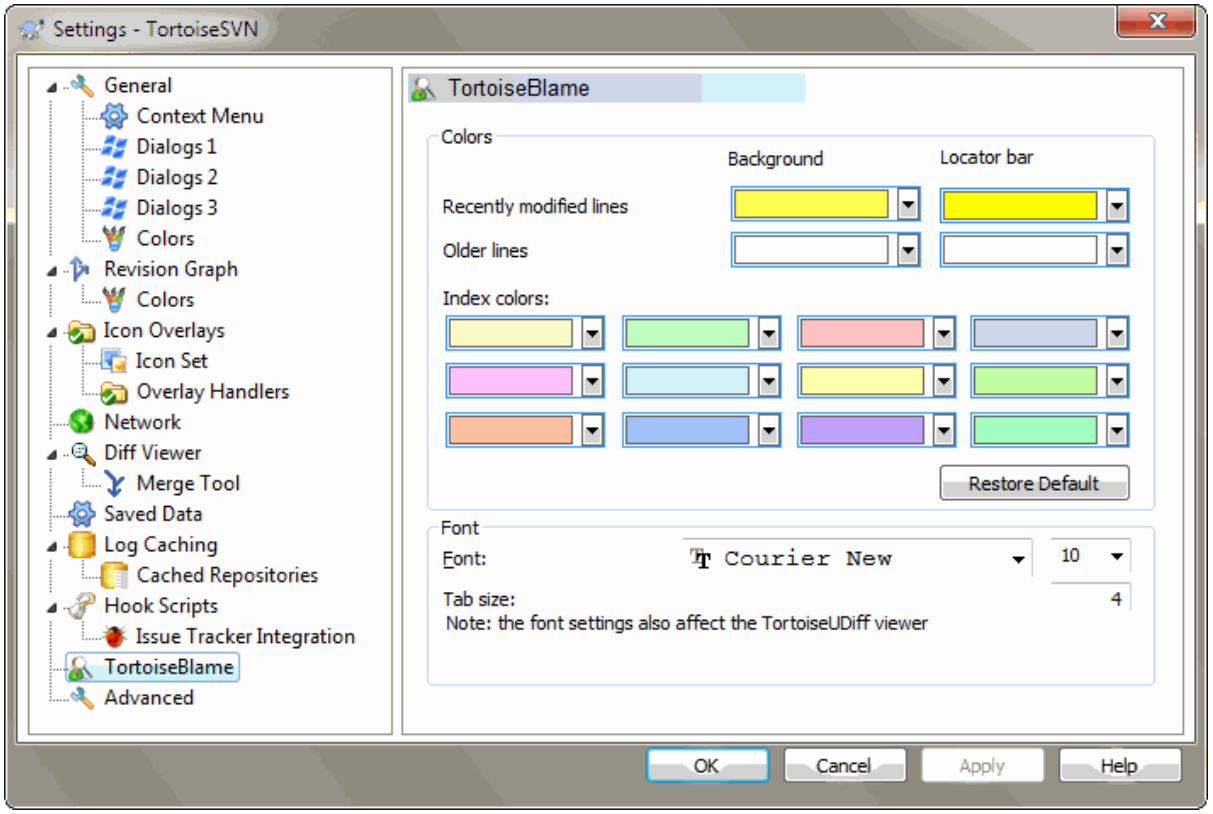


Figura 4.89. Janela de Configurações, Opção Tortoise Autoria

The settings used by TortoiseBlame are controlled from the main context menu, not directly with TortoiseBlame itself.

Cores
: TortoiseBlame can use the background colour to indicate the age of lines in a file. You set the endpoints by specifying the colours for the newest and oldest revisions, and TortoiseBlame uses a linear interpolation between these colours according to the repository revision indicated for each line.

  You can specify different colours to use for the locator bar. The default is to use strong contrast on the locator bar while keeping the main window background light so that you can still read the text.

Fonte
: You can select the font used to display the text, and the point size to use. This applies both to the file content, and to the author and revision information shown in the left pane.

Abas
: Defines how many spaces to use for expansion when a tab character is found in the file content.

4.31.10. TortoiseUDiff Settings

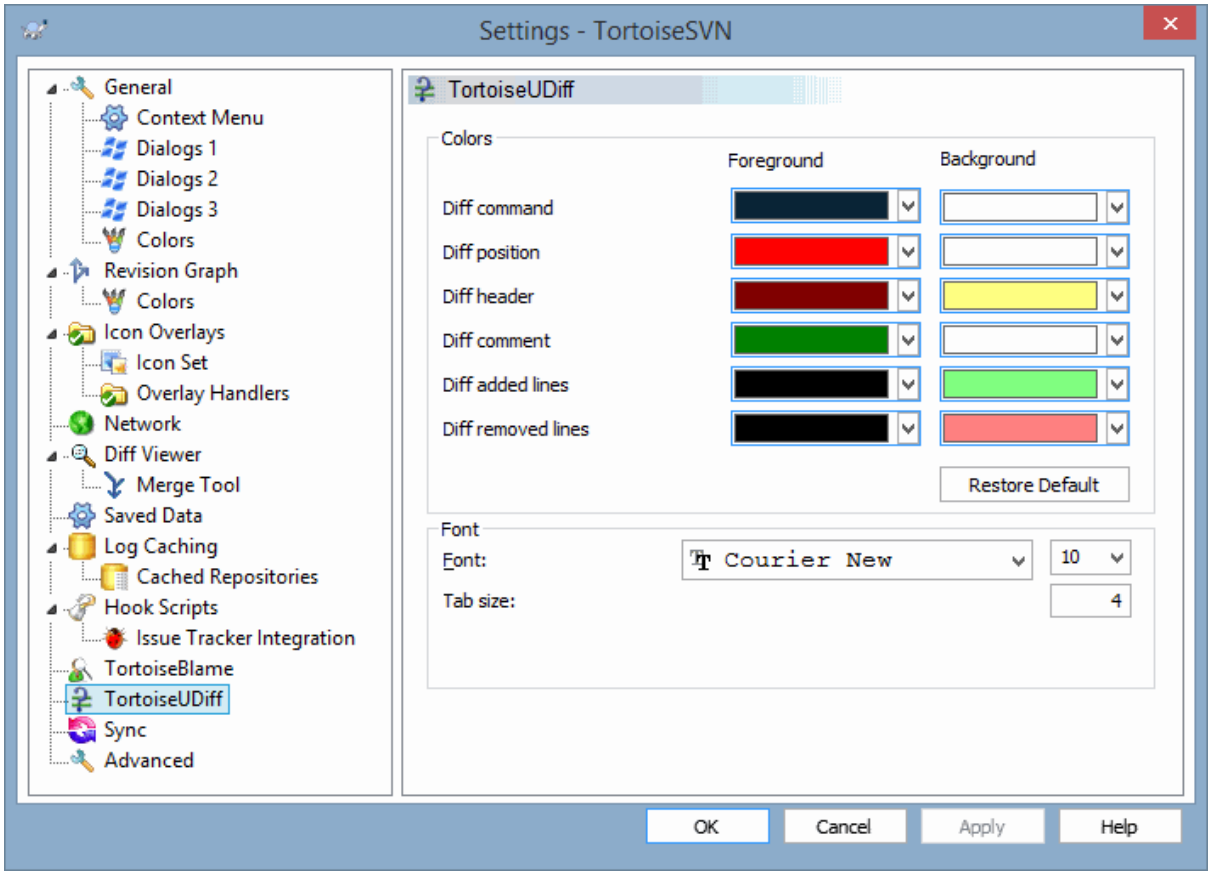


Figura 4.90. The Settings Dialog, TortoiseUDiff Page

The settings used by TortoiseUDiff are controlled from the main context menu, not directly with TortoiseUDiff itself.

Cores
: The default colors used by TortoiseUDiff are usually ok, but you can configure them here.

Fonte
: You can select the font used to display the text, and the point size to use.

Abas
: Defines how many spaces to use for expansion when a tab character is found in the file diff.

4.31.11. Exporting TSVN Settings

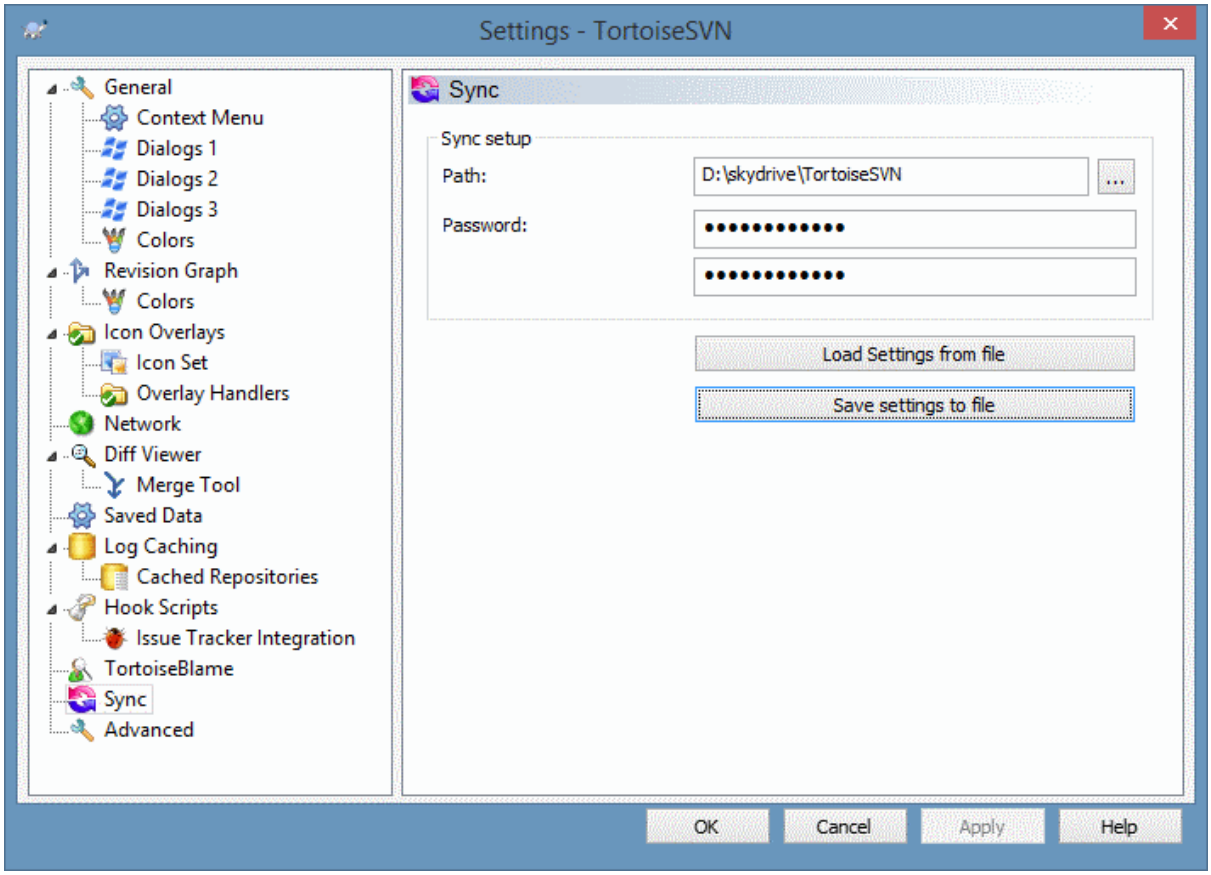


Figura 4.91. The Settings Dialog, Sync Page

You can sync all TortoiseSVN settings to and from an encrypted file. The file is encrpyted with the password you enter so you don't have to worry if you store that file on a cloud folder like OneDrive, GDrive, DropBox, ...

When a path and password is specified, TortoiseSVN will sync all settings automatically and keep them in sync.

You can also export/import an encrypted files with all the settings manually. When you do that, you're asked for the path of the file and the password to encrypt/decrypt the settings file.

When exporting the settings manually, you can also optionally include all local settings which are not included in a normal export or in a sync. Local settings are settings which include local paths which usually vary between computers. These local settings include the configured diff and merge tools and hook scripts.

4.31.12. Configurações Avançadas

A few infrequently used settings are available only in the advanced page of the settings dialog. These settings modify the registry directly and you have to know what each of these settings is used for and what it does. Do not modify these settings unless you are sure you need to change them.

AllowAuthSave
: Sometimes multiple users use the same account on the same computer. In such situations it's not really wanted to save the authentication data. Setting this value to `false` disables the `save authentication` button in the authentication dialog.

AllowUnversionedObstruction
: If an update adds a new file from the repository which already exists in the local working copy as an unversioned file, the default action is to keep the local file, showing it as a (possibly) modified version of the new file from the repository. If you would prefer TortoiseSVN to create a conflict in such situations, set this value to `false`.

AlwaysExtendedMenu
: As with the explorer, TortoiseSVN shows additional commands if the **Shift** key is pressed while the context menu is opened. To force TortoiseSVN to always show those extended commands, set this value to `true`.

AutoCompleteMinChars
: The minimum amount of chars from which the editor shows an auto-completion popup. The default value is `3`.

AutocompleteRemovesExtensions
: The auto-completion list shown in the commit message editor displays the names of files listed for commit. To also include these names with extensions removed, set this value to `true`.

BlockPeggedExternals
: File externals that are pegged to a specific revision are blocked by default from being selected for a commit. This is because a subsequent update would revert those changes again unless the pegged revision of the external is adjusted.

  Set this value to `false` in case you still want to commit changes to such external files.

BlockStatus
: If you don't want the explorer to update the status overlays while another TortoiseSVN command is running (e.g. Update, Commit, ...) then set this value to `true`.

CacheTrayIcon
: To add a cache tray icon for the TSVNCache program, set this value to `true`. This is really only useful for developers as it allows you to terminate the program gracefully.

ColumnsEveryWhere
: 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, set this value to `true`. Note that the extra columns are only available in XP. Vista and later doesn't support that feature any more. However some third-party explorer replacements do support those even on Windows versions later than XP.

ConfigDir
: You can specify a different location for the Subversion configuration file here. This will affect all TortoiseSVN operations.

CtrlEnter
: In most dialogs in TortoiseSVN, you can use **Ctrl+Enter** to dismiss the dialog as if you clicked on the OK button. If you don't want this, set this value to `false`.

