TortoiseSVN

Windows用Subversionクライアント

Version 1.7.15

Stefan Küng
Lübbe Onken
Simon Large

TortoiseSVN: Windows用Subversionクライアント: Version 1.7.15

：Stefan Küng, Lübbe Onken, 、Simon Large
翻訳: 倉澤 望 (鍋太郎) (nabetaro @ caldron.jp), 藤本 理弘 (fujimoto@internet.ne.jp)

発行日 2014/08/09 17:23:48 (r25753)

目次

序章 xiii
1. TortoiseSVNとは xiii
2. TortoiseSVNの特徴 xiii
3. ライセンス xiv
4. 開発 xv
4.1. TortoiseSVN の歴史 xv
4.2. 謝辞 xv
5. このガイドの読み方 xv
6. 本書で使用している表現 xvi
1. さぁはじめましょう 1
1.1. TortoiseSVN のインストール 1
1.1.1. 必要なシステム 1
1.1.2. インストール 1
1.2. バージョン管理の基本概念 1
1.3. 試してみましょう 2
1.3.1. リポジトリの作成 2
1.3.2. プロジェクトのインポート 3
1.3.3. 作業コピーのチェックアウト 3
1.3.4. ファイルの変更 4
1.3.5. ファイルの追加 4
1.3.6. プロジェクトの変更履歴を見る 4
1.3.7. 変更を取り消す 5
1.4. さあ使ってみよう 5
2. バージョン管理の基本概念 7
2.1. リポジトリ 7
2.2. バージョン管理モデル 7
2.2.1. ファイル共有の問題 8
2.2.2. ロック・変更・アンロックモデル 8
2.2.3. コピー・変更・マージモデル 9
2.2.4. Subversionではどうしているのか 11
2.3. Subversionの動作 12
2.3.1. 作業コピー 12
2.3.2. リポジトリ URL 14
2.3.3. リビジョン 14
2.3.4. 作業コピーのリポジトリ追跡方法 16
2.4. まとめ 16
3. リポジトリ 18
3.1. リポジトリの作成 18
3.1.1. コマンドラインクライアントを使用したリポジトリの作成 18
3.1.2. TortoiseSVNを使用したリポジトリの作成 18
3.1.3. リポジトリへのローカルアクセス 19
3.1.4. ネットワークフォルダー上のリポジトリへのアクセス 20
3.1.5. リポジトリのレイアウト 20
3.2. リポジトリのバックアップ 22
3.3. サーバー側フックスクリプト 22
3.4. チェックアウトリンク 23

3.5. リポジトリへのアクセス 23
4. 日常の使用ガイド 25
4.1. 機能概要 25
4.1.1. アイコンオーバーレイ 25
4.1.2. コンテキストメニュー 25
4.1.3. ドラッグ&ドロップ 27
4.1.4. 共通のショートカット 28
4.1.5. 認証 28
4.1.6. ウィンドウの最大化 29
4.2. リポジトリへのデータのインポート 29
4.2.1. インポート 29
4.2.2. その場でインポート 31
4.2.3. 特殊なファイル 31
4.3. 作業コピーのチェックアウト 31
4.3.1. チェックアウトの深さ 32
4.4. 変更のリポジトリへのコミット 34
4.4.1. コミットダイアログ 34
4.4.2. 変更リスト 37
4.4.3. コミット一覧からの項目の除外 37
4.4.4. コミットログメッセージ 37
4.4.5. コミットの進行状況 39
4.5. 他人の変更に伴う作業コピーの更新 40
4.6. 競合の解決 41
4.6.1. ファイルの競合 42
4.6.2. プロパティの競合 43
4.6.3. ツリーの競合 43
4.7. ステータス情報の取得 46
4.7.1. アイコンオーバーレイ 46
4.7.2. 詳細なステータス 48
4.7.3. Windows エクスプローラーの TortoiseSVN 列 48
4.7.4. ローカルとリモートの状態 49
4.7.5. 差分の表示 51
4.8. 変更リスト 51
4.9. リビジョンログダイアログ 53
4.9.1. リビジョンログダイアログの起動 53
4.9.2. リビジョンログのアクション 54
4.9.3. 追加情報の取得 54
4.9.4. ログメッセージの追加取得 59
4.9.5. 現在の作業コピーのリビジョン 60
4.9.6. マージ追跡機能 60
4.9.7. ログメッセージや作者の変更 61
4.9.8. ログメッセージの絞り込み 61
4.9.9. 統計情報 63
4.9.10. オフラインモード 66
4.9.11. 表示の更新 66
4.10. 差分の表示 66
4.10.1. ファイルの差分 67
4.10.2. 改行コードと空白のオプション 68

4.10.3. フォルダーの比較 68
4.10.4. TortoiseIDiff を使用した画像の差分 69
4.10.5. Office ドキュメントの差分 70
4.10.6. 外部差分・マージツール 70
4.11. 新しいファイルやディレクトリの追加 71
4.12. ファイルやフォルダーのコピー・移動・名前の変更 72
4.13. ファイルやディレクトリの無視 73
4.13.1. 無視リストでのパターンマッチ 74
4.14. 削除、移動、名前変更 75
4.14.1. ファイルやフォルダーの削除 75
4.14.2. ファイルやフォルダーの移動 76
4.14.3. ファイル名の大文字・小文字が競合した場合の対処 77
4.14.4. ファイル名の変更の修復 77
4.14.5. バージョン管理外のファイルの削除 77
4.15. 変更の取り消し 78
4.16. クリーンアップ 79
4.17. プロジェクト設定 80
4.17.1. Subversion のプロパティ 80
4.17.2. TortoiseSVN のプロジェクトプロパティ 83
4.17.3. プロパティエディター 86
4.18. 外部項目 90
4.18.1. 外部フォルダー 91
4.18.2. 外部ファイル 92
4.19. ブランチ/タグの作成 93
4.19.1. ブランチ/タグの作成 93
4.19.2. ブランチやタグを作成するその他の方法 95
4.19.3. チェックアウトするか切り替えるか... 96
4.20. マージ 97
4.20.1. リビジョン範囲のマージ 98
4.20.2. ブランチの再統合 99
4.20.3. 2つの異なるツリーをマージする 100
4.20.4. マージオプション 101
4.20.5. マージ結果のレビュー 102
4.20.6. マージ追跡 103
4.20.7. マージ中に発生した競合の扱い 103
4.20.8. 完全なブランチをマージ 104
4.20.9. 機能ブランチの保守 105
4.21. ロック 105
4.21.1. Subversion でロックがどのように働くか 106
4.21.2. ロックの取得 106
4.21.3. ロックの解除 107
4.21.4. ロック状態のチェック 108
4.21.5. ロックしていないファイルを読み込み専用にするには 108
4.21.6. ロックのフックスクリプト 108
4.22. パッチの作成及び適用 109
4.22.1. パッチファイルの作成 109
4.22.2. パッチファイルの適用 110
4.23. 誰がその行を変更したか? 110

4.23.1. ファイルの注釈履歴 111
4.23.2. 注釈履歴の差分 113
4.24. リポジトリブラウザー 113
4.25. リビジョングラフ 116
4.25.1. リビジョングラフのノード 117
4.25.2. 表示の変更 117
4.25.3. グラフの使用 119
4.25.4. 表示の更新 120
4.25.5. ツリーの剪定 120
4.26. Subversion 作業コピーをエクスポート 121
4.26.1. 作業コピーをバージョン管理外へ 122
4.27. 作業コピーの再配置 122
4.28. バグ追跡ツール／課題追跡システムとの統合 123
4.28.1. ログメッセージへの課題IDの付与 124
4.28.2. 課題追跡システムからの情報取得 127
4.29. Web ベースのリポジトリビューアーとの統合 128
4.30. TortoiseSVN の設定 129
4.30.1. 一般設定 129
4.30.2. リビジョングラフの設定 137
4.30.3. アイコンオーバレイ設定 139
4.30.4. ネットワーク設定 143
4.30.5. 外部プログラムの設定 144
4.30.6. 保存データの設定 148
4.30.7. ログキャッシュ 149
4.30.8. クライアント側フックスクリプト 152
4.30.9. TortoiseBlame の設定 156
4.30.10. 高度な設定 157
4.30.11. TortoiseSVN の設定のエクスポート 161
4.31. 最終ステップ 161
5. SubWCRev プログラム 162
5.1. SubWCRev コマンドライン 162
5.2. キーワード置換 162
5.3. キーワード例 163
5.4. COM インターフェイス 164
6. IBugtraqProvider インターフェイス 167
6.1. 命名規則 167
6.2. IBugtraqProvider インターフェイス 167
6.3. IBugtraqProvider2 インターフェイス 169
A. よくある質問（FAQ） 172
B. こんなときは…… 173
B.1. 大量のファイルの同時移動・コピー 173
B.2. ログメッセージの入力の強制 173
B.2.1. サーバー上のフックスクリプト 173
B.2.2. プロジェクトプロパティ 173
B.3. リポジトリからの選択したファイルの更新 174
B.4. リポジトリのリビジョンのロールバック（取り消し） 174
B.4.1. リビジョンログダイアログの使用 174
B.4.2. マージダイアログの使用 174

B.4.3. svndumpfilter の使用 175
B.5. ファイルやフォルダーに対して 2 リビジョン間の比較 175
B.6. 共通のサブプロジェクトを含める 175
B.6.1. svn:externals の使用 175
B.6.2. ネストした作業コピーの使用 176
B.6.3. 相対位置の使用 176
B.7. リポジトリへのショートカットの作成 176
B.8. バージョン管理外のファイルの無視 177
B.9. 作業コピーをバージョン管理外に 177
B.10. 作業コピーの削除 177
C. 管理者向けの便利な小技 178
C.1. グループポリシーでの TortoiseSVN のデプロイ 178
C.2. 更新チェックのリダイレクト 178
C.3. SVN_ASP_DOT_NET_HACK 環境変数の設定 179
C.4. コンテキストメニューエントリの無効化 179
D. TortoiseSVN の自動化 181
D.1. TortoiseSVNのコマンド 181
D.2. Tsvncmd URL ハンドラ 182
D.3. TortoiseIDiff コマンド 183
E. コマンドラインインターフェイスのクロスリファレンス 184
E.1. 規約と基本規則 184
E.2. TortoiseSVNのコマンド 184
E.2.1. チェックアウト 184
E.2.2. 更新 184
E.2.3. リビジョンの更新 185
E.2.4. コミット 185
E.2.5. 差分 185
E.2.6. ログの表示 186
E.2.7. 変更をチェック 186
E.2.8. リビジョングラフ 186
E.2.9. リポジトリブラウザー 186
E.2.10. 競合の編集 187
E.2.11. 解決済み 187
E.2.12. 名前変更 187
E.2.13. 削除 187
E.2.14. 変更の取り消し 187
E.2.15. クリーンアップ 187
E.2.16. ロックの取得 187
E.2.17. ロックの解除 187
E.2.18. ブランチ・タグ 188
E.2.19. 切り替え 188
E.2.20. マージ 188
E.2.21. エクスポート 188
E.2.22. 再配置 189
E.2.23. ここにリポジトリを作成 189
E.2.24. 追加 189
E.2.25. インポート 189
E.2.26. 注釈履歴 189

E.2.27. 無視リストに追加 189
E.2.28. パッチを作成 189
E.2.29. パッチの適用 189
F. 実装の詳細 190
F.1. アイコンオーバーレイ 190
G. 言語パックとスペルチェッカー 192
G.1. 言語パック 192
G.2. スペルチェッカー 192
用語集 193
索引 196

図の一覧

1.1. バージョン管理外フォルダーの TortoiseSVN メニュー 2
1.2. インポートダイアログ 3
1.3. ファイルの差分を見る 4
1.4. ログダイアログ 5
2.1. 典型的なクライアント/サーバーシステム 7
2.2. 回避したい問題 8
2.3. ロック・変更・アンロックモデル 9
2.4. コピー・変更・マージモデル 10
2.5. ...コピー・変更・マージモデル（の続き） 11
2.6. リポジトリのファイルシステム 13
2.7. リポジトリ 15
3.1. バージョン管理外フォルダーの TortoiseSVN メニュー 18
4.1. エクスプローラーのアイコンオーバーレイ表示 25
4.2. バージョン管理下のフォルダーのコンテキストメニュー 26
4.3. バージョン管理されたフォルダー内のショートカットに対するエクスプローラーのファイルメニュー 27
4.4. バージョン管理下のディレクトリに対する右ドラッグメニュー 27
4.5. 認証ダイアログ 28
4.6. インポートダイアログ 30
4.7. チェックアウトダイアログ 32
4.8. コミットダイアログ 35
4.9. コミットダイアログのスペルチェッカー 38
4.10. コミットの状況を表示している進行ダイアログ 39
4.11. 更新が完了したときの進行ダイアログ 40
4.12. エクスプローラーのアイコンオーバーレイ表示 46
4.13. エクスプローラーのプロパティページの Subversion タブ 48
4.14. 変更をチェック 49
4.15. 変更リストがあるコミットダイアログ 52
4.16. リビジョンログダイアログ 53
4.17. リビジョンログダイアログの上部のリストのコンテキストメニュー 54
4.18. 2つのリビジョンを選択した時の上部のリストのコンテキストメニュー 57
4.19. ログダイアログの下のリストのコンテキストメニュー 58
4.20. マージ追跡リビジョンを表示したログダイアログ 60
4.21.「作者別コミット数」ヒストグラム 63
4.22.「作者別コミット数」円グラフ 64
4.23.「時期別コミット数」グラフ 65
4.24. オフライン移行ダイアログ 66
4.25. リビジョンの比較ダイアログ 68
4.26. 画像差分ビューアー 69
4.27. バージョン管理外のファイルでのエクスプローラーコンテキストメニュー 71
4.28. バージョン管理下のディレクトリに対する右ドラッグメニュー 72
4.29. バージョン管理外のファイルでのエクスプローラーコンテキストメニュー 73
4.30. バージョン管理下のファイルに対するエクスプローラーのコンテキストメニュー 75
4.31. 変更の取り消しダイアログ 78
4.32. Subversion のプロパティページ 80
4.33. プロパティの追加 81
4.34. svn:externals プロパティページ 86

4.35. svn:keywords プロパティページ 86
4.36. svn:eol-style プロパティページ 87
4.37. tsvn:bugtraq プロパティページ 88
4.38. ログメッセージサイズのプロパティページ 89
4.39. 言語プロパティページ 89
4.40. svn:mime-type プロパティページ 90
4.41. svn:needs-lock プロパティページ 90
4.42. svn:executable プロパティページ 90
4.43. ブランチ/タグの作成ダイアログ 94
4.44. 切り替えダイアログ 96
4.45. マージウィザード - リビジョン範囲の選択 98
4.46. マージウイザード - マージの再統合 100
4.47. マージウィザード - ツリーのマージ 101
4.48. 競合の解決ダイアログ 104
4.49. マージ再統合ダイアログ 105
4.50. ロックダイアログ 107
4.51. 変更をチェックダイアログ 108
4.52. パッチ作成ダイアログ 109
4.53. 注釈履歴ダイアログ 111
4.54. TortoiseBlame 112
4.55. リポジトリブラウザー 114
4.56. リビジョングラフ 116
4.57. URL からエクスポートダイアログ 121
4.58. 再配置ダイアログ 122
4.59. 課題追跡システムのプロパティダイアログ 124
4.60. 課題追跡システムクエリダイアログの例 128
4.61. 設定ダイアログの「全般」ページ 130
4.62. 設定ダイアログ、「コンテキストメニュー」ページ 132
4.63. 設定ダイアログ、「ダイアログ1」ページ 133
4.64. 設定ダイアログ、「ダイアログ2」ページ 134
4.65. 設定ダイアログ、「色」ページ 136
4.66. 設定ダイアログ、「リビジョングラフ」ページ 137
4.67. 設定ダイアログ、リビジョングラフの「色」ページ 138
4.68. 設定ダイアログ、「アイコンオーバーレイ」ページ 139
4.69. 設定ダイアログ、「アイコンセット」ページ 142
4.70. 設定ダイアログ、「オーバーレイハンドラー」ページ 142
4.71. 設定ダイアログ、「ネットワーク」ページ 143
4.72. 設定ダイアログ、「差分ビューアー」ページ 144
4.73. 設定ダイアログ、「差分・マージの高度な設定」ダイアログ 147
4.74. 設定ダイアログ、「保存されたデータ」ページ 148
4.75. 設定ダイアログ、「ログキャッシュ」ページ 149
4.76. 設定ダイアログ、ログキャッシュ統計 151
4.77. 設定ダイアログ、「フックスクリプト」ページ 152
4.78. 設定ダイアログ、フックスクリプトの設定 153
4.79. 設定ダイアログ、「課題追跡システムとの統合」ページ 155
4.80. 設定ダイアログ、「TortoiseBlame」ページ 156
4.81. タスクバーでのデフォルトのグループ化 158
4.82. タスクバーでのリポジトリ毎のグループ化 158

4.83. タスクバーでのリポジトリ毎のグループ化 159
4.84. タスクバーでのグループ化にリポジトリのカラーオーバーレイが付いた様子 159
C.1. アップグレードの通知を表示するコミットダイアログ 178

表の一覧

2.1. リポジトリにアクセスするURL 14
5.1. 使用できるコマンドラインスイッチ一覧 162
5.2. 使用できるコマンドラインスイッチ一覧 162
5.3. COM オートメーションのサポート 164
C.1. メニューエントリとその値 179
D.1. 使用できるコマンドとオプションの一覧 182
D.2. 使用できるオプションの一覧 183

序章

バージョン管理は、情報の変更を管理する技術です。これは、たとえばソフトウェアのあちこちに変更を加えたり、その翌日にはその変更の一部を取り消したりチェックしたりすることに時間を費やしてきたようなプログラマたちにとって、長い間重要な技でした。そのような開発者たちが同時に働くチームを想像してみてください。恐らく同じファイルに同時に変更を加える場面があるでしょう。なぜ、このような潜在的な混沌を管理する良いシステムが求められているかがわかるでしょう。

1. TortoiseSVNとは

TortoiseSVN は、 *Apache™* Subversion® バージョン管理システムの無料でオープンソースのWindows版クライアントです。 TortoiseSVN はファイルやディレクトリを時間を超えて管理します。ファイルは リポジトリ に集中的に格納されます。リポジトリは普通のファイルサーバーとよく似ていますが、ファイルを古いバージョンに戻したり、いつ、だれがデータに変更を加えたのかを履歴から確認したりすることができます。このため、 Subversion などのバージョン管理システムは、一般的に「タイムマシン」のようなものと考えられています。

バージョン管理システムによっては、ソフトウェア構成管理（SCM）システムを兼ねている場合もあります。このようなシステムはソースコードのツリーを管理するために最適化して設計されており、ソフトウェア開発に特化した多くの機能を持っています。例えば、プログラム言語をネイティブに理解したり、ソフトウェアを構築するのに必要なツールが付属していたりといった具合にです。しかし、Subversionはそういったシステムではありません。あらゆるファイルの集合（ソースコードを含む）を管理する汎用的なシステムです。

2. TortoiseSVNの特徴

TortoiseSVNはSubversionクライアントとして、どのような点が優れているのでしょうか？主な特徴は次の通りです。

シェル統合

TortoiseSVNはWindowsのシェル（すなわちエクスプローラー）に統合されています。つまり、すでに使い慣れた方法で操作することができます。そして、バージョン管理の機能が必要になるたびに違うアプリケーションに切り替える必要はありません。

Windowsのエクスプローラーに限らず、TortoiseSVNのコンテキストメニューは、他のファイルマネージャや、「ファイルを開く」ダイアログのような一般のWindowsアプリケーションで共通に使用されているダイアログでも使用することができます。しかし、TortoiseSVNはWindowsエクスプローラーの拡張機能として開発されていることを認識しておいてください。つまり、他のアプリケーションの中では、アイコンオーバーレイが表示されないといったように、統合が不完全になることもあります。

アイコンオーバレイ

バージョン管理されているファイルやフォルダーの状態は、小さなオーバレイアイコンによって表されます。これで作業コピーの状態をすぐに確認することができます。

グラフィカルユーザーインターフェイス（GUI）

あるファイルやフォルダーの変更を一覧する際、リビジョンをクリックすると、それをコミットした際のコメントを読むことができます。また、変更されたファイルの一覧からファイルをダブルクリックするだけで、すぐに変更内容を確認することができます。

コミットダイアログでは、コミットに含まれる可能性のある項目の一覧が表示されます。各項目と共に表示されるチェックボックスで、コミットに含めたいファイルを選択することができます。バージョン管理されていないファイルも一覧に表示されるので、新しいファイルの追加し忘れを防ぐことができます。

Subversionコマンドへの容易なアクセス
: エクスプローラーのコンテキストメニューから、すべてのSubversionコマンドを利用できます。TortoiseSVNは独自のサブメニューにそのコマンドを追加します。

TortoiseSVNはSubversionのクライアントであるため、Subversion自体が持つ特徴も紹介します。

ディレクトリのバージョン管理
: CVSは個々のファイルの履歴しか追跡できませんが、Subversionでは、時系列でディレクトリツリー全体に行われた変更を追跡する「仮想的な」バージョン管理ファイルシステムを実装しています。ファイルおよびディレクトリがバージョン管理されます。そのため、クライアント側からファイルやディレクトリを操作するmove（移動）やcopy（コピー）コマンドが実装されています。

不可分コミット
: コミットは完全にリポジトリに格納されるか、全くされないかのどちらかになります。これにより、開発者は論理的に一貫した変更を作成し、コミットすることができます。

バージョン管理されたメタデータ
: すべてのファイルやディレクトリには、不可視の「プロパティ」が添付されています。任意のキーと値の組み合わせを作成し格納することができます。プロパティはファイルの内容と同様に、時系列でバージョン管理されます。

ネットワークレイヤーの選択
: Subversionはリポジトリへのアクセスが抽象化されており、新しいネットワークメカニズムを容易に実装できるようになっています。Subversionの「先進的」なネットワークサーバーは、Apache　ウェブサーバーのモジュールになっており、HTTPの一種であるWebDAV/DeltaVで通信します。これは、Subversionに安定性や相互運用性の面で大きな利点を与え、認証、認可、データ圧縮、リポジトリ閲覧のような、様々な重要な機能を使用することができます。もっと小さいスタンドアロンのSubversionサーバーもあります。このサーバーはsshで簡単にトンネル通信ができる独自プロトコルで通信します。

一貫したデータの取り扱い
: Subversionはテキストファイル（人が読める形式）でもバイナリファイル（人が読めない形式）でも、ファイルの差分をバイナリ差分アルゴリズムを用いて表現します。どちらのタイプのファイルでも、リポジトリに圧縮して格納されます。また、差分はネットワークを使って双方向に通信されます。

効率的なブランチやタグの作成
: ブランチやタグを作成するときのコストは、プロジェクトの規模に比例しません。Subversionはブランチやタグを作成するのに、ハードリンクによく似た単純なプロジェクトのコピーで行います。そのため、この操作はとても小さいコストであり、一定の時間で完了し、リポジトリのほんの小さな領域しか使用しません。

3. ライセンス

TortoiseSVNは、GNU General Public License (GPL) に基づいて開発されているオープンソース・プロジェクトです。個人用・商用にかかわらず、何台のPCにも、無料でダウンロードでき無料で使用できます。

多くの人々はインストーラーをダウンロードしますが、このプログラムのソースコードもすべて読めるようになっています。ブラウザーで http://code.google.com/p/tortoisesvn/source/browse/ から参照できます。現在開発中のものは `/trunk/` 以下に、既にリリースされたバージョンは `/tags/` 以下にあります。

4. 開発

TortoiseSVN や Subversion は、開発者のコミュニティによって開発されています。世界中の様々な国の出身者が、偉大なソフトウェアを開発するために協力しています。

4.1. TortoiseSVN の歴史

2002年、Tim Kemp は Subversion が大変すぐれたバージョン管理システムなのに、使いやすい GUI クライアントがないことに気がつきました。Windows のシェルに統合した Subversion クライアントという考え方は、TortoiseCVS という良く似た CVS クライアントにヒントを得ました。彼は TortoiseCVS のソースコードを解析し、TortoiseSVN の元にしました。彼はそれからプロジェクトを開始し、`tortoisesvn.org` のドメインを登録し、ソースコードをオンラインに公開しました。

同じ頃、Stefan Küng はフリーの良いバージョン管理システムを探していて、Subversion と TortoiseSVN に出会いました。 TortoiseSVN はまだ開発中でしたので、彼もこのプロジェクトに参加しました。彼はすぐに、既存のコードの大部分を書き直して、コマンドや機能を追加し始め、オリジナルのコードが全く残らないほどになりました。

Subversion が安定するにつれて、Subversion クライアントとして TortoiseSVN を使用するユーザーがどんどん増えてきました。ユーザー層は急速に拡大しました（そして日々拡大しています）。Lübbe Onken は TortoiseSVN のロゴとアイコンをデザインしました。また彼は、ウェブサイトと翻訳の管理をしています。

4.2. 謝辞

Tim Kemp
: TortoiseSVN プロジェクトの創始に

Stefan Küng
: TortoiseSVN が現在の姿になるまでに彼が費やした努力とリーダーシップに

Lübbe Onken
: 美しいアイコン、ロゴ、バグつぶし、翻訳と翻訳管理に

Simon Large
: マニュアルを管理してくれたことに

Stefan Fuhrmann
: ログキャッシュとリビジョングラフの開発に

The Subversion Book
: Subversion のすばらしい入門書に（第2章はここからのコピー）

The Tigris Style project
: 本書で再利用されているスタイルに

われらが貢献者達
: パッチや、バグレポート、新しいアイデア、メーリングリストでの質疑応答などに

われらが寄進者達
: 送ってくれた音楽で楽しんだ時間に

5. このガイドの読み方

本書は、データ管理に Subversion を使用しようとしているコンピューターに詳しい人で、コマンドラインのクライアントよりも GUI のクライアントを使用したい人向けに書かれています。TortoiseSVN は Windows のシェル拡張ですので、ユーザーは Windows エクスプローラーに慣れており、使い方を知っていることを前提とします。

序章 では TortoiseSVN とは何かを説明します。また、TortoiseSVN プロジェクトやそこで作業している人々のコミュニティ、使用や頒布にあたってのライセンス条件を簡単に説明しています。

1章さぁはじめましょう では、TortoiseSVN をPCにインストールする方法、さっそく使い始める方法を説明しています。

2章バージョン管理の基本概念 では、TortoiseSVN の基盤となる Subversion バージョン管理システムを簡単に解説しています。これは Subversion プロジェクトのドキュメントを借用したもので、バージョン管理の異なる方法や Subversion がどのように動作するのかを説明しています。

3章リポジトリ では、単独のPCで Subversion や TortoiseSVN を試すのに便利なように、ローカルリポジトリのセットアップ方法を説明しています。また、リポジトリ管理についても若干の説明をしています。サーバー上のリポジトリを使用する場合も関係します。また、必要に応じてサーバーをセットアップする方法もこの章で説明しています。

4章日常の使用ガイド はもっとも重要な章で、TortoiseSVN のすべての主要な機能とその使い方を説明しています。チュートリアルの形を採って、作業コピーのチェックアウトから始め、変更、コミット、等を行います。そこからさらに高度な話題に進みます。

5章SubWCRev プログラム は TortoiseSVN に含まれる独立したプログラムです。作業コピーからの情報を展開し、ファイルに書き出すことができます。プロジェクトにビルド情報を含めるために使用することができます。

付録B こんなときは…… では、他の部分で明示されていない作業を行う上で、よく発生する質問に対する答えを示しています。

付録D TortoiseSVN の自動化 では、TortoiseSVN の GUI ダイアログをコマンドラインから呼び出す方法を示します。ユーザーの操作が必要なスクリプトを作成するのに便利です。

付録E コマンドラインインターフェイスのクロスリファレンス では、TortoiseSVN のコマンドと Subversion のコマンドラインクライアント `svn.exe` の同等の機能との関連を示しています。

6. 本書で使用している表現

容易にに読み進められるよう、TortoiseSVN の画面やメニューの名前はすべて、異なるフォントになるようマークアップを施しました。例えば **ログダイアログ** といった具合です。

メニューの選択は矢印で表しました。**TortoiseSVN → ログを表示** は、コンテキストメニューの TortoiseSVN から ログを表示 を選択するということです。

TortoiseSVN ダイアログの中で現れるローカルコンテキストメニューは、**コンテキストメニュー → 名前を付けて保存 ...** のように表します。

ユーザーインターフェイスのボタンは、「続けるには OK を押してください」というように表します。

ユーザーアクションは太字で示します。Alt+A は、キーボードの Alt キーを押したままで A キーを押します。右ドラッグは、マウスの右ボタンを押したままで項目を新しい位置に ドラッグ します。

システム出力とキーボード入力は、このように **異なる** フォントで表します。

重要

重要な注釈はこのアイコンでマークをつけました。

ヒント

人生を楽にする豆知識です。

注意

自分が何をしているか注意すべきところです。

警告

最大限の注意を払う必要があるところです。この警告を無視すると、データが破損したり、その他のトラブルが起きたりする可能性があります。

第1章 さぁはじめましょう

この章は、TortoiseSVN を初めて試す方を対象にしたものです。TortoiseSVN のインストールの仕方や、ローカルのリポジトリをセットアップする方法、よく使われる一通りの操作を説明します。

1.1. TortoiseSVN のインストール

1.1.1. 必要なシステム

TortoiseSVN は Windows XP Service Pack 3 以上で動作し、32ビット版と64ビット版のどちらでも使用できます。64ビット版 Windows 用のインストーラーは32ビットの拡張部分も含みます。つまり、32ビットアプリケーションで TortoiseSVN のコンテキストメニューやオーバーレイを使用するために、別途32ビット版をインストールする必要はありません。

重要

Windows XP を使用している場合は、必ず Service Pack 3 以上をインストールしてください。サービスパックをインストールしないと、動作しないかもしれません。

Windows 98、Windows ME、Windows NT4 に対するサポートはバージョン1.2.0以前で終了し、Windows 2000 と XP の SP2 までは1.7.0以前で終了しました。もしそれらが必要であれば、まだ旧バージョンをダウンロードしてインストールすることができます。

1.1.2. インストール

TortoiseSVN はインストーラーを使用すると簡単にインストールできます。インストーラーのファイルをダブルクリックし、画面の指示に従ってください。後はインストーラーが面倒を見てくれます。最後にPCを再起動するのをお忘れなく。

重要

TortoiseSVN をインストールするには、Administrator 権限が必要です。

TortoiseSVN のユーザーインターフェイスを翻訳する言語パックが、たくさんの言語で提供されています。言語パックのインストール方法は、付録G 言語パックとスペルチェッカー を参照してください。

インストールした後に何か問題が発生した場合は、http://tortoisesvn.net/faq.html からオンラインFAQを参照して下さい。

1.2. バージョン管理の基本概念

実際のファイルで事故を起こさないように、 Subversion の主要な動きと使われている用語を知っておくことは重要です。

リポジトリ

Subversion は、すべてのバージョン管理対象のファイルとその履歴を格納するデータベースを使用します。このデータベースを リポジトリ と呼びます。一般に、リポジトリは Subversion のサーバープログラムが動作しているファイルサーバーに配置し、必要に応じて（TortoiseSVN のような）Subversion クライアントに内容を提供します。バックアップを1つしか取らないならば、すべてのデータの重要なマスターコピーであるリポジトリのバックアップを取ってください。

作業コピー

実際にどのような作業をするのかを紹介します。各開発者は自分のローカルのPCに、自分用の作業コピーを持ちます（サンドボックスと呼ばれることもあります）。リポジトリから最新バージョンを取り出し、他の誰にも影響せずに作業し、作業が完了したら変更をコミットしてリポジトリに書き戻します。

Subversion の作業コピーにはプロジェクトの履歴は含まれていませんが、リポジトリに存在している変更前のファイルのコピーとして維持されています。これにより、どのような変更を加えたのかを容易に確認することができます。

TortoiseSVN をスタートメニューから使用することはあまりありません。 TortoiseSVN はシェル拡張ですので、まず Windows エクスプローラーを起動します。エクスプローラーでフォルダを右クリックすると、このようなコンテキストメニューにいくつかの新しい項目が表示されます。

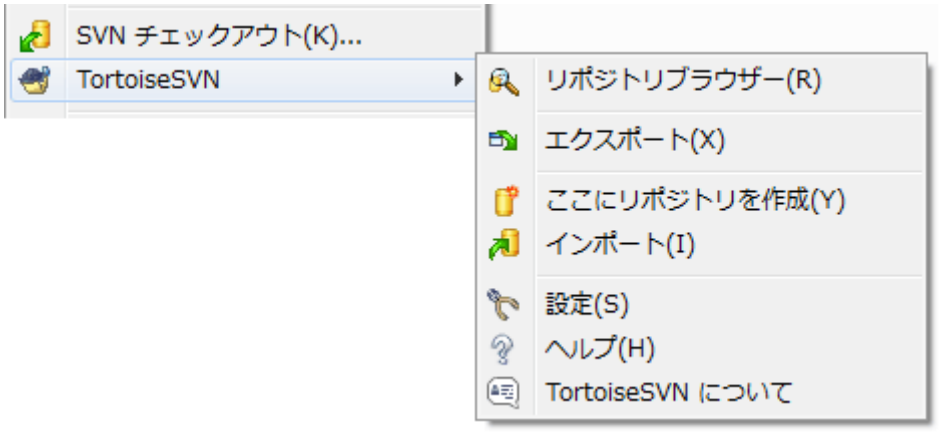


図1.1 バージョン管理外フォルダーの TortoiseSVN メニュー

1.3. 試してみましょう

ここでは、小さなテスト用のリポジトリを使用して、よく使用される機能をいくつか試してみます。当然、これは単なるクイックスタートガイドなので、全機能を説明するわけではありません。いったん試したみた後は、このユーザーガイドの残り部分を時間をかけて読んでいただければ、もっと詳しいことが分かります。また、正しい Subversion サーバーの設定に関する知識も、そちらで説明しています。

1.3.1. リポジトリの作成

現実のプロジェクトでは、どこか安全なサーバーにリポジトリを構築し、それを制御するために Subversion サーバーをインストールします。今回は練習なのでサーバーを使用せず、Subversion のローカルリポジトリ機能を使用し、リポジトリをローカルのハードディスクの中に構築して直接アクセスするようにします。

最初にPC上で新しい空のディレクトリを作成します。どこに作成しても良いのですが、練習として `C:¥svn_repos` に作成したとします。そして、すぐに作成したフォルダーを右クリックし、コンテキストメニューから **TortoiseSVN → ここにリポジトリを作成...** を実行します。こうすれば、リポジトリがフォルダー内に作成され、使用可能な状態になります。また、**フォルダー構造を作成** ボタンをクリックすれば、デフォルトの内部フォルダー構造を作成することができます。

重要

ローカルリポジトリ機能は、テストや評価にはとても便利です。しかし、1人の開発者が1台のPCのみを使って作業をして場合を除いて、 Subversion 専用のサーバーを設置すべきです。小さい会社では、サーバーのセットアップ作業を嫌って、ネットワークフォルダー上のリポジトリをアクセスするようにしたがるかもしれません。しかし、それはやめてください。データを失うことになります。なぜこの方

> 法が良くないのか、またサーバーをセットアップする方法については、「ネットワークフォルダー上のリポジトリへのアクセス」をお読みください。

1.3.2. プロジェクトのインポート

これでリポジトリができましたが、その中身はまだ空です。 C:¥Projects¥Widget1 に追加したいファイルのセットがあるとします。エクスプローラーで Widget1 フォルダーに移動し、右クリックします。そして、TortoiseSVN → インポート... を実行すると、次のインポートダイアログが表示されます。

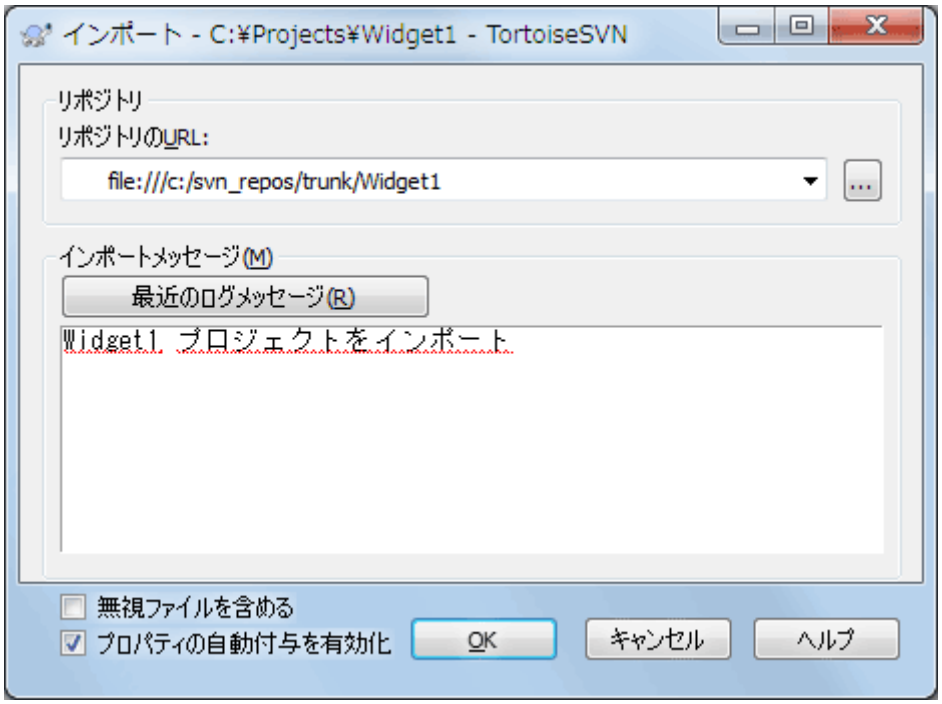


図1.2 インポートダイアログ

Subversion のリポジトリは、インターネット上のどこのリポジトリでも指定できるようにURLで参照します。この場合は、私たちが持っているローカルリポジトリのURL、 file:///c:/svn_repos/trunk を指すようにする必要があります。そして、プロジェクト名 Widget1 を追加します。 file: の後に3つのスラッシュあることや、スラッシュの向きにも注意してください。

このダイアログのもう1つの重要な機能が、 **インポートメッセージ** 欄です。この中には、いつ、どのような理由で、何をしたのか、メモを残すことが出来ます。これは後に参考になるものです。今回は簡単に 「Widget1プロジェクトをインポート」とでも入力してください。 OK ボタンをクリックすると、フォルダーがリポジトリに登録されます。

1.3.3. 作業コピーのチェックアウト

これでプロジェクトがリポジトリに登録されましたので、次に日常の作業に使用する作業コピーを作成します。フォルダーをインポートしたとき、フォルダーが自動的に作業コピーになる訳ではないので注意してください。 Subversion では、新しい作業コピーを作ることを **チェックアウト** といいます。リポジトリの Widget1 フォルダーを、PC上の開発用フォルダー C:¥Projects¥Widget1-Dev にチェックアウトしてみましょう。フォルダーを作成し、次にそれを右クリックして、TortoiseSVN → チェックアウト... を選択します。チェックアウト元のURLを、 file:///c:/svn_repos/trunk/Widget1 のような形式で入力し、 OK をクリックします。これで開発用フォルダーに、リポジトリからファイルが読み込まれます。

この作業コピーのファイルをエクスプローラーで見ると、右下に緑色のチェックが入っていることに気が付くと思います。これは、この作業コピーがリポジトリの中にある同一ファイルと同じ内容であることを示しています。

1.3.4. ファイルの変更

さて、Widget1-Dev の中にある、Widget1.c と ReadMe.txt の2つのファイルについて、内容を変更してみましょう。これらのファイルは、アイコンのオーバーレイが赤に変わるはずです。これは、作業コピーがローカルで変更されたことを示しています。

何が変更されたのでしょうか?変更されたファイルの1つを右クリックし、TortoiseSVN → 差分 を選択すると、ファイル比較ツールが現われ、どの行が変更されたのかを見ることができます。

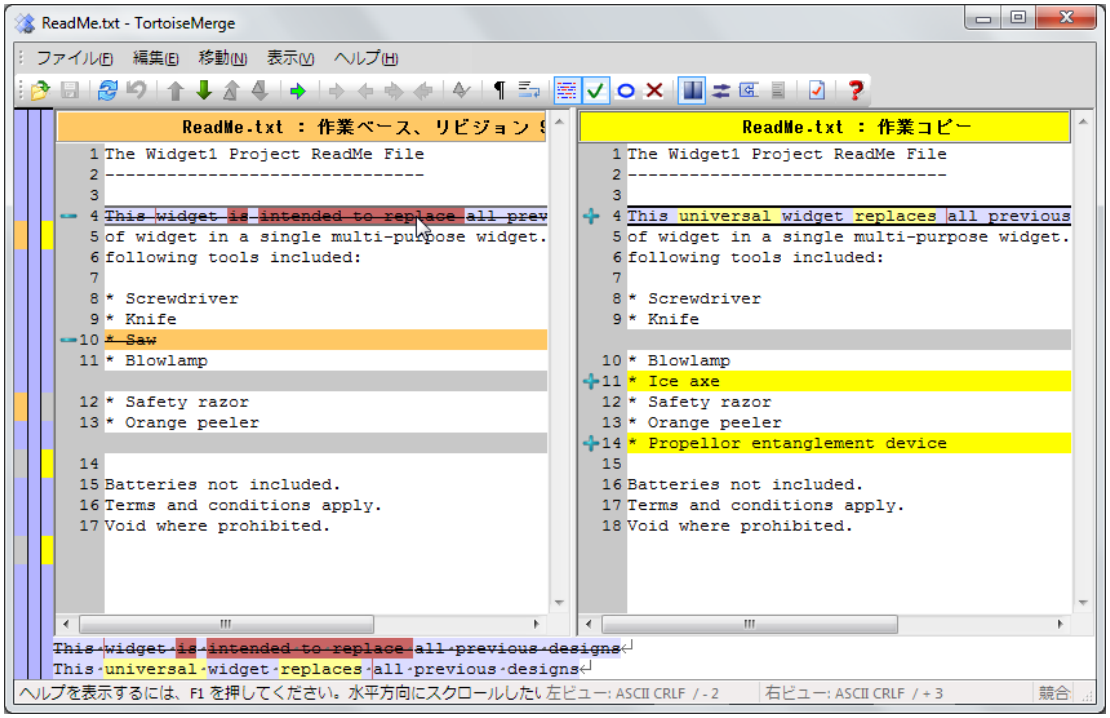


図1.3 ファイルの差分を見る

さて、変更の内容に問題がないようでしたら、リポジトリを更新しましょう。これを**コミット**と呼びます。Widget1-Dev フォルダーを右クリックし、TortoiseSVN → コミット を選択してください。フォルダー内のファイルのうち、変更があったものがチェックボックス付きで表示されます。変更したファイルの一部だけをコミットすることも出来ますが、この場合は両方のファイルをコミットしましょう。メッセージの項に、何をどのような理由で変更したか書き込み、OK をクリックします。ファイルがリポジトリにアップロードされていることを表示する進捗ダイアログが表示され、コミットが完了します。

1.3.5. ファイルの追加

プロジェクトを使った作業が進むにつれ、新規のファイルを追加する必要が出てくるかもしれません。Extras.c というファイルにある新しい機能を追加し、参照を既存の Makefile に追加するとしましょう。このフォルダーを右クリックして、TortoiseSVN → 追加 を実行します。追加ダイアログが表示され、バージョン管理されていないファイルがすべて表示されるので、追加したいファイルを選択します。他にファイルを追加するには、追加したいファイルを右クリックし、TortoiseSVN → 追加 を実行するという方法もあります。

次にフォルダーをコミットするとき、新しいファイルは 追加 と表示され、既存のファイルは 変更 と表示されます。変更されたファイルをダブルクリックすると、どのような変更が行われたのかを確認することができます。

1.3.6. プロジェクトの変更履歴を見る

TortoiseSVN で最もよく使われる機能のひとつが、ログダイアログです。ここで、コミットされたファイルやフォルダの一覧と、コミット時に入力された詳細なメッセージを見ることができます。

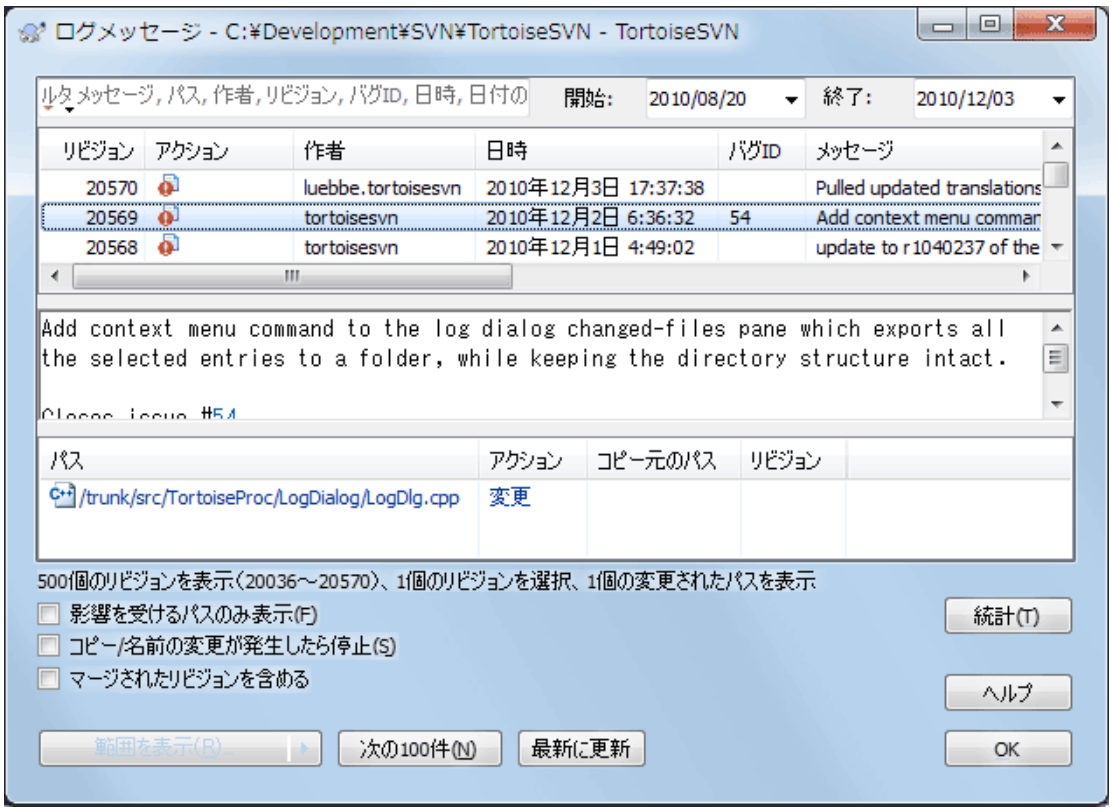


図1.4 ログダイアログ

上の欄には、コミットされたリビジョンの一覧が、コミット時のメッセージの冒頭と一緒に表示されます。リビジョンを1つ選択すると、中央の欄にはそのリビジョンのログメッセージの全文、下の欄には変更されたファイルやフォルダの一覧が表示されます。

それぞれの欄でコンテキストメニューを使用すると、その情報に関する様々な操作ができます。下の欄のファイルをダブルクリックすると、そのリビジョンでの変更の詳細を見ることができます。詳しくは、「リビジョンログダイアログ」を参照してください。

1.3.7. 変更を取り消す

どのバージョン管理システムでも、以前に行った変更を取り消す機能を持っています。元に戻したくなった時、TortoiseSVN では簡単に実行することができます。

まだコミットしていない変更を取り消して、ファイルを編集前の状態に戻したくなった場合は、**TortoiseSVN → 変更の取り消し...** を選択します。これで変更が取り消され(ごみ箱の中に入ります)、作業をする前にコミットされたバージョンに戻すことができます。一部の変更だけを取り消したい場合は TortoiseMerge を使用すれば、行単位に変更を確認したり、変更を取り消したりすることができます。

特定のリビジョンで行った変更を取り消したい場合は、ログダイアログを使用して変更を取り消したいリビジョンを探します。そして、**コンテキストメニュー → このリビジョンにおける変更を取り消す** を選択すると、そのリビジョンで変更された部分が変更前の状態に戻ります。

1.4. さあ使ってみよう

この章では、 TortoiseSVN の中でもっとも重要でよく使われる機能を手早く紹介しましたが、まだまだ紹介しきれない機能がたくさんあります。このマニュアルの残りの部分を、時間をかけて読むことを強くお勧めします。特に、 4章日常の使用ガイドには日常の操作についてより詳細なことが書かれています。

有益で読みやすい文章にするために大変苦労しましたが、まだまだ充分ではないことを認識しています。時間をかけて、テストリポジトリで恐れずに試してみてください。使いこむことが最良の学習です。

第2章 バージョン管理の基本概念

この章は Subversion book にある同章をわずかに変更したものです。Subversion book のオンライン版は、http://svnbook.red-bean.com/ から読むことができます。

この章では短く簡単に Subversion の入門の説明します。バージョン管理システムが初めてならば、この章を読めばわかるでしょう。一般的なバージョン管理の概念から始めて、Subversion の根底にある考え方を説明し、Subversion の使い方の簡単な例を紹介します。

この章ではプログラムソースコードの共有を例として示していますが、Subversion はどんなファイルでも扱えるということに留意してください。プログラマーを助けるためだけのものではありません。

2.1. リポジトリ

Subversion は情報共有の一元管理システムです。この核となるのがデータを格納する リポジトリ です。リポジトリは情報を ファイルシステムツリー （簡単にいえばファイルやディレクトリの階層）の形で格納します。複数の クライアント がリポジトリに接続し、そのファイルを読み書きします。データを書き込むと他の人が情報を使用できるようになりますし、データを読み込むと他の人の情報をクライアントが受信します。

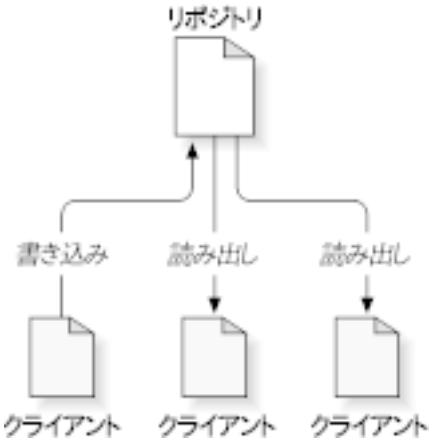


図2.1 典型的なクライアント/サーバーシステム

このどこが画期的なのでしょうか。普通のファイルサーバーと同じようにもみえます。確かにリポジトリ も ファイルサーバーの一種ですが、ふだん見かけるものとは違います。Subversion のリポジトリが特別なのは、そこに書き込まれた すべての変更を記憶している ことです。すべてのファイルに対するすべての変更、さらにディレクトリツリー自体に対して行われた変更、たとえばファイルやディレクトリの追加、削除、再配置、などを記憶しています。

クライアントがリポジトリからデータを読み出すときには、普通はファイルシステムツリーの最新のバージョンだけが見えます。しかし、ファイルシステムの 以前の 状態も見ることができます。たとえばクライアントは、「先週の水曜日にこのディレクトリには何が入っていたのか？」、とか「最後にこのファイルを変更したのは誰で、その人は何を変更したのか？」 といった履歴に関する調査をすることができます。これらはあらゆる バージョン管理システム の核心になる関わる問いです。つまりバージョン管理システムは時間と共に、データを記録し変更内容を追跡するように設計されているのです。

2.2. バージョン管理モデル

どのバージョン管理システムでも、根本的な問題を解決しなければなりません。それは、どのようにユーザーに情報の共有をさせつつ、偶然にも他人の邪魔をしないようにするかです。リポジトリ内の他人の変更を、誤って上書きしてしまうことは容易に起こりうることです。

2.2.1. ファイル共有の問題

次のような場面を想定してみてください。2人の同僚、ハリーとサリーがいたとします。2人は同時に同じリポジトリ内のファイルを編集しようとしています。始めにハリーが変更を保存してから、(数分後に)サリーが自分の新しいファイルを偶然にも上書きすることができます。(システムが変更を記憶しているので)ハリーのバージョンが永遠に失われるというわけではありませんが、ハリーが行った変更はサリーの新バージョンに隠されて 見えなくなります 。サリーがハリーの変更を取り込まなかったからです。おそらく事故でですが、ハリーの作業は事実上失われたり、少なくとも後のバージョンのファイルには現れなくなったりします。このような状況を確実に回避したいのです。

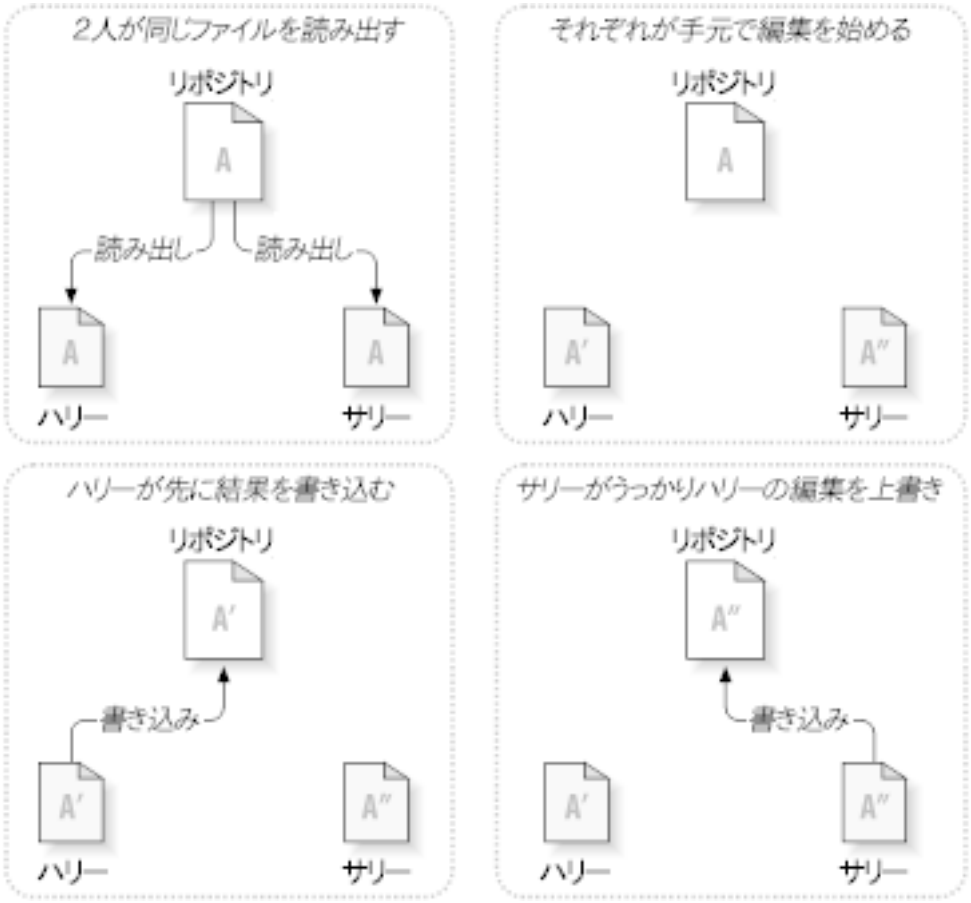


図2.2 回避したい問題

2.2.2. ロック・変更・アンロックモデル

多くのバージョン管理システムでは、 ロック・変更・アンロック モデルというとても単純な方法で、この問題に対処しています。このようなシステムでは、リポジトリはひとつのファイルにつき同時に一人しか変更できないようにしています。この場合、はじめにハリーは変更を加える前に ロック をしなければなりません。ファイルをロックすることは、図書館から本を貸りるようなものです。ハリーがファイルにロックをかけると、サリーはそこに変更を加えられなくなります。サリーがファイルをロックしようとしても、リポジトリはその要求を拒否します。できるのは、ファイルを読むことと、ハリーが変更を終えてロックを解除するのを待つことだけです。ハリーが変更を終えてロックを解除すると、サリーはロックして編集できるようになります。

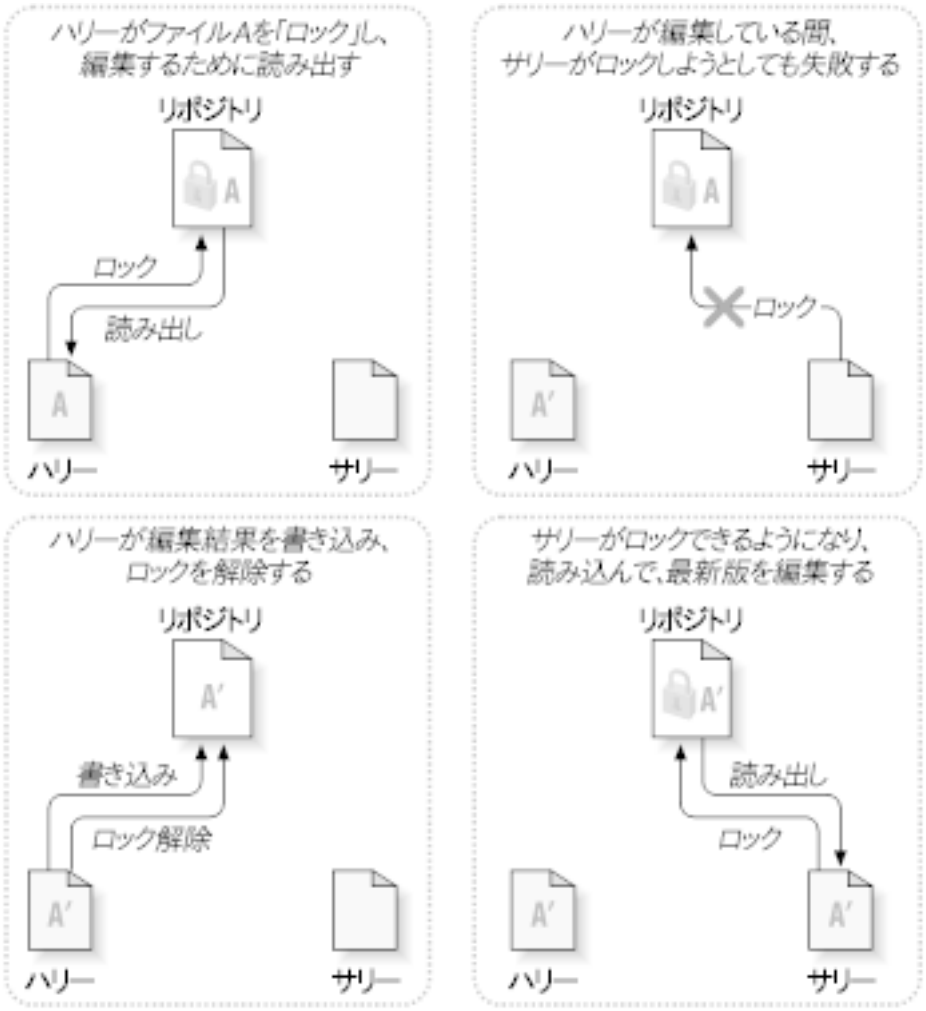


図2.3 ロック・変更・アンロックモデル

ロック・変更・アンロックモデルの問題点は、少々制限が厳しいことで、しばしばユーザーの作業の障害になります。

- ロックは管理上の問題を起こす可能性があります。ときどきハリーは、ファイルをロックしたまま忘れてしまうことがあります。いっぽうサリーは、ファイルが編集できるようになるのをずっと待っていて、その間何もできません。ハリーが休暇を取ってしまったりすると、サリーは管理者にハリーのロックを解除してもらわなければなりません。この状況では不要な遅れと、時間の浪費が発生します。

- ロックは意味のない直列化を起こす可能性があります。 ハリーがテキストファイルの先頭を編集していて、サリーは単に同じファイルの最後を編集したいとしたらどうでしょう。変更が重なることはありません。簡単にファイルを同時に編集でき、互いを適切にマージできるとすれば、なんの障害も発生しないでしょう。この状況では待つ必要はないはずです。

- ロックは誤った安心感を与える可能性があります。 ハリーはファイルAをロックして編集し、サリーはファイルBをロックして編集するとします。しかし、AとBが互いに依存しあっている場合、変更すると意味的に矛盾が発生します。突然AとBが一緒には動作しなくなります。ロックするシステムはこの問題に対しては無力です。ある意味、誤った安心感を与えていると言えるでしょう。ハリーもサリーもファイルをロックしたことで安全な状態に入ったと感じ、自分の作業は保護されていると錯覚してしまうのです。

2.2.3. コピー・変更・マージモデル

Subversion や CVS などのバージョン管理システムは、ロックの代わりに コピー・変更・マージ モデルを使用します。このモデルでは、ユーザーのクライアントが個別にリポジトリを読み込み、ファイルやプロジェクトの個人的な 作業コ

ピー を作成します。そこからユーザーは並行して作業し、個人のコピーを変更します。最後に個人のコピーを新しい最終版にマージします。バージョン管理システムはマージの補助を行いますが、正しくマージする最終責任は人間が負うことになります。

例を挙げます。ハリーとサリーはそれぞれ同じプロジェクトの作業コピーを、リポジトリからコピーして作成したとします。2人とも同時に作業し、それぞれのコピーの同じファイル A に変更を加えました。最初にサリーがリポジトリに変更を保存しました。そのあとハリーが変更を適用しようとしましたが、ハリーのファイルAは 最新ではない とリポジトリに言われてしまいました。一方、リポジトリのファイルAには、ハリーが最後にコピーしたときから、何らかの変更が加わっています。そこでハリーは、リポジトリから新しい変更点を取得し、作業コピーのファイルAに マージ するように指示を出します。幸い、サリーの変更はハリーの変更に重なりません。そのため、いったん両方の変更点を統合してしまえば、作業コピーの内容をリポジトリに書き戻すことができます。

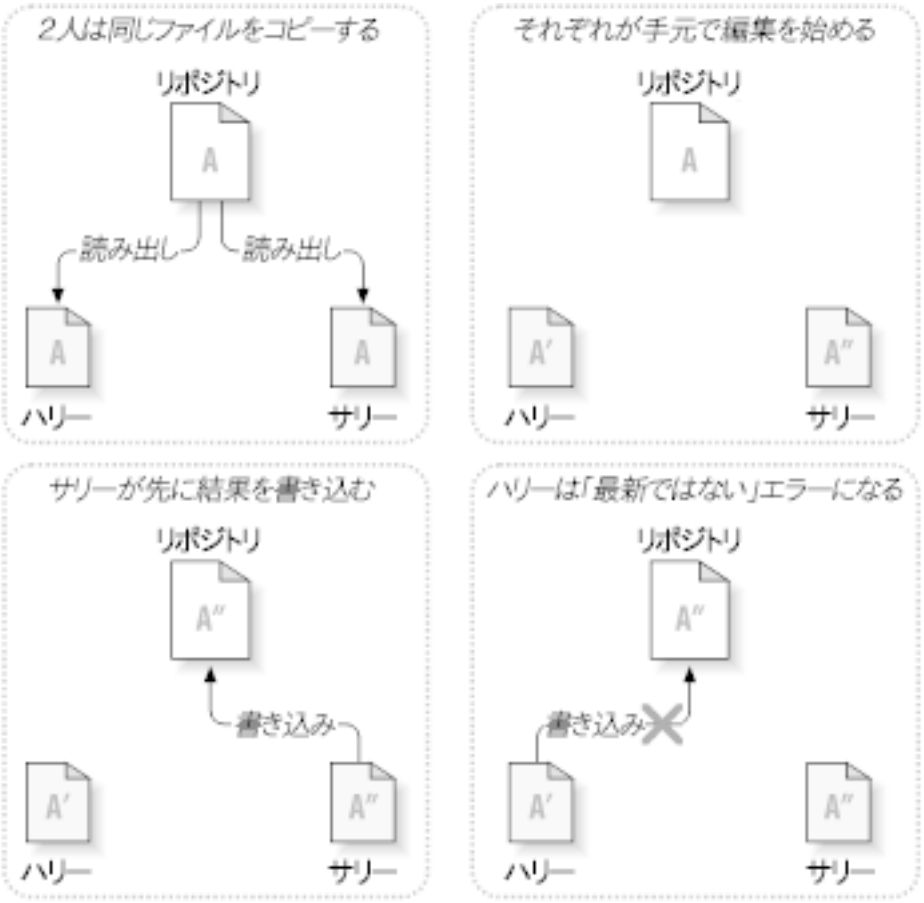


図2.4 コピー・変更・マージモデル

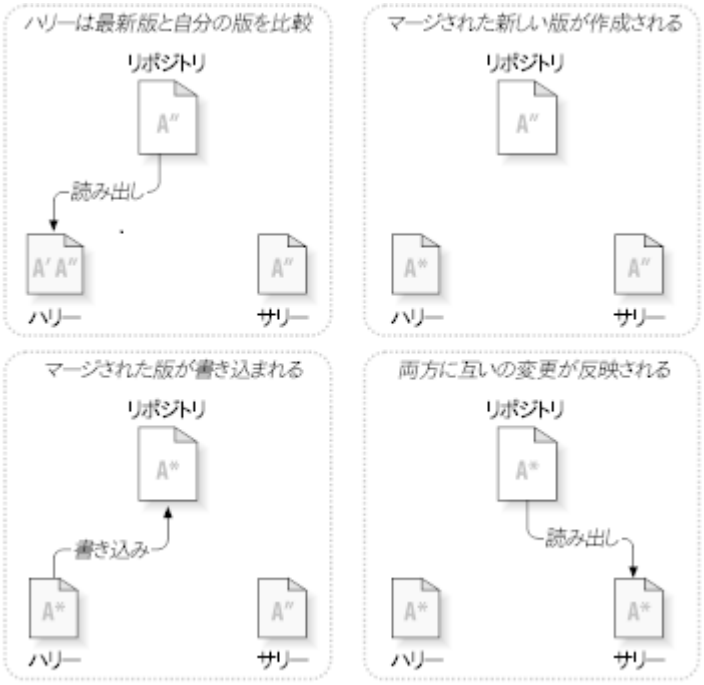


図2.5 ...コピー・変更・マージモデル（の続き）

では、サリーの変更がハリーの変更に　　重なっていた　　ら?そのときはどうなるのでしょう?この状況は　　競合 と呼ばれ、普段はあまり問題になりません。ハリーがクライアントプログラムに、リポジトリの最新の変更を自分の作業コピーにマージするよう指示を出すと、作業コピーのファイルAは競合している状態になります。このとき彼は競合した変更を両方とも見ることができ、どちらを採用するかを選択することができます。ソフトウェアが自動的に競合を解決できないことに注意してください。理解し正しく選択する力を持っているのは人間だけです。ハリーがいったん重なった変更を解決したら、（おそらくサリーと競合について話し合ったあとで）マージしたファイルを安全にリポジトリに保存できます。

コピー・変更・マージモデルは少々無秩序に見えますが、実際にはとてもスムーズに事が進みます。ユーザーは他の人を待つこともなく、並行して作業を進められます。同じファイルに対して作業を行った場合でも、ほとんどの変更は重ならないことが分かると思います。そして、競合を解決するのにかかる時間は、ロックシステムで失われる時間よりもずっと少ないのです。

このことは、最終的にひとつの重要な要因にたどり着きます。ユーザー間のコミュニケーションです。ユーザーがお互いに意見をやりとりしなければ、構文上・意味上の競合が増えていきます。どんなシステムもユーザーに完璧なコミュニケーションを強要できませんし、意味上の競合も検出できません。したがって、ロックシステムなら競合は防げるなどという間違った思い込みで安心するなど、全く意味がありません。実際には、ロックは生産性を下げる以外の何物でもないように思えます。

ロック・変更・アンロックモデルの方がよいといわれる事例もあります。マージできないファイルがある場合です。例えば、リポジトリに画像イメージが含まれている場合、2人が同時にイメージを変更してもこれをマージする方法はありません。ハリーとサリーのどちらも自分の行った変更を失ってしまいます。

2.2.4. Subversionではどうしているのか

通常、 Subversion はコピー・変更・マージモデルを使用しますし、ほとんどの場合この方法が求められていると思います。しかし、バージョン1.2で、 Subversion はファイルのロックもサポートしました。したがって、マージできないファイルがあったり、管理するのに単にロックポリシーを強制したりしたい場合にも、 Subversion は必要に応じてそういった機能を提供することができます。

2.3. Subversionの動作

2.3.1. 作業コピー

作業コピーについてはすでに説明したとおりですので、今度は Subversion クライアントでどのように作業コピーを作成したり使用したりするかを見ていきましょう。

Subversion の作業コピーは、ローカルシステム上にある通常のディレクトリツリーで、ここにファイルが集められています。お望みのファイルをどれでも編集することができますし、それがソースコードのファイルなら、いつも通りそのファイルからプログラムをコンパイルできます。作業コピーは自分の個人的な作業場所です。 Subversion が他人の変更をここに組み込むことはありませんし、同様に、明確に指示するまで自分の変更が他人に公開されることもありません。

作業コピーのファイルに変更を加え、それがうまく動作することを確認したあとで、その変更を同じプロジェクトで一緒に作業しているほかの人に（リポジトリに書き込むことで） 公開 するためのコマンドが Subversion には用意されています。また、他の人の変更が公開されたときは、その変更を（リポジトリから読み出して）自分の作業コピーにマージするコマンドが用意されています。

作業コピーには、 Subversion のコマンドの実行を補助するために、作成・保守される特別なファイルがあります。具体的には、作業コピーのディレクトリごとに `.svn` という名前のサブディレクトリが作られます。これは作業コピーの 管理ディレクトリ として知られています。管理ディレクトリのそれぞれのファイルは、まだ公開していない変更があるか、また他の人の作業によって最新でなくなっているかを Subversion が認識する助けになります。

典型的な Subversion リポジトリでは、いくつものプロジェクトのファイル（やソースコード）を保持し、リポジトリのファイルシステムツリー以下に、プロジェクトごとのサブディレクトリを置いています。この配置では、ユーザーの作業コピーは大抵リポジトリの特定のサブツリーに相当します。

例えば、2つのソフトウェアプロジェクトがあるリポジトリがあったとします。

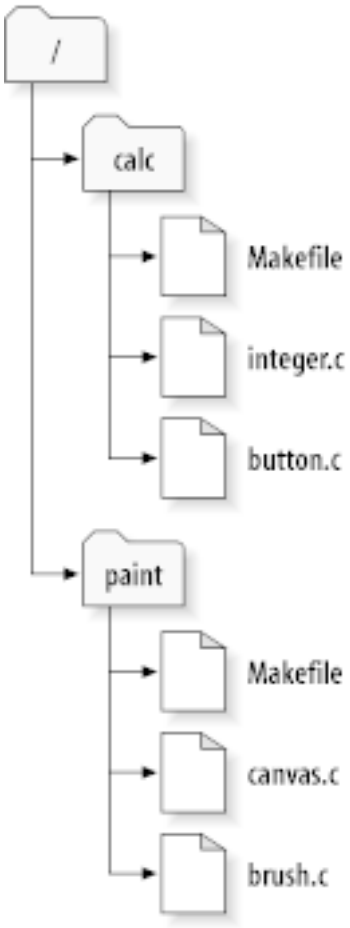


図2.6 リポジトリのファイルシステム

具体的には、リポジトリのルートディレクトリに paint と calc というサブディレクトリがあります。

作業コピーを取得するために、リポジトリのサブツリーを チェックアウト しなければなりません（チェックアウト は、リソースのロックや予約が必要なように思われますが、そんなことはありません。単に、プロジェクトのプライベートなコピーを作成するだけです）。

button.c を変更したとしましょう。 .svn ディレクトリには、ファイルの更新日時と元の内容が記録されていますから、Subversion はファイルが変更されていることがわかります。しかし、変更は明確に指示するまで公開はされません。変更を公開する操作は通常、変更をリポジトリへ コミット する（もしくは チェックイン する）と呼ばれています。

変更を他の人に公開するには、Subversion のコミット（commit）コマンドを使用します。

さて、button.c への変更を、リポジトリにコミットした後で、他のユーザーが /calc の作業コピーをチェックアウトすると、変更が反映された最新版のファイルが取得されます。

自分にサリーという相棒がいて、/calc の作業コピーを自分と同時にチェックアウトしたとします。自分が button.c の変更をコミットしても、サリーの作業コピーは変更されません。 Subversion はユーザーが操作しないと作業コピーを更新しないからです。

サリーのプロジェクトを最新状態にするには、Subversion に作業コピーを 更新 するよう指示します。これは Subversion の更新（update）コマンドを使用します。これによって、自分の変更がサリーの作業コピーに組み込まれ、またチェックアウト後にコミットされた他の人の変更も同様に組み込まれます。

サリーは、どのファイルを更新するかを指定する必要はありません。Subversion は .svn ディレクトリの情報とリポジトリ内の詳細情報を使用して、どのファイルを更新するか決めるのです。

2.3.2. リポジトリ URL

Subversion のリポジトリは、ローカルディスクや様々なネットワークプロトコル経由など、多様な方法でアクセスできます。しかし、リポジトリの位置は常に URL で表します。URL スキーマはアクセス方法を表します。

スキーマ	アクセス方法
file://	ローカルないしネットワークドライブのリポジトリに直接アクセスします。

表2.1 リポジトリにアクセスするURL

Subversion のURLの大部分には標準的な文法を使用し、URLの中にサーバー名やポート番号を含めることができます。file:// によるアクセス方法は、通常はローカルアクセスに使用しますが、ネットワーク上のホストに対する UNC パスにも使用できます。そのためURLは file://hostname/path/to/repos のような形式になります。同一マシン内では URL の hostname の部分を省略するか localhost としてください。このためローカルパスを表すには、file:///path/to/repos のようにスラッシュが3つ並べることになります。

また Windows 上で file:// スキームを使用する場合は、同じマシン上のカレントドライブとは別のドライブにあるリポジトリにアクセスするためには、非公式の「標準」構文を使う必要があります。以下のURLパス構文のどちらか一方を使えばアクセスできます。ここで X はリポジトリのあるドライブです。

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

Windows の(URL以外の)パスの形式ではバックスラッシュを使用しますが、URLには通常スラッシュを使用することに注意してください。

ネットワークフォルダー上でも FSFS リポジトリには安全にアクセスできますが、BDB リポジトリにはこの方法ではアクセス できません 。

警告

Berkeley DB リポジトリは、ネットワークフォルダー上で作成したりアクセスしたりしないでください。これはリモートファイルシステム上に存在 できません 。ネットワークドライブにドライブ文字を割り当ててもダメです。ネットワークフォルダー上で Berkeley DB を使用した場合、結果が予想できません。すぐに不可思議なエラーに遭遇するかもしれませんし、数ヶ月後にリポジトリデータベースが破損されるかもしれません。

2.3.3. リビジョン

svn commit (コミット)の操作は、ファイルやディレクトリの変更を、単一の不可分トランザクションで公開します。作業コピーで、ファイル内容の変更、ファイルやディレクトリの作成、削除、名前の変更、コピーなど行い、それ全体を単位とした完全な変更セットをコミットできます。

リポジトリでは、コミットを不可分トランザクションとして扱い、変更をすべて配置するか、何も配置しないかのどちらかになります。 Subversion はプログラムの異常終了や、システムの異常終了、ネットワーク障害、他のユーザーの操作に直面しても、この最小単位を保持しようとします。

リポジトリがコミットを受け付けるごとに、ファイルシステムツリーの状態が新しく作成されます。これを リビジョン と呼びます。リビジョンごとに一意の自然数が割り当てられ、直前のリビジョンよりも1だけ大きくなります。リポジトリを新規作成した直後のリビジョンは 0 で、ルートディレクトリ以外はなにも含まれていません。

リポジトリを視覚化するよい方法は、ツリーの連続で表すというものです。0 から始まるリビジョン番号の配列が、左から右に追加されていく状況を想像してください。それぞれのリビジョン番号には対応したファイルシステムツリーがあり、それぞれのツリーはコミット後のリポジトリの状態を示す「スナップショット」です。

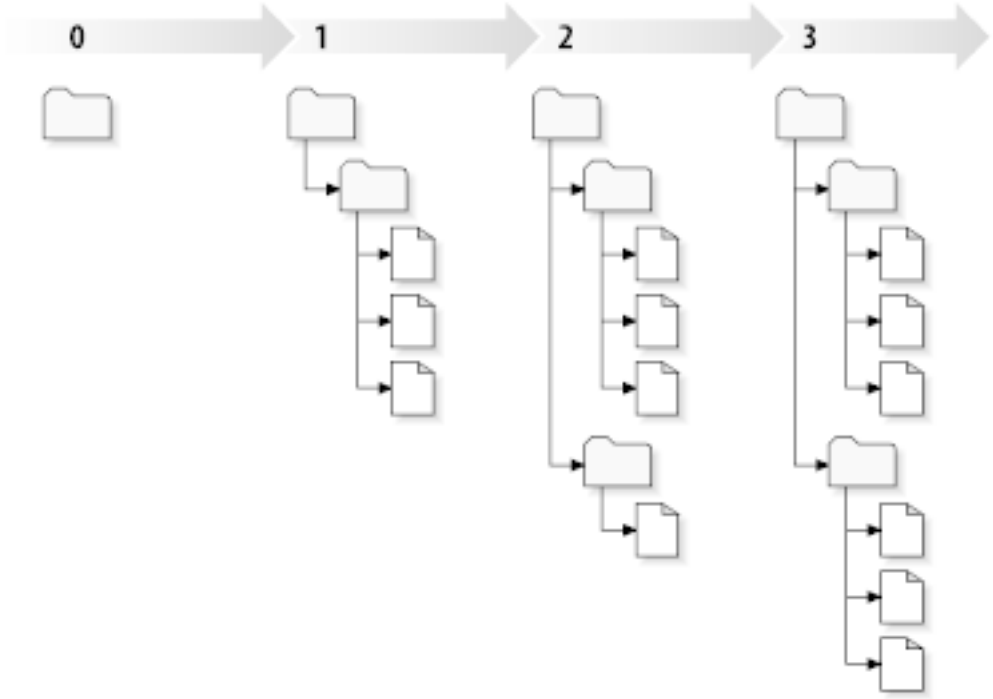


図2.7 リポジトリ

グローバルリビジョン番号

多くのバージョン管理システムとは異なり、Subversion のリビジョン番号は ツリー全体 に対して付与されるもので、個々のファイルに付与されるものではありません。それぞれのリビジョン番号はツリー全体を特定し、ある変更をコミットした後のリポジトリの特定の状態を表します。言い換えると、リビジョン N は N 番目のコミット後のリポジトリファイルシステムの状態を表すと言えます。Subversion のユーザーが、「リビジョン 5 の `foo.c`」と言うとき、それが実際に意味するものは、「リビジョン 5 に現れる `foo.c`」です。一般に、あるファイルのリビジョン N と M は異なっているとは 限りません 。

