TortoiseSVN

Subversion クライアント for Windows

Version 1.6.16

Stefan Küng
Lübbe Onken
Simon Large

TortoiseSVN: Subversion クライアント for Windows: Version 1.6.16

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

発行日 2011/06/10 17:41:21 (r21547)

目次

図の一覧

表の一覧

序

- チームで作業していますか?
- 作業していたファイルに対して、他の誰かが同じファイルを同時に作業していた、なんてことはありませんか? そのせいで変更を失ってしまったことは?
- ファイルを保存して、その変更を取り消したいと思ったことはありませんか?ファイルが少し前の状態と、どれくらい違うのか知りたいと思ったことはないですか?
- プロジェクトにバグを発見したとき、そのバグがいつ混入したのか知りたいと思ったことはありませんか?

これらの質問に「はい」と答えたのなら、TortoiseSVN はあなたのためにこそあります! 読み進めて、TortoiseSVN がどのようにあなたの作業の助けになるかを見つけてください。そんなに難しくありません。

1. 読者

本書は、データ管理するのに Subversion を使用したいが、コマンドラインクライアントを使用するのに抵抗がある、コンピュータに詳しい人向けに書かれています。TortoiseSVN は Windows のシェル拡張であるため、ユーザは Windows のエクスプローラになじみがあり、使い方を知っていると仮定しています。

2. このガイドの読み方

序 では TortoiseSVN プロジェクト、作業している人々のコミュニティ、利用条件、頒布について簡単に説明しています。

1章はじめに では TortoiseSVN とは何か、何をする物なのか、由来はどこなのか、PC へのインストールの基本について説明しています。

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の簡単に解説しています。これは Subversion プロジェクトよりドキュメントを借りてきており、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 PC での Subversion や TortoiseSVN のテストに便利な、ローカルリポジトリのセットアップの方法を説明しています。また、サーバにあるリポジトリにも関係する、リポジトリ管理についても少し説明しています。また必要な場合にサーバを瀬戸アップする方法についての節もあります。

4章日常操作ガイド は最も重要な節で、TortoiseSVN の全機能とその使い方を説明しています。作業コピーのチェックアウト、修正、変更点のコミットなどのチュートリアルの形を取っています。そこからさらに高度な話題に進んでください。

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。あなたのプロジェクトの構築情報を含めるのに便利です。

付録B どうしたら…… 節では、他の部分で明示されていない作業を行う上で、共通の質問に対する答えを述べています。

付録D TortoiseSVN の自動化 節では TortoiseSVN 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。ユーザの操作が必要なスクリプトに対して便利です。

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

3. TortoiseSVN は自由!

TortoiseSVN は自由です。お金を払う必要はありませんし、どんな用途にも使用できます。GNU General Public License (GPL) のもと開発されています。

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

4. コミュニティ

TortoiseSVN と Subversion のどちらも、それぞれのプロジェクトで作業する人々からなるコミュニティで開発を行っています。世界中のそれぞれの国から参加し、一緒にすばらしいプログラムを開発しています。

5. 謝辞

Tim Kemp
: TortoiseSVN プロジェクトの立ち上げに

Stefan Küng
: TortoiseSVN が今の形になるための大変な作業に

Lübbe Onken
: きれいなアイコン、ロゴ、バグつぶし、翻訳と翻訳管理に

Simon Large
: 文書化の手伝いや、バグつぶしに

The Subversion Book
: Subversion のすばらしい導入に (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の助け合いに

われらが寄進者達
: 送ってくれた音楽で楽しんだ時間に

6. 本書で使用している用語

簡単に読み進められるよう、TortoiseSVN の画面やメニューの名前にはすべて、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 ダイアログごとに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。

ユーザインターフェースのボタンはこのように表示します。「続けるには OK を押してください」

ユーザの行うアクションは太字で表しました。Alt+A は、キーボードの Alt キーを押しながら A キーを押すといった具合です。右ドラッグ は、マウスの右ボタンを押しながら別の場所に 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

便利な Tips です。

注意

自分が何をしているか注意しなければならない箇所です。

警告

極端に注意を払わなければなりません。この警告を無視すると、データの破損やその他困ったことが起きるでしょう。

第1章 はじめに

バージョン管理は情報の変化に対応する技術です。これは (小さな変更を加えては、次の日にそれをチェックしたり取り消したりといった) プログラマにとって長い間重要なツールであり続けました。同時に働いている (もっといえば全く同じファイルを同時に変更を加える) プログラマチームを想像してみてください。こんな潜在的な混沌を管理する良いシステムがなぜ必要なのかお判りになると思います。

1.1. TortoiseSVN とは?

TortoiseSVN は、Subversion バージョン管理システム用のフリーでオープンソースなクライアントで、ファイルやディレクトリを時系列で管理します。ファイルはリポジトリという中心的な場所に格納されます。リポジトリは通常のファイルサーバとよく似ていますが、ファイルやディレクトリに加えた変更をすべて記録しています。これにより、ファイルを古いバージョンに戻したり、いつ、誰が、どのようにデータに変更を加えたかを履歴から確認できます。そのため、Subversion や一般的なバージョン管理システムを一種の「タイムマシン」と考える人たちもたくさんいます。

いくつかのバージョン管理システムは、ソフトウェア構成管理 (SCM) システムでもあります。こういったシステムはソースコードのツリーを管理するために特別にあつらえられており、ソフトウェア開発に特化したたくさんの機能を持っています。- 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です。しかし、Subversion はそういったシステムではありません。いずれの (ソースコードを含む) ファイルの集合を管理する汎用システムです。

1.2. TortoiseSVN の歴史

2002年、Tim Kemp は Subversion というすばらしいバージョン管理システムと出会いましたが、使いやすい GUI クライアントが不足していました。Windows シェル統合した Subversion クライアントというアイディアは TortoiseCVS という CVS 用の似たようなクライアントから着想しました。

Tim は TortoiseCVS のソースコードを解析し、TortoiseSVN の基礎に利用しました。その後プロジェクトを開始し、`tortoisesvn.org` ドメインを登録し、ソースコードをオンラインに配置しました。この間、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 のソースコードに出会いました。TortoiseSVN はまだ使い物になりませんでしたが、このプロジェクトに参加しプログラミングを始めました。Stefan はすぐに、オリジナルのコードが全く残らないほど既存のコードをほとんど書き換え、コマンドや機能を追加していきました。

Subversion が安定するようになるにつれて、Subversion クライアントとして TortoiseSVN を使用するユーザが増えてきました。ユーザベース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 のロゴとすばらしいアイコンで協力すると申し出てくれました。また彼はウェブサイトと翻訳の管理をしてくれています。

1.3. TortoiseSVN の特徴

TortoiseSVN はどのようにすばらしい Subversion クライアントなのでしょうか? 以下は簡単な特徴の一覧です。

シェル統合

TortoiseSVN は Windows シェル (例: エクスプローラ) にシームレスに統合されます。これは、すでによく知っているツールを使い続けられるということです。そして、バージョン管理機能が必要となるたびにちがうアプリケーションに変更する必要はありません!

ですが、Windows エクスプローラを使うよう強制されるわけではありません。TortoiseSVN のコンテキストメニューは、他の多くのファイルマネージャや、ほとんどの標準的なWindowsアプリケーションで共通のファイル/オープン

ダイアログで動作します。しかし、TortoiseSVN はわざわざ Windows エクスプローラの拡張として開発されていることを覚えておいてください。他のアプリケーションでは完全に統合されているわけではなく、例えばアイコンオーバレイが表示されないといったことがあり得ます。

アイコンオーバレイ
ファイルやフォルダごとにバージョン管理の状態をあらわす小さなオーバレイアイコンを表示します。これで作業コピーの状態をすぐに見ることができます。

Subversion コマンドへの容易なアクセス
すべての Subversion コマンドにエクスプローラのコンテキストメニューからアクセスできます。TortoiseSVN は独自のサブメニューも追加します。

TortoiseSVN は Subversion クライアントですから、Subversion 自体が持つ特徴も以下に述べます。

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 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。ファイルとディレクトリがバージョン管理されます。その結果、クライアント側の move や copy コマンドでファイルやディレクトリの操作を実際に行うことができます。

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な固まりごとに作成し、変更をコミットできます。

バージョン管理のメタデータ
ファイルやディレクトリごとに不可視の「属性」が添付されています。望むまま任意のキー/値を作成・格納できます。属性はファイルの内容のようにずっとバージョン管理されていきます。

ネットワークレイヤの選択
Subversion にはリポジトリアクセス用に抽象的な概念を持っていて、新しいネットワークメカニズムを簡単に実装できるようになっています。Subversion の「先進的」 ネットワークサーバは Apache ウェブサーバのモジュールになっており、WebDAV/DeltaV と呼ばれる HTTP の一種を話します。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、例えば、認証、認可、データ圧縮、リポジトリ閲覧といった、様々な重要な特徴をすぐに使用できます。また、もっと小さなスタンドアロン Subversion サーバも使用できます。このサーバは ssh で簡単にトンネル通信ができる独自プロトコルを話します。

一貫したデータの取り扱い
Subversion はファイルの差分を、バイナリ差分アルゴリズムを用いて表現します。これ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) でも同様です。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送信されます。

効果的なブランチ・タグ付け
ブランチやタグをつけるコストは、プロジェクトの規模に比例する必要はありません。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く、一定の時間で完了し、リポジトリのとても小さな領域しか使用しません。

いじりやすさ
Subversion は歴史的なしがらみを持っていません。よく練られた API でできた C の共有ライブラリの集まりで実装されています。これにより、Subversion は非常に保守しやすく、他の言語やアプリケーションから利用しやすくなっています。