Debug
: Set this to `true` if you want a dialog to pop up for every command showing the command line used to start TortoiseProc.exe.

DebugOutputString
: Set this to `true` if you want TortoiseSVN to print out debug messages during execution. The messages can be captured with special debugging tools only.

DialogTitles
: The default format (value of 0) of dialog titles is `url/path - name of dialog - TortoiseSVN`. If you set this value to 1, the format changes to `name of dialog - url/path - TortoiseSVN`.

DiffBlamesWithTortoiseMerge
: TortoiseSVN allows you to assign an external diff viewer. Most such viewers, however, are not suited for change blaming (Seção 4.24.2, “Diferenças de Autoria”), so you might wish to fall back to TortoiseMerge in this case. To do so, set this value to `true`.

DlgStickySize
: This value specifies the number of pixels a dialog has to be near a border before the dialog sticks to it. The default value is 3. To disable this value set the value to zero.

FixCaseRenames
: Some apps change the case of filenames without notice but those changes aren't really necessary nor wanted. For example a change from `file.txt` to `FILE.TXT` wouldn't bother normal Windows applications, but Subversion is case sensitive in these situations. So TortoiseSVN automatically fixes such case changes.

  If you don't want TortoiseSVN to automatically fix such case changes for you, you can set this value to `false`.

FullRowSelect
: The status list control which is used in various dialogs (e.g., commit, check-for-modifications, add, revert, ...) uses full row selection (i.e., if you select an entry, the full row is selected, not just the first column). This is fine, but the selected row then also covers the background image on the bottom right, which can look ugly. To disable full row select, set this value to `false`.

GroupTaskbarIconsPerRepo
: This option determines how the Win7 taskbar icons of the various TortoiseSVN dialogs and windows are grouped together. This option has no effect on Vista!

1. The default value is 0. With this setting, the icons are grouped together by application type. All dialogs from TortoiseSVN are grouped together, all windows from TortoiseMerge are grouped together, ...

Figura 4.92. Barra de tarefas com agrupamento padrão

2. If set to 1, then instead of all dialogs in one single group per application, they're grouped together by repository. For example, if you have a log dialog and a commit dialog open for repository `A`, and a check-for-modifications dialog and a log dialog for repository `B`, then there are two application icon groups shown in the Win7 taskbar, one group for each repository. But TortoiseMerge windows are not grouped together with TortoiseSVN dialogs.

Figura 4.93. Barra de tarefas com agrupamento por repositório

3. Se definir para 2, então o agrupamento funciona como a configuração definiada para 1, exceto que a janela do TortoiseSVN, TortoiseMerge, TortoiseBlame, TortoiseIDiff e do TortoiseUDiff serão agrupadas juntas. Por exemplo, se você tem a janela de submissão aberta e então dá um duplo clique em um arquivo modificado, a janela aberta de diferenças do TortoiseMerge será colocada junto no mesmo ícone do grupo na barra de tarefas como o ícone da janela de submissão.

Figura 4.94. Barra de tarefas com agrupamento por repositório

4. If set to 3, then the grouping works as with the setting set to 1, but the grouping isn't done according to the repository but according to the working copy. This is useful if you have all your projects in the same repository but different working copies for each project.

5. If set to 4, then the grouping works as with the setting set to 2, but the grouping isn't done according to the repository but according to the working copy.

HideExternalInfo
: If this is set to `false`, then every `svn:externals` is shown during an update separately.

  If it is set to `true` (the default), then update information for externals is only shown if the externals are affected by the update, i.e. changed in some way. Otherwise nothing is shown as with normal files and folders.

GroupTaskbarIconsPerRepoOverlay
: This has no effect if the option `GroupTaskbarIconsPerRepo` is set to 0 (see above).

  If this option is set to `true`, then every icon on the Win7 taskbar shows a small colored rectangle overlay, indicating the repository the dialogs/windows are used for.

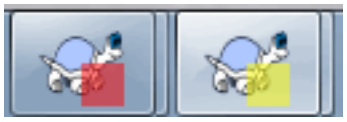


Figura 4.95. Taskbar grouping with repository color overlays

IncludeExternals
: By default, TortoiseSVN always runs an update with externals included. This avoids problems with inconsistent working copies. If you have however a lot of externals set, an update can take quite a while. Set this value to `false` to run the default update with externals excluded. To update with externals included, either run the `Update to revision...` dialog or set this value to `true` again.

LogFindCopyFrom
: When the log dialog is started from the merge wizard, already merged revisions are shown in gray, but revisions beyond the point where the branch was created are also shown. These revisions are shown in black because those can't be merged.

  If this option is set to `true` then TortoiseSVN tries to find the revision where the branch was created from and hide all the revisions that are beyond that revision. Since this can take quite a while, this option is disabled by default. Also this option doesn't work with some SVN servers (e.g., Google Code Hosting, see *issue #5471* [http://code.google.com/p/support/issues/detail?id=5471]).

LogMultiRevFormat
: A format string for the log messages when multiple revisions are selected in the log dialog.

  You can use the following placeholders in your format string:

  %1!ld!
  : é substituído com o número da revisão

  %2!s!
  : é substituído com a mensagem de log curta da revisão

LogStatusCheck
: The log dialog shows the revision the working copy path is at in bold. But this requires that the log dialog fetches the status of that path. Since for very big working copies this can take a while, you can set this value to `false` to deactivate this feature.

MaxHistoryComboItems
: Comboboxes for URLs and paths show a history of previously used URLs/paths if possible. This settings controls how many previous items are saved and shown. The default is 25 items.

MergeLogSeparator
: When you merge revisions from another branch, and merge tracking information is available, the log messages from the revisions you merge will be collected to make up a commit log message. A pre-defined string is used to separate the individual log messages of the merged revisions. If you prefer, you can set this to a value containing a separator string of your choice.

NumDiffWarning
: If you want to show the diff at once for more items than specified with this settings, a warning dialog is shown first. The default is 10.

OldVersionCheck
: TortoiseSVN checks whether there's a new version available about once a week. If an updated version is found, the commit dialog shows a link control with that info. If you prefer the old behavior back where a dialog pops up notifying you about the update, set this value to `true`.

RepoBrowserTrySVNParentPath
: The repository browser tries to fetch the web page that's generated by an SVN server configured with the SVNParentPath directive to get a list of all repositories. To disable that behavior, set this value to `false`.

ScintillaBidirectional
: This option enables the bidirectional mode for the commit message edit box. If enabled, right-to-left language text editing is done properly. Since this feature is expensive, it is disabled by default. You can enable this by setting this value to `true`.

ScintillaDirect2D
: This option enables the use of Direct2D accelerated drawing in the Scintilla control which is used as the edit box in e.g. the commit dialog, and also for the unified diff viewer. With some graphic cards however this sometimes doesn't work properly so that the cursor to enter text isn't always visible. If that happens, you can turn this feature off by setting this value to `false`.

OutOfDateRetry
: This parameter specifies how TortoiseSVN behaves if a commit fails due to an out-of-date error:

  0
  : The user is asked whether to update the working copy or not, and the commit dialog is not reopened after the update.

  1
  : This is the default. The user is asked whether to update the working copy or not, and the commit dialog is reopened after the update so the user can proceed with the commit right away.

  2
  : Similar to `1`, but instead of updating only the paths selected for a commit, the update is done on the working copy root. This helps to avoid inconsistent working copies.

  3
  : The user is not asked to update the working copy. The commit simply fails with the out-of-date error message.

PlaySound
: If set to `true`, TortoiseSVN will play a system sound when an error or warning occurs, or another situation which is important and requires your attention. Set this to `false` if you want to keep TortoiseSVN quiet. Note that the project monitor has its own setting for playing sounds, which you can configure in its settings dialog.

ShellMenuAccelerators
: TortoiseSVN uses accelerators for its explorer context menu entries. Since this can lead to doubled accelerators (e.g. the `SVN Commit` has the **Alt-C** accelerator, but so does the `Copy` entry of explorer). If you don't want or need the accelerators of the TortoiseSVN entries, set this value to `false`.

ShowContextMenuIcons
: This can be useful if you use something other than the windows explorer or if you get problems with the context menu displaying incorrectly. Set this value to `false` if you don't want TortoiseSVN to show icons for the shell context menu items. Set this value to `true` to show the icons again.

ShowAppContextMenuIcons
: If you don't want TortoiseSVN to show icons for the context menus in its own dialogs, set this value to `false`.

StyleCommitMessages
: The commit and log dialog use styling (e.g. bold, italic) in commit messages (see Seção 4.4.5, “Registro de Mensagens de Submissão” for details). If you don't want to do this, set the value to `false`.

UpdateCheckURL
: This value contains the URL from which TortoiseSVN tries to download a text file to find out if there are updates available. This might be useful for company admins who don't want their users to update TortoiseSVN until they approve it.

UseCustomWordBreak
: The standard edit controls do not stop on forward slashes like they're found in paths and urls. TortoiseSVN uses a custom word break procedure for the edit controls. If you don't want that and use the default instead, set this value to 0. If you only want the default for edit controls in combo boxes, set this value to 1.

VersionCheck
: TortoiseSVN checks whether there's a new version available about once a week. If you don't want TortoiseSVN to do this check, set this value to `false`.

4.32. Passo Final

Doe!

Even though TortoiseSVN and TortoiseMerge are free, you can support the developers by sending in patches and playing an active role in the development. You can also help to cheer us up during the endless hours we spend in front of our computers.

While working on TortoiseSVN we love to listen to music. And since we spend many hours on the project we need a *lot* of music. Therefore we have set up some wish-lists with our favourite music CDs and DVDs: https://tortoisesvn.net/donate.html Please also have a look at the list of people who contributed to the project by sending in patches or translations.

Capítulo 5. Project Monitor

The project monitor is a helpful tool that monitors repositories and notifies you in case there are new commits.

The projects can be monitored via a working copy path or directly via their repository URLs.

The project monitor scans each project in a configurable interval, and every time new commits are detected a notification popup is shown. Also the icon that is added to the system tray changes to indicate that there are new commits.

Snarl

If *Snarl* [https://snarl.fullphat.net/] is installed and active, then the project monitor automatically uses Snarl to show the notifications about newly detected commits.

5.1. Adding projects to monitor

If you first start the project monitor, the tree view on the left side is empty. To add projects, click on the button at the top of the dialog named Add Project.

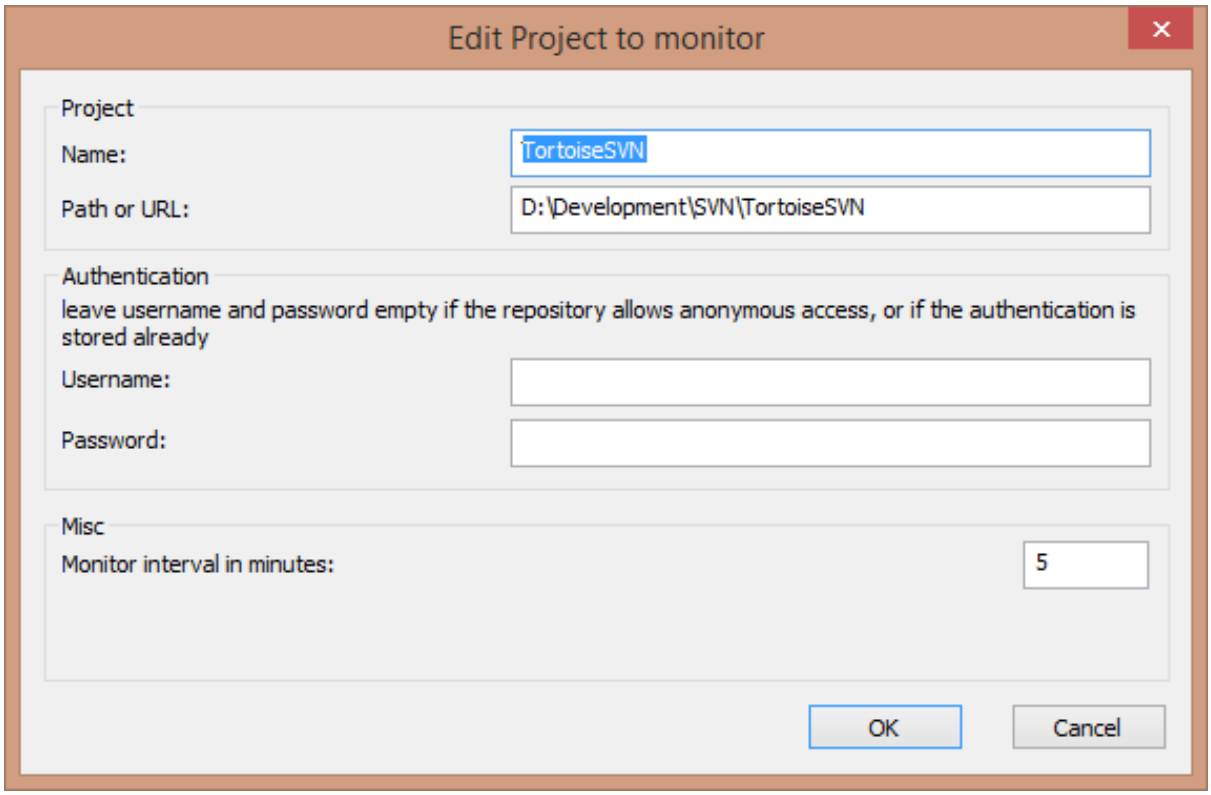


Figura 5.1. The edit project dialog of the project monitor

To add a project for monitoring, fill in the required information. The name of the project is not optional and must be filled in, all other information is optional.

If the box for `Path or Url` is left empty, then a folder is added. This is useful to group monitored projects.

The fields `Username` and `Password` should only be filled in if the repository does not provide anonymous read access, and only if the authentication is not stored by Subversion itself. If you're accessing the monitored repository with TortoiseSVN or other svn clients and you've stored the authentication already, you should leave this empty: you won't have to edit those projects manually if the password changes.