作業コピーが、リポジトリの特定のリビジョンに常に対応しているとは限らないということに充分注意してください。複数の異なるリビジョンが混在している可能性があります。たとえば、最新のリビジョンが 4 であるリポジトリから、作業コピーをチェックアウトしたとします。

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

この時点では作業ディレクトリは、リポジトリのリビジョン 4 と完全に一致します。しかし、ここで `button.c` に変更を加え、変更をコミットしたとします。他にコミットした人がいなければ、このコミットではリポジトリにリビジョン 5 を作成し、作業コピーの内容は以下のようになります。

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

この時点で、サリーが integer.c を変更し、リビジョン 6 を作成したとします。svn update で自分の作業コピーを更新すると、以下のようになります。

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

サリーの integer.c への変更は、自分の作業コピーに現れますが、button.c へ行った自分の変更はそのままです。この例では、Makefile の内容はリビジョン 4、5、6 で全く同じものですが、Subversion は作業コピー中の Makefile のリビジョンを 6 にして、最新であることを表現します。そのため作業コピーのトップでまっさらな更新を行うと、一般的に、作業コピーはリポジトリの特定のリビジョンに完全に一致します。

2.3.4. 作業コピーのリポジトリ追跡方法

作業ディレクトリ内のファイルに対して、Subversion は2つの本質的な情報を .svn/ 管理領域に記録します。

- 作業しているファイルが基づいているリビジョン（これは 作業リビジョン と呼ばれる）、および、
- ローカルコピーをリポジトリから更新したときに記録したタイムスタンプ、です。

この情報とリポジトリとのやりとりによって、 Subversion は、作業ファイルが以下の4つの状態のどれかを確認できます。

変更しておらず、最新版である
: 作業ディレクトリのファイルが変更されておらず、作業リビジョン以降に行われたリポジトリへのコミットでもそのファイルが変更されていない状態です。そのファイルに対するコミット（commit）は何も行われないし、更新（update）も何も行われません。

ローカルで変更しており、最新版である
: 作業ディレクトリのファイルは変更されているが、作業リビジョン以降に行われたリポジトリへのコミットではそのファイルが変更されていない状態です。ローカルにはコミットしていない変更があるので、そのファイルに対するコミット（commit）は成功し、ローカルの変更が公開されます。また、更新（update）では何も行われません。

変更しておらず、最新版ではない
: 作業ディレクトリのファイルは変更されていないが、リポジトリ上のファイルは変更されている状態です。このファイルを最新の公開リビジョンにするため、どこかで更新しなければなりません。このファイルへのコミット（commit）は何も行われませんが、更新（update）では作業コピーに最新の変更が反映されます。

ローカルで変更しておらず、かつ最新版ではない
: ファイルは作業ディレクトリでも、リポジトリ上でも変更された状態です。このファイルへのコミット（commit）は 作業コピーは最新ではありません というエラーになります。まず、そのファイルを更新しなければなりません。ファイルに対して更新（update）を行うと、公開されている変更点が、ローカルで変更を行った作業コピーにマージされます。これが自動的にできないような状況の場合、ユーザーが競合の解決を行うため、そのままになります。

2.4. まとめ

この章では Subversion の基本的な考え方の数々を一覧しました。

- リポジトリによる集中管理、クライアントの作業コピー、リポジトリのリビジョンツリーの配列、といった概念を紹介しました。
- 2人の共同作業者が Subversion 上で、「コピー・変更・マージ」モデルを使用して、どのようにお互いの変更を公開・取得するかを簡単な例で見てきました。
- Subversion が作業コピーの情報を追跡・管理する方法について少し触れました。

第3章 リポジトリ

リポジトリにアクセスするためにどのプロトコルを使用する場合でも、少なくとも1つはリポジトリを作成する必要があります。リポジトリは Subversion のコマンドラインクライアントでも、TortoiseSVN でも作成できます。

まだ Subversion のリポジトリを作成していないのなら、さっそく作成しましょう。

3.1. リポジトリの作成

リポジトリは FSFS バックエンドか、より古い Berkeley Database (BDB) フォーマットで作成できます。FSFS フォーマットは一般的に高速で、管理しやすく、ネットワークフォルダーや Windows 98 でも問題なく動作します。BDB フォーマットは長い間テストされてきているので安定していると考えられてきましたが、 FSFS にも数年の使用実績があり、その主張は弱くなってきています。詳しくは Subversion Book の Choosing a Data Store [http://svnbook.red-bean.com/en/1.7/svn.reposadmin.planning.html#svn.reposadmin.basics.backends] をご覧ください。

3.1.1. コマンドラインクライアントを使用したリポジトリの作成

1. リポジトリのルートフォルダーとして SVN という空のフォルダー(ここでは D:¥SVN¥ とします)を作成してください。
2. D:¥SVN¥ の中に MyNewRepository というフォルダーを作成してください。
3. コマンドプロンプト(またはDOSボックス)を開き、D:¥SVN¥ に移動して、

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

もしくは

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

と入力してください。

これで、D:¥SVN¥MyNewRepository に新しいリポジトリを作成できました。

3.1.2. TortoiseSVNを使用したリポジトリの作成

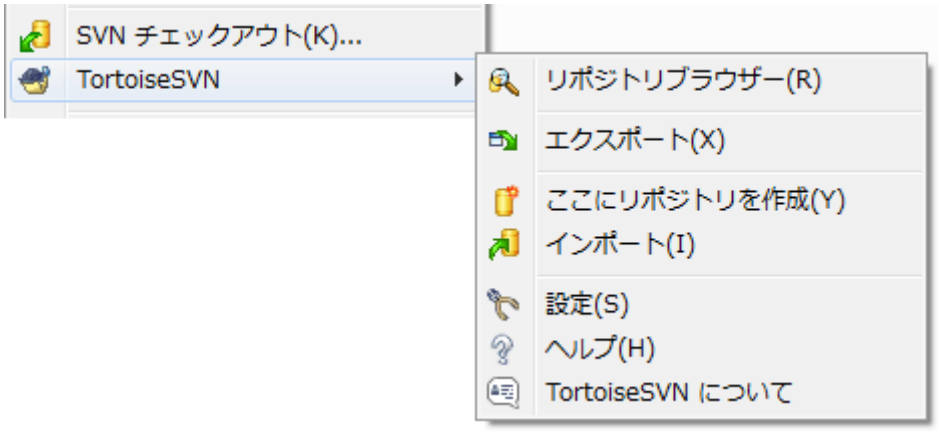


図3.1 バージョン管理外フォルダーの TortoiseSVN メニュー

1. Windows エクスプローラーを開いてください。

2. 新しいフォルダーを作成し、名前をつけてください（例えば SVNRepository）。

3. 新しく作成したフォルダーで右クリックし、TortoiseSVN → ここにリポジトリを作成... を選択してください。

リポジトリは新しいフォルダーの内部に作成されます。 そのファイルを自分で編集しないでください!!! エラーが発生したら、フォルダーが空であるか、または書き込み禁止されていないかを確認してください。

リポジトリ内にディレクトリ構造を作成したいと思うかもしれません。その場合は、「リポジトリのレイアウト」に示したレイアウトオプション参照してください。

ローカルのリポジトリを識別しやすくするために、TortoiseSVN ではリポジトリを作成する際にカスタムフォルダーアイコンが設定されます。公式のコマンドラインクライアントを使用してリポジトリを作成した場合は、このフォルダーアイコンは設定されません。

ヒント

TortoiseSVN で BDB リポジトリを作成することはできません。しかし、コマンドラインクライアントを使用すれば BDB リポジトリを作成できます。一般に FSFS リポジトリの方が保守が容易ですし、BDB にはバージョン間の互換性問題があるので、そうした方が私たちが TortoiseSVN を保守するのが楽になるのです。

この互換性問題のため、 TortoiseSVN は、BDB リポジトリの file:// アクセスをサポートしていません。しかし、サーバー経由でアクセスする svn://、http://、https:// の各プロトコルはサポートしています。ですから、 file:// プロトコルでアクセスしなければならない新しいリポジトリは、FSFS で作成することを強くお勧めします。

もちろん、ローカルのテスト用途でない限り、 file:// アクセスは使用しないこともお勧めします。サーバーを使用した方が、一人で開発する場合を除いて、全てにおいてより安全でより信頼性が高くなります。

3.1.3. リポジトリへのローカルアクセス

ローカルリポジトリにアクセスするには、そのフォルダーへのパスが必要です。Subversion ではすべてリポジトリのパスを file:///C:/SVNRepository/ という形で表すことを覚えておいてください。すべてスラッシュ（「/」）を使用することに注意してください。

ネットワークフォルダー上のリポジトリにアクセスするには、ドライブ文字の割り当てと UNC パスの両方が使えます。UNC パスは file://ServerName/path/to/repos/ といった形です。先頭にスラッシュ（「/」）が2つあることに注意してください。

SVN 1.2 以前では、UNC パスは file:///¥¥ServerName/path/to/repos というもっと曖昧な形でした。この形もまだサポートしていますが、お勧めしません。

警告

Berkeley DB リポジトリは、ネットワークフォルダー上で作成したりアクセスしたりしないでください。これはリモートファイルシステム上に存在 できません 。ネットワークドライブにドライブ文字を割り当ててもダメです。ネットワークフォルダー上で Berkeley DB を使用した場合、結果が予想できません。すぐに不可思議なエラーに遭遇するかもしれませんし、数ヶ月後にリポジトリデータベースが破損されるかもしれません。

3.1.4. ネットワークフォルダー上のリポジトリへのアクセス

原理上、FSFS リポジトリはネットワークフォルダー上に配置でき、file:// プロトコルを用いて複数のユーザーからアクセスできますが、これは絶対にお勧め しません 。実際のところ、このような使い方を私たちは思いとどまってほしいと 強く 思いますし、サポートもしません。

第一に、すべてのユーザーにリポジトリへ直接書き込みアクセスする権限を与えることになるので、誰かが誤ってリポジトリ全体を削除したり、何らかの方法で使用できなくしたりすることができてしまいます。

第二に、すべてのネットワークファイル共有プロトコルが、 Subversion が要求するロック機能をサポートしているわけではなく、リポジトリが壊れた状態に見えてしまう可能性があります。すぐには起こらないかも知れませんが、ある日2人のユーザーが、同時にリポジトリにアクセスしようとすることもあるでしょう。

第三に、ファイルのアクセス権をきちんと設定しなければなりません。ネイティブの Windows 共有ではうまくいくかも知れませんが、 Samba では特に困難です。

file:// アクセスは、ローカルでの1ユーザーのみのアクセスを想定しており、そのようにテストとデバッグを行っています。リポジトリを共有したい場合は、まさに 適切なサーバーをセットアップする必要がある場面であり、それは見かけほど難しくはありません。サーバーの選択・設定のガイドラインは、「リポジトリへのアクセス」をお読みください。

3.1.5. リポジトリのレイアウト

データをリポジトリにインポートする前に、データをどのような構成にするかをはじめに考える必要があります。おすすめのレイアウトを採用すれば、あとがより楽になるでしょう。

リポジトリの構成には標準的な、おすすめの方法があります。多くの人は開発の「メインライン」を保持するためのトランク（trunk）ディレクトリ、ブランチコピーを入れるブランチ（branches）ディレクトリ、そしてタグコピーを入れるタグ（tags）ディレクトリを作成します。リポジトリに単一のプロジェクトでしか使わない場合には、次のように、この3つのディレクトリをリポジトリ最上位に作ります。

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

このレイアウトは一般的に使われていますので、 TortoiseSVN で新しくリポジトリを作成する際には、このディレクトリ構造を作成をするかどうか確認します。

リポジトリに複数のプロジェクトがあるなら、以下のようにサブディレクトリでまとめたレイアウトにしたり、

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

以下のようにプロジェクトでまとめたりします。

```
/paint/trunk
/paint/branches
```

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

プロジェクト同士があまり密接に連携しておらず、それぞれ個々にチェックアウトできるなら、プロジェクトごとにまとめる方法が便利でしょう。プロジェクト同士が連携している場合には、一度に全てチェックアウトしたいと思うことでしょうし、ひとつの配布パッケージに結合されるプロジェクトなどはサブディレクトリでまとめた方がいいでしょう。この方法ならば、ひとつのトランクをチェックアウトするだけで済み、またサブプロジェクト間の関係も容易に把握できます。

トップレベルに /trunk /tags /branches を配置するアプローチを採用する場合でも、ブランチやタグを作るたびにトランクの全体をコピーしなければならないと言うことはありません。また、ある意味ではこの構造は最も柔軟な形となります。

関連がないプロジェクトはリポジトリを分けるという方法もあります。変更をコミットする際、リビジョン番号はプロジェクトごとに振られるのではなく、リポジトリ全体の変更に振られます。関連のない2つのプロジェクトがリポジトリを共有すると、リビジョン番号に大きなギャップが生じます。 Subversion プロジェクトと TortoiseSVN プロジェクトは同じホストアドレスにありますが、独立して開発できるよう、またビルド番号で混乱しないよう、リポジトリが分けられています。

もちろん、以上の標準的なレイアウトを無視しても構いません。自分たちでうまく作業できるように、様々な整理の仕方があるでしょう。ここで選択したものを、永遠に使い続けなければならないわけではないということを覚えておいてください。いつでもリポジトリを再構成できます。ブランチやタグは特定のディレクトリですから、 TortoiseSVN はお好みにあわせて移動したり名前を変更したりできます。

あるレイアウトから別のレイアウトへ切り替えるのは、ただ単にサーバー側で移動するだけです。リポジトリの構成が気に入らなければ、ディレクトリを移動させてください。

そのため、まだ基本的なフォルダー構造を、リポジトリに作成していないのであれば、今のうちにやっておくべきです。これを行うには、2通りの方法があります。 /trunk /tags /branches 構造を単に作成したいだけであれば、リポジトリブラウザーを使用して、3つのフォルダーを作成できます（3つのコミットに分かれます）。もっと深い階層を作成する場合は、まずディスクにフォルダー構造を作成し、その後一度にコミットする方法が簡単です。以下のように行います。

1. ハードディスクに新しい空のフォルダーを作成してください。

2. このフォルダーの中に任意のトップレベルフォルダー構造を作成してください。まだ中にはファイルを置かないでください。

3. このフォルダー構造を含んでいるフォルダーの上で右クリックして、TortoiseSVN → インポート... を選択することで、フォルダ構造をリポジトリにインポートすることができます。インポートダイアログでは、リポジトリのURLを入力してOKをクリックしてください。これで、 temp フォルダーがリポジトリのルートの中に入り、基本的なリポジトリ構造を作ることができます。

なお、インポートするフォルダーの名前はリポジトリに現れず、その内容のみがインポートされます。たとえば、以下のようなフォルダー構造を作成した場合、

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

リポジトリルートに C:¥Temp¥New をインポートすると、次のようになります。

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

3.2. リポジトリのバックアップ

どのタイプのリポジトリを使用する場合でも、定期バックアップの保守作業を行い、バックアップを検査することは非常に重要です。サーバーを破損すると、最新のバージョンのファイルにはアクセスできるかもしれませんが、リポジトリがなければ履歴が永遠に失われてしまいます。

簡単な(しかしお勧めできない)方法は、リポジトリのフォルダーをバックアップメディアに単にコピーすることです。しかし、そのデータにアクセスするプロセスが全くないことを確認しなければなりません。ここでいうアクセスとは、すべての アクセスです。 BDB リポジトリは、ステータスの取得のような、読み込みしか必要でない操作の時でも書き込みを行います。コピー中のリポジトリに何かアクセスがあると(Webブラウザーを開いたままにする、WebSVN、等々)、バックアップがおかしくなります。

お勧めの方法は、

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

を実行して、リポジトリのコピーを安全な方法で作成することです。それからこのコピーをバックアップしてください。 `--clean-logs` オプションは必須ではありませんが、BDB リポジトリをバックアップする際に、冗長なログを削除して容量を節約します。

Subversion のコマンドラインクライアントをインストールすると、 `svnadmin` ツールが自動的にインストールされます。Windows PC でコマンドラインツールをインストールする場合は、Windows インストーラーバージョンをダウンロードするとよいでしょう。これは `.zip` バージョンより効率的に圧縮されているため、ダウンロードサイズが小さく、パスの設定も自動的に行ってくれるためです。Subversion のコマンドラインクライアントの最新バージョンは、Subversion [http://subversion.apache.org/packages.html] のウェブサイトからダウンロードすることができます。

3.3. サーバー側フックスクリプト

フックスクリプトは、リポジトリのイベント(新しいリビジョンの作成やバージョン管理外のプロパティの変更など)を引き金に動作するプログラムです。いずれのフックも、イベントが何か、操作の対象は何か、イベントを発生させた人物のユーザー名、といった情報を受け取ります。フックの出力や終了ステータスによって、フックプログラムは動作の続行、停止、中断を切り替えることになります。実装されているフックについての詳細は、 Subversion Book の Hook Scripts [http://svnbook.red-bean.com/en/1.7/svn.reposadmin.create.html#svn.reposadmin.create.hooks] の章を参照してください。

フックスクリプトは、リポジトリのあるサーバーで実行されます。TortoiseSVN では、イベントが発生した時にローカルで実行される、クライアント側フックスクリプトも設定できます。詳細は「クライアント側フックスクリプト」をご覧ください。

フックスクリプトのサンプルは、リポジトリの `hooks` ディレクトリにあります。これらのサンプルスクリプトは Unix/Linux サーバー用であり、Windows ベースのサーバーの場合は修正が必要となります。フックはバッチファイルもしくは実行形式です。以下のサンプルは pre-revprop-change フックを実装したバッチファイルの例です。

```
rem ログメッセージのみを変更可能にする。
if "%4" == "svn:log" exit 0
echo 「%4」プロパティは変更できません >&2
exit 1
```

標準出力へ出力されたものは、すべて破棄されてしまうことに注意してください。コミットを拒否するダイアログにメッセージを表示したい場合は、標準エラー出力に出力しなければなりません。このバッチファイルでは >&2 を使用しています。

フックを上書きする方法

フックスクリプトがコミットを拒否した場合、それが最終決定となります。しかし、スクリプトの中に 合言葉 を埋め込むことで、その操作を上書きすることができます。スクリプトが処理を拒否しようとした場合に、特殊なパスフレーズ、特殊なパスフレーズ、固定長のフレーズ、プリフィックス付きのファイル名などをログメッセージからスキャンさせ、合言葉を見つけた場合にはコミットを実行することができるようにします。もし合言葉が見つからない場合は、「合言葉が合いません」と表示してブロックするようにします。:-)

3.4. チェックアウトリンク

Subversion リポジトリを他の人に使用できるようにする場合は、ウェブサイトからリンクを張るかもしれません。他の TortoiseSVN のユーザーに チェックアウトリンク を提供すると、アクセスしやすくなります。

TortoiseSVN をインストールするときに、新しく tsvn: プロトコルを登録します。TortoiseSVN のユーザーがこのリンクをクリックすると、リポジトリの URL が入力された状態でチェックアウトダイアログが開くようになります。

このようなリンクを自分のHTMLページに含めるには、次のようなコードを追加してください。

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

もちろん、適切な画像を含めれば、もっと見栄えが良くなるでしょう。 TortoiseSVNのロゴ [http://tortoisesvn.net/images/TortoiseCheckout.png] を使用したり、好きな画像を使用したりすればよいでしょう。

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

また、特定のリビジョンを指定したリンクにすることもできます。次のようにしてください。

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

3.5. リポジトリへのアクセス

TortoiseSVN (または、その他の Subversion クライアント)を使用する場合、リポジトリを配置する場所が必要です。リポジトリをローカルに配置して、file:// プロトコルでアクセスしたり、サーバー上に配置して http:// や svn:// プロトコルでアクセスしたりすることもできます。2つとも暗号化したものを使用することができます。https:// や svn+ssh:// を使うか、SASL とともに svn:// を使用することもできます。

Google Code [http://code.google.com/hosting/] のような公開ホスティングサービスを使用している場合や、誰かがすでにサーバーをセットアップしている場合、何もする必要はありません。4章日常の使用ガイド に進んでください。

サーバーを持っておらず、一人で作業している場合や、 Subversion や TortoiseSVN を試験環境で評価中の場合、ローカルリポジトリを作成するのが最適かもしれません。 3章リポジトリ の始めで説明しているように、自分のPCにリポ

ジトリを作成してください。使い始めるための方法を探すのであれば、この章の残りを読み飛ばして 4章日常の使用ガイド へ直接進んでもかまいません。

複数のユーザーが使用するリポジトリを、ネットワークフォルダー上に構築しようと考えているなら、考え直してください。なぜこの方法が良くないのか、詳細は 「ネットワークフォルダー上のリポジトリへのアクセス」 をお読みください。サーバーの構築は見かけほど難しくありませんし、信頼性や、恐らく速度も向上します。

Subversion サーバーのオプションの詳細や、状況に応じた最適な構成については、 Subversion book の Server Configuration [http://svnbook.red-bean.com/en/1.7/svn.serverconfig.html] を参照してください。

初期のころの Subversion では、サーバーを設定するために、サーバー設定について充分に理解していなければならなかったため、このマニュアルの以前のバージョンでは、サーバーをセットアップする方法を詳細に説明していました。最近は、セットアップや設定プロセスをガイドしてくれる、パッケージ化されたサーバーインストーラーがいくつか配布されているので簡単になりました。私たちが把握しているインストーラーとしては、

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

があります。最新版は Subversion [http://subversion.apache.org/packages.html] のウェブサイトから、いつでも入手することができます。

それ以外の手順については、TortoiseSVN [http://tortoisesvn.net/usefultips.html] のウェブサイトで説明されています。

第4章 日常の使用ガイド

このドキュメントでは、TortoiseSVN クライアントの日々の使い方を説明しています。バージョン管理システムの解説 ではなく 、Subversion (SVN) の解説 でもありません 。やりたいことがおよそ分かっているが、どうやって操作するか思い出せない場合に、調べる場所として便利です。

Subversion でのバージョン管理の入門書が必要なら、Subversion によるバージョン管理 [http://svnbook.red-bean.com/] というすばらしい本をお薦めします。

このドキュメントは、TortoiseSVN や Subversion と同様、現在も作成途中です。間違いがあったら私たちが修正できるよう、メーリングリストに投稿してください。日常操作ガイドのスクリーンショットの中には、現在のソフトウェアの状態を反映していないものがあるかもしれません。ご容赦ください。TortoiseSVN の作業は空き時間でしているのです。

日常操作ガイドを最大限に活用するには、

- すでに TortoiseSVN をインストールしてあること
- バージョン管理システムになじみがあること
- Subversion の基本を知っていること
- サーバーのセットアップが済んでいるなど、Subversion のリポジトリにアクセスできるようになっていること

を前提とします。

4.1. 機能概要

この章では、このマニュアルのほぼすべてに適用される TortoiseSVN の機能の一部をを説明します。多くの機能は、Subversion の作業コピーの中でしか機能しないことに注意してください。

4.1.1. アイコンオーバーレイ

図4.1 エクスプローラーのアイコンオーバーレイ表示

TortoiseSVN のもっとも目に見える機能のひとつが、作業コピーのファイルに現れるアイコンオーバーレイです。これにより、ファイルが変更されているかどうかが一目瞭然となります。オーバーレイが表す内容については、「アイコンオーバーレイ」をご覧ください。

4.1.2. コンテキストメニュー

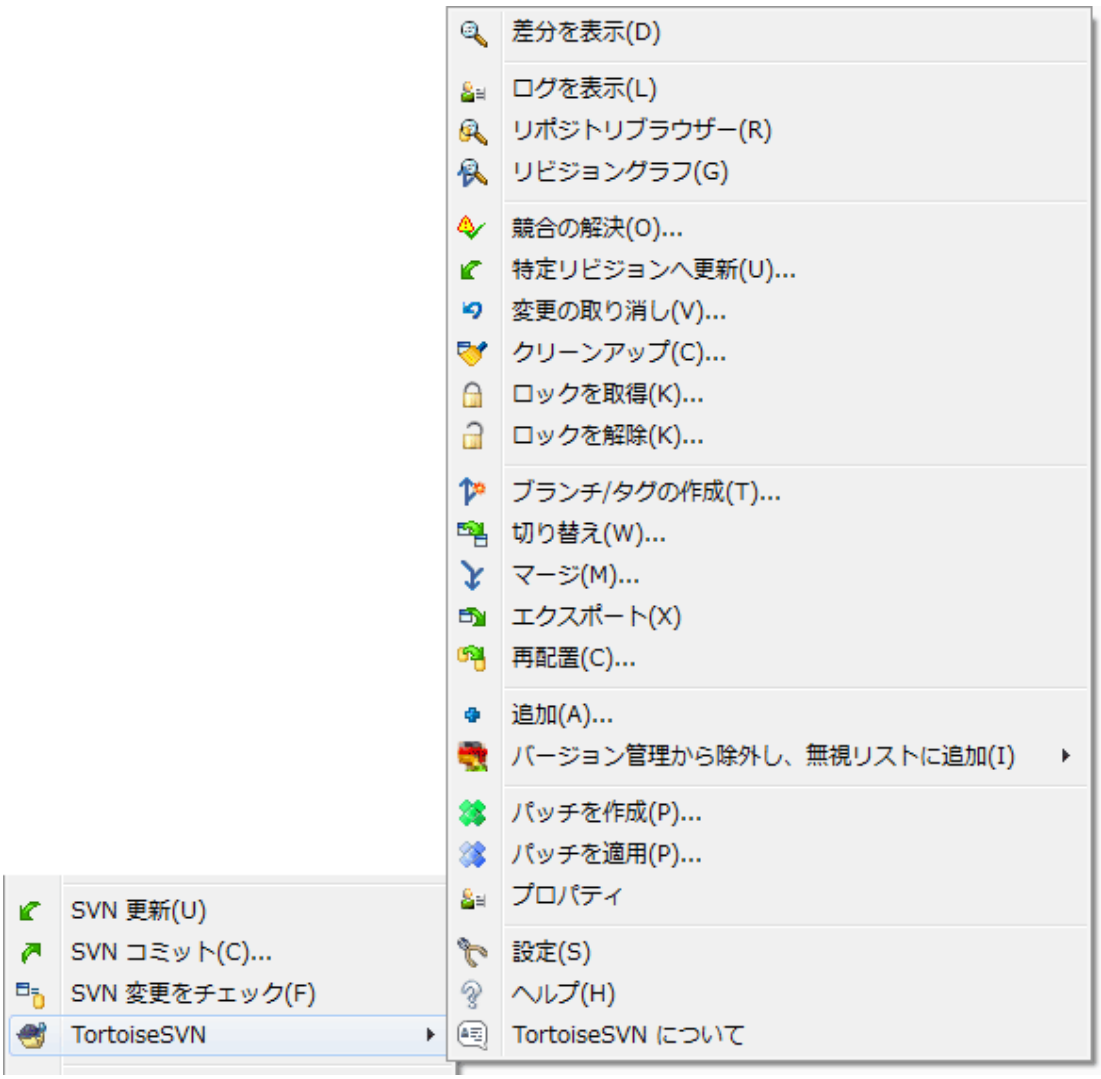


図4.2 バージョン管理下のフォルダーのコンテキストメニュー

TortoiseSVN のコマンドは、すべて Windows エクスプローラーのコンテキストメニューから呼び出します。ファイルやフォルダーを右クリックすると、多くは直接見えています。コマンドは、ファイルやフォルダーであるかどうか、その親フォルダーがバージョン管理下にあるかどうかで変わります。 TortoiseSVN のメニューは、エクスプローラーのファイルメニューにも現れます。

ヒント

めったに使用されない幾つかのコマンドは、拡張コンテキストメニューでのみ使用できます。拡張コンテキストメニューを表示するためには、Shift キーを押しながら右クリックしてください。

場合によっては、TortoiseSVN のエントリを、複数目にするかもしれません。これはバグではありません。

図4.3　バージョン管理されたフォルダー内のショートカットに対するエクスプローラーのファイルメニュー

この例では、バージョン管理下のフォルダー内にバージョン管理外のショートカットがある場合、エクスプローラーのファイルメニューに TortoiseSVN のエントリが 3つ 現れます。ひとつはフォルダー用、ひとつはショートカット用、最後にショートカットが指すオブジェクト用です。見分けるためにアイコンの右下に、ファイル、フォルダー、ショートカットのメニューエントリか、複数選択した項目かを示すマークがつきます。

Windows 2000 を使用している場合、コンテキストメニューはテキストのみとなり、メニューアイコンがつきません。以前のバージョンでは動作していましたが、マイクロソフトが Vista でアイコンハンドラの動作を変更したため、別の表示方法をとらねばならず、申し訳ありませんが Windows 2000 では動作しなくなりました。

4.1.3. ドラッグ&ドロップ

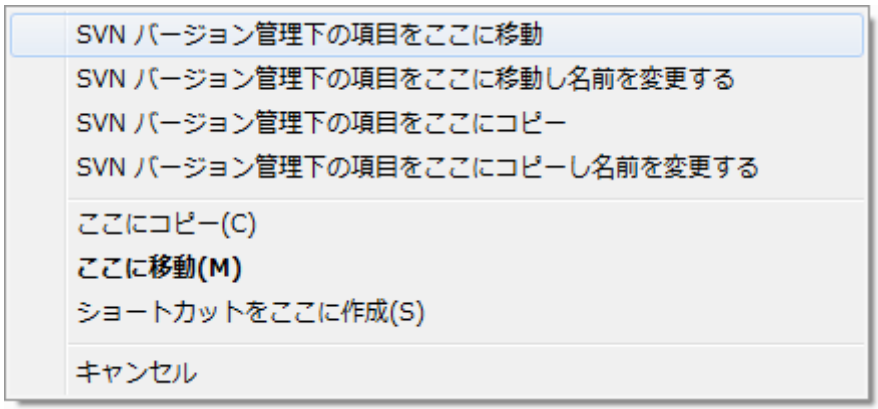


図4.4 バージョン管理下のディレクトリに対する右ドラッグメニュー

作業コピーの中にファイルやフォルダーを右ドラッグしたり、バージョン管理下のディレクトリにバージョン管理外のファイルやフォルダーを右ドラッグしたりすると、そのドラッグハンドラによって他のコマンドが有効になります。

4.1.4. 共通のショートカット

共通の操作は Windows のショートカットでよく知られていますが、ボタンに表示されるわけではありませんし、メニューもありません。「表示を更新する」のように、したいことが明確なのに方法が分からない場合は、ここをチェックしてみてください。

F1
: もちろんヘルプです。

F5
: 現在の表示を最新の情報に更新します。もしかしたら、最も便利なワンキーコマンドかもしれません。例えば……エクスプローラーでは、作業コピーのアイコンオーバーレイを更新します。コミットダイアログでは作業コピーを再走査し、コミットが必要なファイルを探します。リビジョンログダイアログではリポジトリに再接続し、最新の変更をチェックします。

Ctrl+A
: すべて選択します。エラーメッセージが出て電子メールにコピー&ペーストしたいときに使用できます。Ctrl+A でエラーメッセージを選択して...

Ctrl+C
: ...で選択した文字列をコピーします。

4.1.5. 認証

アクセスしようとしているリポジトリがパスワードで保護されている場合、認証ダイアログが表示されます。

図4.5 認証ダイアログ

ユーザー名とパスワードを入力してください。チェックボックスをチェックすると、証明書が保存されます。証明書は Subversion のデフォルトディレクトリ(`%APPDATA%¥Subversion¥auth`)にある、次の3つのサブディレクトリに格納されます。

- `svn.simple` には基本認証(ユーザー名/パスワード)の証明書が格納されます。パスワードは WinCrypt を使用して保存され、プレーンテキスト形式では保存されません。

- svn.ssl.server には SSL サーバー証明書が格納されます。

- svn.username には、(パスワードがいらない)ユーザー名のみで認証用する証明書が格納されます。

すべてのサーバーの認証情報を消去するには、TortoiseSVN の設定ダイアログの **保存されたデータ** ページから行えます。このボタンで、以前のバージョンでレジストリに格納した認証情報と一緒に、Subversion の auth ディレクトリからキャッシュされた認証データを消去します。「保存データの設定」を参照してください。

もし特定のサイトの認証情報を消去するには、これらのディレクトリを探して消去したい情報を含むファイルを見つけ、そのファイルを削除してください。

Windows のログオフ時やシャットダウン時に認証データを削除したい場合は、%APPDATA%¥Subversion¥auth ディレクトリを削除するシャットダウンスクリプトを使用すれば実現できます。例えば、

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

このようなスクリプトをインストールする方法は、http://www.windows-help-central.com/windows-shutdown-script.html に説明があります。

認証とアクセスコントロールに関して、サーバーをどのようにセットアップするかに関する情報は、「リポジトリへのアクセス」を参照ください。

4.1.6. ウィンドウの最大化

TortoiseSVN のダイアログの多くは、大量の情報を表示します。しかし、これを画面全体に最大化するよりは、縦方向・横方向のみ最大化した方が便利な場合があります。 **最大化** ボタンに便利なショートカットを用意しています。マウスで中クリックすると縦に最大化され、右クリックすると横方向に最大化されます。

4.2. リポジトリへのデータのインポート

4.2.1. インポート

すでにいくつかのプロジェクトが登録されているリポジトリにインポートする場合、リポジトリ構造はすでに決まっていることでしょう。新しいリポジトリにデータをインポートする場合には、どのようにリポジトリを構成するかを考えるのに、時間を費やす価値はあります。もっとアドバイスが必要であれば、「リポジトリのレイアウト」をお読みください。

この章では、 Subversion のインポート(import)コマンドについて解説しています。このコマンドは、ディレクトリ階層を一度にリポジトリにインポートするよう設計されています。この動作には、いくつか欠点もあります。

- 「常に無視するファイル」で設定する以外に、インポートするファイルやフォルダーを選択する方法がない。

- インポートしたフォルダーが作業コピーにならない。サーバーからファイルをコピーし反映するには、チェックアウトが必要である。

- 間違った階層のフォルダーを、リポジトリにインポートしてしまいがちである。

このような理由から、リポジトリに初期構造の /trunk /tags /branches を作成する場合を除いて、import コマンドを使用せずに、「その場でインポート」で説明している2段階の手順で実行することをお勧めします。ここでは、基本的なインポートのしかたを示します。

リポジトリにプロジェクトをインポートする前に、次の作業を行ってください。

1. プロジェクトの構築に必要ないファイル(一時ファイル、*.obj のようなコンパイラーが生成したファイル、コンパイルされたバイナリ、……)は削除してください。

2. フォルダーやサブフォルダーにファイルを整理してください。あとでファイルを削除・移動することもできますが、インポートの前にプロジェクトの構造をしっかり決めておくことを強くお勧めします。

そして Windows エクスプローラーで、プロジェクトのディレクトリ構造の最上位フォルダーを選択し、右クリックでコンテキストメニューを出してください。TortoiseSVN → インポート... コマンドを選択すると、次のダイアログボックスが表示されます。

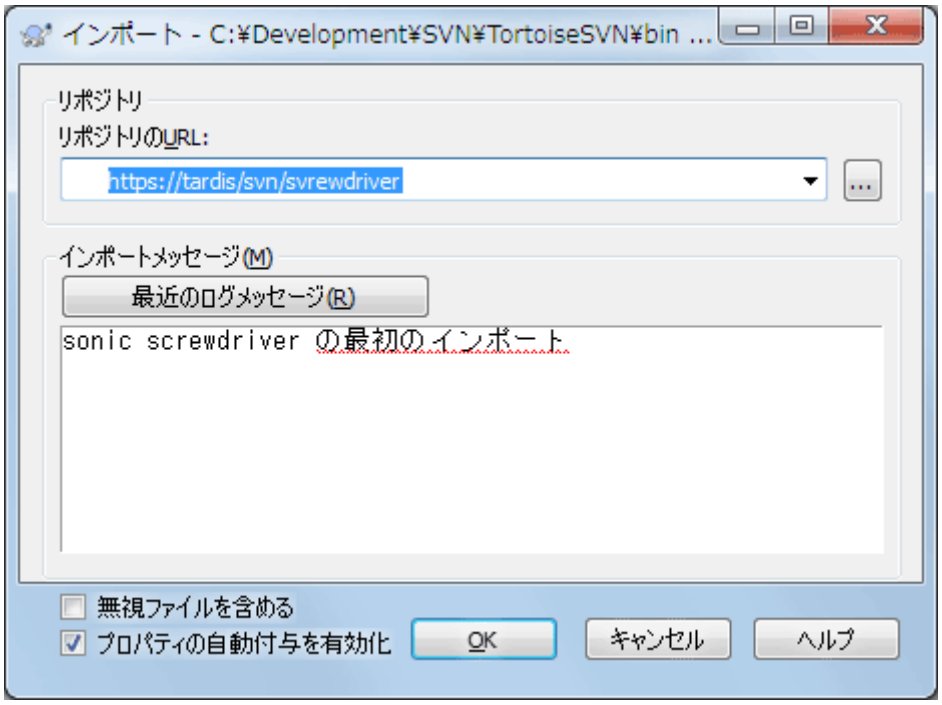


図4.6 インポートダイアログ

このダイアログで、プロジェクトのインポート先となるリポジトリの場所を示すURLを入力してください。リポジトリには、インポートで指定したローカルフォルダーは作成されず、その内容のみがインポートされるということ注意してください。例えば次のような構造があるとします。

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

ここで、 C:¥Projects¥Widget を http://mydomain.com/svn/trunk にインポートすると、Widget サブディレクトリではなく、その中のサブディレクトリが trunk の直下に現れるので、驚くかもしれません。サブディレクトリ名は、 http://mydomain.com/svn/trunk/Widget-X というようにURLの一部として指定する必要があります。インポートコマンドでは、リポジトリの中にサブディレクトリがない場合、自動的に作成されることに注意してください。

インポートメッセージは、ログメッセージとして使われます。

デフォルトでは、常に無視するパターンにマッチしたファイル・フォルダーはインポートされません。無視ファイルを含めるチェックボックスを使用すると、この動作を上書きします。常に無視するパターンの設定の詳細は 「一般設定」をご覧ください。

OK を押すと、TortoiseSVN は全てのファイルが入った完全なディレクトリを、リポジトリにインポートします。これでプロジェクトがバージョン管理下にあるリポジトリに格納されました。なお、インポートしたフォルダーは、バージョン管理下

に入りません。バージョン管理下の 作業コピー を取得するには、インポートしたときのバージョンをチェックアウトする必要があります。もしくは、フォルダーをその場でインポートする方法をご覧ください。

4.2.2. その場でインポート

既にリポジトリが存在していて、新しいフォルダー構造を追加する場合は、次のように操作してください。

1. リポジトリブラウザーを使用して、リポジトリに新しいプロジェクトフォルダーを直接作成します。標準的なレイアウトを使用している場合は、リポジトリのルート直下ではなく、 trunk のサブフォルダーとして作成した方が良いでしょう。リポジトリブラウザーは、ちょうど Windows のエクスプローラーのようにリポジトリ構造を表示しますので、どのように整理されているかを見ることができます。

2. インポートしたいフォルダーの最上位に対して、新しいフォルダーをチェックアウトしてください。ローカルフォルダーが空でないという警告が出ます。バージョン管理下のフォルダーの中に、バージョン管理外の中身があるという状態になります。

3. バージョン管理下のフォルダー上で TortoiseSVN → 追加... を使用して、内容の一部または全部を追加してください。ファイルの追加や削除、フォルダーへの svn:ignore プロパティの設定など、必要な変更を加えることができます。

4. 最上位のフォルダーをコミットしてください。これで新しいバージョン管理下のツリーと、既存フォルダーから作成したローカルの作業コピーを得られます。

4.2.3. 特殊なファイル

バージョン管理下のファイルに、ユーザー別のデータを含める必要がある場合があります。つまり、それぞれのユーザー・開発者別に、環境に合わせて変更しなければならないファイルがある場合です。そういったファイルは、ユーザーがそれぞれいつでもリポジトリにコミットしてしまうことができるために、バージョン管理が難しくなります。

このような場合、テンプレートファイルを使用することをお勧めします。開発者が必要な全てのデータを含むファイルを作成し、そのファイルをバージョン管理下に置きます。開発者はそのファイルをチェックアウトします。それから、それぞれの開発者がそのファイルのコピーを作成し、名前を変更します。そうするとコピーを変更してもなんの問題もありません。

たとえば、TortoiseSVN の構築スクリプトを参照してください。このファイルの名前は TortoiseVars.bat ですが、リポジトリにはありません。TortoiseVars.tmpl ファイルがあるだけです。TortoiseVars.tmpl はテンプレートファイルで、開発者ごとにコピーを作成し、TortoiseVars.bat と名前を変更するようになっています。このファイルの中には、ユーザーがどの行を編集・変更すればいいか判るように、セットアップのしかたをコメントとして追加しています。

ユーザーが不安にならないように、 TortoiseVars.bat を親フォルダーの無視リストに追加してあります。つまり、Subversion の svn:ignore プロパティをそのファイルにセットしています。これにより（バージョン管理外のファイルのように）コミットごとに表示されることがなくなります。

4.3. 作業コピーのチェックアウト

リポジトリから作業コピーを取得するには、 チェックアウト する必要があります。

Windows エクスプローラーで、作業コピーを作成したい場所のディレクトリを選択してください。 右クリック してコンテキストメニューを表示し、 TortoiseSVN → チェックアウト... コマンドを選択してください。すると次のダイアログが表示されます。

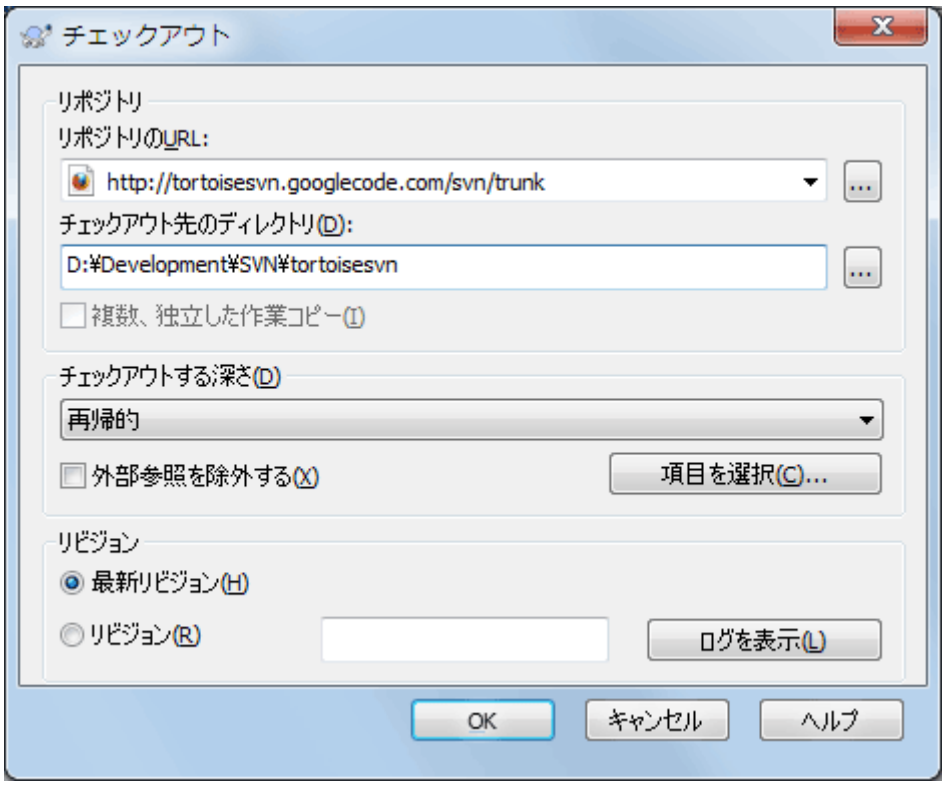


図4.7 チェックアウトダイアログ

存在しないフォルダー名を指定すると、その名前のディレクトリが作成されます。

4.3.1. チェックアウトの深さ

チェックアウトの際、深さ を選択することで、子フォルダーを再帰処理する深さを選択できます。大きなツリーの一部だけをチェックアウトしたい場合は、まずツリーの一番上のフォルダーのみをチェックアウトし、それから選択されたフォルダーを再帰的に更新してください。

再帰的
: 子フォルダーやそのサブフォルダーを含む、ツリー全体をチェックアウトします。

直接の子階層（フォルダーを含む）
: 指定したディレクトリと、直下のすべてのファイルや子フォルダーをチェックアウトしますが、子フォルダー内のファイルはチェックアウトしません。

子階層のファイルのみ
: 指定したディレクトリと、直下のすべてのファイルをチェックアウトしますが、子フォルダーはチェックアウトしません。

この項目のみ
: ディレクトリのみチェックアウトします。その中のファイルや子フォルダーはチェックアウトしません。

作業コピー
: 作業コピーに指定した深さを記憶します。このオプションは、チェックアウトダイアログでは使用されませんが、その他の深さの設定を持つダイアログでは、デフォルト設定となっています。

除外
: フォルダーをすでに取り込んでしまった後で、作業コピーの深さを小さくするために使用します。このオプションは、**特定リビジョンへ更新** ダイアログでしか使用できません。

手っ取り早く必要な項目だけをチェックアウトし、その項目だけを保持しておくのであれば、**項目を選択...** ボタンをクリックしてください。新しいダイアログが開くので、作業コピーに入れたいすべての項目をチェックして、それ以外の項目のチェックを外してください。作業コピーは**部分的なチェックアウト**という状態になります。この作業コピーを更新した場合は、チェックアウトされたファイルだけが更新され、存在しないファイルは更新されません。

作業コピーを部分的にチェックアウトした場合（つまり、チェックアウトする深さに**再帰的**以外を指定した場合）、その下のサブフォルダーを後から追加するには、リポジトリブラウザー（「リポジトリブラウザー」）または「変更をチェック」ダイアログ（「ローカルとリモートの状態」）を使用してください。

Windows のエクスプローラーでは、チェックアウトされたフォルダーで 右クリック し、それから TortoiseSVN → **リポジトリブラウザー** でリポジトリブラウザーを起動してください。作業コピーを追加しようとしているサブフォルダーを選択し、**コンテキストメニュー → 特定リビジョンへ更新...** を実行してください。

「変更をチェック」ダイアログでは、まず、**リポジトリをチェック**ボタンをクリックしてください。リポジトリに存在していてチェックアウトされていないファイルやフォルダーがすべて、**リモート操作による追加**としてダイアログに表示されます。作業コピーに追加したいフォルダーを右クリックし、**コンテキストメニュー → 更新** を実行してください。

この機能は、大きなツリーの一部をチェックアウトする時だけでなく、ひとつの作業コピーを更新する際にも非常に便利です。 Project01 から Project99 といったサブフォルダーがある大きなツリーがあり、Project03、Project25、Project76/SubProj だけをチェックアウトしようとしているとしましょう。以下のような手順となります。

1. 親フォルダーを、深さを「この項目のみ」としてチェックアウトしてください。すると、空の最上位フォルダーができます。
2. リポジトリの内容を表示するため、このフォルダーを選択し、TortoiseSVN → **リポジトリブラウザー** を実行してください。
3. Project03 を右クリックし、**コンテキストメニュー → 項目を特定リビジョンへ更新...** を選択してください。デフォルトの設定のまま、OK をクリックしてください。これでそのフォルダーがすべて取得されます。

   Project25 に対して同じ手順を繰り返します。
4. Project76/SubProj に移動し、同様に行ってください。このとき Project76 フォルダーには、SubProj 以外の項目が取得されていないことに注意してください。内容を取得せずに、中間フォルダーのみが作成されたのです。

作業コピーの深さの変更

作業コピーをある深さでチェックアウトしたあと、**コンテキストメニュー → 項目を特定リビジョンへ更新...** を使用することで、後から深さを変更できます。そのダイアログで、 **深さを記憶する** のチェックボックスにチェックしてください。

古いサーバーを使用する場合

1.5以前のサーバーでは、作業コピーの深さの指定機能がないため、常にリクエストを効率的に処理できません。コマンドは正しく動作しますが、旧式のサーバーは、クライアントに必要でないデータまでフィルターせずに送ってしまうため、大量のネットワーク通信が発生する可能性があります。可能であれば、サーバーを1.5以上にアップグレードしてください。

プロジェクトに外部参照プロジェクトへの参照が含まれていて、同時にチェックアウト したくない 場合、**外部参照を除外する** チェックボックスを使用してください。

重要

外部参照を除外する にチェックを付けたり、深さを増やしたりしたい場合は、作業コピーを更新するのに TortoiseSVN → 更新 ではなく、TortoiseSVN → 特定リビジョンへ更新... を実行してください。通常の更新では外部参照を全て取得し、現在の深さを保持してしまいます。

ディレクトリツリーのトランク(`trunk`)部分か、それ以下をのみをチェックアウトすることをお勧めします。URLでディレクトリツリーの親パスを指定すると、プロジェクトのあらゆるブランチとタグを取得してしまうので、ハードディスクを使い切ってしまうかもしれません。

エクスポート

`.svn` ディレクトリを除いてコピーしたい場合があるかもしれません。例えば、ソースを圧縮したアーカイブファイルを作成するときなどです。どのように行うかは、「Subversion 作業コピーをエクスポート」をご覧ください。

4.4. 変更のリポジトリへのコミット

作業コピーへの変更を送信することを、変更を コミット するといいます。しかしコミットする前に、作業コピーが最新になっているかどうか確認しなければなりません。 TortoiseSVN → 更新 を直接行うか、はじめに TortoiseSVN → 変更をチェック を使用して、サーバーやローカルのファイルに変更がないかどうか確認します。

4.4.1. コミットダイアログ

作業コピーが最新で競合がない場合、変更をコミットする準備ができています。コミットするファイルやフォルダーを選択し、TortoiseSVN → コミット... を実行してください。

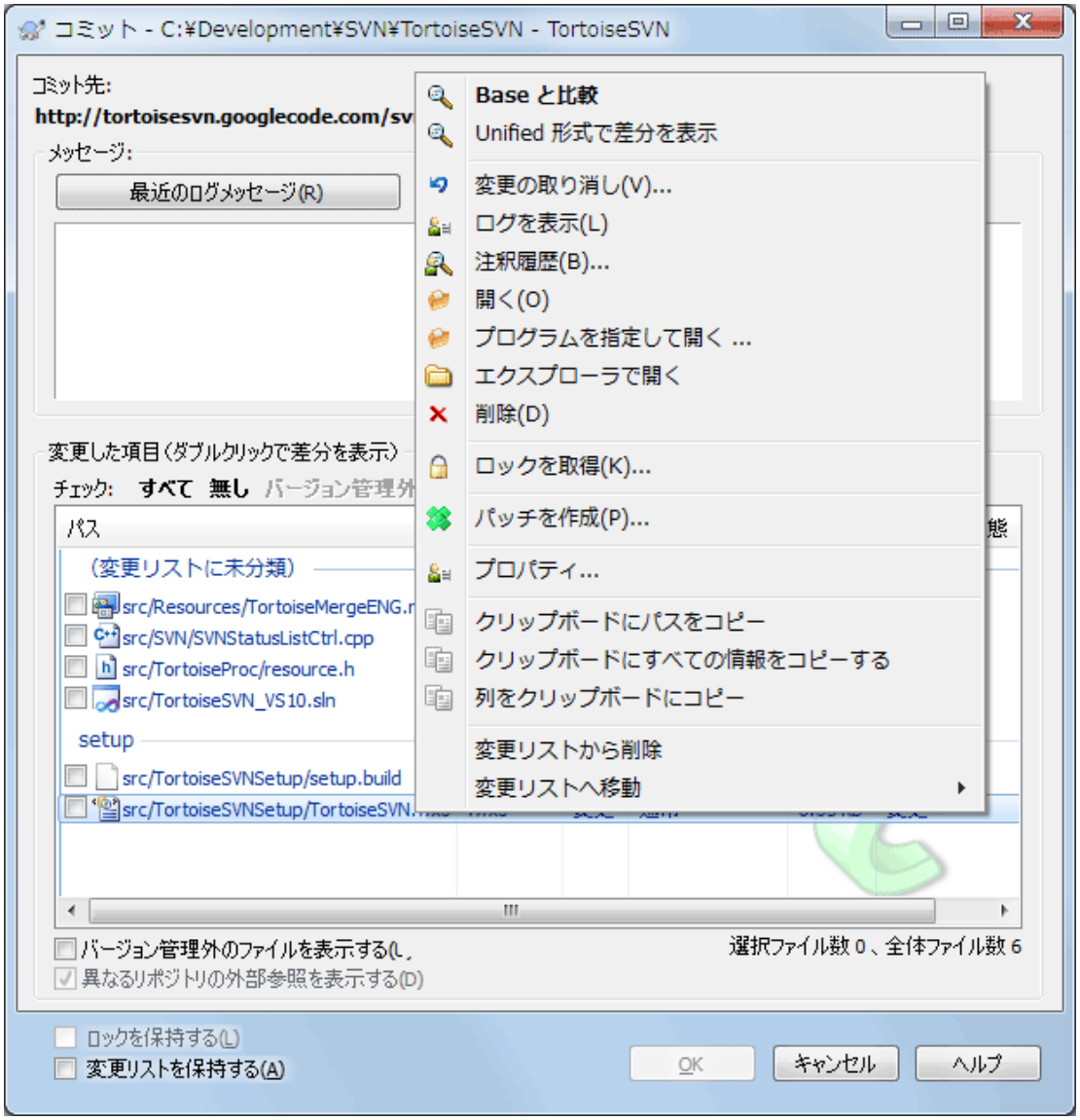


図4.8 コミットダイアログ

コミットダイアログには変更されたファイルや、追加・削除・バージョン管理外のファイルが表示されます。変更されたファイルをコミットしたくない場合は、ファイルのチェックをはずしてください。バージョン管理外のファイルを含める場合は、追加・コミットするファイルにチェックをつけてください。

別のリポジトリパスに切り替えた項目には、(s)マークが付きます。何かを切り替えてブランチで作業した後、トランクに戻すのを忘れている可能性もあります。これは警告マークです。

ファイルをコミットするかフォルダーをコミットするか

ファイルをコミットする際、コミットダイアログは選択したファイルのみ表示します。フォルダーをコミットする際には、コミットダイアログはファイルを自動で選択します。作成した新しいファイルを忘れていた場合でも、フォルダーをコミットすればそのファイルを探します。フォルダーをコミットするというのは、全てのファイルを変更したとマークつけるという意味ではありません。単に楽ができるということです。

コミットダイアログにある大量のバージョン管理外ファイル

コミットダイアログにバージョン管理外のファイル（例えば、コンパイラーが生成したファイルやエディターのバックアップなど）を表示しすぎだと思う場合は、以下のような方法で対処できます。

- 設定ページで除外リストにファイル（またはワイルドカード）を追加します。これはすべての作業コピーに影響します。
- TortoiseSVN → 無視リストに追加 で svn:ignore にファイルを追加します。svn:ignore プロパティを設定したディレクトリにのみ影響します。Subversion のプロパティダイアログで、そのディレクトリの svn:ignore プロパティを変更できます。

詳細は「ファイルやディレクトリの無視」をご覧ください。

コミットダイアログで変更のあるファイルをダブルクリックすると、変更を確認するよう外部差分ツールが起動します。スクリーンショットにあるようにコンテキストメニューでもっとオプションを指定できます。ここからファイルをドラッグして、テキストエディターやIDEといったアプリケーションに持っていくこともできます。

項目の左にあるチェックボックスをクリックして、選択状態を切り替えられます。ディレクトリに対して、 Shift を押しながら選択すると、再帰的に動作します。

下の欄に表示される列はカスタマイズすることができます。列見出しの上で右クリックすると、表示される列を選択するコンテキストメニューが表示されます。また、列の境界上にマウスを持っていくとドラッグハンドルが表示され、列幅を変更できます。以上のカスタマイズは保存されるので、次回も同じ列見出しで表示されます。

デフォルトでは変更をコミットすると、コミット完了後に保持していたファイルのロックが自動的に解除されます。ロックを保持したままにしたい場合は、**ロックを保持** チェックボックスにチェックしておきます。チェックボックスのデフォルト状態は、Subversion 設定ファイルの no_unlock オプションから取得されます。Subversion 設定ファイルの編集については、「一般設定」をご覧ください。

ドラッグ&ドロップ

作業コピーが同じリポジトリからチェックアウトされているなら、別の場所からコミットダイアログにファイルをドラッグできます。例えば、遠くの階層を見るのに、複数エクスプローラーのウィンドウを開かなければならないような、巨大な作業コピーも扱えるかもしれません。長々フォルダーを変更チェックする、トップレベルフォルダーからコミットするのを避けたければ、あるフォルダーのコミットダイアログを開き、他のウィンドウから同時に不可分コミットしたい項目をドラッグしてください。

作業コピーの中にあるバージョン管理外のファイルも、コミットダイアログにドラッグできます。その際は自動的に追加が実行されます。

コミットダイアログの下部のリストから、ログメッセージ欄にファイルをドラッグすると、その欄に全ファイルのパスが文字列で挿入されます。コミット時のログメッセージに、コミットにより影響を受けるパスを記述したい場合に便利です。

外部での名前変更の修復

Subversion　の外部でファイルの名前が変更された場合、ファイル一覧で紛失ファイルとして表示されたり、バージョン管理外ファイルとして表示されたりします。履歴を失わないように　Subversion

に関連を通知する必要があります。古い名前(紛失)と新しい名前(バージョン管理外)を選択し、 **コンテキストメニュー → 移動を修復** を実行すれば、名前の変更が行われたことを示すことができます。

外部でのコピーの修復

ファイルのコピーをしたのに、Subversion コマンドで行うのを忘れた場合、新しいファイルが履歴を失わないように、そのコピーを修復できます。古い名前(通常ないし変更)と新しい名前(バージョン管理外)を選択し、**コンテキストメニュー → コピーの修復** を実行するだけで、ファイルのコピーが行われたことを示すことができます。

4.4.2. 変更リスト

コミットダイアログは、関連するファイルのグループ化を助ける Subversion の変更リスト機能をサポートしています。この機能については、「変更リスト」をご覧ください。

4.4.3. コミット一覧からの項目の除外

バージョン管理下のファイルが頻繁に変更されても、コミットしたくないことがあります。これはビルドプロセスの弱点を表しています。なぜそのファイルをバージョン管理下に置いたのでしょう?テンプレートファイルを使用するべきではありませんか?しかし、時にこれはやむを得ないことがあります。古典的な理由としては、使用しているIDEが、ビルドする際に必ずタイムスタンプを更新してしまうということがあります。すべてのビルド設定を含んでいるため、プロジェクトファイルをバージョン管理下に置かなければなりませんが、単にタイムスタンプが更新されただけではコミットする必要はありません。

このような厄介なことを解決するのに、`ignore-on-commit` という変更リストを用意されています。この変更リストに追加されたファイルは、コミットダイアログで自動的にチェックが外されます。チェックを入れれば、変更をコミットすることができます。

4.4.4. コミットログメッセージ

コミットする変更を説明するログメッセージを必ず入力してください。後日プロジェクトのログメッセージを閲覧するときに、いつ、何が起きたかを確認できます。メッセージは長くても簡潔でもお好みでよいのですが、多くのプロジェクトで、使用する言語や、厳密なフォーマットといった記述する内容のガイドラインがあります。

メールで使用するような規約を用いて、ログメッセージの簡単な整形を行うことができます。`text` にスタイルを適用する場合、`*text*` で太字、`_text_` で下線、`^text^` で斜体になります。

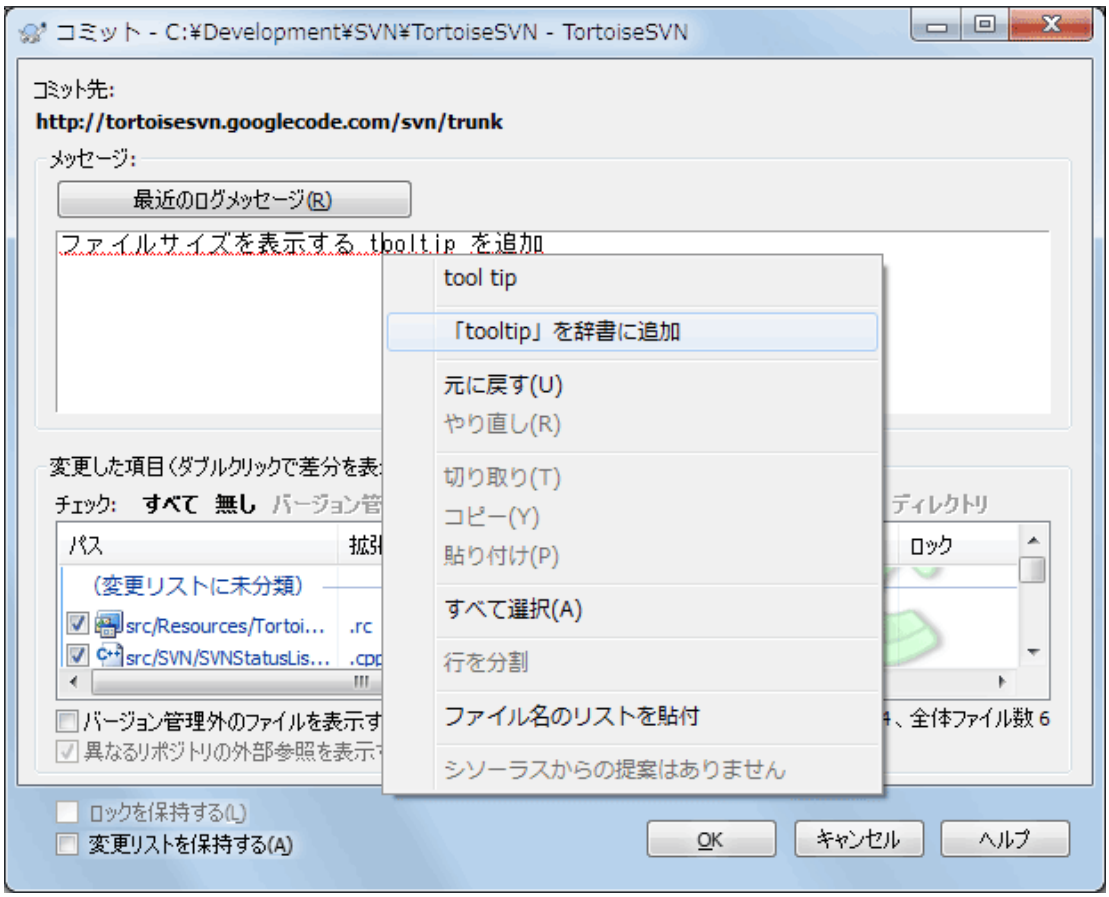


図4.9 コミットダイアログのスペルチェッカー

TortoiseSVN にはログメッセージを正しく書けるよう、スペルチェッカーが内蔵されています。これで間違った単語を強調表示できます。訂正候補にアクセスするには、コンテキストメニューを使用してください。もちろん、すべて の技術用語が登録されているわけではないので、正しい綴りの単語がエラーとして表示される可能性があります。こうした用語は、コンテキストメニューから個人辞書に登録できます。

ログメッセージウィンドウは、ファイル名や関数の自動補完機能も持っています。コミットしようとしている(テキスト)ファイルから、クラス名や関数名を(ファイル名と同様に)抽出するのに正規表現を使用します。3文字入力したり、 Ctrl +Space を押したりしたとき、このリストにある単語と一致すると、名前全体を選択するドロップダウンを表示します。TortoiseSVN が標準で持っている正規表現は、 TortoiseSVN をインストールした bin フォルダーに格納されています。自分で正規表現を定義し、 %APPDATA%¥TortoiseSVN¥autolist.txt に格納しておくこともできます。もちろん自分の autolist は TortoiseSVN をアップデートしても上書きされません。正規表現になじみがないのであれば、 http://ja.wikipedia.org/wiki/正規表現 [http://ja.wikipedia.org/wiki/%E6%AD%A3%E8%A6%8F%E8%A1%A8%E7%8F%BE] にある説明や、http://www.regular-expressions.info/ にあるオンラインドキュメントやチュートリアルをご覧ください。

完全に正しい正規表現を書くのは難しいので、式の組み合わせを整理しやすくするために、式を入力し、ファイル名を入力すれば、それがマッチするかどうかをテストできるテストダイアログを用意しています。コマンドプロンプトから、 TortoiseProc.exe /command:autotexttest とコマンドを入力すれば起動します。

以前入力したログメッセージを再利用できます。**最近のログメッセージ**をクリックして、その作業コピーで入力した最新のログメッセージ一覧を表示します。保存しているログメッセージの数は、TortoiseSVN 設定ダイアログでカスタマイズできます。

TortoiseSVN の設定ダイアログの **保存データ** ページで、保存されたコミットメッセージをすべて消去することができます。**最近のログメッセージ** ダイアログで Delete キーを押すと、特定の保存されたログメッセージを消去することができます。

入力欄の**コンテキストメニュー → ファイル名のリストを貼付**を実行すると、チェックしたパスをログメッセージに入れることができます。

ファイルリストにあるファイルをログメッセージの入力欄にドラッグするだけでも、ログメッセージにパスを入力することができます。

特殊なフォルダープロパティ

コミットログメッセージのフォーマットを制御したり、スペルチェッカーで使用する言語を指定する、特殊なフォルダープロパティがいくつかあります。詳細な情報は「プロジェクト設定」をご覧ください。

バグ追跡ツールとの統合

バグ追跡ツールが有効なら、**バグID/課題番号()**:

4.4.5. コミットの進行状況

OKを押すと、コミットの進行状況を表示するダイアログが表示されます。

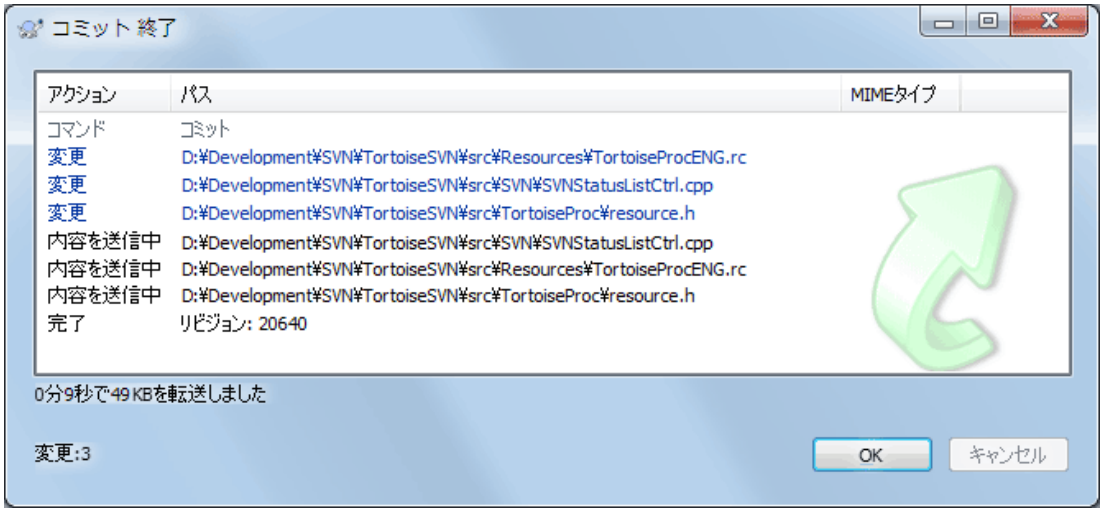


図4.10 コミットの状況を表示している進行ダイアログ

進行ダイアログでは、コミット状態に応じて色分けで表示されます。

青色
: 変更のコミット

紫色
: 新規追加のコミット

暗赤色
: 削除・置換のコミット

黒
　その他すべての項目

これはデフォルトの色設定ですが、設定ダイアログで色をカスタマイズできます。詳細は 「TortoiseSVN の色の設定」をご覧ください。

4.5. 他人の変更に伴う作業コピーの更新

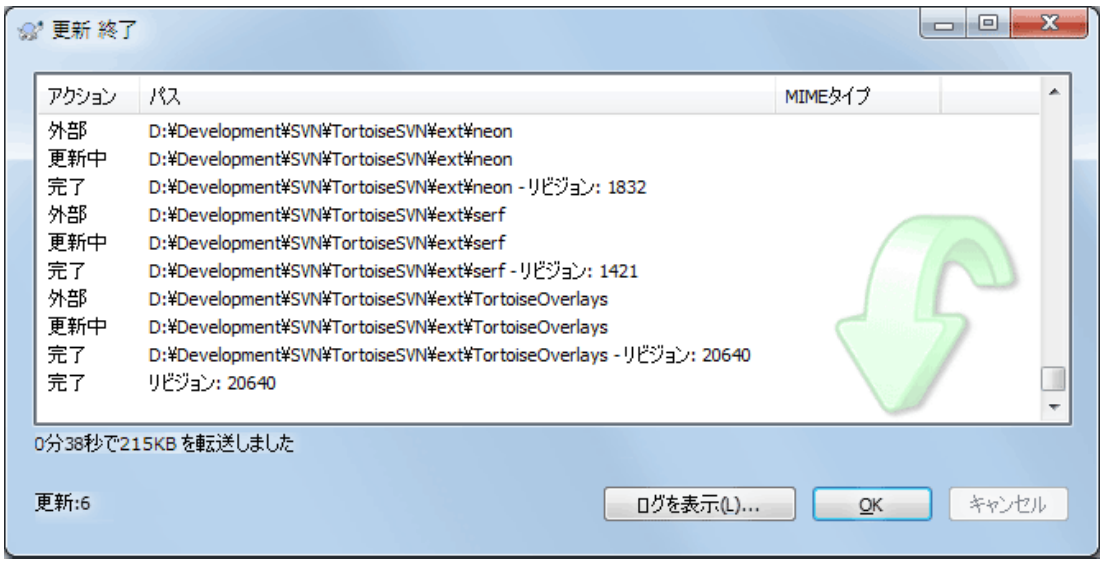


図4.11 更新が完了したときの進行ダイアログ

定期的に、他の人が行った変更を作業コピーに取り込まなければなりません。サーバーから作業コピーに変更を取り込む作業を、更新と言います。更新は単一のファイルに対して行うことも、複数の選択したファイルに対して行うことも、ディレクトリ構造に対して再帰的に行うこともできます。更新するには、更新したいファイルやディレクトリを選択し、右クリックで表示されるエクスプローラーのコンテキストメニューから、TortoiseSVN → 更新 を実行してください。更新の進行を表すウィンドウがポップアップし、更新が始まります。他の人が行った変更が、作業コピーにマージされますが、作業コピーで行った変更はすべて保持されます。更新では、リポジトリに影響を与えません。

進行ダイアログは更新の内容を色分けして、目立つようになっています。

紫色
　作業コピーに新しい項目が追加された。

暗赤色
　作業コピーから余分な項目が削除された、作業コピーの紛失項目が置き換えられた。

緑色
　リポジトリからの変更がローカルの変更に正常にマージできた。

明赤色
　リポジトリからの変更をローカルの変更にマージした結果、競合が発生して解決が必要になった。

黒
　作業コピー内の変更のない項目をリポジトリの新しいバージョンに更新した。

これはデフォルトの色設定ですが、設定ダイアログで色をカスタマイズできます。詳細は 「TortoiseSVN の色の設定」をご覧ください。

更新中に競合（同じファイルの同じ行を同時に変更して、それが一致しなかった場合に発生する）が発生したら、ダイアログでは競合を赤で表します。その行をダブルクリックして、外部マージツールを起動し競合を解決できます。

更新が完了すると、項目の更新・追加・削除・競合などの概略を進行ダイアログに表示します。概略情報は Ctrl+C でクリップボードにコピーできます。

標準の更新コマンドは、ただ作業ディレクトリをリポジトリの最新の HEAD リビジョンに更新します。指定するオプションは何もなく、ほとんどの場面ではこれで充分です。更新作業でこれ以上の制御をしたい場合は、代わりに TortoiseSVN → **特定リビジョンへ更新...** を使用してください。このコマンドでは、最新のものだけでなく、特定のリビジョンに作業コピーを更新することができます。作業コピーがリビジョン100になっていて、それをリビジョン50の状態に戻したいのであれば、リビジョン50に更新するだけです。

同じダイアログで、現在のフォルダーを更新するときの **深さ** を選択することもできます。選択肢の意味については、「チェックアウトの深さ」をご覧ください。デフォルトでは **作業コピー** が選択されており、これは現在の深さを維持します。また、**深さを記憶する**を選択することもできます。これを選択すると、以降の更新ではそれが新しい深さの初期値として使用されます。

項目を選択... ボタンをクリックすると、チェックアウト時に特定の項目を簡単に含めたり除外したりすることができます。新しいダイアログが開かれるので、作業コピーに入れたいすべての項目をチェックして、それ以外の項目のチェックを外してください。

更新の時には、特定の外部プロジェクトを無視するかどうかを選択できます（つまり、プロジェクトは `svn:externals` を使用して参照されます）。

注意

リビジョンを指定してファイルやフォルダーを更新した場合、そのファイルを変更するべきではありません。これをコミットしようとすると、「作業コピーは最新ではありません」のエラーが発生します。ファイルへの変更を取り消し、直前のリビジョンから新しくはじめたい場合、リビジョンログダイアログから以前のリビジョンに戻してしてください。詳しい方法や代替案については、「リポジトリのリビジョンのロールバック（取り消し）」をご覧ください。

特定リビジョンへ更新 は、履歴中の以前の時点ではどうだったかを見るのに時々は便利です。ですが一般的に、単独のファイルを以前のリビジョンに更新するのは、作業コピーを矛盾した状態にしかねないので、よい方法とは言えません。もし名前が変更されたことがあるファイルを更新すると、古いリビジョンではその名前では存在していなかったため、作業コピーから消えてしまい、探さなければならなくなるかもしれません。また、こうしたファイルには通常の緑のオーバーレイが表示されるので、最新のファイルと区別がつかないことにも注意してください。

単にそのファイルの古いリビジョンのローカルコピーが欲しければ、ログダイアログで選んだファイルに対して、**コンテキストメニュー → リビジョンを保存...** コマンドを使用する方がよいでしょう。

複数のファイルやフォルダー

エクスプローラーで複数のファイルやフォルダーを選択し、**更新** を選択した場合、そのファイルやフォルダーをひとつずつ更新していきます。TortoiseSVN は同じリポジトリから取得したファイルやフォルダーは、全く同じリビジョンに確実に更新します。この更新中に他の人からのコミットが発生してもです。

4.6. 競合の解決

リポジトリから自分のファイルを更新・マージしたり、別の URL に作業コピーを切り替えたりすると、時には 競合 することもあります。競合には以下の二種類があります。

ファイルの競合
: 複数の開発者が、同じファイルの同じ行を変更すると、ファイルの競合が発生します。

ツリーの競合
: 開発者がファイルやフォルダーの移動・名前変更・削除を行い、別の開発者も移動・名前変更・削除を行ったり、単に変更すると、ツリーの競合が発生します。

4.6.1. ファイルの競合

ファイルの競合は、複数の開発者があるファイルの同じ行を変更した際に発生します。Subversion はプロジェクトの事情を何も知らないため、開発者間の競合については解決できません。テキストファイルの場合、競合している箇所は以下のようなマークがつけられます。

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

また、競合しているファイルについては、3つのファイルが Subversion によって追加されます。

filename.ext.mine
: 作業コピーを更新する前に、作業コピーに存在していた（競合マーカーがついていない）ファイルです。このファイルには最新の変更のみが含まれています。

filename.ext.rOLDREV（OLDREV=旧リビジョン番号）
: 作業コピーを更新する前（BASE　リビジョン）のファイルです。最新の編集を行う前にチェックアウトした時のファイルです。

filename.ext.rNEWREV（NEWREV=新リビジョン番号）
: 作業コピーを更新したときに Subversion クライアントが受信したファイルです。リポジトリの最新（HEAD リビジョン）に相当します。

TortoiseSVN → 競合の編集 で外部マージツールや競合エディターを起動するか、テキストエディターを使用して手作業で競合を解決するかしてください。コードがどうあるべきか決定し、必要な変更をしてから保存してください。
TortoiseMerge やその他の有名なマージツールを使用した方が、3画面表示でこれらのファイルを確認でき、競合マーカーを気にする必要がないため、より簡単でしょう。テキストエディターを使用する場合は、<<<<<<< という文字列で始まる行を検索してください。

その後、**TortoiseSVN → 競合の解決** コマンドを実行し、リポジトリに変更をコミットしてください。なお、競合の解決コマンドは、実際に競合を解決する訳ではありません。変更をコミットできるよう、`filename.ext.mine` や `filename.ext.r*` ファイルを削除するだけです。

バイナリファイルが競合した場合、 Subversion はそのファイルをマージしようとしません。ローカルファイルには変更が加えられず（自分が変更したままで）、 `filename.ext.r*` ファイルを作られます。自分の変更を取り消し、リポジトリのバージョンにしたい場合は、「変更の取り消し」コマンドを使ってください。リポジトリのバージョンを自分の変更したファイルで置き換えるのなら、「競合の解決」コマンドを実行してからコミットしてください。

親フォルダーを右クリックし、 TortoiseSVN → 競合の解決... を実行すると、「競合の解決」コマンドを複数のファイルに実行できます。指定したフォルダーにある競合したファイルがすべてダイアログに表示されます。解決するものを選択してください。

4.6.2. プロパティの競合

複数の開発者が同じプロパティを変更した場合、プロパティの競合が発生します。ファイルの内容と同様に、競合は開発者のみが解決できます。

一方の変更を取り消してもう一方で上書きする場合は、 ローカルのプロパティを用いて解決する か リモートのプロパティを用いて解決する を選択してください。変更をマージする必要がある場合は、手作業でプロパティを編集 を選択し、どのプロパティをどうするかをマークして解決してください。

4.6.3. ツリーの競合

開発者がファイルやフォルダーを移動・名前変更・削除したときに、別の開発者が移動・名前変更・削除を行っていたり、変更をしていたりすると、ツリーの競合が発生します。ツリーの競合はさまざまな状況で起こり得ますし、その競合を解決するにはそれぞれ異なった手順が必要です。

また、ファイルが Subversion によりローカルで削除されると、ローカルファイルシステムからも削除されるため、ツリーの競合があったとしても競合のオーバーレイを表示できず、競合を解決しようにも右クリックもできません。 競合の編集 オプションを用いる代わりに、 変更をチェック ダイアログを使用してください。

TortoiseSVN は変更をマージするための正しい場所を見つけるのを支援することができますが、競合を整理するにはさらに作業が必要かもしれません。作業ベースを更新すると、常に更新時のリポジトリにある各項目のリビジョンが含まれます。更新後に変更を取り消した場合、そのリポジトリの状態へと戻り、変更開始時の状態には戻りません。

4.6.3.1. ローカルで削除、更新により編集を受信

1. 開発者Aが Foo.c を変更し、リポジトリにコミットする。

2. 開発者Bは同時に、作業コピーで Foo.c を Bar.c に移動したか、単に Foo.c またはその親フォルダーを削除した。

開発者Bの作業コピーを更新した結果、次のようにツリーの競合が発生します。

- Foo.c が作業コピーからすでに削除されていますが、ツリーの競合としてマークされます。
- 削除ではなく名前変更の結果が競合した場合、 Bar.c は追加としてマークされますが、開発者Aの変更点は含まれていません。

開発者Bは、現在、開発者Aの変更を保持するかどうかを選ばなければなりません。ファイルの名前変更の場合では、名前変更されたファイル Bar.c に対して、 Foo.c に行った変更をマージできます。単なるファイルやディレクトリの削除の場合には、開発者Aの変更を保持し、削除を取り消すという選択もできます。もしくは、何もせずに競合を解決状態にし、事実上開発者Aの変更を破棄することもできます。

名前が変更された Bar.c のオリジナルファイルが見つかれば、競合の編集ダイアログは変更をマージするか訊いてきます。更新を実行した場所によっては、ソースファイルが見つからない可能性もあります。

4.6.3.2. ローカルで編集、更新により削除を受信

1. 開発者Aが Foo.c を Bar.c に移動し、リポジトリにコミットします。

2. 開発者Bは Foo.c を、自分の作業コピーで変更します。

またはフォルダーの移動の場合...