1.4. TortoiseSVN のインストール

1.4.1. 必要なシステム

TortoiseSVN は Windows 2000 SP2, WinXP 以上で動作します。Windows 98, Windows ME, Windows NT4 は TortoiseSVN 1.2.0 からサポートされませんが、本当に必要なら旧バージョンをまだダウンロードできます。

TortoiseSVN のインストール中やインストール後に問題が発生したら、はじめに 付録A よくある質問 (FAQ) を参照してください。

1.4.2. インストール

TortoiseSVN には簡単に使えるインストーラがあります。インストーラファイルをダブルクリックし、後は手順に従ってください。後の面倒はインストーラが見てくれます。

重要

TortoiseSVN をインストールするには、管理者権限が必要です。

1.4.3. 言語パック

TortoiseSVN のユーザインタフェー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;-)

言語パックごとに `.exe` インストーラ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、説明に従ってください。再起動すると翻訳が有効になります。

1.4.4. スペルチェッカ

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が含まれています。プロジェクト言語が、あなた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 のローカルユーザインターフェースと同様に、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の `bin` サブフォルダ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、スペルチェッカファイルを `C:¥Program Files¥TortoiseSVN¥Languages` に置けます。そのフォルダがそこになければ、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。

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 以下で読むことができます。http://svnbook.red-bean.com/ (訳注 日本語では http://subversion.bluegate.org/doc/book.html で読めます)

この章では簡単におおざっぱな Subversion の概要を説明します。バージョン管理システムが初めてなら、この章が明らかにしてくれるでしょう。一般的なバージョン管理の概念から始めて、Subversion の根底にあるアイディアを説明し、Subversion の使い方の簡単な例をお見せします。

この章の例では複数のプログラムソースコードの共有を扱いますが、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。コンピュータプログラマ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 (典型的なのは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利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。

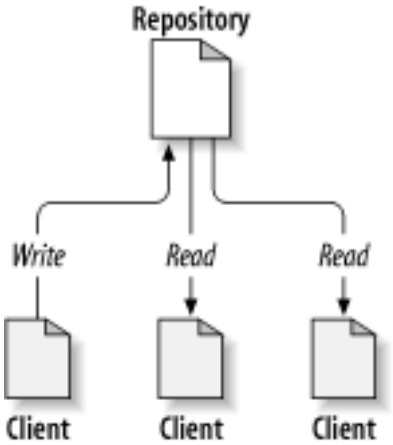


図2.1 典型的なクライアント・サーバシステム

このどこが興味深いというのでしょうか? そこまでは普通のファイルサーバと同じように聞こえます。確かにリポジトリはファイルサーバの一種ですが、いつも目にす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ということです。これはすべてのファイルに対するすべての変更と、さらにディレクトリツリー自体に対して行われたファイルやディレクトリの追加・削除・再配置といった変更も含まれています。

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後のバージョンだけが見えます。が、ファイルシステムの 以前の状態も閲覧することができます。 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どのファイルがあったの?」、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?」 といった履歴に関する質問をすることができます。この手の質問はすべての バージョン管理システム のキモになるような質問です。つまりバージョン管理システムは、時間と共に修正されるデータを記録したり、修正内容を追跡したり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

バージョン管理システムは皆、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。リポジトリ内の他の人の変更を、誤って上書きしてしまうことは容易に起こりえます。

2.2.1. ファイル共有の問題

こんなシナリオを考えてみてください。2 人の同僚 Harry と Sally がいます。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 Harry が変更を保存してから、(数分後に) Sally 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かので) Harry 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、Harry が行った修正は Sally の新バージョンに隠されて見えません。Sally が Harry の変更を取り込まなかったからです。おそらく偶然ですが、Harry の作業は事実上失われたままです。こういった状況を確実に回避したいのです!

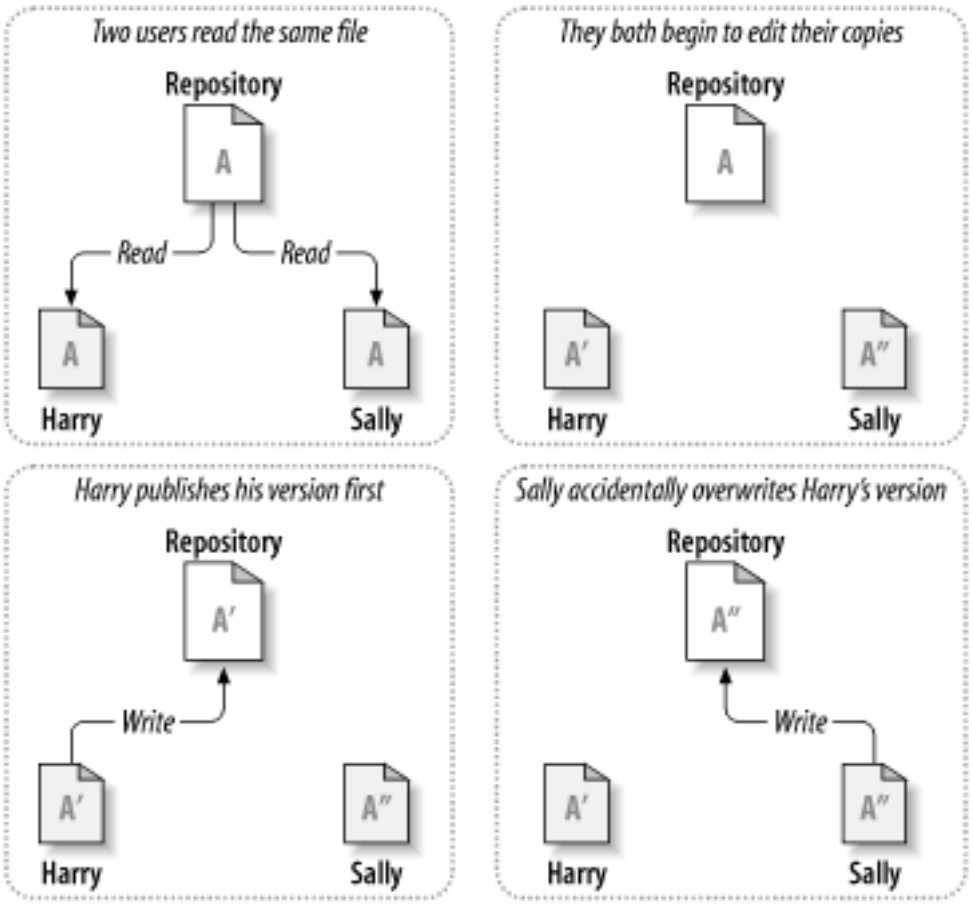


図2.2 回避したい問題

2.2.2. ロック・変更・アンロック法

多くのバージョンコントロールシステムでは、とても単純な方法である ロック・修正・アンロック モデルでこの問題に取り組んでいます。こういったシステムでは、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 Harry は変更を加える前に ロック をしなければなりません。ファイルは貸し出された図書館の本のように、Harry が ファイルにロックをかけると、Sally はそこに変更を加えられなくなります。Sally 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、Harry が変更を終えてロックを解放するのを待つことだけです。Harry が変更を終えてロックを解放すると Sally はロック・編集できるようになります。

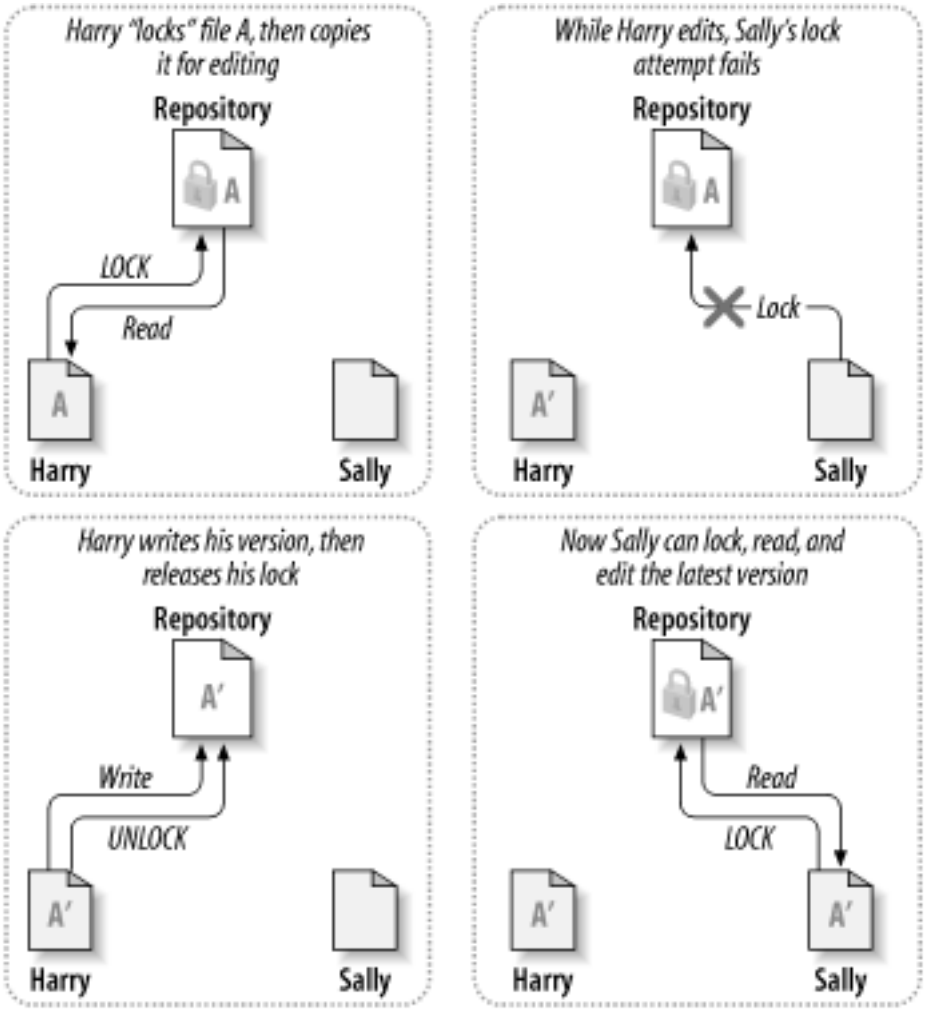


図2.3 ロック・変更・アンロック法

ロック・修正・アンロックモデルの問題は、少々制限が厳しいことで、しばしばユーザ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。時々 Harry はファイルをロックしたままそれを忘れてしまうことがあります。いっぽう Sally 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。Harry が休暇を取ってしまったりすると、Sally は管理者に Harry のロックを解放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。