The `Monitor interval in minutes` specifies the minutes to wait in between checks. The smallest interval is one minute.

check interval

If there are a lot of users monitoring the same repository and the bandwidth on the server is limited, a repository admin can set the minimum for check intervals using an `svnrobots.txt` file. A detailed explanation on how this works can be found on the project monitor website:

http://stefanstools.sourceforge.net/svnrobots.html [http://stefanstools.sourceforge.net/svnrobots.html]

5.2. Monitor dialog

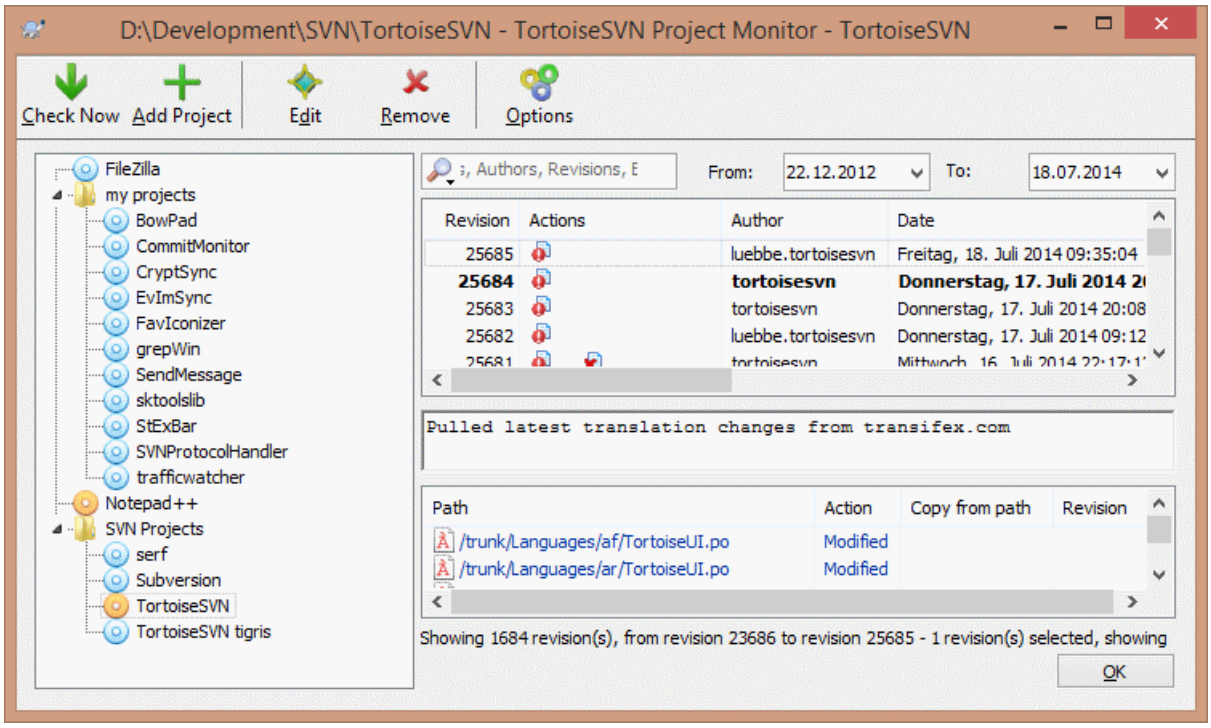


Figura 5.2. The main dialog of the project monitor

The project monitor shows all monitored projects on the left in a tree view. The projects can be moved around, for example one project can be moved below another project, making it a child/subproject.

A click on a project shows all the log messages of that project on the right.

Projects that have updates are shown in bold, with the number of new commits in brackets at the right. A click on a project marks it automatically as read.

5.2.1. Principais operações

The toolbar at the top of the dialog allows to configure and operate the project monitor.

Checar agora
: While each monitored project is checked according to the interval that's set up, clicking this button will force a check of all projects immediately. Note that if there are updates, the notification won't show up until all projects have been checked.

Adicionar projeto
: Opens a new dialog to set up a new project for monitoring.

Editar
: Opens the configuration dialog for the selected project.

Remover
: Removes the selected project after a confirmation dialog is shown.

Mark all as read
: Marks all revisions in all projects as read. Note that if you select a project with unread revisions, those revisions are automatically marked as read when you select another project.

  If you hold down the **Shift** key when clicking the button, all error states are also cleared if there are any.

Update all
: Runs an **Update** on all monitored working copies. Projects that are monitored via an url are not updated, only those that are set up with a working copy path.

Opções
: Shows a dialog to configure the behavior of the project monitor.

Capítulo 6. SubWCRev

SubWCRev is Windows console program which can be used to read the status of a Subversion working copy and optionally perform keyword substitution in a template file. This is often used as part of the build process as a means of incorporating working copy information into the object you are building. Typically it might be used to include the revision number in an “About” box.

6.1. The SubWCRev Command Line

SubWCRev reads the Subversion status of all files in a working copy, excluding externals by default. It records the highest commit revision number found, and the commit timestamp of that revision, it also records whether there are local modifications in the working copy, or mixed update revisions. The revision number, update revision range and modification status are displayed on stdout.

SubWCRev.exe is called from the command line or a script, and is controlled using the command line parameters.

```
SubWCRev WorkingCopyPath [SrcVersionFile DstVersionFile] [-nmdfe]
```

`WorkingCopyPath` is the path to the working copy being checked. You can only use SubWCRev on working copies, not directly on the repository. The path may be absolute or relative to the current working directory.

If you want SubWCRev to perform keyword substitution, so that fields like repository revision and URL are saved to a text file, you need to supply a template file `SrcVersionFile` and an output file `DstVersionFile` which contains the substituted version of the template.

You can specify ignore patterns for SubWCRev to prevent specific files and paths from being considered. The patterns are read from a file named `.subwcrevignore`. The file is read from the specified path, and also from the working copy root. If the file does not exist, no files or paths are ignored. The `.subwcrevignore` file can contain multiple patterns, separated by newlines. The patterns are matched against the paths relative to the repository root and paths relative to the path of the `.subwcrevignore` file. For example, to ignore all files in the `doc` folder of the TortoiseSVN working copy, the `.subwcrevignore` would contain the following lines:

```
/trunk/doc
/trunk/doc/*
```

Or, assuming the `.subwcrevignore` file is in the working copy root which is checked out from trunk, using the patterns

```
doc
doc/*
```

is the same as the example above.

To ignore all images, the ignore patterns could be set like this:

```
*.png
*.jpg
*.ico
*.bmp
```

Importante

The ignore patterns are case-sensitive, just like Subversion is.

Dica

To create a file with a starting dot in the Windows explorer, enter `.subwcrevignore.`. Note the trailing dot.

There are a number of optional switches which affect the way SubWCRev works. If you use more than one, they must be specified as a single group, e.g. `-nm`, not `-n -m`.

Alternar	Descrição
-n	If this switch is given, SubWCRev will exit with `ERRORLEVEL 7` if the working copy contains local modifications. This may be used to prevent building with uncommitted changes present.
-N	If this switch is given, SubWCRev will exit with `ERRORLEVEL 11` if the working copy contains unversioned items that are not ignored.
-m	If this switch is given, SubWCRev will exit with `ERRORLEVEL 8` if the working copy contains mixed revisions. This may be used to prevent building with a partially updated working copy.
-d	If this switch is given, SubWCRev will exit with `ERRORLEVEL 9` if the destination file already exists.
-f	If this switch is given, SubWCRev will include the last-changed revision of folders. The default behaviour is to use only files when getting the revision numbers.
-e	If this switch is given, SubWCRev will examine directories which are included with `svn:externals`, but only if they are from the same repository. The default behaviour is to ignore externals.
-E	If this switch is given, same as `-e`, but it ignores the externals with explicit revisions, when the revision range inside of them is only the given explicit revision in the properties. So it doesn't lead to mixed revisions.
-x	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.
-F	If this switch is given, SubWCRev will ignore any `.subwcrevignore` files and include all files.
-q	If this switch is given, SubWCRev will perform the keyword substitution without showing working copy status on stdout.

Tabela 6.1. List of available command line switches

If there is no error, SubWCRev returns zero. But in case an error occurs, the error message is written to stderr and shown in the console. And the returned error codes are:

Código de Erro	Descrição
1	Erro de sintaxe. Um ou mais parâmetros de linha de comando são inválidos.
2	O arquivo ou pasta especificada na linha de comando não foi encontrado.
3	O arquivo de entrada não pôde ser aberto, ou o arquivo de destino não pode ser criado.
4	Could not allocate memory. This could happen if e.g. the source file is too big.
5	O arquivo fonte não pode ser escaneado apropriadamente.
6	SVN error: Subversion returned with an error when SubWCRev tried to find the information from the working copy.
7	The working copy has local modifications. This requires the `-n` switch.

Código de Erro	Descrição
8	The working copy has mixed revisions. This requires the `-m` switch.
9	The output file already exists. This requires the `-d` switch.
10	O caminho especificado não é uma cópia de trabalho ou parte de uma.
11	The working copy has unversioned files or folders in it. This requires the `-N` switch.

Tabela 6.2. List of SubWCRev error codes

6.2. Alteração da palavra-chave

If a source and destination files are supplied, SubWCRev copies source to destination, performing keyword substitution as follows:

Palavra-chave	Descrição
$WCREV$	Replaced with the highest commit revision in the working copy.
$WCREV&$	Replaced with the highest commit revision in the working copy, ANDed with the value after the & char. For example: `$WCREV&0xFFFF$`
$WCREV-$, $WCREV+$	Replaced with the highest commit revision in the working copy, with the value after the + or - char added or subtracted. For example: `$WCREV-1000$`
$WCDATE$, $WCDATEUTC$	Replaced with the commit date/time of the highest commit revision. By default, international format is used: `yyyy-mm-dd hh:mm:ss`. Alternatively, you can specify a custom format which will be used with `strftime()`, for example: `$WCDATE=%a %b %d %I:%M:%S %p$`. For a list of available formatting characters, look at the *referência online* [http://msdn.microsoft.com/en-us/library/fe06s4ak.aspx].
$WCNOW$, $WCNOWUTC$	Replaced with the current system date/time. This can be used to indicate the build time. Time formatting can be used as described for `$WCDATE$`.
$WCRANGE$	Replaced with the update revision range in the working copy. If the working copy is in a consistent state, this will be a single revision. If the working copy contains mixed revisions, either due to being out of date, or due to a deliberate update-to-revision, then the range will be shown in the form 100:200.
$WCMIXED$	`$WCMIXED?TText:FText$` 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.
$WCUNVER$	`$WCUNVER?TText:FText$` is replaced with `TText` if there are unversioned items in the working copy, or `FText` if not.
$WCEXTALLFIXED$	`$WCEXTALLFIXED?TText:FText$` is replaced with `TText` if all externals are fixed to an explicit revision, or `FText` if not.
$WCISTAGGED$	`$WCISTAGGED?TText:FText$` is replaced with `TText` if the repository URL contains the tags classification pattern, or `FText` if not.
$WCURL$	Replaced with the repository URL of the working copy path passed to SubWCRev.
$WCINSVN$	`$WCINSVN?TText:FText$` 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.

Palavra-chave	Descrição
$WCLOCKDATE$, $WCLOCKDATEUTC$	Replaced with the lock date. Time formatting can be used as described for `$WCDATE$`.
$WCLOCKOWNER$	Substituído com o nome do dono do bloqueio
$WCLOCKCOMMENT$	Substituído com o comentário do bloqueio
$WCUNVER$	`$WCUNVER?TText:FText$` is replaced with `TText` if there are unversioned files or folders in the working copy, or `FText` if not.

Tabela 6.3. Lista de palavras-chave disponíveis

SubWCRev does not directly support nesting of expressions, so for example you cannot use an expression like:

```
#define SVN_REVISION    "$WCMIXED?$WCRANGE$:$WCREV$$"
```

But you can usually work around it by other means, for example:

```
#define SVN_RANGE       $WCRANGE$
#define SVN_REV         $WCREV$
#define SVN_REVISION    "$WCMIXED?SVN_RANGE:SVN_REV$"
```

Dica

Some of these keywords apply to single files rather than to an entire working copy, so it only makes sense to use these when SubWCRev is called to scan a single file. This applies to `$WCINSVN$`, `$WCNEEDSLOCK$`, `$WCISLOCKED$`, `$WCLOCKDATE$`, `$WCLOCKOWNER$` and `$WCLOCKCOMMENT$`.

6.3. Exemplo de Palavra-chave

The example below shows how keywords in a template file are substituted in the output file.

```
// Test file for SubWCRev

char *Revision      = "$WCREV$";
char *Revision16    = "$WCREV&0xFF$";
char *Revisionp100  = "$WCREV+100$";
char *Revisionm100  = "$WCREV-100$";
char *Modified      = "$WCMODS?Modified:Not modified$";
char *Unversioned   = "$WCUNVER?Unversioned items found:no unversioned items$";
char *Date          = "$WCDATE$";
char *CustDate      = "$WCDATE=%a, %d %B %Y$";
char *DateUTC       = "$WCDATEUTC$";
char *CustDateUTC   = "$WCDATEUTC=%a, %d %B %Y$";
char *TimeNow       = "$WCNOW$";
char *TimeNowUTC    = "$WCNOWUTC$";
char *RevRange      = "$WCRANGE$";
char *Mixed         = "$WCMIXED?Mixed revision WC:Not mixed$";
char *ExtAllFixed   = "$WCEXTALLFIXED?All externals fixed:Not all externals fixed$";
char *IsTagged      = "$WCISTAGGED?Tagged:Not tagged$";
char *URL           = "$WCURL$";
char *isInSVN       = "$WCINSVN?versioned:not versioned$";
char *needslck      = "$WCNEEDSLOCK?TRUE:FALSE$";
```

```
char *islocked      = "$WCISLOCKED?locked:not locked$";
char *lockdateutc   = "$WCLOCKDATEUTC$";
char *lockdate      = "$WCLOCKDATE$";
char *lockcustutc   = "$WCLOCKDATEUTC=%a, %d %B %Y$";
char *lockcust      = "$WCLOCKDATE=%a, %d %B %Y$";
char *lockown       = "$WCLOCKOWNER$";
char *lockcmt       = "$WCLOCKCOMMENT$";

#if $WCMODS?1:0$
#error Source is modified
#endif

// End of file
```

After running `SubWCRev.exe path\to\workingcopy testfile.tmpl testfile.txt`, the output file `testfile.txt` would looks like this:

```
// Test file for SubWCRev

char *Revision      = "22837";
char *Revision16    = "53";
char *Revisionp100  = "22937";
char *Revisionm100  = "22737";
char *Modified      = "Modified";
char *Unversioned   = "no unversioned items";
char *Date          = "2012/04/26 18:47:57";
char *CustDate      = "Thu, 26 April 2012";
char *DateUTC       = "2012/04/26 16:47:57";
char *CustDateUTC   = "Thu, 26 April 2012";
char *TimeNow       = "2012/04/26 20:51:17";
char *TimeNowUTC    = "2012/04/26 18:51:17";
char *RevRange      = "22836:22837";
char *Mixed         = "Mixed revision WC";
char *ExtAllFixed   = "All externals fixed";
char *IsTagged      = "Not tagged";
char *URL           = "https://svn.code.sf.net/p/tortoisesvn/code/trunk";
char *isInSVN       = "versioned";
char *needslck      = "FALSE";
char *islocked      = "not locked";
char *lockdateutc   = "1970/01/01 00:00:00";
char *lockdate      = "1970/01/01 01:00:00";
char *lockcustutc   = "Thu, 01 January 1970";
char *lockcust      = "Thu, 01 January 1970";
char *lockown       = "";
char *lockcmt       = "";

#if 1
#error Source is modified
#endif

// End of file
```

Dica

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.