1. 開発者Aは、親フォルダー FooFolder を BarFolder に移動し、リポジトリへコミットします。

2. 開発者Bは Foo.c を、自分の作業コピーで変更します。

開発者Bの作業コピーを更新した結果、ツリーが競合しました。単純なファイルの競合では次のようになります。

- Bar.c は作業コピーに通常ファイルとして追加されます。
- Foo.c は(履歴と共に)追加されたとマークされ、ツリーが競合状態になります。

フォルダーの競合では次のようになります。

- BarFolder は作業コピーに、通常のフォルダーとして追加されます。
- FooFolder は(履歴と共に)追加されたとマークされ、ツリーが競合状態になります。

  Foo.c は変更されたとマークされます。

開発者Bは今回、開発者Aの再配置を受け入れ、変更を移動先のファイルにマージするか、開発者Aの変更を取り消し、ローカルのファイルを保持するかを決めねばなりません。

再配置された結果にローカルの変更をマージするためには、開発者Bはまず、競合したファイル Foo.c がリポジトリ中でどのようなファイル名に名前変更・移動されたのかを探さなければなりません。ログダイアログを使用すると調べることができます。その後、その変更を手でマージしなければなりません。現在のところ、この手順を自動化したり、単純化したりする方法がないからです。いったん変更を移植してしまえば、競合したパスは余分なものとなり、削除できます。競合の編集ダイアログの 削除 ボタンを押せば、不要なファイルを削除して競合を解決済みにできます。

開発者Aの変更は誤りだと開発者Bが判断した場合は、競合の編集ダイアログの 保持 ボタンを選択します。これは競合したファイルやフォルダーを、解決済みとしてマークしますが、開発者Aの変更は手作業で削除する必要があります。何が移動されたかを見つけ出すには、またログダイアログが役に立ちます。

4.6.3.3. ローカルで削除、更新により削除を受信

1. 開発者Aが Foo.c を Bar.c に移動し、リポジトリにコミットします。

2. 開発者Bが Foo.c を Bix.c に移動します。

開発者Bの作業コピーを更新した結果、次のようにツリーの競合が発生します。

- Bix.c は、履歴と共に、追加されたとマークされます。
- Bar.c は、作業コピーへ「通常」の状態で追加されます。
- Foo.c は、削除されたとマークされ、ツリーが競合状態になります。

競合を解決するには、 Foo.c がリポジトリで名前変更・移動されたことに対する競合したファイルの名前を、開発者Bが調べなければなりません。これはログダイアログで行えます。

開発者Bが、Foo.c の新しい名前の方を残すと決めた場合、開発者Aにしてもらうか、自分で名前変更できます。

開発者Bが、手作業で競合を解決した後で、競合の編集ダイアログのボタンで、ツリーの競合を解決済みにする必要があります。

4.6.3.4. ローカルで紛失、マージにより編集を受信

1. トランクで作業している開発者Aが、Foo.c を変更しリポジトリにコミットします。

2. ブランチで作業している開発者Bが、Foo.c を Bar.c に移動し、リポジトリにコミットします。

開発者Aのトランクへの変更を、開発者Bのブランチにマージすると、作業コピーは以下のようにツリーの競合状態になります。

- Bar.c は、状態が「通常」で、すでに作業コピーにあります。
- Foo.c は、紛失マークがつき、ツリーの競合状態になります。

この競合を解決するには、開発者Bが競合の解決ダイアログでファイルに解決済みにし、競合リストから取り除く必要があります。さらに、紛失ファイル Foo.c を、リポジトリから作業コピーへコピーするか、Foo.c へ行った開発者Aの変更を、名前変更した Bar.c にマージするか、競合に解決マークを付ける他は何もしないかを決めなければなりません。

紛失したファイルをリポジトリから取得し、それから解決済みとすると、自分のコピーが再度削除されます。まず競合を解決しなければなりません。

4.6.3.5. ローカルで編集、マージにより削除を受信

1. トランクで作業している開発者Aが、Foo.c を Bar.c に移動し、リポジトリにコミットします。

2. ブランチで作業している開発者Bが、Foo.c を変更しリポジトリにコミットします。

フォルダーの移動と同等ですが、Subversion 1.6 でもまだ検出できません……

1. トランクで作業している開発者Aが、親フォルダー FooFolder を BarFolder に移動し、リポジトリにコミットします。

2. ブランチで作業している開発者Bが、自分の作業コピーにある Foo.c を変更します。

開発者Aのトランクへの変更を、開発者Bのブランチにマージすると、作業コピーは以下のようにツリーの競合状態になります。

- Bar.c には追加マークが付きます。
- Foo.c はツリーの競合として、変更マークが付きます。

開発者Bは今回、開発者Aの再配置を受け入れ、変更を移動先のファイルにマージするか、開発者Aの変更を取り消し、ローカルのファイルを保持するかを決めねばなりません。

再配置された結果にローカルの変更をマージするためには、開発者Bはまず、競合したファイル Foo.c がリポジトリ中でどのようなファイル名に名前変更・移動されたのかを探さなければなりません。マージ元のログダイアログを使用すると調べることができます。競合エディターは、どのパスがマージで使用されたのかを知らず、作業コピーのログを表示するだけですので、自分で探さなければなりません。現在のところ、この手順を自動化したり、単純化したりする方法がないからです。いったん変更を移植してしまえば、競合したパスは余分なものとなり、削除できます。競合の編集ダイアログの 削除 ボタンを押せば、不要なファイルを削除して競合を解決済みにできます。

開発者Bが、開発者Aの変更は誤りだと判断した場合、競合の編集ダイアログの 保持 ボタンを選択しなければなりません。これは 競合したファイルやフォルダーを、解決済みとしてマークしますが、開発者Aの変更は手で削除する必要があります。何が移動されたかを見つけ出すには、またマージソースのログダイアログが役に立ちます。

4.6.3.6. ローカルで削除、マージにより削除を受信

1. トランクで作業している開発者Aが、Foo.c を Bar.c に移動し、リポジトリにコミットします。

2. ブランチで作業している開発者Bが、Foo.c を Bix.c に移動し、リポジトリにコミットします。

開発者Aのトランクへの変更を、開発者Bのブランチにマージすると、作業コピーは以下のようにツリーの競合状態になります。

- Bix.c は通常(未変更)状態としてマークされます。
- Bar.c は履歴と共に追加マークが付きます。
- Foo.c には紛失マークが付き、ツリーの競合状態となります。

競合を解決するために、開発者Bは競合ファイル Foo.c がリポジトリ中で名前変更・削除されたファイル名を探さなければなりません。マージソースのログダイアログを使用するとわかります。競合の解決ダイアログは、マージで使用されるパスを知らず、作業コピーのログを表示するだけですので、自分で探さなければなりません。

開発者Bが、Foo.c の新しい名前の方を残すと決めた場合、開発者Aにしてもらうか、自分で名前変更できます。

開発者Bが、手作業で競合を解決した後で、競合の編集ダイアログのボタンで、ツリーの競合を解決済みにする必要があります。

4.6.3.7. 他ツリーの競合

他にも、競合にファイルではなくフォルダーが含まれるために、単純にツリーの競合としてマークされている場合があります。例えば、トランクとブランチの両方に同じ名前のフォルダーを追加してからマージしようとした場合には、ツリーの競合と表示されます。マージ対象のフォルダーを維持したい場合は、単に解決済みとしてください。マージ元のどちらか一方を使用する場合は、最初のターゲットの1つに対してSVN 削除を実行し、再びマージを実行する必要があります。もっと複雑な場合は、手作業で解決する必要があります。

4.7. ステータス情報の取得

作業コピーで作業をしていると、ファイルに対して変更・追加・削除や名前の変更を行ったのかどうか、また他の人が変更してコミットしたファイルがあるかどうか知る必要があるでしょう。

4.7.1. アイコンオーバーレイ

図4.12 エクスプローラーのアイコンオーバーレイ表示

Subversion のリポジトリから作業コピーにチェックアウトすると、Windows エクスプローラー上でファイルのアイコンが変化しているのが分かります。これこそ TortoiseSVN を有名にした理由のひとつです。TortoiseSVN はオーバーレイアイコンと呼ばれるアイコンを、元のファイルアイコンの上に重ねて表示します。ファイルの Subversion における状態によってオーバーレイアイコンが変わります。

作業コピーにチェックアウトしたばかりのファイルには緑のチェックマークをオーバーレイします。Subversion の状態は 通常 です。

ファイルの編集をはじめるとすぐに状態が 変更 に変わり、アイコンオーバレイも赤いエクスクラメーションマークに変わります。これで最後に作業コピーを更新してからどのファイルに変更を加え、コミットする必要があるかどうか分かります。

更新中に 競合 が発生したら、黄色いエクスクラメーションマークのアイコンに変わります。

ファイルに `svn:needs-lock` プロパティを設定していると、Subversion はそのファイルにロックをかけるまで読み取り専用にします。そのようなファイルには、編集前にロックするまで、このオーバーレイが表示されます。

ファイルをロックしており、かつ Subversion の状態が 通常 であれば、もう他人にファイルの変更を許可しても良いのにロックを解除し忘れないように、このアイコンが表示されます。

バージョン管理から 削除 されるカレントフォルダー内のファイルやフォルダー、またバージョン管理が紛失したフォルダー内にあるファイルにこのアイコンがつけられます。

バージョン管理に 追加 されるファイルやフォルダーには+印がつきます。

横棒印は、ファイルやフォルダーが、バージョン管理的に 無視 されていることを示します。このオーバーレイはオプションです。

バージョン管理下にないファイルやフォルダーで、無視されていない場合には、このアイコンが表示されます。このオーバーレイはオプションです。

実際のところ、上記のアイコンのすべてがシステムで使用されている訳ではないことが分かると思います。Windows で使用できるオーバーレイの数はかなり制限されていますし、また TortoiseCVS の古いバージョンを一緒に使用していると、使用できるオーバーレイスロットが不足します。TortoiseSVN は「Good Citizen #」（良い市民）になれるように、他のアプリケーションにも使用する機会を与えるように、オーバーレイの使用を制限しています。

Tortoise クライアントが今より増えた場合（TortoiseCVS、TortoiseHg、...）、アイコンの限界は現実的な問題になります。この問題を回避するために、TortoiseSVN プロジェクトではすべての Tortoise クライアントで共通に使用できる、DLL で読み込むアイコンセットを導入しました。すでに統合されているかどうかを確認するには、クライアントの供給元に確認してください。:-)

Subversion の状態に対応するアイコンオーバーレイの付き方や、その他の技術詳細は、「アイコンオーバーレイ」をご覧ください。

4.7.2. 詳細なステータス

図4.13 エクスプローラーのプロパティページの Subversion タブ

オーバーレイアイコンより詳しい、ファイルやディレクトリに関する詳細情報を確認したい場合、エクスプローラーのプロパティダイアログから、Subversion が提供するすべての情報を確認することができます。ファイルやディレクトリを選択し、コンテキストメニューの Windows メニュー → プロパティ を選択するだけです(TortoiseSVN のサブメニュー内のものではなく、エクスプローラーのプロパティであることに注意してください)。 Subversion の管理下のファイルやフォルダーでプロパティを表示すると、TortoiseSVN 独自のプロパティページが表示されます。ここで、選択したファイルやフォルダーに関するすべての情報を見ることができます。

4.7.3. Windows エクスプローラーの TortoiseSVN 列

Windows エクスプローラーの詳細ビューに、アイコンオーバーレイと同様(またそれ以上)の情報を表示する列を表示することができます。

列の見出しを右クリックして出てくるコンテキストメニューで、その他... を選択してください。「詳細表示」の列とその順番を設定するダイアログが表示されます。 SVN で始まる項目が出てくるまでスクロールしてください。表示したい項目にチェックをつけ、OK を押してダイアログを閉じてください。元々表示していた項目の右に列が追加されているはずです。お好みで、ドラッグ&ドロップで順番を変えたり大きさを変えたりしてください。

重要

Vista では、Windows エクスプローラーに列を追加することができません。特定のファイル形式ではなく 全ての ファイルにそのような列を追加することを Microsoft が許さなくなったためです。

ヒント

このレイアウトを作業コピー全体に適用するなら、これをデフォルトビューにするといいでしょう。

4.7.4. ローカルとリモートの状態

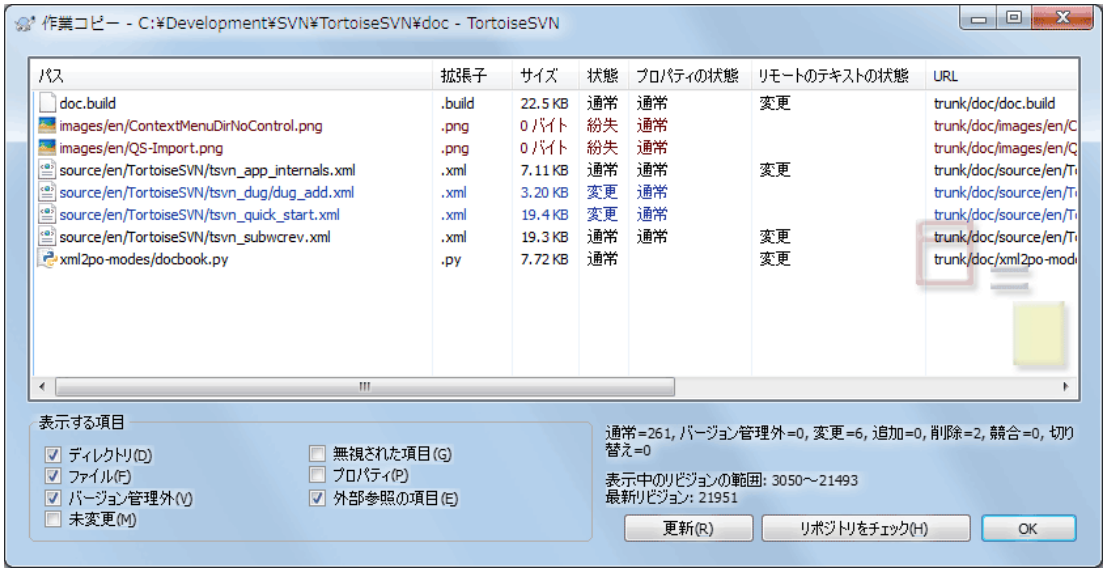


図4.14 変更をチェック

自分でどのファイルを変更したか、また他の人がどのファイルを変更しコミットしたかを知ることは非常に有用です。これには TortoiseSVN → **変更をチェック...** コマンドが役に立ちます。このダイアログは作業コピー内にある何らかの変更をしたファイルや、バージョン管理外のファイルをすべて表示します。

リポジトリをチェック をクリックすると、リポジトリで行われた変更を参照できます。競合が起こりそうな場合、更新前にこれで変更点を確認できます。リポジトリの全てのフォルダーを更新しないで、選択したファイルのみ更新することもできます。デフォルトでは、 **リポジトリをチェック** ボタンは、リモートの状態を作業コピーにチェックアウトした深さまで検索します。リポジトリ内のチェックアウトしていないファイルやフォルダーまで確認したい場合は、Shift キーを押しながら **リポジトリをチェック** ボタンをクリックしてください。

このダイアログでは、状態が色分けして表示されます。

青色
: ローカルで変更された項目です。

紫色
: 追加された項目です。履歴から追加された項目には **テキストの状態** 列に + 印がつきます。また、項目のコピー元がツールチップに表示されます。

暗赤色
: 削除された・紛失した項目です。

緑色
: ローカル及びリポジトリ内で変更された項目。更新時にマージされます。更新時に競合する かも しれません。

明赤色
: ローカルで変更されリポジトリでは削除された、ないしリポジトリで変更されローカルで削除された項目。更新時に競合が 発生します 。

黒
: 変更がなくバージョン管理外の項目です。

これはデフォルトの色設定ですが、設定ダイアログで色をカスタマイズできます。詳細は 「TortoiseSVN の色の設定」をご覧ください。

異なるリポジトリのパスに切り替えた項目には、(s)印が示されます。ブランチで何かの作業を行った後で、トランクに切り替え忘れることがあるかもしれません。これが警告のサインです。コンテキストメニューを使用すれば、通常のパスに切り替えることができます。

このダイアログのコンテキストメニューから、変更点の差分を表示することができます。 自分 が行ったローカルの変更をチェックするには、**コンテキストメニュー → 作業ベースと比較** を使用してください。他人が行ったリポジトリへの変更は、**コンテキストメニュー → Unified形式で差分を表示** を使用してください。

個々のファイルの変更を取り消すこともできます。誤ってファイルを消してしまった場合には、紛失 と表示されますし、変更の取り消し を実行すればファイルを回復することができます。

バージョン管理外のファイルや無視されたファイルは、**コンテキストメニュー → 削除** でごみ箱に送ることができます。完全に(ごみ箱を経由させずに)ファイルを削除するには、Shift キーを押したまま **削除** をクリックしてください。

ファイルの中を確認したければ、テキストエディターや IDE のような別のアプリケーションにファイルをドラッグするか、エクスプローラーのフォルダーにドラッグするかすれば、コピーを保存することができます。

リストの列はカスタマイズできます。列見出しの上で右クリックすると、表示する列を選択するコンテキストメニューが表示されます。また、列の境界上にマウスを持っていくとドラッグハンドルが表示され、列幅を変更できます。カスタマイズ結果は保存されるので、次回以降も同じ列見出しになります。

関連のない複数の仕事を一度に行った場合、変更リストにファイルをグループ化することもできます。詳しくは「変更リスト」をご覧ください。

ダイアログの下部に、作業コピーで使われている、リポジトリリビジョンの範囲の概要を表示しています。これは 更新 リビジョンではなく コミット リビジョンであり、ファイルが更新されたリビジョンではなく、ファイルが最後にコミットされた時のリビジョン範囲を表しています。作業コピー全体ではなく、表示されている項目のみのリビジョン範囲であることに注意してください。作業コピー全体に対してこの情報を表示する場合は、 **表示する項目** の **未変更** チェックボックスにチェックを入れてください。

ヒント

作業コピーに含まれるすべてのファイルやフォルダーを一度に表示したいような場合は、 **変更をチェック** ダイアログが一番簡単です。**表示する項目** の **未変更** チェックボックスにチェックをつけると、作業コピー内のすべてのファイルが表示されます。

外部での名前変更の修復

Subversion の外部でファイルの名前が変更された場合、ファイル一覧で紛失ファイルとして表示されたり、バージョン管理外ファイルとして表示されたりします。履歴を失わないように Subversion

に関連を通知する必要があります。古い名前(紛失)と新しい名前(バージョン管理外)を選択し、 **コンテキストメニュー → 移動を修復** を実行すれば、名前の変更が行われたことを示すことができます。

外部でのコピーの修復

ファイルのコピーをしたのに、Subversion コマンドで行うのを忘れた場合、新しいファイルが履歴を失わないように、そのコピーを修復できます。古い名前(通常ないし変更)と新しい名前(バージョン管理外)を選択し、**コンテキストメニュー → コピーの修復** を実行するだけで、ファイルのコピーが行われたことを示すことができます。

4.7.5. 差分の表示

ファイルの中でどのような変更をしたのか確認する場合は、変更のあるファイルを選択して、TortoiseSVN のコンテキストメニューから 差分を表示 を選択してください。外部差分ビューアーが起動し、現在のファイルとチェックアウト・更新を行った直後のコピー(作業ベース、BASE リビジョン)とを比較できます。

ヒント

作業コピー内にない場合や、複数のバージョンがある場合でも、差分を表示できます。

エクスプローラーで比較したい2つのファイルを選択(例えば Ctrl を押しながらクリック)し、TortoiseSVN のコンテキストメニューから 差分を表示 を選択してください。最後にクリックしたファイル(フォーカスのある、つまり点線の四角で囲まれているもの)を最新として扱います。

4.8. 変更リスト

常に一度にひとつの作業しか行わず、作業コピーには論理的にひとかたまりの変更しか存在しなければ理想的でしょう。しかし現実では、関連のない複数の作業を同時に行わなければならない場合がよくあります。そしてその場合、コミットダイアログには、変更したファイルがすべて混ざった状態で表示されてしまいます。 変更リスト 機能は、ファイルをグループにまとめ、どうするべきかを判りやすくしてくれます。もちろん、変更したファイルが重複していない場合に限ります。異なる仕事で同じファイルを変更した場合、変更を分割する方法はありません。

変更リストは様々な場所で目にすることができますが、最も重要なのは、コミットダイアログと変更のチェックダイアログです。いくつかの機能とたくさんのファイルに対して作業を行い、その後で変更のチェックダイアログを開きましょう。初めて変更のチェックダイアログを開くと、変更のあるファイルがすべて表示されます。機能単位に変更リストを作成し、ファイルをグループ化するとしておくとよいでしょう。

変更リストに項目を追加するには、ファイルを選択して **コンテキストメニュー → 変更リストへ移動** を使用してください。初期状態では変更リストがありませんので、最初に新しい変更リストを作成します。変更リストに何のために使用するのか分かる名前を付けて、OK をクリックしてください。ダイアログが項目のグループを表示するように変化します。

いったん変更リストを作成すると、別の変更リストや Windows エクスプローラーからも、項目をドラッグ&ドロップできます。エクスプローラーからドラッグすれば、ファイルを変更する前に変更リストに追加できるので便利です。変更のチェックダイアログからも同じことができますが、未変更ファイルすべてを表示しなければなりません。

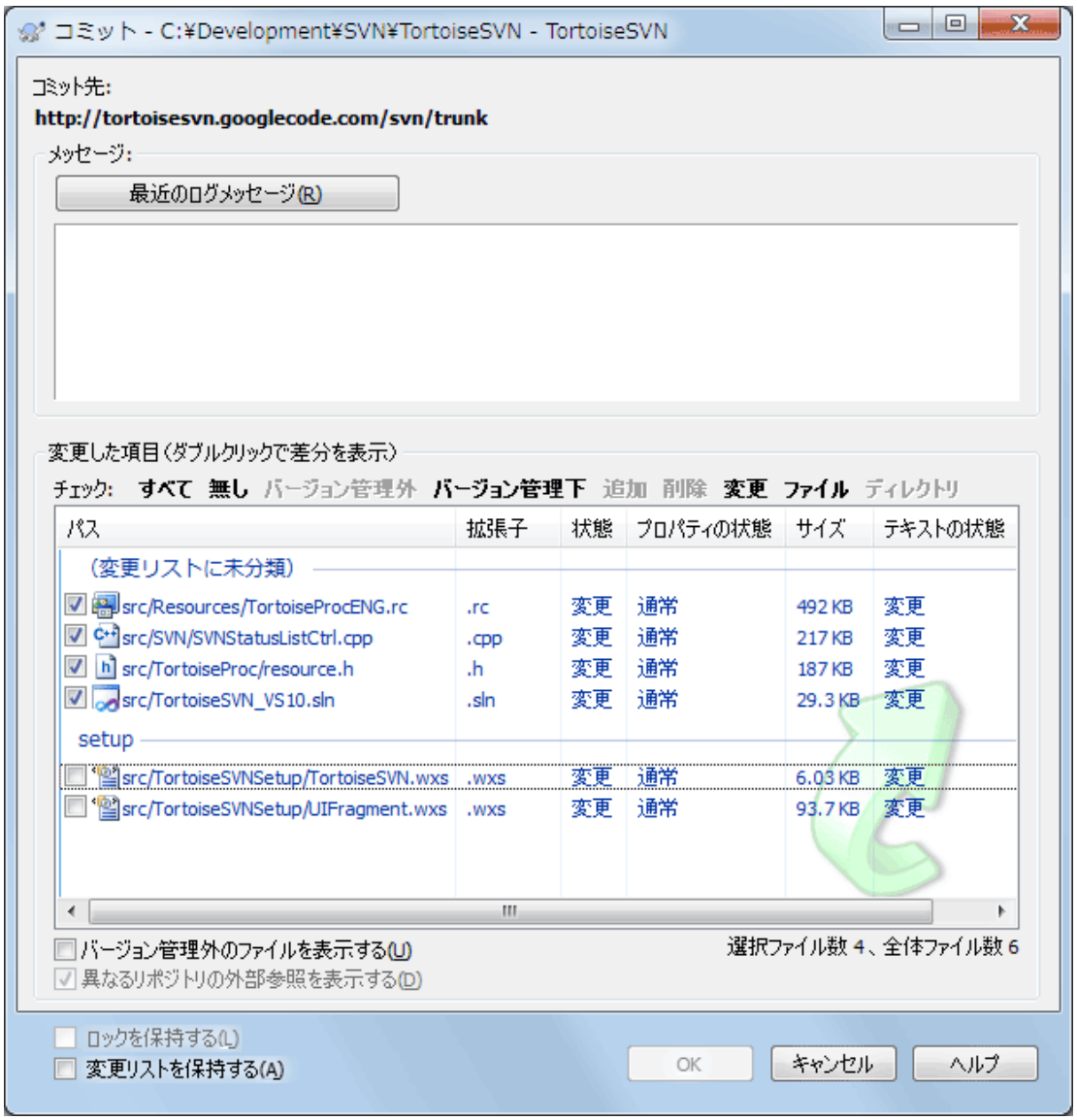


図4.15 変更リストがあるコミットダイアログ

コミットダイアログでは、同じファイルが変更リストでグループ化されて表示されます。グループを視覚的に表示する機能とは別に、グループの見出しをコミットするファイルを選択するために使用することもできます。

XP では、グループ見出しを右クリックすると、グループ全体を選択・非選択にするコンテキストメニューが現れます。Vista ではコンテキストメニューは必要ありません。全項目を選択するにはグループヘッダを選択し、その後すべてチェックする場合には、選択している項目のどれかをチェックしてください。

TortoiseSVN は `ignore-on-commit` という名前の変更リストを、自身で使用するために予約しています。これは、ローカルで変更があってもコミットしたくない、バージョン管理下のファイルをマークするのに使用します。この機能は、「コミット一覧からの項目の除外」で説明します。

通常、変更リストに属するファイルをコミットすれば、変更リストに所属させておく必要はなくなると考えられます。そのためデフォルトでは、コミットしたファイルは、変更リストから除外されます。変更リストに入れたままにしておきたい場合は、コミットダイアログの下部にある **変更リストを保持する** をチェックしてください。

ヒント

変更リストは純粋にローカルクライアントの機能です。変更リストを作成・削除しても、リポジトリや他人の作業コピーには影響しません。単に自分でファイルを管理する、便利な方法でしかありません。

4.9. リビジョンログダイアログ

変更してコミットするたびに、変更点ついてのログメッセージを残しておきましょう。これを見れば、後からどんな変更をなぜ行ったのかがわかりますし、開発プロセスの詳細なログにもなります。

リビジョンログダイアログでは、すべてのログメッセージが取得され表示されます。画面は3つの部分に分かれています。

- 上部のリストには、ファイルやフォルダーの変更がコミットされたリビジョンの一覧が表示されます。この欄にはコミット日時、そのリビジョンをコミットした人、ログメッセージの冒頭などが表示されます。

  青く表示されている行は、何かがこの開発ラインに(おそらくブランチから)コピーされてきたことを示しています。

- 中央の欄には、選択したリビジョンのログメッセージ全体が表示されます。
- 下部のリストには、選択したリビジョンで変更したファイルやフォルダーの一覧が表示されます。

他にも、コンテキストメニューのコマンドを使用すれば、プロジェクトの履歴についての追加情報を取得できます。

4.9.1. リビジョンログダイアログの起動

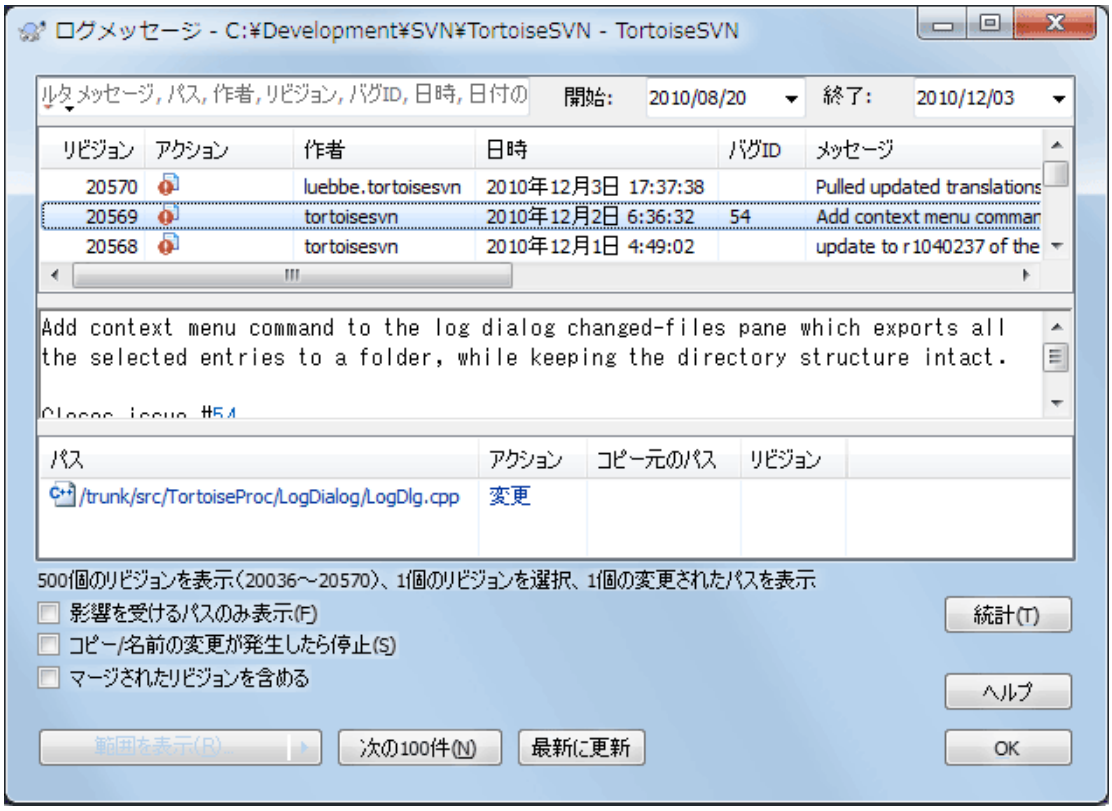


図4.16 リビジョンログダイアログ

ログダイアログは、様々な場所から表示することができます。

- TortoiseSVN のコンテキストサブメニューから

- プロパティページから
- 更新が終わった進行ダイアログから。前回の更新から変更のあったリビジョンのみ、ログダイアログに表示されます。

リポジトリが使用できない場合、 **オフラインにしますか？** というダイアログが表示されます。詳しくは、「オフラインモード」で説明します。

4.9.2. リビジョンログのアクション

上のリストの **アクション** 列には、そのリビジョンで何が行われたのかを示すアイコンが表示されます。4種類のアイコンが、それぞれ1列ずつ表示されます。

そのリビジョンでファイルやディレクトリが変更された場合、変更 アイコンが1列目に表示されます。

そのリビジョンでファイルやディレクトリが追加された場合、追加 アイコンが2列目に表示されます。

そのリビジョンでファイルやフォルダーが削除された場合、削除 アイコンが3列目に表示されます。

そのリビジョンでファイルやフォルダーが置換された場合、置換 アイコンが4列目に表示されます。

4.9.3. 追加情報の取得

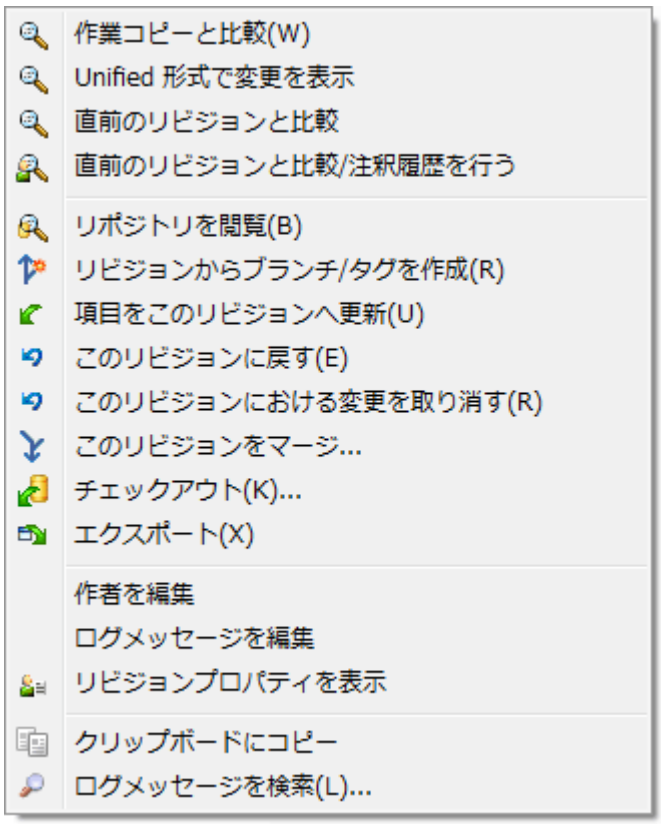


図4.17 リビジョンログダイアログの上部のリストのコンテキストメニュー

ログダイアログの上部のリストには、より詳細な情報にアクセスできるコンテキストメニューがあります。このメニューにはファイルのログでしか現れないものや、フォルダーのログでしか現れないものがあります。

作業コピーと比較
: 作業コピーと選択したリビジョンを比較します。デフォルトの差分ツールは、 TortoiseSVN と共に提供されている TortoiseMerge です。フォルダーに対するログダイアログの場合、変更のあったファイルが一覧表示され、個々のファイルに対して変更を確認することができます。

作業ベースと比較/注釈履歴
: 選択したリビジョンと作業ベースのファイルの注釈履歴を取得し、差分ツールを使用して比較します。詳しくは 「注釈履歴の差分」を参照してください。(ファイルのみ)

Unified形式で変更を表示
: 選択したリビジョンで行われた変更点を、Unified差分ファイル(GNUパッチ形式)で表示します。差分のみが数行で表示されます。視覚的なファイル比較よりも分かりにくいのですが、すべての変更をまとめてコンパクトな形式で確認できます。

  Shift キーを押しながらメニュー項目をクリックすると、最初にUnified差分ファイルのためのオプションを設定するためのダイアログが開きます。これらのオプションの中に、行末文字や空白文字の変更を無視するものが含まれています。

直前のリビジョンと比較
: 選択したリビジョンを直前のリビジョンと比較します。作業コピーとの比較と同じように動作します。フォルダーに対してこの操作を実行すると、まず比較するファイルを選択する「変更されたファイル」ダイアログが表示されます。

直前のリビジョンと比較/注釈履歴
: ファイルを選択する「変更されたファイル」ダイアログが表示されます。選択したリビジョンと直前のリビジョンの注釈履歴を取得し、差分ツールを使用して結果を比較します。(フォルダーのみ)

リビジョンを保存...
: 選択したリビジョンをファイルに保存し、そのファイルの古いバージョンを入手します。(ファイルのみ)

開く/プログラムを指定して開く...
: ファイル形式に対応するデフォルトのビューアー、もしくは選択したプログラムのどちらかを用いて、選択したファイルを開きます。(ファイルのみ)

注釈履歴...
: 選択したリビジョンまでのファイルの注釈履歴を表示します。(ファイルのみ)

リポジトリの閲覧
: 選択したファイルやフォルダーを閲覧するため、リポジトリブラウザーを選択したリビジョンで開きます。

リビジョンからブランチ/タグを作成
: 選択したリビジョンからブランチやタグを作成します。タグを作成し忘れて、リリースに入れることを前提としていない変更をすでにコミットしてしまったような場合に便利です。

項目を特定リビジョンへ更新
: 作業コピーを選択したリビジョンに更新します。時間をさかのぼり、作業コピーを過去の状態にしたい場合や、リポジトリに他のコミットが行われた後で、作業コピーを一度に更新したい場合に便利です。単体のファイルではなく、作業コピー全体のディレクトリを更新するとよいでしょう。そうでなければ作業コピーに矛盾が発生してしまいます。

  以前の変更点を永久に取り消したい場合は、代わりに **このリビジョンに戻す** を使用してください。

このリビジョンに戻す
: 以前のリビジョンに戻します。既にいくつか変更を行った後で、リビジョン N へ戻したい場合に使用するコマンドです。変更の取り消しは作業コピーで行われますので、変更をコミットするまではリポジトリに影響を 与えません 。選択

したリビジョン以降の変更が すべて 取り消され、ファイルやフォルダーが以前のリビジョンに置き換えられてしまうので注意してください。

作業コピーが未変更の状態の場合、この操作を実行すると、作業コピーが変更の状態になります。すでに変更されている場合、このコマンドは作業コピーの変更を 取り消す 方向にマージします。

内部的には、 Subversion は選択されたリビジョンの後に行われたすべての変更の逆マージを実行し、以前のコミットの効果を元に戻すという動作をします。

この操作の後で、 元に戻したものを元に戻し 、作業コピーを以前の変更されていない状態に戻す場合、Windows エクスプローラーから TortoiseSVN → **変更の取り消し** を実行し、逆マージ操作で行われたローカルの変更を取り消してください。

以前のリビジョンで、どんなファイルやフォルダーがあったのかを見たいだけであれば、代わりに **特定リビジョンへ更新** や **リビジョンを保存...** を使用してください。

このリビジョンにおける変更を取り消す

選択したリビジョンの変更を取り消します。作業コピーの変更が取り消されるので、この操作はリポジトリにまったく影響を 及ぼしません 。そのリビジョンで行われた変更のみが元に戻されることに注意してください。作業コピーのファイル全体が以前のリビジョンに置き換えられるわけではありません。関係ない変更を行った後で、以前の変更を元に戻すのに便利です。

作業コピーが未変更の状態の場合、この操作を実行すると、作業コピーが変更の状態になります。すでに変更されている場合、このコマンドは作業コピーの変更を 取り消す 方向にマージします。

内部的には、Subversion は選択したリビジョンで行われた変更の逆マージを実行し、以前のコミットの効果を元に戻すという動作をします。

このリビジョンに戻す で説明しているように、元に戻したものを元に戻す こともできます。

このリビジョンをマージ...

選択したリビジョンを異なる作業コピーにマージします。フォルダー選択ダイアログでマージ結果を格納する作業コピーを選択できますが、確認ダイアログはなく、またマージのテストもできません。変更されていない作業コピーを用い、選択したリビジョンをマージするのに失敗したら変更を取り消すとよいでしょう。これはあるブランチから別のブランチへ、選択したリビジョンをマージするのに便利な機能です。

チェックアウト...

選択したリビジョンの選択したフォルダーを新たにチェックアウトします。URL とリビジョンを確認するダイアログが現れるので、チェックアウトする場所を選択してください。

エクスポート...

選択したリビジョンの選択したファイルやフォルダーをエクスポートします。URL とリビジョンを確認するダイアログが現れるので、エクスポートする場所を選択してください。

作者・ログメッセージを編集

以前に行ったコミットのログメッセージや作者を編集します。どのように行うかは、「ログメッセージや作者の変更」をご覧ください。

リビジョンプロパティを表示

ログメッセージと作者に限らず、任意のリビジョンプロパティを表示・編集します。「ログメッセージや作者の変更」をご覧ください。

クリップボードにコピー
: 選択したリビジョンのログ詳細をクリップボードにコピーします。これにはリビジョン番号、著者、日時、ログメッセージ、変更した項目の一覧がコピーされます。

ログメッセージを検索...
: 入力したテキストでログメッセージを検索します。ログメッセージの入力された部分と、 Subversion が作成した動作の概要（下の欄に表示）も検索対象になります。大文字小文字は区別されません。

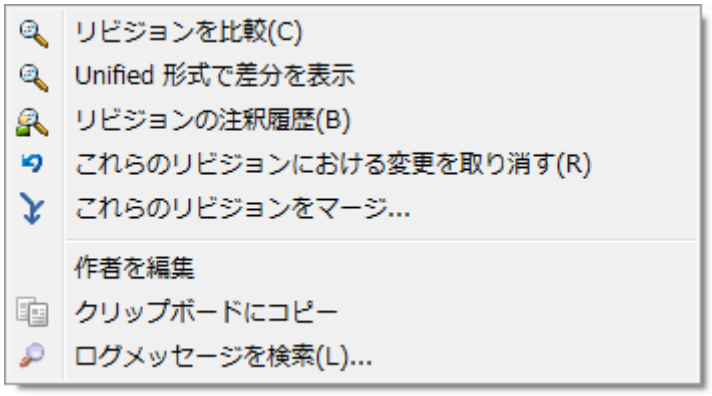


図4.18 2つのリビジョンを選択した時の上部のリストのコンテキストメニュー

同時に2つのリビジョンを（Ctrl を押しながら）選択すると、コンテキストメニューの内容が以下の機能に変わります。

リビジョンを比較
: 選択した2つのリビジョンを差分ツールで比較します。デフォルトの差分ツールは、TortoiseSVN に同梱されている TortoiseMerge です。

  フォルダーに対して実行した場合、変更のあったファイルを一覧表示するダイアログが表示され、差分を表示するオプションを指定できます。リビジョン比較ダイアログについては、「フォルダーの比較」をご覧ください。

リビジョンの注釈履歴
: 2つのリビジョンの注釈履歴をとり、差分ツールで比較できます。詳細は、「注釈履歴の差分」をご覧ください。

Unified形式で差分を表示
: 選択した2つのファイルの差分を、Unified差分ファイルで表示します。ファイルとフォルダーで動作します。

クリップボードにコピー
: 前述したように、ログメッセージをクリップボードにコピーします。

ログメッセージを検索...
: 前述したように、ログメッセージを検索します。

2つ以上のリビジョンを選択（通常 Ctrl や Shift を押しながら選択）すると、コンテキストメニューには「これらのリビジョンにおける変更を取り消す」項目が現れます。これがリビジョンのグループを変更を一度に取り消す、最も簡単な方法です。

また、前述のとおり、選択したリビジョンを別の作業コピーにマージできます。

選択したすべてのリビジョンが、同じ作者のものであれば、すべてのリビジョンの作者を一気に編集できます。

図4.19 ログダイアログの下のリストのコンテキストメニュー

ログダイアログの下の欄にもコンテキストメニューがあります。次の操作を行うことができます。

変更を表示
　選択されたファイルの選択されたリビジョンにおける変更を表示する。

変更の注釈履歴
　選択したファイルの、選択したリビジョンと直前のリビジョンの注釈をとり、視覚差分ツールで比較できます。詳細は、「注釈履歴の差分」をご覧ください。

Unified形式で変更を表示
　ファイルの変更点をUnified差分ファイルで表示します。このコンテキストメニューは、ファイルに　変更　がある時のみ有効です。

開く/プログラムを指定して開く...
　選択したファイルを、ファイル形式に応じた規定のビューアーか選択したプログラムで開きます。

注釈履歴...
　注釈履歴ダイアログを開き、選択したリビジョンの担当情報を参照できます。

このリビジョンにおける変更を取り消す
　選択したファイルのこのリビジョンでの変更を取り消します。

プロパティを表示
　選択した項目の Subversion のプロパティを表示します。

ログを表示
　単体のファイルを選択しているとき、ファイルのリビジョンログを表示します。

マージのログを取得する
　単体のファイルを選択しているとき、ファイルのマージされた変更を含めリビジョンログを表示します。詳細は 「マージ追跡機能」をご確認ください。

リビジョンを保存...
　選択したリビジョンをファイルに保存し、そのファイルの古いバージョンを入手します。

エクスポート...
　　このリビジョンの選択された項目を、ファイルの階層を保持したままエクスポートします。

ヒント

「変更」と「差分」という用語が使い分けられていることにお気づきと思います。では差分とはなんでしょうか?

Subversion では、リビジョン番号を2つの異なる意味で使用しています。一般的にリビジョンといえば、その時点でのリポジトリの状態を表しますが、一方ではリビジョンを作成した際の一連の変更を表す場合に使用することもあります。例えば、「r1234での変更」といえば、r1234の変更のコミットで実装された機能 X のことを表します。どちらの意味で使用されているかを明確にするために、2つの用語を使い分けます。

リビジョン N とリビジョン M を選択すると、その2つのリビジョンの 差分 をコンテキストメニューで表示できるようになります。Subversion のコマンドでは、これは `diff -r M:N` に相当します。

リビジョン N を選択した場合、そのリビジョンで行った 変更 をコンテキストメニューで表示できるようになります。Subversion のコマンドでは、これは `diff -r N-1:N` や `diff -c N` に相当します。

下のリストには、選択したリビジョンで変更されたすべてのファイルが表示されます。そのため、コンテキストメニューには常に 変更を表示 となるのです。

4.9.4. ログメッセージの追加取得

ログダイアログは以下のような場合、変更履歴を最後まで表示しないことがあります。

- 大きなリポジトリでは何百何千もの変更があり、すべて取得するには時間がかかるため。しかし通常参照する変更は最近のものだけでしょう。デフォルトでは取得するログメッセージは100件に制限されていますが、 TortoiseSVN → 設定「TortoiseSVN ダイアログ設定1」)で変更できます。
- **コピー/名前の変更が発生したら停止** がチェックされていると、ログの表示は選択されたファイルやフォルダーが他の場所からコピーされてきた時点で停止します。これはブランチ(やタグ)を参照するときに、ブランチの作成時点で停止するので、ブランチ内部の変更をすばやく表示できるので便利です。

  通常、このオプションをチェックしないままにしておきたいのではないでしょうか。TortoiseSVN はチェックボックスの状態を記憶していますので、その設定が尊重されます。

  マージダイアログからログダイアログを表示すると、このチェックボックスは、デフォルトで常にチェックされます。これは通常のマージ作業ではブランチ内部の変更を調べるものであり、ブランチの作成時点を超えて遡るようなことはないからです。

  Subversion は現在、名前の変更をコピー・削除で行います。そのため、このオプションにチェックがついていると、ファイルやフォルダーの名前を変更した時点でログの表示が停止することに注意してください。

ログメッセージをもっと見たい場合は、**次の100件** をクリックして次の100件のログメッセージを取得してください。必要なだけ繰り返せます。

このボタンの隣に多機能ボタンがあります。前回使用したオプションが記憶されていますが、三角形をクリックすると、他のオプションを指定できます。

表示する範囲を指定... を使用すると、表示するリビジョンの範囲を指定することができます。ダイアログが表示されるので、開始リビジョンと終了リビジョンを指定してください。

すべて表示 を使用すると、HEAD からリビジョン 1 までさかのぼって すべての ログメッセージを表示することができます。

ログダイアログを開いている間に他のコミットが実行された場合など、最新のリビジョンに更新したい場合は、F5 キーを押してください。

ログキャッシュを読み込みなおすには、Ctrl+F5 キーを押してください。

4.9.5. 現在の作業コピーのリビジョン

ログダイアログは、現在の作業コピーのリビジョンではなく、HEAD リビジョンからログを表示するため、まだ作業コピーに取得されていない部分のログメッセージが表示されることがよくあります。これを明確にするために、作業コピーのリビジョンに一致するコミットメッセージは太字で表示されます。

フォルダーのログが表示される際、太字で表示されるリビジョンは、フォルダー内のファイルの最新のリビジョンになります。これには作業コピーのクロールが必要です。クロールは別スレッドで実行するため、ログ表示に遅延は起こりませんが、フォルダーによってはすぐに太字で表示されないことがあります。

4.9.6. マージ追跡機能

Subversion 1.5 以降では、プロパティでマージ情報を保持しています。これにより、マージでの変更点の詳細な履歴を得られます。例えば、ブランチで新機能を開発し、それをトランクにマージした場合、たとえブランチでの開発で1000回コミットしたとしても、トランクではマージした1回分のログしか存在しなくなります。

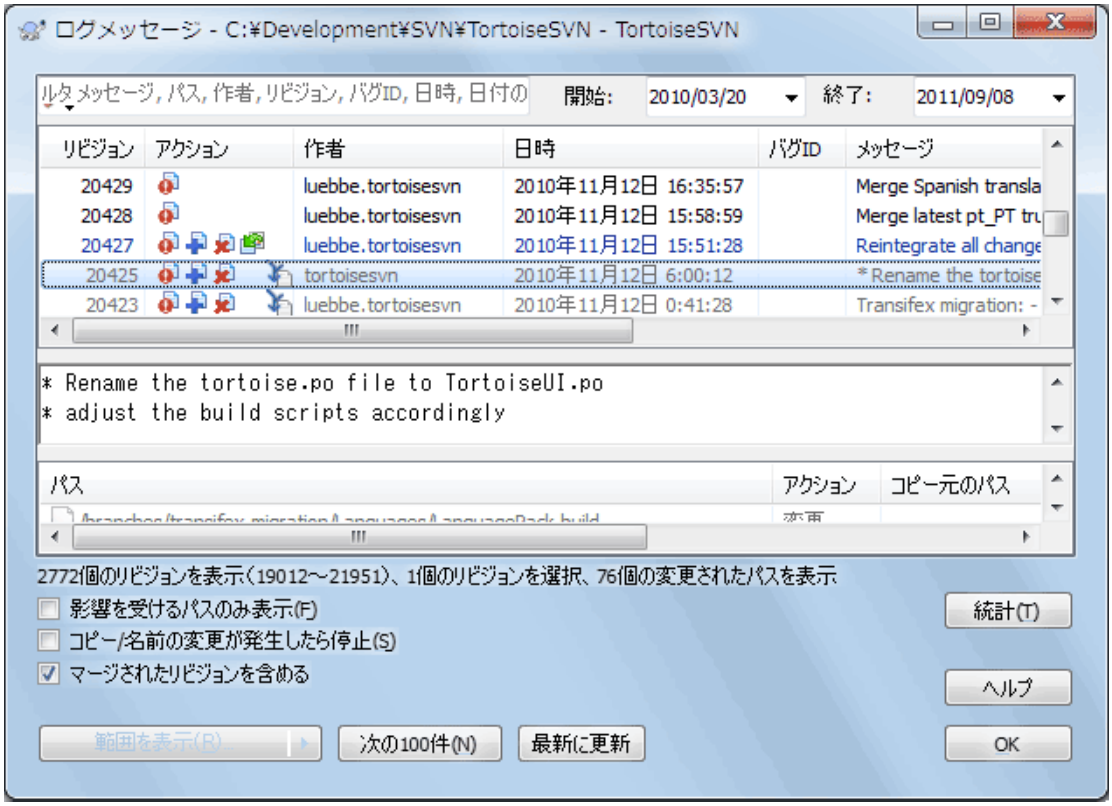


図4.20 マージ追跡リビジョンを表示したログダイアログ

マージを含んだコミットのリビジョンの詳細を見る場合、 **マージされたリビジョンを含める** チェックボックスを使用してください。これによりログメッセージを再取得しますが、マージされたリビジョンに由来するのログメッセージが挿入されます。マージされたリビジョンは、別ツリーで変更されているので、灰色で表示されます。

もちろん、マージは決して単純ではありません。ブランチでの機能開発中、メインラインコードと同期を取り続けるために、トランクからたびたび行わなければならないでしょう。そのため、ブランチのマージ履歴には、別レイヤのマージ履歴が含まれていることでしょう。ログダイアログでは、異なるレイヤをインデントレベルで表します。

4.9.7. ログメッセージや作者の変更

リビジョンプロパティは、項目の Subversion プロパティとは異なるものです。リビジョンプロパティは、ログメッセージやコミット日時、コミットした人（作者）といった、リポジトリの特定のリビジョン番号に関連する説明的な項目です。

時には、一度入力したログメッセージを変更したいこともあるでしょう。綴り間違いがあったり、またメッセージを改善したり、その他の理由で変更したくなるかもしれません。また、コミットした作者を変更したくなるかもしれません。認証を設定し忘れてしまったとか……

Subversion では、必要に応じてリビジョンプロパティをいつでも変更できます。しかし、その変更を元に戻せない（この変更はバージョン管理外になります）ので、デフォルトではこの機能は無効になっています。有効にするには、 pre-revprop-change フックを設定する必要があります。この手順の詳細については、 Subversion Book の Hook Scripts [http://svnbook.red-bean.com/en/1.7/svn.reposadmin.create.html#svn.reposadmin.create.hooks] をご覧ください。 Windows マシンでのフックの実装については、「サーバー側フックスクリプト」をご覧ください。

いったん必要なフックをサーバーに設定してしまえば、任意のリビジョンに対して、作者やログメッセージ（またはその他のリビジョンプロパティ）を変更できます。ログダイアログの上部のリストのコンテキストメニューを使用してください。中央の欄のコンテキストメニューを使用して、ログメッセージを編集することもできます。

警告

Subversion のリビジョンプロパティはバージョン管理されないため、リビジョンプロパティ（例えば `svn:log` コミットメッセージプロパティ）を変更すると、直前の値を 完全に 上書きしてしまいます。

重要

TortoiseSVN はすべてのログ情報のキャッシュを保持しているため、変更を行った作者やログメッセージはローカル環境でのみ表示されます。ログキャッシュが更新されるまでの間、他のユーザーには、キャッシュされた（古い）作者やログメッセージが表示されます。

4.9.8. ログメッセージの絞り込み

興味あるログメッセージを限定して表示させたい場合は、何百もあるリストをスクロールするよりも、ログダイアログの上端にある絞り込みコントロールを使用したほうが便利です。日付の範囲を指定すると、その範囲のメッセージだけに絞り込むことができます。検索ボックスに文字列を入力すると、その文字列が含まれるメッセージだけに絞って表示できます。

検索アイコンクリックすると、検索対象となる情報の種類や、 正規表現 を使用するかどうかを選択できます。普段は単純な文字列検索で充分でしょうが、より複雑な検索条件を指定したい場合は、正規表現を使用することができます。検索ボックスの上にマウスを置くと、正規表現の記号や特殊文字の使い方がツールチップで表示されます。絞り込み文字列がログ項目と一致するかどうかがチェックされ、絞り込み文字列と 一致 した項目のみが表示されます。

部分文字列での検索の仕方は、検索エンジンと似ています。複数の文字列を空白で区切って入力すると、すべての文字列を含む項目が検索されます。文字列の前に - を付けると、その文字列が含まれないものが検索されます（語句の否定検索）。そして、最初に ! を付けると、検索文字列全体が否定になります。前に + を付けた文字列は、以前に

- を付けて除外した文字列であっても、含まれるものが検索されます。なお、含まれる・除外される指示の順番が重要であることに気を付けてください。空白を含む文字列を検索する場合は、文字列を引用符で囲んでください。引用符の文字を含む文字列を検索する場合は、引用符を2つ並べてエスケープしてください。なお、バックスラッシュ文字はエスケープ文字 ではなく、部分文字列検索では特別な意味を持たないことに注意してください。次のサンプルを見たほうが簡単でしょう。

```
Alice Bob -Eve
```

は、Alice と Bob を含み、Eve を含まない文字列を検索します。

```
Alice -Bob +Eve
```

は、Alice を含んでかつ Bob を含まないか、Eve を含む文字列を検索します。

```
-Case +SpecialCase
```

は、Case を含まないが、SpecialCase は含む文字列を検索します。

```
!Alice Bob
```

は、Alice も Bob も含まない文字列を検索します。

```
!-Alice -Bob
```

は、ド・モルガンの法則を思い出してください。NOT(NOT Alice AND NOT Bob) なので、(Alice OR Bob) となります。

```
”Alice and Bob”
```

は、「Alice and Bob」という文字列を検索します。

```
””
```

は、テキストのどこかに二重引用符が含まれている文字列を検索します。

```
”Alice says ””hi”” to Bob”
```

は、「Alice says "hi" to Bob」 という文字列を検索します。

正規表現についての説明はこのマニュアルの範囲外です。 http://ja.wikipedia.org/wiki/正規表現 [http://ja.wikipedia.org/wiki/%E6%AD%A3%E8%A6%8F%E8%A1%A8%E7%8F%BE] にある説明や、 http://www.regular-expressions.info/ にあるオンラインのドキュメントやチュートリアルをご覧ください。

絞り込みは、すでに取得したメッセージのみが対象になることに注意してください。リポジトリからメッセージをダウンロードすることはありません。

また、 **影響を受けるパスのみ表示** を使用すると、下の欄のパス名を絞り込みすることもできます。影響を受けるパスとは、ログを表示するために使われたパスを含むパスです。フォルダーのログを取得する場合は、そのフォルダー以下すべてが対象になります。ファイルの場合は、そのファイルのみが対象になります。通常、パスのリストには影響を受けるパス以外にも、同時にコミットされたパスが灰色で表示されます。ボックスをチェックすると、それらのパスは非表示になります。

運用上、ログメッセージを特定の書式に統一したい場合があります。変更点の説明文が、上部のリストに表示される冒頭の部分に表示されない場合などです。 `tsvn:logsummary` プロパティを使用すると、上部のリストに表示されているログメッセージの一部を抽出できます。このプロパティの使い方は、「TortoiseSVN のプロジェクトプロパティ」をご覧ください。

リポジトリブラウザーからログのフォーマット変更はできません

ログのフォーマット変更は、Subversion のプロパティとして保持されているため、チェックアウトした作業コピーを使用するときのみ表示されます。リモートからプロパティを取得するのには時間がかかるため、リポジトリブラウザーから起動した場合はこの機能が表示されません。

4.9.9. 統計情報

ログダイアログの 統計 ボタンは、リビジョンについての興味深い情報を表示します。作業者が何人いるのか、コミットは何回行われているのか、週ごとの進行状況、等といった情報です。これで、誰ががんばり屋で誰がなまけ者か一目で分かります。;-)

4.9.9.1. 統計ページ

このページでは、期間やリビジョン数、最小・最大・平均などの集計値のような参考になる数値を提供します。

4.9.9.2.「作者別コミット数」ページ

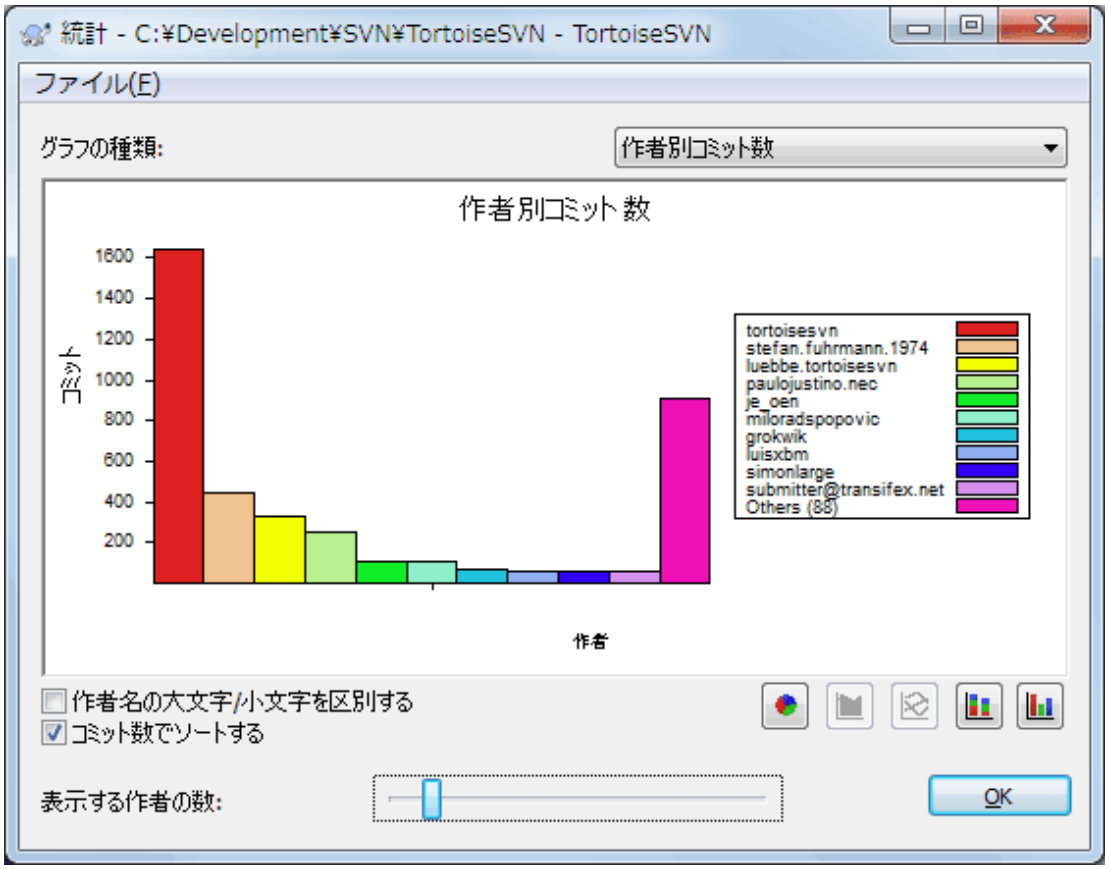


図4.21「作者別コミット数」ヒストグラム

このグラフは、プロジェクト内のどのユーザーがアクティブかを、ヒストグラム、積み重ね棒グラフ、円グラフで表します。

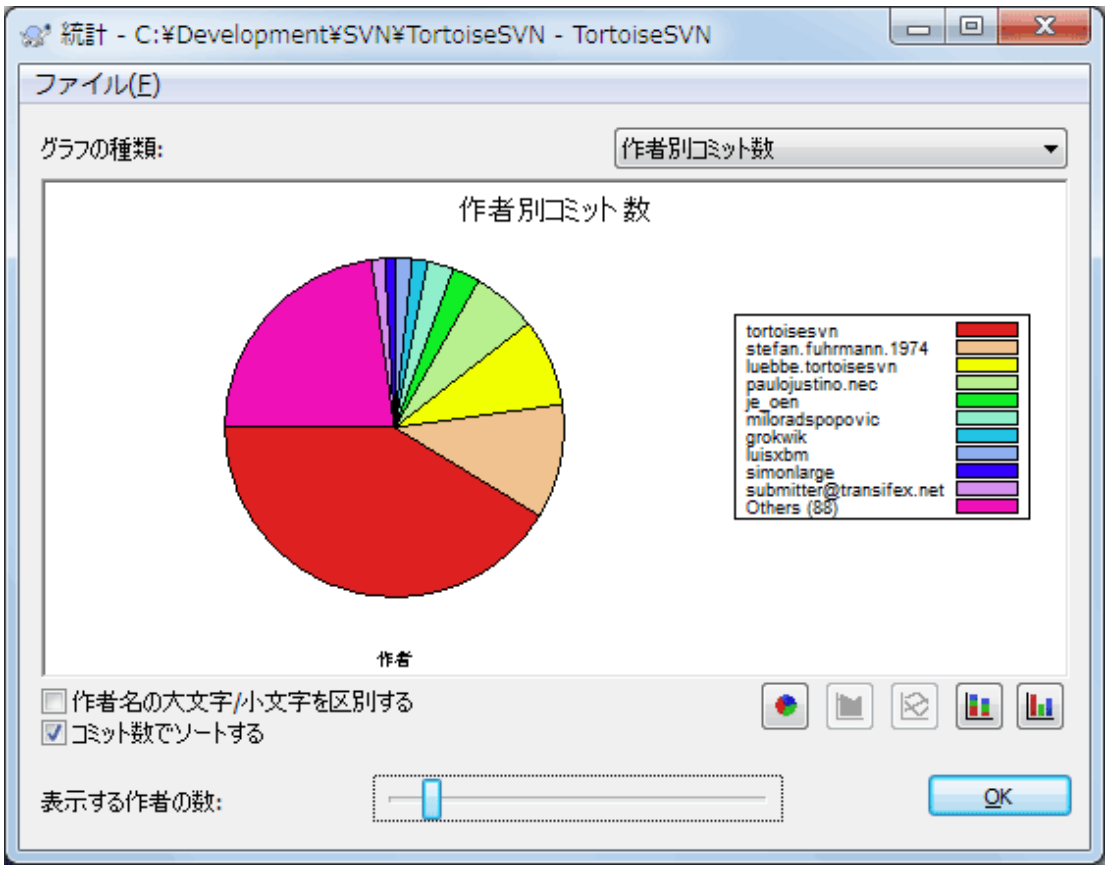


図4.22「作者別コミット数」円グラフ

少数の主要な作者と、多数の細かい協力者がいる環境では、細分化されてグラフが読みにくくなります。グラフの下にあるスライダーを使用して閾値(全コミットにおけるパーセンテージ)を設定すると、活動状況が閾値以下の人が その他カテゴリにグループ化されます。

4.9.9.3.「時期別コミット数」ページ

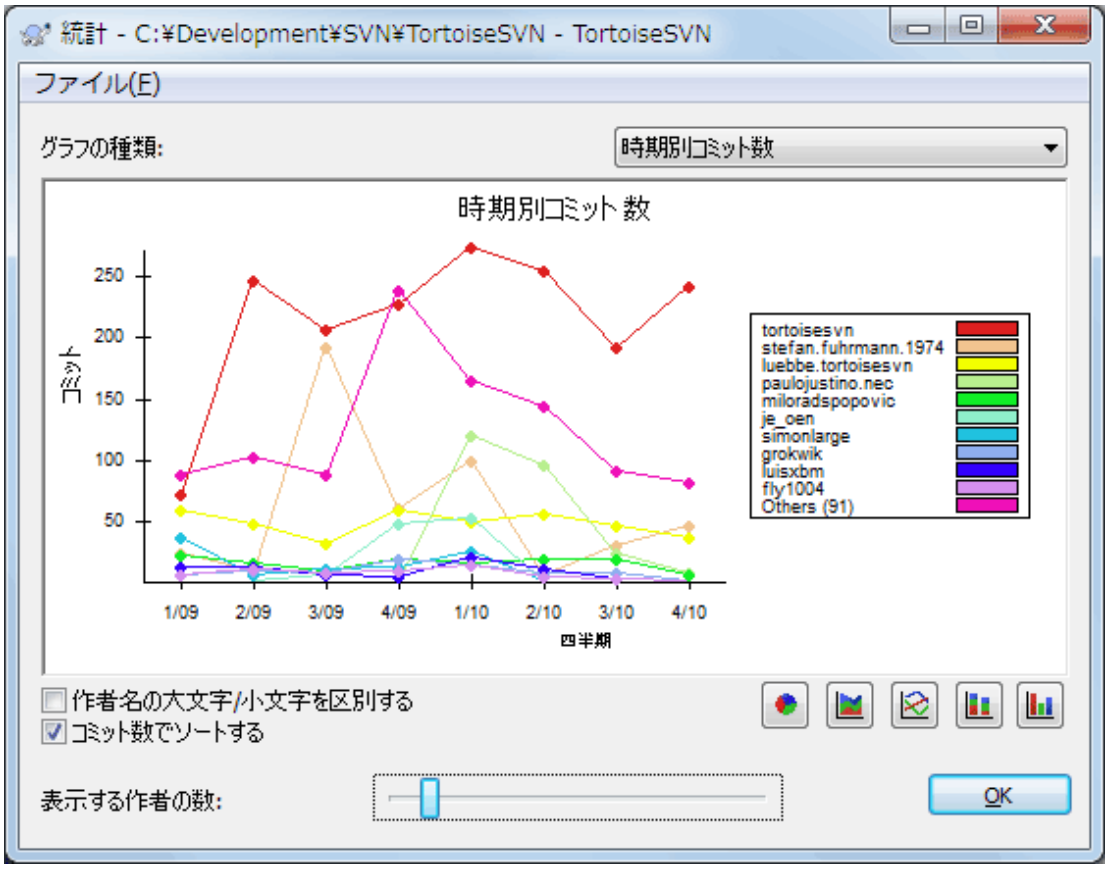


図4.23「時期別コミット数」グラフ

このページでは、プロジェクトの活動状況を、一定期間のコミット回数 および 作者別で表示します。これによりプロジェクトがいつ活動していたか、そのとき誰が作業したかといった判断材料になるでしょう。

多数の作者がいる場合、グラフにたくさんの線が表示されます。この画面ではさらに2つの表示方法が使用できます。折れ線グラフ は、作者の活動状況ごとに表示し、 積み重ね面グラフ は、作者の活動状況を積み重ねて表示します。後者では線が交差しなくなるためため、グラフは読みやすくなりますが、個別の作者の状況は読みにくくなります。

デフォルトでは、分析する際、ユーザー名の大文字と小文字が区別されます。そのため、PeterEgan と PeteRegan というユーザーは別人として扱われます。しかし多くの場合、ユーザー名の大文字と小文字を区別しておらず、時には混同して入力されます。つまり、DavidMorgan と davidmorgan を同一として扱うということです。**作者名の大文字/小文字を区別しない** チェックボックスを使用して、どのように扱うかを決めてください。

この統計はログダイアログと同じ期間が対象になることに注意してください。1リビジョンしか表示していないときには、統計には大した情報は現れません。

4.9.10. オフラインモード

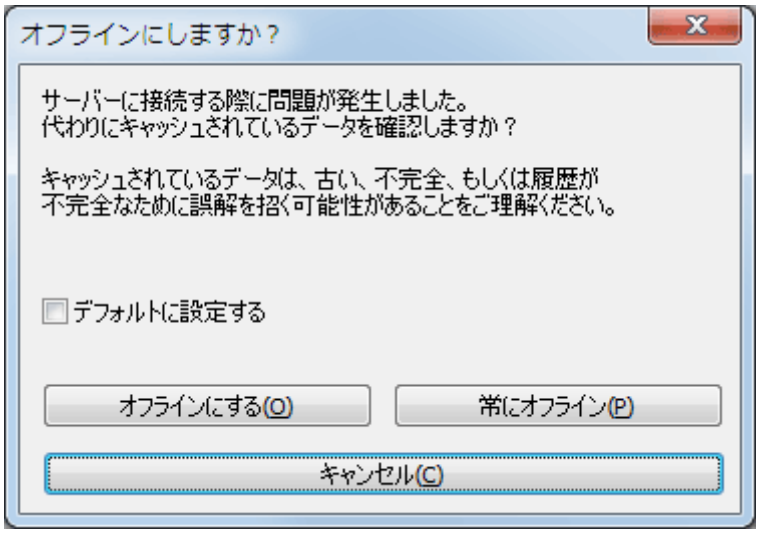


図4.24 オフライン移行ダイアログ

ログキャッシュを有効にしている場合、サーバーに接続できなくても、ログダイアログやリビジョングラフはオフラインモードで使用できます。これはキャッシュからのデータを使用し、作業を継続できますが、おそらく最新でも完全でもありません。

ここでは以下の3つの選択肢があります。

オフラインにする
: 現在の操作を完全にオフラインモードにしますが、ログデータが次回要求されたときに、リポジトリへの接続を再試行します。

常にオフライン
: リポジトリのチェックを明確に要求されるまで、オフラインモードのままにします。「表示の更新」をご覧ください。

キャンセル
: おそらく陳腐化したであろうデータで操作を継続したくない場合は、単にキャンセルとしてください。

デフォルトに設定する　チェックボックスは、このダイアログを再度表示しないようにし、選択を常に使用します。ここで設定したとしても、TortoiseSVN → 設定 で、デフォルト値の変更（削除）が行えます。

4.9.11. 表示の更新

新しいログメッセージを取得するためにサーバーを再チェックする場合、単純に　F5　で表示を更新できます。ログキャッシュを使用している場合（デフォルトで有効）、新しいログメッセージがあるかリポジトリをチェックし、新しいもののみを取得します。ログキャッシュがオフラインモードになっている場合、オンラインモードに切り替えを試みます。

ログキャッシュを使用しており、メッセージの内容や作者を更新したい場合、Shift+F5　や　Ctrl+F5　で表示しているメッセージをサーバーから再取得し、ログキャッシュを更新できます。これは現在表示されているメッセージにしか効果はなく、そのリポジトリのキャッシュ全体が無効になるわけではないことに注意してください。

4.10. 差分の表示

プロジェクト開発を進める上でもっとも共通の需要のひとつが、何を変更したかを確認するということです。同じファイルの2つのリビジョン間や、2つの異なるファイルの違いを確認したくなることがあるでしょう。　TortoiseSVN　には、TortoiseMerge というテキストファイルの差分を表示する内蔵ツールがあり、TortoiseIDiff という画像ファイルの差分を見るためのツールもあります。もちろん、任意の差分プログラムを使うこともできます。

4.10.1. ファイルの差分

ローカルの変更

自分　　が作業コピー内でどんな変更を行ったか確認する場合は、エクスプローラーのコンテキストメニューで TortoiseSVN → 差分を表示 を選択してください。

別のブランチ/タグとの差分

(ブランチで作業していて)トランクの変更点を見る場合や、(トランクで作業していて)特定のブランチの変更点を見る場合、エクスプローラーのコンテキストメニューを使用できます。Shift キーを押したままファイルを右クリックし、TortoiseSVN → URL を指定して差分を表示 を選択してください。その後で表示されるダイアログでは、ローカルのファイルと比較するリポジトリの URL を指定してください。

2つのツリー、たとえば2つのタグや、ブランチ/タグとトランクを選択して差分を取るときは、リポジトリブラウザーを使用することもできます。そこでコンテキストメニューの リビジョンを比較 を使用することで比較ができます。「フォルダーの比較」をご覧ください。

以前のリビジョンとの差分

特定のリビジョンと作業コピーの差分を取る場合は、リビジョンログダイアログを使用します。比較したいリビジョンを選択し、コンテキストメニューから 作業コピーと比較 を選択してください。

最後にコミットしたリビジョンと、変更されていないと思われる自分の作業コピーとの差分を見る場合は、単にファイルを右クリックし、TortoiseSVN → 直前のバージョンとの差分を表示 を選択してください。ここでは、(作業コピーに記録されている)最終コミット日時以前のリビジョンと、作業ベースとの差分を表示します。これにより、現在の作業コピーの元となる直前の変更点を参照できます。作業コピーよりも新しい変更点は表示されません。

2つのリビジョン間の差分

すでにコミットした2つのリビジョン間の差分を取るのなら、リビジョンログダイアログを使用し、(通常 Ctrl を使用して)比較したい2つのリビジョンを選択してください。それから、コンテキストメニューで リビジョンを比較 を選択してください。

フォルダーのリビジョンログから行った場合、リビジョン比較ダイアログが現れ、フォルダーにある変更されたファイルの一覧を表示します。「フォルダーの比較」をご覧ください。

コミットしたすべての変更

特定のリビジョンで行ったすべてのファイルの変更点を一目で確認したいなら、Unified差分ファイル(GNUパッチ形式)を使用できます。ここでは差分が本文内の数行のみで表示されます。視覚的なファイル比較よりは読みにくくなりますが、すべての変更を一度に確認できます。リビジョンログダイアログから確認したいリビジョンを選択し、コンテキストメニューから Unified形式で差分を表示 を選択してください。

ファイル間の差分

2つの異なるファイルの差分を確認したい場合、エクスプローラーで直接両方(Ctrl　を押しながら)選択してください。その後、エクスプローラーのコンテキストメニューからTortoiseSVN → 差分 を選択します。

作業コピーのファイル・フォルダーと URL 間の差分

作業コピーのファイルと、Subversion リポジトリのファイルとの差分を確認する場合は、エクスプローラーで直接そのファイルを選択し、Shift キーを押しながら右クリックしてコンテキストメニューを表示させ、TortoiseSVN → URL を指定して差分を表示　を選択してください。作業コピーのフォルダーの場合も同様です。 TortoiseMerge　ではパッチファイルの場合の同様、変更のあったファイルの一覧が表示され、それぞれの差分を個別に確認できます。

注釈履歴の差分

リビジョンログダイアログから差分と注釈履歴を組み合わせて表示させ、ファイルの差分ばかりでなく、作者、リビジョン、更新日時を確認することができます。詳しくは、「注釈履歴の差分」をご覧ください。