- ロックは不要な直列化を起こす可能性があります。Harry がテキストファイルの先頭を編集していて、Sally 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ます。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。こういった状況ではロックは必要ありません。

- ロックは誤った安心感を与える可能性があります。Harry はファイル A をロックして編集し、Sally はファイル B をロックして編集するとします。A と B はお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 がもう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。Harry も Sally 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・マージ法

Subversion や CVS、その他のバージョンコントロールシステムは、コピー・変更・マージ モデルをロックの代わりに使用します。このモデルではユーザのクライアントごと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。そこからユーザは平行して作業し、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、正しくできたかどうかの最終的な責任は人間が負うことになります。

以下に例を挙げます。Harry と Sally 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 Sally がリポジトリに変更を保存しました。そのあと Harry が変更を適用しようとしましたが、Harry 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 Harry が最後にコピーしたときから何か変更が加わっています。そこで Harry 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。おそらく Sally の変更は Harry 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。

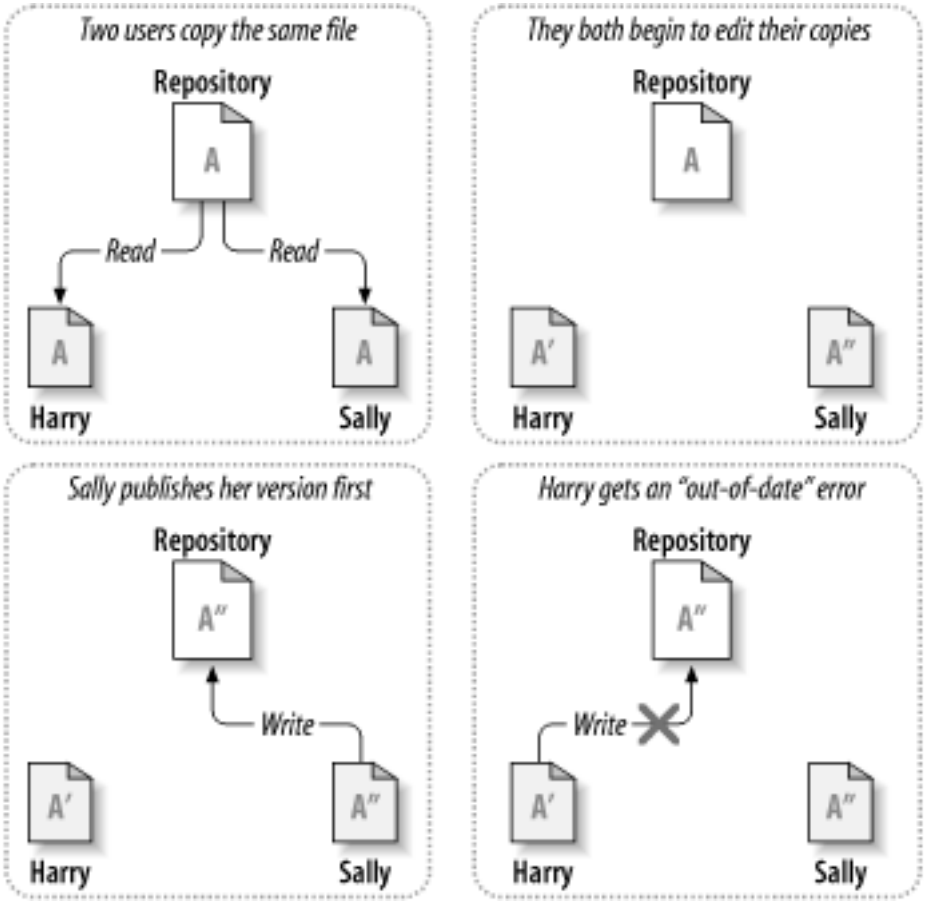


図2.4 コピー・変更・マージ法

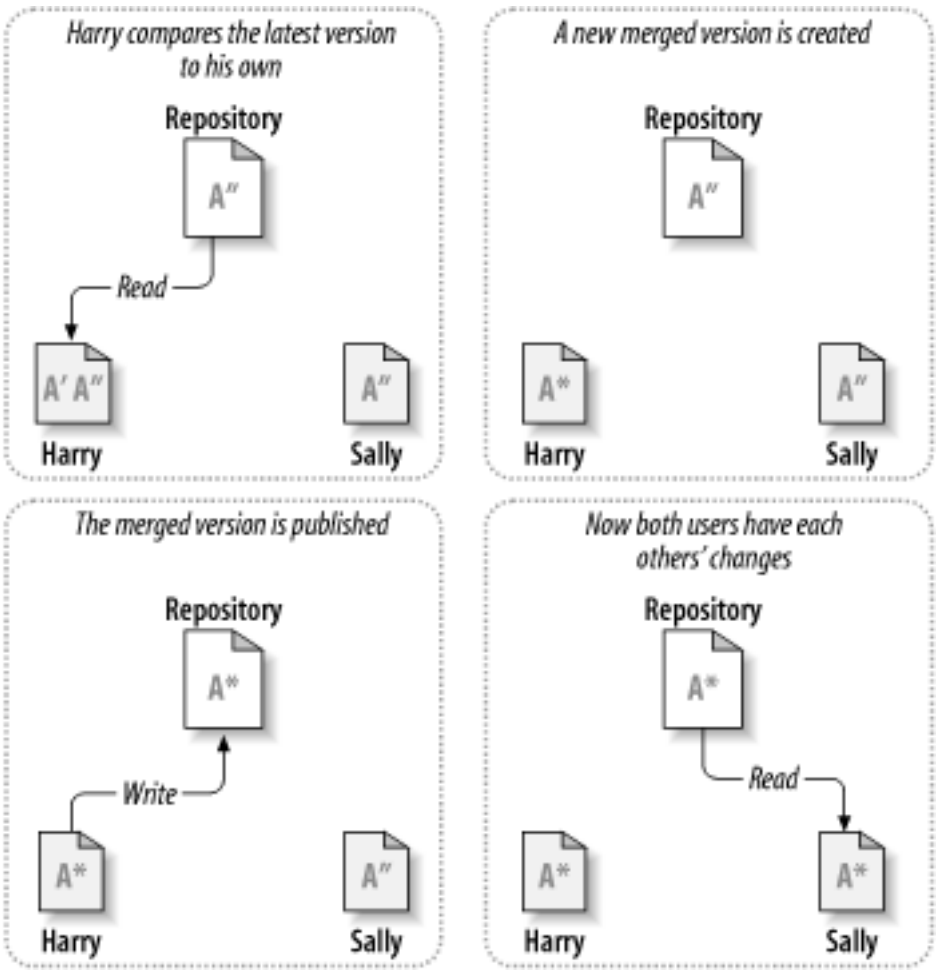


図2.5 ...コピー・変更・マージ法 (の続き)

では、Sally の変更が Harry 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。Harry 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 は競合状態としてマークされます。このとき彼は競合した変更を両方とも見ることができ、どちらを選ぶか選択できます。ソフトウェアが自動的に競合を解消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。Harry がいったん重なった変更を解消したら、(Sally 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混沌としているように見えますが、実際にはとてもスムーズに事が進みます。ユーザは他の人を待つこともなく、平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判ると思います。そして、競合を解消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。ユーザ間のコミュニケーションです。ユーザ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なにものでもないように見えます。

ロック・変更・アンロックモデルの方がよいといわれる一般的な状況がひとつあります。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 人の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。Harry と Sally、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 は何を行うか?