6.4. 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:

Método	Descrição
.GetWCInfo	This method traverses the working copy gathering the revision information. Naturally you must call this before you can access the information using the remaining methods. The first parameter is the path. The second parameter should be true if you want to include folder revisions. Equivalent to the `-f` command line switch. The third parameter should be true if you want to include svn:externals. Equivalent to the `-e` command line switch.
.GetWCInfo2	The same as `GetWCInfo()` but with a fourth parameter that sets the equivalent to the `-E` command line switch.
.Revision	The highest commit revision in the working copy. Equivalent to `$WCREV$`.
.Date	The commit date/time of the highest commit revision. Equivalent to `$WCDATE$`.
.Author	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	Verdadeiro se houver modificações locais
.HasUnversioned	Verdadeiro se existirem itens não versionados.
.Url	Replaced with the repository URL of the working copy path used in `GetWCInfo`. Equivalent to `$WCURL$`.
.IsSvnItem	Verdadeiro se o item estiver versionado.
.NeedsLocking	True if the item has the `svn:needs-lock` property set.
.IsLocked	Verdadeiro se o item estiver bloqueado
.LockCreationDate	String representing the date when the lock was created, or an empty string if the item is not locked.
.LockOwner	String representing the lock owner, or an empty string if the item is not locked.
.LockComment	The message entered when the lock was created.

Tabela 6.4. COM/automation methods supported

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");

revObject1 = new ActiveXObject("SubWCRev.object");
revObject2 = new ActiveXObject("SubWCRev.object");
revObject3 = new ActiveXObject("SubWCRev.object");
revObject4 = new ActiveXObject("SubWCRev.object");

revObject1.GetWCInfo(
    filesystem.GetAbsolutePathName("."), 1, 1);
revObject2.GetWCInfo(
    filesystem.GetAbsolutePathName(".."), 1, 1);
```

```
revObject3.GetWCInfo(
    filesystem.GetAbsolutePathName("SubWCRev.cpp"), 1, 1);
revObject4.GetWCInfo2(
    filesystem.GetAbsolutePathName("..\\.."), 1, 1, 1);

wcInfoString1 = "Revision = " + revObject1.Revision +
                "\nMin Revision = " + revObject1.MinRev +
                "\nMax Revision = " + revObject1.MaxRev +
                "\nDate = " + revObject1.Date +
                "\nURL = " + revObject1.Url + "\nAuthor = " +
                revObject1.Author + "\nHasMods = " +
                revObject1.HasModifications + "\nIsSvnItem = " +
                revObject1.IsSvnItem + "\nNeedsLocking = " +
                revObject1.NeedsLocking + "\nIsLocked = " +
                revObject1.IsLocked + "\nLockCreationDate = " +
                revObject1.LockCreationDate + "\nLockOwner = " +
                revObject1.LockOwner + "\nLockComment = " +
                revObject1.LockComment;
wcInfoString2 = "Revision = " + revObject2.Revision +
                "\nMin Revision = " + revObject2.MinRev +
                "\nMax Revision = " + revObject2.MaxRev +
                "\nDate = " + revObject2.Date +
                "\nURL = " + revObject2.Url + "\nAuthor = " +
                revObject2.Author + "\nHasMods = " +
                revObject2.HasModifications + "\nIsSvnItem = " +
                revObject2.IsSvnItem + "\nNeedsLocking = " +
                revObject2.NeedsLocking + "\nIsLocked = " +
                revObject2.IsLocked + "\nLockCreationDate = " +
                revObject2.LockCreationDate + "\nLockOwner = " +
                revObject2.LockOwner + "\nLockComment = " +
                revObject2.LockComment;
wcInfoString3 = "Revision = " + revObject3.Revision +
                "\nMin Revision = " + revObject3.MinRev +
                "\nMax Revision = " + revObject3.MaxRev +
                "\nDate = " + revObject3.Date +
                "\nURL = " + revObject3.Url + "\nAuthor = " +
                revObject3.Author + "\nHasMods = " +
                revObject3.HasModifications + "\nIsSvnItem = " +
                revObject3.IsSvnItem + "\nNeedsLocking = " +
                revObject3.NeedsLocking + "\nIsLocked = " +
                revObject3.IsLocked + "\nLockCreationDate = " +
                revObject3.LockCreationDate + "\nLockOwner = " +
                revObject3.LockOwner + "\nLockComment = " +
                revObject3.LockComment;
wcInfoString4 = "Revision = " + revObject4.Revision +
                "\nMin Revision = " + revObject4.MinRev +
                "\nMax Revision = " + revObject4.MaxRev +
                "\nDate = " + revObject4.Date +
                "\nURL = " + revObject4.Url + "\nAuthor = " +
                revObject4.Author + "\nHasMods = " +
                revObject4.HasModifications + "\nIsSvnItem = " +
                revObject4.IsSvnItem + "\nNeedsLocking = " +
                revObject4.NeedsLocking + "\nIsLocked = " +
                revObject4.IsLocked + "\nLockCreationDate = " +
                revObject4.LockCreationDate + "\nLockOwner = " +
                revObject4.LockOwner + "\nLockComment = " +
                revObject4.LockComment;
```

```
WScript.Echo(wcInfoString1);
WScript.Echo(wcInfoString2);
WScript.Echo(wcInfoString3);
WScript.Echo(wcInfoString4);
```

The following listing is an example on how to use the SubWCRev COM object from C#:

```
using LibSubWCRev;
SubWCRev sub = new SubWCRev();
sub.GetWCInfo("C:\\PathToMyFile\\MyFile.cc", true, true);
if (sub.IsSvnItem == true)
{
    MessageBox.Show("versioned");
}
else
{
    MessageBox.Show("not versioned");
}
```

Capítulo 7. IBugtraqProvider interface

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

We can't provide information and tutorials on how you have to implement a COM object in your preferred programming language, but we have example plugins in C++/ATL and C# in our repository in the `contrib/issue-tracker-plugins` folder. In that folder you can also find the required include files you need to build your plugin. (Seção 3, “Licença” explains how to access the repository.)

Importante

You should provide both a 32-bit and 64-bit version of your plugin. Because the x64-Version of TortoiseSVN can not use a 32-bit plugin and vice-versa.

7.1. Naming conventions

If you release an issue tracker plugin for TortoiseSVN, please do *not* name it *Tortoise<Something>*. We'd like to reserve the *Tortoise* prefix for a version control client integrated into the windows shell. For example: TortoiseCVS, TortoiseSVN, TortoiseHg, TortoiseGit and TortoiseBzr are all version control clients.

Please name your plugin for a Tortoise client *Turtle<Something>*, where *<Something>* refers to the issue tracker that you are connecting to. Alternatively choose a name that sounds like *Turtle* but has a different first letter. Nice examples are:

- Gurtle - Um plugin de rastreador de reportes para Google code
- TurtleMine - Um plugin de rastreador de reportes para Redmine
- VurtleOne - Um plugin de rastreador de reportes para VersionOne

7.2. The IBugtraqProvider interface

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

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

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

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

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

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

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

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

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

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

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

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

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

This is the main method of the plugin. This method is called from the TortoiseSVN commit dialog when the user clicks on the plugin button.

The `parameters` string is the string the user has to enter in the settings dialog when he configures the plugin. Usually a plugin would use this to find the URL of the issue tracker and/or login information or more.

The `commonRoot` string contains the parent path of all items selected to bring up the commit dialog. Note that this is *not* the root path of all items which the user has selected in the commit dialog. For the branch/tag dialog, this is the path which is to be copied.

The `pathList` parameter contains an array of paths (as strings) which the user has selected for the commit.

The `originalMessage` parameter contains the text entered in the log message box in the commit dialog. If the user has not yet entered any text, this string will be empty.

The `newMessage` return string is copied into the log message edit box in the commit dialog, replacing whatever is already there. If a plugin does not modify the `originalMessage` string, it must return the same string again here, otherwise any text the user has entered will be lost.

7.3. The IBugtraqProvider2 interface

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

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

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

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

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

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

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

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

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

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

This method is called from the TortoiseSVN commit dialog when the user clicks on the plugin button. This method is called instead of `GetCommitMessage()`. Please refer to the documentation for `GetCommitMessage` for the parameters that are also used there.

The parameter `commonURL` is the parent URL of all items selected to bring up the commit dialog. This is basically the URL of the `commonRoot` path.

The parameter `bugID` contains the content of the bug-ID field (if it is shown, configured with the property `bugtraq:message`).

The return parameter `bugIDOut` is used to fill the bug-ID field when the method returns.

The `revPropNames` and `revPropValues` return parameters can contain name/value pairs for revision properties that the commit should set. A plugin must make sure that both arrays have the same size on return! Each property name in `revPropNames` must also have a corresponding value in `revPropValues`. If no revision properties are to be set, the plugin must return empty arrays.

```
HRESULT CheckCommit (
  [in] HWND hParentWnd,
```

```
  [in] BSTR parameters,
  [in] BSTR commonURL,
  [in] BSTR commonRoot,
  [in] SAFEARRAY(BSTR) pathList,
  [in] BSTR commitMessage,
  [out, retval] BSTR * errorMessage
);
```

This method is called right before the commit dialog is closed and the commit begins. A plugin can use this method to validate the selected files/folders for the commit and/or the commit message entered by the user. The parameters are the same as for `GetCommitMessage2()`, with the difference that `commonURL` is now the common URL of all *checked* items, and `commonRoot` the root path of all checked items.

For the branch/tag dialog, the `commonURL` is the source URL of the copy, and `commonRoot` is set to the target URL of the copy.

The return parameter `errorMessage` must either contain an error message which TortoiseSVN shows to the user or be empty for the commit to start. If an error message is returned, TortoiseSVN shows the error string in a dialog and keeps the commit dialog open so the user can correct whatever is wrong. A plugin should therefore return an error string which informs the user *what* is wrong and how to correct it.

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

  // The common root of all paths that got committed.
  [in] BSTR commonRoot,

  // All the paths that got committed.
  [in] SAFEARRAY(BSTR) pathList,


  // The text already present in the commit message.
  [in] BSTR logMessage,

  // The revision of the commit.
  [in] ULONG revision,


  // An error to show to the user if this function
  // returns something else than S_OK
  [out, retval] BSTR * error
);
```

This method is called after a successful commit. A plugin can use this method to e.g., close the selected issue or add information about the commit to the issue. The parameters are the same as for `GetCommitMessage2`.

```
HRESULT HasOptions(
  // Whether the provider provides options
  [out, retval] VARIANT_BOOL *ret
);
```

This method is called from the settings dialog where the user can configure the plugins. If a plugin provides its own configuration dialog with `ShowOptionsDialog`, it must return TRUE here, otherwise it must return FALSE.

```
HRESULT ShowOptionsDialog(
```

```
  // Parent window for the options dialog
  [in] HWND hParentWnd,

  // Parameters for your provider.
  [in] BSTR parameters,

  // The parameters string
  [out, retval] BSTR * newparameters
);
```

This method is called from the settings dialog when the user clicks on the "Options" button that is shown if `HasOptions` returns TRUE. A plugin can show an options dialog to make it easier for the user to configure the plugin.

The `parameters` string contains the plugin parameters string that is already set/entered.

The `newparameters` return parameter must contain the parameters string which the plugin constructed from the info it gathered in its options dialog. That `paramameters` string is passed to all other IBugtraqProvider and IBugtraqProvider2 methods.

Apêndice A. Frequently Asked Questions (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* [https://tortoisesvn.net/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* [https://sourceforge.net/p/tortoisesvn/tickets/] which tells you about some of the things we have on our To-Do list, and bugs which have already been fixed. If you think you have found a bug, or want to request a new feature, check here first to see if someone else got there before you.

If you have a question which is not answered anywhere else, the best place to ask it is on one of the mailing lists:

- <users@tortoisesvn.tigris.org> is the one to use if you have questions about using TortoiseSVN.
- If you want to help out with the development of TortoiseSVN, then you should take part in discussions on <dev@tortoisesvn.tigris.org>.
- If you want to help with the translation of the TortoiseSVN user interface or the documentation, send an e-mail to <translators@tortoisesvn.tigris.org>.

Apêndice B. Como eu faço...

Esse appendix contem solução para o seu problema/questão, você pode ter quando está usando o TortoiseSVN

B.1. Mover/copiar vários arquivos de uma só vez

Moving/Copying single files can be done by using TortoiseSVN → Rename.... But if you want to move/copy a lot of files, this way is just too slow and too much work.

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

B.2. Usuario forçando para entrar uma messagem de log

There are two ways to prevent users from committing with an empty log message. One is specific to TortoiseSVN, the other works for all Subversion clients, but requires access to the server directly.

B.2.1. Rotinas de eventos no servidor

If you have direct access to the repository server, you can install a pre-commit hook script which rejects all commits with an empty or too short log message.

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.

This method is the recommended way if your users also use other Subversion clients than TortoiseSVN. The drawback is that the commit is rejected by the server and therefore users will get an error message. The client can't know before the commit that it will be rejected. If you want to make TortoiseSVN have the OK button disabled until the log message is long enough then please use the method described below.

B.2.2. Propriedades do projeto

TortoiseSVN uses properties to control some of its features. One of those properties is the `tsvn:logminsize` property.