フォルダー間の差分

TortoiseSVN の内蔵ツールは、ディレクトリ階層の比較をサポートしていません。しかし、その機能をサポートしている外部ツールを使えばディレクトリを比較できます。「外部差分・マージツール」 で使用できるツールをご紹介します。

サードパーティの差分ツールを設定している場合、差分コマンドを選択する際にShift を押すと代替ツールを使用できます。その他の差分ツールの設定については、「外部プログラムの設定」をご覧ください。

4.10.2. 改行コードと空白のオプション

プロジェクトを続けていくと、時には改行コードを CRLF から LF に変更したり、セクションのインデントを変更したりすることがあります。不幸なことに、かなりの行を変更しなければならないにもかかわらず、コードの意味に変化はありません。以下のオプションでは、比較や差分を適用する際の変更点の把握方法を管理します。これらの設定は、 TortoiseMerge の設定のほか、**マージ** や **注釈履歴** ダイアログから行うことができます。

改行コードを無視する は、改行コードのみの変更を無視します。

空白を比較する は、インデントや行内の空白の追加・削除を変更点に含めます。

空白の変更を無視する は、空白の数や種類（インデントや、タブとスペースの変更など）のみの変更を無視します。空白がないところに追加されたり、存在する空白が完全に削除されたりした場合は、変更として表示されます。

すべて空白を無視する は、空白のみの変更をすべて無視します。

当然、内容の変更のある行は、常に差分に含まれます。

4.10.3. フォルダーの比較

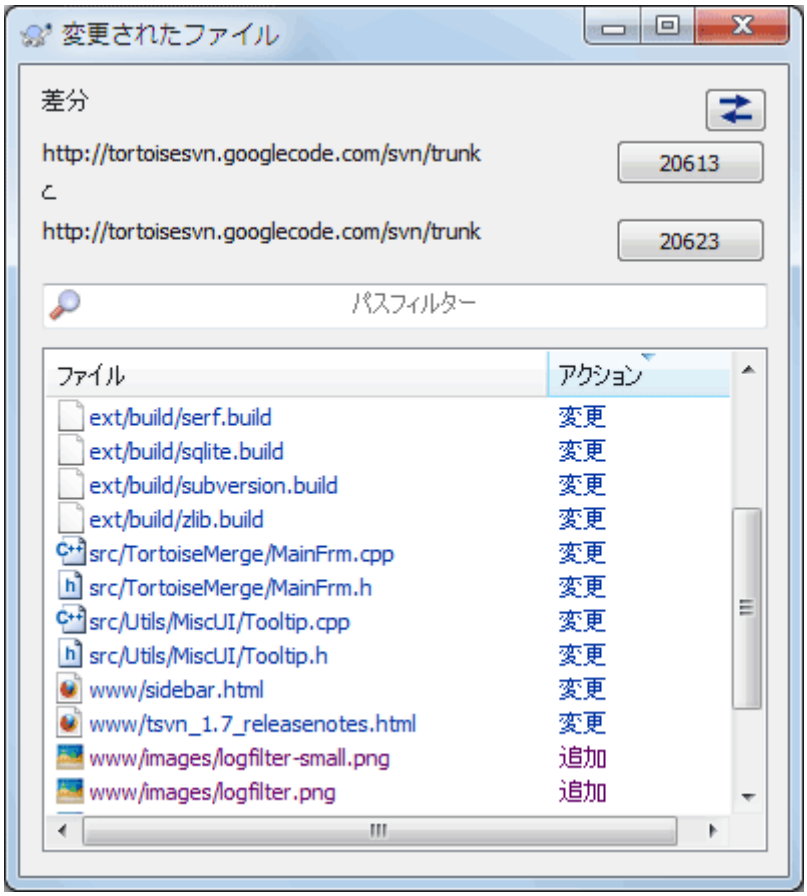


図4.25 リビジョンの比較ダイアログ

リポジトリブラウザーで2つのツリーを選択したり、ログダイアログでフォルダーのの2つのリビジョンを選択したりすると、**コンテキストメニュー → リビジョンを比較** が実行できるようになります。

このダイアログでは、変更されたすべてのファイルが表示され、コンテキストメニューから個々に比較や注釈履歴の取得を行えます。

変更ツリー　をエクスポートすることができます。これは、プロジェクトのツリー構造の中で、変更されたファイルのみを他人に渡したい場合に便利です。この操作は選択したファイルにのみ作用しますので、あらかじめエクスポートするファイル（通常はすべて）を選択してから、**コンテキストメニュー → 選択をエクスポート...** を実行してください。そして、変更ツリーを保存する場所を指定してください。

また、**コンテキストメニュー → 選択ファイルのリストを保存...** を使用すると、変更されたファイルの 一覧 をエクスポートすることができます。

ファイルの一覧 と 行われた操作（変更・追加・削除）をエクスポートする場合、**コンテキストメニュー → 選択範囲をクリップボードにコピー** を使用してください。

最上部のボタンは、比較の方向を変更します。AからBへの変更を表示しているとき、必要に応じて、BからAへの変更も表示できます。

リビジョン番号が表示されているボタンで、リビジョン範囲を変更することができます。範囲を変更すると、2つのリビジョン間の差分が自動的に更新されます。

ファイル名の一覧が非常に長い場合、検索ボックスを使用して、特定のテキストを含むファイル名のみを抽出できます。単純文字列検索を行いますので、Cのソースファイルを指定したい場合は、`*.c` ではなく `.c` と指定してください。

4.10.4. TortoiseIDiff を使用した画像の差分

テキストファイルの差分を取るツールは、TortoiseMerge を含めてたくさんありますが、私たちはよく画像ファイルに、どんな変更が加えられたかを知りたくなることもありました。そのために TortoiseIDiff を作成しました。

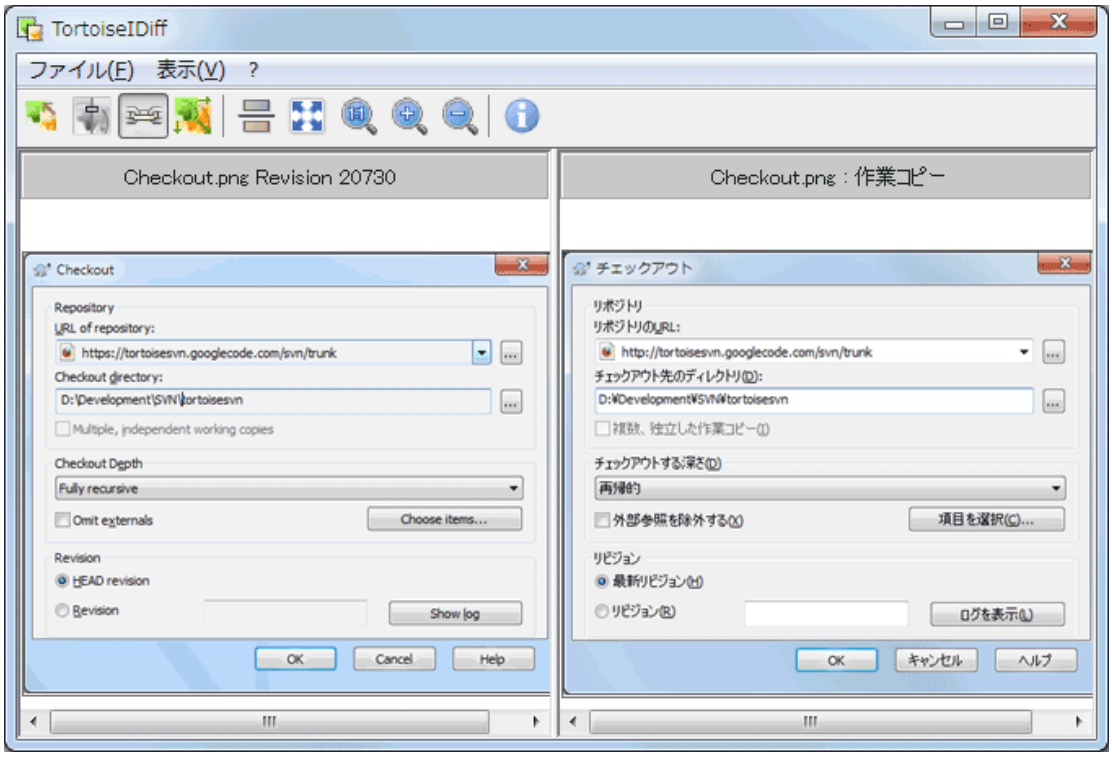


図4.26 画像差分ビューアー

一般的な画像ファイル形式のファイルを選択して TortoiseSVN → **差分を表示** を実行すると、画像の差分を表示するために TortoiseIDiff が起動します。デフォルトでは画像が横に並んで表示されますが、表示メニューやツールバーで上下に表示するように切り替えることができます。また、ライトボックスを使用した時のように、画像を重ねて表示することもできます。

画像をズームイン・アウトしたり、移動したりすることもできます。画像を左ドラッグすることで、移動することができます。**画像位置を同期** オプションを選択しておくと、スクロールバーやマウスホイールで画像を移動する時、両方の画像が同期して移動するようになります。

画像情報ボックスには、ピクセル単位のサイズや、解像度、色深度など、画像ファイルの詳細を表示されます。このボックスが邪魔になるなら、**表示** → **画像情報** で隠すことができます。これと同じ情報は、画像のタイトルバーの上にマウスを移動すると表示されます。

画像を重ねて表示している場合、画面左端のスライダーを使用して、画像の相対的な輝度（アルファブレンド）を制御できます。スライダー上をクリックすると、透明度を直接設定できますし、スライダをドラッグすると、確認しながら透明度を変更できます。Ctrl+Shift+ホイール でも透明度を変更できます。

スライダの上にあるボタンを押すと、透明度を 0% と 100% で切り替えられます。また、ボタンをダブルクリックすると、もう一度ボタンをクリックするまで、1秒ごとに透明度を自動で交互に切り替わります。複数の小さな変更を探すのに便利でしょう。

重ね合せでは分からない差分を確認したい場合もあります。プリント基板の画像イメージに2つのリビジョンがあり、どの配線が変更されたかを調べたい場合などです。アルファブレンドを無効にすると、各ピクセルの色が XOR されて差分が表示されます。変更のない部分は真っ白になり、変更点だけ色が表示されます。

4.10.5. Office ドキュメントの差分

テキストファイル以外の文書の差分を表示するには、ファイルフォーマットを理解できるソフトウェアを使用する必要があります。通常は、その文書を作成するために使用したソフトウェアを使用します。よく使用される Microsoft Office や OpenOffice.org には差分を表示するための機能があり、TortoiseSVN には有名な拡張子のファイルの差分を表示する際に、これらのソフトウェアをしかるべき設定で呼び出すことができるスクリプトが添付されています。どのような拡張子がサポートされているかを確認したり、新しく追加したりするためには、TortoiseSVN → **設定**を実行し、**外部プログラム** セクションの **高度な設定** をクリックしてください。

Office 2010 使用時の問題

Office 2010 の クイック実行 版をお使いの場合、文書の差分を表示しようとすると、Windows Script Host から、「ActiveX コンポーネントはオブジェクトを作成できません: word.Application」というようなエラーが表示されることがあります。差分機能を使用するには、MSI 版の Office を使用する必要があるようです。

4.10.6. 外部差分・マージツール

私たちが提供するツールで必要なことが実現できない場合は、オープンソース・商用のツールがたくさんあるので、その中から使用できるものを試してみてください。それぞれに特徴がありますし、この一覧は完全なものではありませんが、検討に値するものをいくつか以下に紹介します。

WinMerge
: WinMerge [http://winmerge.sourceforge.net/] は、ディレクトリの差分も扱える画期的なオープンソースの差分ツールです。

Perforce Merge
: Perforce は商用の RCS ですが、差分・マージツールは無料でダウンロードできます。詳細な情報は Perforce [http://www.perforce.com/perforce/products/merge.html] をご覧ください。

KDiff3
: KDiff3 は、ディレクトリも扱えるフリーの差分ツールです。こちら [http://kdiff3.sf.net/] からダウンロードできます。

SourceGear DiffMerge
: SourceGear Vault は商用の RCS ですが、差分・マージツールは無料でダウンロードできます。詳細な情報は SourceGear [http://www.sourcegear.com/diffmerge/] をご覧ください。

ExamDiff
: ExamDiff Standard はフリーウェアです。ファイルを扱えますが、ディレクトリは扱えません。ExamDiff Pro はシェアウェアで、ディレクトリに対する差分や編集機能などたくさん追加されています。どちらもバージョン 3.2 以降なら unicode を扱えます。PrestoSoft [http://www.prestosoft.com/] からダウンロードできます。

Beyond Compare
: ExamDiff Pro に似て、ディレクトリの差分と unicode を扱えるすばらしいシェアウェアです。Scooter Software [http://www.scootersoftware.com/] からダウンロードしてください。

Araxis Merge
: Araxis Merge はファイルやフォルダーの両方に対応した、差分やマージに便利な商用ツールです。マージ時に3方向の比較を行うことができ、また関数の順番の変更に対応できる同期リンク機能を持っています。 Araxis [http://www.araxis.com/merge/index.html] からダウンロードしてください。

TortoiseSVN でこれらのツールを使用するよう設定する方法は「外部プログラムの設定」をご覧ください。

4.11. 新しいファイルやディレクトリの追加

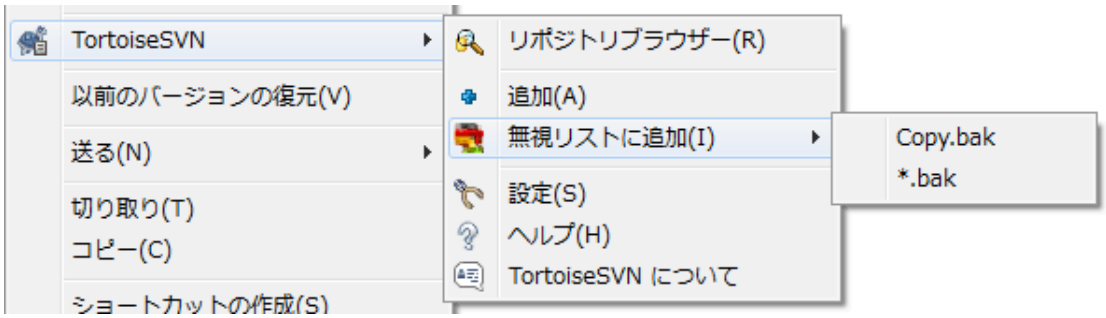


図4.27 バージョン管理外のファイルでのエクスプローラーコンテキストメニュー

開発工程の中で新しいファイルやディレクトリを作成したら、ソース管理に追加する必要があります。追加するファイルやディレクトリを選び、TortoiseSVN → 追加 を実行してください。

ファイルやディレクトリをソース管理に追加すると、 追加 アイコンオーバレイが表示され、他の開発者もそのファイルやディレクトリを使用できるよう、早く作業コピーをコミットすることを促します。ファイルやディレクトリの追加だけではリポジトリに影響 しません 。

複数の追加

すでにバージョン管理下にあるフォルダーでも追加コマンドを実行できます。この場合、追加ダイアログにはバージョン管理下のフォルダー内にある、バージョン管理外の全ファイルが表示されます。新しいファイルが複数あり、そのファイルを一度に追加する必要があるときに便利です。

作業コピーの外部にあるファイルを追加するには、ドラッグ&ドロップを使用して次のように操作します。

1. 追加するファイルを選択する
2. 作業コピー内の新しい位置にそれらを右ドラッグする
3. マウスの右ボタンを離す
4. **コンテキストメニュー → SVN この作業コピーにファイルを追加** を選択する。そのファイルが作業コピーにコピーされ、バージョン管理に追加される。

作業コピーにあるファイルを、コミットダイアログへ左ドラッグ&ドロップするだけでも追加できます。

ファイルやフォルダーを間違って追加した場合は、コミットする前に **TortoiseSVN → 追加を元に戻す...** を使用して、追加を取り消しすことができます。

4.12. ファイルやフォルダーのコピー・移動・名前の変更

リポジトリ内の別なプロジェクトにある既存のファイルを流用したくなり、プロジェクトをまたがってコピーしたくなる場合があります。このようにファイルを単純にコピーして追加しても良いのですが、そうするとそれまでの履歴が見えなくなってしまいます。また、後で元のファイルにバグ修正が行われた場合、新しいコピーが Subversion 内の元のファイルに関連づけられていないと、修正を自動的にマージすることができません。

作業コピーの中にあるファイルやフォルダーをコピーするには、右ドラッグメニューを使うのがもっとも簡単です。ファイルやフォルダーを右ドラッグで、作業コピーから他のフォルダー、または同じフォルダー内に向けてドラッグすると、マウスのボタンを離すと同時にコンテキストメニューが現れます。

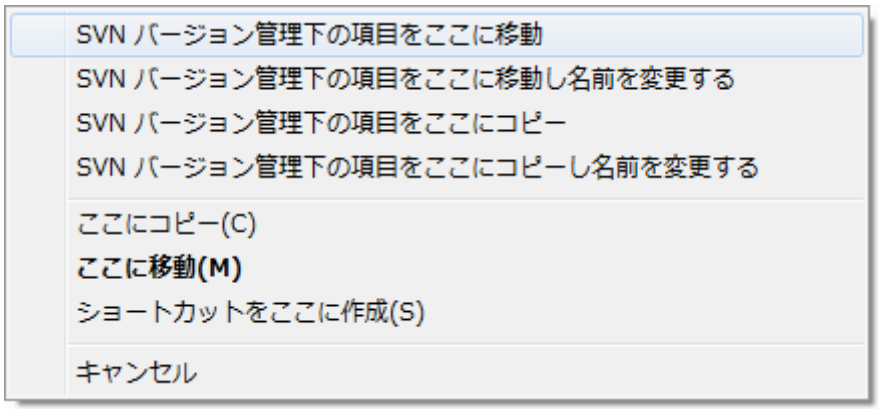


図4.28 バージョン管理下のディレクトリに対する右ドラッグメニュー

これで既存のバージョン管理下の内容を、新しい場所にコピーできます。必要なら同時に名前を変更することもできます。

作業コピーにあるバージョン管理下のファイルや、2つの作業コピーの間でコピーや移動を行う場合は、おなじみのカット&ペースト操作を使用することもできます。バージョン管理下のファイルをクリップボードにコピーするには、 Windows

標準の コピー や 切り取り を実行してください。クリップボードにバージョン管理下のファイルがある状態で、TortoiseSVN → 貼り付け（注意: Windows 標準の 貼り付け ではありません）を使用すると、新しい作業コピーの場所にファイルをコピーや移動することができます。

TortoiseSVN → ブランチ/タグの作成 を使用して、ファイルやフォルダーを作業コピーからリポジトリの別の場所にコピーできます。詳細は、「ブランチ/タグの作成」をご覧ください。

コンテキストメニュー → リビジョンからブランチ/タグを作成 を使用して、ログダイアログにあるファイルやフォルダーの旧バージョンをリポジトリの新しい場所へ、ログダイアログから直接、コピーできます。詳細は、「追加情報の取得」 をご覧ください。

また、リポジトリブラウザーを使用して、ファイルを任意の場所に配置したり、リポジトリから作業コピーに直接コピーしたりすることができます。さらにリポジトリ内の2つの場所の間でコピーできます。詳細は「リポジトリブラウザー」をご覧ください。

リポジトリ間のコピーはできません

TortoiseSVN では、リポジトリの 内部 でファイルやフォルダーはコピーや移動を行うと履歴が保持されますが、リポジトリ間をまたがってファイルやフォルダーをコピーや移動を行うと、履歴が保持されません。リポジトリが同じサーバー上にあったとしても同様です。現状のファイルの内容をコピーし、別なリポジトリに新しい内容として追加することはできます。

同じサーバーの2つの URL が、同じリポジトリを指しているか違うリポジトリを指しているかはっきりしない場合、リポジトリブラウザーを使用して一方の URL を開き、リポジトリルートがどこにあるかを探してください。同じリポジトリブラウザーウィンドウに両方の場所があれば、それらは同じリポジトリにあります。

4.13. ファイルやディレクトリの無視

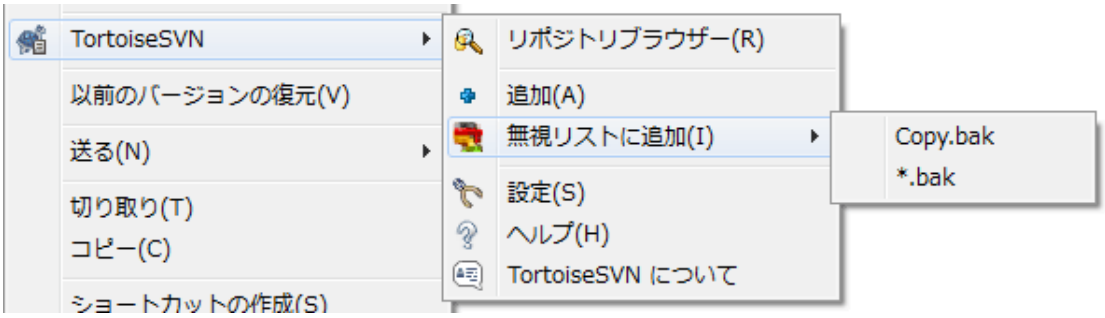


図4.29 バージョン管理外のファイルでのエクスプローラーコンテキストメニュー

ほとんどのプロジェクトでは、バージョン管理するべきでないファイルやフォルダーがあります。*.obj, *.lst のようなコンパイラーが生成するファイルや、もしかしたら実行ファイルを格納する出力フォルダーも該当するかもしれません。変更をコミットする際、TortoiseSVN ではコミットダイアログにバージョン管理外のファイルが表示されます。もちろんこの表示をオフにすることもできますが、そうすると、新しいソースファイルを追加し忘れるおそれがあります。

こういった問題を避けるには、生成されたファイルをプロジェクトの無視リストに追加しておくといいでしょう。こうすると、生成されたファイルはコミットダイアログに現れなくなりますが、本当にバージョン管理する前のソースファイルは現れたままになります。

バージョン管理外のファイルの上で右クリックし、コンテキストメニューで TortoiseSVN → 無視リストに追加 コマンドを選択すると、選択したファイルのみを追加するか、または同じ拡張子を持つファイルすべてを追加するかを

選択するサブメニューが表示されます。複数のファイルを選択した場合はサブメニューが表示されずに、指定されたファイル・フォルダーが追加されます。

無視リストから項目を削除したい場合は、その項目で右クリックし、TortoiseSVN → 無視リストから削除 を選択してください。フォルダーの svn:ignore プロパティを直接変更することもできます。この場合、次項で説明するファイル名展開を使用して、より汎用的なパターンで指定することができます。プロパティを直接指定する方法については、「プロジェクト設定」をご覧ください。複数の無視パターンは改行で区切ってください。空白で区切っても動作しません。

常に無視するパターンのリスト

他にファイルを無視する方法として、 常に無視するパターンのリスト に追加する方法があります。大きな違いは、常に無視するパターンのリストがクライアントのプロパティであるという点です。これは全ての Subversion プロジェクトに適用されますが、クライアント PC でしか適用されません。ふつうは、できれば svn:ignore プロパティを使用した方がいいでしょう。指定されたプロジェクトエリアで適用され、プロジェクトをチェックアウトした人すべてに働くからです。詳細は「一般設定」をご覧ください。

バージョン管理下の項目を無視する

バージョン管理下のファイルやフォルダーを無視させることはできません。これは Subversion の仕様です。間違えてバージョン管理下に入れてしまった場合は、「バージョン管理外のファイルの無視」にある、「バージョン管理外」にする方法をご覧ください。

4.13.1. 無視リストでのパターンマッチ

Subversion の無視パターンは、ワイルドカードとしてメタ文字を使用してファイルを特定する、UNIX で使用されてきたファイル名展開を使用します。特別な意味を持つのは次の文字です。

*
: 空文字列（文字なし）を含む任意の文字列にマッチします。

?
: 任意の1文字にマッチします。

[...]
: 角かっこで囲まれた文字のうちの1文字にマッチします。かっこ内では、「-」を挟んだ2つの文字は、辞書的にその2文字間にある何れかの文字にマッチします。例えば、[AGm-p] は A 、G 、m 、n 、o 、p のいずれかにマッチします。

パターン比較では大文字と小文字を区別しますが、これは Windows で問題の原因になるかもしれません。両方指定すれば、強制的に大文字小文字を区別しないようにできます。例えば、*.tmp を大文字小文字にかかわらず無視するには、*.[Tt][Mm][Pp] のようなパターンで指定してください。

展開方法の公式な定義が必要であれば、シェルコマンド言語の IEEE 仕様書 Pattern Matching Notation [http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html#tag_02_13] を参照してください。

常に無視するパターンのリストにはパスを含めない

パターンにパス情報を含めないでください。パターンの比較は、ファイル名やフォルダー名ごとに行われます。CVS フォルダーをすべて無視したい場合、無視リストには、単に CVS と追加してください。

以前のバージョンのように、CVS */CVS と指定する必要はありません。prog フォルダーにある tmp フォルダーをすべて無視したいけれども、 doc フォルダーにあるものは無視したくない場合は、常に無視するパターンのリストではなく svn:ignore プロパティを使用してください。常に無視するパターンを使用して、これを確実に実現する方法はありません。

4.14. 削除、移動、名前変更

Subversion では、ファイルやフォルダーの名前の変更や移動が可能です。そのため TortoiseSVN のサブメニューには、「削除」や「名前を変更」コマンドがあります。

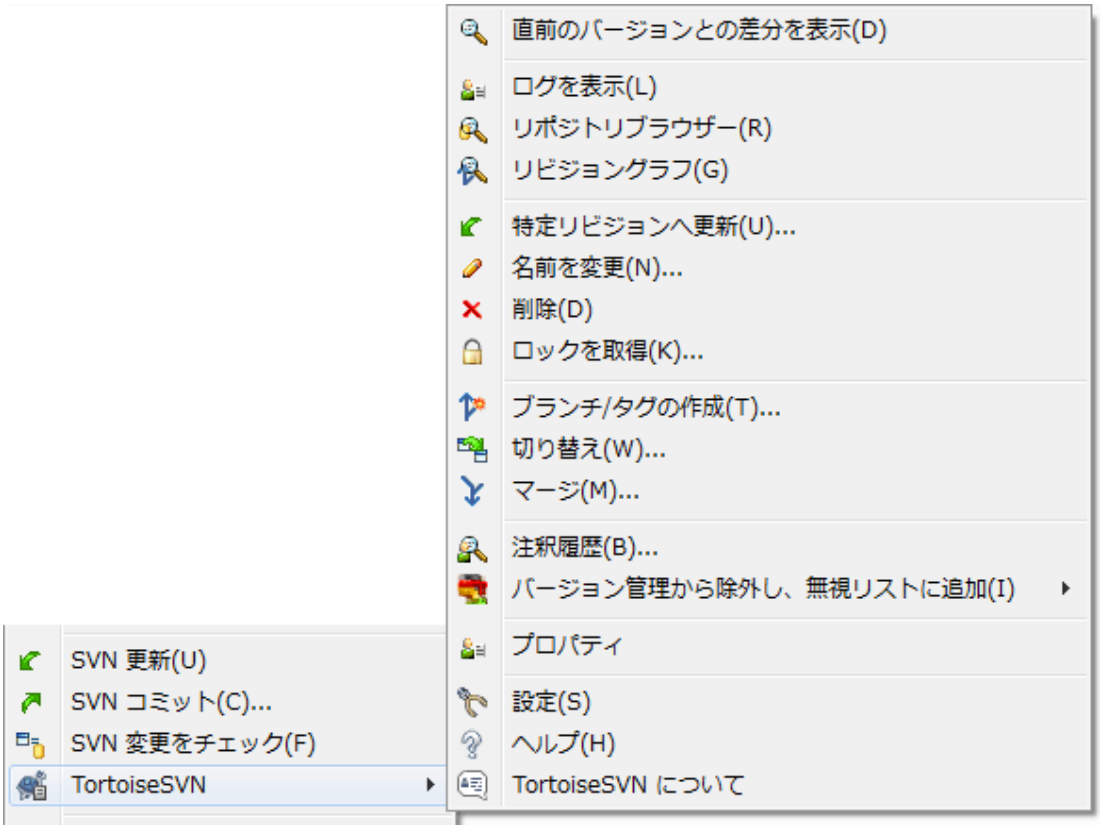


図4.30 バージョン管理下のファイルに対するエクスプローラーのコンテキストメニュー

4.14.1. ファイルやフォルダーの削除

Subversionからファイルやフォルダーを削除するには、TortoiseSVN → 削除 を実行してください。

TortoiseSVN → 削除 を実行すると、ファイルは作業コピーからはすぐに削除され、次のコミットでリポジトリから削除するようにマークを付けます。ファイルの親フォルダーには「削除」 オーバーレイアイコンを表示します。変更をコミットする前ならば、親フォルダーで TortoiseSVN → 変更の取り消し を実行すると、ファイルを取り戻すことができます。

TortoiseSVN → 削除 でフォルダーを削除すると、作業コピーには残りますが、オーバーレイアイコンが変わり削除のマークが付きます。変更をコミットする前ならば、そのフォルダーで TortoiseSVN → 変更の取り消し を実行すると、フォルダーを取り戻すことができます。ファイルとフォルダーの動作の違いは Subversion の部分の動作であり、TortoiseSVN によるものではありません。

項目をリポジトリから削除しても、バージョン管理外のファイルやフォルダーとして残しておきたい場合は、拡張コンテキストメニュー → 削除(作業コピーは保持) を使用してください。エクスプローラーのリストウインドウ(右画面)で Shift キーを押したまま項目を右クリックすると、拡張コンテキストメニューが表示されます。

TortoiseSVN のコンテキストメニューを使用せずに ファイル をエクスプローラーで削除した場合、コミットの際に、コミットダイアログにそのファイルが表示され、バージョン管理からも削除するかどうか確認してきます。しかし、作業コピーに対して更新を実行すれば、 Subversion は紛失したファイルを特定し、リポジトリから最新のファイルを取得して置き換えます。バージョン管理下のファイルを削除する場合は、必ず TortoiseSVN → 削除 を使用するようにしてください。そうすると Subversion はファイルを本当に削除するべきかどうか推測する必要がなくなります。

TortoiseSVN のコンテキストメニューを使用せずに フォルダー をエクスプローラーで削除した場合、作業コピーが破損されてしまいコミットできなくなります。作業コピーに対して更新を実行すれば、Subversion はリポジトリから最新のフォルダーを取得し、紛失したフォルダーと置き換えます。よって、フォルダーを正しい方法で削除するには、 TortoiseSVN → 削除 を使用してください。

削除したファイルやフォルダーを取り戻す

ファイルやフォルダーを削除してリポジトリにコミットした後では、 TortoiseSVN → 変更の取り消し コマンドだけでは取り戻せなくなります。しかし、ファイルやフォルダーが全く失われてしまった訳ではありません。もしファイルやフォルダーを削除したリビジョンが分かるのであれば（分からない場合はログダイアログから探し出して）、リポジトリブラウザーを開いてそのリビジョンに切り替えます。そして、削除されたファイルやフォルダーを選択して、右クリックして コンテキストメニュー → コピー... を実行し、コピー先として作業コピーのパスを指定してください。

4.14.2. ファイルやフォルダーの移動

ファイルやフォルダーの名前をその場で変更するならば、コンテキストメニュー → 名前を変更... を使用してください。新しい名前を入力すれば完了です。

作業コピー内でファイルを（例えば異なるサブフォルダーへ）移動する場合は、マウスの右ドラッグ&ドロップ操作を使用して次のように操作してください。

1. 移動したいファイルやディレクトリを選択する

2. 作業コピー内の新しい位置にそれらを右ドラッグする

3. マウスの右ボタンを離す

4. ポップアップメニューの コンテキストメニュー → SVN バージョン管理ファイルをここに移動する を選択する

親フォルダーでのコミット

名前の変更や移動は、削除した後に追加という形で行われるため、コミットは名前変更または移動したファイルの親フォルダーに対して行わなければなりません。そうでないと、名前変更または移動したファイルがコミットダイアログに表示されません。名前変更または移動した際の削除の部分をコミットしないと、リポジトリ上にそれが残ってしまい、共同作業者が更新した際に、古いファイルが削除されません。つまり、新旧の 両方 のファイルがある状態になってしまいます。

フォルダー内のいずれかのファイルに変更を加える前に、必ず フォルダーの名前の変更をコミットしてください。そうしないと、作業コピーが混乱してしまいます。

ファイルの移動やコピーは、他にも Windows のコピー/切り取りコマンドを使って行うこともできます。コピーしたいファイルを選択し、右クリックしてエクスプローラーの コンテキストメニュー → コピー を実行します。それからコピー先のフォ

ルダーを開いて右クリックし、 TortoiseSVN → 貼り付け を選択します。ファイルを移動する場合は、 コンテキストメニュー → コピー の代わりに コンテキストメニュー → 切り取り を使用します。

また、リポジトリブラウザーを用いて、項目を移動することもできます。詳細は「リポジトリブラウザー」をご覧ください。

外部参照の SVN 移動は行わない

svn:externals で作成したフォルダーに対して、TortoiseSVN の 移動 や 名前を変更 コマンドを実行しては いけません 。この操作は外部参照の項目を親リポジトリから削除してしまうことになり、おそらく他の人たちを驚かせてしまいます。外部参照フォルダーを移動する必要があるときには、通常の(シェルでの)移動を行い、移動元と移動先の各親フォルダーに対して svn:externals プロパティを設定しなおしてください。

4.14.3. ファイル名の大文字・小文字が競合した場合の対処

リポジトリ内に、大文字か小文字かのみが異なる名前のファイル(例えば TEST.TXT と test.txt)がある場合、Windows のクライアントでは、親ディレクトリの更新やチェックアウトができなくなります。Subversion は文字の大小を区別していますが、Windows はそうではないからです。

これは2人が別々の作業コピーから、大文字か小文字かのみが異なる名前のファイルをコミットした場合に発生します。また Linux のような、大文字と小文字を区別するファイルシステムを使用するシステムからコミットした場合も、発生します。

この場合は、どちらかを一方のみを残し、もう一方をリポジトリから削除(または名前変更)しなりません。

2つのファイルが同じ名前にならないようにする方法

大文字・小文字が競合するようなチェックインを拒否するサーバースクリプトが、http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/ にあります。

4.14.4. ファイル名の変更の修復

時に、親切な IDE がリファクタリングの一環でファイルの名前変更を行います。そしてもちろん Subversion に通知しません。変更をコミットすると、 Subversion には古いファイル名が紛失ファイル、新しいファイル名がバージョン管理外のファイルとして見えることになります。新しいファイルを追加することもできますが、Subversion にはそのファイルの関連が分からないので、履歴が追跡できなくなってしまいます。

改善策は、実際には名前変更であったと Subversion に通知することで、これは コミット ダイアログや 変更をチェック ダイアログから行うことができます。古い名前(紛失)と新しい名前(バージョン管理外)の両方を選択し、 コンテキストメニュー → 移動を修復 を使用すれば、2つのファイルを名前変更の関係にすることができます。

4.14.5. バージョン管理外のファイルの削除

通常は、生成される全てのファイルが Subversion に無視されるように、無視リストを設定するでしょう。しかし、クリーンビルドを行うために、これらの無視される項目を削除したい場合はどうしますか?通常は、 makefile でこれを行うよう設定しますが、 makefile がデバッグ中だったり、他のビルドシステムに変更しているときには、これらを丸ごとクリアする手段があると便利です。

TortoiseSVN では、そのような場合に 拡張コンテキストメニュー → バージョン管理外の項目を削除... という機能を提供しています。エクスプローラーのリストウインドウ(右画面)で Shift キーを押したままフォルダーを右クリックすると、拡

張コンテキストメニューが表示されます。これにより、作業コピーにあるバージョン管理外のファイルの一覧ダイアログが現れます。削除する項目を選択または選択解除してください。

ここで項目が削除される際、ごみ箱が使用されますので、ここで誤ってバージョン管理下に置くはずのファイルを削除してしまっても元に戻せます。

4.15. 変更の取り消し

最後に更新を実行したときから、ファイルに行った変更をすべて取り消す場合は、ファイルを選択して右クリックしてポップアップメニューを表示させてから、TortoiseSVN → **変更の取り消し** コマンドを選択してください。ダイアログがポップアップ表示され、変更されて取り消すことができるファイルの一覧が表示されます。変更を取り消すファイルを選択し、OK をクリックしてください。

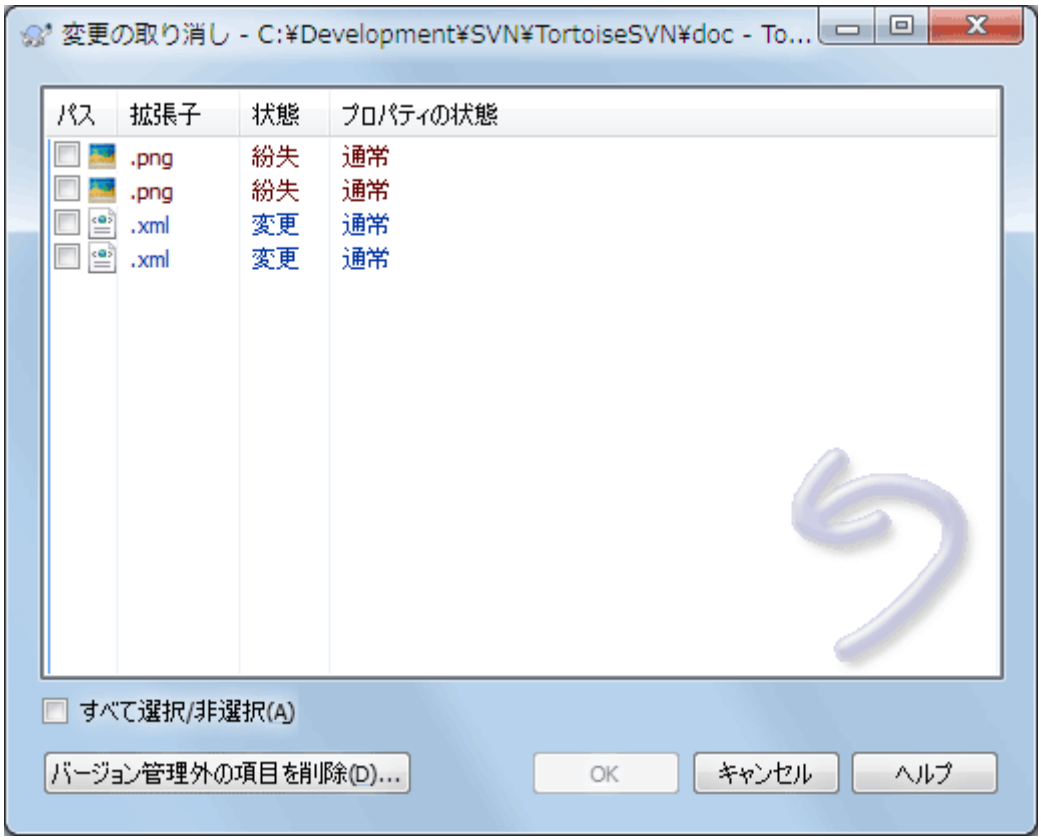


図4.31 変更の取り消しダイアログ

削除や名前変更の操作を元に戻す場合は、すでに削除した項目が存在しておらず右クリックできないので、親フォルダー上で変更の取り消し操作を行ってください。

追加した項目を元に戻す場合、コンテキストメニューに TortoiseSVN → **追加を取り消す...** が現れます。実際には変更の取り消しなのですが、わかりやすく名前が変わります。

ダイアログの中の列は、**変更をチェック** ダイアログの中の列と同じ方法でカスタマイズできます。詳細は「ローカルとリモートの状態」をご覧ください。

変更の取り消しが作業コピーをクリーンアップするために使用されることがあるので、バージョン管理外の項目を削除するためのボタンもあります。このボタンをクリックすると、バージョン管理外の項目がすべて一覧表示されるダイアログが表示され、削除する項目を選択することができます。

コミットした変更を元に戻す

変更の取り消し はローカルの変更しか取り消しできません。すでにコミットしてしまった変更は 取り消せない のです。あるリビジョンでコミットされた方法については、詳しくは「リビジョンログダイアログ」をご覧ください。

変更の取り消しが遅い

変更を取り消す際、予想よりも時間がかかると感じられるかも知れません。これは、間違えて変更の取り消しを実行したとき、変更したファイルを取り戻せるように、変更したファイルをごみ箱に送っているからです。しかし、ごみ箱がいっぱいになると、Windows はファイルを置く場所を探すのに長い時間がかかります。解決法は簡単です。ごみ箱を空にするか、TortoiseSVN の設定で **変更を取り消す際にごみ箱を使用する** のチェックを外してください。

4.16. クリーンアップ

サーバーの問題等で、 Subversion コマンドが正常に終了しなかった場合、作業コピーが矛盾した状態のままになってしまうことがあります。この場合、**TortoiseSVN → クリーンアップ** をフォルダーに対して実行する必要があります。作業コピーの最上層で行うのがいいでしょう。

クリーンアップダイアログではそれ以外にも、作業コピーを **クリーン** 状態にするために便利なオプションが用意されています。

作業コピーの状態のクリーンアップ
: 前述のように、このオプションは、矛盾した作業コピーを使用可能な状態にしようとします。これは、作業コピーのデータベースの内部状態にのみ影響し、それ以外のデータには影響しません。これは、以前の TortoiseSVN クライアントや他の SVN クライアントで**クリーンアップ**コマンドを実行した時と同じ動作です。

すべての変更を再帰的に取り消す
: このコマンドは、まだコミットされていないローカルの変更をすべて取り消します。

  【注意】TortoiseSVN → **変更の取り消し...** コマンドを使用したほうが、事前に元に戻すファイルを確認して選択することができるので便利です。

バージョン管理外のファイルやフォルダーを削除、無視するファイルやフォルダーを削除
: 作業コピーに生成されたファイルをすべて削除するのに、手っ取り早く簡単な方法です。バージョン管理されていないファイルやフォルダーは、すべてごみ箱に移動されます。

  【注意】TortoiseSVN → **変更の取り消し...** のダイアログを使用しても同じことができます。こちらはバージョン管理されていないファイルやフォルダーを一覧表示し、削除するファイルを選択することができます。

シェルのオーバーレイを更新
: ときどき、特にエクスプローラーの左側のツリービューでは、シェルのオーバーレイでは状態のキャッシュが変更を認識するのに失敗して、現在の状態が表示されなくなってしまうことがあります。このような場合、このコマンドで強制的に更新させることができます。

外部参照を含める
: チェックすると、 `svn:externals` プロパティに含まれているすべてのファイルやフォルダーにも、すべての処理が実行されます。

4.17. プロジェクト設定

4.17.1. Subversion のプロパティ

図4.32 Subversion のプロパティページ

Subversion のプロパティは、 Windows のプロパティダイアログから確認したり設定したりできますが、他に TortoiseSVN → プロパティ や コンテキストメニュー → プロパティ の TortoiseSVN の状態リストでも確認できます。

自分でプロパティを定義できるほか、 Subversion で特別な意味をもつプロパティを追加することもできます。これは `svn:` で始まります。`svn:externals` もその一種です。外部参照の扱い方は「外部項目」をご覧ください。

4.17.1.1. svn:keywords

Subversion は、ファイル自身にファイル名やリビジョン情報を埋め込む CVS 風のキーワード展開をサポートしています。現在、キーワードは次のものをサポートしています。

$Date$
: 最後にコミットされた日時です。作業コピーに対して更新を実行した時点の情報に基づいています。それより新しい情報をリポジトリに確認することは ありません 。

$Revision$
: 最後にコミットを行ったリビジョンです。

$Author$
: 最後にコミットを行ったユーザーです。

$HeadURL$
: このファイルのリポジトリ上のフル URL です。

Id
: 前述の4つのキーワードの短く組み合わせたものです。

以上のキーワードの使用法については、 Subversion book の svn:keywords section [http://svnbook.red-bean.com/en/1.7/svn.advanced.props.special.keywords.html] で、これらのキーワードの詳しい解説と、有効にする方法や使用する方法を説明をしています。

Subversion のプロパティについて詳しくは、 Special Properties [http://svnbook.red-bean.com/en/1.7/svn.advanced.props.html] をご覧ください。

4.17.1.2. プロパティの追加と編集

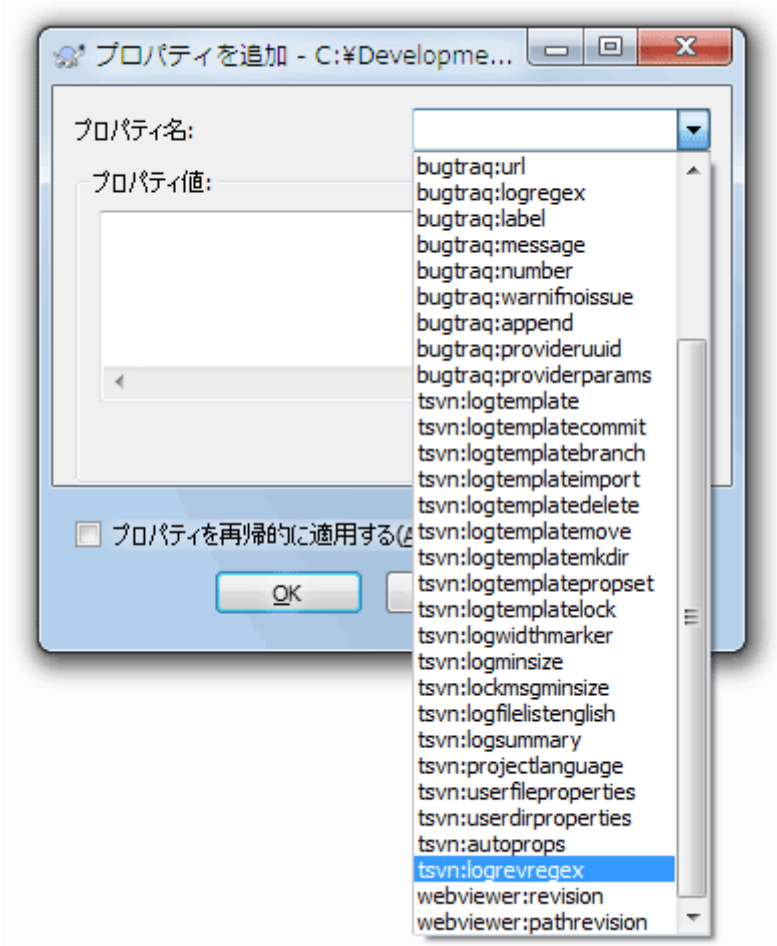


図4.33 プロパティの追加

新しいプロパティを追加するには、まず **新規...** をクリックしてください。メニューから必要なプロパティ名を選択し、プロパティダイアログに必要な情報を入力します。これらのプロパティダイアログの詳細は「プロパティエディター」をご覧ください。

独自のダイアログを持たないプロパティを追加するには、**新規...** メニューの高度な設定を選択します。そして、既存のプロパティをコンボボックスから選択するか、カスタムプロパティの名前を入力するかします。

一度にたくさんの項目に対してプロパティを設定する場合は、エクスプローラーでファイルやフォルダーを複数選択し、**コンテキストメニュー → プロパティ** を選択してください。

現在のフォルダー以下にある すべての ファイルやフォルダーに対してプロパティを適用する場合、**プロパティを再帰的に適用する** チェックボックスをチェックしてください。

既存のプロパティを編集する場合、プロパティの一覧から編集するプロパティを選択し、**編集...** をクリックしてください。

既存のプロパティを削除したい場合、プロパティの一覧から削除するプロパティを選択し、**削除** をクリックしてください。

`svn:externals` プロパティは、同じリポジトリからでも完全に異なるリポジトリからでも、他のプロジェクトを取得するのに使用できます。詳細は「外部項目」をご覧ください。

プロパティの編集は最新リビジョンに対して

プロパティはバージョン管理されているため、以前のリビジョンのプロパティを編集することはできません。ログダイアログや、リポジトリブラウザー上で最新以外のリビジョンを表示している場合は、プロパティと値の一覧は見られますが、編集はできません。

4.17.1.3. プロパティのエクスポートとインポート

しばしば、何度も同じプロパティのセット(例: `bugtraq:logregex`)を適用していることに気付くでしょう。あるプロジェクトから別のプロジェクトへ、プロパティを簡単にコピーする場合は、エクスポート・インポート機能を使用できます。

すでにプロパティが設定されているファイルやフォルダーで、TortoiseSVN → **プロパティ** を使用し、エクスポートしたいプロパティを選択して **エクスポート...** をクリックしてください。プロパティの名前と値を保存するファイル名を入力するよう促されます。

プロパティを適用したいフォルダーで、TortoiseSVN → **プロパティ** を実行し、**インポート...** をクリックしてください。インポート元のファイル名を尋ねてきますので、先ほど保存したエクスポートファイルを指定してください。プロパティはそのフォルダーに、非再帰的に追加されます。

プロパティをディレクトリ構造に対して再帰的に追加する場合は、前述の手順の後、プロパティダイアログでプロパティを選び、**編集...**をクリックしてください。**プロパティを再帰的に適用する** チェックボックスをチェックした後、OK をクリックしてください。

インポートファイルはバイナリで TortoiseSVN 専用です。インポートやエクスポートでのプロパティの受け渡し専用で、このファイルを編集する必要はありません。

4.17.1.4. バイナリプロパティ

TortoiseSVN は、ファイルを使用することでバイナリのプロパティ値を扱うことができます。バイナリプロパティ値を読むには、ファイルに **値を保存...** してください。バイナリ値を設定するには、バイナリエディター等の適切なツールを使用してファイルを作成し、そのファイルから **開く...** としてください。

バイナリプロパティはあまり使用されませんが、アプリケーションによっては便利です。例えば、巨大な画像ファイルを格納している場合や、アプリケーションが読み込むファイルが巨大な場合、プレビューを素早く得るのに、プロパティにサムネイルを格納した方が良いかもしれません。

4.17.1.5. プロパティの自動設定

ファイルやフォルダーをリポジトリに追加する際に、自動的にプロパティが設定されるように Subversion や TortoiseSVN を設定できます。これには2通りの方法があります。

Subversion の設定ファイルを編集すると、自分のクライアントでこの機能を有効にできます。TortoiseSVN の設定ダイアログの **一般** ページに、直接そこに行くための編集ボタンがあります。設定ファイルは、Subversion の動作を制御する、単純なテキストファイルです。ここで2か所を変更する必要があります。1つめは、`miscellany` という見出しのセクションで、`enable-auto-props = yes` という行のコメントを解除します。2つめは、その下にある、どのファイル形式に、どのプロパティを追加するかを定義したセクションを編集します。この方法は Subversion の標準機能で、どの Subversion クライアントでも動作します。しかし、各クライアントでそれぞれ定義しなければなりません。つまり、この設定をリポジトリに伝播する方法はありません。

もうひとつの方法は、`tsvn:autoprops` プロパティをフォルダーに設定する方法で、次節で説明しています。この方法は、TortoiseSVN クライアントでしか動作しませんが、更新を行った際に、すべての作業コピーに伝播します。

どちらの方法を選択しても、auto-props はファイルが作業コピーに追加された時にのみ適用されることに注意してください。auto-props は、すでにバージョン管理下にあるファイルのプロパティを変更することはありません。

新しいファイルに正しいプロパティが確実に適用されるようにしたい場合は、リポジトリに pre-commit を設定して、必要なプロパティが設定されていないとコミットを拒否するようにするべきです。

プロパティのコミット

Subversion のプロパティはバージョン管理されます。プロパティを変更したり追加したりした後は、変更をコミットする必要があります。

プロパティの競合

他のユーザーが同じプロパティを変更したりして、コミット時に競合が発生した場合、Subversion は `.prej` ファイルを生成します。競合を解決した後にこのファイルを削除してください。

4.17.2. TortoiseSVN のプロジェクトプロパティ

TortoiseSVN は自身が持つ特殊なプロパティをいくつか持っています。これは `tsvn:` で始まります。

- `tsvn:logminsize` はコミット時に入力するログメッセージの長さの最小値を設定します。ここで指定した長さより短いメッセージを入力するとコミットできません。この機能はコミットごとに適切に説明するメッセージを入力するのを忘れないようにするのに便利です。このプロパティが設定されていなかったり、値が 0 に設定されていたりした場合、空のログメッセージが許可されます。

  `tsvn:lockmsgminsize` はロックメッセージの長さの最小値を設定します。ここで指定した長さより短いメッセージを入力するとロックできません。この機能はロックするごとに適切に説明するメッセージを入力するのを忘れないようにするのに便利です。このプロパティが設定されていなかったり、値が 0 に設定されていたりした場合、空のロックメッセージが許可されます。

- `tsvn:logwidthmarker` は、プロジェクトでログメッセージが行の最大長（通常 80 文字）以内にそろえたい場合に使用します。このプロパティが 0 以外に設定されていると、ログメッセージ入力ダイアログでは、入力したテキストが長すぎないかどうかを確認できるように最大長を示すマーカーが設置され、表示時のワードラップが無効になります。【注】この機能はログメッセージに固定長フォントを選択していないと、正しく動作しません。

- `tsvn:logtemplate` は、プロジェクトでログメッセージの整形ルールを指定したい場合に使用します。このプロパティは複数行の文字列を保持し、コミット時にコミットメッセージボックスにそれが挿入されます。これにより必要な情報を持つコミットメッセージを編集できます。【注】`tsvn:logminsize` を併用する場合、必ずテンプレートより長い値を設定してください。そうでなければ保護機構が働きません。

  操作に固有のテンプレートを `tsvn:logtemplate` の代わりに使用することもできます。操作に固有のテンプレートが設定されている場合はそれが使用されますが、操作に固有のテンプレートが設定されていない場合は、`tsvn:logtemplate` が使用されます。

  操作に固有のテンプレートは次の通りです。

  - `tsvn:logtemplatecommit` は作業コピーからのすべてのコミットに使用されます。
  - `tsvn:logtemplatebranch` は、ブランチ/タグを作成する際、もしくはリポジトリブラウザーでファイルやフォルダーを直接コピーする際に使用されます。
  - `tsvn:logtemplateimport` は、インポート時に使用されます。

  - tsvn:logtemplatedelete は、リポジトリブラウザーで直接項目を削除する際に使用されます。
  - tsvn:logtemplatemove は、リポジトリブラウザーで名前変更や項目の移動を行う際に使用されます。
  - tsvn:logtemplatemkdir は、リポジトリブラウザーでディレクトリを作成する際に使用されます。
  - tsvn:logtemplatepropset は、リポジトリブラウザーでプロパティを変更する際に使用されます。
  - tsvn:logtemplatelock は、ロックを取得する際に使用されます。

- 新しくファイルの追加やインポートを行った際に、拡張子を元にプロパティを付加するように、 Subversion は「autoprops」 を設定できます。これはクライアントごとに、 Subversion 設定ファイルに適切に autoprops が設定されているかどうかに依存します。tsvn:autoprops をフォルダーに設定しておくと、インポートやファイル追加の際に、ユーザーのローカルに autoprops をマージするようになります。この形式は subversion の autoprops と同じで、.sh 拡張子を持つファイルに2つのプロパティを セットしたい場合は、*.sh = svn:eol-style=native;svn:executable のようになります。

  ローカルの autoprops と tsvn:autoprops が競合する場合は、プロジェクトごとに指定されている、プロジェクト設定を優先します。

- コミットダイアログでは、変更したファイルをファイルごとに状態(追加、変更、など)込みで貼りつけるオプションがあります。tsvn:logfilelistenglish は状態を英語で挿入するか、各国語で挿入するかを定義できます。このプロパティがセットされていなければ、デフォルトでは true となります。

- TortoiseSVN は OpenOffice や Mozilla で使われるスペルチェッカーモジュールを使用できます。スペルチェッカーをインストールしている場合、このプロパティでどのスペルチェッカーを使用するか決定します。つまり、どの言語でこのプロジェクトのログメッセージを書くかということでもあります。tsvn:projectlanguage では、ログメッセージ入力時にスペルチェックエンジンがどの言語でチェックを行うかを設定します。自分の言語の値は MSDN: Language Identifiers [http://msdn.microsoft.com/en-us/library/dd318693.aspx] で確認できます。

  この値を 10 進数や頭に 0xを頭に付けた 16 進数で入力できます。たとえば、日本の場合はは0x0411 や 1041 を入力してください。

- tsvn:logsummary プロパティは、ログメッセージのサマリとしてログダイアログに表示する、ログメッセージの一部を抽出するのに使用します。

  tsvn:logsummary プロパティの値は、正規表現グループを含む正規表現文字列を、1 行で表したものでなければなりません。そのグループにマッチするものであれば、なんでもサマリとして扱います。

  例: ¥[SUMMARY¥]:¥s+(.*) は、ログメッセージ中の「[SUMMARY]」以降をすべてサマリとして扱います。

- tsvn:logrevregex プロパティには、ログメッセージにあるリビジョンの参照にマッチする正規表現を定義します。これはログダイアログで、リビジョンの参照をリンクにするのに使用します。このリンクをクリックすると、そのリビジョンまでスクロール(リビジョンがログダイアログにあるか、ログキャッシュにある場合)したり、そのリビジョンを表示する、新しいログダイアログを開いたりします。

  正規表現は、リビジョン番号だけでなく参照全体にマッチしなければなりません。リビジョン番号はマッチした参照文字列から、自動的に抽出します。

  このプロパティが設定されていない場合、リビジョン参照をリンクにする、デフォルト正規表現を使用します。

- 新しいプロパティを加える際に、コンボボックスから選択するか、任意のプロパティ名を入力できます。プロジェクトでカスタムプロパティを使用し、そのプロパティをコンボボックスに表示する(プロパティ名の入力ミスを防ぐ)場

合、tsvn:userfileproperties や tsvn:userdirproperties でカスタムプロパティを作成できます。このプロパティはフォルダーに適用してください。そのフォルダー以下でファイルを作成すると、定義したプロパティ名ごとにカスタムプロパティを表示します。

TortoiseSVN はいくつかのバグ追跡ツールと統合できます。このとき bugtraq: で始まるプロジェクトプロパティを使用します。詳細情報は「バグ追跡ツール／課題追跡システムとの統合」をご覧ください。

TortoiseSVN は、webviewer: で始まるプロジェクトプロパティを使用して、いくつかの Web ベースのリポジトリブラウザーとも統合できます。詳細は「Web ベースのリポジトリビューアーとの統合」をご覧ください。

フォルダーへのプロジェクトプロパティの設定

これらのプロパティは、作業するシステムのフォルダーに設定しなければなりません。これらのプロパティを使用する TortoiseSVN のコマンドを使用するときは、クリックしたフォルダーからプロパティが読み取られます。フォルダーにプロパティが設定されていない場合、プロパティはフォルダー階層を上位に向かって、バージョン管理外フォルダーか、ツリーのルート(例: C:¥)に到達するまで検索されます。すべてのユーザーが、例えば trunk/ からチェックアウトし、サブフォルダーからチェックアウトしないことが確認できるのであれば、 trunk/ にそのプロパティを設定するだけで充分です。確認できない場合、各サブフォルダーに対し再帰的にプロパティを設定するべきです。もしプロジェクトの異なる階層に、同じプロパティで異なる値を設定した場合は、クリックしたフォルダの階層に依存して結果が変わります。

プロジェクトプロパティ(例:tsvn:、bugtraq:、webviewer:)に限っていえば、 **プロパティを再帰的に適用する** チェックボックスを使うと、そのフォルダ以下のすべてのサブフォルダーにプロパティを設定しますが、ファイルには設定しません。

TortoiseSVN で新しいサブフォルダーを作業コピーに追加する際、親フォルダーに設定されているプロジェクトプロパティは、新しいサブフォルダーにも自動的に追加されます。

リポジトリブラウザーを使用する際の制限

リモートからプロパティを取得するのは時間がかかります。そのため、上記の機能のうちいくつかは、作業コピーで使用しているリポジトリブラウザーでは動作しません。

- リポジトリブラウザーを使用してプロパティを追加する場合、標準的なliteral>svn:
- リポジトリブラウザーを使用して、プロパティを再帰的に設定もしくは削除することは出来ません。
- 子階層のフォルダーがリポジトリブラウザーで追加された際に、プロジェクトのプロパティは自動で広まることはありません。
- tsvn:autopropsは、リポジトリブラウザーで追加されたファイル上のプロパティを設定しません。

注意

TortoiseSVN のプロジェクトプロパティは非常に便利ですが、TortoiseSVN でしか動作せず、またいくつかは TortoiseSVN の新しいバージョンでしか動作しません。プロジェクトのメンバが様々な Subversion のクライアントを使用している場合や、古い TortoiseSVN しか持てない場合、プロジェクトポリシーを強制するのに、リポジトリフックを使用することになるでしょう。プロジェクトプロパティは、ポリシー実装の補助にしかならず、強制することはできません。

4.17.3. プロパティエディター

一部のプロパティは、自動処理に使用されるため、特定の値や特定の書式にする必要があります。正しい書式を設定やすくするため、 TortoiseSVN では幾つかの特定のプロパティについて、設定可能な値を表示したり、独立コンポーネントに分配したりするための編集ダイアログを提供しています。

4.17.3.1. 外部参照項目

図4.34 svn:externals プロパティページ

`svn:externals` プロパティは、同じリポジトリもしくは完全に異なるリポジトリから、他のプロジェクトを取得するのに使用できます。詳細は「外部項目」をご覧ください。

チェックアウトした外部フォルダーや、外部項目の Subversion の URL のサブフォルダーの名前を設定する必要があります。外部フォルダーの最新リビジョンをチェックアウトしておけば、外部項目がリポジトリの中で変更されたときに、作業コピーを更新させることができます。しかし、特定の安定した時点のものを外部で参照できるようにするには、使用する特定のリビジョンを指定することもできます。また、この場合、ペグリビジョンと同じリビジョンを指定したいかもしれません。外部項目が将来何らかの時点で改名されると、Subversionは作業コピーでこの項目を更新できなくなります。ペグリビジョンを指定することで、最新版ではなくペグリビジョンの時の名前をSubversionに検索させることができます。

4.17.3.2. SVN キーワード

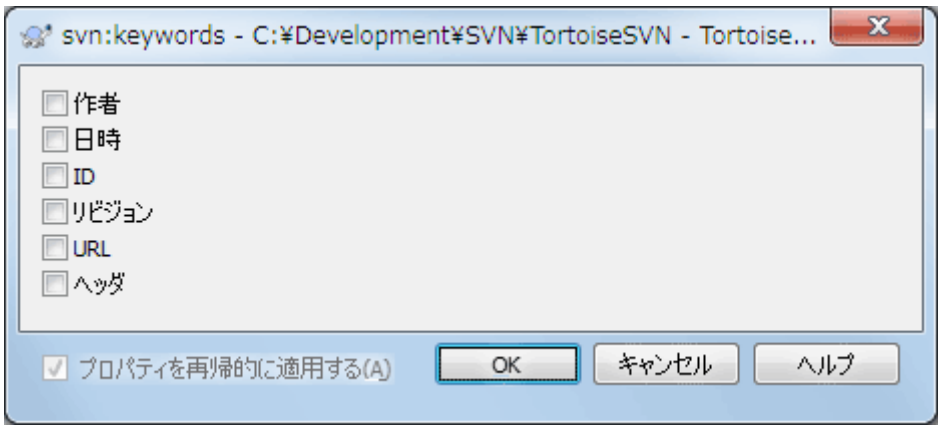


図4.35 svn:keywords プロパティページ

ファイル内に展開したいキーワードを選択してください。

4.17.3.3. 改行コード

図4.36 svn:eol-style プロパティページ

使用したい行末のスタイルを選択します。それで、TortoiseSVN は適切なプロパティ値を使用することができます。

4.17.3.4. 課題追跡システムとの統合

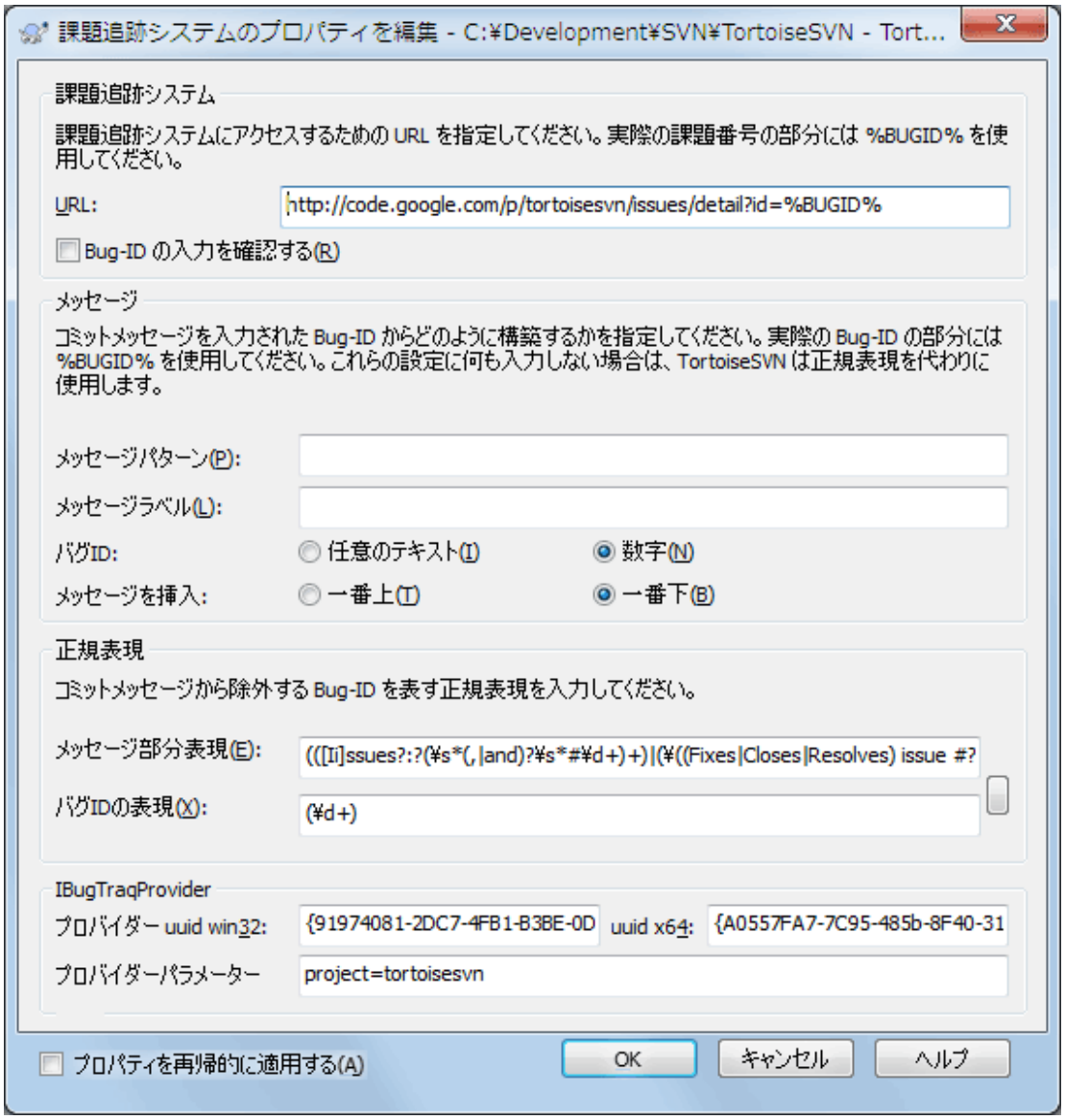


図4.37 tsvn:bugtraq プロパティページ

4.17.3.5. ログメッセージの大きさ

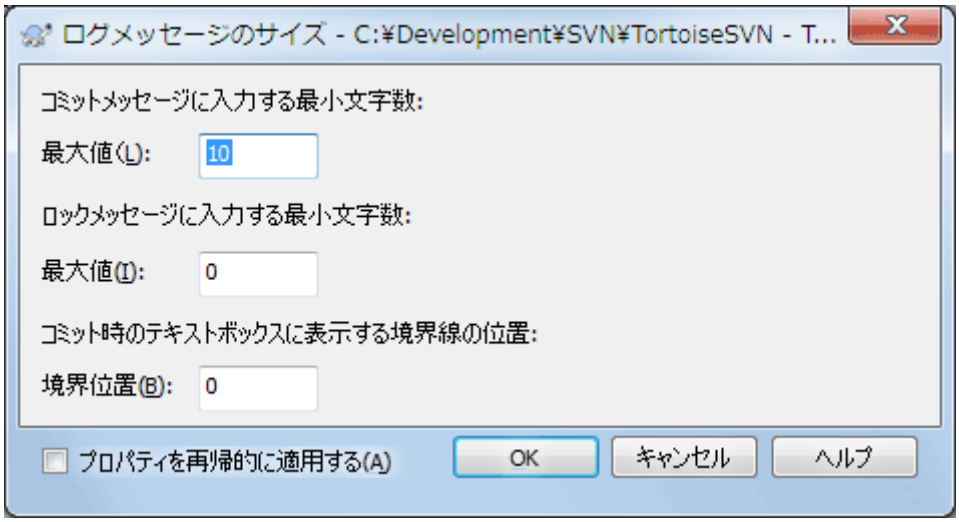


図4.38 ログメッセージサイズのプロパティページ

これら3つのプロパティは、ログメッセージの書式を制御します。最初の2つを設定すると、コミットダイアログやロックダイアログで、メッセージが最小文字列の長さに達するまで OK ボタンが無効になります。境界の位置は、それらのログメッセージの幅制限が設定されているプロジェクトのためのガイドとして指定されたカラム幅でマーカーを示しています。値をゼロに設定すると、プロパティが削除されます。

4.17.3.6. プロジェクトの言語

図4.39 言語プロパティページ

コミットダイアログのログメッセージのスペルチェックに使用する言語を選択してください。右のログメッセージのペインでクリックして選択すると、ファイルリストのチェックボックスが有効になりますので、**ファイル名のリストを貼付** を選択してください。デフォルトでは Subversion の状態が、既定の言語で表示されます。このボックスがチェックされるとステータスが常に英語のみになります。英語のログメッセージを必要とするプロジェクトで有効です。

4.17.3.7. MIMEタイプ

図4.40 svn:mime-type プロパティページ

4.17.3.8. svn:needs-lock

図4.41 svn:needs-lock プロパティページ

このプロパティは、作業コピーがロックされていない場合、ファイルを読み取り専用の状態でチェックアウトするかどうかを制御します。

4.17.3.9. svn:executable

図4.42 svn:executable プロパティページ

このプロパティは、Unix / Linux システム上で、チェックアウト時にファイルに実行属性を付与するかどうかを指定します。これは、Windows でチェックアウトする場合には影響しません。

4.18. 外部項目

時には、たくさんの異なるチェックアウトから作業コピーを構成するのは、便利なことがあります。たとえば、リポジトリ内の異なる場所もしくは異なるリポジトリにある、異なるサブディレクトリが必要になるかもしれません。全てのユーザーが同じレイアウトを保持するには、`svn:externals` プロパティを設定し、必要なところから指定したリソースを取得します。

4.18.1. 外部フォルダー

さて、ここで/project1の作業コピーをD:¥dev¥project1にチェックアウトとするとしましょう。D:¥dev¥project1フォルダーを選択して右クリックし、コンテキストメニューからWindows メニューの → プロパティを選択します。プロパティダイアログが表示されます。次に Subversion タブに移動すると、プロパティを設定することができます。プロパティ... をクリックして表示されるダイアログで、svn:externalsが既に存在すればそれをダブルクリックし、そうでなければ新規... ボタンを押し、メニューからsvn:externalsを選択します。新たな外部項目を追加するには、新規... ボタンをクリックしてから、表示されるダイアログで必要な情報を入力します。

注意

正しく動作させるためには、URLを適切にエスケープする必要があります。例えば、空白文字は%20に置き換えなければなりません。

ローカルのパスに空白や特殊文字を使用したい場合、二重引用符で囲ったり、Unix シェル形式のエスケープ文字 ¥ (バックスラッシュ) を特殊文字の前に置いてください。もちろんこれは、パス区切り文字に /(スラッシュ)を使わなければならない、ということでもあります。この挙動は Subversion 1.6 の新機能で、それ以前のクライアントでは動作しないことに注意してください。

リビジョン番号を指定する

上に示したように、すべての外部参照定義について、リビジョン番号を指定することを強く意識するべきです。そうすることは、異なる外部参照情報のスナップショットを取り出す際、正確にどのスナップショットを取り出すかを決めることを意味します。コントロールが及ばないサードパーティのリポジトリに変更があっても驚かないと言う常識に加え、リビジョン番号を指定するということは、作業コピーを以前のリビジョンに戻すということでもあります。外部参照定義はまた、以前のリビジョンを見るように元に戻り、リポジトリが過去のリビジョンになっているなら、外部参照の作業コピーは一致するように以前を参照するように更新されます。ソフトウェアプロジェクトにおいて、これは複雑な古いコードベースを構築するのが成功するか失敗するかといおった違いに現れます。

外部プロジェクトが同じリポジトリにある場合、メインプロジェクトの変更をコミットすると、変更がコミットリストに含まれます。

外部プロジェクトが別のリポジトリにある場合、メインプロジェクトのコミット時に、外部プロジェクトに対する変更は通知されますが、別々にコミットする必要があります。

svn:externals の定義に絶対 URL を使用し、作業コピーを再配置しなければならない(つまり、リポジトリの URL を変更する)場合、外部参照は変化せず、もう動作しないかも知れません。

このような問題を避けるため、Subversion クライアントバージョン 1.5 以降では、相対外部参照 URL をサポートします。相対 URL を指定する4つの異なる方法をサポートしています。以下の例では、2つのリポジトリ(http://example.com/svn/repos-1 と http://example.com/svn/repos-2)があると仮定します。C:¥Working に http://example.com/svn/repos-1/project/trunk のチェックアウトがあり、トランクに svn:externals プロパティをセットしています。

親ディレクトリへの相対パス

この URL は、以下の例のように、常に ../ という文字列で始まります。

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

これは、C:¥Working¥common¥foo-widget へ http://example.com/svn/repos-1/widgets/foo を抽出します。

URL が、ディスクに書かれている外部参照のディレクトリではなく、svn:externals プロパティにあるディレクトリの URL への相対パスであることに注意してください。

リポジトリのルートへの相対パス
この URL は、以下の例のように、常に ^/ という文字列で始まります。

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

これは、C:¥Working¥common¥foo-widget へ http://example.com/svn/repos-1/widgets/foo を抽出します。

同じ SVNParentPath(複数のリポジトリを保持する共通ディレクトリ)にある他のリポジトリに、容易に参照できます。例は以下のようになります。

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

これは、C:¥Working¥common¥claw-hammer へ http://example.com/svn/repos-2/hammers/claw を抽出します。

スキームへの相対パス
// で始まる URL は URL のスキーム部のみをコピーします。これは同じホスト名に対して、ネットワークの場所によって異なるスキームでアクセスしなければならない場合に便利です。例えば、インターネットにあるクライアントは http:// を使用するのに、外部クライアントは svn+ssh:// を使用するということです。以下に例を挙げます。

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

これは、C:¥Working をチェックアウトするのに使用した方法により、http://example.com/svn/repos-1/widgets/foo か svn+ssh://example.com/svn/repos-1/widgets/foo を抽出します。

サーバーのホスト名への相対パス
/ で始まる URL は URL のスキーム部とホスト名部をコピーします。以下に例を挙げます。

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

これは C:¥Working¥common¥foo-widget に http://example.com/svn/repos-1/widgets/foo を抽出します。しかし、svn+ssh://another.mirror.net/svn/repos-1/project1/trunk というように別のサーバーから作業コピーをチェックアウトすると、外部参照は svn+ssh://another.mirror.net/svn/repos-1/widgets/foo を抽出します。

また、必要に応じて、URLのペグリビジョンを指定することができます。

TortoiseSVN がプロパティをどのように扱うかについての詳細な情報は、「プロジェクト設定」を参照してください。

共通サブプロジェクトへの他のアクセス方法については、「共通のサブプロジェクトを含める」を参照してください。

4.18.2. 外部ファイル

Subversion 1.6 では、フォルダーと同じ文法を用いて、単一ファイルの外部項目を作業コピーに追加できます。しかし、いくつか制限事項があります。

- 外部ファイルへのパスは、既存のバージョン管理下にあるフォルダーにファイルを配置しなければなりません。一般的に、svn:externals がセットされているフォルダーに、直接ファイルを配置するのがほとんどだと思いますが、バージョン管理下のサブフォルダーにすることもできます。対照的に、外部ディレクトリは、必要に応じてバージョン管理外の中間サブフォルダーを、自動的に作成します。

- 外部ファイルの URL が、外部ファイルを挿入する URL と同じリポジトリになければなりません。つまり、リポジトリ間の外部ファイルはサポートしていません。

外部ファイルの挙動は、あらゆる点でその他のバージョン管理下のファイルと同等ですが、通常のコマンドでは、移動・削除ができません。代わりに svn:externals プロパティを変更しなければなりません。

4.19. ブランチ/タグの作成

バージョン管理システムの特徴のひとつに、開発の別のラインに変更を隔離できることがあります。このラインは ブランチ と呼ばれています。ブランチは、開発のメインラインにコンパイルエラーやバグで混乱させることなく、新機能のを試験するために使用されます。新機能が充分に安定したら、開発ブランチをメインブランチ(トランク)に マージ して書き戻します。

バージョン管理システムのもうひとつの特徴として、リリースしたリビジョンなど特定のリビジョンをマークする機能があります。このため、いつでも確実にビルドや環境を再作成できます。このプロセスは タグ付け と呼ばれています。

Subversion には、ブランチやタグを作成するための特別なコマンドはありませんが、代わりに「簡易コピー」と呼ばれるものを使用します。簡易コピーは、Unix のハードリンクと似ています。つまり、リポジトリの完全なコピーを作成する代わりに、指定したツリーやリビジョンを指す内部リンクを作成します。そのため、ブランチやタグの作成は、非常に高速で、リポジトリに追加スペースをほとんど使用しません。

4.19.1. ブランチ/タグの作成

プロジェクトを標準的なディレクトリ構成でインポートしているなら、ブランチ/タグバージョンを作成するのはとても簡単です。

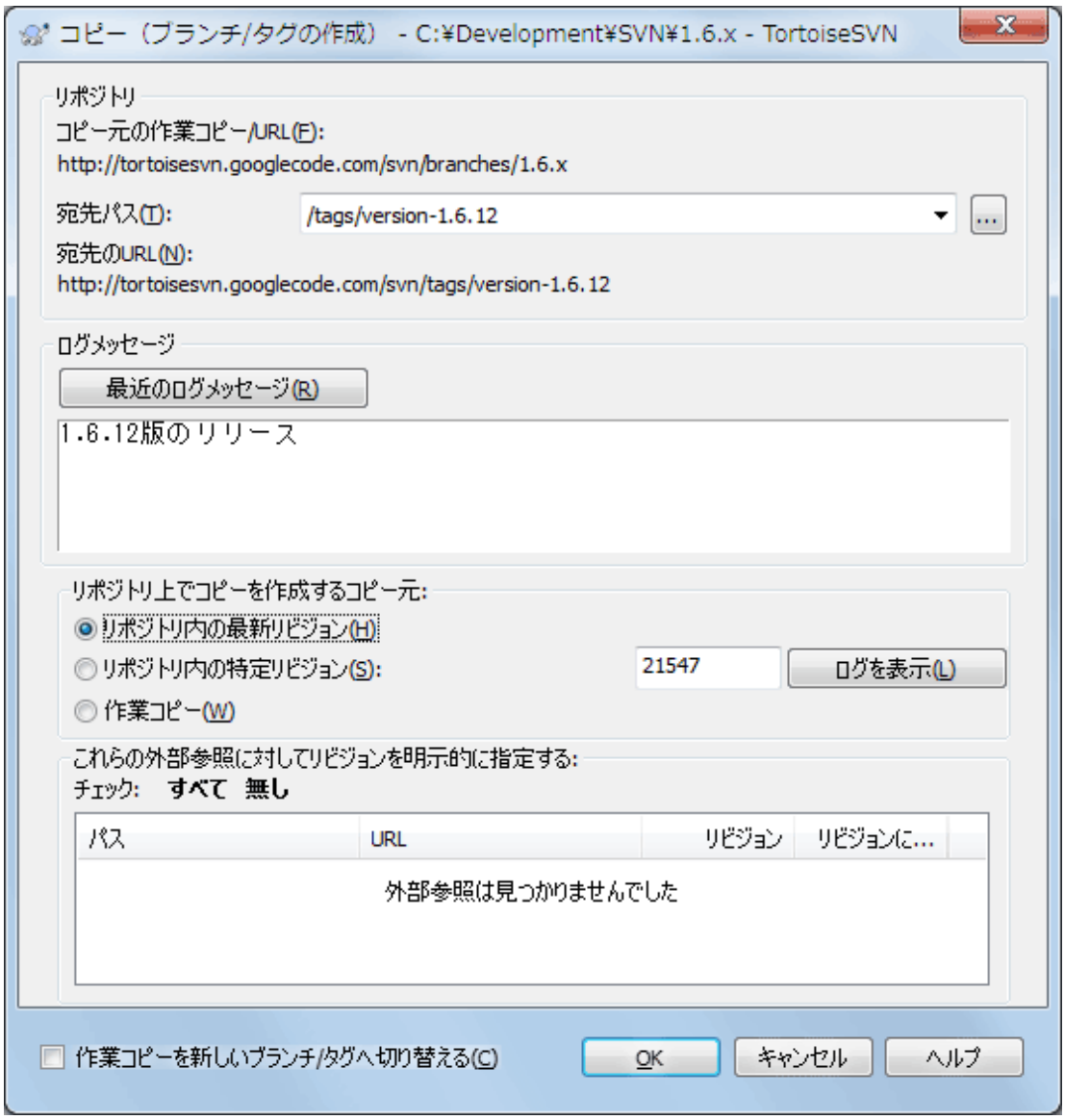


図4.43 ブランチ/タグの作成ダイアログ

ブランチやタグにコピーしたい作業コピーのフォルダーを選択してから、TortoiseSVN → ブランチ/タグの作成... コマンドを実行してください。

新しいブランチのコピー先 URL のデフォルトは、作業コピーの基準になったコピー元 URL になっています。ブランチ/タグの URL を新しいパスに変更してください。つまり、

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

の代わりに、

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

のようにするということです。前回使用した命名規則を思い出せなければ、右にあるボタンをクリックすれば、リポジトリブラウザーが開かれ、既存のリポジトリの構造を確認することができます。

そして、コピーの作成元を選択してください、ここでは3つの選択肢があります。

リポジトリ内の最新リビジョン
: 新しいブランチは、リポジトリの最新である最新リビジョンから直接コピーされます。データを作業コピーから転送する必要がないため、ブランチは非常に高速に作成されます。

リポジトリ内の特定リビジョン
: 新しいブランチは直接リポジトリからコピーされますが、古いリビジョンを指定できます。先週プロジェクトをリリースした際に、タグを作り忘れたときなどに便利です。リビジョン番号を思い出せなければ、右のボタンを押すとリビジョンログが表示されるので、そこからリビジョン番号を選択してください。こちらも作業コピーから転送されるデータはなく、ブランチは非常に高速に作成されます。

作業コピー
: 新しいブランチは、ローカルの作業コピーと同一になります。作業コピーを以前のリビジョンへ更新したり、ローカルで変更を行ったりすると、これがコピーに取り込まれます。当然、そのような複雑なタグを作成する場合、リポジトリにまだ反映されていないデータは、作業コピーからリポジトリに転送されます。

作業コピーを新しく作成したブランチに自動的に切り替えたい場合、**作業コピーを新しいブランチ/タグへ切り替える** チェックボックスをチェックしてください。しかしそうするには、まず作業コピーに変更が含まれていないことを確認してください。もし含まれていると、切り替えたときにブランチの作業コピーに変更点がマージされていまいます。

もし、`svn:externals` プロパティが指定され、作業コピーに他のプロジェクトが含まれていると、外部項目がブランチ/タグダイアログの下部に表示されます。各外部項目とそのターゲットパス、ソースURL、リビジョンが表示されます。外部項目のリビジョンは、作業コピーから決定されます。つまり、外部項目が明示的に指している先のリビジョンです。

新しいタグを常に一貫した状態にするためには、すべての外部項目の持つリビジョンが、現在の作業コピーのリビジョンで固定されていることを確認する必要があります。外部項目を確認せずにいると、将来 HEAD リビジョンが更新されたときに、新しいタグをチェックアウトすると、外部項目はその時点での HEAD リビジョンが取得されてしまい、コンパイルに失敗する可能性があります。そのため、タグを作成するときには、常に外部項目に明示的なリビジョン番号を設定することをお勧めします。

ブランチやタグを作成するとき、外部項目に明示的にリビジョンが設定されていた場合、 TortoiseSVN は自動的に `svn:externals` プロパティを更新します。リポジトリの HEAD や特定のリビジョンからブランチやタグが作成された場合、TortoiseSVN はまずブランチやタグを作成し、プロパティを調整します。そしてそれぞれのプロパティを追加コミットします。ブランチやタグが作業コピーから作成された場合は、先にプロパティを更新し、それからブランチやタグを作成し、そしてプロパティ値を元の値に戻します。

リポジトリに新しいコピーをコミットするには **OK** を押してください。ログメッセージを書くのを忘れないでください。コピーは リポジトリの内部 で行われることに注意してください。

作業コピーを新しく作成したブランチに切り替えなければ、ブランチやタグを作成しても、作業コピーには影響を 与えない ことに注意してください。作業コピーからブランチを作成したとしても、その変更はトランクではなく新しいブランチにコミットされます。そのため、作業コピーのマークは、変更されたままになっているかもしれません。

4.19.2. ブランチやタグを作成するその他の方法

作業コピーがなくてもブランチやタグを作成する方法があります。リポジトリブラウザーを開き、フォルダーを新しい場所にドラッグします。 Ctrl キーを押しながらドラッグするとコピーになり、そうでなければフォルダーはコピーされずに移動します。

マウスの右ボタンでフォルダーをドラッグする方法もあります。マウスのボタンを離すと、コンテキストメニューでフォルダーを移動するかコピーするかを選択することができます。もちろん、ブランチやタグを作成するときは、フォルダーを移動ではなくコピーしてください。

ログダイアログから行う方法もあります。例えばトランクの、特定のリビジョン（HEAD リビジョンでも古いリビジョンでも良い）のログダイアログを表示し、右クリックして **リビジョンからブランチ/タグを作成** を選択します。

4.19.3. チェックアウトするか切り替えるか...