デフォルトで Subversion はコピー・変更・マージ法を使用しますし、ほとんどの場合この方法が常に必要だと思います。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する場合に、Subversion はそういったお好みの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 クライアントで作業コピーの作成・使用について見ていきましょう。

Subversion の作業コピーは、手元の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。作業コピーはあなたの個人的な作業エリアです。Subversion が他の人の変更をここに組み込むことはありませんし、同様に、明示されるまであなたの変更が他の人から使用できるようにな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、Subversion はその変更を同じプロジェクトであなたと一緒に 作業しているほかの人に 公開 するためのコマンドを (リポジトリに書き込むことで) 用意します。もし他の人が自分自身の変更を公開したとき にはSubversionはその変更を自分の作業コピーにマージするコマンドを (リポジトリの内容を読み出して) 用意します。

作業コピーには、Subversion によって作成・保守される特別なファイルがあり、その助けを受けてコマンドを実行します。特に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あり、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があります。この配置では、ユーザの作業コピーは大抵リポジトリの特殊な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。

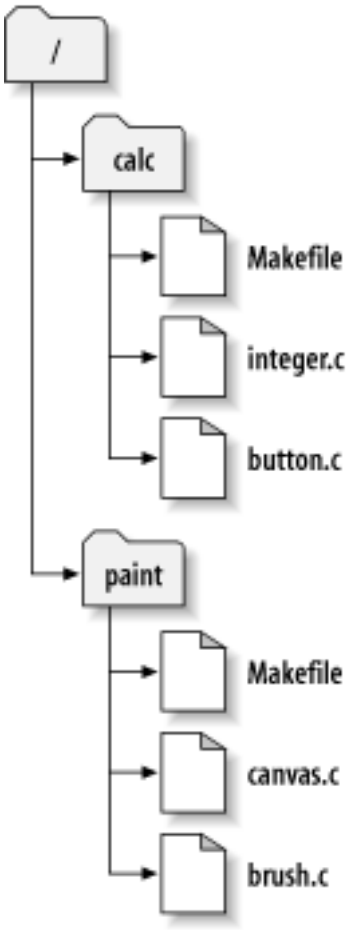


図2.6 リポジトリのファイルシステム

言い換えると、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のに、チェックアウト をリポジトリのサブツリーに対して行わなければなりません。チェックアウト という用語はロックやリソース確保のように聞こえますがそうではなく、単にあなた専用のコピーを作成することです。

button.c への変更を行うことを思い浮かべてください。.svn ディレクトリはファイルの変更日付と元の内容を記憶していますから、Subversion はファイルが変更されていることを教えてくれます。しかし、あなたの行った変更を (あなたが明示するまで) 公開にしません。あなたが変更を公開することは、リポジトリへのコミット (やチェックイン) と呼ばれています。

あなたの変更を他の人に公開するには、Subversion の commit コマンドを使用してください。

さて、button.c への変更を、リポジトリにコミットしましたから、他のユーザが /calc の作業コピーをチェックアウトすると、あなたが行った変更が入っている最新版のファイルを見ることになります。

あなたに Sally という相棒がいて、/calc の作業コピーをあなたと同時にチェックアウトしたとします。あなたが button.c の変更をコミットしても、Sally の作業コピーは変更されないままです。Subversion はユーザのリクエストによってのみ作業コピーを更新するからです。

Sally のプロジェクトを最新にするには、Subversion に作業コピーを更新するよう指示します。これは Subversion の update コマンドを使用します。これによりあなたの変更が Sally 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。

Sally がどのファイルを更新するか指定する必要がないことに注意してください。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用い、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 リポジトリはたくさんの異なる方法 (ローカルディスクや様々なネットワークプロトコル) でアクセスできます。しかし、リポジトリの位置は常に URL で表されます。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 リポジトリアクセス URL

ほとんどの場合、Subversion の URL には標準文法を使用し、URL の中にサーバ名やポート番号を含められます。file:// アクセスメソッドは通常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 といった形式になります。ローカルマシンでは URL の hostname の部分を空にするか localhost としてください。このためローカルパスは、file:///path/to/repos のようにスラッシュが 3 つ並ぶことになります。

また Windows プラットフォームで file:// スキームを使うユーザは、同じマシン上にあるが、クライアント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、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うまくアクセスできます。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
...
```

URL は通常スラッシュを使用しますが、Windows のパス形式 (URL ではない) はバックスラッシュを用います。

ネットワーク共有上でも 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アクセスできません。

警告

ネットワーク共有上で Berkeley DB リポジトリを、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。ドライブレターにマッピングしたネットワークドライブでもダメです。ネットワーク共有上で Berkeley DB を使おうとした場合、(すぐに不可思議なエラーに遭遇するか、数ヶ月後にリポジトリデータベースが破損するか) 結果が予想できません。

2.3.3. リビジョン

svn commit 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。作業コピーでは、ファイルの内容を変更したり、ファイルやディレクトリの作成・削除・名前変更・コピーを、完全な変更セットの単位としてコミットできます。

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、全てをそこに置くか何も置かないかのどちらかになります。Subversion はプログラムのクラッシュや、システムクラッシュ、ネットワーク障害、他のユーザ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、前のリビジョンよりも大きくなります。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うまい方法は、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。

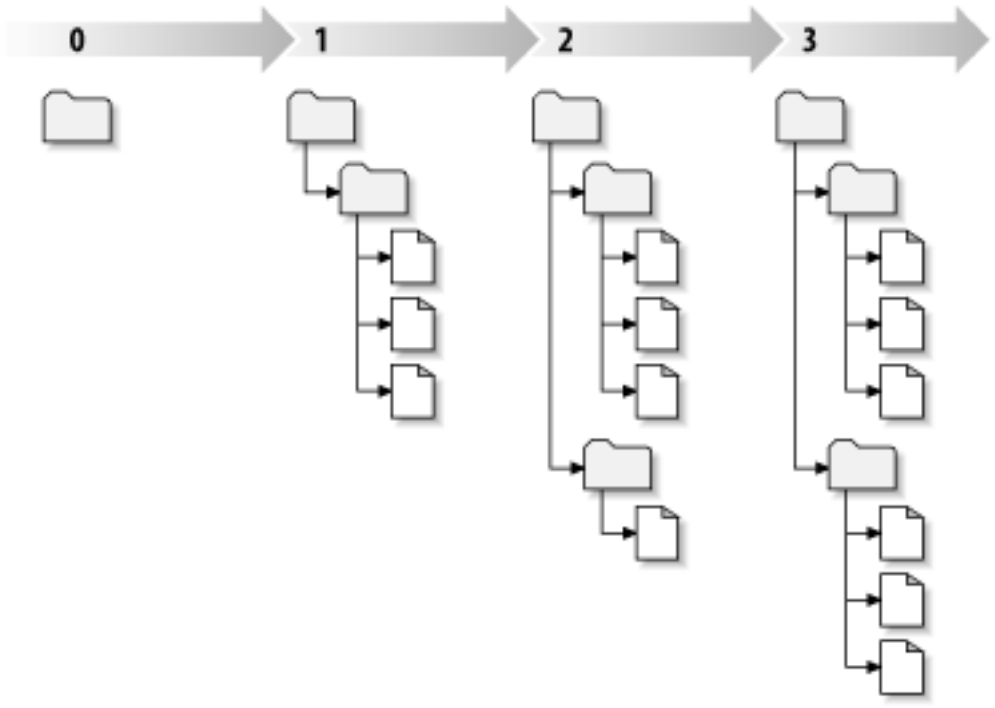


図2.7 リポジトリ

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違い、Subversion　のリビジョン番号はツリー全体に対してつけられるもので、個々のファイルにつけ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指定し、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示すものです。これについて考える別の方法は、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あらわしていると考えることです。Subversion ユーザが、「`foo.c` のリビジョン 5」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。一般的に、あるファイルのリビジョン N と M は異なっている必要はありません。

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十分注意してください。複数の異なるリビジョンが混在する可能性があります。たとえば、最新リビジョンが 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、Sally が integer.c を変更し、リビジョン 6 を作成したとします。svn update であなた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
```

Sally の integer.c への変更は、あなたの作業コピーに現れますが、button.c へ行ったあなたの変更はそのままです。この例では、Makefile のテキストはリビジョン 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バー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はどのリビジョンを基にしている (これは 作業リビジョン と呼ばれる) か。また、
- 手元の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。

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。

変更なし、かつ最新
: 作業ディレクトリのファイルは更新されておらず、作業リビジョン以降に起きたリポジトリへのコミットでもそのファイルに変更がない状態。そのファイルへの コミット は何も行わないし、更新 も何も行いません。

手元で更新、かつ最新
: 作業ディレクトリが更新されており、その基になったリビジョン以降のリポジトリへのコミットではそのファイルに変更がない状態。手元にはコミットしていない変更があるので、そのファイルへの コミット は成功します。また、更新 ではそのファイルへは何も行いません。

変更なし、かつ最新ではない
: 作業ディレクトリのファイルに変更はないが、リポジトリには変更がある状態。このファイルを現在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 はなにもしませんが、更新 は作業コピーへ最新の変更を格納します。

手元で更新、かつ最新ではない
: ファイルは作業コピーでも、リポジトリでも変更されています。 ファイルに対する commit はout-of-date エラーになります。まず、そのファイルを更新しなくてはなりません。ファイルに対する update は公開されている変更点 を作業コピーの変更にマージしようとします。これが自動的にできないような状況の場合、ユーザが競合の解消を行うため、Subversion はそのままにしておきます。

2.4. まとめ

この章では Subversion の基本的なコンセプトの数々をカバーしました。

- 中心となるリポジトリ、クライアントの作業コピー、リポジトリリビジョンツリーの並び、といった概念を導入しました。
- 2 人の共同作業者が Subversion を用いて、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。これには「コピー・変更・マージ」モデルを使用します。
- 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

リポジトリへアクセスするのにどのプロトコルを使用しても、最低　　1　　つはリポジトリを作成する必要があります。Subversion 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

リポジトリを作成するのに　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、ネットワーク共有や　Windows98　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るため、その主張も弱くなってきています。詳細は Subversion Book の Choosing a Data Store [http://svnbook.red-bean.com/en/1.5/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でのリポジトリ作成

1. リポジトリの root フォルダとして SVN という空のフォルダ (例: D:¥SVN¥) を作成してください。
2. D:¥SVN¥ の中に MyNewRepository というフォルダ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、以下を入力してください。

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

もしくは

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

さて、D:¥SVN¥MyNewRepository にある新しいリポジトリを作成しました。

3.1.2. TortoiseSVN でのリポジトリ作成

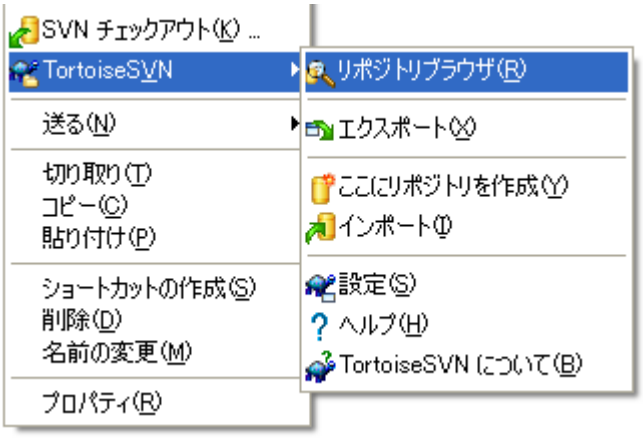


図3.1 バージョン管理外フォルダの TortoiseSVN メニュー

1. Windows エクスプローラを開く

2. 新しいフォルダを作成し名前 (例: SVNRepository) をつける

3. 新しく作成したフォルダ上で 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、フォルダが空か書込が禁止されていないかを確認してください。