If you set that property on a folder, then TortoiseSVN will disable the OK button in all commit dialogs until the user has entered a log message with at least the length specified in the property.

For detailed information on those project properties, please refer to Seção 4.18, “Configurações do Projeto”.

B.3. Atualizar os arquivos selecionados a partir do repositório

Normally you update your working copy using TortoiseSVN → Update. But if you only want to pick up some new files that a colleague has added without merging in any changes to other files at the same time, you need a different approach.

Use TortoiseSVN → Check for Modifications. and click on Check repository to see what has changed in the repository. Select the files you want to update locally, then use the context menu to update just those files.

B.4. Reverter (Desfazer) revisões no repositório

B.4.1. Usar a janela de auditoria de revisão

By far the easiest way to revert the changes from one or more revisions, is to use the revision log dialog.

1. Selecione o arquivo ou pasta onde deseja reverter as modificações. Se você quiser reverter todas as modificações, isto precisa ser a pasta de nível superior.
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. If you want to pick out individual revisions and ranges, use the **Ctrl** key while selecting revisions. Right click on the selected revision(s), then select Context Menu → Revert changes from this revision.
4. Or if you want to make an earlier revision the new HEAD revision, right click on the selected revision, then select Context Menu → Revert to this revision. This will discard *all* changes after the selected revision.

You have reverted the changes within your working copy. Check the results, then commit the changes.

B.4.2. Use a janela de combinação

If you want to enter revision numbers as a list, you can use the Merge dialog. The previous method uses merging behind the scenes; this method uses it explicitly.

1. In your working copy select TortoiseSVN → Merge.
2. In the Merge Type dialog select Merge a range of revisions.
3. In the From: field enter the full repository URL of your working copy folder. This should come up as the default URL.
4. In the Revision range to merge field enter the list of revisions to roll back (or use the log dialog to select them as described above).
5. Make sure the Reverse merge checkbox is checked.
6. In the Merge options dialog accept the defaults.
7. Clique Merge para completar a mesclagem.

Você reverteu as mudanças dentro de sua cópia de trabalho. Verifique se ocorreu conforme o esperado, em seguida, confirmar as alterações.

B.4.3. Use `svndumpfilter`

Since TortoiseSVN never loses data, your “rolled back” revisions still exist as intermediate revisions in the repository. Only the HEAD revision was changed to a previous state. If you want to make revisions disappear completely from your repository, erasing all trace that they ever existed, you have to use more extreme measures. Unless there is a really good reason to do this, it is *not recommended*. One possible reason would be that someone committed a confidential document to a public repository.

The only way to remove data from the repository is to use the Subversion command line tool `svnadmin`. You can find a description of how this works in the *Repository Maintenance* [http://svnbook.red-bean.com/en/1.8/svn.reposadmin.maint.html].

B.5. Compare duas revisões de um arquivo ou pasta

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

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

If you want to compare two trees to see what has changed, for example the trunk and a tagged release, you can use TortoiseSVN → Revision Graph Select the two nodes to compare, then use Context Menu → Compare HEAD Revisions. This will show a list of changed files, and you can then select individual files to view the changes in detail. You can also export a tree structure containing all the changed files, or simply a list of all changed files. Read Seção 4.11.3, “Comparando Diretórios” for more information. Alternatively use Context Menu → Unified Diff of HEAD Revisions to see a summary of all differences, with minimal context.

B.6. Inclua um subprojeto em comum

Às vezes você vai querer incluir outro projeto dentro de sua cópia de trabalho, talvez alguma biblioteca de código . Há pelo menos 4 maneiras de lidar com isso.

B.6.1. Use svn:externals

Set the `svn:externals` property for a folder in your project. This property consists of one or more lines; each line has the name of a sub-folder which you want to use as the checkout folder for common code, and the repository URL that you want to be checked out there. For full details refer to Seção 4.19, “Itens Externos”.

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.

Se o projeto externo está no mesmo repositório, quaisquer alterações feitas serão incluídas na lista de submissão quando submeter seu projeto principal.

If the external project is in a different repository, any changes you make to the external project will be shown or indicated when you commit the main project, but you have to commit those external changes separately.

Of the three methods described, this is the only one which needs no setup on the client side. Once externals are specified in the folder properties, all clients will get populated folders when they update.

B.6.2. Use uma cópia de trabalho aninhada

Create a new folder within your project to contain the common code, but do not add it to Subversion.

Select TortoiseSVN → Checkout for the new folder and checkout a copy of the common code into it. You now have a separate working copy nested within your main working copy.

The two working copies are independent. When you commit changes to the parent, changes to the nested WC are ignored. Likewise when you update the parent, the nested WC is not updated.

B.6.3. Use um local relativo

If you use the same common core code in several projects, and you do not want to keep multiple working copies of it for every project that uses it, you can just check it out to a separate location which is related to all the other projects which use it. For example:

```
C:\Projects\Proj1
```

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

and refer to the common code using a relative path, e.g. `..\..\Common\DSPcore`.

If your projects are scattered in unrelated locations you can use a variant of this, which is to put the common code in one location and use drive letter substitution to map that location to something you can hard code in your projects, e.g. Checkout the common code to `D:\Documents\Framework` or `C:\Documents and Settings\{login}\My Documents\framework` then use

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

to create the drive mapping used in your source code. Your code can then use absolute locations.

```
#include "X:\superio\superio.h"
```

This method will only work in an all-PC environment, and you will need to document the required drive mappings so your team know where these mysterious files are. This method is strictly for use in closed development environments, and not recommended for general use.

B.6.4. Adicione o projeto para o repositório

The maybe easiest way is to simply add the project in a subfolder to your own project working copy. However this has the disadvantage that you have to update and upgrade this external project manually.

To help with the upgrade, TortoiseSVN provides a command in the explorer right-drag context menu. Simply right-drag the folder where you unzipped the new version of the external library to the folder in your working copy, and then select Context Menu → SVN Vendorbranch here. This will then copy the new files over to the target folder while automatically adding new files and removing files that aren't in the new version anymore.

B.7. Criar atalho para um repositório

If you frequently need to open the repository browser at a particular location, you can create a desktop shortcut using the automation interface to TortoiseProc. Just create a new shortcut and set the target to:

```
TortoiseProc.exe /command:repobrowser /path:"url/to/repository"
```

Of course you need to include the real repository URL.

B.8. Ignorar arquivos que já estão versionados

If you accidentally added some files which should have been ignored, how do you get them out of version control without losing them? Maybe you have your own IDE configuration file which is not part of the project, but which took you a long time to set up just the way you like it.

If you have not yet committed the add, then all you have to do is use TortoiseSVN → Undo Add... to undo the add. You should then add the file(s) to the ignore list so they don't get added again later by mistake.

If the files are already in the repository, they have to be deleted from the repository and added to the ignore list. Fortunately TortoiseSVN has a convenient shortcut for doing this. TortoiseSVN → Unversion and add to ignore list will first mark the file/folder for deletion from the repository, keeping the local copy. It also adds this

item to the ignore list so that it will not be added back into Subversion again by mistake. Once this is done you just need to commit the parent folder.

B.9. Unversion a working copy

If you have a working copy which you want to convert back to a plain folder tree without the `.svn` directory, you can simply export it to itself. Read Seção 4.27.1, “Removendo uma cópia de trabalho do controle de versão” to find out how.

B.10. Remover uma cópia de trabalho

If you have a working copy which you no longer need, how do you get rid of it cleanly? Easy - just delete it in Windows Explorer! Working copies are private local entities, and they are self-contained. Deleting a working copy in Windows Explorer does not affect the data in the repository at all.

Apêndice C. Dicas Úteis para Administradores

This appendix contains solutions to problems/questions you might have when you are responsible for deploying TortoiseSVN to multiple client computers.

C.1. Deploy TortoiseSVN via group policies

The TortoiseSVN installer comes as an MSI file, which means you should have no problems adding that MSI file to the group policies of your domain controller.

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

TortoiseSVN must be installed under *Computer Configuration* and not under *User Configuration*. This is because TortoiseSVN needs the CRT and MFC DLLs, which can only be deployed *per computer* and not *per user*. If you really must install TortoiseSVN on a per user basis, then you must first install the MFC and CRT package version 12 from Microsoft on each computer you want to install TortoiseSVN as per user.

You can customize the MSI file if you wish so that all your users end up with the same settings. TSVN settings are stored in the registry under `HKEY_CURRENT_USER\Software\TortoiseSVN` and general Subversion settings (which affect all Subversion clients) are stored in config files under `%APPDATA%\Subversion`. If you need help with MSI customization, try one of the MSI transform forums or search the web for “MSI transform”.

C.2. Redirecionar a verificação de atualização

TortoiseSVN checks if there's a new version available every few days. If there is a newer version available, a notification is shown in the commit dialog.

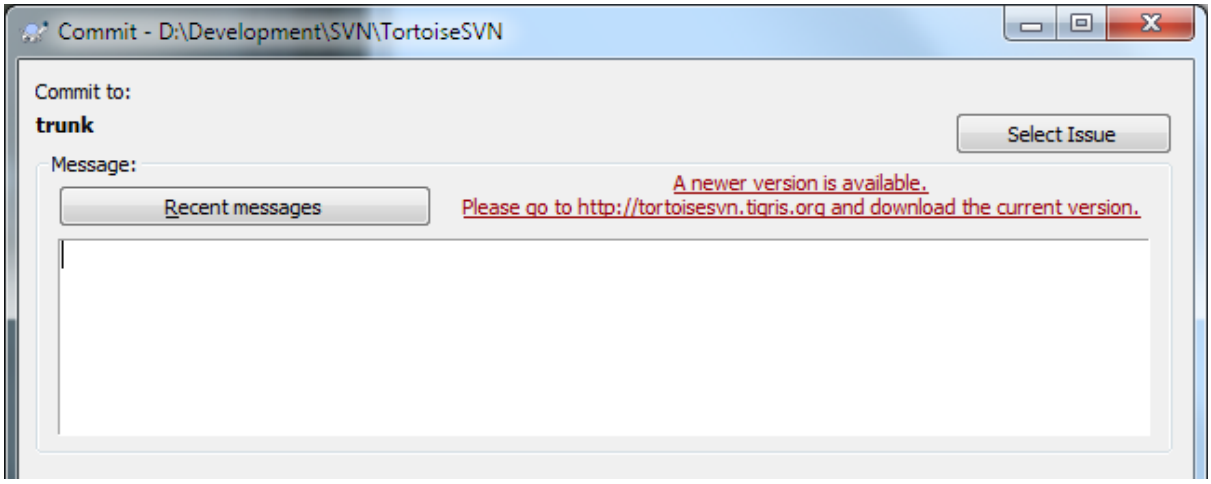


Figura C.1. Dialogo do commit, mostrando a notificação atualizada

If you're responsible for a lot of users in your domain, you might want your users to use only versions you have approved and not have them install always the latest version. You probably don't want that upgrade notification to show up so your users don't go and upgrade immediately.

Versions 1.4.0 and later of TortoiseSVN allow you to redirect that upgrade check to your intranet server. You can set the registry key `HKCU\Software\TortoiseSVN\UpdateCheckURL` (string value) to an URL pointing to a text file in your intranet. That text file must have the following format:

```
1.9.1.6000
A new version of TortoiseSVN is available for you to download!
http://192.168.2.1/downloads/TortoiseSVN-1.9.1.6000-svn-1.9.1.msi
```

The first line in that file is the version string. You must make sure that it matches the exact version string of the TortoiseSVN installation package. The second line is a custom text, shown in the commit dialog. You can write there whatever you want. Just note that the space in the commit dialog is limited. Too long messages will get truncated! The third line is the URL to the new installation package. This URL is opened when the user clicks on the custom message label in the commit dialog. You can also just point the user to a web page instead of the MSI file directly. The URL is opened with the default web browser, so if you specify a web page, that page is opened and shown to the user. If you specify the MSI package, the browser will ask the user to save the MSI file locally.

C.3. Alternado `SVN_ASP_DOT_NET_HACK` variável do ambiente

As of version 1.4.0 and later, the TortoiseSVN installer doesn't provide the user with the option to set the `SVN_ASP_DOT_NET_HACK` environment variable anymore, since that caused many problems and confusion for users who always install *everything* no matter whether they know what it is for.

But the feature is still available in TortoiseSVN and other svn clients. To enable it you have to set the Windows environment variable named `ASPDOTNETHACK` to `1`. Actually, the value of that environment variable doesn't matter: if the variable exists the feature is active.

Importante

Please note that this hack is only necessary if you're still using VS.NET2002. All later versions of Visual Studio do *not* require this hack to be activated! So unless you're using that ancient tool, DO NOT USE THIS!

C.4. Desabilitar opções do menu de contexto

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.

Valor	Opção do Menu
0x0000000000000001	Obter
0x0000000000000002	Atualizar
0x0000000000000004	Submeter
0x0000000000000008	Adicionar
0x0000000000000010	Reverter
0x0000000000000020	Limpar
0x0000000000000040	Resolver
0x0000000000000080	Alternar
0x0000000000000100	Importar

Valor	Opção do Menu
0x0000000000000200	Exportar
0x0000000000000400	Criar Repositório aqui
0x0000000000000800	Ramificar/Rotular...
0x0000000000001000	Combinar
0x0000000000002000	Apagar
0x0000000000004000	Renomear
0x0000000000008000	Atualizar para Revisão
0x0000000000010000	Diff
0x0000000000020000	Log
0x0000000000040000	Conflitos
0x0000000000080000	Reposicionar
0x0000000000100000	Verificar alterações
0x0000000000200000	Ignorar
0x0000000000400000	Navegador do Repositório
0x0000000000800000	Autoria
0x0000000001000000	Criar Correção
0x0000000002000000	Aplicar correção
0x0000000004000000	Gráfico de revisões
0x0000000008000000	Bloquear
0x0000000010000000	Remover bloqueio
0x0000000020000000	Propriedades
0x0000000040000000	Comparar com URL
0x0000000080000000	Excluir não versionados
0x0000000100000000	Mesclar tudo
0x0000000200000000	Diferença com a revisão anterior
0x0000000400000000	Colar
0x0000000800000000	Atualiza a cópia de trabalho
0x0000001000000000	Diferenças mais tarde
0x0000002000000000	Comparar com 'filename'
0x0000004000000000	Unified diff
0x2000000000000000	Configurações
0x4000000000000000	Ajuda
0x8000000000000000	Sobre

Tabela C.1. Entradas do menu e seus valores

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.