……それが（それほどではありませんが）問題です。チェックアウトは、リポジトリの指定したブランチから作業コピーへすべてダウンロードしますが、TortoiseSVN → **切り替え...** は変更のあったデータのみを作業コピーに転送します。ネットワーク負荷をとるか、忍耐力をとるかですね。:-)

新しく作成したブランチやタグで作業するには、いくつかの方法があります。次のように行ってください。

- TortoiseSVN → **チェックアウト** は空のフォルダーに新鮮なチェックアウトを行います。ローカルディスクのどこにもチェックアウトでき、お好みにあわせ、いくつでもリポジトリから作業コピーを作成できます。
- 現在の作業コピーを、リポジトリに新しく作成したコピーに切り替えます。再びプロジェクトの最上位フォルダーを選択し、コンテキストメニューから TortoiseSVN → **切り替え...** を使用してください。

  続くダイアログで、たった今作成したブランチの URL を入力してください。**HEAD リビジョン** ラジオボタンを選択し、**OK** をクリックしてください。作業コピーが新しいブランチ/タグに切り替えられます。

  切り替えは、ローカルの変更を決して破棄しないので、ちょうど更新のように動作します。まだコミットしていない作業コピーへの変更は、切り替えるとマージされます。こうして欲しくなければ、切り替える前にコミットをしておくか、作業コピーの変更を取り消し、既にコミットしてあるリビジョン（通常は HEAD リビジョン）まで戻さなければなりません。
- トランクやブランチで作業したくても、まっさらなチェックアウトに時間を費やしたくない場合、トランクをチェックアウトした作業コピーを Windows エクスプローラーでコピーし、TortoiseSVN → **切り替え...** を使用して、新しいブランチに切り替える方法もあります。

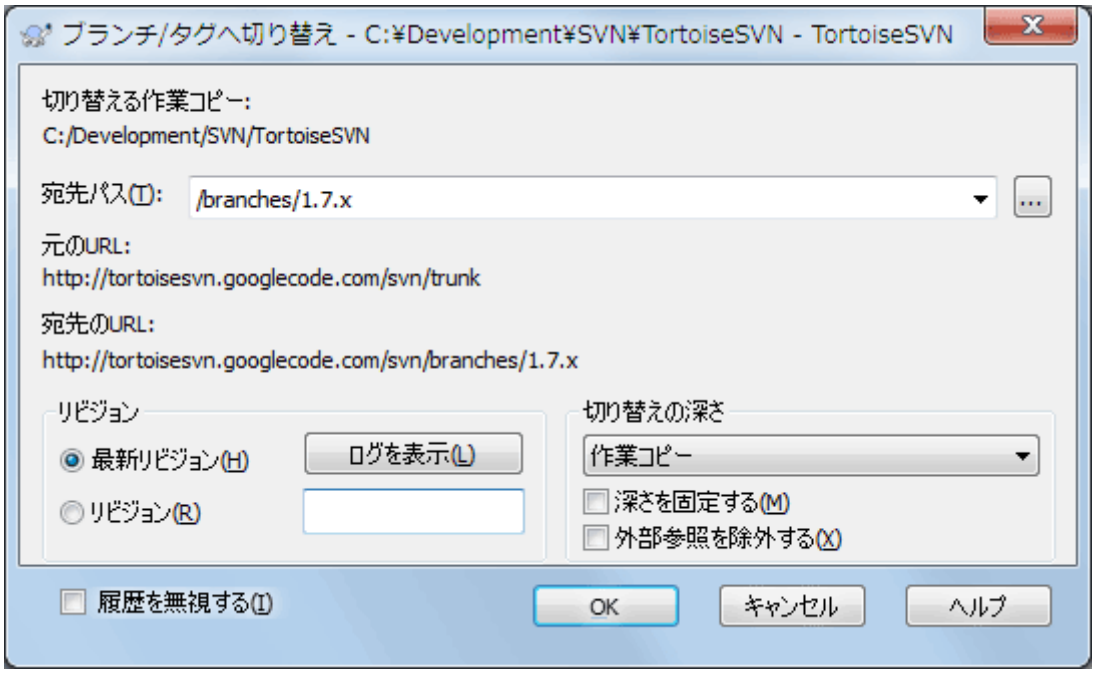


図4.44 切り替えダイアログ

Subversion 自身はタグとブランチを区別しませんが、通常は以下のように少し異なる使い方をします。

- タグは通常、プロジェクトの特定の場面の静的なスナップショットとして作られます。通常は開発には使用しません（それはブランチの役割であり、最初に `/trunk` `/branches` `/tags` というリポジトリ構造を推奨する理由です）。タグのリ

ビジョンで作業することは よくありません が、ローカルファイルが書き込み保護されていないので、間違った操作を止める方法はありません。但し、リポジトリ内の /tags/ 以下のパスにコミットしようとした場合、TortoiseSVN は警告するようになっています。

- 既にタグ付けしたリリースに、変更を加える必要が発生することもあります。これを扱う正しい方法は、タグから新しいブランチを作成し、そのブランチをコミットすることです。変更をこのブランチに行い、その後、新しいブランチから Version_1.0.1 といったように、新しいタグを作成してください。

- ブランチから作成した作業コピーを変更し、コミットする場合、すべての変更は新しいブランチに行われ、トランクには行われません 。変更点のみ格納されます。残りは簡易コピーのままになります。

4.20. マージ

ブランチを開発の独立したラインでメンテナンスするのに使用する所では、何らかの段階で、ブランチに対して行われた変更をトランクにマージしたくなるでしょう。逆もまた然りです。

Subversion 内でどのようにブランチやマージを作成しているるかを、はじめる前に理解しておくのは（かなり複雑になってしまいますが）重要です。Subversion book の Branching and Merging [http://svnbook.red-bean.com/en/1.7/svn.branchmerge.html] の章を読むのを強くお勧めします。ここには詳しい説明とたくさんの使用例があります。

注意する次のポイントは、マージは 常に 作業コピーで行われるということです。ブランチの中に、変更をマージしたい場合、ブランチの作業コピーをチェックアウトして、その作業コピーで TortoiseSVN → マージ... を行い、マージウィザードを起動しなければなりません。

一般的には、マージは変更されていない作業コピーに対して行うのが良いでしょう。作業コピーに何か他に変更が加えられているなら、まずコミットしてください。思うようにマージできないのなら、変更の取り消しを行う必要があるかもしれません。 **変更の取り消し** コマンドは、マージする前に行った変更を すべて 取り消してしまいます。

マージを行うには、以下で説明するような、少し異なる方法で行う、3つのよくあるケースがあります。マージウィザードの最初のページで、どちらで行うか確認しますので、選択してください。

リビジョンの範囲をマージ
: これは、1つ以上のリビジョンがブランチ（もしくはトランク）に存在し、それらの変更を別のブランチに反映したいような場合に使用します。

  Subversion に以下のことを行うように指示を出しています。「ブランチ A のリビジョン 1 [FROM] から、ブランチ A のリビジョン 7 [TO] までの必要な変更点を計算し、(トランクやブランチ B の) 作業コピーに変更点を適用する」

ブランチの再統合
: この方法は、Subversion book で述べられている機能ブランチを作成する際についてを扱います。トランクのすべての変更を、週ごとに機能ブランチに移し、機能が完全になったらトランクにマージします。機能ブランチをトランクと同期させておくため、ブランチの最新バージョンとトランクは、ブランチへの変更を除いて、完全に一致するはずです。

  これは以下に述べるツリーのマージの特殊なケースで、(通常) 開発ブランチからマージする URL のみが必要です。これは使用する正しいリビジョン範囲を計算するのに、Subversion のマージ追跡機能を使用し、トランクの変更でブランチを完全に更新できたかを確認するのに、追加チェックを使用します。これは自分が最後に同期を取ってから、他の誰かがトランクにコミットした作業を、偶然元に戻していないことを確認します。

  マージ後には、全ブランチでの成果物が、メインの開発ラインに完全にマージされます。このブランチは冗長になり、削除できます。

一度再統合によるマージを行うと、このブランチを開発に使用し続けるべきではありません。というのは、既存ブランチをトランクに再同期しようとする場合に、マージ追跡機能は再統合を、まだブランチへマージされていないトランクの変更として認識し、ブランチからトランクへのマージをブランチに対して行います! これを解決するには、単純にトランクから新しいブランチを作成し、次のフェーズの開発を続けることです。

異なる2つのツリーをマージ

これは、再統合する方法のもっと一般的なケースで、Subversion に対して、「トランクの HEAD リビジョン(開始点)からブランチの HEAD リビジョン(終了点)までの必要な変更点を計算し、(トランクの)作業コピーに変更点を適用しなさい」と指示します。結果的に、トランクがまさしくブランチと同様になっているということになります。

サーバー・リポジトリが、マージ追跡をサポートしていない場合、これはブランチをトランクにマージする唯一の方法になります。別のケースとしては、ベンダブランチを使用していて、新しくベンダから落としたコードの変更を、トランクにマージする必要がある場合もありえます。詳細は、Subversion Book の vendor branches [http://svnbook.red-bean.com/en/1.7/svn.advanced.vendorbr.html] をご覧ください。

4.20.1. リビジョン範囲のマージ

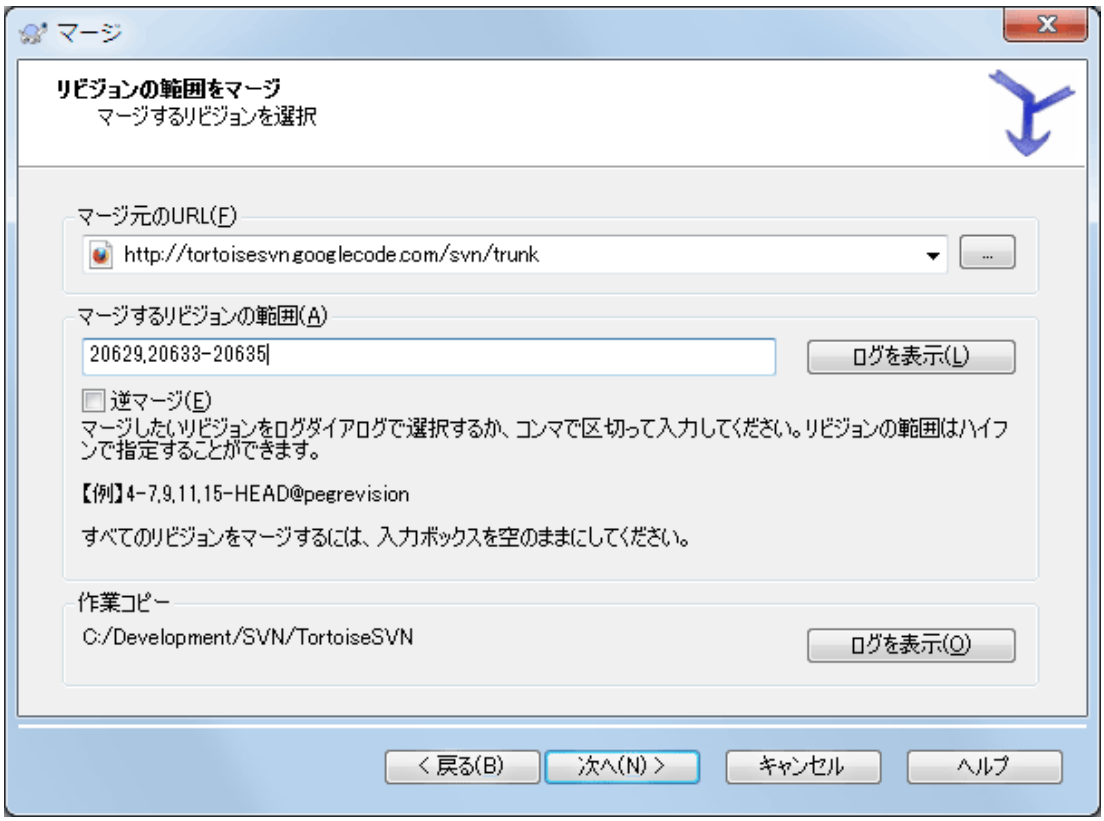


図4.45 マージウィザード - リビジョン範囲の選択

マージ元のURL 欄には、作業コピーに取り込みたい変更があるブランチやタグのフォルダーの URL をすべて入力してください。... をクリックしてリポジトリを閲覧し望みのブランチを探すこともできます。以前このブランチからマージしたことがあるのであれば、以前使用されたURLの履歴のドロップダウンリストから選択するだけです。

マージするリビジョンの範囲 フィールドに、マージするリビジョンのリストを入力してください。ここではリビジョン1つ、コンマで区切ったリビジョンの指定、ダッシュでつないだリビジョンの範囲と以上の組み合わせが使用できます。

マージ時にペグリビジョンを特定する必要がある場合、ペグリビジョンをリビジョン番号の最後に、`5-7,10@3` のように追加してください。この例では、リビジョン5、6、7、10とペグリビジョンの3がマージされます。

重要

TortoiseSVN とコマンドラインクライアントを比較すると、リビジョン範囲を指定する方法に重要な違いがあります。これを思い浮かべる簡単な方法は、フェンスの柱と、フェンスの板について考えることです。

コマンドラインクライアントでは、前 と 後 で指定した 2 本の「フェンスの柱」のリビジョンを使用して、マージする変更を指定します。

TortoiseSVN では、「フェンスの板」を使用して、マージする変更セットを指定します。マージするためのリビジョンを指定するログダイアログを使用する場合、チェンジセットとしてどこに各リビジョンが現れるか、明白になるためです。

かたまりとしてリビジョンをマージしていると、subversion book にある方法では、今回 100-200 をマージし、次回に 200-300 をマージすることになります。TortoiseSVN では、今回 100-200 をマージし、次回に 201-300 をマージします。

この違いは、メーリングリストでたくさんの論争を巻き起こしてきました。私たちは、コマンドラインクライアントと違うことを認めます。しかし、大多数の GUI ユーザーにとって、私たちが実装した方法の方が理解しやすいと信じています。

必要なリビジョンの範囲を選択する最も簡単な方法は、**ログを表示** をクリックして、最近の変更点一覧をログメッセージと共に表示することです。単一リビジョンからマージしたい場合は、そのリビジョンを選択するだけです。複数のリビジョンからマージしたい場合は、その範囲を（通常 Shift キーを押しながら）選択してください。**OK** をクリックすると、マージするリビジョン番号のリストに入力されます。

すでにコミットしてしまった変更を取り消して、作業コピーを元に戻すようにマージする場合、戻したいリビジョンを選択して、**逆マージ**チェックボックスを必ずチェックしてください。

このブランチからの変更をすでにマージしてある場合、うまくいけば変更をコミットしたときのメッセージに、マージした最後のリビジョンを記録してあるかもしれません。この場合、作業コピーのログメッセージを追跡するのに **ログを表示** を使用できます。リビジョンをチェンジセットとして考えているのを思い出し、前回のマージの終点を今回のマージの始点をしてください。たとえば、前回リビジョン 37 から 39 までマージした場合、今回のマージの始点をリビジョン 40 にしてください。

Subversion のマージ追跡機能を使用する場合、どのリビジョンをすでにマージしたかを覚えておく必要はありません。Subversion が記録しています。リビジョン範囲をからのままにしておくと、まだマージしていないリビジョンすべてが対象になります。詳細は「マージ追跡」をご覧ください。

マージ追跡機能を使用する場合は、ログダイアログには前回マージされたリビジョンや、共通の祖先以前の時点（つまりブランチがコピーされる以前）のリビジョンが、グレーで表示されます。**マージ不可能なリビジョンを隠す**チェックボックスを使用すると、こうしたリビジョンを隠し、マージすることができるリビジョンだけを表示することができます。

他の人が変更をコミットしている場合、HEAD リビジョンを指定する場合は注意してください。最後の更新のあとに誰かがコミットを行っており、想定しているリビジョンを指していないかもしれません。

次へ >をクリックすると、「マージオプション」に進みます。

4.20.2. ブランチの再統合

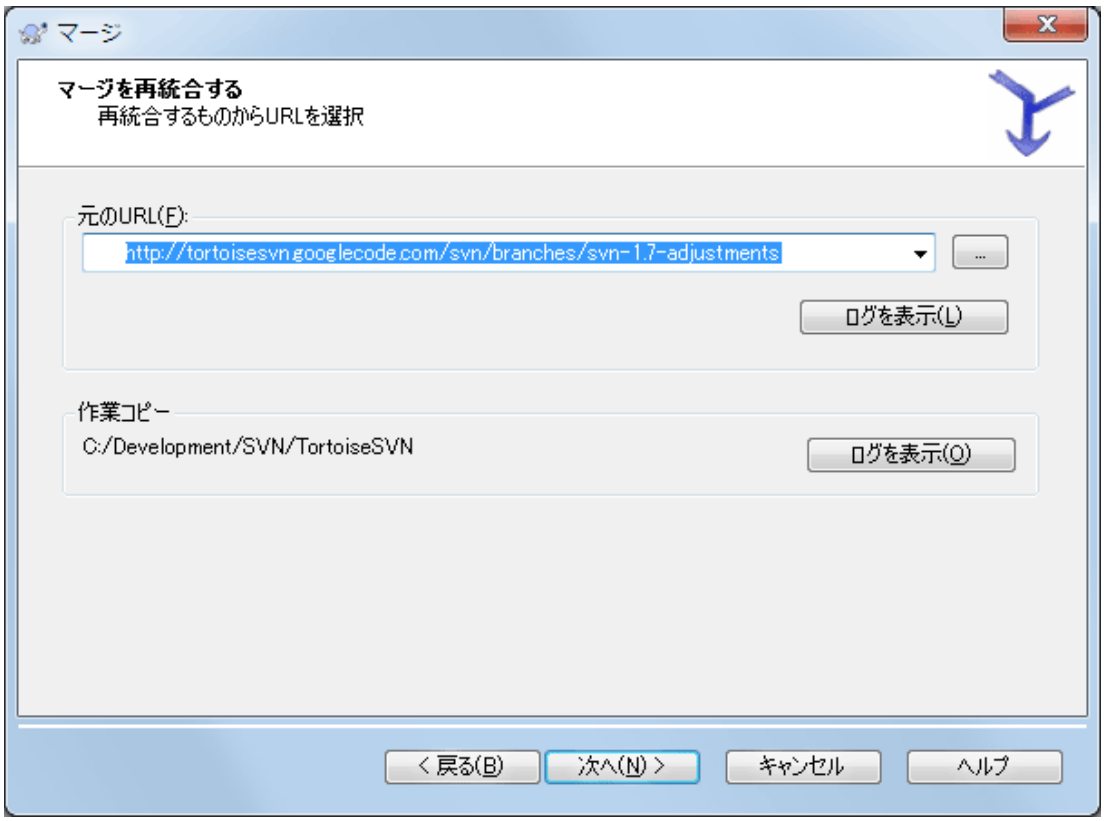


図4.46 マージウイザード - マージの再統合

機能ブランチをトランクにマージするには、トランクの作業コピーから、マージウィザードを起動しなければなりません。

元のURL: フィールドには、マージして戻したいブランチの、完全なフォルダー URL を入力してください。... をクリックして、リポジトリの参照もできます。

再統合マージを適用するには、いくつか条件があります。まず、サーバーがマージ追跡をサポートしていなければなりません。作業コピーの深さに制限があってはなりません(部分的なチェックアウトであってはならない)。ローカルの変更や切り替えられた項目、HEAD 以外のリビジョンに更新された項目があってはなりません。ブランチでの開発中にトランクに行われた変更は、ブランチをまたがってマージされます(もしくはマージ済みとマークされます)。マージ対象になるリビジョンの範囲は自動的に計算されます。

4.20.3. 2つの異なるツリーをマージする

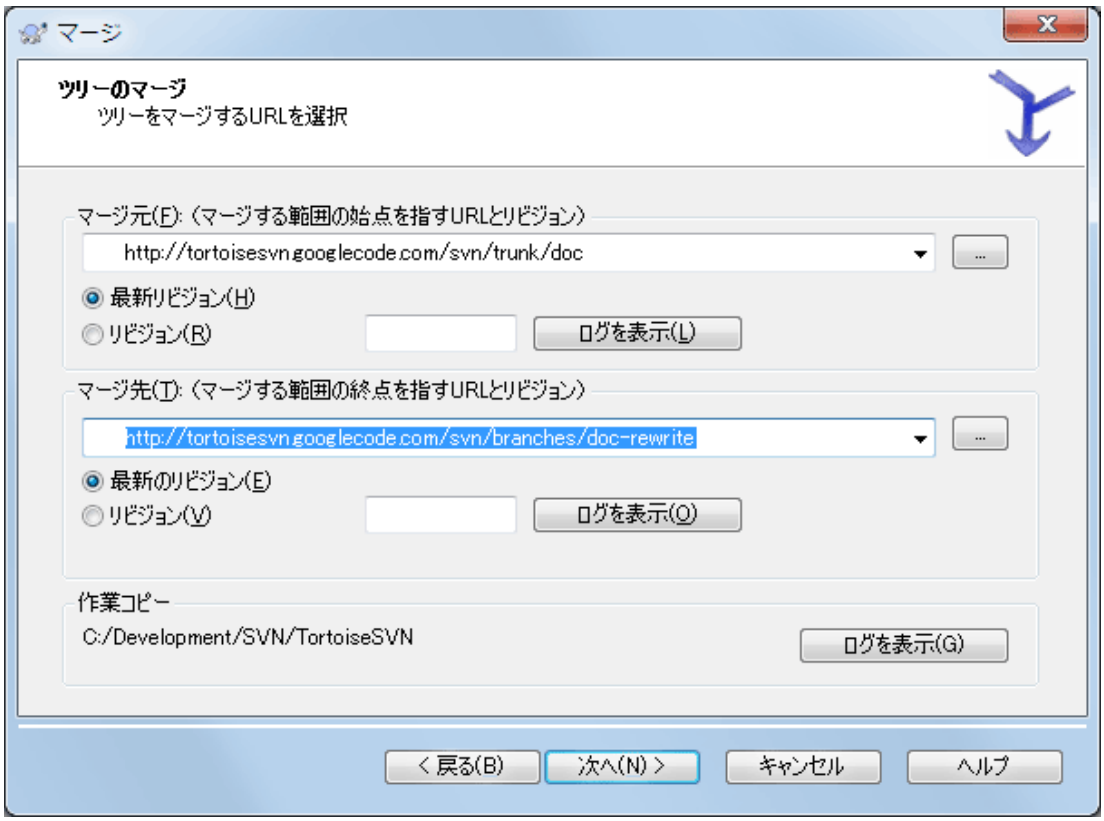


図4.47 マージウィザード - ツリーのマージ

機能ブランチをトランクにマージするために、この方法を使用する場合、トランクの作業コピーでマージウィザードを起動する必要があります。

開始点: 欄には、トランク のフォルダーの URL を入力してください。間違っているように思われるかもしれませんが、トランクが、ブランチの変更を追加する起点となることを思い出してください。... をクリックして、リポジトリを参照することもできます。

終了点: 欄に、機能ブランチのフォルダーの URL を入力してください。

開始点リビジョン 欄と **終了点リビジョン** 欄の両方に、同期をとりたい2つのツリーの最新リビジョン番号を入力してください。他の誰もコミットしていないという確認が取れているのなら、どちらも HEAD リビジョンを指定できます。同期をとってから、誰かがコミットする機会があったのなら、最近のコミットの内容を失わないように、リビジョン番号を指定してください。

リビジョンを選択するには **ログを表示** も使用できます。

4.20.4. マージオプション

ウィザードのこのページでは、マージの実行前に、さらにオプションの指定ができます。ほとんどの場合デフォルトの設定を使用するだけでしょう。

マージする深さを指定することもできます。これは、作業コピー内をどの程度深くたどってマージを行うかということです。深さについては「チェックアウトの深さ」で説明しています。デフォルトの深さは **作業コピー** で、既存の深さをそのまま使用します。

大抵の場合、マージ時にはファイルの履歴を考慮した方が好ましいので、共通の祖先からの変更がマージされます。時には、リポジトリに入っていなくても、関連している可能性が高いファイル同士ををマージする必要もあるでしょう。例えば、第三者が作成したライブラリのバージョン1と2が、別々なディレクトリに入れられていた場合です。理屈からみれば両者は関連していますが、Subversionにはそれが別個にインポートされたファイルの集合としか認識できず、両者が関連

しているとは分かりません。この2つのツリーの差分をマージしようとすると、全体を追加して全体を削除する操作になってしまいます。Subversionが履歴の差分を無視して現状の差分のみを使用するようにするには、 **履歴を無視する** をチェックしてください。本件についての詳細な説明は、 Subversion book の Noticing or Ignoring Ancestry [http://svnbook.red-bean.com/en/1.7/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry] を参照してください。

改行コードや空白の変更に対する扱い方を指定できます。このオプションについては 「改行コードと空白のオプション」 で説明しています。デフォルトの振る舞いは、空白や改行コードの差異も実際の変更としてマージします。

強制的にマージする をチェックすると、ローカルで変更されたファイルやバージョン管理されていないファイルに対して削除を行うようなツリーの競合を無視します。こうしたファイルが削除されると復元する方法がないので、このオプションはデフォルトではチェックされていません。

マージ追跡を使用していて、実際にはマージせずにマージしたという印をつけたい場合、 **マージを実施せず、マージしたことにする** チェックボックスをチェックしてください。そうする理由として2つ考えられます。1つは、マージそのものがマージアルゴリズムに対して複雑すぎ、手でコードを変更した後、マージアルゴリズムが行うようにマージにより変更したという印をつけたい場合です。もう1つは、特定のリビジョンがマージされるのを防ぐ場合です。すでにマージしたという印がついていれば、マージ追跡を関知するクライアントでは、マージを防げます。

すべての設定が終われば、あとは **マージ** ボタンを押すだけです。このとき **マージのテスト** を押すと、作業コピーを 変更せずに マージ結果をプレビューすることができます。これでマージを実行した時に変更が入るファイルの一覧や、競合が発生する 可能性のある ファイルを見ることができます。マージ処理はマージを追跡することではるかに複雑になるため、あらかじめマージが競合なしで完了するかを確実に判断する方法がないので、実際には問題なくマージできるようなファイルであっても、マージのテストでは競合すると表示される場合もあります。

マージ進行ダイアログには、マージの各状態を、リビジョン範囲とともに表示します。ここには想定していたよりも1つ多くリビジョンを表示するかもしれません。例えば、リビジョン 123 をマージするように指示した場合、進行ダイアログには、「リビジョン 122 ～ 123 をマージ」と表示されます。マージは差分と密接に関連していることを思い出してください。マージ処理は、リポジトリの二点間における差分の一覧を生成し、その差分を作業コピーに適用する形で動作します。進行ダイアログは、単純にこの差分の始点と終点を表示しているに過ぎません。

4.20.5. マージ結果のレビュー

さて、マージが完了しました。マージ結果を見て期待通りになっているかを確認するのがいいでしょう。通常マージはかなり複雑です。ブランチがトランクからかなりずれてしまえば、しばしば競合を引き起こします。

バージョン 1.5 未満の Subversion のクライアント・サーバーは、マージ情報を格納しておらず、マージしたリビジョンは、手作業で追跡しなければなりません。ある変更点のテストを行い、そのリビジョンをコミットする際に、マージで取り込んだリビジョン番号を 常に コミットログに記録するべきです。あとで別のマージを適用しようとしたときに、再度変更を取り込むことがないよう、すでにマージした内容を知る必要があります。これについては Subversion Book の Best Practices for Merging [http://svnbook.red-bean.com/en/1.4/svn.branchmerge.copychanges.html#svn.branchmerge.copychanges.bestprac] をご覧ください。

サーバーとすべてのクライアントが Subversion 1.5 以上の場合、マージ追跡機構がマージしたリビジョンを記録し、再度マージすることがないようにします。これにより、単純にリビジョン範囲全体を指定し、実際に新しいリビジョンのみがマージされるといったことができるようになります。

ブランチ管理は重要です。このブランチをトランクに合わせて最新の状態を維持したければ、たびたびマージを確実に行わないと、ブランチとトランクがだんだん離れていってしまうのです。もちろん上で説明したように、変更点を再度マージしてしまうことは避けなければなりません。

ヒント

昨日ブランチからトランクへマージが完了すると、トランクには新機能のすべてのコードが含まれ、ブランチは必要なくなります。必要ならリポジトリから削除してかまいません。

重要

Subversion　ファイルをフォルダーとマージはできませんし、逆もまた同様です。ただフォルダーとフォルダー、ファイルとファイルで行えます。ファイルをクリックしてマージダイアログを開いたら、ファイルのパスを与えなければなりません。フォルダーを選択してダイアログを開いたらマージするフォルダーの URL を指定しなければなりません。

4.20.6. マージ追跡

Subversion 1.5 は、マージ追跡機構を導入しました。あるツリーから別のツリーへ変更をマージすると、マージしたリビジョン番号を格納し、この情報を様々な異なる用途に使用します。

- 再度同じリビジョンをマージする危険（再マージ問題）を避けられます。一度マージ済みとマークされたリビジョンは、将来のマージで、マージ範囲にそのリビジョンが含まれていてもスキップします。
- トランクにブランチをマージする際、トランクのログの一部として、ブランチのコミットもログダイアログに表示し、変更のトレーサビリティが向上します。
- マージダイアログの中からログダイアログを表示すると、すでにマージしたリビジョンを灰色で表示します。
- ファイルに対して注釈履歴を表示する際、マージした人ではなく、マージされたリビジョンのオリジナル作者を表示するよう選択できます。
- マージされたリビジョンのリストにある、実際にはマージされていないリビジョンに対して、未マージ としてマークできます。

マージを行う際、クライアントはマージ追跡情報を、`svn:mergeinfo` に格納します。マージをコミットする際には、サーバーはその情報をデータベースに格納し、マージやログ、注釈情報などのリクエストに適切に応答します。システムが適切に動作するには、サーバーとリポジトリ、全クライアントを確実にアップグレードしなければなりません。以前のクライアントでは、`svn:mergeinfo` プロパティを格納しませんし、以前のサーバーでは、新しいクライアントが要求した情報を提供できません。

マージ追跡についてそれ以外のことについては、Subversion のMerge Tracking [http://svn.apache.org/repos/asf/subversion/trunk/notes/merge-tracking/index.html] の説明をご覧ください。

4.20.7. マージ中に発生した競合の扱い

マージが常にすんなり完了するとは限りません。時には競合を起こしますし、複数の範囲をマージしている場合は、一般的に、次の範囲のマージを行う前に、競合を解決しておきたいでしょう。TortoiseSVN は、競合の解決 ダイアログを表示して、この処理の手助けをしてくれます。

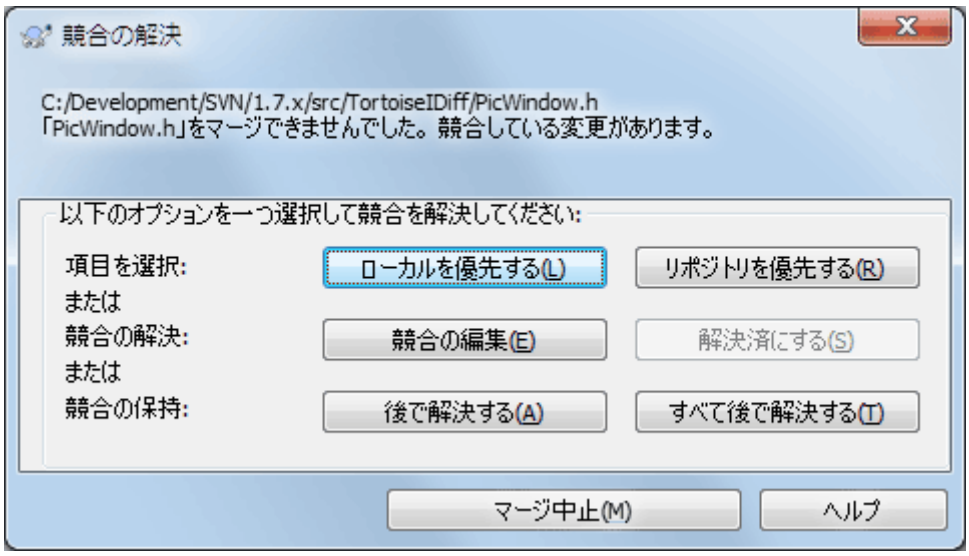


図4.48 競合の解決ダイアログ

リポジトリにコミット済みの他の変更と、ローカルでの他の変更が競合した場合は、円滑にマージされている可能性が高いです。マージ可能なすべての変更がマージされます。マージ競合コールバックダイアログは、競合を解決する3つの異なる方法を提供します。

1. マージ対象にバイナリファイルが含まれていると、そのファイルの競合をマージすることはできません。どちらかのファイルを選択する必要があります。 ローカルを優先する を実行すると、自分の作業コピー内にあるローカル版が採用され、 リポジトリを優先する を実行すると、リポジトリ内のマージ元のファイルが採用されます。

   テキストファイルをマージする場合、最初の2つのボタンを押すと競合しない行はそのままマージされ、競合する行については指定したバージョンが常に優先されます。ローカルを優先するを選択すると、すべての競合点でローカルの変更が選択されます。つまり、マージ元の変更よりも、ローカルの内容が優先されます。同様に、リポジトリを優先するを実行すると、すべての競合点でリポジトリの変更が選択されます。つまり、自分の作業コピーにあったものより、マージ元の変更が優先されます。簡単なようですが、競合が予想以上に多くの行を上書きしてしまい、予期しない結果をもたらすことがあります。

2. 通常は、自分で競合を確認して解決すると思います。この場合、マージツールが起動する際に**競合の編集**を選択してください。結果に問題がなければ、**解決済にする**をクリックしてください。

3. 最後の選択は、解決を先延ばしにして、マージを継続することです。現在競合したファイルと、残りのマージするファイル双方に対して、その選択ができます。しかし、そのファイルにまだ変更があると、マージは完了しないでしょう。

この対話的コールバックが必要なければ、マージ進行ダイアログに **非対話的なマージ** チェックボックスがあります。マージ時にこれをセットし、マージ結果に競合がある場合、ファイルに競合マークが付き、マージを継続します。マージがすべて完了したあとで、競合を解決しなければなりません。セットしない場合は、競合マークがファイルに付く前に、マージ中に競合解決を行う機会があります。ファイルが複数のマージ（複数のリビジョンがそのファイルに変更を与える）を受ける場合、影響を受ける行によって、続くマージが成功するという利点があります。しかしもちろん、マージ実行中には、コーヒーを入れにそこを離れるわけにはいきません。

4.20.8. 完全なブランチをマージ

機能ブランチの変更を、トランクにすべてマージして戻す場合、拡張コンテキストメニュー（Shift　キーを押したままファイルを右クリック）にある TortoiseSVN → **マージの再統合...** を使用できます。

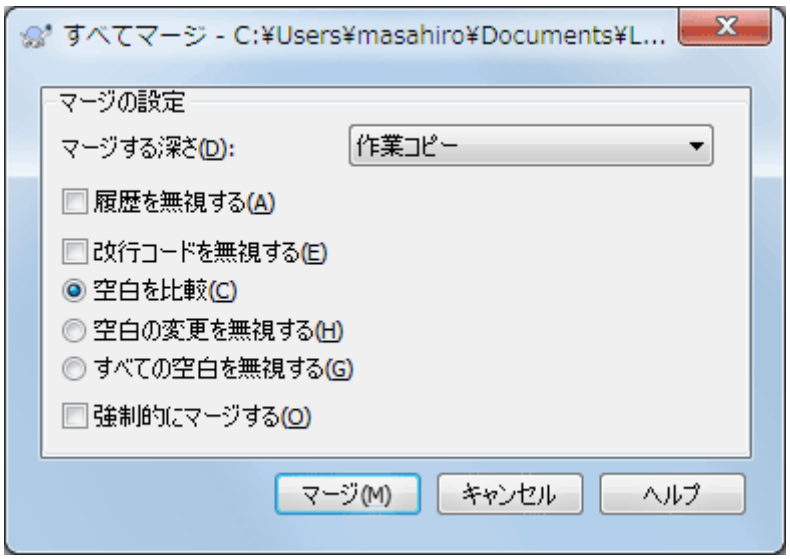


図4.49 マージ再統合ダイアログ

このダイアログは非常に簡単です。しなければならないことは、「マージオプション」にあるように、マージオプションを設定することだけです。TortoiseSVN は、マージ追跡を使用して残りを自動的に処理します。

4.20.9. 機能ブランチの保守

別々のブランチで新しい機能を開発する際、機能が完成した時の再統合指針を立てると良いと思います。別の作業がトランク（trunk）で同時に進んでいる場合、時間がたつにつれて差異は深刻になり、マージを行うのは悪夢のようになります。

機能が比較的単純で、開発に時間がかからなそうであれば、機能が完成するまで全体を分けたブランチを維持し、ブランチの変更をトランクにマージするという、単純なアプローチを採用できます。マージウィザードでは、これを単純に **リビジョンの範囲をマージ** で、リビジョン範囲にブランチのリビジョン期間を指定して行います。

その機能に時間がかかり、トランクに対して変更を説明する必要がある場合、ブランチの同期を維持する必要があります。これは、トランクの変更点 プラス 新機能をブランチが持つように、単純にトランクへの変更を定期的にブランチへマージするということです。同期プロセスでは **リビジョンの範囲をマージ** を用います。機能が完成したら、**ブランチを再統合する** や **異なる2つのツリーをマージ** のどちらかを用いて、trunk にマージできます。

4.21. ロック

Subversion は一般的に、「コピー・変更・マージモデル」で先に説明したとおり、「コピー・変更・マージ」法を使用して、ロックしない方が最もよく動作します。しかし、ロックするポリシーの形で実現する必要があるかもしれません。

- 例えば画像ファイルといった、「マージできない」ファイルを使っている場合。同じファイルを 2 人の人が変更した場合、マージできません。そのためどちらかの人の変更が失われます。
- 自分の会社が過去に、ロックするリビジョン管理システムを常に使用していて、管理するのに 「ロックが一番だ」 と決まっている場合。

第一に Subversion サーバーをバージョン 1.2 以降に確実にアップグレードする必要があります。それ以前のバージョンでは、ロックを全くサポートしていません。file:// アクセスを使用するなら、もちろんクライアントの方を更新する必要があります。

4.21.1. Subversion でロックがどのように働くか

デフォルトではロックを行わず、コミットアクセスできる人が、いつでもどのファイルでも変更をコミットすることができます。他の人は自分の作業コピーを定期的に更新し、ローカルに行った変更をリポジトリにマージするでしょう。

ファイルの ロックを取得 すると、自分しかそのファイルをコミットできなくなります。他のユーザーがコミットしようとしても、自分がロックを解除するまでできません。ロックしたファイルは、どんな方法でもリポジトリ内の変更ができません。そのため、ロック所有者を除いて削除も名前の変更もできなくなります。

重要

ロックはユーザー単位ではなく、特定のユーザーと作業コピーに結び付けられています。ある作業コピーでロックを獲得すると、同じユーザーでも別な作業コピーをコミットすることはできなくなります。

例えば、John というユーザーがオフィスのPCで作業コピーを持っていたとします。そこに彼は、ある画像ファイルの作業を開始し、そのファイルのロックを獲得したとします。オフィスを離れるとき、そのファイルをまだ終えていないので、彼はそのロックを解除しませんでした。家に帰って、John はこちらでも作業コピーを持ち、プロジェクトの作業をもう少しすることにしました。しかし、オフィスの作業コピーでロックを獲得していたため、同じ画像ファイルを変更したりコミットしたりすることはできません。

しかし他のユーザーは、自分がロックしたことを知る必要はありません。定期的にロック状態をチェックしなければ、まず他のユーザーはコミットが失敗して気が付くでしょう。ほとんどこのケースでしょうがあまり便利ではありません。ロックの管理を簡単にするには、新しい Subversion のプロパティで `svn:needs-lock` があります。ファイルにこのプロパティが（値は何でも）セットされていると、ファイルをチェックアウトや更新すると常にローカルのコピーは読み取り専用になります。ファイルにロックを取得しない限りこのままです。この動作は、まずロックを取得するまでファイルの編集ができないということを警告しています。バージョン管理下で読み取り専用のファイルは、編集前にロックする必要があることを示すように、TortoiseSVN では特別なオーバーレイアイコンでマークされます。

ロックは所有者と共に作業コピーの場所に記録されます。自宅や職場など複数の作業コピーがある場合、ロックは作業コピーの中のひとつだけに記録されます。

仕事仲間の一人がロックを取得し、解除しないまま休暇を取ってしまったら、どうしたらいいでしょう? Subversion は強制的にロックする方法を用意しています。他の誰かが持っているロックを解除することを、ロックの 強制解除 と呼び、他の誰かが既にロックしているファイルを強制的にロックすることを、ロックの 横取り と呼びます。当然、仕事仲間と友人でいたいなら、軽々しく行うことではありません。

ロックはリポジトリに記録されます。また、ロックトークンはローカルの作業コピーに作成されます。他の誰かがロックを強制解除したりして、食い違いが発生すると、ローカルのロックは無効になります。リポジトリは常に決定的なリファレンスです。

4.21.2. ロックの取得

ロックを取得したい作業コピーのファイルを選択し、**TortoiseSVN → ロックの取得...** コマンドを選択してください。

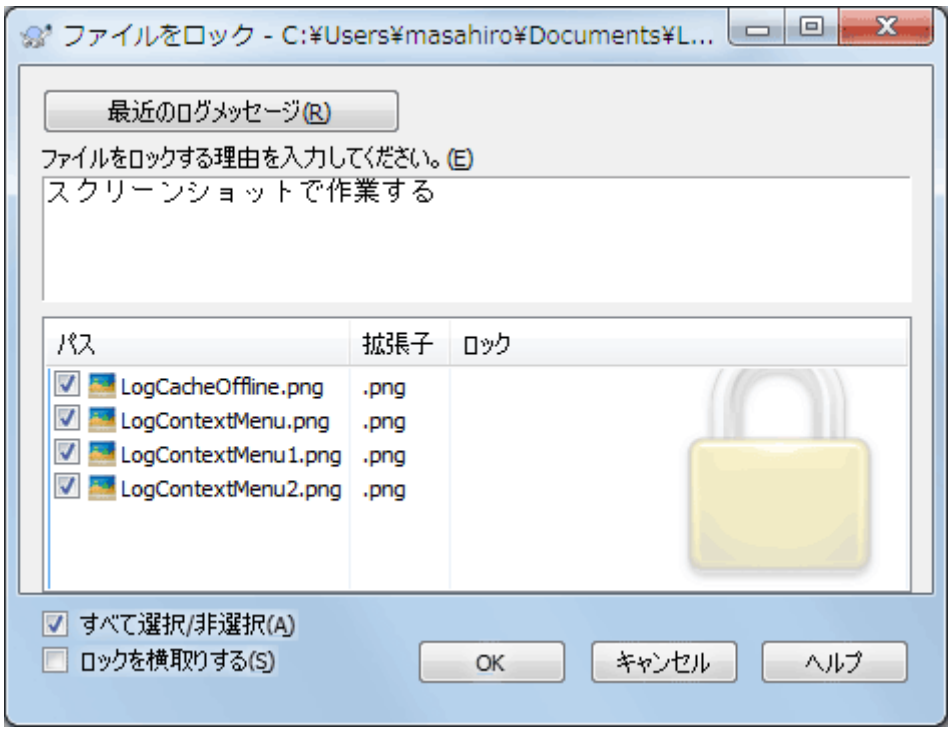


図4.50 ロックダイアログ

ダイアログが現れ、コメントを入力できるようになります。そのため他の人はなぜロックしたのかが判ります。コメントはオプションで、現在のところ Svnserve ベースリポジトリでしか使用できません。他の誰かからロックを横取りする場合(のみ)、**ロックを奪う** にチェックをつけてください。その後 **OK** をクリックしてください。

プロジェクトのプロパティの `TSVN:logtemplatelock` を設定すると、ロック時のメッセージを記入するユーザーにメッセージのひな形を提供することができます。プロパティを設定する方法は、「プロジェクト設定」を参照してください。

フォルダーを選択して、TortoiseSVN → **ロックを取得...** を使用すると、ロックするよう選択したすべてのサブフォルダー内の すべてのファイルがある状態でロックダイアログが開きます。本当に全階層をロックするのなら、これでできます。ですが、本当に仕事仲間をプロジェクトから閉め出してしまうのなら、彼らの中の評価は非常に悪くなるでしょう。慎重に使用してください...

4.21.3. ロックの解除

もう必要でなくなったロックの解除を忘れないように、ロックされたファイルはコミットダイアログに表示され、デフォルトで選択されています。コミットを続けると、変更されていないとしても、選択したままのファイルのロックが解除されます。特定のファイルでロックを解除したくない場合、そのファイルのチェックを外せます(変更されていない場合)。変更したファイルのロックを保持したければ、変更をコミットする前に **ロックを保持** チェックボックスを有効にしておく必要があります。

手動でロックを解除するには、ロックを解除したいファイルを作業コピーで選択し、TortoiseSVN → **ロックを解除**コマンドを選択してください。追加で入力することはありません。そこで TortoiseSVN はリポジトリに接続し、ロックを解除します。フォルダーに対してこのコマンドを使用し、再帰的に全ロックの解除を行えます。

4.21.4. ロック状態のチェック

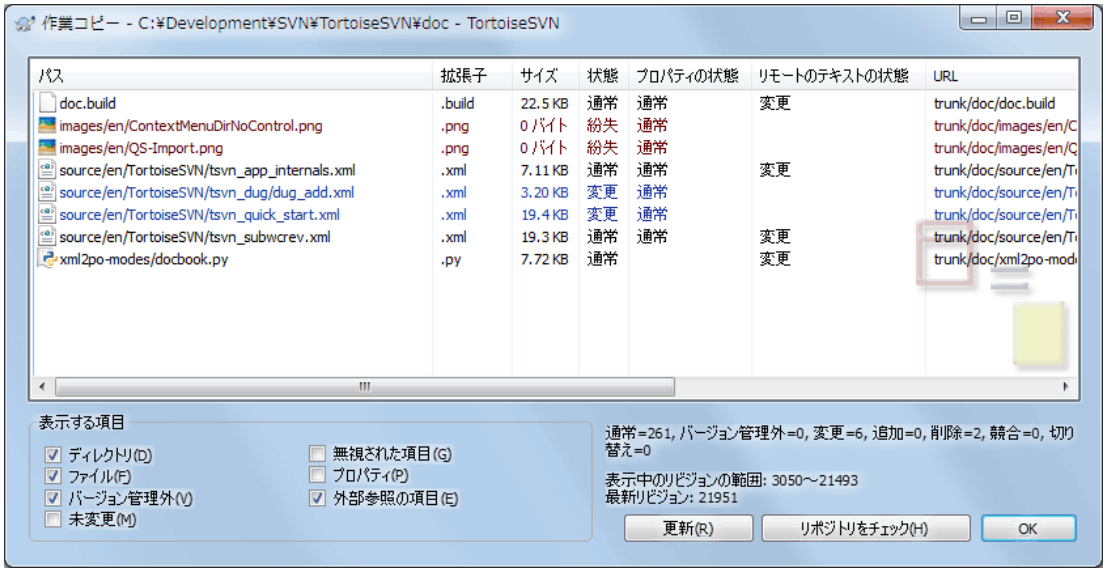


図4.51 変更をチェックダイアログ

誰がロックをかけているかを確認するのに、TortoiseSVN → 変更をチェック... を使用できます。ローカルに保持したロックの印はすぐに現れます。他の人が保持しているロックをチェックするには（そして誰がロックを強制解除したり横取りしたりしたかを見るには）、**リポジトリをチェック** をクリックしてください。

ここのコンテキストメニューから、他の人が保持しているロックを、強制解除したり横取りしたりするように、ロックを取得したり解除したりもできます。

ロックの強制解除・横取りは避ける

ほかの誰かのロックを、断りもなく強制解除・横取りすると、潜在的に作業を失う原因になります。マージ不可能なファイルの種類で作業していて、他の誰かのロックを横取りした場合、自分がロックを解除すると、自由に上書きできます。Subversion はデータを失いませんが、ロックがもたらすはずだった、チーム作業の保護を失ってしまいます。

4.21.5. ロックしていないファイルを読み込み専用にするには

上記のように、ロックを使用するのに最も効果的な方法は、`svn:needs-lock` プロパティを設定することです。プロパティの設定については 「プロジェクト設定」 をご覧ください。このプロパティを持つファイルは、ロックを取得していないと、チェックアウトや更新をしたときに常に読み込み専用になります。

TortoiseSVN は、忘れないように特別なオーバーレイアイコンで表示します。

ファイルやフォルダーをリポジトリに追加した際に、自動的にプロパティを設定するように Subversion を設定できます。詳細情報は「プロパティの自動設定」をご覧ください。

4.21.6. ロックのフックスクリプト

Subversion 1.2 以降で作成したリポジトリでは、リポジトリの `hooks` ディレクトリに4つフックテンプレートが追加されています。それぞれロック取得の前後、ロック解除の前後に呼ばれます。

ファイルがロックされたときに、そのファイルを表すのに email を送信するような、post-lock や post-unlock フックスクリプトをサーバーにインストールするのは名案です。そういったスクリプトが適切な場所にあると、誰かがファイルをロック・ロック解放するとユーザーすべてに通知されます。リポジトリフォルダーの hooks/post-lock.tmpl に、サンプルフックスクリプトがあります。

フックを使用して、ロックの破壊・横取りを禁止したり、管理者に制限したりもできます。または、ロックの破壊・横取りが発生したら、ロックの所有者に email を送りたいかもしれません。

詳細は「サーバー側フックスクリプト」を参照してください。

4.22. パッチの作成及び適用

(TortoiseSVN のような)オープンソースプロジェクトでは、リポジトリは誰もが読み込みアクセスできるようになっていて、誰もがプロジェクトに貢献できるようになっています。では、どのように貢献をコントロールするのでしょう? 誰もが変更をコミットできるようにすると、プロジェクトは永久に不安定になり、ややもすると永久に壊れた状態になるかもしれません。こういった状況では、変更を開発チーム(書き込む権限がある)への パッチ ファイルの送信といった形で管理しています。開発チームはまずパッチをレビューし、リポジトリに送信したり、拒否して作者に返却したりします。

パッチファイルは単なるUnified差分ファイルで、作業コピーと元になったリビジョンとの差分を表しています。

4.22.1. パッチファイルの作成

まず、変更したものの make とテスト をする必要があります。それから、親フォルダーで TortoiseSVN → コミット... する代わりに、TortoiseSVN → パッチを作成... を選択してください、

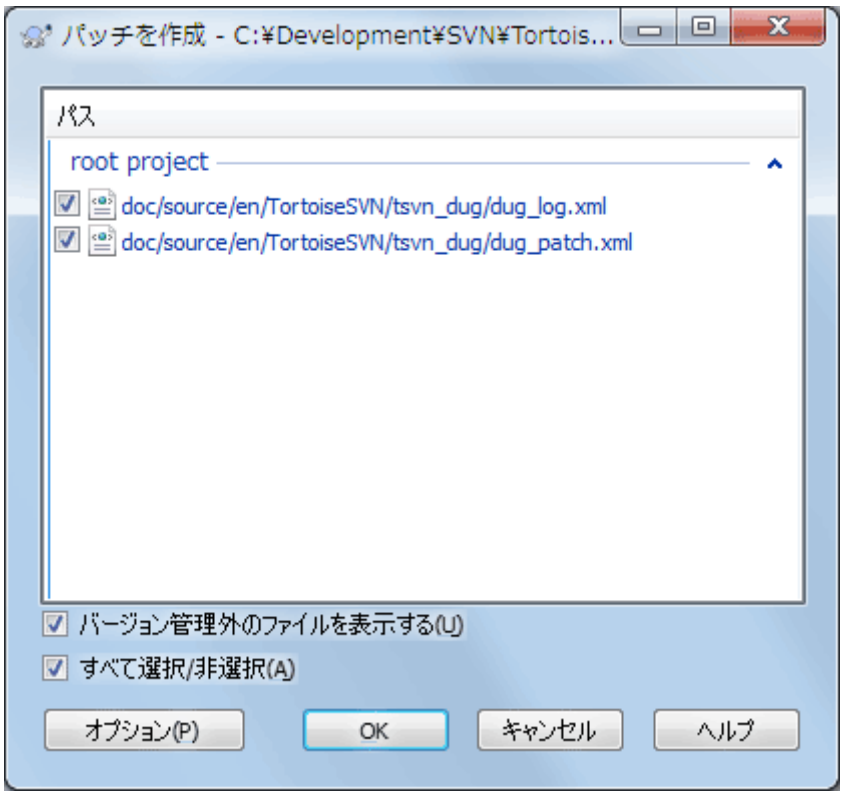


図4.52 パッチ作成ダイアログ

ここで完全なコミットをするときのように、パッチに含めるファイルを選択できます。これで選択したファイルに対する、リポジトリを最後に更新したときからの変更点をまとめたファイルを得ることができます。

ダイアログの中の列は、**変更をチェック** ダイアログの中の列と同じ方法でカスタマイズできます。詳細は「ローカルとリモートの状態」をご覧ください。

オプションボタンはをクリックすると、パッチの作り方を指定できます。たとえば、出力するパッチファイルで行末文字や空白文字の変更を含めないようにすることができます。

違うファイルの変更点を含めた分割したパッチも生成できます。もちろん、パッチファイルを生成してから、同じ ファイルに変更を加えて別のパッチを作成すると、2つ目のパッチには 両方 の変更点が含まれることになります。

それでは、任意のファイル名でファイルを保存してください。パッチファイルは任意の拡張子でかまいませんが、`.patch` や `.diff` を拡張子に使用するのが通例となっています。これでパッチファイルを送信する準備ができました。

また、ファイルに保存せずクリップボードに保存することもできます。電子メールに貼り付けて誰かにレビューしてもらう際に、そうしたくなると思います。また、1つのマシンに2つ作業コピーがあり、片方からもう片方へ変更を持っていくときに、クリップボードにパッチがあるとべんりです。

必要であれば、**コミット**や**変更をチェック**ダイアログでパッチファイルを作成することができます。ファイルを選択して、コンテキストメニューからパッチを作成するメニュー項目を選択してください。**オプション**ダイアログを表示する場合は、Shift を押しながら右クリックしてください。

4.22.2. パッチファイルの適用

作業コピーにパッチファイルを適用します。パッチを作成したフォルダーと同じ階層で行う必要があります。よくわからなければ、パッチファイルの最初の行を見てください。たとえば、最初のファイルが `doc/source/english/chapter1.xml` に対するもので、パッチファイルの最初の行が `Index: english/chapter1.xml` なら、`doc/source/` フォルダーでパッチを適用する必要があります。適切な作業コピーを使用していても、適用するフォルダー階層が間違っていると、TortoiseSVN は適切な階層を使用するよう注意を促します。

パッチファイルを作業コピーに適用するために、少なくともリポジトリの読み込み権限が必要です。これは、他の開発者がリビジョンを変更していないかどうか、マージプログラムが参照するからです。

フォルダーのコンテキストメニューから、TortoiseSVN → **パッチを適用...** をクリックしてください。すると「ファイルを開く」 ダイアログボックスが現れ、適用するパッチファイルを選択できます。デフォルトでは `.patch` ファイルか `.diff` ファイルのみが表示されていますが、「すべてのファイル」を選択できます。あらかじめパッチをクリップボードに保存していれば、「ファイルを開く」ダイアログの **クリップボードから開く...** を使用できます。

その他には、パッチファイルが `.patch` や `.diff` といった拡張子を持つ場合、直接そのファイルを右クリックして、TortoiseSVN → **パッチを適用...** を選んでください。この場合、作業コピーの場所を入力することになります。

この2つは同じことを違う方法で行っているだけです。1つ目は作業コピーを選択してからパッチファイルを閲覧し、2つ目はパッチファイルを選択してから作業コピーを閲覧します。

一度パッチファイルや作業コピーの場所を選択すると、パッチファイルの変更を作業コピーにマージするように TortoiseMerge が起動します。小さなウィンドウに変更のあったファイルを表示するので、その中の項目をダブルクリックして変更の確認や、マージしたファイルの保存を行ってください。

他の開発者のパッチが適用されたので、今度は誰もがリポジトリから変更点にアクセスできるよう、コミットする必要があります。

4.23. 誰がその行を変更したか?

時々どの行が変更されたかだけでなく、ファイル内で、誰がどの行を変更したのか正確に知りたいことがあります。その場合は、TortoiseSVN → **注釈履歴...** コマンドでできます。また annotate コマンドも役に立つので、時々参照されています。

このコマンドは、ファイルの行ごとに作業者と変更されたリビジョンを表示します。

4.23.1. ファイルの注釈履歴

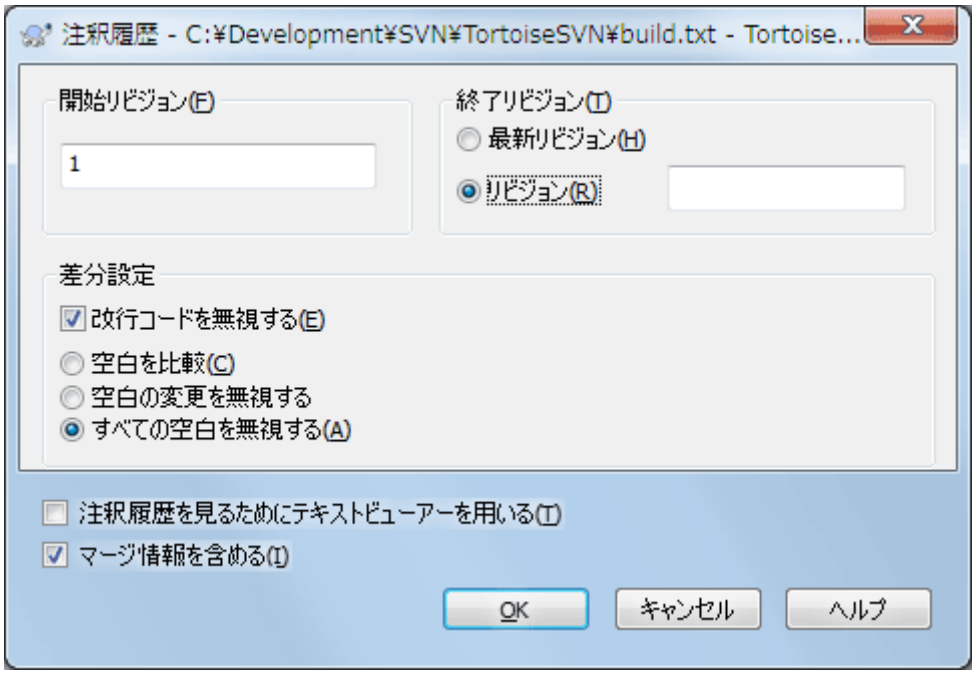


図4.53 注釈履歴ダイアログ

ずっと前のリビジョンに興味がなければ、注釈を見始めるリビジョンを設定できます。全リビジョンの注釈履歴を見るのなら、これを 1 に設定してください。

デフォルトでは、 TortoiseBlame を使用して注釈を見ます。これは、異なるリビジョンを色分けして、見やすくしてくれます。注釈を印刷したり編集したりしたければ、**注釈履歴を見るためにテキストビューアーを用いる** を選択してください。

改行コードや空白の変更に対する扱い方を指定できます。このオプションについては 「改行コードと空白のオプション」で説明しています。デフォルトの振る舞いは、空白や改行コードの差異も実際の変更としますが、インデントの変更を無視してオリジナルの作者を見たい場合は、ここで適切なオプションを選択できます。

このオプションには、サーバーから取得するためかなり時間がかかる可能性はありますが、必要に応じてマージ情報を含めることができます。行が別のソースからマージされると、注釈履歴情報は、変更が元のソースだけでなく、それがこのファイルにマージされたリビジョンで作成されたリビジョンを表示します。

OK を押すと、TortoiseSVN は注釈ファイルを作成するため、データの取得を始めます。なお、ファイルへの変更の量とリポジトリへのネットワーク接続に依存しますが、終了するまでに数分かかります。注釈プロセスが終了すると、結果を一時ファイルに書き出し、結果を表示します。

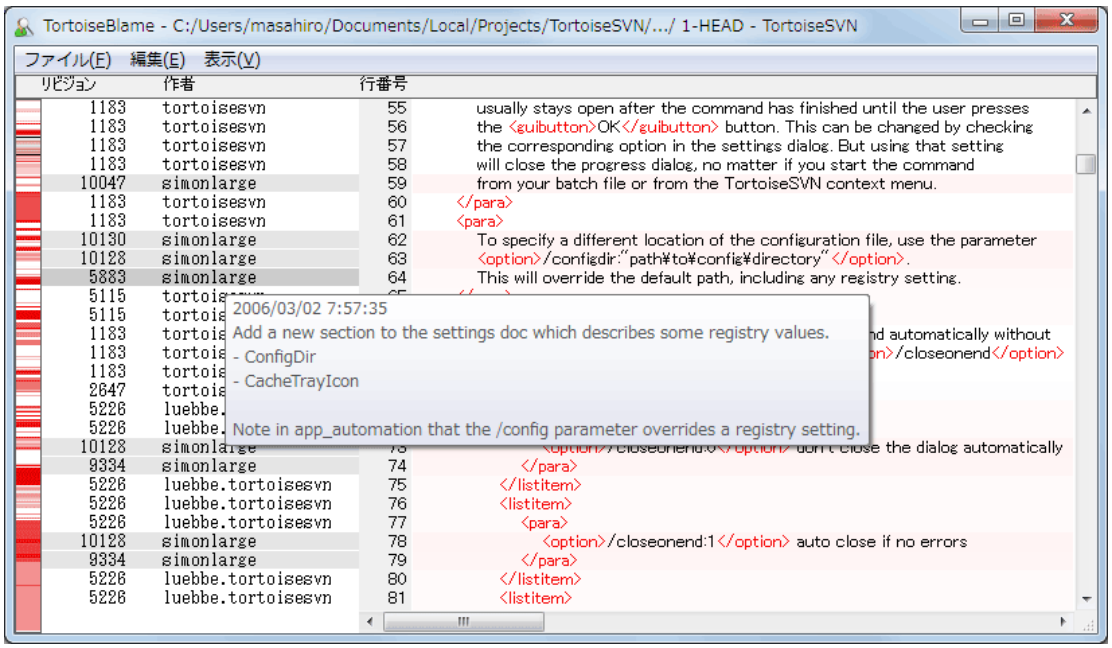


図4.54 TortoiseBlame

TortoiseSVN に含まれる TortoiseBlame は、注釈ファイルを読みやすくしてくれます。マウスをある行の注釈情報列の上に持っていくと、同じリビジョンの行の背景を濃くして表示します。同じ更新者が行ったけれどもリビジョンが異なる行は、薄く色を付けて表示します。ディスプレイが 256 色モードに設定されていると、きれいに色分けされない可能性があります。

行の上で 左クリック すると、同じリビジョンのすべての行に色分けされます。同じ作業者の、他のリビジョンの行は薄い色で色分けされます。この色分けは貼り付いて、色分けを失わないようにマウスを移動できます。リビジョンをもう一度クリックすると、色分けが解除されます。

注釈情報の列の上ならどこでもマウスを持ってくると、ヒントボックスの中にリビジョンのコメント(ログメッセージ)を表示します。そのリビジョンのログメッセージをコピーするには、注釈情報の列を右クリックしてでてくるコンテキストメニューを使用してください。

注釈レポートを **編集 → 検索...** を使用して検索できます。これでリビジョン番号、作業者、ファイルの内容を検索できます。ログメッセージは検索できません。ログダイアログを使用して検索してください。

また、**編集 → 指定した行へ移動...** で指定した行番号にジャンプします。

注釈情報列の上にマウスを持っていくと、リビジョンを比較したり、履歴を検査するのに便利なコンテキストメニューが使用可能になります。この時マウスのある行のリビジョン番号を参照します。**コンテキストメニュー → 前リビジョンの注釈履歴** では、同じファイルに対する前のリビジョンを上限とした注釈履歴レポートを生成します。今見ている行に対する最終更新の、直前のファイルの状態を表す注釈履歴レポートを得られます。**コンテキストメニュー → 変更を表示** は、参照しているリビジョンの変更点を表示する diff ビューアーを起動します。**コンテキストメニュー → ログを表示**は、参照しているリビジョンから始まる、リビジョンログダイアログを表示します。

もっと視覚的に変更点の新旧を見たい場合、**表示 → 各行の変更時期を色分けする** を選択してください。これにより、新しい行は赤、古い行は青のグラデーションで表します。デフォルトの色分けは、薄い色になっています
が、TortoiseBlame の設定で変更できます。

注釈履歴の起動時にマージ追跡を使用していると、マージされた行が若干異なって表示されます。別のパスからのマージによって変更された行には、マージが行われたリビジョンではなく、マージ元のファイルが最後に変更された時のリビ

ジョンと作者が表示されます。この場合、リビジョンと作者が斜体で表示されます。注釈情報列の上にマウスを移動させると、マージされたリビジョンがツールチップ上に表示されます。マージした行をこのような形で表示させたくない場合、注釈履歴を起動するときに **マージ情報を含める** チェックボックスをオフにしてください。

表示 → マージ元のパス を選択すると、マージに関係するパスが表示されます。その行がマージ以外の操作によって最後に変更されたときのパスが表示されます。

注釈履歴情報の中のリビジョンは、その行の内容が変更された最後のリビジョンを表します。ファイルを他からコピーして作成したとき、それから行を変更するまで、その行の注釈履歴のリビジョンは元のソースファイルの最後の変更であり、コピーが作成されたリビジョンではありません。これはマージ情報と共に表示されるパスにも適用されます。パスは、最後の変更がその行に作成されたリポジトリの場所を示しています。

TortoiseBlame の設定は、**TortoiseSVN → 設定...** の TortoiseBlame タブでアクセスできます。「TortoiseBlame の設定」をご覧ください。

4.23.2. 注釈履歴の差分

注釈履歴レポートの限界の1つは、ファイルを特定のリビジョンにおける、各行を変更した最後の人しか見ることができません。時には、誰が変更したかと同様に、誰が作成したかを知りたいことがあります。 TortoiseBlame の行を右クリックしてコンテキストメニューを表示しすると、そのリビジョンでの変更を見ることができます。しかし、変更点と注釈履歴を 同時に 確認したい場合は、差分と注釈履歴のレポートを組み合わせなければなりません。

リビジョンログダイアログには、以下のようないくつかのオプションがあります。

リビジョンの注釈履歴
: 上の欄でリビジョンを2つ選択し、**コンテキストメニュー → リビジョンの注釈履歴**を選択してください。これにより、この2つのリビジョンの注釈履歴を取得し、差分ビューアーで注釈を比較できます。

変更の注釈履歴
: 上の欄でリビジョン1つ選択し、下の欄でファイルを選択したうえで、**コンテキストメニュー → 変更の注釈履歴** を実行してください。選択したリビジョンのとその前のリビジョンの注釈データを取得し、差分ビューアーで注釈を比較できます。

作業ベースとの比較と注釈
: 画面上部のリビジョンをひとつ選択し、**コンテキストメニュー → 作業ベースと比較/注釈履歴**を選択すると、そのファイルのログを表示します。選択したリビジョンと作業ベースにあるファイルの注釈データを取得し、差分ビューアーで注釈を比較します。

4.24. リポジトリブラウザー

時に、作業コピーではなくリポジトリを直接操作する必要もあるでしょう。そのようなときのために リポジトリブラウザー があります。エクスプローラーのようにかつアイコンオーバレイが表示されるので、作業コピーと同じように操作できます。そのため、リポジトリブラウザーでリポジトリの構造や状態を確認できます。

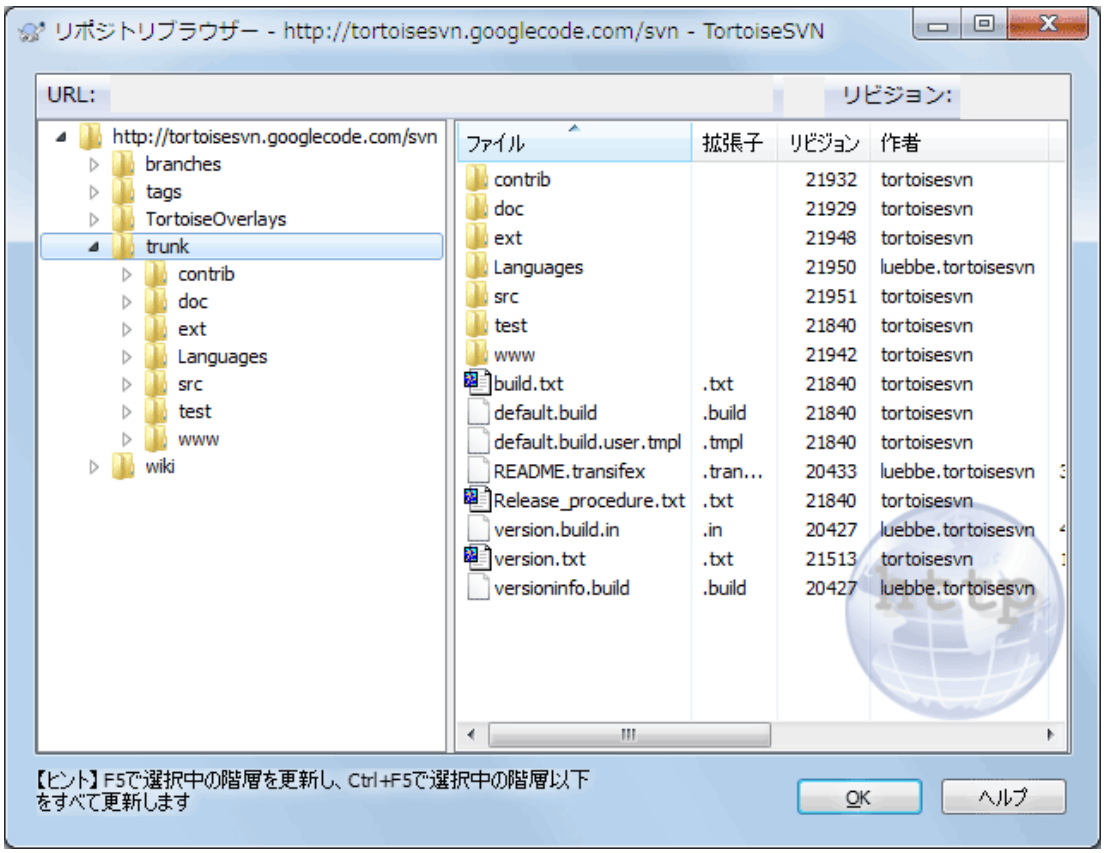


図4.55 リポジトリブラウザー

リポジトリブラウザーでは、コピー、移動、名前の変更、... といったコマンドをリポジトリ上で直接実行できます。

リポジトリブラウザーは、Windows エクスプローラーによく似ており、コンピューター上のファイルと同じように、特定リビジョンのリポジトリの内容を表示できます。左画面にディレクトリツリーが表示され、右画面に選択したディレクトリの内容が表示されます。リポジトリブラウザーウインドウの上部では、参照したいリポジトリの URL やリビジョンを入力できます。