ヒント

TortoiseSVN は、もう BDB リポジトリを作成するオプションを提供しません。しかし、依然として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。一般的に FSFS リポジトリの方が保守が容易ですし、BDB のバージョン間の差異からくる互換性問題に対して、私たちが TortoiseSVN を保守するのが容易になるのです。

TortoiseSVN の将来のバージョンでは、BDB リポジトリに対する file:// アクセスは、互換性問題によりサポートされません。もちろん、サーバ経由でアクセスする svn:// プロトコル、http:// プロトコル、https:// プロトコルは常にサポートされます。このため、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 で作成するのを強くお勧めします。

また、もちろん、ローカルのテスト用途は別として、file:// アクセスを全く使用しないのをお勧めします。サーバを用いるのは、開発者が一人で使用するのを除くすべてにおいて、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、そのフォルダへのパスが必要です。Subversion 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。一貫してスラッシュ ("/") を使用することに注意してください。

ネットワーク共有にあるリポジトリにアクセスするには、ドライブへのマッピングと 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

ネットワーク共有上で Berkeley DB リポジトリを、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。ドライブレターにマッピングしたネットワークドライブでもダメです。ネットワーク共有上で Berkeley DB を使おうとした場合、(すぐに不可思議なエラーに遭遇するか、数ヶ月後にリポジトリデータベースが破損するか) 結果が予想できません。

3.1.4. ネットワーク共有にある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共有上に配置でき、file:// プロトコルを用いて複数のユーザがアクセスできますが、これは絶対にお勧めできません。実際、私たちは思いとどまらせようと強く思いますし、そのような使用をサポートしません。

第一に、すべてのユーザ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り、誰かが事故で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できます。

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 が求めるロック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る可能性があります。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、ファイルのパーミッションを、まったく同じように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 では特に難しいです。

file:// アクセスは、ローカルでの一ユーザ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ければ、それはまさに適切なサーバをセットアップする必要があり、あなたが思うほど難しくはありません。サーバの選択・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。お勧めレイアウトを採用するなら、あとで楽ができるでしょう。

リポジトリの編成にはある程度、標準化された、おすすめの方法が あります。ほとんどの人々は trunkディレクトリに開発の「主系」、ブランチコピーがある branches ディレクトリ、そしてタグコピーがある tags ディレクトリを作成します。リポジトリがただ一つのプロジェクトを含む場合には、以下のように、この三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
```

リポジトリに複数のプロジェクトがあるなら、以下のようにブランチ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
```

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インデックス化するのは意味があると思います。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ブランチでインデックス化する方がいいでしょう。この方法だと、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見て取れます。

トップレベルに /trunk /tags /branches アプローチを採用する場合、別にあらゆるブランチやタグについて全体のトランク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、ある点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選択もあります。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、以上の共通のレイアウトを無視することだってできます。あなた方によく作業できるようにならどのような類の変更でも起こせます。ここで選択したものが、永遠に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、ただ単にサーバ側での移動が問題になります。リポジトリを再構成する方法をとりたくなければ、ディレクトリを移動してください。

そのため、まだ基本的なフォルダ構造を、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を使用して、3つのフォルダを作成できます (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を作成してください。
2. このフォルダの中にお好みのトップレベルフォルダ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. このフォルダの上で 右クリック し、TortoiseSVN → インポート... を選択してリポジトリ内のこの構造にインポートしてください。リポジトリルートの中に基本リポジトリレイアウトを作成するため、これで一時フォルダをインポートできます。

リポジトリには、中身だけでインポートしたフォルダ名は現れないことに注意してください。たとえば、以下のようなフォルダ構造を作成してください。

```
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、以下のようになります。

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

3.2. リポジトリのバックアップ

使用するリポジトリがどのタイプでも、通常のバックアップをメンテナンスし、そのバックアップを確認するのはリポジトリを維持するのに非常に重要です。サーバが破損しても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、全くないことを保証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、ステータスを取得するといった、読み込みしか必要でない操作の時でさえ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 (ウェブブラウザを開いたままにする、WebSVN、等々)、バックアップの意味がなくなります。

お勧めの方法は、リポジトリのコピーを安全に作成するのに以下を実行することです。

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

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、ディスクスペースを節約します。

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

3.3. サーバ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属性の変更など) 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、イベントの引き金を引いた人物のユーザ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/1.5/svn.reposadmin.create.html#svn.reposadmin.create.hooks] の章を参照してください。

このフックスクリプトは、リポジトリのあるサーバで実行されます。TortoiseSVN は、各イベントでのローカルに実行する、クライアントサイド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。

シンプルなフックスクリプトは、リポジトリの hooks ディレクトリにあります。このサンプルスクリプトは Unix/Linux サーバ用で Windows ベースのサーバでは修正する必要があります。フックはバッチファイルや実行形式です。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 Only allow log messages to be changed.
if "%4" == "svn:log" exit 0
echo Property '%4' cannot be changed >&2
exit 1
```

標準出力に送った物はすべて捨てられてしまうことに注意してください。コミットリジェクトダイアログにメッセージを表示したい場合は、標準エラー出力に送らなくてはなりません。このバッチファイルでは >&2 を使用しています。

3.4. チェックアウトリンク

Subversion リポジトリを他の人に利用できるようにするにあたり、自分のウェブサイトからリンクを張ることになるかもしれません。他の TortoiseSVN のユーザにチェックアウトリンクを提供するというアクセスしやすい方法があります。

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 のユーザがそういった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、以下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、ぴったりの画像をつけてもいいでしょう。TortoiseSVN ロゴ [http://tortoisesvn.tigris.org/images/TortoiseCheckout.png] を使用できますし、自前の画像も使用できます。

```
<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。以下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 (や、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 つのサーバプロトコルともに暗号化することもできます。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章日常操作ガイド に進んでください。

サーバを持っておらず、一人で作業している場合や、Subversion や TortoiseSVN を隔離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 のはじ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。この章の残りをスキップし、使い始める方法を探しに 4章日常操作ガイド へ直接行ってもかまいません。

ネットワーク共有上に、マルチユーザリポジトリをセットアップしようと考えているなら、考え直してください。なぜそれがまずいアイディアだと考えているかは、「ネットワーク共有にあるリポジトリへのアクセス」をお読みください。サーバのセットアップは印象ほど難しくありませんし、信頼性の向上と、もしかしたら高速化をもたらすことでしょう。

以下の節は、そのようなサーバを Windows マシンにセットアップする方法の段階的ガイドです。もちろん、Linux マシンにもサーバをセットアップできますが、このガイドでは扱いません。Subversion サーバのオプションのより詳細な情報や、あなたにピッタリなアーキテクチャの選択する方法は、Subversion book の Server Configuration [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.html] にあります。

3.6. svnserve ベースのサーバ

3.6.1. はじめに

Subversion には、Svnserve (通常の TCP/IP 接続上でカスタムプロトコルを使用する軽量スタンドアロンサーバ) があります。小規模構成にしたい場合や、本格的な Apache サーバを使用できない場合には申し分ありませんし、

ほとんどの場合、svnserve は Apache ベースサーバよりセットアップが簡単で、早く実行できますが、高度な機能のいくつかはありません。また今では、さらに安全にしやすくする SASL サポートも含まれています。

3.6.2. svnserve のインストール

1. Subversion の最新版は、http://subversion.tigris.org/getting.html [http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91] から取得してください。またはパッケージ化済みインストーラを、CollabNet の http://www.collab.net/downloads/subversion から取得してください。このインストーラは、svnserve を Windows サービスとしてセットアップし、セキュリティのために SASL を使う場合に必要となるいくつかのツールも含んでいます。

2. 既に Subversion をインストールしてあり、svnserve が稼働している場合、継続する前に停止する必要があります。

3. Subversion インストーラを実行します。サーバ上で実行しているなら (お勧め)、ステップ 4 までスキップできます。

4. Windows エクスプローラを開き、Subversion のインストールディレクトリ (通常 C:¥Program Files¥Subversion) にある bin ディレクトリに行ってください。svnserve.exe, intl3_svn.dll, libapr.dll, libapriconv.dll, libapriutil.dll, libdb*.dll, libeay32.dll, ssleay32.dll があるのでこのファイルか、bin ディレクトリのすべてのファイルをサーバ用のディレクトリ (例: c:¥svnserve) にコピーしてください。

3.6.3. svnserve の実行

svnserve をインストールしたら、サーバを実行する必要があります。起動する単純な手順は、DOS のシェルから以下のようにするか、Windows のショートカットを作成することです。

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

svnserve が起動し、ポート 3690 でリクエストを待ち受けます。--daemon スイッチは、svnserve に daemon プロセスとして実行するように指示します。そのためすでに実行されている場合は、手動で終了する必要があります。

まだリポジトリを作成していないなら、Apache サーバのセットアップ「設定」の説明を参照してください。

svnserve が作動しているかテストするため、TortoiseSVN → リポジトリブラウザ でリポジトリを参照してください。

リポジトリが c:¥repos¥TestRepo にあり、サーバ名が localhost とすると、リポジトリブラウザに以下のように入力します。

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

セキュリティを高め、URL を入力する時間を節約するために、--root スイッチを用いてサーバ上の指定したディレクトリにルートを設定し、そこだけアクセスするようにできます。

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

ガイドのように以前行ったテストの物を使用して、svnserve を以下のように実行します。

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

すると TortoiseSVN のリポジトリブラウザでは、次のように省略できるようになります。

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

--root スイッチは、あなたのサーバの svnserve の場所と違うパーティションやドライブにリポジトリがある場合にも、指定する必要があることに注意してください。

svnserve は複数のリポジトリを提供します。そのリポジトリは、定義したルートフォルダ以下に配置され、ルートからの相対パスでアクセスできます。

警告

ネットワーク共有上で Berkeley DB リポジトリを、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。ドライブレターにマッピングしたネットワークドライブ

でもダメです。ネットワーク共有上で Berkeley DB を使おうとした場合、(すぐに不可思議なエラーに遭遇するか、数ヶ月後にリポジトリデータベースが破損するか) 結果が予想できません。

3.6.3.1. svnserve をサービスとして実行

ユーザが　svnserve　を実行するのは、通常、最善の方法というわけではありません。サーバに常にユーザがログインすることになり、リブート後には、svnserve　を再起動するのを忘れないようにしなければならないのです。よりよい方法は、svnserve を Windows サービスとして実行することです。Subversion 1.4 で開始すると、svnserve は ネイティブな Windows サービスとしてインストールできます。

svnserve をネイティブ Windows サービスとしてインストールするには、以下のコマンドを 1 行で記述し、Windows の起動時に自動起動するサービスを作成してください。

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

パスにスペースを含んでいる場合は、以下のように、(エスケープした) クォートでパスを囲んでください。

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

サービス作成後に、説明の追加もできます。これは Windows のサービスマネージャに表示されます。

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

sc で使用するコマンドラインフォーマットは、かなり見慣れないものであることに注意してください。キー= 値 の組では、キーと = の間にはスペースがあってはなりませんが、値との間にはスペースがなくてはなりません。

ヒント

現在　Microsoft　はサービスを、ローカルサービスアカウントかネットワークサービスアカウントのどちらかで動作させるのを推奨しています。サービスおよびサービス アカウントのセキュリティ計画ガイド [http://www.microsoft.com/technet/security/topics/serversecurity/serviceaccount/default.mspx]　をご覧ください。ローカルサービスアカウントで動作するサービスを作成するには、上記の例に以下を加えてください。