Apêndice D. Automatizando o TortoiseSVN

Since all commands for TortoiseSVN are controlled through command line parameters, you can automate it with batch scripts or start specific commands and dialogs from other programs (e.g. your favourite text editor).

Importante

Remember that TortoiseSVN is a GUI client, and this automation guide shows you how to make the TortoiseSVN dialogs appear to collect user input. If you want to write a script which requires no input, you should use the official Subversion command line client instead.

D.1. Comandos do TortoiseSVN

The TortoiseSVN GUI program is called `TortoiseProc.exe`. All commands are specified with the parameter `/command:abcd` where `abcd` is the required command name. Most of these commands need at least one path argument, which is given with `/path:"some\path"`. In the following table the command refers to the `/command:abcd` parameter and the path refers to the `/path:"some\path"` parameter.

There's a special command that does not require the parameter `/command:abcd` but, if nothing is specified on the command line, starts the project monitor instead. If `/tray` is specified, the project monitor starts hidden and only adds its icon to the system tray.

Since some of the commands can take a list of target paths (e.g. committing several specific files) the `/path` parameter can take several paths, separated by a `*` character.

You can also specify a file which contains a list of paths, separated by newlines. The file must be in UTF-16 format, without a *BOM* [https://en.wikipedia.org/wiki/Byte-order_mark]. If you pass such a file, use `/pathfile` instead of `/path`. To have TortoiseProc delete that file after the command is finished, you can pass the parameter `/deletepathfile`. If you don't pass `/deletepathfile`, you have to delete the file yourself or the file gets left behind.

The progress dialog which is used for commits, updates and many more commands usually stays open after the command has finished until the user presses the OK button. This can be changed by checking the corresponding option in the settings dialog. But using that setting will close the progress dialog, no matter if you start the command from your batch file or from the TortoiseSVN context menu.

To specify a different location of the configuration file, use the parameter `/configdir:"path\to\config\directory"`. This will override the default path, including any registry setting.

To close the progress dialog at the end of a command automatically without using the permanent setting you can pass the `/closeonend` parameter.

- `/closeonend:0` não fechar o diálogo automaticamente
- `/closeonend:1` fechar automaticamente se não houver erros
- `/closeonend:2` fechar automaticamente se não houver erros e conflitos
- `/closeonend:3` fechar automaticamente se não houver erros, conflitos e combinações

To close the progress dialog for local operations if there were no errors or conflicts, pass the `/closeforlocal` parameter.

The table below lists all the commands which can be accessed using the TortoiseProc.exe command line. As described above, these should be used in the form `/command:abcd`. In the table, the `/command` prefix is omitted to save space.

Comando	Descrição
:about	Shows the about dialog. This is also shown if no command is given.

:log	Opens the log dialog. The `/path` specifies the file or folder for which the log should be shown. Additional options can be set:
	• `/startrev:xxx`,
	• `/endrev:xxx`,
	• `/strict` enables the 'stop-on-copy' checkbox,
	• `/merge` enables the 'include merged revisions' checkbox,
	• `/datemin:"{datestring}"` sets the start date of the filter, and
	• `/datemax:"{datestring}"` sets the end date of the filter. The date format is the same as used for svn date revisions.
	• `/findstring:"filterstring"` fills in the filter text,
	• `/findtext` forces the filter to use text, not regex, or
	• `/findregex` forces the filter to use regex, not simple text search, and
	• `/findtype:X` with X being a number between 0 and 511. The numbers are the sum of the following options:
	  • `/findtype:0` filtrar por tudo
	  • `/findtype:1` filtrar por mensagens
	  • `/findtype:2` filtrar pelo caminho
	  • `/findtype:4` filter by authors
	  • `/findtype:8` filtrar por revisão
	  • `/findtype:16` não usado
	  • `/findtype:32` filtrar pelo ID do erro
	  • `/findtype:64` não usado
	  • `/findtype:128` filtrar por data
	  • `/findtype:256` filter by date range
	• If `/outfile:path\to\file` is specified, the selected revisions are written to that file when the log dialog is closed. The revisions are written in the same format as is used to specify revisions in the merge dialog.
	An svn date revision can be in one of the following formats:
	• {2006-02-17}
	• {15:30}
	• {15:30:00.200000}
	• {"2006-02-17 15:30"}

Comando	Descrição
	• {"2006-02-17 15:30 +0230"} • {2006-02-17T15:30} • {2006-02-17T15:30Z} • {2006-02-17T15:30-04:00} • {20060217T1530} • {20060217T1530Z} • {20060217T1530-0500}
:checkout	Opens the checkout dialog. The `/path` specifies the target directory and the `/url` specifies the URL to checkout from. If you specify the key `/blockpathadjustments`, the automatic checkout path adjustments are blocked. The `/revision:XXX` specifies the revision to check out.
:import	Opens the import dialog. The `/path` specifies the directory with the data to import. You can also specify the `/logmsg` switch to pass a predefined log message to the import dialog. Or, if you don't want to pass the log message on the command line, use `/logmsgfile:caminho`, where `caminho` points to a file containing the log message.
:update	Updates the working copy in `/path` to HEAD. If the option `/rev` is given then a dialog is shown to ask the user to which revision the update should go. To avoid the dialog specify a revision number `/rev:1234`. Other options are `/nonrecursive`, `/ignoreexternals` and `/includeexternals`. The `/stickydepth` indicates that the specified depth should be sticky, creating a sparse checkout. The `/skipprechecks` can be set to skip all checks that are done before an update. If this is specified, then the Mostrar log button is disabled, and the context menu to show diffs is also disabled after the update.
:commit	Opens the commit dialog. The `/path` specifies the target directory or the list of files to commit. You can also specify the `/logmsg` switch to pass a predefined log message to the commit dialog. Or, if you don't want to pass the log message on the command line, use `/logmsgfile:caminho`, where `caminho` points to a file containing the log message. To pre-fill the bug ID box (in case you've set up integration with bug trackers properly), you can use the `/bugid:"o código do erro aqui"` to do that.
:add	Adds the files in `/path` to version control.
:revert	Reverts local modifications of a working copy. The `/path` tells which items to revert.
:cleanup	Cleans up interrupted or aborted operations and unlocks the working copy in `/path`. You also have to pass the `/cleanup` to actually do the cleanup. Use `/noui` to prevent the result dialog from popping up (either telling about the cleanup being finished or showing an error message). `/noprogressui` also disables the progress dialog. `/nodlg` disables showing the cleanup dialog where the user can choose what exactly should be done in the cleanup. The available actions can be specified with the options `/cleanup` for status cleanup, `/breaklocks` to break all locks, `/revert` to revert uncommitted changes, `/delunversioned`, `/delignored`, `/refreshshell`, `/externals`, `/fixtimestamps` and `/vacuum`.
:resolve	Marks a conflicted file specified in `/path` as resolved. If `/noquestion` is given, then resolving is done without asking the user first if it really should be done.
:repocreate	Cria um repositório em `/path`

Comando	Descrição
:switch	Opens the switch dialog. The `/path` specifies the target directory and `/url` the URL to switch to.
:export	Exports the working copy in `/path` to another directory. If the `/path` points to an unversioned directory, a dialog will ask for an URL to export to the directory in `/path`. If you specify the key `/blockpathadjustments`, the automatic export path adjustments are blocked.
:dropexport	Exports the working copy in `/path` to the directory specified in `/droptarget`. This exporting does not use the export dialog but executes directly. The option `/overwrite` specifies that existing files are overwritten without user confirmation, and the option `/autorename` specifies that if files already exist, the exported files get automatically renamed to avoid overwriting them. The option `/extended` can specify either `alterações locais` to only export files that got changed locally, or `Não versionado` to also export all unversioned items as well.
:dropvendor	Copies the folder in `/path` recursively to the directory specified in `/droptarget`. New files are added automatically, and missing files get removed in the target working copy, basically ensuring that source and destination are exactly the same. Specify `/noui` to skip the confirmation dialog, and `/noprogressui` to also disable showing the progress dialog.
:merge	Opens the merge dialog. The `/path` specifies the target directory. For merging a revision range, the following options are available: `/fromurl:URL`, `/revrange:texto`. For merging two repository trees, the following options are available: `/fromurl:URL`, `/tourl:URL`, `/fromrev:xxx` and `/torev:xxx`.
:mergeall	Opens the merge all dialog. The `/path` specifies the target directory.
:copy	Brings up the branch/tag dialog. The `/path` is the working copy to branch/tag from. And the `/url` is the target URL. If the urls starts with a ^ it is assumed to be relative to the repository root. To already check the option `Switch working copy to new branch/tag` you can pass the `/switchaftercopy` switch. To check the option `Criar pastas intermediárias` pass the `/makeparents` switch. You can also specify the `/logmsg` switch to pass a predefined log message to the branch/tag dialog. Or, if you don't want to pass the log message on the command line, use `/logmsgfile:caminho`, where `caminho` points to a file containing the log message.
:settings	Abre a janela de configurações.
:remove	Removes the file(s) in `/path` from version control.
:rename	Renames the file in `/path`. The new name for the file is asked with a dialog. To avoid the question about renaming similar files in one step, pass `/noquestion`.
:diff	Starts the external diff program specified in the TortoiseSVN settings. The `/path` specifies the first file. If the option `/path2` is set, then the diff program is started with those two files. If `/path2` is omitted, then the diff is done between the file in `/path` and its BASE. If the specified file also has property modifications, the external diff tool is also started for each modified property. To prevent that, pass the option `/ignoreprops`. To explicitly set the revision numbers use `/startrev:xxx` and `/endrev:xxx`, and for the optional peg revision use `/pegrevision:xxx`. If `/blame` is set and `/path2` is not set, then the diff is done by first blaming the files with the given revisions. The parameter `/line:xxx` specifies the line to jump to when the diff is shown.
:shelve	Shelves the specified paths in a new shelf. The option `/shelfname:name` specifies the name of the shelf. An optional log message can be specified with `/logmsg:message`. If option `/checkpoint` is passed, the modifications of the files are kept.

Comando	Descrição
:unshelve	Applies the shelf with the name `/shelfname:name` to the working copy path. By default the last version of the shelf is applied, but you can specify a version with `/version:X`.
: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. If the specified url also has property modifications, the external diff tool is also started for each modified property. To prevent that, pass the option `/ignoreprops`. If a unified diff is requested, an optional `prettyprint` option can be specified which will show the merge-info properties in a more user readable format.
:conflicteditor	Starts the conflict editor specified in the TortoiseSVN settings with the correct files for the conflicted file in `/path`.
:relocate	Opens the relocate dialog. The `/path` specifies the working copy path to relocate.
:help	Abre o arquivo de ajuda.
:repostatus	Opens the check-for-modifications dialog. The `/path` specifies the working copy directory. If `/remote` is specified, the dialog contacts the repository immediately on startup, as if the user clicked on the Verificar o repositório button.
:repobrowser	Starts the repository browser dialog, pointing to the URL of the working copy given in `/path` or `/path` points directly to an URL. An additional option `/rev:xxx` can be used to specify the revision which the repository browser should show. If the `/rev:xxx` is omitted, it defaults to HEAD. If `/path` points to an URL, the `/projectpropertiespath:path/to/wc` specifies the path from where to read and use the project properties. If `/outfile:path\to\file` is specified, the selected URL and revision are written to that file when the repository browser is closed. The first line in that text file contains the URL, the second line the revision in text format.
:ignore	Adds all targets in `/path` to the ignore list, i.e. adds the `svn:ignore` property to those files.
:blame	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	Saves a file from an URL or working copy path given in `/path` to the location given in `/savepath:caminho`. The revision is given in `/revision:xxx`. This can be used to get a file with a specific revision.
:createpatch	Creates a patch file for the path given in `/path`. To skip the file Save-As dialog you can pass `/savepath:caminho` to specify the path where to save the patch

Comando	Descrição
	file to directly. To prevent the unified diff viewer from being started showing the patch file, pass `/noview`. If a unified diff is requested, an optional `prettyprint` option can be specified which will show the merge-info properties in a more user readable format.
:revisiongraph	Mostra o gráfico de revisões para o caminho informado em `/path`. To create an image file of the revision graph for a specific path, but without showing the graph window, pass `/output:path` with the path to the output file. The output file must have an extension that the revision graph can actually export to. These are: `.svg`, `.wmf`, `.png`, `.jpg`, `.bmp` and `.gif`. Since the revision graph has many options that affect how it is shown, you can also set the options to use when creating the output image file. Pass these options with `/options:XXXX`, where `XXXX` is a decimal value. The best way to find the required options is to start the revision graph the usual way, set all user-interface options and close the graph. Then the options you need to pass on the command line can be read from the registry `HKCU\Software\TortoiseSVN\RevisionGraphOptions`.
:lock	Locks a file or all files in a directory given in `/path`. The 'lock' dialog is shown so the user can enter a comment for the lock.
:unlock	destrava um ou todos os arquivos em dado diretório `/path`.
:rebuildiconcache	Rebuilds the windows icon cache. Only use this in case the windows icons are corrupted. A side effect of this (which can't be avoided) is that the icons on the desktop get rearranged. To suppress the message box, pass `/noquestion`.
:properties	Shows the properties dialog for the path given in `/path`. For dealing with versioned properties this command requires a working copy. Revision properties can be viewed/changed if `/path` is an URL and `/rev:XXX` is specified. To open the properties dialog directly for a specific property, pass the property name as `/property:name`.
:sync	Exports/imports settings, either depending on whether the current settings or the exported settings are newer, or as specified. If a path is passed with `/path`, then the path is used to store or read the settings from. The parameter `/askforpath` will show a file open/save dialog for the user to chose the export/import path. If neither `/load` nor `/save` is specified, then TortoiseSVN determines whether to export or import the settings by looking at which ones are more recent. If the export file is more recent than the current settings, then the settings are loaded from the file. If the current settings are more recent, then the settings are exported to the settings file. If `/load` is specified, the settings are imported from the settings file. If `/save` is specified, the current settings are exported to the settings file.

Comando	Descrição
	The parameter `/local` forces a settings export to include local settings, i.e. settings that refer to local paths.