`svn:externals`プロパティに含まれるフォルダーは、リポジトリブラウザーからも見ることができます。この場合、フォルダーには、リポジトリ構造の一部ではなくリンクであることを示す、小さい矢印が表示されます。

Windows エクスプローラーのように、右画面の列見出しをクリックすると、並び順をお好みに変更できます。また、エクスプローラーのように両方の画面でコンテキストメニューを使用できます。

ファイルに対するコンテキストメニューでは、以下のことができます。

- 選択したファイルを、ファイル形式に応じた規定のビューアーか選択したプログラムで開きます。
- 選択されたファイルを編集します。これは一時的な作業コピーをチェックアウトし、そのファイル形式に応じたデフォルトのエディターを起動します。エディターが閉じられると、変更が保存されていた場合はコミットダイアログが表示されるので、コメントを入力して変更をコミットしてください。
- そのファイルのリビジョンログの表示や、そのファイルの出自がわかるような全リビジョンのグラフを表示します。
- 誰が、どの行をいつ変更したのかを参照する、ファイルの注釈履歴を表示します。
- 単一のファイルをチェックアウトします。これはそのファイルだけが含まれる「部分的な」作業コピーを作成します。
- ファイルの削除や名前の変更をします。
- 自分のハードディスクに、そのファイルのバージョン管理外のコピーを保存します。

- アドレスバーに表示されているURLをクリップボードにコピーします。
- リポジトリの別の場所や、同じリポジトリに属する作業コピーにファイルをコピーします。
- ファイルのプロパティを表示・編集します。
- リポジトリブラウザーを起動して、直接この場所を表示するためショートカットを作成します。

フォルダーに対するコンテキストメニューでは、以下のことができます。

- そのフォルダーのリビジョンログの表示や、そのフォルダーの出自がわかるような全リビジョンのグラフを表示します。
- 自分のハードディスクに、フォルダーのバージョン管理外のコピーをエクスポートします。
- 自分のハードディスクに、フォルダーの作業コピーをチェックアウトします。
- リポジトリに新しいフォルダーを作成します。
- バージョン管理されていないファイルやフォルダーを、直接リポジトリに追加します。実質的には Subversion のインポート操作と同じです。
- フォルダーの削除や名前の変更
- リポジトリの別の部分か、同じリポジトリに属する
  作業コピーのどちらかに、フォルダーのコピーを作成します。これは、チェックアウトされた作業コピーを作成せずにブランチ/タグを作成するために使用することもできます。
- フォルダーのプロパティの表示・編集
- 比較用にフォルダーをマークします。マークしたフォルダーは太字で表示されます。
- あらかじめマークを付けたフォルダーに対して、Unified差分ファイルとして、またはデフォルト差分ツールを用いて差分を取ったファイルのリストとして差分を表示できます。これは2つのタグや、トランクとブランチにどんな変更があるのか比較するのに便利です。

右画面で2つの項目を選択すると、差分をUnified差分ファイルや、またはデフォルトの差分ツールを用いて差分を取ったファイルのリストとして表示できます。

右画面で複数のフォルダーを選択すると、一度にすべてを共通の親フォルダーにチェックアウトできます。

選択した2つのタグが同じルート(通常 /trunk/)からコピーされたものなら、**コンテキストメニュー → ログ表示...** を使用して2つのタグ間のリビジョンのリストを表示できます。

外部項目(svn:externalを使用して参照される)もリポジトリブラウザーで表示され、フォルダーの中を開いて見ることもできます。外部項目の上には赤い矢印が表示されます。

いつものように、表示を再度読み込むのに F5 を使用できます。これにより現在表示しているものを、すべて再度読み込みます。先読みをしたり、まだ開いていないノードの情報を読み込むには、Ctrl+F5 を使用してください。この後では、どのノードを展開しても、情報を取得するのにかかるネットワーク遅延は発生しません。

リポジトリブラウザーをドラッグ&ドロップでも操作できます。エクスプローラーからリポジトリブラウザーへフォルダーをドラッグすると、リポジトリにインポートを行います。複数の項目をドラッグすると、それぞれコミットを行うことに注意してください。

リポジトリ内で項目を移動する場合、単に新しい場所へ 左ドラッグ してください。移動ではなくコピーを作成したい場合は、Ctrl-左ドラッグ してください。コピーの際は、エクスプローラーと同様にカーソルが「+」マークになります。

ファイルやフォルダーを他の場所へコピー・移動したい場合や、それと同時に新しい名前を付けたい場合には、左ドラッグをするのではなく 右ドラッグ や Ctrl-右ドラッグ をしてください。この場合、ファイルやフォルダーの名前を入力する名前を変更ダイアログが表示されます。

以上の方法でリポジトリに変更を加える際には、必ずログメッセージ入力ダイアログが表示されます。間違えてドラッグした場合は、その操作をそこでキャンセルできます。

時々パスを開こうとしたときに、項目の詳細の箇所にエラーメッセージが表示されることがあります。これはおそらく、無効な URL を指定したか、アクセス権限がないか、その他サーバーの問題だと思われます。このメッセージをコピーして email に含める場合、右クリックして **コンテキストメニュー → クリップボードにエラーメッセージをコピーする** を使用するか、単純に Ctrl+C としてください。

4.25. リビジョングラフ

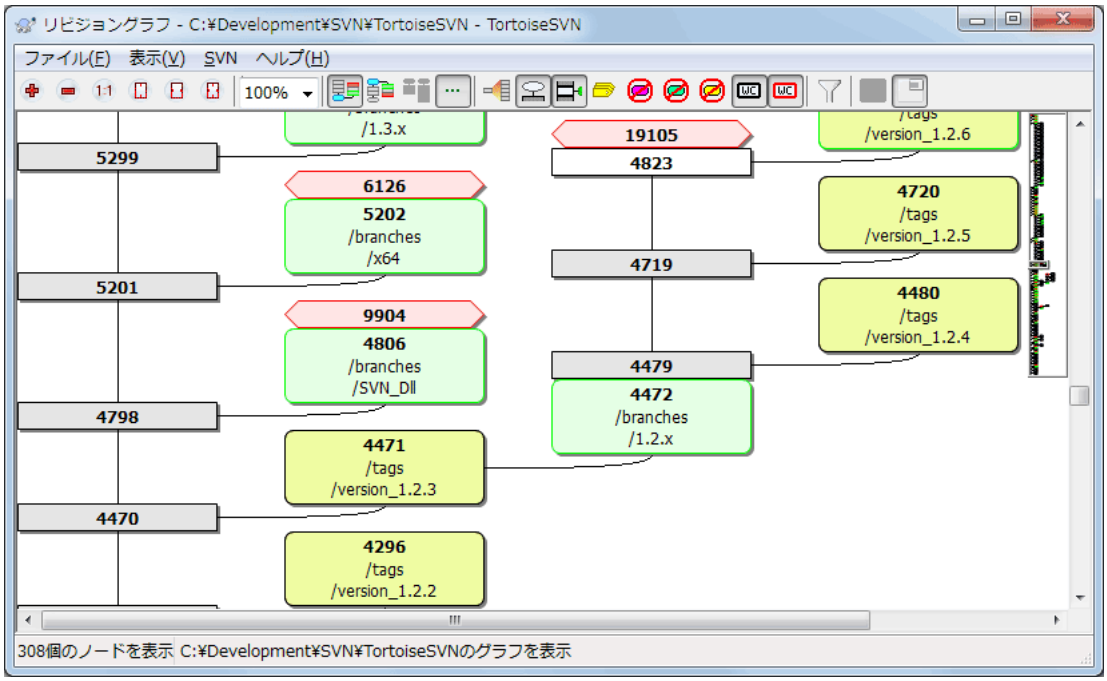


図4.56 リビジョングラフ

時にはブランチやタグがトランクのどこから分かれたのか知る必要があります。この種の情報を見るのに理想的なのは、グラフやツリー構造で見ることです。この必要があるときには、**TortoiseSVN → リビジョングラフ...** としてください。

このコマンドは、リビジョンの履歴を解析し、どのポイントからコピーを取得したか、いつブランチ・タグを削除したかを表示するツリーを作成しようとします。

重要

TortoiseSVN がグラフを生成するには、リポジトリのルートからログメッセージをすべて取得しなければなりません。言うまでもなく、数千リビジョンのリポジトリがある場合（サーバーの速度やネットワークの帯域などにもよりますが）数分かかるでしょう。現在 500,000 リビジョンを越えている Apache プロジェクトのようなところで試せば、しばらく待っていなければなりません。

朗報としては、ログキャッシュを使用している場合は、一度この遅さを我慢すればいいということです。これ以後は、ローカルにログデータを保持しています。ログキャッシュは TortoiseSVN 設定で有効にできます。

4.25.1. リビジョングラフのノード

リビジョングラフの各ノードは、現在表示されているツリーで何かしらの変更が行われたリビジョンを表しています。形や色でノードの違いを区別できます。形は固定ですが、色は TortoiseSVN → **設定** を使用して設定できます。

追加・コピーした項目
: 追加されたり、他のファイルやフォルダーによって作られたりした項目は、長方形で囲んで表示されます。デフォルトの色は緑色です。タグやトランクでは特別なものとして扱われ、異なる色で表示されます。これは TortoiseSVN → **設定** で設定することができます。

削除した項目
: 必要のなくなったブランチなど削除された項目は、八角形（角を落とした長方形）で表示されます。デフォルト色は赤です。

名前を変更した項目
: 名前変更した項目も八角形でも表示されます。デフォルトの色は青です。

ブランチの最新リビジョン
: グラフには通常、分岐したリビジョンしか表示されませんが、それぞれのブランチの最新版である HEAD リビジョンも確認できると便利です。**HEADリビジョンを表示する** を選択すると、各ブランチの HEAD リビジョンのノードが楕円で表示されます。ここで言う HEAD リビジョンとは、そのパスでコミットした最終リビジョンであり、リポジトリ全体の HEAD リビジョンではないことにご注意ください。

作業コピーのリビジョン
: 作業コピーのリビジョングラフを表示した場合、 **作業コピーのリビジョンを表示する** を選択すると、作業コピーの元になった BASE リビジョンが太い輪郭で表示されます。

変更した作業コピー
: 作業コピーのリビジョングラフを表示させた場合、 **作業コピーの変更を表示** を選択すると、変更された作業コピーを表す追加ノードを表示できます。これは輪郭が太い楕円のノードで、デフォルトでは赤で表します。

通常の項目
: 他の項目は、通常の長方形で表します。

デフォルトのいグラフには、項目の追加・コピー・削除のみが表示されています。プロジェクト内のすべてのリビジョンが表示されると、重要な時に非常に巨大なグラフになってしまいます。本当に、変更が行われた すべての リビジョンが必要ならば、**表示** メニューやツールバーに、表示させるためのオプションがあります。

デフォルトの表示（グループ化無効）では、ノードは縦方向に、厳密にリビジョン順で配置されるため、実行された順番が視覚的に分かります。同じ列にある2つのノードの順番は明白です。2つののノードが隣接する列にある場合、ノードが重なる心配がないため、補正は少なくて済みます。そしてそのため、少々明白でない場合があります。そのような最適化が、複雑なグラフを妥当な大きさにしておくために必要です。なお、この順序づけでは、 古い方 にあるノードの 端 （つまり、グラフを古いノードを下に表示する場合は、ノードの下端）を、参照として使用します。ノードの形がすべて同じ大きさであるわけではないため、参照する端は重要です。

4.25.2. 表示の変更

リビジョングラフは非常に複雑になることがありますので、任意の表示にできるようにいくつもの機能があります。ツールバーの**表示**メニューから使用できます。

ブランチでグループ化
: デフォルト（グループ化無効）では、すべての行がリビジョンの時系列になるよう厳密に配置されます。その結果、歴史が長くコミット頻度が低いブランチは、変更量の割に長く列を占有するすることになるため、グラフが非常に広くなってしまいます。

このモードでは変更がブランチ別に分類され、全体のリビジョン順にはなりません。ブランチ上の連続したリビジョンは、(通常は)連続した線で表示されます。サブブランチについても同様に配置されますが、グラフを小さく収めるために、古いブランチと同じ列に後から発生したのブランチが配置されるようになっています。そのため、異なるリビジョンが同じ行に表示されることがあります。

古いものを先頭に表示
通常、グラフは古いものを下に配置し、上の方に伸びていきます。このオプションにより、上から下に伸びていくようになります。

ツリーを上端でそろえる
グラフが、いくつかのツリーにばらばらにされた場合、**ブランチでグループ化**　オプションを使用しているかによって、そのツリーは自然なリビジョン順か、ウィンドウの下部に整列して現れるでしょう。すべてのツリーを上部から下部へのばすには、このオプションを使用してください。

線の交差を避ける
このオプションは通常は有効になっており、グラフの線が交差して混乱しないように表示されます。しかし、これは列が論理的でない場所に配置される可能性もあります。たとえば、列よりも大きい斜線があると、グラフの描画に必要な領域が多くなってしまいます。これが問題になる場合には、**表示**メニューからオプションを無効にすることができます。

パス名を差分で表示
長いパス名はたくさんのスペースを取ってしまい、ノードボックスを非常に大きくしてしまいます。パスの変更された部分のみを表示し、共通部分を点に置換する場合、このオプションを使用してください。つまり、/trunk/doc/html から ブランチ /branches/1.2.x/doc/html を作成した場合、/branches/1.2.x/.. という短縮形で表示します。最後のふたつの階層(doc と html)が変更されていないからです。

すべてのリビジョンを表示
予想通り、(グラフ化したツリーの) 変更されたすべてのリビジョンを表示します。履歴が長いと、非常に大きなグラフになります。

最新リビジョン(HEAD)を表示
これにより、各ブランチの最新リビジョンが、グラフ上に常に現れることを保証します。

正確なコピー元
ブランチ・タグが作成された際、デフォルトの挙動では、変更があった最後のノードからのブランチとして表示します。ブランチは特定のリビジョンからよりも、その時の HEAD リビジョンから作成されますので、厳密に言うと正しくありません。そこで、コピーを作成したリビジョンを使用して、より正しい(しかしあまり有用でない)表示を行えます。このリビジョンが、ソースブランチの HEAD リビジョンよりも古い可能性があることに注意してください。

タグを折りたたむ
プロジェクトに多くのタグがある場合、それぞれのタグがグラフ上で独立したノードとして表示されるため、開発にとってより重要なブランチの構造が読み取りにくくなります。一方、リビジョン間で比較を行う場合は、タグの内容に簡単にアクセスできる必要があります。このオプションは、タグのノードを非表示にする代わりに、コピー元のノード上にツールチップとして表示するようにします。コピー元のノードの右側にタグアイコンがあれば、タグが作成されたことが分かります。これで、表示をとても簡素化することができます。

なお、タグからコピーが作成された場合、通常は新しいブランチがタグから作成された場合ですが、タグはまとめられずに独立したノードとして表示されます。

削除されたパスを隠す
リポジトリの HEAD リビジョンに存在していないパス(削除されたブランチなど)を非表示にします。

タグを折りたたむを選択した場合、タグの作成元になったために表示されていた削除済みのブランチは、タグと一緒に非表示になります。タグが作成された最後のリビジョンは、削除された別のリビジョンと区別するために、削除されたノードの色で表示されます。

タグを折りたたむを選択すると、タグを表示する必要がなくなったブランチは、再度非表示になります。

未使用のブランチを非表示にする
: それぞれのファイルやサブフォルダーに対して、変更をコミットしていないブランチを隠します。これは必ずしも、そのブランチが使われなかったことを表すわけではありません。ただ、この部分では変更されていないことを表します。

作業コピーのリビジョン表示
: グラフ用に取得した項目の更新リビジョンに一致する、グラフのリビジョンをマークします。更新したばかりの時は、HEAD リビジョンになるでしょうが、最後に更新してから、別の人がコミットしていると、作業コピーのリビジョンは若干古くなります。ノードは太い輪郭線でマークされます。

作業コピーの変更表示
: 作業コピーにローカルな変更がある場合、このオプションはその変更を分かれた楕円のノードとして描画し、作業コピーの最終更新リビジョンの後ろに接続します。デフォルトの輪郭色は赤です。最新の変更を取得するため、F5 を押してグラフを再表示する必要があります。

フィルター
: リビジョングラフは、時に必要以上に大量のリビジョンを含んでしまいます。このオプションではダイアログを表示し、表示するリビジョン範囲を制限したり、特定のパスを非表示にしたりといったことができます。

  特定のパスを非表示にし、そのノードに子ノードがある場合、子ノードは別なツリーとして表示されます。すべての子ノードを非表示にする場合は、**子階層すべてを削除**チェックボックスを使用して下さい。

ツリーストライプ
: 複数のツリーを含むグラフでは、それぞれのツリーを識別するのに、交互に背景色が付いていると便利です。

縮小表示
: 現在の表示ウィンドウをドラッグできる矩形で表した、グラフ全体の小さな画像を表示します。これによりグラフをもっと簡単にナビゲートできます。非常に大きいグラフの場合、極端な縮小のため役に立たない可能性があります。その場合表示されないことにご注意ください。

4.25.3. グラフの使用

大きなリビジョングラフを参照するのに、外観ウィンドウを使用してください。小さなウィンドウに、図の全体を表示し、現在の表示範囲を強調しています。強調範囲をドラッグすると、表示領域が変化します。

リビジョンの日時、作者。コメントは、マウスをリビジョンボックスの上にかざしたときに出るヒントボックスに表示します。

2つのリビジョンを選択（Ctrl-左クリック）すると、そのリビジョン間の差分を表示するコンテキストメニューを使用できます。ブランチ作成ポイントで差分を表示できますが、通常ブランチ終了ポイント（つまり HEAD リビジョン）について表示したいことでしょう。

差分をUnified差分ファイル（最小の文脈で全差分を1ファイルにまとめたもの）で見られます。**コンテキストメニュー → リビジョンを比較** を選択すると、変更したファイルの一覧が表示されます。ファイル名を ダブルクリック すると両リビジョンを取得し、視覚差分ツールで比較します。

リビジョン上で 右クリック すると、履歴を表示するのに **コンテキストメニュー → ログ表示** を使用できます。

別の作業コピーにある、選択したリビジョンの変更もマージできます。フォルダー選択ダイアログにより、マージ結果を格納する作業コピーを選択できますが、確認ダイアログはなく、また動作チェックもできません。変更されていない作業コ

ピーを用い、選択したリビジョンをマージするのに失敗したら、変更を取り消すのがよい方法です。これはあるブランチから別のブランチへ、選択したリビジョンをマージするのに便利です。

リビジョングラフの読み方

初めて見たユーザーは、ユーザーのメンタルモデルと異なるリビジョングラフに驚くかもしれません。例えばリビジョンが、ファイルやフォルダーに対して、複数のコピーやブランチを変更すると、ひとつのリビジョンから複数のノードが生成されます。ツールバーの左端から初めて、ひとつひとつメンタルモデルと合うまで、グラフをカスタマイズするのもいい実践法です。

フィルターオプションはすべて、可能な限り情報を少しも失わないように試みます。それにより、例えばいくつかのノードは色が変わる可能性があります。予期しない結果となった場合は、常に最後に行ったフィルターを取り消し、特定のリビジョンやブランチに対して何が特別であったのか、理解するように努めてください。多くの場合、はじめに予想したフィルター操作の結果は、的確でないか誤解しています。

4.25.4. 表示の更新

新しい情報を取得するためサーバーを再チェックする場合、単純に F5 で表示を更新できます。ログキャッシュを使用する場合（デフォルトで有効）、新しいログメッセージがあるかリポジトリをチェックし、新しいもののみを取得します。ログキャッシュがオフラインモードだった場合、オンラインモードにしようともします。

ログキャッシュを使用しており、メッセージの内容や作者を変更しようとする場合、必要なメッセージを再読込するのにログダイアログを使用するべきです。リビジョングラフはリポジトリのルートから動作しますので、ログキャッシュ全体を無効にせねばならず、キャッシュにためるのに非常に長い時間がかかります。

4.25.5. ツリーの剪定

大きなツリーはナビゲートしにくい場合があり、一部を隠したり、小さなツリー群に分割したりしたくなると思います。ノードに出入りするノードリンクの点の上にマウスを移動すると、このための複数のポップアップボタンを目にすることになります。

付随するサブツリーを折りたたむには、-ボタンをクリックしてください。

折りたたまれたツリーを展開するには、+ボタンをクリックしてください。ツリーが折りたたまれている場合、隠されたサブツリーを表すため、表示したままとなります。

×ボタンをクリックすると、取り付けたサブツリーを分割し、独立したツリーとしてグラフに表示します。

○ボタンを押すと、分割したツリーを再度取り付けます。ツリーが遠いところで分割された際には、分割したサブツリーがあることを示すため、このボタンを表示したままとなります。

グラフの背景をクリックすると、**すべて展開** や **すべて連結する** を提供する、メインコンテキストメニューを表示します。折りたたんだり分割したブランチがなければ、コンテキストメニューを表示しません

4.26. Subversion 作業コピーをエクスポート

時には `.svn` ディレクトリを除いてコピーすることもあるかもしれません。圧縮したソースの tarball を作成するときや、web サーバーにエクスポートするときなどです。コピーを作成し、`.svn` ディレクトリを全て手作業で削除するのではなく、TortoiseSVN では **TortoiseSVN → エクスポート...** コマンドを用意しています。URL からのエクスポートと作業コピーからのエクスポートは多少異なります。

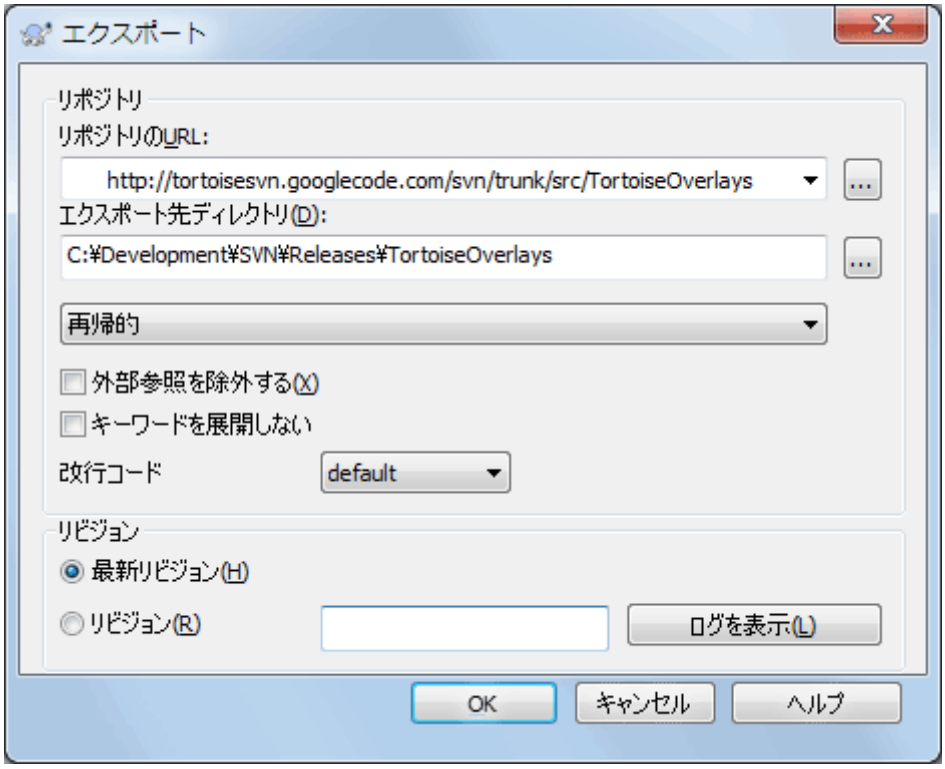


図4.57 URL からエクスポートダイアログ

バージョン管理外のフォルダーでこのコマンドを実行すると、TortoiseSVN は選択したフォルダーをターゲットと見なし、エクスポートする URL とリビジョンを入力するダイアログを開きます。このダイアログでは、最上位フォルダーのみかどうか、外部参照を省略するかどうか、`svn:eol-style` プロパティが設定されているファイルの行末を書き換えるかどうかといったオプションがあります。

またもちろん、リポジトリから直接エクスポートもできます。リポジトリブラウザーを用いて、リポジトリ内の適切なサブディレクトリを探し、**コンテキストメニュー → エクスポート** としてください。前述の **URL からエクスポート** ダイアログを表示します。

作業コピーでこのコマンドを実行すると、`.svn` フォルダーを含まない、まっさらな作業コピーを保存する場所を訊いてきます。デフォルトではバージョン管理下のファイルのみですが、**バージョン管理外のファイルもエクスポートする** チェックボックスを使用すると、リポジトリになく作業コピーにあるバージョン管理外のファイルも含めることができます。必要なら `svn:externals` を使用した外部参照を省略することもできます。

その他、作業コピーからエクスポートするには、作業コピーのフォルダーを別の場所に右ドラッグし、**コンテキストメニュー → SVN ここにエクスポート** や **コンテキストメニュー → SVN すべてをここにエクスポート** を選んでもできます。後者は、バージョン管理外のファイルもエクスポートします。

作業コピーからエクスポートする際、ターゲットフォルダーにエクスポートするフォルダーと同じ名前が存在する場合は、既存のフォルダーを上書きするか、自動的に名前を生成して新しいフォルダーを作成するか(Target　(1)など)を選択できます。

単一ファイルのエクスポート

エクスポートダイアログは、Subversion ではできる 単一ファイルのエクスポートを、許可していません。

TortoiseSVN　で単一ファイルをエクスポートするには、リポジトリブラウザーを使う必要があります (「リポジトリブラウザー」)。単純にエクスポートしたいファイルを、リポジトリブラウザーからエクスプローラーの目的の場所にドラッグするか、リポジトリブラウザーのコンテキストメニューを使ってファイルをエクスポートします。

変更したツリーのエクスポート

プロジェクトツリー構造のコピーをエクスポートしたいのに、特定のリビジョンやふたつのリビジョン間にしかないファイルのみを含めたい場合、「フォルダーの比較」 で説明する、リビジョン比較機構を使用してください。

4.26.1. 作業コピーをバージョン管理外へ

時には作業コピーから .svn ディレクトリを削除し、通常のフォルダーに戻したい場合もあることと思います。本当に必要としているのは、新しくまっさらなディレクトリツリーを生成するのではなく、コントロールディレクトリを削除するだけの、その場でエクスポートコマンドです。

答は驚くほど単純です。フォルダーを自身にエクスポートしてください! TortoiseSVN はこの特殊なケースを検出し、作業コピーをバージョン管理下から外すかどうかを聞いてきます。はい　と答えると、コントロールディレクトリを削除し、まっさらなバージョン管理されていないディレクトリになります。

4.27. 作業コピーの再配置

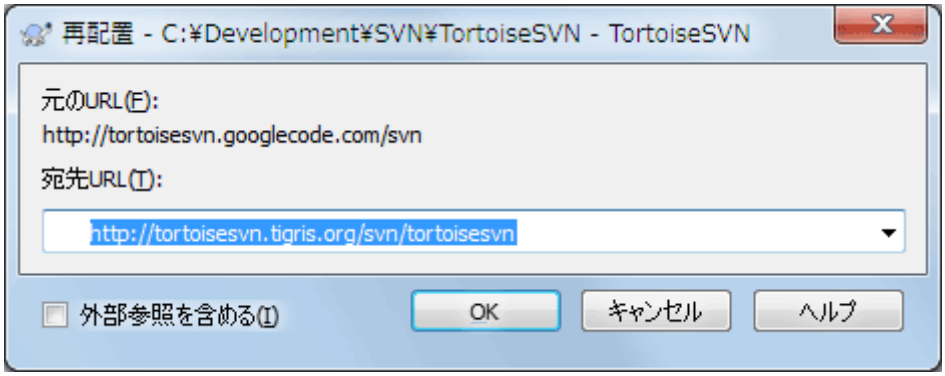


図4.58 再配置ダイアログ

もしリポジトリの位置(IP/URL)が何らかの理由で変更された場合、たぶん作業が止まってしまうでしょう。コミットができなくなり、新しい場所から再び作業コピーをチェックアウトして、変更したデータをすべて新しい作業コピーに書き戻すということはしたくないでしょう。その時に使うのは、TortoiseSVNの → 再配置コマンドです。ここで行われることは少しだけです。新しいURLで各ファイルとフォルダーに関連付けられているすべてのURLを書き換えます。

> **注記**
>
> この操作は作業コピーのルートでのみ実行できます。そのため、コンテキストメニュー項目は作業コピーのルートでのみ表示されます。

この操作の一部として、TortoiseSVN がリポジトリに接続するのに驚かれるかもしれません。これは、新しい URL が既存の作業コピーと、本当に同じリポジトリを指しているかを簡単にチェックするためだけに行います。

警告

これはほとんど行われない操作でしょう。再配置コマンドは、リポジトリのルートの URL が変更されたときのみ使用します。考えられる理由は以下のようなものでしょう。

- サーバーの IP アドレスが変更された。
- プロトコルが変更された。(http:// から https:// など)
- サーバーのセットアップでリポジトリのルートパスが変更された。

その他としては、作業コピーが同じリポジトリの同じ場所を参照していたのに、リポジトリ自身が移動してしまった時に、再配置する必要があります。

以下のような場合には使用しないでください。

- 異なる Subversion のリポジトリに移動したい。その場合は新しいリポジトリの場所からまっさらなチェックアウトを実行するべきです。
- 同じリポジトリの、異なるブランチやディレクトリに切り替えたい。この場合、TortoiseSVN → **切り替え...** を使用するべきです。詳細は、「チェックアウトするか切り替えるか...」をご覧ください。

以上のような場合に再配置を使用すると、作業コピーを破損してしまい、更新、コミット、etc. でわけの分からないエラーメッセージを見ることになります。こうなってしまったら、新しくチェックアウトするしかありません。

4.28. バグ追跡ツール／課題追跡システムとの統合

ソフトウェア開発ではふつう、変更を特定のバグIDや課題IDに結びつけます。バグ追跡ツール(課題追跡システム)を使用している場合、Subversion で行った変更を、課題追跡システムの特定のIDと関連付けできると便利です。多くの課題追跡システムではそのために、ログメッセージを解釈してコミットに関連する課題IDを抽出するための pre-commit フックスクリプトを提供しています。しかし、pre-commit フックスクリプトが正しく解釈できるようにログメッセージを書けるかどうかはユーザーに依存しているため、ときどき失敗する傾向があります。

TortoiseSVN は、次の2つの方法でユーザーを補助します。

1. ユーザーがログメッセージを入力するとき、課題IDを含む行が自動的に正しい書式で追加されるようにします。これによって、ユーザーがバグ追跡ツールで正しく解釈できない形で課題IDを入力してしまうリスクを減らせます。

   また、TortoiseSVN は入力されたログメッセージのうち、課題追跡システムが認識した部分を強調表示できます。これにより、ログメッセージが正しく解釈されたか、ユーザーが知ることができます。

2. ユーザーがログメッセージを閲覧する際に、TortoiseSVN は、ログメッセージ内の各課題IDに、課題に言及しているページをブラウザーで開くリンクを作成します。

4.28.1. ログメッセージへの課題IDの付与

TortoiseSVN は任意のバグ追跡ツールと統合できます。そのためには、bugtraq: で始まるプロパティを、フォルダーに対して定義する必要があります。(「プロジェクト設定」)

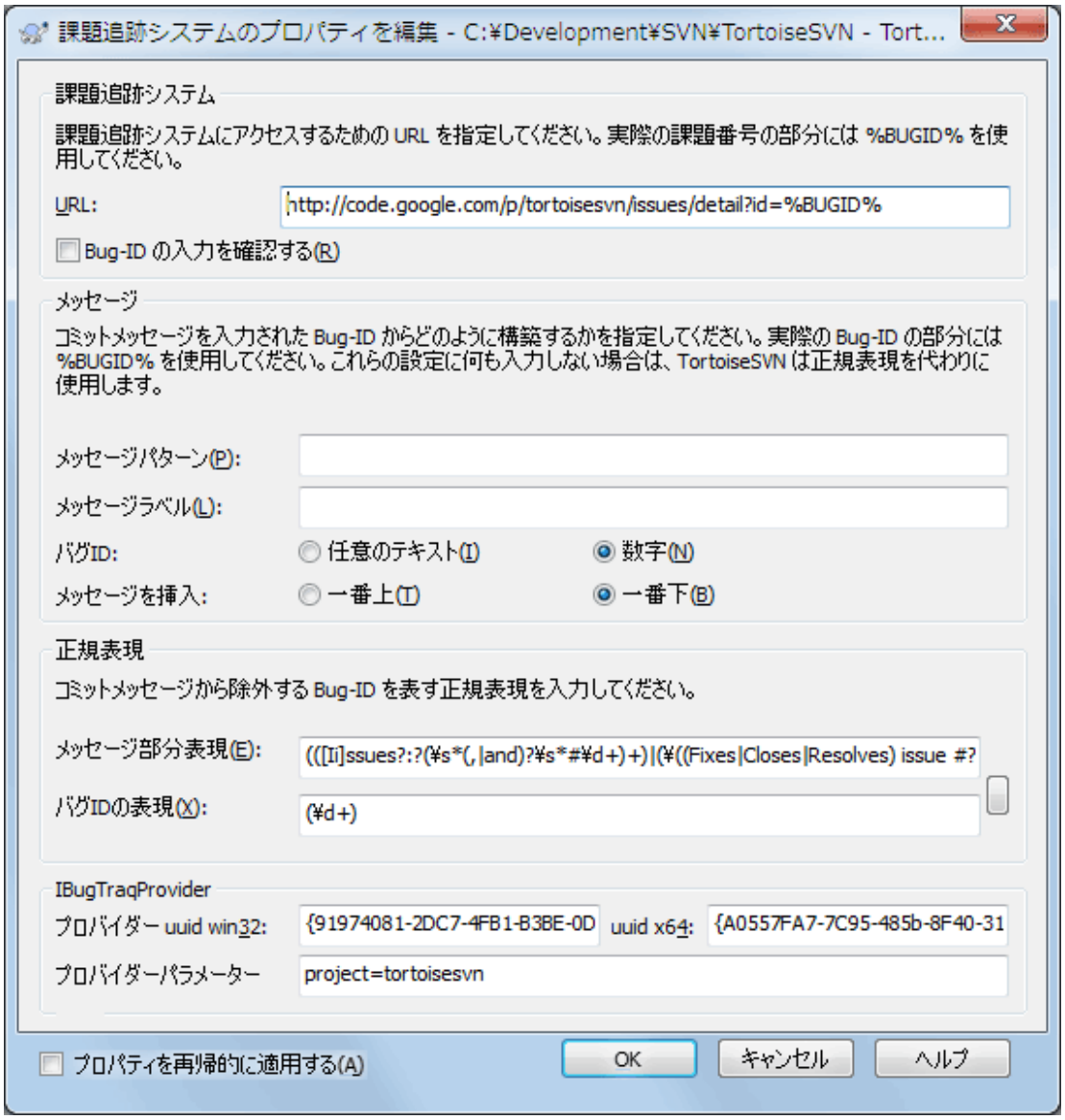


図4.59 課題追跡システムのプロパティダイアログ

課題追跡システムのプロパティを編集する場合、適切な値を簡単に設定するために、専用のプロパティエディターを使用します。

TortoiseSVN と課題追跡システムを統合するには、2つ方法があります。ひとつは単純文字列による方法、もうひとつは正規表現 による方法です。プロパティにはどちらの方法も使用できます。

bugtraq:url
: バグ追跡ツールのURLを設定します。適切にURIエンコードし、中に `%BUGID%` を含む必要があります。`%BUGID%` は入力された課題IDに置き換えられます。これによって TortoiseSVN のログダイアログ上にリンクを表示し、リビジョンログを参照する際に、バグ追跡ツールに直接ジャンプできるようになります。このプロパティがないと、TortoiseSVN は課題IDを表示するだけで、リンクを表示しません。例えば TortoiseSVN の開発プロジェクトでは、`http://issues.tortoisesvn.net/?do=details&id=%BUGID%` という設定を使用しています。

絶対URLの代わりに相対URLも使用できます。これは、課題追跡システムが、自分のソースリポジトリと同じドメインやサーバーにある場合に便利です。ドメイン名を変更したとしても、bugtraq:url プロパティを調整する必要がないからです。相対 URL を指定するには以下の2つの方法があります。

文字列が ^/ で始まる場合は、リポジトリルートからの相対 URL であると見なされます。例えば、リポジトリが http://tortoisesvn.net/svn/trunk/ にある場合、^/../?do=details&id=%BUGID% は、http://tortoisesvn.net/?do=details&id=%BUGID% と解釈されます。

/ で始まる文字列の URL は、サーバーのホスト名からの相対 URL であると見なします。例えば、リポジトリが http://tortoisesvn.net のどこかにある場合、/?do=details&id=%BUGID% は、http://tortoisesvn.net/?do=details&id=%BUGID% と解釈されます。

bugtraq:warnifnoissue
課題IDテキストフィールドが空の場合、TortoiseSVN が警告を発するようにする場合は、 true に設定してください。有効な値は true/false です。 定義されていなければ、false として扱われます。

4.28.1.1. テキストボックスを使用した課題ID

単純文字列によるアプローチでは、課題IDを入力できる独立した入力フィールドが表示されます。そして、ユーザーが入力したログメッセージに独立した行が追加されます。

bugtraq:message
入力フィールドモードでバグ追跡システムを有効にします。このプロパティを設定すると、変更をコミットする際に課題IDを入力するよう促されます。この書式でログメッセージの末尾に行が追加されます。中に %BUGID% を含んでいる必要があり、これはコミットの際に課題IDに置換されます。この機能を使用することで、コミットログに課題IDへの参照を一定の書式で確実に含め、バグ管理ツールがこれを解釈して課題IDと特定のコミットを関連付けることができるようになります。例えば Issue : %BUGID% などと設定することができますが、これは使用するツールに依存します。

bugtraq:label
このテキストは、コミットダイアログの課題IDを入力するエディットボックスのラベルとして表示されます。設定されていないと、バグID/課題番号(): bugtraq:numbertrue true/false true bugtraq:appendtrue/false true

4.28.1.2. 正規表現を使用した課題ID

正規表現 でのアプローチでは、独立した入力フィールドは表示されませんが、ログメッセージを入力中に課題追跡システムによって認識された部分をマークします。これは、ログメッセージの入力中に表示されます。これは課題IDが、ログメッセージのどこにあってもかまわないということです。この方法はとても柔軟で、 TortoiseSVN プロジェクトではこれを使用しています。

bugtraq:logregex
このプロパティを設定すると、正規表現 モードでバグ追跡システムが有効になります。ここには単一の正規表現か、改行で区切って2つの正規表現を設定できます。

正規表現を2つ設定した場合、1つ目の正規表現は、含まれている課題IDを探すための前処理フィルターとして使用します。2つ目の正規表現は、最初の正規表現の結果から、課題IDのみを抽出します。これにより、課題IDの一覧を抽出したり、自然言語表現から課題IDを抽出したりすることができます。例えば複数のバグを修正し、 「This change resolves issues #23, #24 and #25」というような文字列を入力したとします。

このログメッセージの中にある表現から課題IDを抽出するには、[Ii]ssues?:?(¥s*(,|and)?¥s*#¥d+)+ と (¥d+)のような正規表現を使用することができます(TortoiseSVN プロジェクトで使用しているもののひとつです)。

1つ目の正規表現で、ログメッセージから「issues #23, #24 and #25」の部分を抜き出します。2つ目の正規表現は、最初の正規表現の出力から、生の十進数の数値を抽出します。つまり、課題IDとして使われている、「23」,「24」,「25」が返ります。

はじめの正規表現を少しかみ砕くと、「issue」 という単語(先頭が大文字でも可)で始まっていなければなりません。この後に「s」(複数の場合)やコロンを続けることもできます。この後、0個以上の空白と、オプションのコンマまたは「and」、さらに0個以上の空白が来て、さらに「#」と十進数の数値、という形のグループが1つ以上来ます。

正規表現が1個のみの場合は、単体の課題IDと正規表現文字列のグループとがマッチしなければなりません。例: `[Ii]ssue(?:s)? #?(¥d+)`。この方法は、trac のような一部の課題追跡システムには必要ですが、その正規表現を構築するのは大変です。使用する課題追跡システムのドキュメントにある場合にのみ、この方法を使用するのをお勧めします。

正規表現になじみがなければ、http://ja.wikipedia.org/wiki/正規表現 [http://ja.wikipedia.org/wiki/%E6%AD%A3%E8%A6%8F%E8%A1%A8%E7%8F%BE] にある解説や、http://www.regular-expressions.info/ にあるオンラインのドキュメントやチュートリアルをご覧ください。

正規表現を常に正しく書くことは困難なので、手助けのために、課題追跡システムのプロパティダイアログにテストダイアログを用意しています。エディットボックスの右側のボタンをクリックすると起動します。ここに文字列を入力し、正規表現を変更すると結果を表示させることができます。正規表現が無効であれば、エディットボックスの背景が赤くなります。

`bugtraq:message` と `bugtraq:logregex` の両方を設定した場合、`logregex` が優先されます。

ヒント

ログメッセージを解釈する pre-commit フックスクリプトを持つ課題追跡システムがなくても、ログメッセージで言及している問題にリンクするために、この機能を使用することができます。

またリンクが必要なくても、課題IDがログダイアログの独立した列に表示されるので、どの課題に対応する変更かがわかりやすくなります。

`tsvn:` プロパティのいくつかは `true/false` 値をとります。TortoiseSVN は `yes` を `true` と同義に、`no` を `false` と同義に解釈します。

フォルダーへのプロパティの設定

これらのプロパティは、作業するシステムのフォルダーに設定しなければなりません。ファイルやフォルダーをコミットする際に、このフォルダーからプロパティが読み取られます。フォルダーにプロパティが設定されていない場合、プロパティはフォルダー階層を上位に向かって、バージョン管理外フォルダーか、ツリーのルート(例: `C:¥`)に到達するまで検索されます。すべてのユーザーが、例えば `trunk/` からチェックアウトし、サブフォルダーからチェックアウトしないことが確認できるのであれば、`trunk/` にそのプロパティを設定するだけで充分です。サブフォルダーからもチェックアウトする可能性があるのであれば、プロパティを各サブフォルダーに再帰的に設定するべきです。プロジェクトの深い階層で設定されたプロパティは、高いレベルの(`trunk/` に近い)ものより優先されます。

プロジェクトプロパティ(例:`tsvn:`、`bugtraq:`、`webviewer:`)に限っていえば、**プロパティを再帰的に適用する** チェックボックスを使うと、そのフォルダ以下のすべてのサブフォルダーにプロパティを設定しますが、ファイルには設定しません。

TortoiseSVN で新しいサブフォルダーを作業コピーに追加する際、親フォルダーに設定されているプロジェクトプロパティは、新しいサブフォルダーにも自動的に追加されます。

リポジトリブラウザーからは課題追跡システムの情報を利用できない

バグ追跡システムとの統合は、Subversion のプロパティとして保持されているため、チェックアウトした作業コピーを使用するときのみ表示されます。リモートからプロパティを取得するのには時間がかかるため、作業コピーからリポジトリブラウザーを起動しなければ表示されません。リポジトリのURLを入力してリポジトリブラウザーを起動した場合は表示されません。

同じ理由で、プロジェクトのプロパティは、リポジトリブラウザーを使用して子フォルダーを追加しても、自動的には継承されません。

This issue tracker integration is not restricted to TortoiseSVN; it can be used with any Subversion client. For more information, read the full Issue Tracker Integration Specification [http://tortoisesvn.googlecode.com/svn/trunk/doc/notes/issuetrackers.txt] in the TortoiseSVN source repository. (「ライセンス」 explains how to access the repository.)

4.28.2. 課題追跡システムからの情報取得

前節では、ログメッセージへ課題情報の追加について扱いました。しかし、課題追跡システムから情報を取得する必要がある場合、どのようにしたらよいのでしょう?コミットダイアログは、課題追跡システムと対話できるような、外部プログラムを統合するためのCOMインターフェイスを備えています。通常、自分に割り当てられた未解決の課題の一覧を問い合わせて、このコミットに関連する課題を選択できるといいのではないでしょうか。

そのようなインターフェイスはいずれも、課題追跡システムに強く依存しているので、私たちはこの部分を提供することはできませんし、そのようなプログラムの作成方法について説明することは、このマニュアルの範疇を超えてしまいます。インターフェイスの定義や、C#やC++/ATLで書かれたサンプルプラグインは、TortoiseSVN リポジトリ [http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins]の `contrib` フォルダーから取得できます(リポジトリのアクセスの仕方は、「ライセンス」で説明します)。API の概要も 6章IBugtraqProvider インターフェイス で得られます。それ以外の(動作する)サンプルプラグインとして、C# では Gurtle [http://code.google.com/p/gurtle/] があり、これは課題管理システム Google Code [http://code.google.com/hosting/] と連動するために必要な COM インターフェイスを実装しています。

説明のために、システム管理者が、インストールした課題管理システムプラグインを提供し、TortoiseSVN の設定ダイアログでそのプラグインを使用する作業コピーをセットアップしたとしましょう。プラグインに割り当てられた作業コピーからコミットダイアログを開くと、ダイアログの上部に新しいボタンが現れるはずです。

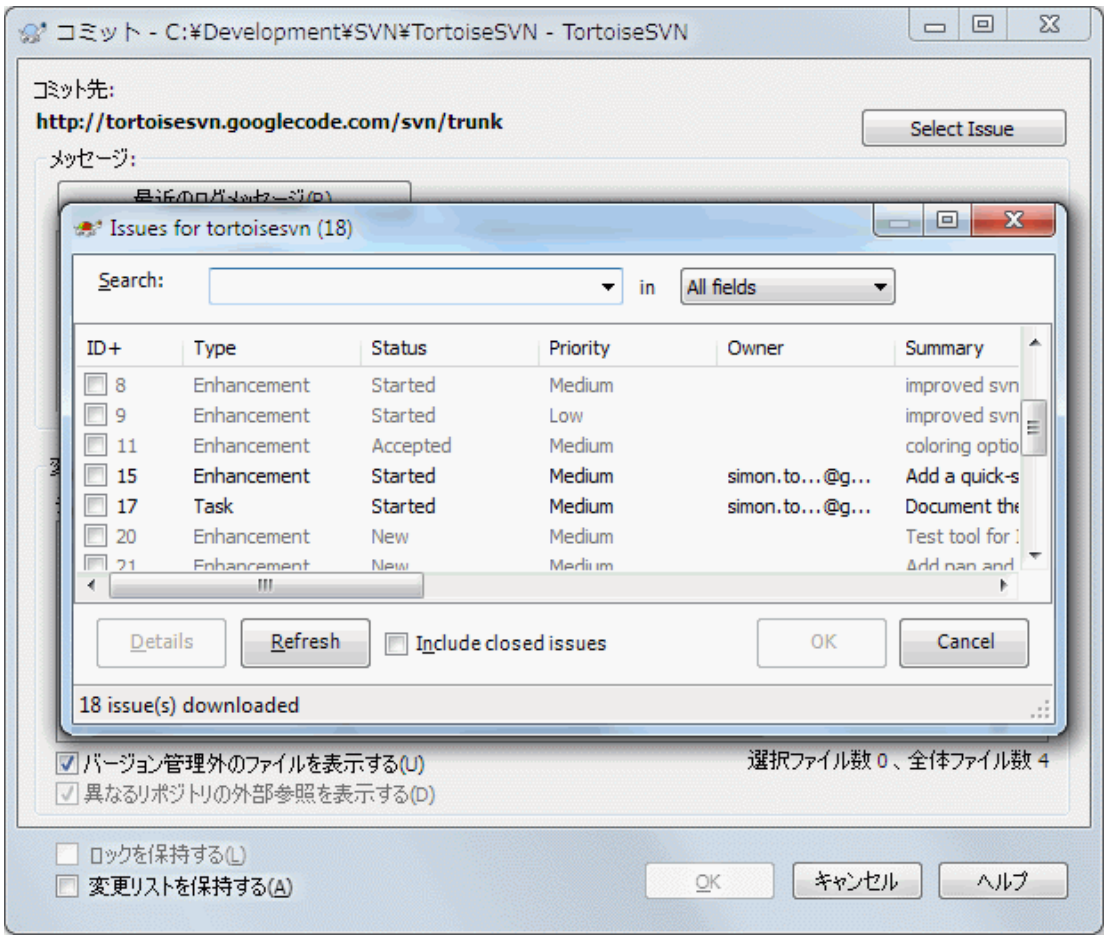


図4.60 課題追跡システムクエリダイアログの例

この例では、複数の未解決課題を選択できます。その後、プラグインがログメッセージに追加する、特定のフォーマットのテキストを生成できます。

4.29. Web ベースのリポジトリビューアーとの統合

Subversion のリポジトリビューアーとして使用することができるウェブベースのシステムとして、ViewVC [http://www.viewvc.org/] や WebSVN [http://websvn.tigris.org/] などがあります。TortoiseSVN は、そういったビューアーと連携することができます。

TortoiseSVN は任意のリポジトリビューアーと統合できます。そのためには、リンク用に用意されているプロパティを定義する必要があります。以下のようにフォルダーに設定しなければなりません。(「プロジェクト設定」)

webviewer:revision
: リビジョンを指定して、その変更点を表示させるために使用する URL を指定します。適切に URI エンコードし、%REVISION% という文字列を含んでいる必要があります。%REVISION% の部分は、リビジョン番号に置き換わります。このプロパティを設定すると、ログダイアログのコンテキストメニューに、**コンテキストメニュー → リビジョンをウェブビューアーで表示** が表示されるようになります。

webviewer:pathrevision
: ファイルとリビジョンを指定して、その変更点を表示させるために使用する URL を指定します。適切に URI エンコードし、%REVISION% と %PATH% という文字列を含んでいる必要があります。%PATH% の部分は、リポジトリのルートからの相対パスに置き換わります。このプロパティを設定すると、ログダイアログのコンテキストメニューに、**コンテキストメニュー → パスのリビジョンをウェブビューアーで表示** が表示されるようになり

ます。例えば、ログダイアログの下の欄に表示された /trunk/src/file を右クリックすると、URL の中の %PATH% の部分が /trunk/src/file に置き換わります。

絶対 URL の代わりに相対 URL も使用できます。これは、ウェブビューアーが、ソースリポジトリと同じドメイン・サーバーにある場合に便利です。ドメイン名を変更したとしても、webviewer:revision プロパティや webviewer:pathrevision プロパティを調整する必要はありません。フォーマットは bugtraq:url プロパティと同じです。「バグ追跡ツール／課題追跡システムとの統合」をご覧ください。

フォルダーへのプロパティの設定

これらのプロパティは、作業するシステムのフォルダーに設定しなければなりません。ファイルやフォルダーをコミットする際に、このフォルダーからプロパティが読み取られます。フォルダーにプロパティが設定されていない場合、プロパティはフォルダー階層を上位に向かって、バージョン管理外フォルダーか、ツリーのルート(例: C:¥)に到達するまで検索されます。すべてのユーザーが、例えば trunk/ からチェックアウトし、サブフォルダーからチェックアウトしないことが確認できるのであれば、trunk/ にそのプロパティを設定するだけで充分です。サブフォルダーからもチェックアウトする可能性があるのであれば、プロパティを各サブフォルダーに再帰的に設定するべきです。プロジェクトの深い階層で設定されたプロパティは、高いレベルの(trunk/ に近い)ものより優先されます。

プロジェクトプロパティ(例:tsvn:、bugtraq:、webviewer:)に限っていえば、**プロパティを再帰的に適用する**　チェックボックスを使うと、そのフォルダ以下のすべてのサブフォルダーにプロパティを設定しますが、ファイルには設定しません。

TortoiseSVN で新しいサブフォルダーを作業コピーに追加する際、親フォルダーに設定されているプロジェクトプロパティは、新しいサブフォルダーにも自動的に追加されます。

リポジトリブラウザーを使用する際の制限

リポジトリビューアーとの統合は、Subversion のプロパティとして保持されているため、チェックアウトした作業コピーを使用するときのみ表示されます。リモートからプロパティを取得するのには時間がかかるため、作業コピーからリポジトリブラウザーを起動しなければ表示されません。リポジトリの URLを入力してリポジトリブラウザーを起動した場合は表示されません。

同じ理由で、プロジェクトのプロパティは、リポジトリブラウザーを使用して子フォルダーを追加しても、自動的には継承されません。

4.30. TortoiseSVN の設定

マウスポインタをしばらくエディットボックスやチェックボックスなどの上に置いておくと、ヘルプのツールチップがポップアップして、設定がなんのために変更できるのかが分かります。

4.30.1. 一般設定

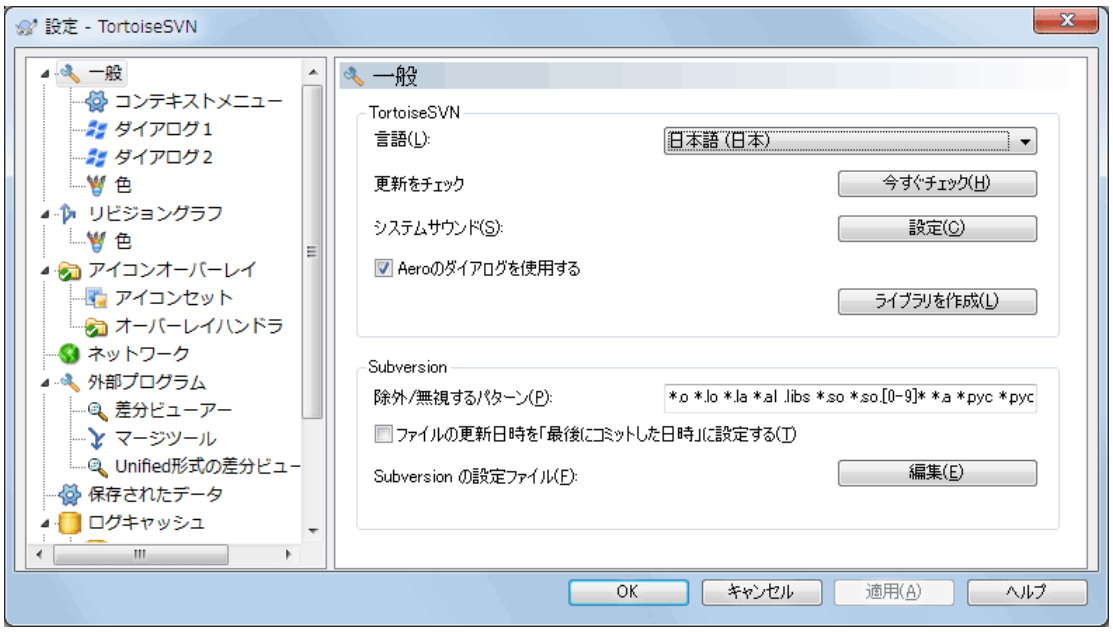


図4.61 設定ダイアログの「全般」ページ

このダイアログでは、好みの言語を指定したり、Subversion 特有の設定を行うことができます。

言語
: ユーザーインターフェイスの言語を選択してください。他に何かありますか?

更新をチェック
: チェックすると、TortoiseSVN　は週に一度、ダウンロードサイトに接続して、プログラムの新バージョンがリリースされているかを調べます。結果をすぐに知りたければ、**今すぐチェック** を押してください。新バージョンがダウンロードされるわけではありません。単に新バージョンがあるという、情報ダイアログが表示されるだけです。

システムサウンド
: TortoiseSVN にはデフォルトで3つのカスタムサウンドがインストールされています。

  - エラー
  - 注意
  - 警告

  Windows のコントロールパネルで別のサウンド(もしくは完全に OFF)を選択できます。**設定**はコントロールパネルへのショートカットです。

Aero のダイアログを使用
: Windows Vista 以降のシステムでは、ダイアログに Aero スタイルを使用するかどうか制御することができます。

ライブラリを作成
: Windows 7 では、システムのさまざまな場所に散在している作業コピーをグループ化するためのライブラリを作成することができます。

常に無視するパターン

: 常に無視するパターンは、バージョン管理外のファイルを、コミットダイアログなどで表示されないようにするのに使用します。パターンにマッチしたファイルは、インポートでも無視されます。除外するファイルやディレクトリは、名前や

拡張子で判断されます。パターンは空白で区切り、 bin obj *.bak *.~?? *.jar *.[Tt]mp のようにします。このパターンにはパス区切り文字を含めないでください。また、ファイルとディレクトリを区別することができないことに注意してください。パターンマッチの文法の詳細は、「無視リストでのパターンマッチ」 を参照してください。

ここで指定した無視パターンは、同じPCで動作する他の Subversion クライアント(コマンドラインクライアントを含む)でも有効になることに注意してください。

注意

Subversion の設定ファイルで global-ignores パターンを設定していると、この設定を上書きしてしまいます。Subversion の設定ファイルには、下にある **編集** ボタンでアクセスできます。

この無視パターンはすべてのプロジェクトに影響を与えます。バージョン管理されませんので、他のユーザーには影響しません。一方、バージョン管理の svn:ignore プロパティを使用し、バージョン管理からファイルやディレクトリを除外することもできます。詳細は「ファイルやディレクトリの無視」をご覧ください。

ファイルの更新日時を「最後にコミットした日時」に設定

このオプションは、チェックアウトや更新時にコミット時の日時をファイルの更新日時として使うよう、TortoiseSVN に指示します。そうでなければ、TortoiseSVN は現在の日時を使用します。ソフトウェアを開発しているなら、ビルドシステムがコンパイルの要否を判断するのにタイムスタンプを使用するので、現在の日時にしておくのが最善でしょう。「最後にコミットした日時」 を使用していると、ファイルを古いリビジョンに戻す場合、プロジェクトがコンパイルされない可能性があります。

Subversion 設定ファイル

Subversion 設定ファイルを直接編集するには **編集** を使用してください。いくつかの設定は TortoiseSVN では直接変更することができないため、その場合にこれが必要です。Subversion の config ファイルについての詳細情報は、Runtime Configuration Area [http://svnbook.red-bean.com/en/1.7/svn.advanced.confarea.html] をご覧ください。Automatic Property Setting [http://svnbook.red-bean.com/en/1.7/svn.advanced.props.html#svn.advanced.props.auto] の節には興味深い項目があり、これはここで設定します。Subversion は複数の設定情報を読み込めますが、その優先順位を知っておかなくてはならないことに注意してください。詳細は Configuration and the Windows Registry [http://svnbook.red-bean.com/en/1.7/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] を参照してください。

更新時に svn:externals にローカルの変更を適用する

このオプションは、TortoiseSVN が作業コピーを更新する時に、svn:externals プロパティにローカルで行われた変更を常に適用することを示します。

4.30.1.1. コンテキストメニュー設定

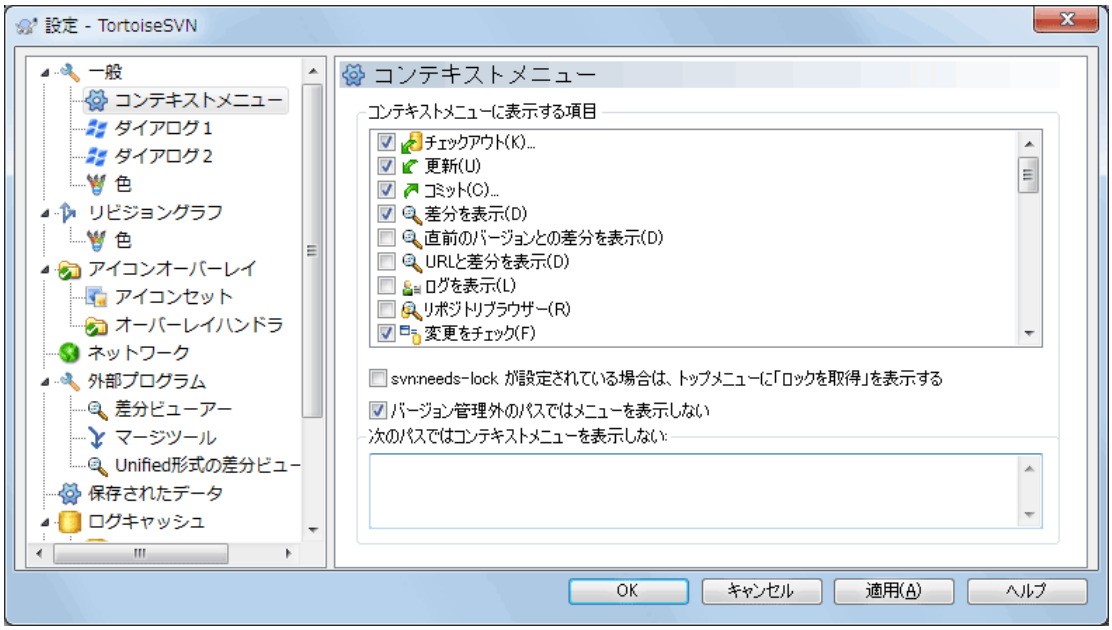


図4.62 設定ダイアログ、「コンテキストメニュー」ページ

このページでは、メインコンテキストメニューと TortoiseSVN サブメニューに現れる TortoiseSVN コンテキストメニューエントリを指定できます。デフォルトでは、ほとんどの項目にチェックが付いておらず、サブメニューに表示されません。

ロックを取得 は特殊です。もちろん上のリストを用いてトップレベルに表示できますが、ロックの必要がないほとんどのファイルにとっては乱雑なだけです。しかし、`svn:needs-lock` プロパティが設定されているファイルには、いつでも編集できるように、このアクションが必要です。この場合トップレベルに表示している方が便利です。このボックスをチェックしておくと、`svn:needs-lock` プロパティが設定されているファイルを選択すると、**ロックを取得** がいつでもトップレベルに表示されるようになります。

Subversion のバージョン管理下にあるフォルダーを離れているときは、ほとんどの場合、TortoiseSVN のコンテキストメニューを使用しません。バージョン管理外のフォルダーで本当にコンテキストメニューを必要とするのは、チェックアウト時くらいでしょう。**バージョン管理外のパスではメニューを表示しない**のオプションをチェックすると、 バージョン管理外フォルダーのコンテキストメニューでは、TortoiseSVN のメニュー項目が表示されなくなります。ただし、バージョン管理下のフォルダーでは、すべての項目を表示します。 Shift を押しながらコンテキストメニューを表示させると、バージョン管理外のフォルダーでもすべてのメニュー項目を表示させることができます。

TortoiseSVN のコンテキストメニューにも現れて欲しくない、コンピューター上のパスがある場合、下にあるボックスの項目を一覧できます。

4.30.1.2. TortoiseSVN ダイアログ設定1

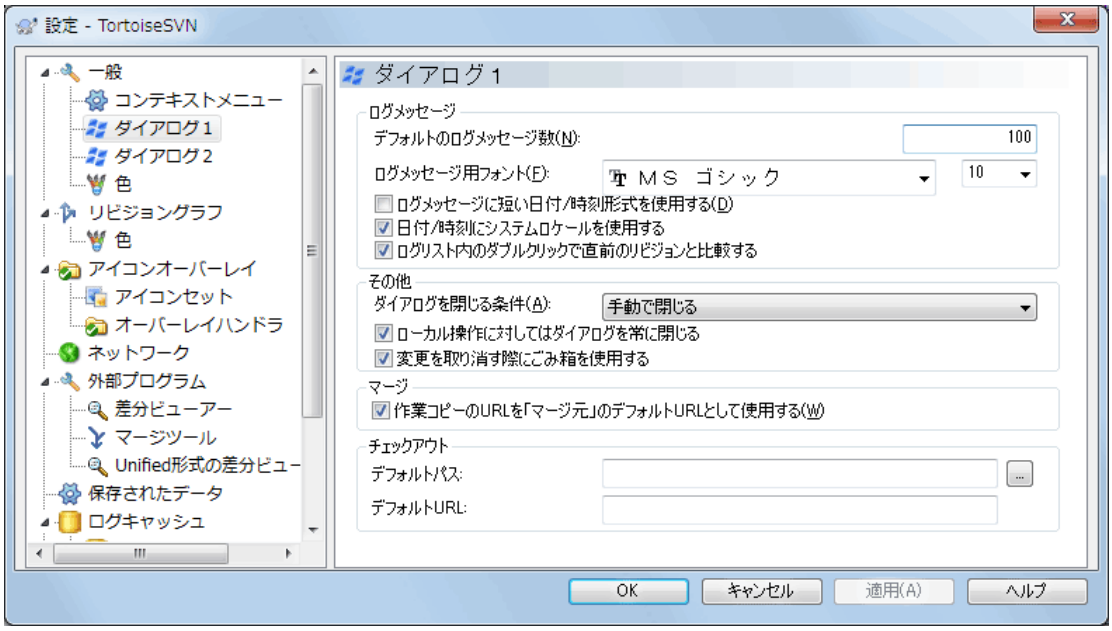


図4.63 設定ダイアログ、「ダイアログ1」ページ

このダイアログでは、TortoiseSVN のダイアログをお好みに設定できます。

デフォルトのログメッセージ数
: 最初に TortoiseSVN → ログを表示 を選択したとき、TortoiseSVN が取得するログメッセージをこの数に制限します。サーバーへの接続が遅い場合に便利です。もっとメッセージを取得するには 全て表示 や 次の100件 を使用してください。

ログメッセージ用フォント
: リビジョンログダイアログの中央の欄で、ログメッセージが表示される際に使用されるフォントとサイズを指定します。これは、コミットダイアログでログメッセージを編集する際にも使われます。

ログメッセージに短い日時形式を使用する
: 標準の長いメッセージ形式で使用するには画面が狭い場合、短い形式を使用します。

ログリスト内でダブルクリックすることで過去のリビジョンと比較する
: ログダイアログの上の欄のリストを使用してリビジョンの比較を頻繁に行う場合は、このオプションを有効にすると、ダブルクリックで比較できるようになります。差分の取得には時間がかかるため、デフォルトでこのオプションは有効になっていません。多くの人は、間違ってダブルクリックした際に長時間待たされるのを避けたいということから、このオプションをデフォルトで有効にしていません。

ダイアログを閉じる条件
: TortoiseSVN のコマンドがエラーがなく終了すると、進行ダイアログを自動で閉じることができます。この設定でダイアログを閉じる条件を選択できます。デフォルト（推奨）設定は**手動で閉じる**で、全てのメッセージを見て、何が起きたのか確認できます。しかし、ある種のメッセージは無視して、重要な変更がなければ自動で閉じてもらいたいということもあります。

  マージ、追加、削除がなければ自動的に閉じるは、単純な更新のみの場合は進行ダイアログを閉じますが、リポジトリからの変更をマージする必要がある場合や、ファイルの追加・削除がある場合はダイアログを開いたままにします。また処理中に、なんらかの競合やエラーが発生した場合にも開いたままになります。

競合がなければ自動的に閉じるはもっと基準をゆるめ、マージ、追加、削除が起きてもダイアログを閉じますが、競合が発生した場合はダイアログを開いたままにします。

エラーがなければ自動的に閉じるは、競合が発生しても、Subversion が処理を継続できないようなエラーが発生しない限りダイアログを閉じます。例えば、サーバーにアクセスできずに更新が失敗したり、作業コピーが最新ではなくコミットができない場合などです。

ローカル操作に対してはダイアログを常に閉じる
: ファイルの追加や変更の取り消しといったローカル操作は、リポジトリへのアクセスは必要なく、短時間で完了します。そのため、進行状況ダイアログは重要でないことがしばしばあります。エラーが発生しない時に限り、これら操作後に自動で進行状況ダイアログを閉じる場合は、このオプションを選択してください。

変更を取り消す際にごみ箱を使用する
: ローカルの変更を取り消す際、既に行っていた変更は破棄されます。TortoiseSVN では安全のために、元のコピーに戻す前に、変更したファイルをごみ箱に入れます。ごみ箱に入れるのをスキップしたい場合は、このオプションのチェックを外してください。

作業コピーの URL を「元:」のデフォルトURLとして使用する
: マージダイアログでは、デフォルトの動作は 元: の URL を記憶している所にマージします。しかし、階層内のいくつもの違う場所からマージを実行したい人もいますし、現在の作業コピーの URL からはじめる方が簡単な場合があります。また、別のブランチにある平行したパスを参照しながら編集するのにも使用できます。

デフォルトチェックアウトパス
: チェックアウトのデフォルトパスを指定できます。チェックアウトした物をすべて一カ所に保持しておく場合、ドライブやフォルダーをあらかじめ設定しておいてくれ、最後に新しいフォルダー名を追加するだけですみます。

デフォルトチェックアウト URL
: チェックアウトのデフォルト URL も指定できます。非常に大きいプロジェクトのサブプロジェクトをちょくちょくチェックアウトする場合、URL をあらかじめ設定しておいてくれ、最後にサブプロジェクト名を追加するだけですみます。

4.30.1.3. TortoiseSVN ダイアログ設定2

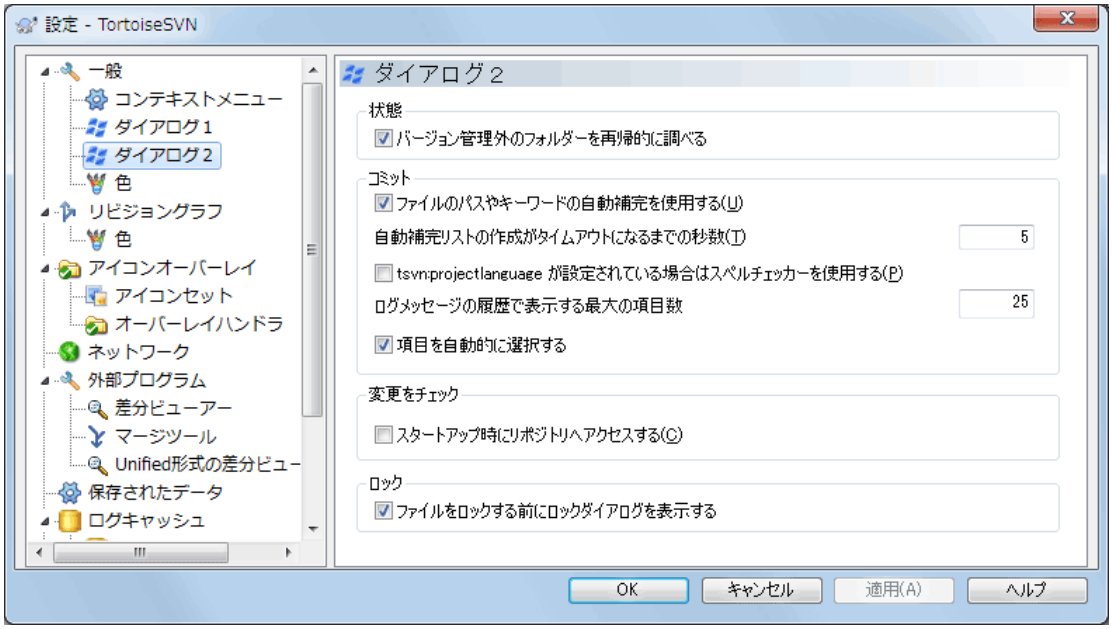


図4.64 設定ダイアログ、「ダイアログ2」ページ

バージョン管理外のフォルダーを再帰的に調べる

チェックすると（デフォルト状態）、**追加、コミット、変更をチェック**ダイアログに、バージョン管理外のフォルダーやそれ以下の階層のファイルやフォルダがすべて表示されます。チェックしないと、以下の階層は表示されません。チェックを外せば、上記のダイアログが煩雑にならずに済みます。この場合、バージョン管理外のフォルダーを追加するよう選択すると、再帰的に追加されます。

変更をチェック　ダイアログでは、無視される項目を表示するか選択できます。このチェックボックスにチェックして、無視されたフォルダーを表示すると、その中にある項目も同様に見られるようになります。

ファイルのパスやキーワードの自動補完を使用する

コミットダイアログはコミットするファイル名の一覧を保持しています。一覧にある項目の始めの 3 文字をタイプすると、自動補完ボックスがポップアップし、Enter を押すとファイル名が補完されます。チェックするとこの機能が有効になります。

自動補完のパースがタイムアウトするまでの時間(秒)

大きなファイルをチェックしていて、自動補完パーサが遅すぎることがあります。コミットダイアログが長い時間止まってしまうのなら、ここで指定した時間でタイムアウトするようになります。重要な自動補完情報を見逃してしまうのなら、タイムアウトを長くできます。

`tsvn:projectlanguage` が設定されている場合はスペルチェッカーを使用する

コミット時にスペルチェッカーを使用しない場合、ここにチェックをつけてください。プロジェクトプロパティで必要だと指定したときだけ、動作するようになります。

ログメッセージの履歴で表示する最大の項目数

コミットダイアログでログメッセージを入力すると、TortoiseSVN　は後で再利用できるようにメッセージを記憶します。デフォルトでは、リポジトリごとに最新のログメッセージを 25 個記憶していますが、ここでその数をカスタマイズできます。たくさんの異なるリポジトリがあるなら、レジストリがいっぱいにならないように、少なくしておくといいでしょう。

この設定は、このコンピューターで入力したメッセージにのみ適用されることに、ご注意ください。ログキャッシュには何も影響を与えません。

コミットに失敗した場合はコミットもしくはブランチ/タグダイアログを再表示する

なんらかの理由でコミットが失敗（作業コピーの更新が必要、pre-commitフックによるコミットのリジェクト、ネットワークエラー,他）する場合、このオプションを使用することで再度コミットできるようにコミットダイアログを開いた状態にできます。

項目を自動的に選択する

コミットダイアログの通常動作は、全ての（バージョン管理下の）変更があるファイルをコミットするよう、自動的に選択します。はじめに何も選択せず、コミットする項目を手作業で選択する場合、このチェックボックスのチェックを外してください。

スタートアップ時にリポジトリへアクセスする

デフォルトでは「変更をチェック」ダイアログで作業コピーのチェックが行われ、**リポジトリをチェック**がクリックされた時だけリポジトリへの接続を行います。常にリポジトリをチェックするようにしたい場合、これを設定すると、この動作が自動的に行われるようになります。

ファイルをロックする前にロックダイアログを表示する

ひとつないし複数のファイルを選択し、TortoiseSVN → **ロック** を使用してそのファイルをロックする際、プロジェクトによっては、なぜそのファイルをロックしたかを説明するロックメッセージを書くことになっています。ロックメッセージを使用しない場合には、このチェックボックスのチェックをはずすと、ダイアログをスキップし、すぐにファイルをロックします。

フォルダーにたいしてロックコマンドを使用する場合、ロックするファイルを選択するため、常にロックダイアログを表示します。

プロジェクトで tsvn:lockmsgminsize プロパティを使用している場合、そのプロジェクトはロックメッセージが 必須なので、設定がどうであってもロックダイアログを表示します。