```
obj= "NT AUTHORITY¥LocalService"
```

ローカルサービスアカウントには、Subversion とそのリポジトリに適切な権限を与え、またフックスクリプトで使用するアプリケーションにも同様に権限を与える必要があることに注意してください。ビルトイングループは、"LOCAL SERVICE" と呼ばれています。

一度サービスをインストールしたら、起動するのにサービスマネージャを使用する必要があります(サーバのリブート時に自動起動するならこの時のみ)。

より詳細な情報は、Windows Service Support for Svnserve [http://svn.collab.net/repos/svn/trunk/notes/windows-service.txt] をご覧ください。

svnserve の以前のバージョンを SVNService ラッパーを使ってインストールしており、今度はネイティブサポートするものに使用したい場合、そのラッパのサービス登録解除をする必要があります (まずサービスの停止を忘れないでください!)。サービスのレジストリエントリを削除するには、単純に

```
svnservice -remove
```

というコマンドを使用してください。

3.6.4. svnserve での Basic 認証

デフォルトの svnserve セットアップでは読取専用の匿名アクセスを提供しています。これは svn:// URL を、チェックアウト・更新に使用でき、リポジトリを見るのに TortoiseSVN のリポジトリブラウザを利用できるということです。しかしいずれの変更もコミットできません。

リポジトリへ書込アクセスを行えるようにするには、リポジトリディレクトリ内にある conf/svnserve.conf ファイルを編集する必要があります。このファイルは svnserve daemon の設定を制御します。また有用なドキュメントも含んでいます。

匿名書込アクセスを許可するシンプルな設定は、

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

です。しかし、svn:author 属性が空のため、誰がリポジトリに変更を加えたのかがわかりません。誰が変更したのかコントロールできなくなってしまいます。いくらか危険な設定です!

これを解決するひとつの解答がパスワードデータベースを作成することです。

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

userfile は svnserve.conf と同じディレクトリに存在するファイルです。このファイルは、ファイルシステムのどこか他の場所に置くことができ (同じアクセス権が必要な複数のリポジトリがある場合に便利)、絶対パスか conf ディレクトリに対する相対パスで指定できます。パスを含める場合 /the/unix/way のように書かねばなりません。¥ やドライブレターは動作しません。userfile の構造は

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

のようになります。この例は未認証 (匿名) ユーザの全アクセスを拒否し、userfile に並べたユーザに読み書きアクセスを与えます。

ヒント

同一のパスワードデータベースで複数のリポジトリを保守する場合、認証レルムを使用すると (TortoiseSVN が資格証明をキャッシュできるので 1 度入力すればよいため) 楽になります。詳細な情報は Subversion book の、特に Create a 'users' file and realm [http://svnbook.red-bean.com/en/1.5/

svn.serverconfig.svnserve.html#svn.serverconfig.svnserve.auth.users] 節や、Client Credentials Caching [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.netmodel.html#svn.serverconfig.netmodel.credcache] 節にあります

3.6.5. SASL によるよりよいセキュリティ

3.6.5.1. SASL とは?

Cyrus Simple Authentication and Security Layer は、カーネギーメロン大学で書かれたオープンソースソフトウェアです。これは、いくつものネットワークプロトコル (と Subversion 1.5 以降) に汎用認証機構と暗号化機能を追加し、svnserve サーバと TortoiseSVN クライアントは、どちらもこのライブラリを利用できます。

有効なオプションのより完全な議論のために、Subversion book の Using svnserve with SASL [http://svnbook.red-bean.com/en/1.5/svn.serverconfig.svnserve.html#svn.serverconfig.svnserve.sasl] 節を見るべきです。Windows サーバで安全な認証や暗号化をセットアップする、簡単な方法を探しているだけであれば、巨悪であるインターネット経由で、リポジトリへ安全にアクセスするために読み進めてください。

3.6.5.2. SASL 認証

サーバ上で特定の SASL 機構を有効にするには、3 つのことを行う必要があります。ひとつ目は、リポジトリの `svnserve.conf` ファイルに `[sasl]` セクションを作成し、以下のキーと値の組を設定することです。