Tabela D.1. Lista de comandos e opções disponíveis

Examples (which should be entered on one line):

```
TortoiseProc.exe /command:commit
                 /path:"c:\svn_wc\file1.txt*c:\svn_wc\file2.txt"
                 /logmsg:"test log message" /closeonend:0

TortoiseProc.exe /command:update /path:"c:\svn_wc\" /closeonend:0

TortoiseProc.exe /command:log /path:"c:\svn_wc\file1.txt"
                 /startrev:50 /endrev:60 /closeonend:0
```

D.2. Controlador de URL do Tsvncmd

Using special URLs, it is also possible to call TortoiseProc from a web page.

TortoiseSVN registers a new protocol `tsvncmd:` which can be used to create hyperlinks that execute TortoiseSVN commands. The commands and parameters are the same as when automating TortoiseSVN from the command line.

The format of the `tsvncmd:` URL is like this:

`tsvncmd:command:cmd?parameter:paramvalue?parameter:paramvalue`

with `cmd` being one of the allowed commands, `parameter` being the name of a parameter like `path` or `revision`, and `paramvalue` being the value to use for that parameter. The list of parameters allowed depends on the command used.

The following commands are allowed with `tsvncmd:` URLs:

- `:update`
- `:commit`
- `:diff`
- `:repobrowser`
- `:checkout`
- `:export`
- `:blame`
- `:repostatus`
- `:revisiongraph`
- `:showcompare`
- `:log`

A simple example URL might look like this:

```
<a href="tsvncmd:command:update?path:c:\svn_wc?rev:1234">Update</a>
```

or in a more complex case:

```
<a href="tsvncmd:command:showcompare?
url1:https://svn.code.sf.net/p/stefanstools/code/trunk/StExBar/src/setup/Setup.wxs?
url2:https://svn.code.sf.net/p/stefanstools/code/trunk/StExBar/src/setup/Setup.wxs?
revision1:188?revision2:189">compare</a>
```

D.3. Comandos do TortoiseIDiff

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.

Opção	Descrição
:left	Caminho para o arquivo mostrado a esquerda.
: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.

Tabela D.2. Lista de opções disponíveis

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
```

D.4. Comandos TortoiseUDiff

The unified diff viewer has only two command line options:

Opção	Descrição
:patchfile	Path to the unified diff file.
:p	Activates pipe mode. The unified diff is read from the console input.

Tabela D.3. Lista de opções disponíveis

Examples (which should be entered on one line):

```
TortoiseUDiff.exe /patchfile:"c:\diff.patch"
```

If you create the diff from another command, you can use TortoiseUDiff to show that diff directly:

```
svn diff | TortoiseUDiff.exe /u
```

this also works if you omit the `/p` parameter:

```
svn diff | TortoiseUDiff.exe
```

Apêndice E. Command Line Interface Cross Reference

Sometimes this manual refers you to the main Subversion documentation, which describes Subversion in terms of the Command Line Interface (CLI). To help you understand what TortoiseSVN is doing behind the scenes, we have compiled a list showing the equivalent CLI commands for each of TortoiseSVN's GUI operations.

> **Nota**
>
> Even though there are CLI equivalents to what TortoiseSVN does, remember that TortoiseSVN does *not* call the CLI but uses the Subversion library directly.

If you think you have found a bug in TortoiseSVN, we may ask you to try to reproduce it using the CLI, so that we can distinguish TortoiseSVN issues from Subversion issues. This reference tells you which command to try.

E.1. Convenções e regras básicas

In the descriptions which follow, the URL for a repository location is shown simply as `URL`, and an example might be `https://svn.code.sf.net/p/tortoisesvn/code/trunk/`. The working copy path is shown simply as `PATH`, and an example might be `C:\TortoiseSVN\trunk`.

> **Importante**
>
> Because TortoiseSVN is a Windows Shell Extension, it is not able to use the notion of a current working directory. All working copy paths must be given using the absolute path, not a relative path.

Certain items are optional, and these are often controlled by checkboxes or radio buttons in TortoiseSVN. These options are shown in [square brackets] in the command line definitions.

E.2. Comandos do TortoiseSVN

E.2.1. Obter

```
svn checkout [-depth ARG] [--ignore-externals] [-r rev] URL PATH
```

The depth combo box items relate to the `-depth` argument.

If Omit externals is checked, use the `--ignore-externals` switch.

If you are checking out a specific revision, specify that after the URL using `-r` switch.

E.2.2. Atualizar

```
svn info URL_of_WC
svn update [-r rev] PATH
```

Updating multiple items is currently not an atomic operation in Subversion. So TortoiseSVN first finds the HEAD revision of the repository, and then updates all items to that particular revision number to avoid creating a mixed revision working copy.

If only one item is selected for updating or the selected items are not all from the same repository, TortoiseSVN just updates to HEAD.

No command line options are used here. Update to revision also implements the update command, but offers more options.

E.2.3. Atualizar para Revisão

```
svn info URL_of_WC
svn update [-r rev] [-depth ARG] [--ignore-externals] PATH
```

The depth combo box items relate to the `-depth` argument.

If Omit externals is checked, use the `--ignore-externals` switch.

E.2.4. Submeter

In TortoiseSVN, the commit dialog uses several Subversion commands. The first stage is a status check which determines the items in your working copy which can potentially be committed. You can review the list, diff files against BASE and select the items you want to be included in the commit.

```
svn status -v PATH
```

If Show unversioned files is checked, TortoiseSVN will also show all unversioned files and folders in the working copy hierarchy, taking account of the ignore rules. This particular feature has no direct equivalent in Subversion, as the `svn status` command does not descend into unversioned folders.

If you check any unversioned files and folders, those items will first be added to your working copy.

```
svn add PATH...
```

When you click on OK, the Subversion commit takes place. If you have left all the file selection checkboxes in their default state, TortoiseSVN uses a single recursive commit of the working copy. If you deselect some files, then a non-recursive commit (`-N`) must be used, and every path must be specified individually on the commit command line.

```
svn commit -m "LogMessage" [-depth ARG] [--no-unlock] PATH...
```

`LogMessage` here represents the contents of the log message edit box. This can be empty.

If Keep locks is checked, use the `--no-unlock` switch.

E.2.5. Diff

```
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.

You can also diff any 2 files using TortoiseSVN, whether or not they are version controlled. TortoiseSVN just feeds the two files into the chosen diff program and lets it work out where the differences lie.

E.2.6. Log

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

By default, TortoiseSVN tries to fetch 100 log messages using the --limit method. If the settings instruct it to use old APIs, then the second form is used to fetch the log messages for 100 repository revisions.

If Stop on copy/rename is checked, use the `--stop-on-copy` switch.

E.2.7. Procurar por Modificações

```
svn status -v PATH
or
svn status -u -v PATH
```

The initial status check looks only at your working copy. If you click on Check repository then the repository is also checked to see which files would be changed by an update, which requires the `-u` switch.

If Show unversioned files is checked, TortoiseSVN will also show all unversioned files and folders in the working copy hierarchy, taking account of the ignore rules. This particular feature has no direct equivalent in Subversion, as the `svn status` command does not descend into unversioned folders.

E.2.8. Gráfico de Revisões

O gráfico de revisões é uma característica apenas do TortoiseSVN. Não existe comando equivalente para a linha de comando.

What TortoiseSVN does is an

```
svn info URL_of_WC
svn log -v URL
```

where URL is the repository *root* and then analyzes the data returned.

E.2.9. Navegador de

```
svn info URL_of_WC
svn list [-r rev] -v URL
```

You can use `svn info` to determine the repository root, which is the top level shown in the repository browser. You cannot navigate Up above this level. Also, this command returns all the locking information shown in the repository browser.

The `svn list` call will list the contents of a directory, given a URL and revision.

E.2.10. Conflitos

This command has no CLI equivalent. It invokes TortoiseMerge or an external 3-way diff/merge tool to look at the files involved in the conflict and sort out which lines to use.

E.2.11. Resolvido

```
svn resolved PATH
```

E.2.12. Renomear

```
svn rename CURR_PATH NEW_PATH
```

E.2.13. Apagar

```
svn delete PATH
```

E.2.14. Reverter

```
svn status -v PATH
```

The first stage is a status check which determines the items in your working copy which can potentially be reverted. You can review the list, diff files against BASE and select the items you want to be included in the revert.

When you click on OK, the Subversion revert takes place. If you have left all the file selection checkboxes in their default state, TortoiseSVN uses a single recursive (`-R`) revert of the working copy. If you deselect some files, then every path must be specified individually on the revert command line.

```
svn revert [-R] PATH...
```

E.2.15. Limpar

```
svn cleanup PATH
```

E.2.16. Obter bloqueio

```
svn status -v PATH
```

The first stage is a status check which determines the files in your working copy which can potentially be locked. You can select the items you want to be locked.

```
svn lock -m "Mensagem de bloqueio" [--force] PATH...
```

`LockMessage` here represents the contents of the lock message edit box. This can be empty.

If Steal the locks is checked, use the `--force` switch.

E.2.17. Liberar bloqueio

```
svn unlock PATH
```

E.2.18. Ramificar/Rotular...

```
svn copy -m "LogMessage" URL URL
or
svn copy -m "LogMessage" URL@rev URL@rev
or
svn copy -m "LogMessage" PATH URL
```

The Branch/Tag dialog performs a copy to the repository. There are 3 radio button options:

- Última revisão no repositório
- especC which correspond to the 3 command line variants above.

`LogMessage` here represents the contents of the log message edit box. This can be empty.

E.2.19. Alternar

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

E.2.20. Combinar

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

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

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

The Unified diff shows the diff operation which will be used to do the merge.

E.2.21. Exportar

```
svn export [-r rev] [--ignore-externals] URL Export_PATH
```

This form is used when accessed from an unversioned folder, and the folder is used as the destination.

Exporting a working copy to a different location is done without using the Subversion library, so there's no matching command line equivalent.

What TortoiseSVN does is to copy all files to the new location while showing you the progress of the operation. Unversioned files/folders can optionally be exported too.

In both cases, if Omit externals is checked, use the `--ignore-externals` switch.

E.2.22. Reposicionar

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

E.2.23. Criar repositó aqui

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

E.2.24. Adicionar

```
svn add PATH...
```

Se você selecionou uma pasta, o TortoiseSVN inicia fazendo uma busca recursiva nela por ítens que podem ser adicionados.

E.2.25. Importar

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

`LogMessage` here represents the contents of the log message edit box. This can be empty.

E.2.26. Autoria

```
svn blame -r N:M -v PATH
svn log -r N:M PATH
```

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

E.2.27. Adicionar à lista de ignorados

```
svn propget svn:ignore PATH > tempfile
{edit new ignore item into tempfile}
svn propset svn:ignore -F tempfile PATH
```

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

E.2.28. Criar Correção

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

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

E.2.29. Aplicar correção

Applying patches is a tricky business unless the patch and working copy are at the same revision. Luckily for you, you can use TortoiseMerge, which has no direct equivalent in Subversion.

Apêndice F. Detalhes da Implementação

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

F.1. Sobreposição dos Ícones

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.

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.

Since there are Tortoise clients available for other version control systems, we've created a shared component which is responsible for showing the overlay icons. The technical details are not important here, all you need to know is that this shared component allows all Tortoise clients to use the same overlays and therefore the limit of 11 available slots isn't used up by installing more than one Tortoise client. Of course there's one small drawback: all Tortoise clients use the same overlay icons, so you can't figure out by the overlay icons what version control system a working copy is using.

- *Normal*, *Modified* and *Conflicted* are always loaded and visible.
- *Deleted* is loaded if possible, but falls back to *Modified* if there are not enough slots.
- *Read-Only* is loaded if possible, but falls back to *Normal* if there are not enough slots.
- *Locked* is loaded if possible, but falls back to *Normal* if there are not enough slots.
- *Added* is loaded if possible, but falls back to *Modified* if there are not enough slots.

Apêndice G. Pacotes de Idiomas e Corretores Ortográficos

O instalador padrão tem suporte apenas para o Inglês, porém você pode fazer o download de pacotes de idiomas e corretores ortográficos separadamente após a instalação.

G.1. Pacotes de Idioma

The TortoiseSVN user interface has been translated into many different languages, so you may be able to download a language pack to suit your needs. You can find the language packs on our *translation status page* [https://tortoisesvn.net/translation_status_dev.html]. And if there is no language pack available, why not join the team and submit your own translation ;-)

Each language pack is packaged as a `.msi` installer. Just run the install program and follow the instructions. After the installation finishes, the translation will be available.

The documentation has also been translated into several different languages. You can download translated manuals from the *support page* [https://tortoisesvn.net/support.html] on our website.

G.2. Corretor ortográfico

TortoiseSVN uses the Windows spell checker if it's available (Windows 8 or later). Which means that if you want the spell checker to work in a different language than the default OS language, you have to install the spell checker module in the Windows settings (`Settings > Time & Language > Region & Language`).

TortoiseSVN will use that spell checker if properly configured with the `tsvn:projectlanguage` project property.

In case the Windows spell checker is not available, TortoiseSVN can also use spell checker dictionaries from *OpenOffice* [https://openoffice.org] and *Mozilla* [https://mozilla.org].

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

Or you can install the dictionaries yourself. If you have OpenOffice or Mozilla installed, you can copy those dictionaries, which are located in the installation folders for those applications. Otherwise, you need to download the required dictionary files from http://wiki.services.openoffice.org/wiki/Dictionaries.

Once you have got the dictionary files, you probably need to rename them so that the filenames only have the locale chars in it. Example:

- `en_US.aff`
- `en_US.dic`

Then just copy them into the `%APPDATA%\TortoiseSVN\dic` folder. If that folder isn't there, you have to create it first. TortoiseSVN will also search the `Languages` sub-folder of the TortoiseSVN installation folder (normally this will be `C:\Program Files\TortoiseSVN\Languages`); this is the place where the language packs put their files. However, the %APPDATA%-folder doesn't require administrator privileges and, thus, has higher priority. The next time you start TortoiseSVN, the spell checker will be available.

Se você instalar vários dicionários, TortoiseSVN usa essas regras para selecionar uma para usar.

1. Verifique a configuração `tsvn:projectlanguage`.Veja Seção 4.18, “Configurações do Projeto” para informações sobre a configuração.