4.30.1.4. TortoiseSVN の色の設定

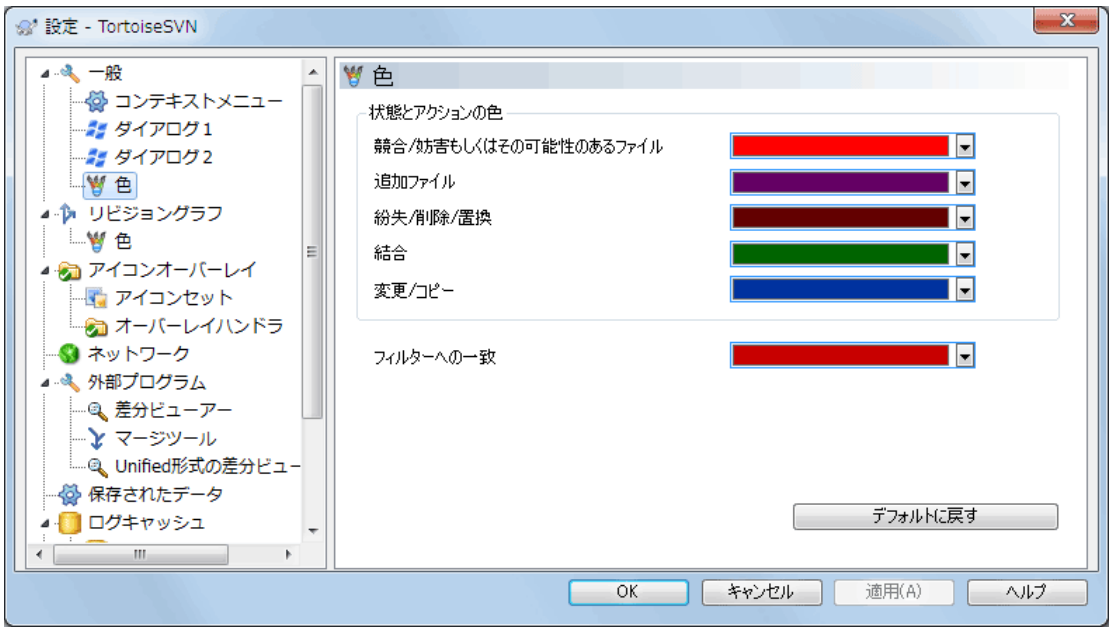


図4.65 設定ダイアログ、「色」ページ

このダイアログでは、TortoiseSVN のダイアログで使用するテキストの色をお好みに変更できます。

競合・妨害
: 更新中やマージ中に発生した競合。既存のバージョン管理外のファイル・フォルダーと同じ名前のバージョン管理下のものがあると更新は妨害されます。

  進行ダイアログのエラーメッセージにも使用します。

追加したファイル
: リポジトリに追加した項目。

紛失・削除・置換
: リポジトリから削除された項目、作業コピーから紛失した項目、作業コピーから削除された項目、同じ名前で置き換えられた項目。

マージ終了
: リポジトリからの変更を、競合もなく作業コピーにうまくマージしました。

変更・コピー
: 履歴に追加されるかリポジトリ内のパスをコピーした項目。ログダイアログでも、コピーされた項目を含む場合に使用されます。

削除したノード
: リポジトリからすでに削除された項目。

ノードを追加
: (追加・コピー・移動操作で) リポジトリに追加した項目。

名前を変更されたノード
: リポジトリで名前の変更があった項目。

置換
: 元の項目が削除され、同じ名前で新しく作成された項目。

絞り込みへの一致
: ログダイアログで絞り込みをする際、検索語は検索結果中でこの色で強調表示されます。

4.30.2. リビジョングラフの設定

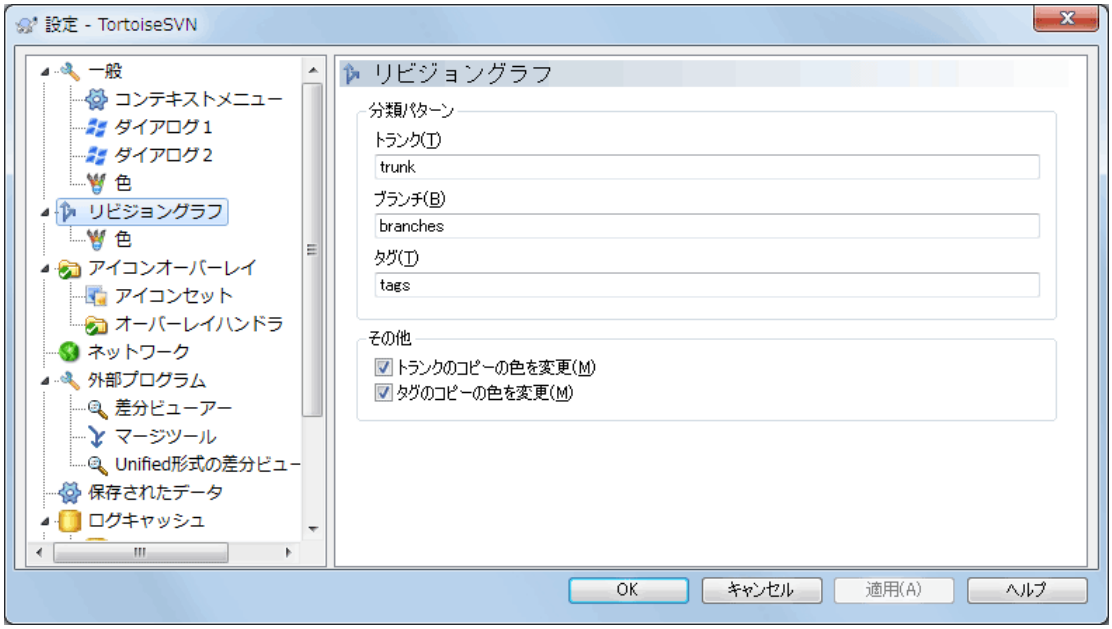


図4.66 設定ダイアログ、「リビジョングラフ」ページ

分類パターン
: リビジョングラフは、トランク、ブランチ、タグを見分けることで、リポジトリを明確に可視化しようとします。これは Subversion に組み込まれた分類ではなく、パス名から抽出した情報です。デフォルト設定では、Subversion のドキュメントで推奨されている英語名規約ですが、もちろん使い方はことなるかもしれません。

  三つの矩形で表すパスを、認識するため使用するパターンを指定してください。パターンは大文字小文字を区別しませんが、小文字で書いてください。* や ? といったワイルドカードはいつものように使用できます。また、; で複数のパターンを区切ってください。比較に使用されてしまうため、余計な空白は含めないでください。

色の変更
: リビジョングラフで、ノードタイプ(つまり追加、削除、名前変更といったノード)を表すのに使用する色です。ノードの分類を識別するため、ノードタイプと分類の両方を表示するよう、リビジョングラフに色をませるよう指示できます。このチェックが付いていないと、ノードタイプを表すためだけに色を使用します。使用するそれぞれの色を指定するには、色選択ダイアログを使用してください。

4.30.2.1. リビジョングラフの色

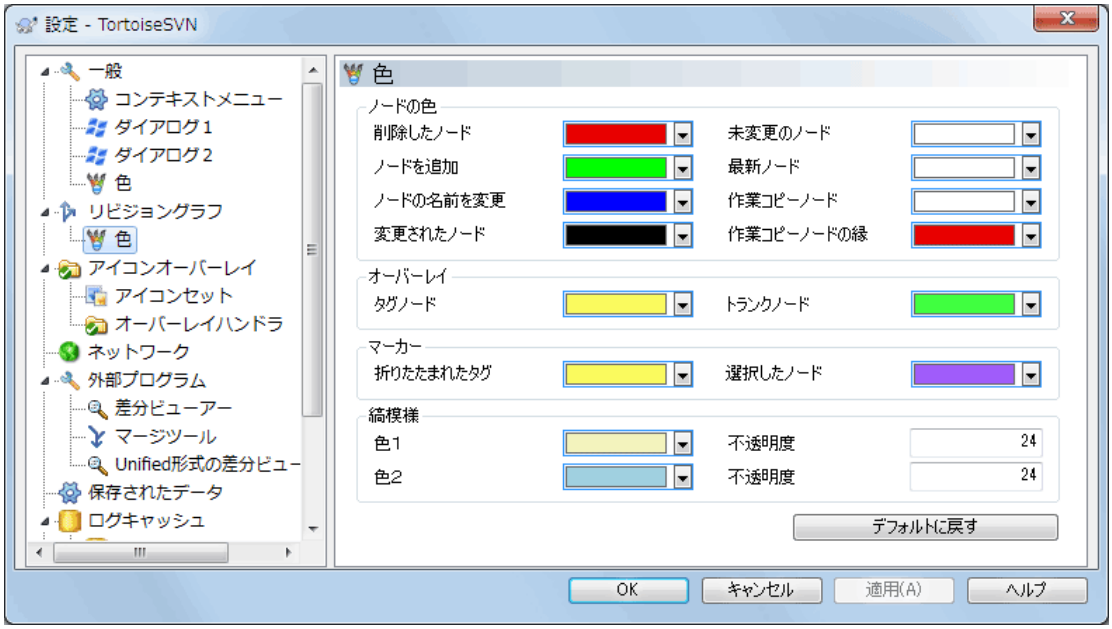


図4.67 設定ダイアログ、リビジョングラフの「色」ページ

このページでは使用する色を設定できます。ここで指定する色は、無地の色であることにご注意ください。ほとんどのノードでは、ノードタイプ色と背景色、必要に応じて分類色を混ぜた色で着色されます。

削除したノード
: 削除され、同じリビジョンのどこにもコピーされていない項目です。

追加したノード
: 新しく追加されたか、コピー(履歴とともに追加)された項目です。

名前変更ノード
: ある場所から削除され、同じリビジョンの別の場所に追加された項目です。

変更ノード
: 追加も削除もされず、単純に変更された項目です。

未変更ノード
: (グラフに表示する)変更がまったくそのリビジョンで発生しなかったとしても、コピー元として使用されるリビジョンを表示するのに使用される場合があります。

最新リビジョン(HEAD)ノード
: 現在のリポジトリの HEAD リビジョンです。

作業コピーノード
: 変更された作業コピー向けに追加ノードを表示するよう選択した場合、グラフに追加された最終コミットリビジョンは、この色を使用します。

作業コピーノードの縁
: 作業コピーが変更されたことを表示するように選択した場合、変更を検出すると作業コピーの境界線にこの色を使用します。

タグノード
: タグとして分類されたノードは、この色が混ざることがあります。

トランクノード
: トランクとして分類されたノードは、この色が混ざることがあります。

たたまれたタグマーカー
: 領域を節約するためにタグを折りたたんだ場合、コピー元において、タグはこの色のブロックを使用してマークされます。

選択したノードマーカー
: ノードを選択するためクリックしたままにした場合、選択状態を表すマーカはこの色のかたまりとなります。

縞模様
: グラフがサブツリーに分割された時にその色を使い、背景にその他の色が付いている場合、分割ツリーを際だたせるため、縞模様にします。

4.30.3. アイコンオーバレイ設定

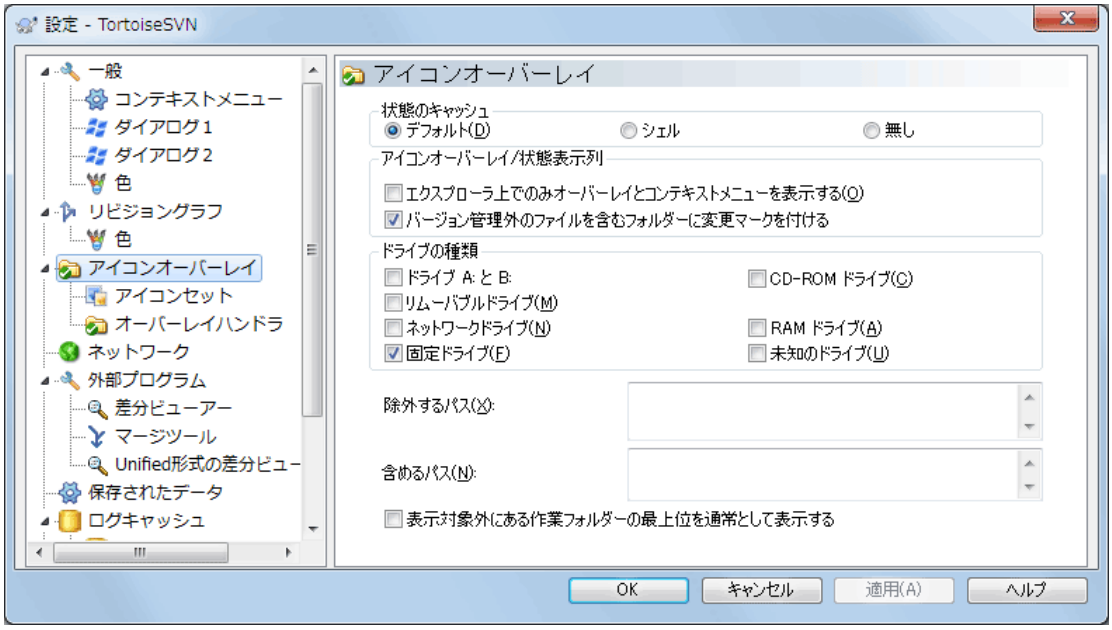


図4.68 設定ダイアログ、「アイコンオーバーレイ」ページ

このページでは、TortoiseSVN が表示するアイコンオーバーレイの項目を選択できます。

作業コピーの状態を取得するには時間がかかるため、エクスプローラーがオーバーレイアイコンを表示するのに負荷がかからないように、 TortoiseSVN は状態をキャッシュに格納します。システムや作業コピーの大きさをもとに、TortoiseSVN が使用するキャッシュタイプを以下から選択してください。

デフォルト
: 別のプロセス(TSVNCache.exe)がすべての状態情報をキャッシュします。このプロセスは、全ドライブの変更を監視し、作業コピー内のファイルが更新されれば、その状態情報を再取得します。優先度が最低で動作しますので、他のプログラムが、これによりリソースをとられてしまうことはありません。これは状態情報が リアルタイム で更新されないということでもありますが、オーバーレイ表示は数秒で更新されます。

  【利点】オーバーレイには再帰的な状態が表示されます。つまり、作業コピーの深い階層にあるファイルを更新すると、作業コピーの最上位までの全フォルダーにも更新オーバーレイが表示されます。さらにプロセスからシェルに通知が送られるので、通常、左のツリービューのオーバーレイ表示も変更されます。

【欠点】プロジェクトについての作業をしていないときでも、プロセスが常に動作しています。さらに、作業コピーの数や大きさにも依存しますが、およそ10～50MBの RAM を消費します。

シェル
シェル拡張 DLL の内部で直接キャッシュしますが、表示されているフォルダーのみをキャッシュします。別のフォルダーに移るたびに、ステータス情報を再取得されます。

【利点】とても少ないメモリの消費量で（およそ1MBの RAM を使用）、状態を リアルタイム に表示できます。

【欠点】単一のフォルダーしかキャッシュしないため、オーバーレイには再帰的な状態が表示されません。大きな作業コピーでは、デフォルトのキャッシュよりも、エクスプローラーのフォルダー表示に時間がかかることがあります。また、MIME タイプ列が使用できなくなります。

無し
この設定では、TortoiseSVN はエクスプローラーで状態をまったく取得しません。このため、バージョン管理下にあるファイルやフォルダーには、常に「通常」がオーバーレイ表示されます。それ以外のオーバーレイは表示されず、追加の列も無効になります。

【利点】追加でメモリを使用することは絶対になく、閲覧中にエクスプローラーが遅くなることもありません。

【欠点】ファイルやフォルダーの状態情報が、エクスプローラーに表示されません。作業コピーが変更されているかどうかを確認するには、「変更をチェック」ダイアログを使用する必要があります。

デフォルトでは、ファイルを開く・保存ダイアログでも、Windows エクスプローラーと同様に、オーバレイアイコンやコンテキストメニューが表示されます。 Windows エクスプローラーで のみ 表示させる場合は、**エクスプローラー上でのみオーバーレイとコンテキストメニューを表示**をチェックしてください。

バージョン管理外のファイルを含むフォルダーに、変更済マークをつけるようにもできます。これにより、まだバージョン管理下に入れてない新しいファイルを作成したのを、覚えておくのに便利です。このオプションは、キャッシュタイプ（前述）が デフォルト の場合にのみ有効です。

次のグループでは、オーバーレイを表示するストレージの種類を選択できます。デフォルトでは、ハードディスクドライブしか選択されていません。アイコンオーバレイをすべて無効にもできますが、そんなことしたいですか?

ネットワークドライブはとても遅いので、デフォルトでは、ネットワークフォルダー上にある作業コピーにはアイコンを表示しません。

USB フラッシュドライブは、ドライブタイプがデバイス毎に特定されるので特殊です。ある種のものは固定ドライブに見え、別のある種のものはリムーバブルドライブに見えます。

除外するパス は TortoiseSVN に対して、それらのパスにはアイコンオーバーレイやステータス列を 表示しない ことを伝えるために使用します。これは、変更しないライブラリのみが含まれるためオーバーレイの必要ない巨大な作業コピーが存在する場合、もしくは TortoiseSVN に特定のフォルダーのみ見て欲しい場合に有用です。

ここで指定するパスはどれも再帰的に適用されることを想定しています。そのため、どの子階層フォルダーもオーバーレイ表示されません。名前のついたフォルダー のみ を除外したいなら、? をパスの後ろに追加してください。

含めるパス にも同じことが言えます。オーバーレイが無効になるドライブタイプや、除外パスで指定したパスでも、オーバーレイされるようになります。

ユーザーは時々これら3つの設定がどのように作用するかを疑問に思うかもしれません。任意のパスに対して、一致するものが見つかるまでディレクトリ構造を上向きに探索し、含めるリストと除外するリストをチェックします。最初に一致する

ものが見つかった場合、含めるもしくは除外するルールに従います。もし競合した場合は、単一ディレクトリの設定は再帰的なものより優先されます。そして、含めるルールは除外するルールよりも優先されます。

分かりやすい例を示します。

```
除外するパス:
	C:
	C:¥develop¥?
	C:¥develop¥tsvn¥obj
	C:¥develop¥tsvn¥bin

含めるパス:
	C:¥develop
```

このように設定すると、 C: ドライブのオーバーレイ表示は無効になりますが、 c:¥develop 以下については、明確に除外されている c:¥develop フォルダー自身を除き、すべてのプロジェクトにオーバーレイ表示されます。頻繁に変更されるバイナリが入るフォルダーも除外されます。

TSVNCache.exe が検索するパスを制限することもできます。特定のフォルダー以下のみを検索したい場合は、全ドライブタイプを無効にし、検索対象にしたい特定のフォルダーのみを含めてください。

SUBST ドライブを除外する

作業コピーにアクセスするのに、以下のように SUBST ドライブを使用すると便利なことがあります。

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

しかし、 TSVNCache はファイルが変更されたという通知をひとつしか受け取れず、それは通常割り当て元のパスに対してであるため、オーバーレイ表示を更新できません。そのため、 subst パスのオーバーレイ表示はまったく更新されない可能性があります。

これに対処する簡単な方法は、表示対象から割り当て元のパスを除外することです。そうすれば、代わりに subst パスにオーバーレイ表示されます。

時には、 TSVNCache による変更の検索・監視を減らすために、作業コピーを含む領域を除外することがありますが、フォルダーに作業コピーが含まれていることを視覚表示したい場合があります。 **表示対象外にある作業フォルダーの最上位を通常として表示する** チェックボックスをチェックすると、除外されたパス(チェックされていないドライブタイプや、除外するパスとして指定されたもの)にある作業コピーの最上位フォルダーには、通常の最新の状態である緑色のチェックマークが表示されます。フォルダーのオーバーレイ表示は正しくはないかもしれませんが、作業コピーは見つけやすくなるでしょう。ファイルには全くオーバーレイ表示されません。オーバーレイ表示がなくても、コンテキストメニューは有効であることにご注意ください。

この特殊な例外として、A: ドライブと B: ドライブは'通常' として除外フォルダーを表示する オプションのようには動作しません。これは、Windows がエクスプローラーの起動時に、PC にフロッピードライブがあっても、数秒の遅延が考えられるドライブを強制的に参照するからです。

4.30.3.1. アイコンセットの選択

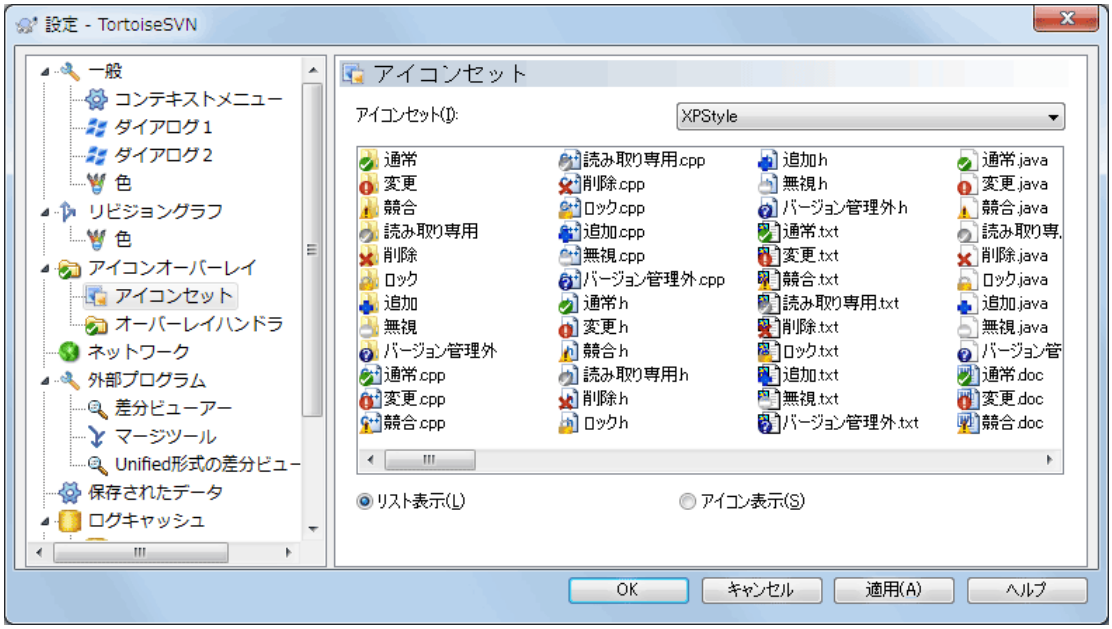


図4.69 設定ダイアログ、「アイコンセット」ページ

オーバーレイアイコンを任意のものに変更できます。オーバーレイアイコンセットを変更したときに、変更を有効にするにはコンピューターの再起動が必要なことに注意してください。

4.30.3.2. オーバーレイハンドラーを有効にする

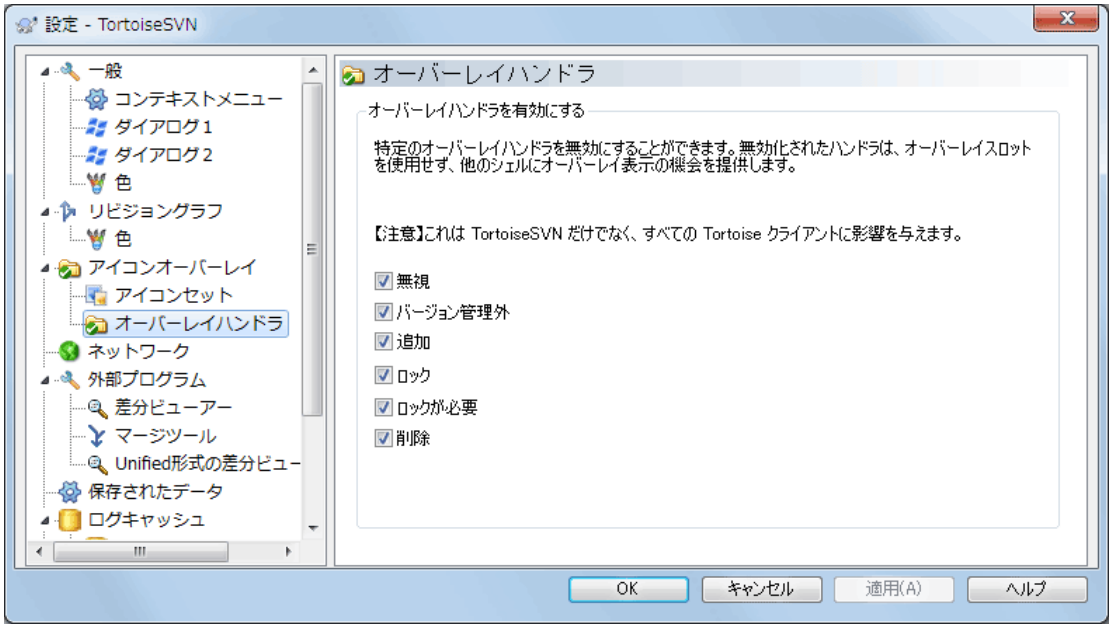


図4.70 設定ダイアログ、「オーバーレイハンドラー」ページ

使用可能なオーバーレイの数が厳しく制限されているため、任意のものがロードされることを保証するために、いくつかのハンドラを無効にすることができます。 TortoiseSVN は他の Tortoise クライアント(例えば TortoiseCVS や TortoiseHg)と共用される TortoiseOverlays コンポーネントを使用しているため、この設定はそれらのクライアントに影響します。

4.30.4. ネットワーク設定

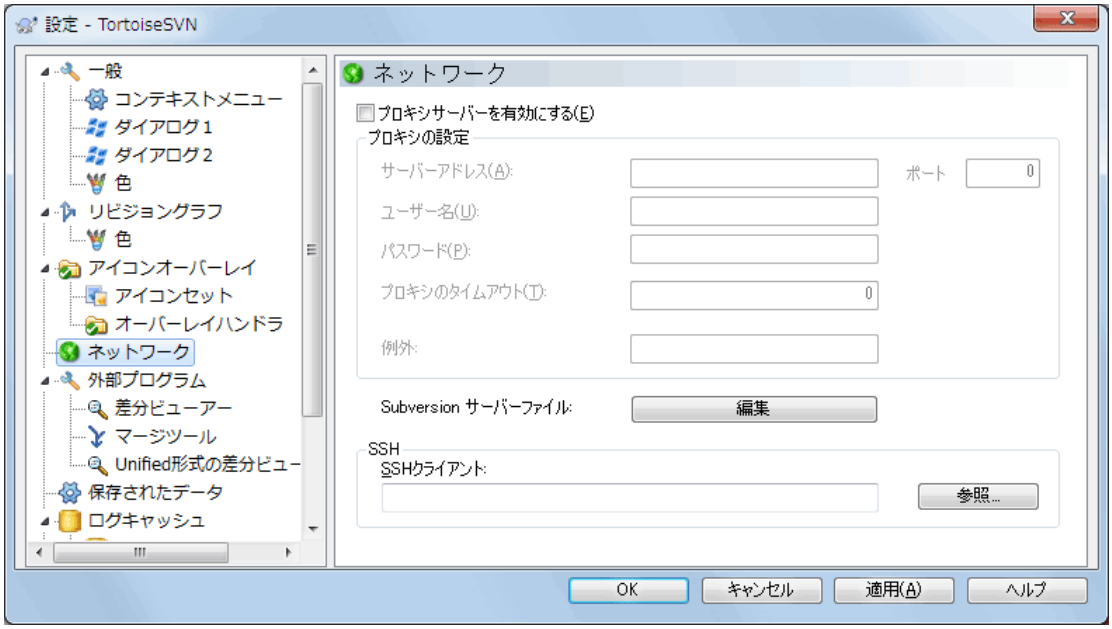


図4.71 設定ダイアログ、「ネットワーク」ページ

リポジトリごとにプロキシの設定をする必要がある場合、 Subversion の `servers` ファイルを直接編集する必要があります。直接編集する場合は **編集** ボタンを押してください。このファイルの使用法の詳細は、Runtime Configuration Area [http://svnbook.red-bean.com/en/1.7/svn.advanced.confarea.html] で説明しています。

TortoiseSVN が svn+ssh リポジトリと安全な接続を確立するのに使用するプログラムを指定できます。私たちは TortoisePlink.exe をお勧めします。これは TortoiseSVN に含まれている一般的な Plink プログラムです。ですがウィンドウなしアプリとしてコンパイルされているため、認証するたびに DOS ボックスがポップアップする、ということはありません。

実行ファイルのフルパスを指定しなければなりません。TortoisePlink.exe の場合、TortoiseSVN の標準 bin ディレクトリにあります。その場所を指定するには **参照** ボタンが役に立ちます。パスに空白が含まれる場合、以下のように引用符で囲んでください。

```
"C:¥Program Files¥TortoiseSVN¥bin¥TortoisePlink.exe"
```

ウィンドウを持たないということの側面として、いずれのエラーメッセージも行き場がないということがあります。そのため、認証が失敗したときに「標準出力に書き込めません」といった簡単なメッセージしか得られません。このため、まず標準の Plink をセットアップすることをお勧めします。すべて動作すれば、TortoisePlink を全く同じパラメーターで使用できます。

TortoisePlink は、Plink の派生物のためドキュメントがありません。コマンドラインパラメーターは PuTTY のウェブサイト [http://www.chiark.greenend.org.uk/~sgtatham/putty/] で参照してください。

パスワードを繰り返し入力しなくてもいいように、Pageant のようにパスワードキャッシュツールの使用も検討する必要があるでしょう。これも PuTTY のウェブサイトでダウンロードできます。

最終的に、サーバーとクライアントにSSHを設定するのは、このヘルプファイルの範疇を超えている重要なプロセスです。しかし、TortoiseSVN FAQ にリストされている Subversion/TortoiseSVN SSH How-To [http://tortoisesvn.net/ssh_howto] にガイドがあります。

4.30.5. 外部プログラムの設定

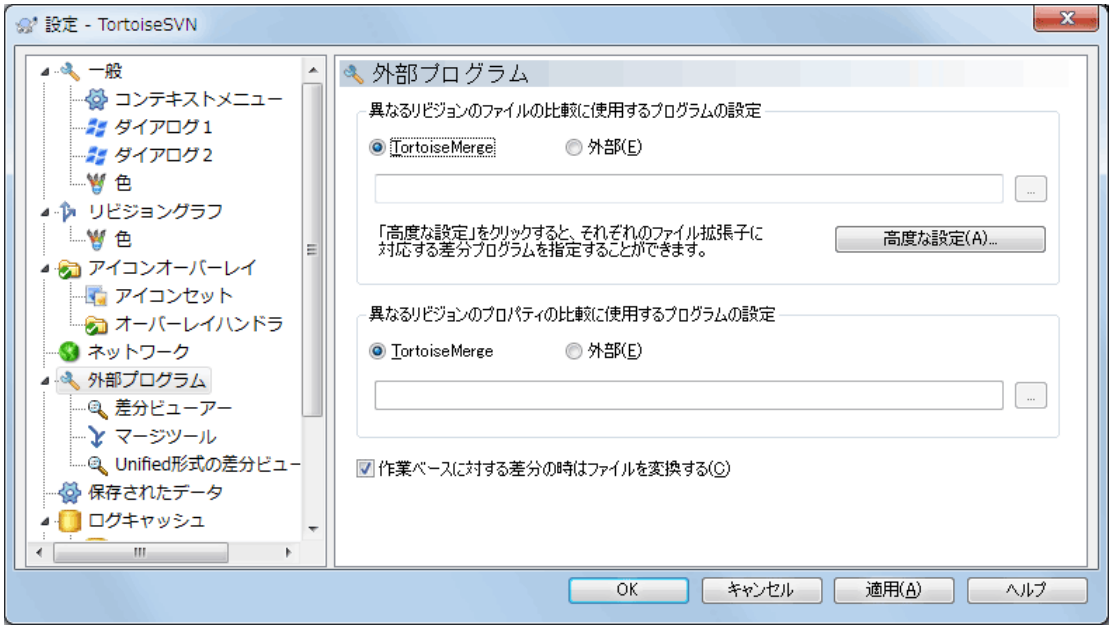


図4.72 設定ダイアログ、「差分ビューアー」ページ

ここでは、TortoiseSVN が使用する外部の差分・マージツールを定義できます。デフォルトでは TortoiseSVN と一緒にインストールされる TortoiseMerge が使われます。

TortoiseSVN の外部の差分・マージツールとしてよく使われているものの一覧は、「外部差分・マージツール」をご覧ください。

4.30.5.1. 差分ビューアー

外部の差分ツールは、ファイルを異なるリビジョン間で比較するのに使用します。外部ツールは、それぞれのコマンドラインオプションに従って、コマンドラインからファイル名を取得する必要があります。TortoiseSVN は % が先頭についた置換パラメーターを使用します。このパラメーターが現れると、適切な値に置換されます。パラメーターの順番はお使いの差分ツールに合わせてください。

%base
　変更前のファイル

%bname
　変更前ファイルのウィンドウタイトル

%mine
　変更後のファイル

%yname
　変更後ファイルのウィンドウタイトル

%burl
　元ファイルのURL(もしあれば)

%yurl
　2番目のファイルのURL(もしあれば)

%brev
　元ファイルのリビジョン（もしあれば）

%yrev
　2番目のファイルのリビジョン（もしあれば）

%peg
　ペグリビジョン（もしあれば）

ウィンドウタイトルは単なるファイル名ではありません。TortoiseSVN は、それを表示名として扱うため、それにふさわしい名前を作成します。たとえば、あるファイルのリビジョン 123 と作業コピーの差分を表示する場合、それぞれの名前は filename : リビジョン 123 と filename : 作業コピー となります。

ExamDiff Pro の設定例:

```
C:¥Path-To¥ExamDiff.exe %base %mine --left_display_name:%bname
    --right_display_name:%yname
```

KDiff3 の設定例:

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

WinMerge の設定例:

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

Araxis の設定例:

```
C:¥Path-To¥compare.exe /max /wait /title1:%bname /title2:%yname
    %base %mine
```

UltraCompare の設定例:

```
C:¥Path-To¥uc.exe %base %mine -title1 %bname -title2 %yname
```

DiffMerge の設定例:

```
C:¥Path-To¥DiffMerge.exe -nosplash -t1=%bname -t2=%yname %base %mine
```

キーワード、特にファイルの リビジョン 展開に svn:keywords を使用する場合、キーワードの現在値によって、厳密には違いがでるかもしれません。また、svn:eol-style = native を使用しているなら、BASE ファイルの行末は純粋に LFになっているでしょうが、ローカルのファイルの行末は CR-LF かもしれません。通常 TortoiseSVN は 差分操作の前に、BASE ファイルに対しキーワード展開や行末を分析して自動的にこの違いを隠蔽します。しかしこの操作は時間がかかる可能性があります。BASE に対する差分の時はファイルを変換する にチェックがなければ、TSVN はファイルに対する事前処理をスキップします。

Subversion のプロパティ用に別の差分ツールを指定することもできます。プロパティは短く簡単な文字列である傾向があるので、シンプルでもっとコンパクトなビューアーを使いたくなるかもしれません。

代替差分ツールを設定すると、コンテキストメニューから TortoiseMerge と サードパーティツールにアクセスできます。コンテキストメニュー → 差分を表示 で優先差分ツールを使用し、Shift+ コンテキストメニュー → 差分を表示 で代替差分ツールを使用します。

4.30.5.2. マージツール

外部マージプログラムを競合したファイルの解決に使用します。パラメーターを差分プログラムと同様の方法で置換します。

%base
　誰も変更していないもとのファイル

%bname
　変更前ファイルのウィンドウタイトル

%mine
　自分の変更がある自分のファイル

%yname
　変更後ファイルのウィンドウタイトル

%theirs
　リポジトリにあるファイル

%tname
　リポジトリにあるファイルのウィンドウタイトル

%merged
　マージ操作をした結果の競合ファイル

%mname
　マージしたファイルのウィンドウタイトル

Perforce Merge の設定例:

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

KDiff3 の設定例:

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

Araxis の設定例:

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

WinMerge (2.8 以降) の設定例:

```
C:¥Path-To¥WinMerge.exe %merged
```

DiffMerge の設定例:

```
C:¥Path-To¥DiffMerge.exe -caption=%mname -result=%merged -merge
    -nosplash -t1=%yname -t2=%bname -t3=%tname %mine %base %theirs
```

4.30.5.3. 差分・マージの高度な設定

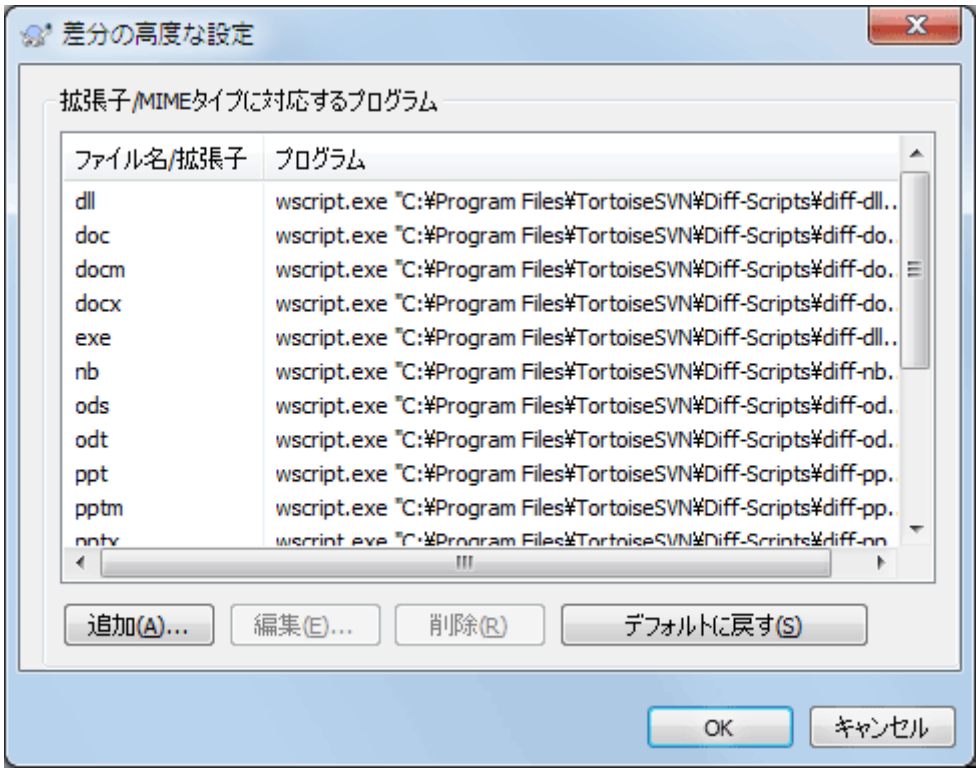


図4.73 設定ダイアログ、「差分・マージの高度な設定」ダイアログ

高度な設定では、ファイルの拡張子ごとに異なる差分・マージプログラムを定義できます。たとえば、`.jpg` ファイルの「差分」を取るのに Photoshop を関連付けできます。:-) `svn:mime-type` で差分・マージプログラムと関連付けることもできます。

拡張子を関連付けるには、拡張子を指定する必要があります。Windows ビットマップファイルを指定するには、`.BMP`としてください。`svn:mime-type` プロパティを使用して関連付けるには、`text/xml` のようにスラッシュを含めて mime type を指定してください。

4.30.5.4. Unified 形式の差分ビューアー

Unified-Diff ファイル（パッチファイル）のビューアーです。パラメーターは必要ありません。**デフォルト**の設定では、TortoiseSVN と一緒にインストールされる TortoiseUDiff を使用するよう設定され、追加・削除された行を色分け表示します。

Unified差分ファイルは単なるテキストファイルなので、任意のテキストエディターを使用しても結構です。

4.30.6. 保存データの設定

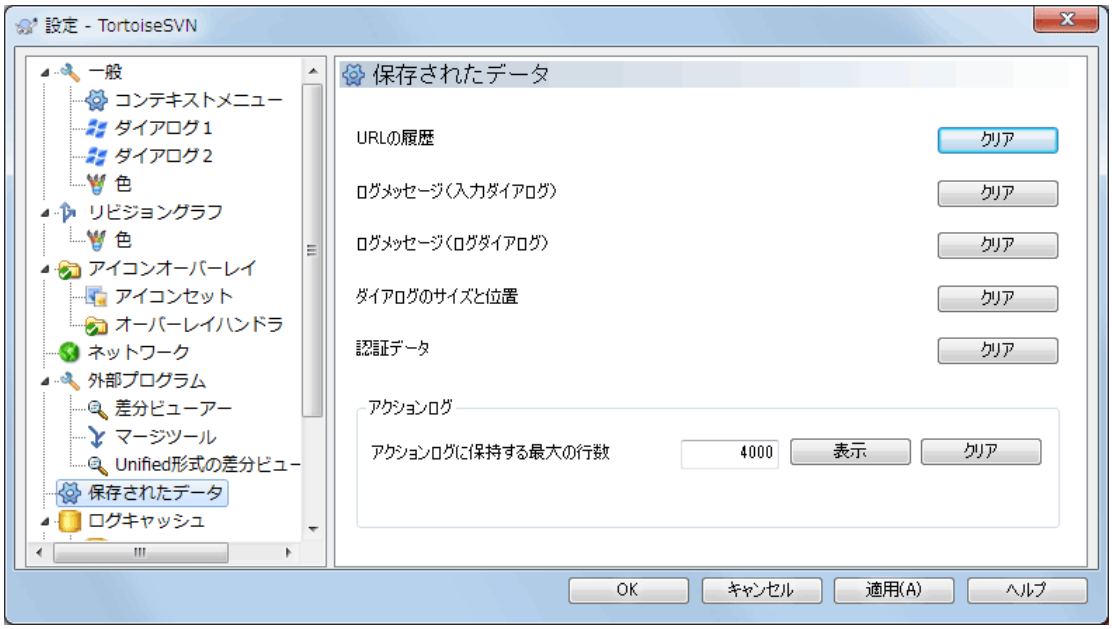


図4.74 設定ダイアログ、「保存されたデータ」ページ

便宜上、TortoiseSVN はたくさんの設定を保存し、最近の情報を記憶しています。データのキャッシュを削除する場合、ここで行います。

URLの履歴
: 作業コピーのチェックアウトや、変更点のマージ、リポジトリブラウザーの使用の際は常に、TortoiseSVN は直近に使用した URL を記録し、コンボボックスに提供します。時には URL がもう存在しなくなり、リストが散らかりますので、定期的に掃除するのに便利です。

  コンボボックスに表示されている 1 項目だけを、その場で削除したい場合、下矢印をクリックしてコンボボックスを開き、削除したい項目にマウスをあわせて、Shift+Del を押してください。

ログメッセージ(入力ダイアログ)
: TortoiseSVN は最近の入力したコミットログを保存しています。リポジトリごとに保存していますので、たくさんのリポジトリにアクセスする場合、極度に肥大化してしまうことがあります。

ログメッセージ(ログダイアログ)
: TortoiseSVN は、次回ログを参照した際の時間節約のため、ログ参照ダイアログが取得したログメッセージをキャッシュしています。メッセージがキャッシュされている状態で、他の誰かがログメッセージを編集した場合、キャッシュを消去しないとその変更を参照できません。ログメッセージのキャッシュは **ログキャッシュ** タブで有効にできます。

ダイアログのサイズと位置
: 多くのダイアログは、前回使用したときのサイズと画面内の位置を記憶しています。

認証データ
: Subversion サーバーの認証を通れば、ユーザー名とパスワードを毎回入力しなくてもいいように、ローカルにキャッシュしています。セキュリティ上の理由や、リポジトリに別のユーザー名でアクセスするのに消去したいかもしれません……ジョンはPCがあなたに使われているのを知っていますか?

  特定のサーバーのみ、認証データをクリアしたければ、キャッシュデータの探し方について、「認証」をご覧ください。

アクションログ

TortoiseSVN は、進行ダイアログに書き込まれたすべてのログを保持しています。これは例えば、最近の更新コマンドで何が起きたのかをチェックしたいときに便利です。

ログファイルは長さを制限しており、大きくなりすぎたときには古い内容から破棄していきます。デフォルトでは 4000 行保持しますが、この数字はカスタマイズできます。

ここからログファイルの内容の確認や消去ができます。

4.30.7. ログキャッシュ

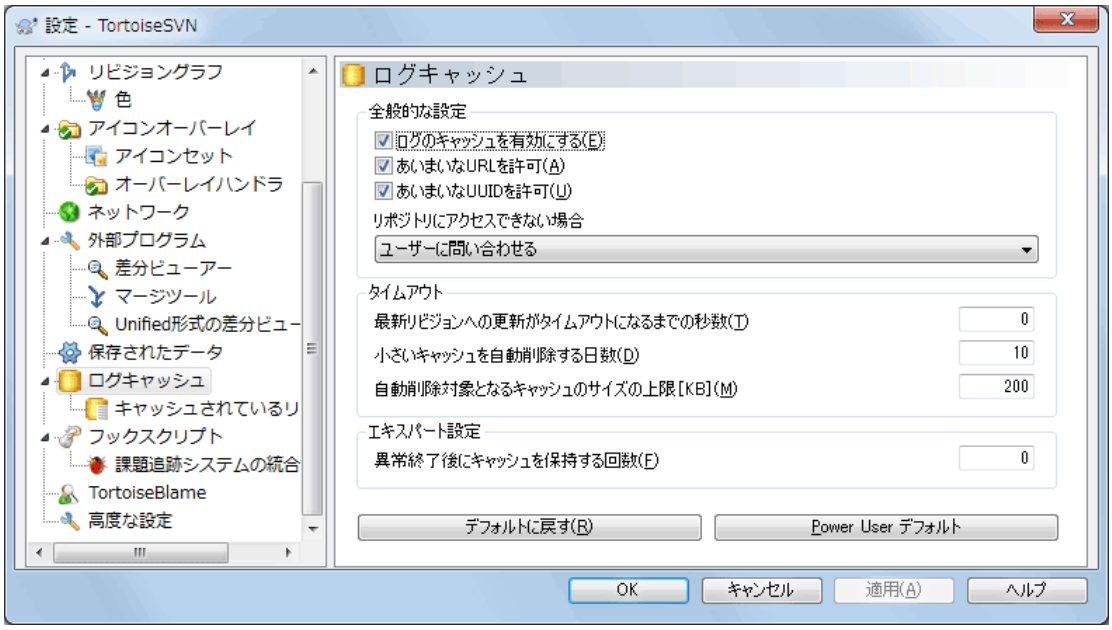


図4.75 設定ダイアログ、「ログキャッシュ」ページ

このダイアログでは、TortoiseSVNのログキャッシュ機能の設定ができます。これはログメッセージと変更したパスのローカルコピーを保持し、サーバーからダウンロードの時間消費を防ぎます。ログキャッシュを使用すると、ログダイアログやリビジョングラフが劇的に速くなります。その他にも、オフライン時にもログメッセージを参照できるといった便利な特徴もあります。

ログのキャッシュを有効にする

ログデータが必要な時は常にログのキャッシュを有効にします。チェックすると、キャッシュが有効ならキャッシュからデータを取得します。キャッシュにメッセージがなければ、サーバーから取得しキャッシュに追加します。

キャッシュを無効にすると、データを常にサーバーから直接取得し、ローカルに保存しません。

あいまいなURLを許可

時には、すべてのリポジトリが同じ URL を使用するサーバーに、接続しなければならないかもしれません。svnbridge の旧バージョンではこうなるでしょう。そのようなリポジトリにアクセスする必要がある場合、このオプションをチェックする必要があります。そうでなければ、性能向上のため、チェックしないままにしてください。

あいまいなUUIDを許可

いくつかのホスティングサービスでは、すべてのリポジトリが同じ UUID を与えます。また、自分のリポジトリフォルダーをコピーして新しいリポジトリを作成、のようなことさえしたかもしれません。さまざまな理由でこれはまずい考えです。UUID は一意でなければなりません。しかし、このチェックボックスをチェックすると、そのような状況でもログキャッシュが動作します。必要なければ、性能向上のため、チェックしないままにしてください。

リポジトリにアクセスできない場合
: オフラインで作業していたり、リポジトリサーバーがダウンしていたりした場合でも、ログキャッシュを使用して、すでにキャッシュに保持したログメッセージを提供できます。もちろんキャッシュは最新の状態ではありませんから、この機能を使用するかどうかの選択肢があります。

  サーバーに接続せず、キャッシュからログデータを取得する際、ダイアログはオフライン状態であることをタイトルバーに表示します。

HEAD リビジョンへの更新がタイムアウトになるまでの秒数
: ログダイアログを表示する際、通常、新しいログメッセージをチェックするのにサーバーに接続すると思います。ここでタイムアウトに0以外を指定したばあい、前回の接続からその秒数経過しているときだけ、サーバーに接続します。頻繁にログダイアログを開き、サーバーが遅い場合、これによりサーバーとのやりとりを抑えることができますが、表示されるデータが完全に最新であるとは限りません。この機能を使用する場合は、折衷案として300（5分）とするのをお勧めします。

小さいキャッシュを自動削除する日数
: たくさんのリポジトリをブラウズすると、たくさんのログキャッシュを蓄積することになります。そのリポジトリを頻繁に使用しなければ、キャッシュが大きくふくらむことはありません。そのため、TortoiseSVN はデフォルトで、ここにセットした時間後にキャッシュを削除します。キャッシュの削除を制御するのに、この項目を使用してください。

自動削除対象となるキャッシュのサイズの上限
: 大きなキャッシュの再取得は、より負荷が高くなります。そのため、TortoiseSVN は小さなキャッシュしか削除しません。この値で、適切な閾値を調節してください。

異常終了後にキャッシュを保持する回数
: 時折、キャッシュに何か障害が発生し、それが元で異常終了することがあります。これが発生すると、問題の再発防止のため、通常自動的にキャッシュを削除します。ナイトリービルドのような不安定さの残るバージョンを使用する場合などで、キャッシュを保持するために指定することができます。

4.30.7.1. キャッシュされているリポジトリ

このページでは、ローカルにキャッシュしたリポジトリの一覧を、キャッシュに使用した領域とともに参照できます。リポジトリを選択すると、その下のボタンを使用できます。

更新　をクリックすると、キャッシュを完全に再読込し、抜けたところを補完します。大きなリポジトリでは、非常に時間がかかりますが、オフラインで作業し、もっともよい状態のキャッシュが必要な場合に便利です。

エクスポート ボタンをクリックすると、キャッシュ全体を CSV ファイルでエクスポートします。ログデータを外部プログラムで処理をする際に便利ですが、主に開発者用でしょう。

削除　をクリックすると、選択したリポジトリのキャッシュデータをすべて削除します。これはそのリポジトリのキャッシュを無効にするものではなく、次回ログデータが必要になると、新しいキャッシュを作成します。

4.30.7.2. ログキャッシュの統計データ

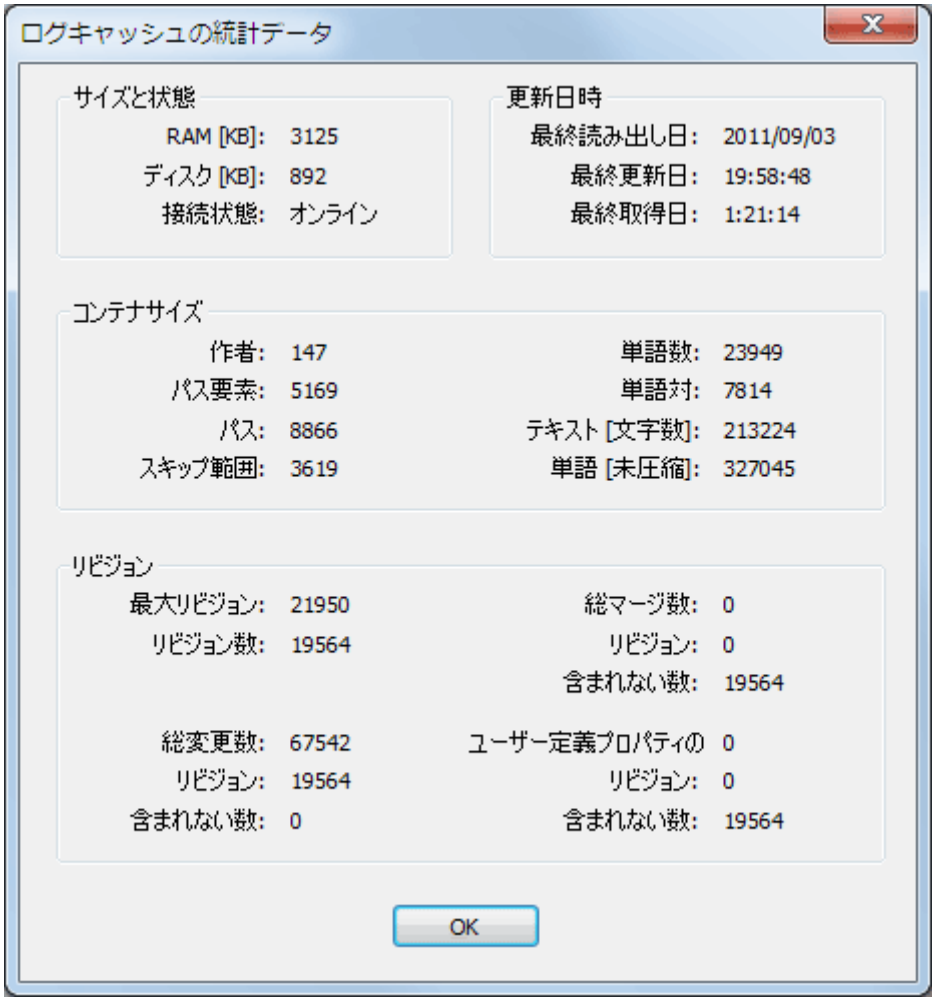


図4.76 設定ダイアログ、ログキャッシュ統計

詳細 ボタンをクリックすると、特定のキャッシュの詳細統計を表示します。ここではたくさんの項目が表示され、これは主に TortoiseSVN の開発者が興味を引く項目です。そのため、すべてを詳細に説明することはしません。

RAM
: このキャッシュを提供するのに必要なメモリ量です。

ディスク
: キャッシュに使用するディスクスペースです。データは圧縮されますので、ディスクの使用量は、一般的にかなり控えめになります。

接続状態
: リポジトリが有効なキャッシュを、前回使用したかどうかを表示します。

最終更新日
: 最後にキャッシュの内容を更新した日時です。

最終取得日
: サーバーから HEAD リビジョンを最後に取得した日時です。

作者
: キャッシュに記録したメッセージの作者数です。

パス
: `svn log -v` で見られるリストにある、パスの数です。

スキップ範囲
: 取得していないリビジョン範囲の数で、単純にリクエストしていないものです。キャッシュ内のホール数の尺度になります。

最大リビジョン
: キャッシュに格納されている、リビジョン番号の最大値です。

リビジョン数
: キャッシュに格納されているリビジョン番号の数です。キャッシュの完全性に関する別の尺度になります。

4.30.8. クライアント側フックスクリプト

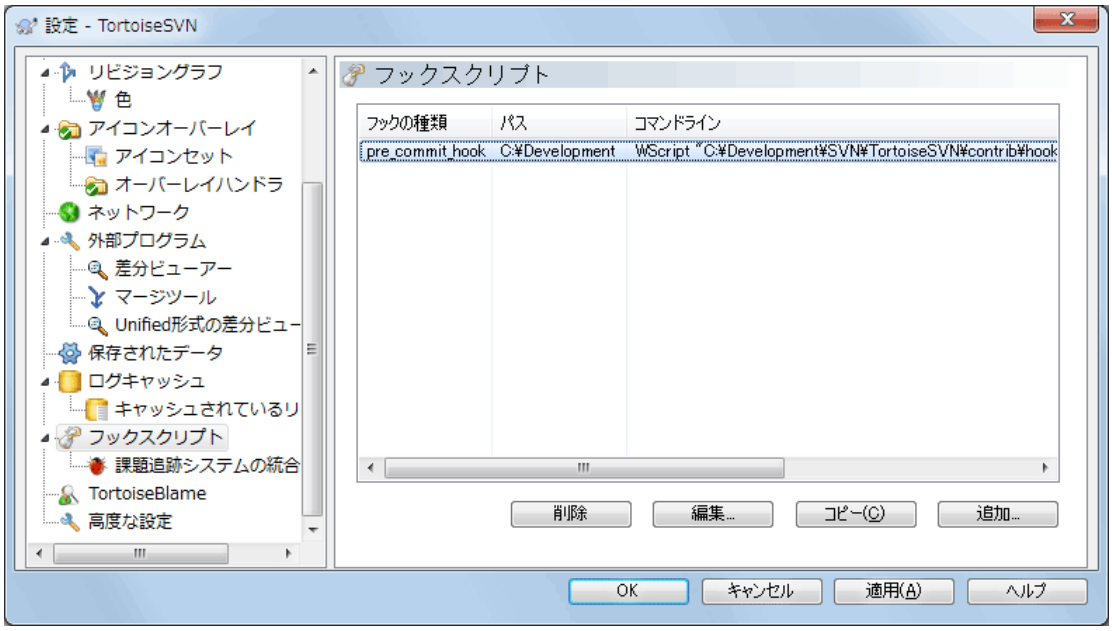


図4.77 設定ダイアログ、「フックスクリプト」ページ

このダイアログでは、Subversion のアクションに合わせて自動的に実行される、フックスクリプトのセットアップを行えます。対照的に「サーバー側フックスクリプト」で説明しているフックスクリプトは、クライアントのローカルで実行されます。

ここで言うフックは、コミット後にバージョン番号を更新するため SubWCRev.exe のようなプログラムを起動したり、再構築のトリガになったりします。

様々なセキュリティ上、実装上の理由により、フックスクリプトはプロジェクトプロパティとしてではなく、マシンローカルに定義されます。他の誰かがリポジトリにコミットするのを考慮する必要なく、何を行うかを定義してください。もちろん、スクリプトをバージョン管理下に置くかどうかを、いつでも選べます。

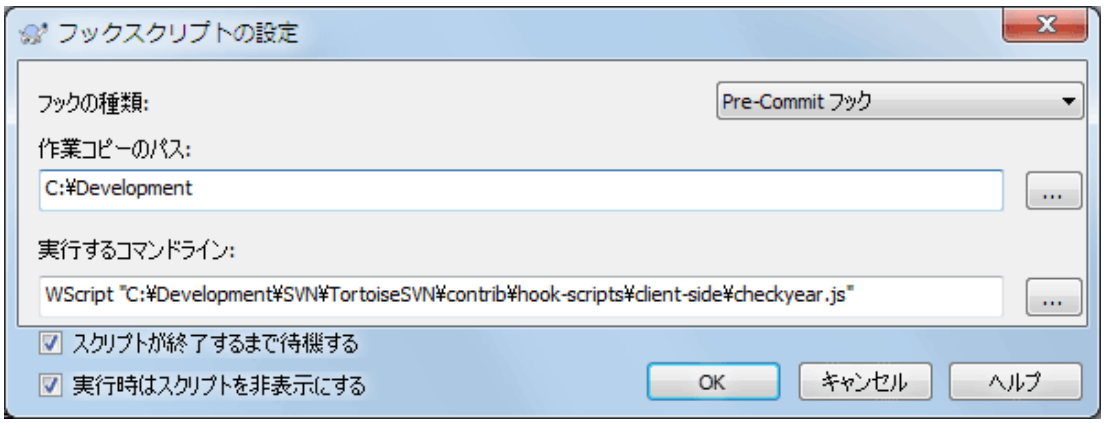


図4.78 設定ダイアログ、フックスクリプトの設定

新しいフックスクリプトを追加するには、単に 追加 をクリックし、詳細を入力してください。

現在、6種類のフックスクリプトが使用できます。

Start-commit
: コミットダイアログを表示する前に呼ばれます。フックがバージョン管理下のファイルを変更したり、コミットするファイルのリストやコミットメッセージに影響を与える場合に使用してください。しかし、初期の段階でフックが呼ばれるため、コミット用に選択したオブジェクトの全リストは有効でないことに注意するべきです。

Pre-commit
: コミットダイアログの OK をクリックした後、実際のコミットが始まる前に呼ばれます。このフックは実際にコミットするファイルのリストを持ちます。

Post-commit
: コミットが完了した後で(成功・失敗に関わらず)呼ばれます。

Start-update
: リビジョンへの更新ダイアログが表示される前に呼ばれます。

Pre-update
: 実際のSubversionの更新や切り替え操作が始まる前に呼び出されます。

Post-update
: 更新、切り替え、チェックアウト、が終了した後に(成功したかどうかにかかわらず)呼び出されます。

Pre-connect
: リポジトリに接続を試みる前に呼び出されます。通常、およそ5分に1度の割合で呼び出されます。

フックは特定の作業コピーのパスに対して定義されます。最上位のパスのみ指定する必要があります。もし、サブフォルダーを操作する場合、TortoiseSVN は自動的に上位へマッチするかどうか検索を行います。

次に、実行するコマンドラインを、フックスクリプトや実行ファイルのパスから始めて指定する必要があります。ここではバッチファイルや実行可能ファイル、その他の Windows と関連付けられたファイル(例えば Perl スクリプト)を指定します。なお、Windows のセキュリティ上の制限で、Windows は UNC パスで指定されたスクリプトを実行しないため、この形式で指定することはできません。

コマンドラインには、TortoiseSVN が設定するパラメーターを含めます。呼ばれるフックによってパラメーターは異なります。各フックには以下の順番で渡されるパラメーターがあります。

Start-commit
: `PATHMESSAGEFILECWD`

Pre-commit
: PATHDEPTHMESSAGEFILECWD

Post-commit
: PATHDEPTHMESSAGEFILEREVISIONERRORCWD

Start-update
: PATHCWD

Pre-update
: PATHDEPTHREVISIONCWD

Post-update
: PATHDEPTHREVISIONERRORCWD

Pre-connect
: no parameters are passed to this script. You can pass a custom parameter by appending it to the script path.

各パラメーターの意味は以下に説明する通りです。

PATH
: 操作を開始したときのすべてのパスを格納した一時ファイルのパスです。各パスは、一時ファイル内に 1 行ごと格納されています。

DEPTH
: コミット・更新が完了した際の深さです。

  以下のような有効な値があります。

  -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
: コミット用のログメッセージを格納しているファイルのパスです。このファイルには、UTF-8 エンコードのテキストが格納されています。start-commit フックの実行が完了すると、ログメッセージを読み返し、フックが変更する機会を与えます。

REVISION
: 更新したりコミットが完了したときの、リポジトリのリビジョンです。

ERROR
: エラーメッセージを含むファイルのパスです。エラーがない場合、このファイルは空になります。

CWD
: スクリプトを実行する現在の作業ディレクトリです。これはすべてに影響を与えるパスの、共通のルートディレクトリに設定されます。

便宜上パラメーターに名前を付けましたが、フックの設定で、その名前を参照する必要はないことに注意してください。個々のフックに挙げたパラメーターはすべて、好むと好まざるとに関わらず常に渡されます ;-)

Subversion の操作を、フックスクリプトが完了するまで止めておきたい場合は、**スクリプトが終了するまで待機** をチェックしてください。

通常、スクリプト実行時に不格好な DOS ウィンドウを隠しておきたいと思います。そのため **実行時はスクリプトを非表示にする** をデフォルトでチェックしてあります。

クライアントフックスクリプトのサンプルは、 TortoiseSVN リポジトリ [http://tortoisesvn.googlecode.com/svn/trunk/contrib/hook-scripts] サイトの `contrib` フォルダーから入手できます(リポジトリへのアクセス方法は、「ライセンス」をご覧ください。)

TortoiseSVN のインストールディレクトリに、 `ConnectVPN.exe` という小さいツールが含まれています。このツールを pre-connect フックとして使用すると、TortoiseSVN がリポジトリにアクセスしようとする前に、自動的に VPN に接続できるようになります。ツールの最初の引数として、VPN 接続の名前を渡すようにしてください。

4.30.8.1. 課題追跡システムとの統合

TortoiseSVN は、コミットダイアログで課題管理システムにクエリを発行するために、COM プラグインを使用できます。そのようなプラグインの使用を、「課題追跡システムからの情報取得」で説明しています。システム管理者が、すでにインストールして登録したプラグインを提供している場合、どのように作業コピーと統合するかをここで指定します。

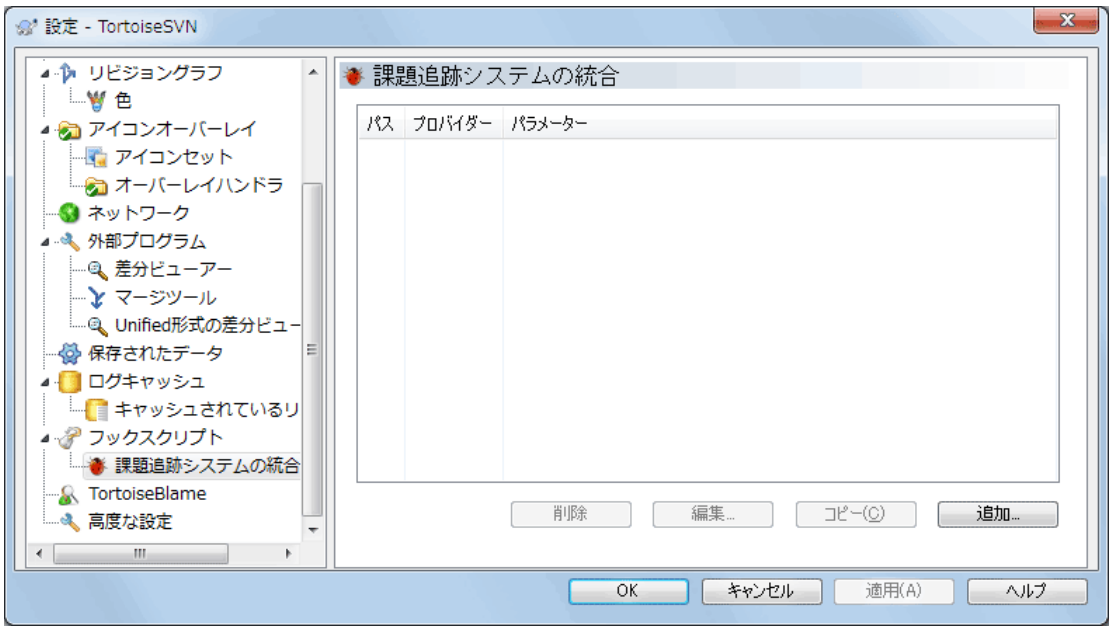


図4.79 設定ダイアログ、「課題追跡システムとの統合」ページ

特定の作業コピーでプラグインを使用するには、**追加...** をクリックしてください。ここで作業コピーのパス、登録済み課題管理システムプラグインのドロップダウンリストでの選択、渡す任意のパラメーターの指定ができます。パラメーターはプ

ラグインに固有ですが、自分に割り当てられた課題をプラグインが抽出できるように、課題管理システムのユーザー名を含めるかも知れません。

プロジェクトのすべてのユーザーで同じ COM プラグインを使用する場合は、bugtraq:provideruuid、bugtraq:provideruuid64、bugtraq:providerparamsをプロパティとして設定することができます。

bugtraq:provideruuid
: IBugtraqProvider の COM UUID（例えば、{91974081-2DC7-4FB1-B3BE-0DE1C8D6CE4E}）を指定します。（この例は Google Code [http://code.google.com/hosting/] の課題管理システム向けプロバイダーである Gurtle bugtraq provider [http://code.google.com/p/gurtle/] の UUID です）。

bugtraq:provideruuid64
: これは bugtraq:provideruuid と同様ですが、64ビット版の IBugtraqProvider です。

bugtraq:providerparams
: このプロパティは、IBugtraqProvider に渡すパラメーターを指定します。

2つのプロパティに何を設定するか調べるには、IBugtraqProvider プラグインのドキュメントを参照してください。

4.30.9. TortoiseBlame の設定

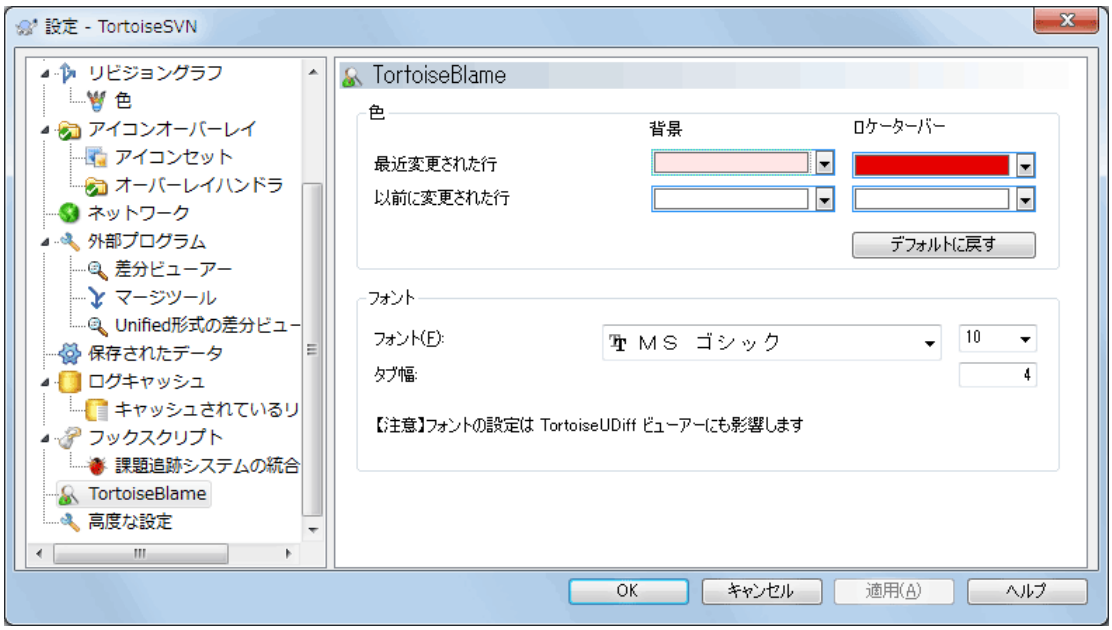


図4.80 設定ダイアログ、「TortoiseBlame」ページ

TortoiseBlame で使用する設定は、TortoiseBlame 自体ではなく、メインのコンテキストメニューから設定します。

色
: TortoiseBlame では、ファイルにある行の古さを背景色で表現します。ここでは、最新と最古の両端のリビジョンの色を指定します。TortoiseBlame はこの 2 色の中間色を使用して、行単位のリポジトリのリビジョンを表現します。

  表示位置調整バーで使用する差分の色を指定することができます。デフォルトでは、メインウィンドウで文字を読みやすいよう背景の明るさを維持しながら、表示位置調整バーと強いコントラストをなす色が使われています。

フォント
: ここでは、テキストを表示するフォントとポイントサイズを指定できます。ここで指定されたフォントは、ファイルの内容と、左画面に表示する作者とリビジョンの両方に使用されます。

タブ
ファイルの内容にタブ文字があった場合、空白を何文字使用して展開するかを定義します。

4.30.10. 高度な設定

使用頻度の低い一部の設定は、設定ダイアログの高度な設定ページでのみ設定できます。これらの設定はレジストリを直接操作するので、それぞれの設定が何をするのかを理解しておく必要があります。これらの変更がどうしても必要な場合を除いて、これらの設定を変更しないでください。

AllowAuthSave
同じコンピューター上で、複数のユーザーが同じアカウントを使用していることがあります。このような場合、認証データを保存すべきでないことがあります。この値を false にせ底すると、認証ダイアログの**認証を保存**ボタンを無効にすることができます。

AllowUnversionedObstruction
アップデートによって、すでにローカルの作業コピーに同じ名前が存在する新しいファイルを取得したとき、デフォルトの動作はローカルファイルを保持し、リポジトリからの新しいファイルは（可能ならば）変更後のバージョンとして表示します。false に設定すると、TortoiseSVN がこれを競合として扱うようになります。

AlwaysExtendedMenu
エクスプローラーでは Shift キーを押しながらコンテキストメニューを開くと、TortoiseSVN の追加コマンドが表示されます。常に追加コマンドを表示させたい場合は、true に設定してください。

AutocompleteRemovesExtensions
コミットメッセージのエディターでは、コミットするファイルが自動保管リストで表示されます。true に設定すると、拡張子を除外した名前を表示するようになります。

BlockStatus
TortoiseSVN　の他のコマンド（例えば更新、コミット、など）が動作している間、エクスプローラーのステータスオーバーレイを更新したくない場合は、この値を true に設定してください。

CacheTrayIcon
TSVNCache にトレイアイコンのキャッシュを追加するには、この値を　true に設定してください。これはプログラムを正規に終了させるために必要なもので、開発者にのみ有用な設定です。

ColumnsEveryWhere
作業コピーを Windows エクスプローラーの詳細表示で表示させたとき、TortoiseSVN が列を追加します。作業コピーばかりでなくどこでも列を追加したい場合、この値を true に設定してください。なお、列の追加は Windows XP でしか行われず、Vista 以降ではこの機能はサポートされていません。

ConfigDir
Subversion の設定ファイルを別な場所に設定することができます。これはすべての TortoiseSVN の操作に影響します。

CtrlEnter
TortoiseSVN の多くのダイアログでは、Ctrl+Enter を押すことで OK ボタンを押したときと同様にダイアログを閉じることができます。この値をfalse に設定すると、この機能は無効になります。

Debug
trueに設定すると、TortoiseProc.exe を使用して開始されるたびにあらゆるコマンドについて、ポップアップダイアログでコマンドラインの内容を表示できます。

DebugOutputString
: TortoiseSVN が実行中にデバッグメッセージを表示させたい場合は、これをtrueに設定してください。メッセージは、特殊なデバッグツールのみでキャプチャすることができます。

DialogTitles
: ダイアログタイトルのデフォルトの書式(0に設定時)は、URL/パス - ダイアログ名 - TortoiseSVN です。この値を1に設定すると、書式が ダイアログ名 - URL/パス - TortoiseSVN になります。

DiffBlamesWithTortoiseMerge
: TortoiseSVN は外部の差分ビューアーを使用することができます。しかし、多くのビューアーには注釈履歴 (「注釈履歴の差分」)が統合されていませんので、その場合はTortoiseMerge に戻したくなることもあるでしょう。そのためには、この値をtrueに設定してください。

FixCaseRenames
: アプリケーションによっては、必要でもないのに勝手に、通知なしでファイル名の大文字/小文字を変えてしまうものがあります。例えば、 file.txt が FILE.TXT に変わっても、 Windows アプリケーションでは困りませんが、Subversion ではこのような場合にも大文字/小文字を区別してしまいます。そのため、TortoiseSVN ではこのような大文字/小文字の変更を自動的に修正します。

  もし、TortoiseSVN が自動的に大文字/小文字の修正を行ってほしくない場合、この値を false に設定してください。

FullRowSelect
: 様々なダイアログ(例えばコミット、変更をチェック、追加、変更の取り消し、など)のリストコントロールを、全行選択(つまり、項目を選択する際に、最初のカラムではなく行全体を選択すること)にします。これは使いやすいのですが、行全体が反転するため、右下の背景画像に重なると醜くなってしまうことがあります。全行選択を解除する場合は、この値を false に設定してください。

GroupTaskbarIconsPerRepo
: このオプションは、様々な TortoiseSVN のダイアログやウィンドウが、Windows 7 のタスクバー内でグループ化される方法を決定します。このオプションは、Windows XP や Vista では効果がありません。

1. デフォルト値は0です。この設定では、アイコンがアプリケーションの種類によってグループ化されます。TortoiseSVNのすべてのダイアログがひとつにグループ化され、TortoiseMergeのすべてのウィンドウがひとつにグループ化される・・・という形です。

図4.81 タスクバーでのデフォルトのグループ化

2. 1に設定すると、すべてのダイアログがアプリケーション単位ではなく、リポジトリ単位でグループ化されます。たとえば、リポジトリAのログダイアログとコミットダイアログが開いていて、リポジトリBの変更のチェックダイアログとログダイアログが開いていた場合、Windows 7のタスクバーには2つのアプリケーションアイコングループが表示され、それぞれリポジトリごとにグループ化されます。但し、TortoiseMergeのウィンドウは、TortoiseSVNのダイアログと同じグループにはなりません。

図4.82 タスクバーでのリポジトリ毎のグループ化

3. 2に設定すると、1に設定した場合と同様にグループ化されますが、 TortoiseSVN、 TortoiseMerge、TortoiseBlame、TortoiseIDiff、TortoiseUDiff のウィンドウも一緒にグループ化されます。たとえば、コミットダイアログが開いているときに変更されたファイルをダブルクリックした場合、TortoiseMerge の差分ウィンドウが開きますが、タスクバー上ではコミットダイアログと同じアイコングループに含められます。

図4.83 タスクバーでのリポジトリ毎のグループ化

4. 3に設定すると、1に設定した場合と同様にグループ化されますが、リポジトリごとではなく作業コピーごとにグループ化されます。これは、同じリポジトリですべてのプロジェクトを管理し、プロジェクトごとに異なる作業コピーを使用している場合に便利です。

5. 4に設定すると、2に設定した場合と同様にグループ化されますが、リポジトリごとではなく作業コピーごとにグループ化されます。

GroupTaskbarIconsPerRepoOverlay

`GroupTaskbarIconsPerRepo` オプションを0に設定した場合は、効果がありません（上記参照）。

このオプションを `true` に設定すると、Windows 7 のタスクバーにはダイアログやウィンドウで使用されているリポジトリを識別するための、色のついた四角の小さいオーバーレイが表示されます。

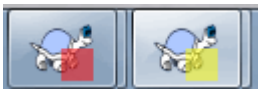


図4.84 タスクバーでのグループ化にリポジトリのカラーオーバーレイが付いた様子

IncludeExternals

デフォルトでは、TortoiseSVN は常に外部参照を含めて更新を行います。これは作業コピーの一貫性の問題を回避することができます。しかし、外部参照のセットが多い場合、更新に時間がかかるようになります。この値を `false` に設定すると、デフォルトでは外部参照を含めずに更新されるようになります。外部参照を含めて更新するには、**特定リビジョンへ更新...**を実行するか、この値を再び `true` に設定します。

LogFindCopyFrom

ログダイアログがマージウィザードから起動されると、すでにマージされたリビジョンは灰色で表示されますが、ブランチが作成された後のリビジョンも表示されます。それらをマージすることができないため、これらのリビジョンは黒で示されています。

このオプションを `true` に設定すると、TortoiseSVN はブランチが作成されたリビジョンを、そのリビジョン以降の隠されたリビジョンからも検索しようとします。これには時間がかかるので、このオプションはデフォルトで無効になっています。このオプションは、 SVN サーバーによっては動作しないことがあります（例えば、Google Code Hosting の issue #5471 [http://code.google.com/p/support/issues/detail?id=5471] を参照）。

LogStatusCheck
ログダイアログでは、作業コピーのリビジョンを太字で表示します。しかし、ログダイアログがパスの状態を確認する必要があります。そのため、非常に大きい作業コピーでは時間がかかります。この値を false に設定すると、この機能を無効にすることができます。

Merge log separator
別のブランチからのリビジョンをマージし、マージ追跡情報が使用可能になると、マージされたリビジョンのログメッセージは、コミット時のログメッセージを構成するために収集されます。各リビジョンの個々のログメッセージは、あらかじめ定義された区切り文字列で区切られます。必要に応じて、任意の区切り文字列を指定してください。

OldVersionCheck
TortoiseSVN は週に一度、新しいバージョンが出ているかどうかをチェックします。新しいバージョンが見つかると、コミットダイアログにリンクコントロールが表示されます。 true に設定すると、以前のようにアップデートを通知ダイアログで行うようになります。

OutOfDateRetry
作業コピーが**最新ではない**とエラーが表示された後で、TortoiseSVN が作業コピーを更新するかどうかユーザーに自動的に確認させたくない場合はこの値を false に設定してください。

RepoBrowserPrefetch
Some servers have problems handling the many requests the repository browser makes while it pre-fetches shown folders. To disable the pre-fetching set this value to false.

RepoBrowserShowExternals
Some servers have problems handling the many requests the repository browser makes while it looks for svn:externals. To disable looking for externals, set this value to false.