```
use-sasl = true
```

ふたつ目は、`svn.conf` ファイルを好みの場所 (たいてい subversion をインストールしたディレクトリ) に作成することです。

みっつ目は、SASL が見つけられるように、ふたつの新しいレジストリエントリを作成することです。`[HKEY_LOCAL_MACHINE¥SOFTWARE¥Carnegie Mellon¥Project Cyrus¥SASL Library]` というレジストリキーを作成し、その中に以下のふたつの string 値を配置してください。`SearchPath` に `sasl*.dll` プラグインを含むディレクトリパス (通常 Subversion をインストールしたディレクトリ) と、`ConfFile` に `svn.conf` ファイルを含むディレクトリです。CollabNet のインストーラを使用している場合、そのレジストリキーはすでに作成されています。

以下を含む `svn.conf` ファイルを編集してください。

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

最後の行には、`sasldb` と呼ばれる認証データベースの場所を指定します。どこでもかまいませんが、あつらえ向きなのはリポジトリの親ディレクトリです。svnserve サービスが、このファイルを読めることを確認してください。

svnserve がすでに動作している場合、更新した設定を読むのを保証するため、svnserve を再起動する必要があります。

さて、すべてセットアップが終わったら、あとは、ユーザを作成しパスワードを設定するだけです。これには `saslpasswd2` プログラムを使用する必要があります。CollabNet のインストーラを使用している場合、インストールディレクトリにあるはずです。以下のように使用してください。

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

-f スイッチでデータベースの場所を与えてください。realm はリポジトリの svnserve.conf ファイルに定義した値と同じでなければなりません。また、username は接続するのに使用するものと一致している必要があります。realm には、空白文字を使用できないことに注意してください。

sasldblistusers2 プログラムを使用して、データベースに格納されているユーザ名を表示できます。

3.6.5.3. SASL 暗号化

異なる暗号化レベルを有効・無効にするには、リポジトリの svnserve.conf ファイルにあるふたつの値を設定します。

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

min-encryption 変数と max-encryption 変数は、サーバで必要な暗号化レベルを制御します。暗号化を完全に無効にする場合、どちらの値も 0 としてください。データの単純なチェックサム比較 (つまり暗号化を利用せず、改ざんを防ぎデータの完全性を保証する) を有効にするには、どちらの値も 1 としてください。暗号化を許可する (しかし必須ではない) 場合、最小値を 0、最大値を数ビット長としてください。暗号化を無条件に必要とするには、どちらの値にも 1 以上を設定してください。先ほどの例では、クライアントには少なくとも 128 ビット暗号化が必要で、256 ビット暗号化以上は必要ないことを示しています。

3.6.6. svn+ssh での認証

svnserve ベースサーバでユーザ認証を行うもう一つの方法は、リクエストを通過させるトンネルに secure shell (SSH) を利用することです。SASL のセットアップようには単純ではありませんが、様々なケースで便利だと思います。

この方法では、svnserve は daemon プロセスで実行させず、secure shell が SSH で認証したユーザに対して svnserve を起動します。これを有効にするには、secure shell daemon がサーバに必要です。

サーバをセットアップする基本的な方法は、付録G SSH を用いた安全な svnserve で得られます。その他のFAQ にある SSH の話題については、「SSH」で検索してください。

svnserve のさらなる情報は Version Control with Subversion [http://svnbook.red-bean.com] にあります。

3.6.7. svnserve でのパスベース認証

Subversion 1.3 で起動すると、svnserve は、Apache サーバで有効な mod_authz_svn パスベース認証方式と同じものをサポートします。リポジトリディレクトリにある conf/svnserve.conf ファイルを編集し、認証ファイルを参照する行を追加する必要があります。

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

authz には、作成したアクセス制限を定義したファイルを指定してください。リポジトリごとに独立したファイルを使用できますし、複数のリポジトリで同じファイルを使用することもできます。ファイル形式の説明については、「パスベース認証」をご覧ください。

3.7. Apache ベースのサーバ

3.7.1. はじめに

Subversion 用にセットアップできるサーバのうち、もっとも柔軟なのは Apache ベースの物です。セットアップするには少し複雑ですが、他のサーバが提供できない特徴があります。

WebDAV
: Apache ベースの Subversion サーバは、多数の他のプログラムがサポートする WebDAV プロトコルを使用します。例えば、リポジトリを Windows エクスプローラで言う「Web フォルダ」としてマウントでき、ファイルシステムのその他のフォルダのようにアクセスできます。

リポジトリの閲覧
: ブラウザにリポジトリの URL を入力することで、Subversion クライアントをインストールしていなくても内容を閲覧できます。これにより、より大きなユーザの輪がそのデータにアクセスできるようになります。

認証
: Apache がサポートする (SSPI や LDAPを含む) 認証メカニズムを使用できます。

セキュリティ
: Apache はとても安定・安全ですから、リポジトリにも自動的に同じセキュリティが確保できます。これには SSL の暗号化も含まれます。

3.7.2. Apache のインストール

Apache をインストールするためには、コンピュータが Windows 2000 か Windows XP+SP1, Windows 2003, Vista, Server 2008 である必要があります。

警告

サービスパック 1 を適用していない Windows XP ではインチキなネットワークデータを流してしまい、リポジトリを破壊する可能性があります!

1. Apache ウェブサーバの最新版を http://httpd.apache.org/download.cgi からダウンロードしてください。バージョン 2.2.x をダウンロードしてください。1.3.xx では動作しません!

   Apache 用の msi インストーラは other files をクリックして、binaries/win32 へと進むと見つかります。apache-2.2.x-win32-x86-openssl-0.9.x.msi (openssl を含むもの) を選択することになるかと思います。

2. いったん Apache2 インストーラを手に入れれば、それをダブルクリックし、インストール中にそのプロセスの案内をしてもらえます。(サーバ用の DNS 名がなく、IP アドレス入力するだけなら) サーバ URL を必ず正しく入力してください。すべてのユーザ向けに、ポート 80 でサービスとして apache をインストールするのをおすすめします。注) すでに IIS や 80 番ポートを他のプログラムを実行しているなら、インストールに失敗するでしょう。その場合、プログラムディレクトリ ¥Apache Group¥Apache2¥conf に移動し、 httpd.conf をそこに置いてください。そのファイルを編集し、Listen 80 を別の空いているポート (例えば Listen 81) に変更してください。その後インストールを再起動してください。今度は問題なく終了するはずです。

3. Apache ウェブサーバの動作確認のため、ウェブブラウザで http://localhost/ にアクセスすると、あらかじめ設定されているウェブサイトが表示されます。

注意

Apache をサービスとしてインストールすると決めた場合、デフォルトではローカルシステムアカウントで実行してしまうという警告があります。より安全な方法として、Apache を実行するにあたって独立したアカウントを作成できます。

リポジトリのあるディレクトリのアクセスコントロールリスト (ディレクトリを右クリック | 属性 | セキュリティ) で、Apache を実行するサーバ上のアカウントをフルコントロールに明示していなければなりません。そうでなければ、ユーザが行った変更をコミットすることができません。

Apache がローカルシステムとして実行しても、まだそのようなエントリー(この場合 SYSTEM アカウントになる) が必要です。

このパーミッション設定を Apache にしない場合、ユーザは「Access denied」エラーメッセージを受け取ることになります。このエラーは Apache のエラーログに 500 エラーとして記録されています。

3.7.3. Subversion のインストール

1. Apache 用 Subversion Win32 バイナリの最新版をダウンロードしてください。必ず、使用する Apache のバージョンと組み合わさる、正しいバージョンを使用してください。そうしないと、再起動しようとした際に、不明瞭なエラーメッセージを得ることになります。Apache 2.2.x を使用する場合、http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=8100 へ進んでください。

2. Subversion インストーラを実行し、説明に従ってください。Subversion インストーラがインストール済みの Apache を認識した場合、それでほとんど終了です。Apache サーバが見つからない場合、いくつか追加の手順があります。

3. Windows エクスプローラで、Subversion のインストールディレクトリ (通常 `c:¥program files¥Subversion`) に移動し、`/httpd/mod_dav_svn.so` と `mod_authz_svn.so` を確認してください。そのファイルを Apache モジュールディレクトリ (通常 `c:¥program files¥apache group¥apache2¥modules`) にコピーしてください。

4. Subversion インストールディレクトリにある `/bin/libdb*.dll` や `/bin/intl3_svn.dll` を、Apache bin ディレクトリにコピーしてください。

5. Apache の設定ファイル (通常 `C:¥Program Files¥Apache Group¥Apache2¥conf¥httpd.conf`)を、メモ帳のようなテキストエディタで以下のように編集してください。

   以下の行のコメントをはずし ('#' マークを消し) てください。

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

   また、以下の 2 行を `LoadModule` セクションの最後に追加してください。

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

3.7.4. 設定

以上で Apache と Subversion をセットアップしました。しかし Apache は、どのように TortoiseSVN のような Subversion クライアントを扱うかをまだ知りません。Subversion リポジトリが使用する URL を Apache に教えるために、Apache 設定ファイル (通常、`c:¥program files¥apache group¥apache2¥conf¥httpd.conf`) をお好みのテキストエディタ (例: メモ帳) で以下のように編集してください。

1. 以下の行を設定ファイルの最後に追加してください。

```
```

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

これで Apache に、すべての Subversion リポジトリの物理的な位置が D:¥SVN 以下にあるように設定できます。リポジトリは外部に URL: http://MyServer/svn/ で提供することになります。アクセス制限は passwd ファイルに記述した users/password の組で行います。

2. passwd ファイルの作成をするには、コマンドプロンプト (DOS ボックス) を再度開き、apache2 フォルダ (通常 c:¥program files¥apache group¥apache2) に移動してください。その後、以下を入力してファイルを作成してください。

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

これで認証に使用するファイルを passwd という名前で作成します。ユーザを追加する場合は、さらに以下のようにしてください。

```
bin¥htpasswd passwd <username>
```

3. 再度 Apache サービスを再起動してください。

4. ブラウザで http://MyServer/svn/MyNewRepository (MyNewRepository には、あらかじめ作成しておいた Subversion リポジトリの名前に置き換えてください) すべてうまくいったのなら、ユーザ名とパスワードを入力でき、リポジトリの内容を見ることができます。

以下は入力する部分の短い説明です。

設定	説明
<Location /svn>	Subversion リポジトリは URL http://MyServer/svn/ から有効になることを意味します。

表3.1 Apache httpd.conf の設定

これは例でしかありません。Apache ウェブサーバにはもっとずっとたくさん設定項目があります。

- リポジトリを、指定したユーザは書込アクセスができ、それ以外のユーザは読込アクセスしかできないようにしたい場合、以下の行を変更してください。

```
Require valid-user
```

これを以下のようにします。

```
<LimitExcept GET PROPFIND OPTIONS REPORT>
```

```
Require valid-user
</LimitExcept>
```

- passwd ファイルを使用して、全リポジトリを単位にアクセス制限・許可を行います。リポジトリ内部のフォルダごとにアクセスを設定するような制御が必要なら、以下の行のコメントアウトを解除し、Subversion アクセスファイルを作成してくださいてください。

```
#AuthzSVNAccessFile svnaccessfile
```

  Apache は確実に、有効なユーザのみが /svn にアクセスできるようにします。また、Subversion の AuthzSVNAccessFile モジュールにユーザ名を渡し、Subversion アクセスファイルに列挙したルールに基づく、よりきめの細かいアクセス制御を行うことができます。パスが repos:path か、単に path として指定されるのに注意してください。特定のリポジトリを指定しない場合、アクセスルールは SVNParentPath の下のすべてのリポジトリに適用されます。mod_authz_svn で使用する認証ポリシーファイルの書式は、「パスベース認証」で説明しています。

- ウェブブラウザでのリポジトリの閲覧を、もっと「魅力的」にするには、以下の行のコメントを外してください。

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

  また、svnindex.xsl, svnindex.css, menucheckout.ico をドキュメントルートディレクトリ (通常 C:/Program Files/Apache Group/Apache2/htdocs) に置いてください。このディレクトリは、Apache 設定ファイルの DocumentRoot ディレクティブで設定します。

  その 3 ファイルは http://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/contrib/svnindex [http://code.google.com/p/tortoisesvn/source/browse/trunk/contrib/svnindex] にあるソースリポジトリから直接取得できます (TortoiseSVN のソースリポジトリへのアクセスのしかたは、「TortoiseSVN は自由!」で説明します)。

  TortoiseSVN リポジトリにある XSL ファイルには、以下のような気の利いた仕掛けがしてあります。ウェブブラウザでのリポジトリ閲覧の際、リポジトリにあるフォルダごとに、右側にアイコンを表示します。このアイコンをクリックすると、この URL 用の TortoiseSVN のチェックアウトダイアログを開きます。

3.7.5. 複数のリポジトリ

SVNParentPath ディレクティブを使用する場合、新しいリポジトリを作成する際に Apache 設定ファイルを変更する必要はありません。単にはじめのリポジトリと同じ場所に、新しいリポジトリを作成してください。これで終わりです! 筆者の会社では、SMB (通常の Windows ファイルアクセス) で指定のフォルダに直接アクセスできています。ここに新しいフォルダを作成し、TortoiseSVN コマンド TortoiseSVN → ここにリポジトリを作成... を実行すると新しいプロジェクトのホームができます...

Subversion 1.3 以降を使用している場合、ブラウザで個々のリポジトリを指定するのではなく上位パスを指定して、有効なプロジェクトの一覧を Apache が表示できるよう、SVNListParentPath on ディレクティブを使用できます。

3.7.6. パスベース認証

mod_authz_svn モジュールは、ユーザ名とリポジトリパスを元にしたアクセス制限のきめ細かい制御を行えます。これは Apache サーバで利用でき、また Subversion 1.3 では svnserve でも同様に利用できます。

サンプルファイルは以下のようになります。