2. Se o idioma do projeto não estiver configurado, ou o idioma não estiver instalado, tente o idioma correspondente da configuração do Windows.

3. If the exact Windows locale doesn't work, try the “Base” language, e.g. `de_CH` (Swiss-German) falls back to `de_DE` (German).

4. Se nada disso funcionar, então o idioma padrão é o Inglês, que está incluido na instalação normal.

Glossário

Adicionar	Um comando do Subversion que é usado para adicionar um arquivo ou diretório para a sua cópia de trabalho. Os novos itens são adicionados para o repositório quando você submetê-los.
Alternar	Assim como “Atualizar para a revisão” altera a versão de uma cópia de trabalho apontando para um ponto diferente no histórico, assim também “Alternar” altera o espaço de uma cópia de trabalho que aponta para uma parte diferente do repositório. Isto é particularmente útil quando se está trabalhando no tronco e na ramificação e apenas poucos arquivos são diferentes. Você pode alternar sua cópia de trabalho entre os dois e somente os arquivos diferentes serão transferidos.
Apagar	Quando você excluir um item controlado (e submeter a alteração) este item não existirá mais no repositório depois da revisão submetida. Mas é claro que continuará a existir em revisões anteriores no repositório, logo você pode continuar acessando o arquivo. Se necessário, você pode copiar um item excluído e “recuperar” o mesmo completamente com o histórico.
Atualizar	Este comando do Subversion baixa as últimas alterações do repositório para a sua cópia de trabalho, combinando qualquer alteração feita por outros usuários com suas alterações locais na cópia de trabalho.
Autoria	Este comando é apenas para arquivos texto, e ele apresenta para cada lista a revisão do repositório a qual pertence a última alteração, e o autor que fez a alteração. Nossa implementação de interface é chamada TortoiseBlame e também mostra a data/hora da submissão e a mensagem de log quando você passar o mouse sobre o número da revisão.
Bloquear	Quando você bloqueia um item controlado, você marca o item no repositório como um arquivo não atualizável, exceto para a cópia de trabalho que bloqueiou o arquivo.
Combinar	O processo pelo qual as mudanças do repositório são adicionadas na sua cópia de trabalho sem perder qualquer alteração que você tenha feita localmente. Algumas vezes essas alterações não podem ser combinar automaticamente e a cópia de trabalho indicará que há conflito. Combinar acontece automaticamente quando você atualiza sua cópia de trabalho. Você pode também combinar mudanças específicas de outra ramificação usando o comando Combinar do TortoiseSVN.
Conflito	Quando as alterações do repositório são combinadas com as alterações locais, algumas vezes estas alterações ocorrem nas mesmas linhas. Neste caso Subversion não pode decidir automaticamente qual versão usar e o arquivo é marcado como em conflito. Você precisa editar este arquivo manualmente e resolver o conflito antes que você possa submeter qualquer alteração adicional.
Cópia de Trabalho	Está a sua estrutura local “isolada”, e é onde você vai trabalhar sobre os arquivos controlados, e isto normalmente ficará em seu disco local. Você cria uma cópia de trabalho executando um “Obter” de um repositório, e você envia suas alterações de volta para o repositório executando um “Submeter”.
Copiar	Em um repositório do Subversion você pode criar uma cópia de de um arquivo apenas ou de uma estrutura inteira. Estas cópias são feitas como “cópias levess> que funcionam mais ou menos como um atalho para o arquivo original, e que consome quase nenhum espa”

Correção	Se uma cópia de trabalho possui alterações apenas em arquivos texto, é possível usar o comando Diferenças do Subversion para gerar um arquivo simples de resumo dessas alterações no formato Unified Diff. Um arquivo deste tipo é oferecido como “Correção”, e ele pode ser enviado por e-mail para alguém (ou para uma lista de discussão) e ser aplicado em outra cópia de trabalho. Alguém sem acesso para submissão pode fazer alterações e enviar um arquivo de correção para alguém com permissão de escrita submeter as alterações. Ou se você não tem certeza sobre a alteração você pode enviar o arquivo de correção para outros revisarem.
Diff	Atalho para “Mostrar Diferenças”. Muito prático quando você quer ver exatamente as alterações que foram feitas.
Exportar	Este comando produz uma cópia de uma diretório controlado, assim como uma cópia de trabalho, mas sem os diretórios locais `.svn`.
FSFS	Um sistema de arquivo proprietário do Subversion para repositórios. Pode ser usado em uma rede compartilhada. O padrão para a versão do repositório 1.2 ou posterior.
GPO	Objeto da política do grupo.
Histórico	Mostra o histórico de revisões de um arquivo ou diretório. Também conhecido como “Log”.
Importar	Comando do Subversion para importar um diretório e sua subestrutura inteira para o repositório em uma única revisão.
Limpar	Uma citação do livro do Subversion: “Limpar recursivamente a cópia de trabalho, removendo travas e encerrando operação não finalizadas. Se você sempre recebe o erro *cópia de trabalho travada*, execute este comando para remove travas obsoletas e voltar a sua cópia de trabalho para o estado normal novamente. ” Note que neste contexto, *trava* refere-se ao travamento do sistema de arquivos, e não ao bloqueio no repositório.
Log	Mostra o histórico de revisão de um arquivo ou diretório. Também conhecido como “Histórico”.
Obter	Um comando do Subversion que cria uma cópia local em uma diretório vazio baixando os arquivos controlados do repositório.
Propriedade	Como complemento para versionamento de diretórios e arquivos, Subversion permite adicionar metadados controlados - referenciado como “propriedades” para cada diretório ou arquivo controlado. Cada propriedade tem um nome e um valor, assim como uma chave de registro. Subversion tem algumas propriedades especiais que são usadas internamente, tal como `svn:eol-style`. TortoiseSVN tem algumas também, tal como `tsvn:logminsize`. Você pode adicionar suas próprias propriedades com qualquer nome ou valor que você queira.
Propriedade da Revisão (revprop)	Assim como os arquivo possuem propriedades, assim também é para cada revisão no repositório. Algumas revprops especiais são adicionadas automaticamente quando a revisão é criada, chamadas: `svn:date svn:author svn:log` que representam a data/hora, o autor e a mensagem de log da submissão respectivamente. Estas propriedades podem ser editadas, mas elas não são controladas, então qualquer alteração é permanente e não pode ser desfeita.
Ramificação	Um termo frequentemente usado em sistemas de controle de versão descreve o que acontece quando o desenvolvimento se ramifica num ponto em

	particular e segue dois caminhos diferentes. Você pode criar uma ramificação desligada da linha principal de desenvolvimento, logo, pode desenvolver novas funcionalidades sem desestabilizar a linha principal. Ou você pode criar uma ramificação da versão estável para a qual você apenas fará correções de erros, enquanto insere novas funcionalidades em uma versão instável no tronco. No Subversion uma ramificação é criada como uma “cópia leve”.
Reposicionar	Se seu repositório for movido, talvez porque você precisou movê-lo para um diretório diferente no servidor, ou porque o nome do domínio do servidor foi alterado, você precisa “realocar” sua cópia de trabalho para a nova URL onde o seu repositório está localizado. Nota: você deverá somente usar este comando se sua cópia de trabalho aponta para o mesmo local no mesmo repositório, mas o repositório foi movido. Em qualquer outra circustância você provavelmente deverá usar “Alternar” ao invés deste comando.
Repositório	Um repositório é um lugar central onde os dados são guardados e mantidos. Um repositório pode ser um lugar onde várias bases de dados ou arquivos são localizados para distribuir pela rede, ou um repositório pode ser um local que é diretamente acessado pelos usuários sem ter que atravessar uma rede.
Resolver	Quando os arquivos na cópia de trabalho são deixados em estado de conflito logo depois de serem unificados, estes conflitos devem ser resolvidos por um ser humano usando um editor (ou talvez o TortoiseMerge). Este processo é referenciado como “Resolvendo Conflitos”. Quando tudo foi resolvido você pode marcar os arquivos conflitantes como resolvidos, e então será permtido submeter estes arquivos.
Reverter	Subversion mantém uma cópia local “original” de cada arquivo de como ele era quando foi feita a última atualização da cópia de trabalho. Se você fez alterações e decidir que quer desfazê-las, você pode usar o comando “reverter” para voltar para a revisão original.
Revisão	Cada vez que você submete um conjunto de alterações, você cria uma nova “revisão” no repositório. Cada revisão representa um estado da estrutura do repositório em um certo ponto de sua história. Se você quer voltar no tempo você pode examiar o repositório como ele era na revisão N. De outro ponto de vista, uma revisão pode referenciar um conjunto de mudanças que foram feitas quando a revisão foi criada.
Revisão BASE	A revisão base atual de um arquivo ou diretório na sua *cópia de trabalho*. Está é a revisão que o arquivo ou diretório está, quando a última obtenção, atualização ou submissão foi feita. A revisão BASE normalmente não é igual a ÚLTIMA revisão.
Submeter	Este comando do Subversion é usado para passar as alterações da sua cópia de trabalho de volta para o repositório, criando uma nova revisão no repositório.
SVN	Uma abreviação frequentemente usada para Subversion. O nome do protocolo personalizado do Subversion usado pelo servidor de respotório “svnserve”.
Última Revisão	A última revisão do arquivo ou diretório no *repositório*.

Índice Remissivo

Símbolos

'cópia de trabalho' não versionada, 133

A

Acesso, 19
ações server-side, 126
adicionar, 77
Adicionar arquivos no repositório, 29
anotação, 123
apagar, 81
arquivos especiais, 30
arquivos temporários, 29
arquivos/diretório não versionados, 78
arrastar direito, 26
arrastar-e-soltar, 26
atalho, 201
atualizar, 40, 198
autenticação, 27, 27
auto-props, 89
automatizando, 207, 213, 214, 214
autoria, 123

B

bloqueando, 117
bugtracker, 136
buscar mudanças, 40

C

Cache de log, 165
Caminhos UNC, 19
checkout esparso, 30
CLI, 216
clique do botão direito, 24
COM, 184, 192
COM SubWCRev interface, 189
combinar conflitos, 115
commit monitor, 181
comparando revisões, 73
comparar, 72
Comparar arquivos, 199
Comparar diretórios, 200
configurações, 142
conflito, 11, 42
conflito de estrutura, 42
Controlador de domínio, 203
Controlador de URL, 213
controle de versão, xi
cópia de segurança, 21
cópia de trabalho, 12
copiar, 105, 126
copiar arquivos, 77
correção, 121
corretor ortográfico, 224
criando o repositório, 18
Criar, 18
    TortoiseSVN, 18
criar a cópia de trabalho, 30

D

deploy, 203
desabilitar funções, 204
desfazer, 83
desfazer alteração, 199
desfazer submissão, 199
detach from repository, 201
dicionário, 224
diferenças, 72, 121
diferenças da imagem, 75
diferenças unificadas, 121

E

editar log/autor, 66
enviar alterações, 33
estatísticas, 68
eventos, 21
Eventos do cliente, 168
expandir palavras-chave, 87
explorer, xi
exportar, 133
exportar modificações, 73
extração versão, 184

F

FAQ, 197
ferramenta de diferenciação, 76
ferramentas de unificação, 76
filtro, 66

G

gerando diferenças, 51
globais, 79
GPO, 203
gráfico, 129
gráfico de revisões, 129
grupo de políticas, 203, 204

H

histórico, 55
histórico de combinações, 115
histórico de erros, 136, 136

I

IBugtraqProvider, 192
ícones, 46
ignorar, 78
importando na pasta, 30
importar, 29
instalar, 1
Interface do Windows, xi

L

limpar, 83, 85
linha de comando, 207, 214, 214
linha de comando do cliente, 216
lista de alterações, 51
Livro do Subversion, 7
log, 55
log de rastreamento de combinação, 65
louvor, 123

M

manipulador de arraste, 26
marcar liberação, 105
maximizar, 28
mensagem de log, 198
mensagem de submissão, 198
mensagem vazia, 198
mensagens de log, 55
mensagens de submissão, 55
menu do contexto, 24
Microsoft Word, 76
modificações, 49
monitoring projects, 181
movendo, 198
mover, 82
mover arquvos, 77
msi, 203
mudanças, 200

N

não versionado, 135, 201
navegador de repositório, 126
Número de versão nos arquivos, 184

O

obter, 30
Obter parcialmente, 30
opções do menu de contexto, 204

P

pacotes de idioma, 224
padrão de exclusão, 143
padrão global de arquivos ignorados, 143
padrões de filtro, 79
palavras-chave, 87
plugin, 192
project monitor, 181
Projetos ASP, 204
Projetos comuns, 200
projetos de fornecedores, 200
propriedades da revisão, 66
propriedades do projeto, 90
propriedades do Subversion, 86
Propriedades do TortoiseSVN, 90
propriedades do windows, 47

R

ramificação, 77, 105
rastreador de reportes, 136, 192
Rede Compartilhada, 19
referências externas, 102, 200
registro, 175
remote commits, 181
remover, 81
remover versionamento, 201
renomear, 82, 126, 198
renomear arquivos, 77
reorganizar, 198
reposicionar, 135
repositório, 7, 29
repositórios externos, 102
resolver, 42
retroceder, 198
reverter, 83, 199
revisão, 14, 129
revprops, 66
Rotinas para eventos, 21, 168
rotinas para eventos no servidor, 21
rótulo, 77, 105

S

servidor movido, 135, 135
servidor proxy, 157
shelve, 53
situação, 46, 49
situação da cópia de trabalho, 46
Sobrepor prioridade, 222
somente leitura, 117
sons, 142
sopreposições, 46, 222
submetendo, 33
submeter, 33
SUBST drives, 155
SubWCRev, 184
SVN_ASP_DOT_NET_HACK, 204

T

TortoiseIDiff, 75
traduções, 224
trocar, 108

U

unificação, 109
  duas árvores, 112
  intervalo de revisão, 110
unshelve, 53
URL alterada, 135
URL do repositório alterada, 135

V

verificação de atualização, 203
verificar nova versão, 203
versão, 203

versionar novos arquivos, 77
ViewVC, 141
vínculo, 22
Vínculo do TortoiseSVN, 22
vínculo externo, 22
visualizador do servidor, 126
visualizar de repositório, 141
visualizar modificações, 46
visualizar web, 141
VS2003, 204

W

website, 22
WebSVN, 141