ShellMenuAccelerators
TortoiseSVN はエクスプローラーのコンテキストメニュー項目に設定されているアクセラレータを使用します（例えば **SVN コミット** のアクセラレータは Alt+C ですが、エクスプローラーの **コピー** 項目と重複します）。TortoiseSVN のエントリでアクセラレータを設定する必要がない場合は false に設定してください。

ShowContextMenuIcons
これは Windows エクスプローラー以外のものを使用したり、コンテキストメニューが正しく表示できないといった問題が発生したりしたときに便利です。 false に設定すると、シェルのコンテキストメニューに TortoiseSVN のアイコンが表示されなくなります。アイコンを表示する場合は、true を設定してください。

ShowAppContextMenuIcons
false に設定すると TortoiseSVN のダイアログで、コンテキストメニューのアイコンが表示されなくなります。.

StyleCommitMessages
コミットダイアログとログダイアログでは、コミットメッセージに装飾（例えばボールド、イタリックなど）を使います（詳しくは「コミットログメッセージ」を参照してください）。これを無効にしたい場合は false に設定してください。

UpdateCheckURL
この値は、TortoiseSVN が有効なアップデートがあるかどうかを検査するためのテキストファイルをダウンロードする URL です。企業の管理者が通知するまでユーザーに TortoiseSVN をアップデートさせたくない場合などに有効です。

VersionCheck
TortoiseSVN は週に一回程度、使用可能な新しいバージョンがあるかどうかを確認します。TortoiseSVN にこのチェックを行わせたくない場合は、この値を false に設定してください。

4.30.11. TortoiseSVN の設定のエクスポート

複数のコンピュータで同じクライアント設定を使用する場合は、Windows のレジストリエディター `regedit32.exe` を使用して実現できます。レジストリキーの `HKCU¥Software¥TortoiseSVN` を開き、reg ファイルにエクスポートしてください。他のコンピュータでは、そのファイルをインポートしてください（通常は、reg ファイルをダブルクリックします）。

Subversion の設定も保存する場合は、`%APPDATA%¥Subversion` にある Subversion の設定ファイルも保存してください。

4.31. 最終ステップ

寄付をお願いします

TortoiseSVN と TortoiseMerge はフリーですが、パッチを送ったり、開発体制内で積極的に役割をこなしたりして、開発者をサポートできます。また、コンピューターの前で延々と時間を過ごす私たちを力づけることができます。

TortoiseSVNで作業している間、私たちは音楽を聴くのが大好きです。我々はプロジェクトに多くの時間を費やしているため、多くの音楽が必要です。そこで、私たちの好きな音楽のCDやDVDのウィッシュリストを、 http://tortoisesvn.net/donate.html で公開しています。また、パッチや翻訳を提供してプロジェクトに貢献している人のリストもご覧ください。

第5章 SubWCRev プログラム

SubWCRev は、Subversion の作業コピーの状態を読み取り、オプションでテンプレートファイルのキーワード置換を行うのに使用する、Windows コンソールアプリケーションです。ビルドプロセスの一部として、ビルドするものに作業コピーの情報を組み込むのにしばしば使用します。典型的なのは、「About」 ボックスにリビジョン番号を含めるのに使用することでしょう。

5.1. SubWCRev コマンドライン

SubWCRevは、デフォルトで作業コピーの外部参照を除いた全ファイルの Subversion の状態を読み込みます。検出した最も大きいコミットリビジョン番号を記録し、そのリビジョンのコミットのタイムスタンプが、それはまた、作業コピーの変更、または更新リビジョンが混在しているかどうかを記録します。リビジョン番号や更新リビジョン範囲、変更状態を標準出力に出力します。

コマンドラインやスクリプトから SubWCRev.exe を呼び出し、コマンドラインパラメーターで制御します。

```
SubWCRev WorkingCopyPath [SrcVersionFile DstVersionFile] [-nmdfe]
```

`WorkingCopyPath` はチェックする作業コピーのパスです。SubWCRev は作業コピーのみで使用でき、直接リポジトリを扱えません。作業コピーへのパスは、絶対パス、相対パスどちらでもかまいません。

リポジトリリビジョンや URL といったフィールドをテキストファイルに保存するため、SubWCRev がキーワード置換を行うようにしたい場合があります。その場合、テンプレートファイル `SrcVersionFile` や、テンプレートの置換バージョンを含む出力ファイル `DstVersionFile` を用意する必要があります。

SubWCRev の動作に影響を与えるオプションスイッチが多数あります。複数を使用する場合は、1つにグループ化して指定する必要があります。例えば、`-n -m` ではなく、`-nm` と指定します。

切り替え	説明
-n	このスイッチが与えられた場合、SubWCRev は `ERRORLEVEL 7` で終了します。このとき作業コピーにはローカルの変更が含まれています。これは、コミットしていない変更が残っていたまま構築するのを防ぎます。

表5.1 使用できるコマンドラインスイッチ一覧

5.2. キーワード置換

元ファイルと先ファイルを与えると、SubWCRev は元ファイルから先ファイルへ、以下のようにキーワード置換を行いながらコピーします。

キーワード	説明
$WCREV$	作業コピー内のもっとも新しいリビジョン番号に置き換わります。

表5.2 使用できるコマンドラインスイッチ一覧

SubWCRev は式のネストを直接サポートしていません。たとえば、次のような式は使用することができません。

```
#define SVN_REVISION "$WCMIXED?$WCRANGE$:$WCREV$$"
```

しかし、次のように形式を変更すれば使用することができます。

```
#define SVN_RANGE       $WCRANGE$
#define SVN_REV         $WCREV$
#define SVN_REVISION    "$WCMIXED?SVN_RANGE:SVN_REV$"
```

ヒント

以上のキーワードのうちいくつかは、作業コピー全体というよりも、単一ファイルに適用されます。そのため、単一ファイルを走査するよう SubWCRev が呼ばれたときにのみ使用する意味があります。これは $WCINSVN$, $WCNEEDSLOCK$, $WCISLOCKED$, $WCLOCKDATE$, $WCLOCKOWNER$ and $WCLOCKCOMMENT$ に適用します。

5.3. キーワード例

以下のサンプルは、テンプレートファイル内のキーワードが、どのように出力ファイルへ置換されるかを示します。

```
// Test file for SubWCRev: testfile.tmpl

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

#if $WCMODS?1:0$
#error Source is modified
#endif

// End of file
```

SubWCRev.exe path¥to¥workingcopy testfile.tmpl testfile.txt の実行後、出力ファイル testfile.txt は以下のようになります。

```
// Test file for SubWCRev: testfile.txt

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

#if 1
#error Source is modified
#endif
```

```
// End of file
```

ヒント

このようなファイルがそのビルドに含まれるため、そのファイルがバージョン管理されていると思われるでしょう。バージョン管理下にあるのはテンプレートファイルであって、生成したファイルではありません。そうでなければ、 バージョンファイルを生成するたびに変更をコミットしなければなりません。そして順次バージョンファイルを更新しなければなりません。

5.4. COM インターフェイス

他のプログラムから Subversion のリビジョン情報にアクセスする必要がある場合、SubWCRev のCOM インターフェイスを使用できます。作成するオブジェクトは SubWCRev.object で、以下のメソッドをサポートしています。

メソッド	説明
.GetWCInfo	この方法は作業コピーを横断してリビジョン情報を集めます。当然、以下のメソッドを呼び出す前に、これを呼び出さなくてはなりません。第 1 引数はパスです。第 2 引数は、フォルダーのリビジョンを含める場合に true としてください。コマンドラインスイッチ -f と等価です。第 3 引数は、svn:externals を含める場合に true としてください。コマンドラインスイッチ -e と等価です。

表5.3 COM オートメーションのサポート

以下のサンプルでは、どのようにインターフェイスを使用するべきかを示しています。

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

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

revObject1 = new ActiveXObject("SubWCRev.object");
revObject2 = new ActiveXObject("SubWCRev.object");
revObject3 = new ActiveXObject("SubWCRev.object");
revObject4 = new ActiveXObject("SubWCRev.object");

revObject1.GetWCInfo(
    filesystem.GetAbsolutePathName("."), 1, 1);
revObject2.GetWCInfo(
    filesystem.GetAbsolutePathName(".."), 1, 1);
revObject3.GetWCInfo(
    filesystem.GetAbsolutePathName("SubWCRev.cpp"), 1, 1);
revObject4.GetWCInfo(
    filesystem.GetAbsolutePathName("..¥¥.."), 1, 1);

wcInfoString1 = "Revision = " + revObject1.Revision +
                "¥nMin Revision = " + revObject1.MinRev +
                "¥nMax Revision = " + revObject1.MaxRev +
                "¥nDate = " + revObject1.Date +
                "¥nURL = " + revObject1.Url + "¥nAuthor = " +
                revObject1.Author + "¥nHasMods = " +
```

```
                revObject1.HasModifications + "¥nIsSvnItem = " +
                revObject1.IsSvnItem + "¥nNeedsLocking = " +
                revObject1.NeedsLocking + "¥nIsLocked = " +
                revObject1.IsLocked + "¥nLockCreationDate = " +
                revObject1.LockCreationDate + "¥nLockOwner = " +
                revObject1.LockOwner + "¥nLockComment = " +
                revObject1.LockComment;
wcInfoString2 = "Revision = " + revObject2.Revision +
                "¥nMin Revision = " + revObject2.MinRev +
                "¥nMax Revision = " + revObject2.MaxRev +
                "¥nDate = " + revObject2.Date +
                "¥nURL = " + revObject2.Url + "¥nAuthor = " +
                revObject2.Author + "¥nHasMods = " +
                revObject2.HasModifications + "¥nIsSvnItem = " +
                revObject2.IsSvnItem + "¥nNeedsLocking = " +
                revObject2.NeedsLocking + "¥nIsLocked = " +
                revObject2.IsLocked + "¥nLockCreationDate = " +
                revObject2.LockCreationDate + "¥nLockOwner = " +
                revObject2.LockOwner + "¥nLockComment = " +
                revObject2.LockComment;
wcInfoString3 = "Revision = " + revObject3.Revision +
                "¥nMin Revision = " + revObject3.MinRev +
                "¥nMax Revision = " + revObject3.MaxRev +
                "¥nDate = " + revObject3.Date +
                "¥nURL = " + revObject3.Url + "¥nAuthor = " +
                revObject3.Author + "¥nHasMods = " +
                revObject3.HasModifications + "¥nIsSvnItem = " +
                revObject3.IsSvnItem + "¥nNeedsLocking = " +
                revObject3.NeedsLocking + "¥nIsLocked = " +
                revObject3.IsLocked + "¥nLockCreationDate = " +
                revObject3.LockCreationDate + "¥nLockOwner = " +
                revObject3.LockOwner + "¥nLockComment = " +
                revObject3.LockComment;
wcInfoString4 = "Revision = " + revObject4.Revision +
                "¥nMin Revision = " + revObject4.MinRev +
                "¥nMax Revision = " + revObject4.MaxRev +
                "¥nDate = " + revObject4.Date +
                "¥nURL = " + revObject4.Url + "¥nAuthor = " +
                revObject4.Author + "¥nHasMods = " +
                revObject4.HasModifications + "¥nIsSvnItem = " +
                revObject4.IsSvnItem + "¥nNeedsLocking = " +
                revObject4.NeedsLocking + "¥nIsLocked = " +
                revObject4.IsLocked + "¥nLockCreationDate = " +
                revObject4.LockCreationDate + "¥nLockOwner = " +
                revObject4.LockOwner + "¥nLockComment = " +
                revObject4.LockComment;

WScript.Echo(wcInfoString1);
WScript.Echo(wcInfoString2);
WScript.Echo(wcInfoString3);
WScript.Echo(wcInfoString4);
```

C# から SubWCRev の COM オブジェクトを使用する方法の例を次に示します。

```
using LibSubWCRev;
SubWCRev sub = new SubWCRev();
sub.GetWCInfo("C:¥¥PathToMyFile¥¥MyFile.cc", true, true);
if (sub.IsSvnItem == true)
{
    MessageBox.Show("バージョン管理対象");
}
else
{
    MessageBox.Show("バージョン管理対象外");
}
```

第6章 IBugtraqProvider インターフェイス

シンプルな bugtraq: プロパティを使用するよりも、より密接に課題管理システムと統合するため、TortoiseSVN は COM プラグインを使用できます。課題管理システムから直接情報を取得できるプラグインとともに使用することでユーザーとやりとりし、TortoiseSVN への未クローズ問題の情報提供やユーザーが入力したログメッセージの検証、さらに問題のクローズといったコミット完了後の処理実行までも行うことができます。

私たちは、各プログラミング言語で COM オブジェクトを実装する方法について、情報やチュートリアルを提供することはできませんが、C++/ATL と C# で書かれたプラグインのサンプルを、私たちのリポジトリの contrib/issue-tracker-plugins フォルダーで提供しています。そのフォルダーからも、プラグインを作成するのに必要なインクルードファイルを入手することができます(リポジトリにアクセスする方法は、「ライセンス」を参照してください)。

重要

プラグインは32ビット版と64ビット版の両方を提供する必要があります。 TortoiseSVN の x64 バージョンでは32ビットのプラグインを使用することができませんし、その逆も同様です。

6.1. 命名規則

もし TortoiseSVN 用の問題追跡プラグインをリリースするならば、Tortoise<何とか>という名前は使用しないでください。私たちはTortoiseというプレフィックスを、Windows に統合するバージョン管理クライアントのために予約しておきたいのです。例えば、TortoiseCVS や TortoiseSVN、TortoiseHg、TortoiseGit、TortoiseBzr など、いずれもバージョン管理クライアントです。

Tortoise クライアントのプラグインは、Turtle<何とか> と命名してください。<何とか> は、接続する問題解決システムの名前になります。または、Turtle と似た発音で、頭文字が異なる言葉を使用しても結構です。例えば、

- Gurtle - Google code のための問題追跡プラグイン
- TurtleMine - Redmine のための問題追跡プラグイン
- VurtleOne - VirsionOne のための問題追跡プラグイン

が良い例です。

6.2. IBugtraqProvider インターフェイス

TortoiseSVN 1.5 以降では、IBugtraqProvider インターフェイスを実装するプラグインを使用することができます。このインターフェイスでは、プラグインが課題追跡を操作するために使用できるいくつかのメソッドが用意されています。

```
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
);
```

このメソッドは、ユーザーがプラグインを追加・設定できる設定ダイアログから呼び出されます。 parameters の文字列は、プラグインに追加情報(例えば問題追跡ツールのURLや、ログイン情報など)を伝えるために使用することができます。プラグインは parameters の文字列を検査し、文字列が有効でなければエラーダイアログを表示する必要すべきです。 hParentWnd はプラグインが表示する各ダイアログの親ウィンドウとして使用します。プラグインは、 parameters の文字列の検査に成功した場合は、TRUE を返さなければなりません。プラグインが FALSE を返すと、設定ダイアログはユーザーが作業コピーのパスにプラグインを追加することを許可しません。

```
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
);
```

プラグインは、TortoiseSVN　のコミットダイアログでプラグインを起動するためのボタン、例えば「問題を選択」や「チケットを選択」などの文字列を提供することができます。文字列がボタン内に収まらないほど長くならないように注意してください。このメソッドがエラーを返した場合(たとえば E_NOTIMPL を返した場合)は、ボタンにはデフォルトのテキストが使用されます。

```
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
);
```

これは、プラグインの中心的なメソッドです。このメソッドは、 TortoiseSVN のコミットダイアログでユーザーがプラグイン用のボタンをクリックしたときに呼び出されます。

parameters の文字列は、設定ダイアログでプラグインの設定を行った際に、ユーザーが設定した文字列です。通常は、プラグインはこの情報を、問題追跡ツールの URL を指定したり、ログイン情報を指定したりするために使用します。

commonRoot の文字列は、コミットダイアログを表示するために選択されたすべての項目の親のパスが入っています。これは、ユーザーがコミットダイアログで選択した全項目のルートパスではないことに注意してください。ブランチ/タグダイアログの場合、これはコピーされるパスです。

pathList パラメーターは、ユーザーがコミットのために選択したパス(文字列)の配列をが含まれています。

originalMessageパラメーターは、コミットダイアログのログメッセージボックスに入力したテキストが含まれています。ユーザーはまだ何もテキストを入力していない場合、この文字列は空になります。

newMessageに返された文字列は、コミットダイアログのログメッセージのエディットボックスにコピーされ、すでに何が書かれていてもそれを置き換えます。プラグインが originalMessage を変更しなかった場合、ここでは同じ文字列を返却する必要があります。そうしなければ、ユーザーが入力した文字列は失われます。

6.3. IBugtraqProvider2 インターフェイス

TortoiseSVN 1.6 では、プラグイン用により機能的な、新しいインターフェイスを追加しました。このIBugtraqProvider2 インターフェイスは IBugtraqProvider を継承しています。

```
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
);
```

このメソッドは、TortoiseSVNのコミットダイアログでユーザーがプラグインボタンをクリックした時に呼び出されます。このメソッドは、 GetCommitMessage() の代わりに呼び出されます。ここで使用されるパラメーターについては、GetCommitMessageのドキュメントを参照してください。

commonURLパラメーターは、コミットダイアログを表示するために選択されたすべての項目の親URLです。これは基本的には commonRoot パスのURLです。

bugID パラメーターは、bug-ID フィールドの内容が内容が含まれています(bugtraq:messageプロパティの設定により、表示される場合)。

返却パラメーターbugIDOutは、メソッドから戻るときにbug-ID フィールドを埋めるために使用されます。

revPropNames や revPropValues 出力パラメーターには、コミット時に設定されるリビジョンのプロパティの名前と値のペアを入れることができます。プラグインは、両方の配列が同じサイズであることを確認する必要があります。revPropNamesには各プロパティの名前、revPropValuesには関連付けられた値を格納します。リビジョンのプロパティが設定される存在しない場合、プラグインが空の配列を返す必要があります。

```
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
);
```

このメソッドは、コミットダイアログが閉じられ、コミットが開始される直前に呼び出されます。プラグインは、コミットのために選択されたファイル・フォルダーや、ユーザーが入力したコミットメッセージを検証するためにこのメソッドを使用することができます。パラメーターにはGetCommitMessage2()の場合と同様で、commonURL についてはすべてのチェックした項目であり、commonRoot はすべてのチェックした項目のルートパスであるということです。

ブランチ/タグダイアログの場合、commonURL はコピー元のURLであり、commonRoot はコピー先のURLが設定されています。

出力パラメーター errorMessage は、TortoiseSVNがユーザーに示すエラーメッセージを設定し、コミットを開始する場合は空にする必要があります。エラーメッセージを返した場合、 TortoiseSVN はダイアログにエラー文字列を表示し、ユーザーが間違いを訂正できるようにコミットダイアログは開いたままにします。したがってプラグインはユーザーに、何が間違っているのかと、それを修正する方法を通知する必要があります。

```
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
);
```

このメソッドは、コミットに成功した時に呼び出されます。プラグインは、例えば、選択した問題を閉じるか、または問題にコミットに関する情報を追加する場合は、このメソッドを使用することができます。パラメーターは、GetCommitMessage2 の場合と同じです。

```
HRESULT HasOptions(
  // Whether the provider provides options
  [out, retval] VARIANT_BOOL *ret
);
```

このメソッドは、ユーザーがプラグインを設定する設定ダイアログから呼び出されます。プラグインがShowOptionsDialogで独自の設定ダイアログを提供している場合 、TRUE を返し、それ以外の場合は FALSE を返す必要があります。

```
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
);
```

このメソッドは設定ダイアログ上で、HasOptions が TRUE を返したときに表示される「オプション」ボタンをユーザーがクリックした時に呼び出されます。プラグインは動作を容易に設定させるために、オプションダイアログを表示することができます。

parameters 文字列には、すでに設定/入力されているプラグインのパラメーターの文字列が含まれています。

newparameters　出力パラメーターには、プラグインが、オプションダイアログで収集した情報から構築されたパラメーター文字列を設定する必要があります。その　paramameters　文字列は、他のすべての　IBugtraqProvider　と IBugtraqProvider2 のメソッドに渡されます。

付録A よくある質問（FAQ）

TortoiseSVNは全力を費やして開発しているので、ドキュメントを完全な状態で維持し続けるのは困難です。私たちはTortoiseSVN のメーリングリストである<dev@tortoisesvn.tigris.org> や <users@tortoisesvn.tigris.org> でよく尋ねられる質問を、オンラインFAQ [http://tortoisesvn.net/faq.html] に公開しています。

私たちは、プロジェクトの 課題追跡ツール [http://code.google.com/p/tortoisesvn/wiki/IssueTracker?tm=3] も維持管理しています。これで私たちのTo-Doリスト上にあるものの一部や、既に修正されたバグについて説明しています。バグを発見した場合や、新しい機能を要求する場合、他の誰かがあなたの前にそこに着いたかどうかを確認するためにまずここをチェックしてください。

回答が得られていない質問がある場合は、次のメーリングリストのいずれかに質問をしてください。

- TortoiseSVN を使用するにあたって質問がある場合に使用する場合は、 <users@tortoisesvn.tigris.org> にメールを送付してください。
- TortoiseSVN の開発に協力して下さる場合は、<dev@tortoisesvn.tigris.org> 上で議論に参加してください 。
- TortoiseSVN のユーザーインターフェイスやドキュメントの翻訳を支援して下さる場合、<translators@tortoisesvn.tigris.org>に電子メールを送信 してください。

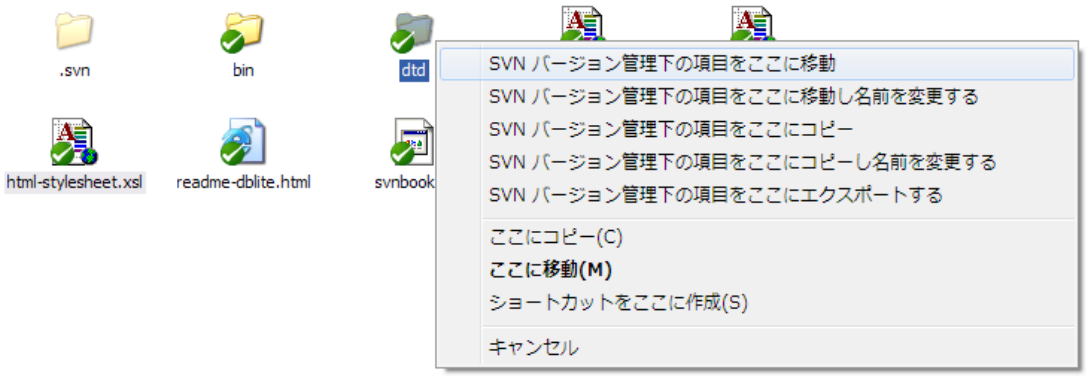
付録B こんなときは……

この付録には TortoiseSVN を使用する際の問題や疑問に対する解決法があります。

B.1. 大量のファイルの同時移動・コピー

ひとつのファイルを移動・コピーするには、TortoiseSVN → 名前を変更... でできます。しかし、この方法でたくさんのファイルを移動・コピーするには、とても遅くたくさん手を動かさなければなりません。

お勧めの方法は、ファイルを新しい場所に右ドラッグすることです。単にファイルの上で右クリックし、マウスボタンを押したままにします。その後、新しい場所にファイルをドラッグし、マウスボタンを離します。コンテキストメニューが表示されるので、コンテキストメニュー → SVN バージョン管理下の項目をここにコピーする か コンテキストメニュー → SVN バージョン管理下の項目をここに移動する を選択します。



B.2. ログメッセージの入力の強制

ログメッセージを空にしてコミットするのを防ぐには、2つ方法があります。TortoiseSVN 特有の方法と、Subversion クライアント全てで使用できる方法ですがサーバーに直接アクセスできる必要があります。

B.2.1. サーバー上のフックスクリプト

リポジトリサーバーに直接アクセスできるのならば、コミット時のログメッセージが、空だったり短すぎると拒否するような pre-commit フックスクリプトをインストールすればいいのです。

サーバーのリポジトリフォルダーには、`hooks` というフックスクリプトのサンプルがあるサブフォルダーがあります。`pre-commit.tmpl` ファイルは、ログメッセージがなかったり短すぎる場合にコミットを拒否するサンプルスクリプトです。このファイルにはこのスクリプトをインストールし使用する方法も書かれています。このファイルの指示に従ってください。

TortoiseSVN 以外に他の Subversion クライアントも使用するようなときに、この方法をお勧めします。欠点は、コミットがサーバーによって拒否され、ユーザーにエラーメッセージが返ることです。クライアントには、コミットの前に拒否されるかどうかわかりません。ログメッセージが十分長くなるまで、TortoiseSVN が OK ボタンを無効にするようにしたければ、以下で説明する方法で行ってください。

B.2.2. プロジェクトプロパティ

TortoiseSVN は機能を制御するのにプロパティを使用します。そういったプロパティのひとつに、`tsvn:logminsize` プロパティがあります。

フォルダーにこのプロパティをセットすると、プロパティで指定した長さ以上ログメッセージを入力するまで、TortoiseSVN はコミットダイアログの OK ボタンを無効にします。

これらのプロジェクトプロパティの詳細については、「プロジェクト設定」を参照してください。

B.3. リポジトリからの選択したファイルの更新

通常、作業コピーを更新するのに、TortoiseSVN → 更新 を使用します。しかし、同僚が、(同時に他のファイルのいずれの変更もマージせず) 追加だけ行った新しいファイルのみを取り出したい場合、違ったアプローチを取る必要があります。

TortoiseSVN → 変更をチェック を使用してください。それから、リポジトリに対して行われた変更を見るのに、リポジトリをチェック をクリックしてください。ローカルに更新したいファイルを選択し、コンテキストメニューから更新してください。

B.4. リポジトリのリビジョンのロールバック(取り消し)

B.4.1. リビジョンログダイアログの使用

ひとつのリビジョンやリビジョンの範囲から、変更を取り消す簡単な方法は、リビジョンログダイアログを使用することです。最近の変更を破棄して以前のリビジョンを HEAD リビジョンにするのもこの方法が使用します。

1. 変更を取り消す必要のあるファイルやフォルダーを選択してください。すべての変更を取り消す場合は、トップレベルフォルダーを選択してください。
2. TortoiseSVN → ログを表示 を選択してリビジョン一覧を表示します。参照したいリビジョンを表示させるために、必要に応じてすべて取得や次の100件を使用してください。
3. 変更を取り消したいリビジョンを選択してください。リビジョン範囲を取り消すのなら、最初のものを選択してから、Shift キーを押したまま最後のものを選択してください。複数のリビジョンを選択する際、選択状態にすき間があってはなりません。選択したリビジョンで 右クリック し、コンテキストメニュー → このリビジョンにおける変更を取り消す を選択してください。
4. もしくは、もっと古いリビジョンを HEAD リビジョンにしたい場合は、選択したリビジョンの上で 右クリック し、コンテキストメニュー → このリビジョンに戻す を選択してください。これにより、選択したリビジョン以降の変更がすべて取り消されます。

変更の取り消しは作業コピーに対して行われます。結果を確認し、変更をコミットしてください。

B.4.2. マージダイアログの使用

広い範囲のリビジョンを取り消したい場合、マージダイアログも使用できます。前の方法は裏でマージしていましたが、この方法では、明示的に使用します。

1. 作業コピーで TortoiseSVN → マージ を選択してください。
2. 元: フィールドに、作業コピーで取り消したい変更のあるブランチやタグの完全なフォルダーの URL を入力してください。デフォルトの URL が入力されているはずです。
3. 元リビジョン フィールドに現在のリビジョン番号を入力してください。他の誰も変更を加えていないことが確認できている場合、HEAD リビジョンを指定できます。
4. "元:" URL を使用する チェックボックスにチェックが着いていることを確認してください。
5. 先リビジョンフィールドに戻す先のリビジョン番号を入力してください。すなわち、戻す先頭のリビジョンの前のものです。
6. マージを完了するには OK をクリックしてください

変更の取り消しは作業コピーに対して行われます。結果を確認し、変更をコミットしてください。

B.4.3. svndumpfilter の使用

TortoiseSVN は決してデータを失わないため、「ロールバックした」リビジョンはまだ中間リビジョンとして存在しています。 HEAD リビジョンが以前の状態になっただけです。すべての痕跡を削除して、リポジトリから完全に見えなくする場合は、もっと極端な手段を採らねばなりません。これは本当に十分な理由がなければ、お勧めしません。考えられる理由としては、誰かが公開リポジトリに機密文書をコミットしたなどでしょう。

リポジトリからデータを削除する唯一の方法は、Subversion コマンドラインツールの `svnadmin` を使うことです。この実行方法の説明は、Repository Maintenance [http://svnbook.red-bean.com/en/1.7/svn.reposadmin.maint.html] にあります。

B.5. ファイルやフォルダーに対して 2 リビジョン間の比較

項目の履歴にある2つのリビジョン（例えば同じファイルのリビジョン100と200）を比較したい場合、単純に **TortoiseSVN → ログを表示** を使用して、そのファイルのリビジョンの履歴を表示します。そこで、比較したい2つのリビジョンを取り出し、**コンテキストメニュー → リビジョンを比較** を使用してください。

2つの別ツリーにある同じ項目（例えばトランクとブランチ）を比較したい場合、両方のツリーを開くのにリポジトリブラウザーを使用できます。それから同じ場所のファイルを選択し、**コンテキストメニュー → リビジョンの比較** を使用してください。

2つのツリーでどこが変更されか（例えばトランクとタグ付けされたリリース）を比較したい場合、**TortoiseSVN → リビジョングラフ** を使用できます。比較する2つのノードを選択し、**コンテキストメニュー → HEADリビジョンを比較** を使用してください。これで変更されたファイルの一覧が表示され、変更の詳細を見るよう特定のファイルを選択できます。また、変更したファイルすべてを含むツリー構造をエクスポートしたり、単純に変更したファイルをすべて表示したりできます。詳細は 「フォルダーの比較」 をご覧ください。その他には、変更の全サマリを最小コンテキストで見るのに、**コンテキストメニュー → HEADリビジョンに対するUnified形式での差分** を使用してください。

B.6. 共通のサブプロジェクトを含める

時に、作業コピーに別プロジェクト、おそらくライブラリのコードを含めたくなるでしょう。オリジナルの（そしてメンテナンスされる）コードとの関連が切れてしまうので、リポジトリにこのコードの複製を作りたくないでしょう。また、複数のプロジェクトでコアコードを共有したいかもしれません。これに対処する方法が少なくとも3つあります。

B.6.1. svn:externals の使用

プロジェクトにあるフォルダーに `svn:externals` プロパティをセットしてください。このプロパティは複数行からなります。各行には、共通コードをチェックアウトするフォルダーとして使用するサブフォルダー名と、そこにチェックアウトするリポジトリの URL を記述します。もっと詳しいことは「外部項目」をご覧ください。

新しいフォルダーをコミットしてください。そこで更新すると、Subversion は各リポジトリから作業コピーに、プロジェクトのコピーを取得します。必要なサブフォルダーは自動的に作成されます。メインの作業コピーを更新する度にも、全外部プロジェクトの最新版を取得します。

外部プロジェクトが同じリポジトリにある場合、メインプロジェクトの変更をコミットすると、変更がコミットリストに含まれます。

外部プロジェクトが別のリポジトリにある場合、メインプロジェクトのコミット時に、外部プロジェクトに対する変更は通知されますが、別々にコミットする必要があります。

説明した3つの方法のうち、クライアント側でセットアップが必要ないものは1つだけです。一度フォルダープロパティに外部参照を設定すると、すべてのクライアントで、更新時に設定したフォルダーを取得します。

B.6.2. ネストした作業コピーの使用

プロジェクトに共通コードを格納する新しいフォルダーを作成してください。ですがまだ Subversion に追加しないでください。

新しいフォルダーで TortoiseSVN → チェックアウト を選択し、共通コードのコピーをここにチェックアウトしてください。これで、メインの作業コピーの中に独立した作業コピーをネストさせます。

2つの作業コピーは独立しています。親作業コピーの変更をコミットした際には、ネストした作業コピーの変更は無視されます。同様に、親作業コピーを更新した際には、ネストした作業コピーは更新されません。

B.6.3. 相対位置の使用

同じ共通コアコードを複数のプロジェクトで使用するが、使用するプロジェクトごとに複数の同じ作業コピーを維持したくなければ、使用するその他のプロジェクトと関連する、独立した場所にチェックアウトできます。例えば以下のようになります。

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

その上で、共通コードには相対パスで参照してください。例: ..¥..¥Common¥DSPcore

プロジェクトが関係ない場所に散乱している場合、以下のバリエーションを使用できます。共通コードを一カ所に置き、その場所をプロジェクト内でハードコードしているものになるようドライブレター変換を使用します。例えば、共通コードを D:¥Documents¥Framework か C:¥Documents and Settings¥{login}¥My Documents¥framework にチェックアウトし、

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

として自分のソースコード内で使用するドライブマッピングを作成してください。自分のコードでは絶対パスを指定できます。

```
#include "X:¥superio¥superio.h"
```

この方法は、全てが PC の環境でしか利用できません。また、必要なドライブマッピングを明文化しておかないと、チームはファイルがどこにあるか分からなくなってしまいます。はっきり言って、この方法は閉じた開発環境向けで、一般的にはお勧めできません。

B.7. リポジトリへのショートカットの作成

特定の場所をリポジトリブラウザーでたびたび開く必要があるのなら、TortoiseProc の自動化インターフェイスを使用して、デスクトップショートカットを作成できます。単に新しいショートカットを作成し、項目の場所を以下のようにセットしてください。

```
TortoiseProc.exe /command:repobrowser /path:"url/to/repository"
```

もちろん本当のリポジトリ URL を含める必要があります。

B.8. バージョン管理外のファイルの無視

たまたま無視するはずのファイルを追加してしまったら、どのようにしたらそのファイルを失わずにバージョン管理下から外せるのでしょうか? おそらくプロジェクトの一部ではない、しかし長時間かけてお好みに設定した IDE の設定ファイルがあることでしょう。

まだ追加をコミットしていない場合は、TortoiseSVNの → 追加を元に戻す... を使用することで追加を取り消すことができます。その後、誤って後で再び追加しないように、無視リストにファイル(複数可)を追加する必要があります。

ファイルがリポジトリ内に既にある場合は、リポジトリから削除する必要がありますし、無視リストに追加します。幸いなことに TortoiseSVN にはこれを行うための便利なショートカットがあります。TortoiseSVN → バージョン管理から除外し、無視リストに追加 を実行すれば、まずリポジトリからファイルやフォルダーを削除するようにマークし、ローカルコピーを保持します。そして、アイテムを無視リストに追加し、誤って再び Subversion に追加されないようにします。この後、必要があるのは親フォルダーをコミットすることだけです。

B.9. 作業コピーをバージョン管理外に

作業コピーから .svn ディレクトリを削除し、通常のフォルダーツリーにする場合、これを単純にエクスポートできます。どのようにするかは、「作業コピーをバージョン管理外へ」をご覧ください。

B.10. 作業コピーの削除

不要になった作業コピーをきれいに削除するには、どのようにすればよいのでしょうか。簡単です。 Windows エクスプローラーで、フォルダーごと削除するだけです。作業コピーはプライベートな存在であり、内部で自己完結しています。Windows エクスプローラーで作業コピーを削除しても、リポジトリ内のデータにはまったく影響しません。

付録C 管理者向けの便利な小技

この付録には TortoiseSVN を複数のコンピュータに配置する上での 問題や疑問に対する解決法があります。

C.1. グループポリシーでの TortoiseSVN のデプロイ

TortoiseSVN インストーラーは MSI ファイルとなっています。これはドメインコントローラのグループポリシーに対して MSI ファイルを追加する際に、問題が発生しないことを意味しています。

Microsoft のナレッジベース 314934 http://support.microsoft.com/?kbid=314934 で得られる方法はよくまとまっています。

TortoiseSVNは、ユーザーの構成ではなくコンピュータの構成の下にインストールする必要があります。TortoiseSVN は CRT と MFC DLL を必要としており、それはユーザーごとではなくコンピュータごとに配布されるからです。どうしてもユーザーごとに TortoiseSVN をインストールする必要がある場合は、TortoiseSVN をインストールするすべてのコンピュータに対して、Microsoft から MFC と CRT のパッケージのバージョン10をインストールする必要があります。

すべてのユーザーを同じ設定にしたいのであれば、 MSI ファイルをカスタマイズすることができます。 TSVN の設定は、レジストリの HKEY_CURRENT_USER¥Software¥TortoiseSVN 以下に登録され、一般的な Subversion の設定は、%APPDATA%¥Subversion にある設定ファイルに記述されています。 MSI のカスタマイズ方法については、 MSI 改変フォーラムに行くか、ウェブで「MSI の改変」を検索してください。

C.2. 更新チェックのリダイレクト

TortoiseSVN は数日おきに、新しいバージョンが出ているかをばチェックします。使用可能な新しいバージョンがある場合は、コミットダイアログに通知が表示されます。

図C.1 アップグレードの通知を表示するコミットダイアログ

自分のドメインにいるたくさんのユーザーに対して責任を負っているなら、確認をとった特定バージョンのみ使用させ、最新版をインストールさせたくないでしょう。おそらくユーザーがすぐにアップグレードしないよう、アップグレードダイアログを表示させたくないことでしょう。

TortoiseSVN 1.4.0 以降では、イントラネットのサーバーにその更新チェックのリダイレクトすることができます。レジストリキーのHKCU¥Software¥TortoiseSVN¥UpdateCheckURL （文字列値）をご使用のイントラネット内のテキストファイルを指すURLに設定してください。そのテキストファイルには、次の形式である必要があります。

```
1.4.1.6000
TortoiseSVN の新しいバージョンが使用可能です。
http://192.168.2.1/downloads/TortoiseSVN-1.4.1.6000-svn-1.4.0.msi
```

ファイルの最初の行はバージョン文字列です。TortoiseSVNのインストールパッケージの正確なバージョン文字列と一致していることを確認する必要があります。2行目は、コミットダイアログに表示する文字列です。任意の文字列を設定することができます。ただし、コミットダイアログの容量が制限されていることに注意してください。メッセージが長すぎると、切り捨てられてしまいます。3行目は、新しいインストールパッケージへのURLです。コミットダイアログでカスタムメッセージラベルをユーザーがクリックすると、このURLが開かれます。また、単に直接ではなく、MSIファイルのWebページを指すことができます。 URLは、Webページを指定した場合には、そのページが開かれ、ユーザーに表示される、デフォルトのWebブラウザーで開かれます。 MSIパッケージを指定した場合、ブラウザーはローカルでMSIファイルを保存するようにユーザーに要求します。

C.3. SVN_ASP_DOT_NET_HACK 環境変数の設定

何でもかんでもすべてを選択してしまうユーザーに多くの問題と混乱を発生させたので、バージョン1.4.0以降では、TortoiseSVNのインストーラーがSVN_ASP_DOT_NET_HACK 環境変数を設定するオプションを提供していません。

しかし、このオプションはユーザーに対して隠されているだけです。TortoiseSVN のインストーラーにこの環境変数を設定するように強制するには、ASPDOTNETHACK プロパティを TRUE にしてください。例えば、インストーラーを起動する際に以下のようにしてください。

```
msiexec /i TortoiseSVN-1.4.0.msi ASPDOTNETHACK=TRUE
```

重要

なお、これは未だに VS.NET2002 を使用している人にのみ必要です。 Visual Studio のそれ以降のバージョンでは、この対策は必要ありません。それよりも古いツールを使用している人は、使用をやめるべきです!

C.4. コンテキストメニューエントリの無効化

バージョン 1.5.0 以降では、TortoiseSVN はコンテキストメニューエントリを無効(実際には非表示)にできます。コンパイル上の理由だけではなく、この機能は気軽につかうべきではありませんので、GUI は用意されておらず、レジストリを直接操作する必要があります。特定のコマンドを使うべきでないユーザーがいる場合に、そのコマンドを無効にできます。なお、エクスプローラー のコンテキストメニューエントリを隠すだけで、他の方法では実行することができます。例えば、コマンドラインや、TortoiseSVN 自体の他のダイアログから実行する場合などです。

コンテキストメニューを表示する際の情報を保持しているレジストリキーは、HKEY_CURRENT_USER¥Software¥TortoiseSVN¥ContextMenuEntriesMaskLow と HKEY_CURRENT_USER¥Software¥TortoiseSVN¥ContextMenuEntriesMaskHigh です。

それぞれのレジストリエントリは、特定のメニューエントリに対応する各ビットの DWORD 値です。ビットをセットすると対応するメニューエントリが無効になります。

値	メニューエントリ
0x0000000000000001	チェックアウト

表C.1 メニューエントリとその値

例:「再配置」・「バージョン管理外の項目を削除する」・「設定」といったメニューエントリを無効にするには、以下のように値を設定してください。

```
  0x0000000000080000
+ 0x0000000080000000
+ 0x2000000000000000
= 0x2000000080080000
```

下位の DWORD 値 (0x80080000) は HKEY_CURRENT_USER¥Software¥TortoiseSVN¥ContextMenuEntriesMaskLow に格納せねばならず、上位の DWORD 値(0x20000000)は HKEY_CURRENT_USER¥Software¥TortoiseSVN¥ContextMenuEntriesMaskHigh に格納せねばなりません。

メニューエントリを、再度有効にするには、単純に2つのレジストリキーを削除してください。

付録D TortoiseSVN の自動化

すべての TortoiseSVN のコマンドがコマンドライン引数で制御できます。バッチファイルで自動化したり、テキストエディターなどの他のプログラムから、特定のコマンドやダイアログを起動したりすることができます。

重要

TortoiseSVN はあくまで GUI クライアントです。ここでは TortoiseSVN のダイアログを表示させて、ユーザーに入力を促す方法を説明します。入力を伴わずに操作をするスクリプトを書くのであれば、公式の Subversion コマンドラインクライアントを使用してください。

D.1. TortoiseSVNのコマンド

TortoiseSVN の GUI プログラムは TortoiseProc.exe から呼び出します。実行するコマンドは /command:<コマンド名> 引数で指定し、<コマンド名> の部分にはコマンド名を指定します。ほとんどのコマンドでは /path:"<パス名>" 引数でパスを指定する必要があります。下記の表で、/command:<コマンド名> で指定するコマンドと、/path:"<パス名>" で指定するパスの意味を説明します。

コマンドによっては、複数の対象のパスを受け付けます（例えば、コミット時に複数のファイルを指定する場合など）。 /path 引数には * で区切って複数のパスを指定することができます。

ファイルで複数のパスを指定することもできます。複数のパスを改行で区切って記述します。ファイルは UTF-16 形式で、 BOM [http://ja.wikipedia.org/wiki/%E3%83%90%E3%82%A4%E3%83%88%E3%82%AA%E3%83%BC%E3%83%80%E3%83%BC%E3%83%9E%E3%83%BC%E3%82%AF] を入れないでください。ファイルで指定したい場合は、/path の代わりに /pathfile を使用してください。/deletepathfile を使用すると、コマンド実行後に TortoiseProc がファイルを自動的に削除します。

多くのコマンド（コミットや更新など）で表示される進行ダイアログは、通常はコマンドが終了した後も OK ボタンを押すまで開いたままになります。設定ダイアログで進行ダイアログを自動的に閉じるように設定することができます。ただしこの場合、コマンドをバッチファイルから起動しても、TortoiseSVN コンテキストメニューから起動しても、進行ダイアログが閉じるようになります。

/configdir:"<設定ファイルのディレクトリのパス>" を指定すると、異なる場所にある設定ファイルを指定することができます。この引数は、レジストリで設定されているデフォルトパスを上書きします。

/closeonend 引数を指定すると、設定を変更することなくコマンド終了時に進行ダイアログを自動で閉じることができます。

- /closeonend:0 ダイアログを自動的に閉じません。
- /closeonend:1 エラーがなければ自動的に閉じます。
- /closeonend:2 エラーや競合がなければ自動的に閉じます。
- /closeonend:3 エラー、競合、マージがなければ自動的に閉じます。

エラーや競合がない場合にローカル操作の進行ダイアログを自動的に閉じるには、/closeforlocal 引数を指定してください。

以下の表には、TortoiseProc.exe のコマンドラインで使用してアクセスすることができる、すべてのコマンドを一覧しています。上で述べたように、/command:<コマンド名> の形で使用してください。この表では紙面を節約するため、頭に付ける /command は省略しています。

コマンド	説明
:about	「バージョン情報」ダイアログを表示します。コマンドが省略された場合もバージョン情報ダイアログを表示します。

表D.1 使用できるコマンドとオプションの一覧

コマンドの例(1行で入力してください)

```
TortoiseProc.exe /command:commit
                 /path:"c:¥svn_wc¥file1.txt*c:¥svn_wc¥file2.txt"
                 /logmsg:"ログメッセージのテスト" /closeonend:0

TortoiseProc.exe /command:update /path:"c:¥svn_wc¥" /closeonend:0

TortoiseProc.exe /command:log /path:"c:¥svn_wc¥file1.txt"
                 /startrev:50 /endrev:60 /closeonend:0
```

D.2. Tsvncmd URL ハンドラ

特殊な URL を使用すると、web ページから TortoiseProc を呼び出すことができます。

TortoiseSVN は、TortoiseSVN コマンドを実行するようなハイパーリンクを作成する、tsvncmd: という新しいプロトコルを登録します。コマンドやパラメーターは、コマンドラインから TortoiseSVN を自動化するのと同じです。

tsvncmd: URL の形式は以下のようになります。

tsvncmd:command:cmd?parameter:paramvalue?parameter:paramvalue

cmd は許可されたコマンドの1つ、parameter は path や revision のような引数の名前、そして paramvalue は引数に設定される値です。引数のリストの数は、使われるコマンドに依存します。

以下のコマンドを tsvncmd: URL に使用できます。

- :update
- :commit
- :diff
- :repobrowser
- :checkout
- :export
- :blame
- :repostatus
- :revisiongraph
- :showcompare

簡単な URL の例としては、次のものがあります。

```
<a href="tsvncmd:command:update?path:c:¥svn_wc?rev:1234">Update</a>
```

また、より複雑なケースでは次のものがあります。

```
<a href="tsvncmd:command:showcompare?
url1:https://stexbar.googlecode.com/svn/trunk/StExBar/src/setup/Setup.wxs?
url2:https://stexbar.googlecode.com/svn/trunk/StExBar/src/setup/Setup.wxs?
revision1:188?revision2:189">compare</a>
```

D.3. TortoiseIDiff コマンド

画像差分ツールには、起動時のふるまいを制御するのに使用する、コマンドラインオプションが少しあります。オプションは TortoiseIDiff.exe により呼ばれます。

以下の表は、コマンドラインで画像差分ツールに渡す、全オプションの一覧です。

オプション	説明
:left	左に表示するファイルのパス。

表D.2 使用できるオプションの一覧

サンプル(一行で入力してください):

```
TortoiseIDiff.exe /left:"c:¥images¥img1.jpg" /lefttitle:"image 1"
                  /right:"c:¥images¥img2.jpg" /righttitle:"image 2"
                  /fit /overlay
```

付録E コマンドラインインターフェイスのクロスリファレンス

時々このマニュアルは、コマンドラインインターフェイス(CLI)の説明をしている Subversion のドキュメントを参照しています。TortoiseSVN があるシーンの裏で何を行っているか理解する助けになるよう、それぞれの TortoiseSVN の GUI 操作と同等な CLI コマンドを示すリストを用意しました。

> **注記**
>
> TortoiseSVN で行うことと同等な物が CLI にありますが、TortoiseSVN が CLI を呼ぶことはなく、直接 Subversion ライブラリを使用するのを覚えていてください。

TortoiseSVN のバグを見つけたら、Subversion の問題か TortoiseSVN の問題かを切り分けるために、CLI を使用してそれを再現するようにお願いするかもしれません。このリファレンスで、どのコマンドを行うべきか確認してください。

E.1. 規約と基本規則

以下の説明では、リポジトリの位置を表す URL (例: `http://tortoisesvn.googlecode.com/svn/trunk/`)は、単に URL と表します。作業コピーのパス(例: `C:¥TortoiseSVN¥trunk`)は、単に PATH と表します。

> **重要**
>
> TortoiseSVN は Windows シェル拡張ですから、カレントワーキングディレクトリの概念は使用できません。作業コピーのパスは相対パスではなく絶対パスで与えられます。

いくつかの項目はオプションで、TortoiseSVN ではチェックボックスやラジオボタンで制御されています。こういったオプションはコマンドラインの定義では [角かっこ] で表されます。

E.2. TortoiseSVNのコマンド

E.2.1. チェックアウト

```
svn checkout [-depth ARG] [--ignore-externals] [-r rev] URL PATH
```

depthの引数は、「チェックアウトする深さ」コンボボックスに関連します。

外部参照を除外する がチェックされているなら、`--ignore-externals` スイッチを使用してください。

特定のリビジョンをチェックアウトしたい場合、`-r` スイッチを使って指定してください。

E.2.2. 更新

```
svn info URL_of_WC
svn update [-r rev] PATH
```

複数の項目を更新するのに、現在の Subversion では不可分操作できません。そこで TortoiseSVN は、まず HEAD リビジョンをリポジトリから探し出し、複数のリビジョン場混じった作業コピーを作らないように、特定のリビジョン番号を持つすべての項目を更新します。

1項目しか選択しないで更新したり、選択した項目がすべて同じリポジトリにない場合、TortoiseSVN は HEAD リビジョンに更新します。

ここでのコマンドラインオプションはありません。**特定リビジョンへ更新** も update コマンドを実装していますが、こちらはもっとオプションがあります。

E.2.3. リビジョンの更新

```
svn info URL_of_WC
svn update [-r rev] [-depth ARG] [--ignore-externals] PATH
```

depthの引数は、「チェックアウトする深さ」コンボボックスに関連します。

外部参照を除外する がチェックされているなら、`--ignore-externals` スイッチを使用してください。

E.2.4. コミット

TortoiseSVN ではコミットダイアログを使って複数の Subversion コマンドを実行します。最初は、コミットできる作業コピーの項目について状態チェックを行います。リストを確認し、コミットするよう選択したファイルと BASE に diff をかけることができます。

```
svn status -v PATH
```

バージョン管理外のファイルを表示にチェックがあると、TortoiseSVN は作業コピーの階層にあるバージョン管理外のファイルやフォルダーも（無視ルールを考慮して）表示します。`svn status` コマンドはバージョン管理外のフォルダー以下は検索しないため、この特殊な機能は直接対応する Subversion コマンドがありません。

バージョン管理外のファイルやフォルダーをチェックすると、最初にその項目を作業コピーへ追加します。

```
svn add PATH...
```

OK をクリックすると、その場で Subversion のコミット（commit）が実行されます。ファイル選択のチェックボックスをすべてデフォルト状態のままにすると、作業コピーの再帰的コミットが実行されます。ファイルを選択を外した場合、コマンドラインで個別にパスを指定した形で、非再帰的コミット（`-N`）が実行されます。

```
svn commit -m "LogMessage" [-depth ARG] [--no-unlock] PATH...
```

LogMessage にはログメッセージエディットボックスの内容を記述してください。空でもかまいません。

ロックを保持 にチェックしているなら、`--no-unlock` スイッチを使用してください。

E.2.5. 差分

```
svn diff PATH
```

メインのコンテキストメニューから差分が実行された場合、BASEリビジョンから変更されたファイルに対して比較を行います。上記のCLIコマンドの出力では比較を行い、結果をUnified差分ファイルで出力します。しかし、TortoiseSVNは

これを使用しているわけではありません。TortoiseSVNはTortoiseMerge（または、任意の差分表示プログラム）でフルテキストファイルの差分を視覚的に表示します、そのため、CLIで直接相当するものはありません。

TortoiseSVN で、バージョン管理下であってもなくても、任意の 2 ファイルを比較することができます。TortoiseSVNは、単に2つのファイルを選択した差分プログラムに渡し、どこに差分があるかを比較させます。

E.2.6. ログの表示

```
svn log -v -r 0:N --limit 100 [--stop-on-copy] PATH
  もしくは
svn log -v -r M:N [--stop-on-copy] PATH
```

デフォルトでは TortoiseSVN は --limit を使用してログメッセージを 100 個取得しようとします。古い API を使用するような設定になっていれば、リポジトリのリビジョンを100個分、ログメッセージを取得するには、2 番目の形式を使用してください。

コピー/名前の変更が発生したら停止 がチェックされている場合は、`--stop-on-copy` スイッチを使用してください。

E.2.7. 変更をチェック

```
svn status -v PATH
  または
svn status -u -v PATH
```

状態チェックの初期状態では、作業コピーに対してのみ行います。**リポジトリをチェック** をクリックすると、更新時にどのファイルが変更されるかリポジトリもチェックしますが、これには -u スイッチが必要です。

バージョン管理外のファイルを表示にチェックがあると、TortoiseSVN は作業コピーの階層にあるバージョン管理外のファイルやフォルダーも（無視ルールを考慮して）表示します。`svn status` コマンドはバージョン管理外のフォルダー以下は検索しないため、この特殊な機能は直接対応する Subversion コマンドがありません。

E.2.8. リビジョングラフ

リビジョングラフは TortoiseSVN のみの機能です。コマンドラインにはありません。

TortoiseSVN は

```
svn info URL_of_WC
svn log -v URL
```

を行い（URLはリポジトリの root）、返ってきたデータを解析します。

E.2.9. リポジトリブラウザー

```
svn info URL_of_WC
svn list [-r rev] -v URL
```

リポジトリブラウザーでトップレベルに表示されるリポジトリのルートを決めるのに `svn info` を使用できます。このレベル以上へは 上 へ行けません。またこのコマンドは、リポジトリブラウザーに表示されるロック情報を返します。

`svn list` は URL とリビジョンで指定した、ディレクトリの内容をリスト表示します。

E.2.10. 競合の編集

このコマンドには、同等の CLI がありません。競合したファイルを見て、どの行を使用するか整理するのに、TortoiseMerge や、その他の3画面差分・マージツールを起動します。

E.2.11. 解決済み

```
svn resolved PATH
```

E.2.12. 名前変更

```
svn rename CURR_PATH NEW_PATH
```

E.2.13. 削除

```
svn delete PATH
```

E.2.14. 変更の取り消し

```
svn status -v PATH
```

第一段階として、作業コピー内で変更の取り消しが行える項目を決定するよう、状態チェックを行います。一覧を確認し、BASE に対するファイルの比較、取り消す項目の選択を行えます。

OK をクリックすると、その場で Subversion の変更の取り消し(revert)が実行されます。ファイル選択のチェックボックスをすべてデフォルト状態のままにすると、作業コピーの再帰的な(-R)変更の取り消しが実行されます。ファイルの選択を外した場合、コマンドラインで個別にパスを指定した形で、変更の取り消しが実行されます。

```
svn revert [-R] PATH...
```

E.2.15. クリーンアップ

```
svn cleanup PATH
```

E.2.16. ロックの取得

```
svn status -v PATH
```

第一段階として、作業コピー内でロックできる項目を決定するよう、状態チェックを行います。ロックしたい項目を選択できます。

```
svn lock -m "LockMessage" [--force] PATH...
```

LockMessage には、ロックメッセージエディットボックスの内容を記述してください。空でもかまいません。

ロックを奪う にチェックが付いている場合、--force スイッチを使用してください。

E.2.17. ロックの解除

```
svn unlock PATH
```

E.2.18. ブランチ・タグ

```
svn copy -m "LogMessage" URL URL
  または
svn copy -m "LogMessage" URL@rev URL@rev
  または
svn copy -m "LogMessage" PATH URL
```

ブランチ/タグダイアログはリポジトリへのコピーを実行します。以下のようにラジオボタンのオプションが3つあります。

- リポジトリ内の最新リビジョン
- リポジトリのリビジョンを指定
- 作業コピー

これは上記のコマンドライン 3 種に該当します。

LogMessage にはログメッセージエディットボックスの内容を記述してください。空でもかまいません。

E.2.19. 切り替え

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

E.2.20. マージ

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

マージのテスト は、--dry-run スイッチ付きの merge と同じ動作をします。

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

Unified diff はマージ処理されるものを差分で表示します。

E.2.21. エクスポート

```
svn export [-r rev] [--ignore-externals] URL Export_PATH
```

この形式はバージョン管理外フォルダーからアクセスし、エクスポート先に指定フォルダーを使用します。

作業コピーを別の場所にエクスポートするのに、Subversion　ライブラリを使用ません。そのため、相当するコマンドラインが存在しません。

TortoiseSVN が行うのは、操作の進行を表示しながら、新しい場所に全ファイルをコピーすることです。バージョン管理外ファイル/フォルダーも、オプションでエクスポートできます。

どちらの場合でも、外部参照を除外する がチェックされていれば、--ignore-externals スイッチを使用してください。

E.2.22. 再配置

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

E.2.23. ここにリポジトリを作成

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

E.2.24. 追加

```
svn add PATH...
```

フォルダーを選択すると、TortoiseSVN はまず、追加できる項目を再帰的に検索します。

E.2.25. インポート

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

LogMessage にはログメッセージエディットボックスの内容を記述してください。空でもかまいません。

E.2.26. 注釈履歴

```
svn blame -r N:M -v PATH
svn log -r N:M PATH
```

注釈情報を参照するのに TortoiseBlame を使用する場合、ツールチップにログメッセージを表示するのに、ファイルのログも必要です。注釈をテキストファイルで参照する場合は、この情報は必要ありません。

E.2.27. 無視リストに追加

```
svn propget svn:ignore PATH > tempfile
{tempfile を編集し無視する項目を追加してください}
svn propset svn:ignore -F tempfile PATH
```

svn:ignore プロパティは複数行の値をとることがあるため、ここでは、直接コマンドラインで指定するのではなく、テキストファイルで変更する方法を示します。

E.2.28. パッチを作成

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

TortoiseSVN は、作業コピーと BASE リビジョンを比較して、unified差分形式のパッチファイルを作成します。

E.2.29. パッチの適用

パッチと作業コピーが同じリビジョンでない場合、パッチの適用は困難な作業です。幸いなことに、Subversionには該当する機能がありませんが、TortoiseMergeを使用sるうことができます。

付録F 実装の詳細

この付録には TortoiseSVN の機能を実装する上での、議論の詳細が納められています。

F.1. アイコンオーバーレイ

すべてのファイルやフォルダーは、Subversion ライブラリが報告する Subversion の状態値を持っています。コマンドラインクライアントでは、1 文字のコードで表示しますが、TortoiseSVN ではアイコンオーバーレイでグラフィカルに表示します。オーバーレイアイコンの数は非常に制限されているので、オーバーレイアイコンごとに複数の状態値を持つこともありえます。

競合 オーバーレイアイコンは、更新や切替の結果でローカルの変更とリポジトリからダウンロードした変更が競合した場合の、**競合**状態を表示するのに使用します。操作が完了できない時に陥る、**妨害**状態を表すときにも表示します。

変更 オーバーレイアイコンは、ローカルで変更がある**変更**状態や、ローカルの変更にリポジトリからの変更点をマージした**マージ**状態、ファイルが削除されて、同名の異なるファイルに置き換わった **置換**状態を表示するのに使用します。

削除 オーバーレイアイコンは、削除するようスケジュールした **削除**状態や、見つからない **紛失**状態の場合に表示されます。通常、紛失ファイルは表示できませんが、紛失ファイルがあるはずの親フォルダーに表示します。

バージョン管理に 追加 されるファイルやフォルダーには + 記号がつきます。

Subversion 内 オーバーレイアイコンは、**通常**状態のファイルや、まだ状態が分からないファイルに表示します。これは、TortoiseSVN が状態を取得するバックグラウンドプロセスを用いているためで、それはオーバーレイアイコンを更新する数秒前かもしれません。

要ロックオーバーレイは、`svn:needs-lock` プロパティが設定されているファイルに表示されます。

ローカルの作業コピーでファイルにロックされている場合、ロック オーバーレイアイコンが使われます。

無視 オーバーレイアイコンは、常に無視するパターンに含まれていたり、親フォルダーに `svn:ignore` プロパティが設定されていたりするために **無視** 状態になった項目を表すのに使用します。このオーバーレイはオプションです。

バージョン管理外 オーバーレイは、**バージョン管理外** 状態の項目を表すのに使用します。これはバージョン管理下にあるフォルダー内の、バージョン管理外の項目です。このオーバーレイはオプションです。

Subversion の状態が none（作業コピーにない項目）の場合はオーバーレイを表示しません。無視 や バージョン管理外 のオーバーレイを無効にした場合は、そのファイルにもオーバーレイを表示しません。

ファイルは Subversion 状態を1つしか持ちません。例えば、ファイルをローカルで変更し、そのファイルを同時に削除するようマークをつけることができます。Subversion は状態値を1つ返します。この場合、**削除**です。この優先順は、Subversion で定義しています。

TortoiseSVN が再帰的な状態を表示する場合（デフォルト設定）、各フォルダーには、自身の状態と以下のすべての子ファイルの状態を反映させたオーバーレイが表示されます。ひとつにまとめたオーバーレイを表示するのに、前述の優先順位が使用されます。この際、もっとも優先順が高いのは、競合状態です。

実はシステムで使用できるアイコンのすべてを見ることはありません。Windows で使用できるオーバーレイの数は 15 個に制限されているからです。Windows は 4 個使用し、残りの 11 個を他のアプリケーションで使用できるようにしています。TortoiseCVS も使用していると、使用できるオーバーレイスロットが足らなくならないように、TortoiseSVN は「Good Citizen (TM)」に徹し、オーバーレイを他のアプリケーションが使用できるように制限しているのです。

Tortoise クライアントには他のバージョン管理システム向けのものもあるため、オーバーレイアイコンの表示を共有するコンポーネントを作成しました。ここでは技術的な細部は重要ではありませんが、知っておいてほしいのは、共有コンポーネントによってすべての Tortoise クライアントが同じオーバーレイを使用することができ、複数の Tortoise クライアントをインストールしたとき、オーバーレイの最大数である11個を超えずに使うことができる点です。もちろん、若干の欠点もあります。すべての Tortoise クライアントが同じオーバーレイを使用するため、作業コピーが使用しているバージョン管理システムをオーバーレイアイコンで見分けることができません。

- 通常、変更、競合 は常に読み込み・表示されます。
- 削除 は可能ならばロードしますが、スロットが足りなければ 変更 にフォールバックします。
- 読み取り専用 は可能ならばロードしますが、スロットが足りなければ 通常 にフォールバックします。
- ロックは可能ならばロードしますが、スロットが足りなければ 通常 にフォールバックします。
- 追加 は可能ならばロードしますが、スロットが足りなければ 変更 にフォールバックします。

付録G 言語パックとスペルチェッカー

標準のインストーラーは英語のみをサポートしていますが、個別に言語パックやスペルチェック辞書をダウンロードし、インストールすることができます。

G.1. 言語パック

TortoiseSVN のユーザーインターフェイスはたくさんの言語に翻訳されており、ご希望の言語パックをダウンロードできるでしょう。言語パックは私たちの翻訳ページ translation status page [http://tortoisesvn.net/translation_status] で見つけられると思います。まだ言語パックが用意されていないのなら、チームに参加して翻訳をしてみましょう ;-)

各言語パックは、.msiインストーラーとしてパッケージ化されています。インストールプログラムを実行し、指示に従うだけです。インストールが完了したら、翻訳版が使用できるようになります。

ドキュメントは、いくつかの言語に翻訳されています。私たちのウェブサイトのサポートページ [http://tortoisesvn.net/support]から翻訳されたマニュアルをダウンロードすることができます。

G.2. スペルチェッカー

TortoiseSVN にはコミットログメッセージをチェックするようスペルチェッカーが含まれています。プロジェクト言語が、自分のネイティブ言語でない場合に特に便利です。スペルチェッカーは OpenOffice [http://openoffice.org] や Mozilla [http://mozilla.org] と同じ辞書を使用します。

インストーラーは自動でUS ないし UK の英語辞書を追加します。その他の言語を使用したければ、一番簡単なのは TortoiseSVN の言語パックをインストールすることです。TortoiseSVN のローカルユーザーインターフェイスと同様に、適当なディレクトリにインストールされます。再起動するとこの辞書が有効になります。

もしくは辞書を自分でインストールすることもできます。OpenOffice か Mozilla がすでにインストールされていれば、そのアプリケーションがインストールされているフォルダーにある辞書をコピーできます。そうでなければ、辞書ファイルを http://wiki.services.openoffice.org/wiki/Dictionaries からダウンロードする必要があります。

辞書ファイルを取得したら、おそらくロケール文字のみの形に名前を変更する必要があります。例:

- en_US.aff
- en_US.dic

これを TortoiseSVN をインストールしたフォルダーの bin サブフォルダーにコピーするだけです。通常は C:¥Program Files¥TortoiseSVN¥bin のはずです。bin サブフォルダーを散らかしたくなければ、その場所の代わりに、スペルチェッカーファイルを C:¥Program Files¥TortoiseSVN¥Languages に置けます。そのフォルダーがそこになければ、はじめに作成してください。TortoiseSVN の次回起動時にスペルチェッカーが有効になります。

複数の辞書をインストールした場合、TortoiseSVN はどの辞書を使用するかを以下のルールで決定します。

用語集

BASE リビジョン	作業コピー にあるファイルやフォルダーの現在のベースリビジョンで、最後にチェックアウト、更新、コミットを実行したときの、ファイルやフォルダーのリビジョンです。 BASE リビジョンは、通常は HEAD リビジョンと同じではありません。
BDB	Berkeley DB 。実績のあるリポジトリ用データベースバックエンドですが、ネットワークフォルダー上では使用できません。1.2以前のリポジトリのデフォルトです。
FSFS	Subversion が持つリポジトリ用のファイルシステムバックエンドです。ネットワークフォルダー上で使用することができます。1.2以降のリポジトリのデフォルトです。
GPO	グループポリシーオブジェクト。
HEAD リビジョン	リポジトリにあるファイルやフォルダーの最新リビジョンです。
SVN	Subversion のよく使われる省略表現。 「svnserve」リポジトリサーバーで使われる、Subversion カスタムプロトコルの名前。
インポート	単一のリビジョンで、フォルダー階層の内容をリポジトリにインポートする Subversion のコマンドです。
エクスポート	このコマンドは、作業コピーのようにバージョン管理下のフォルダーをコピーしますが、`.svn` は作成しません。
クリーンアップ	Subversion book から引用します。「作業コピーの再帰的なクリーンアップ（ロックの解除、未完操作の回復）を行います。 作業コピーがロックされています というエラーが出続ける場合、このコマンドを実行し、古くなったロックを削除し、作業コピーをまた使えるようにします。」ここで言う ロック とは、ファイルシステムのロックを指しており、リポジトリのロックではないことに注意してください。
コピー	Subversion リポジトリでは、単一ファイルやツリー全体のコピーを作成できます。これは、領域を消費しないように、オリジナルへのリンクに少し似ている「簡易コピー」で実装されています。コピーの作成ではコピー内に履歴を保存します。そのためコピーされる前についても追跡できます。
コミット	ローカルの作業コピーの変更点をリポジトリに渡し、リポジトリのリビジョンを新しく作成するのに使用する Subversion のコマンドです。
チェックアウト	空のディレクトリに、リポジトリからバージョン管理下のファイルをダウンロードし、ローカルの作業コピーを作成する Subversion のコマンドです。
パッチ	作業コピーの変更がテキストファイルのみである場合、Subversion の Diff コマンドを使用すると、変更内容を Unified 差分ファイルとして単一ファイルに出力することができます。この種のファイルはよく「パッチ」と呼ばれており、他の誰か（やメーリングリスト）にメールで送ったり、他の作業コピーに適用したりすることができます。コミットのアクセス権を持たない人は、変更を作成し、パッチファイルをコミット権限を持つ人に送ることができます。また、変更に自信がない場合、他の人にパッチを送ってレビューしてもらうこともできます。
ブランチ	バージョン管理システムでよく使用される用語で、ある時点で開発が2つの独立したパスに分岐したことをと表します。メインの開発ラインからブランチを作成すれば、メインライン

を不安定にせずに新機能の開発を行うことができます。また、今後バグフィックスのみを行うための安定版リリースのブランチを作成すれば、新機能の開発は不安定なトランクで行うことができます。Subversion のブランチは、「簡易コピー」として実装されています。

プロパティ

ディレクトリやファイルのバージョン管理をするのに加えて、Subversion はバージョン管理されたメタデータを追加できます。これは、バージョン管理下のディレクトリやファイルごとに「プロパティ」として参照されます。プロパティには、レジストリキーと同じように、それぞれ名前と値があります。Subversion には、svn:eol-style のような内部で使用する特殊なプロパティがいくつかあります。TortoiseSVN にも tsvn:logminsize のような特殊なプロパティがあります。任意の名前と値を持つプロパティの追加もできます。

マージ

リポジトリに追加された変更を、ローカルで行った変更を壊さないように、作業コピーに追加するプロセスです。時には自動的に調整できず、作業コピーが競合と呼ばれる状態になります。

作業コピーを更新する際に、自動的にマージが行われます。また、TortoiseSVN のマージコマンドを使用して、別のブランチにある変更を指定してマージすることもできます。

リビジョン

変更セットをコミットするたびに、新しい「リビジョン」がリポジトリに作成されます。各リビジョンは、履歴の決まった時点のリポジトリツリーの状態を表します。過去にさかのぼる場合は、リビジョン N のような形でリポジトリを調べられます。

言い換えるとリビジョンは、リビジョンが作成された時に行われた変更を示しています。

リビジョンプロパティ（revprop）

ファイルがプロパティを持てるように、リポジトリの各リビジョンもプロパティを持つことができます。リビジョンが作成されるときに、svn:date svn:author svn:log といった特殊なリビジョンプロパティが自動的に作成され、それぞれコミット日時、コミットした人、ログメッセージを表しています。これらのプロパティは編集できますが、バージョン管理できません。そのため変更は永続的で元に戻せません。

リポジトリ

データを集中して格納し保守する場所。複数のデータベースやファイルをネットワーク上に分散して置くこともでき、ネットワークに出ずに直接アクセスできる場所に置くこともできます。

ログ

ファイルやフォルダーのリビジョンの歴史を表します。「履歴」とも呼びます。

ロック

バージョン管理下の項目のロックを取得すると、その作業コピーを除いて、ロックが解除されるまでリポジトリにコミット不可の印が付きます。

作業コピー

ローカルの「サンドボックス」で、バージョン管理ファイルに対して作業を行う場所です。また通常ローカルのハードディスクに記録されています。リポジトリからの「チェックアウト」で作業コピーを作成し、「コミット」で変更点をリポジトリに反映します。

再配置

サーバー上の異なるディレクトリに移動したり、ドメイン名が変更されたりして、リポジトリが移動した場合、作業コピーを「再配置」すれば、作業コピーのリポジトリのURLが新しい場所を指すようになります。

【注意】このコマンドは、作業コピーが指している同じリポジトリの同じ場所を指し、そのリポジトリが移動されてしまったときのみに使用してください。その他の場合には、代わりに「切り替え」コマンドを使用する必要があります。

切り替え

ちょうど「特定リビジョンへ更新」が履歴上の別の位置へ、作業コピーの時間ウィンドウを変更するように、「切り替え」はリポジトリの別の位置へ、作業コピーの場所ウィンドウを変

	更します。トランクとブランチの違いが少ない時、それぞれの作業する際に役に立ちます。その2つの間で作業コピーを切り替えると、差異のあるファイルのみが転送されます。
削除	バージョン管理下のファイルを削除（してコミット）すると、リポジトリ内のそのコミットを行ったバージョン以降、その項目はもう存在しなくなります。しかしもちろん、それ以前のリポジトリのリビジョンには、まだ存在していますから、まだそれにアクセスできます。必要なら削除した項目をコピーし、履歴を含め完全に「復活」できます。
変更の取り消し	作業コピーを最後に更新したとき、 Subversion はそれぞれのファイルの「当時の」コピーをローカルに保持しています。変更を行った後で、その変更を取り消したい場合は、「変更の取り消し」コマンドを使用して当時のコピーに戻せます。
履歴	ファイルやフォルダーのリビジョンの歴史を表します。「ログ」とも呼びます。
差分	「差分の表示」の略。どのような変更が行われたのか正確に見たいときに便利です。
更新	リポジトリから作業コピーへ最新の変更点を取得するコマンド。作業コピーの変更点に、他の人が行った変更をマージします。
注釈履歴	このコマンドはテキストファイル専用で、それぞれの行が最後に変更された時の、リポジトリのリビジョンと作者を注釈表示します。私たちの GUI の実装は TortoiseBlame と呼ばれ、リビジョン番号の上にマウスを置くと、コミット日時やログメッセージも表示します。
競合	リポジトリの変更がローカルにマージされる際、時には同じ行に変更がある場合があります。この場合、 Subversion はどちらを使用するか自動的に決定できません。これを競合と呼びます。それ以降の変更をコミットする前には、ファイルを手作業で編集し競合を解決しなければなりません。
競合の解決	作業コピーのファイルが、マージ後に競合状態になったままになった場合、人間がエディター（または TortoiseMerge）で競合を整理しなければなりません。このプロセスは「競合の解決」と言われています。競合したファイルを解決済みにすると、コミットできるようになります。
追加	ファイルやディレクトリをリポジトリに追加する Subversion コマンドです。新しい項目は、コミットした際にリポジトリに追加されます。

索引

シンボル

その場でインポート, 31
アイコン, 46
アクセス, 19
インストール, 1
インポート, 29
ウィンドウズのプロパティ, 48
ウェブサイト, 23
ウェブビュー, 128
エクスプローラー, xiii
エクスプローラーの列, 48
エクスポート, 121
オーバーレイ, 46, 190
オーバーレイの優先度, 190
キーワード, 80
キーワード展開, 80
クライアントフック, 152
クリーン, 78
クリーンアップ, 79
グラフ, 116
グループポリシー, 178, 179
コピー, 93, 113
コマンドライン, 181, 183
コマンドラインクライアント, 184
コミット, 34
コミットを元に戻す, 174
コミットメッセージ, 53, 173
コンテキストメニュー, 25
コンテキストメニューエントリ, 179
サウンド, 129
サーバーの移動, 122
サーバーサイドアクション, 113
サーバービューアー, 113
サーバー側フックスクリプト, 22
ショートカット, 176
ステータス, 46, 49
スペルチェッカー, 192
タグ, 72, 93
チェックアウト, 31
チェックアウトリンク, 23
チェックイン, 34
ツリーの競合, 42
デプロイ, 178
ドメインコントローラ, 178
ドラッグハンドラ, 27
ドラッグ&ドロップ, 27
ネットワークフォルダー, 20
バグ追跡, 123
バグ追跡システム, 123, 123
バックアップ, 22
バージョン, 178
バージョン抽出, 162
バージョン管理, xiii
バージョン管理の削除, 177
バージョン管理外, 122, 177
バージョン管理外の「作業コピー」, 121
バージョン管理外のファイル・フォルダー, 73
パターンマッチ, 74
パッチ, 109
ファイルのコピー, 72
ファイルのバージョン番号, 162
ファイルの移動, 72
ファイルを比較, 175
ファイル名の変更, 72
フォルダーの比較, 175
フック, 22
フックスクリプト, 22, 152
ブランチ, 72, 93
プラグイン, 167
プロキシサーバー, 143
プロジェクトプロパティ, 83
ベンダプロジェクト, 175
マージ, 97
  リビジョン範囲, 98
  再統合, 99
  2つのツリー, 100
マージの再統合, 104
マージの競合, 103
マージツール, 70
マージ追跡, 103
マージ追跡ログ, 60
リビジョン, 14, 116
リビジョンの比較, 68
リビジョングラフ, 116
リビジョンプロパティ, 61
リポジトリ, 7, 29
リポジトリ URL の変更, 122
リポジトリから分離, 177
リポジトリへのファイル追加, 29
リポジトリビューアー, 128
リポジトリブラウザー, 113
リポジトリ作成, 18

リリースとしてマーク, 93
リンク, 23
レジストリ, 157
ログ, 53
ログや作者の編集, 61
ログキャッシュ, 149
ログメッセージ, 53, 173
ロック, 105
ロールバック, 174
一時ファイル, 29
作成, 18
　TortoiseSVN, 18
作業コピー, 12
作業コピーの作成, 31
作業コピーの状態, 46
共通プロジェクト, 175
再編成, 173
再配置, 122
切り替え, 96
削除, 75, 75
取り消し, 78
右クリック, 25
右ドラッグ, 27
名前変更, 76, 113, 173
変更, 49
変更のエクスポート, 68
変更の取り消し, 78, 174
変更を元に戻す, 174
変更リスト, 51
変更点, 175
変更点の取得, 40
変更点の表示, 46
変更点の送信, 34
外部リポジトリ, 90
外部参照, 90, 175
展開, 74
履歴, 53
差分, 66, 109
差分ツール, 70
差分比較, 51
常に無視する, 130
新規ファイルのバージョン管理, 71
更新, 40, 174
更新チェック, 178
最大化, 29
最新バージョンのチェック, 178
機能の無効化, 179
比較, 66
注釈, 110
注釈履歴, 110
無視, 73
特殊なファイル, 31
画像の差分, 69
移動, 76, 173
移動したサーバー, 122
空のメッセージ, 173
競合, 11, 41
絞り込み, 61
統計, 63
翻訳, 192
自動化, 181, 182, 183
解決, 42
言語パック, 192
設定, 129
認証, 28
認証キャッシュ, 28
読み取り専用, 105
課題追跡システム, 123, 167
賞賛, 110
辞書, 192
追加, 71
除外パターン, 130

A

ASP プロジェクト, 179
auto-props, 82

C

CLI, 184
COM, 162, 167
COM SubWCRev インターフェイス, 164

F

FAQ, 172

G

GPO, 178

I

IBugtraqProvider, 167

M

Microsoft Word, 70
msi, 178

R

revprops, 61

S

SUBST ドライブ, 141
Subversion book, 7
Subversion のプロパティ, 80
SubWCRev, 162
SVN_ASP_DOT_NET_HACK, 179

T

TortoiseIDiff, 69
TortoiseSVN プロパティ, 83
TortoiseSVNリンク, 23

U

UNC パス, 19
Unified差分ファイル, 109
URL の変更, 122
URL ハンドラ, 182

V

ViewVC, 128
VS2003, 179

W

WebSVN, 128
Windowsシェル, xiii