```
[groups]
admin = john, kate
devteam1 = john, rachel, sally
```

```
devteam2 = kate, peter, mark
docs = bob, jane, mike
training = zak
# Default access rule for ALL repositories
# Everyone can read, admins can write, Dan German is excluded.
[/]
* = r
@admin = rw
dangerman =
# Allow developers complete access to their project repos
[proj1:/]
@devteam1 = rw
[proj2:/]
@devteam2 = rw
[bigproj:/]
@devteam1 = rw
@devteam2 = rw
trevor = rw
# Give the doc people write access to all the docs folders
[/trunk/doc]
@docs = rw
# Give trainees write access in the training repository only
[TrainingRepos:/]
@training = rw
```

すべてのパスをチェックするのは、(特にリビジョンログに関して)　時間がかかる操作だということに注意してください。サーバは、リビジョンごとにすべてのパスを変更があるかチェックし、そこが読み込めるかどうかをチェックします。リビジョンにあるファイルが多くなれば、時間をどんどん消費してしまいます。

認証 (authentication) と許可 (authorization) は独立したプロセスです。ユーザがリポジトリパスへアクセスしたい場合、通常の認証条件とファイルへのアクセス許可条件を両方満たさなくてはなりません。

3.7.7. Windows ドメインでの認証

お気づきのように passwd ファイルに、ユーザごと別々に ユーザ名/パスワードのエントリを作成しなければなりません。そして、(安全上の理由で) ユーザに定期的にパスワードを変えて欲しいなら、手動で変更しなければなりません。

しかし少なくとも、Windows　のドメインコントローラを利用するLANの内部からリポジトリにアクセスするなら、この問題の解決法があります。mod_auth_sspi です!

オリジナルの SSPI モジュールはソースコード込みで Syneapps から提供されていましたが、現在開発が止まっています。しかし、がっかりすることはありません。コミュニティが拾い上げ、改良を行っています。新しいホームは SourceForge [http://sourceforge.net/projects/mod-auth-sspi/] にあります。

- apache のバージョンに一致するモジュールをダウンロードし、mod_auth_sspi.so を Apache モジュールフォルダにコピーしてください。
- Apache 設定ファイル を編集し、以下を LoadModule セクションに追加してください。

```

LoadModule sspi_auth_module modules/mod_auth_sspi.so
```

以上の行を

```

```

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

の行の前に挿入してください。

- Subversion ロケーションがこのタイプの認証を使用するように、

```

AuthType Basic
```

を

```

AuthType SSPI
```

に変更しなければなりません。また、

```

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

を <Location /svn> ブロックに追加する必要があります。ドメインコントローラを使用しなければ、<domaincontroller> としたドメインコントローラの名前を取り除いてください。

SSPI を使用して認証を行う場合、パスワードファイルを定義するのに AuthUserFile の行はもう必要ありません。Apache はユーザ名とパスワードを Windows ドメインで認証します。DOMAIN¥username を参照するよう svnaccessfile のユーザ一覧を更新する必要があります。

重要

SSPI 認証は、SSL の安全な接続 (https) でのみ有効です。サーバに通常の http 接続しか使えない場合は、動作しません。

SSL を有効にするには、「SSL を使用したサーバの保護」の章をご覧ください。

ヒント

Subversion の AuthzSVNAccessFile ファイルではユーザ名の大文字小文字は区別します。("JUser" と "juser" は異なります)

Microsoft の世界では、Windows ドメインとユーザ名は大文字小文字を区別しません。ですからネットワーク管理者の中には、ユーザアカウントをキャメルケース (例: JUser) で作成するのを好む人がいます。

Windows ドメインとユーザ名をユーザが入力したとおりに Subversionに渡して SSPI 認証を行う場合、この違いは悪影響を与えます。インターネットエクスプローラは Apache にアカウント作成時の形に自動で整形して渡します。

結論としては、AuthzSVNAccessFile にユーザごとに少なくとも 2 エントリ (小文字のエントリと、インターネットエクスプローラが Apache に渡すのと同じ形式のエントリ) が必要です。ま

た、TortoiseSVN でリポジトリにアクセスする際に、小文字で入力するようユーザに指導する必要があるでしょう。

Apache のエラーログとアクセスログは、こういった問題を読み解き Subversion の AuthzSVNAccessFile モジュールに渡すユーザ名の文字列を決めるのを助ける相棒といえます。すべてが動作するには、svnaccessfile 内のユーザ文字列の正しいフォーマット (例: DOMAIN¥user vs. DOMAIN//user) をテストする必要があります。

3.7.8. マルチ認証ソース

また、Subversion リポジトリに複数の認証ソースを持つこともできます。それには、いずれの認証タイプも権威を持たないようにします。すると Apache は、ユーザ名とパスワードの検証に、複数のソースをチェックするようになります。

Windows ドメイン認証と passwd ファイル両方を使用するというのが、よくあるシナリオです。これにより、Windows ドメインログインできないユーザも、SVN アクセスができるようになります。

- Windows ドメイン認証と passwd ファイル認証の両方を有効にするには、Apache 設定ファイルの <Location> ブロックに以下のエントリを追加してください。

```
AuthBasicAuthoritative Off
SSPIAuthoritative Off
```

以下は、Windows ドメインと passwd ファイル認証を組み合わせた Apache の全設定のサンプルです。

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

  AuthName "Subversion repositories"
  AuthzSVNAccessFile svnaccessfile.txt

# NT Domain Logins.
  AuthType SSPI
  SSPIAuth On
  SSPIAuthoritative Off
  SSPIDomain <domaincontroller>
  SSPIOfferBasic On

# Htpasswd Logins.
  AuthType Basic
  AuthBasicAuthoritative Off
  AuthUserFile passwd

  Require valid-user
</Location>
```

3.7.9. SSL を使用したサーバの保護

Apache 2.2.x では OpenSSL をサポートしていますが、デフォルトでは有効になっていません。手動で有効にする必要があります。

1. apache の設定ファイルで、以下のコメントアウトを外してください。

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

さらにその下の次の部分です。

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

その後、

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

の行を

```
SSLMutex default
```

に変更してください。

2. 次に、SSL 証明書を作成する必要があります。コマンドプロンプト (DOS ボックス) を開き、Apache フォルダ (例: C:¥program files¥apache group¥apache2) に移動して、以下を入力してください。

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

パスフレーズが訊かれます。簡単な単語ではなく、文全体 (詩の一部など) を入力してください。より長い文の方がいいでしょう。サーバの URL も入力する必要があります。質問はすべて任意ですが、すべて答えるのをお勧めします。

通常 privkey.pem ファイルは自動的に作成されますが、作成されなければ以下のコマンドで生成する必要があります。

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

次に以下のコマンドを入力してください。

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

および (1 行で)

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

その後 (1 行で)

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

これで 4000 日経過すると失効する証明書ができます。最後に以下を (1 行で) 入力してください。

```
bin¥openssl x509 -in conf¥server.cert -out conf¥server.der.crt
```

```
                -outform DER
```

ここまでのコマンドは、Apache の conf フォルダにファイルをいくつか (server.der.crt, server.csr, server.key, .rnd, privkey.pem, server.cert) 作成します。

3. Apache サービスを再起動してください。

4. ブラウザで https://servername/svn/project を表示してください…

SSL とインターネットエクスプローラ

SSL を使用してサーバを安全にしようとしたり、Windows ドメインの認証を使用する場合、インターネットエクスプローラでリポジトリを閲覧するのはうまくいきません。が、心配しないでください。インターネットエクスプローラが認証できないだけです。他のブラウザにはこの問題はありませんし、TortoiseSVN やその他の Subversion クライアントは認証できます。

まだ IE でリポジトリの閲覧を行いたいのなら、以下も行ってください。

- Apache 設定ファイルの <Location /path> ディレクティブで区切り SSPIBasicPreferred On と定義してください。IE でも認証できるようになります。しかし、他のブラウザや Subversion はこの location で認証ができなくなります。
- 暗号化していない (SSL を使わない) 認証も提供してください。おかしなことに、SSL で安全にしていない接続では IE は認証の問題が発生しません。
- SSL の「標準」セットアップでは、Apache の仮想 SSL ホストに以下の項目があります。

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

この設定にはきちんとした理由があります (ありました(?))。http://www.modssl.org/docs/2.8/ssl_faq.html#ToC49 をご覧ください。また、NTLM 認証を行いたい場合、keepaliveを使わなければなりません。SetEnvIf 全体をアンコメントすると mod_auth_sspi を組み込んだ Win32 上の Apache で、SSL を使ったWindows 認証を IE で行えるようになります。

SSL アクセスの強制

リポジトリが安全になるよう SSL をセットアップするにあたり、通常の非 SSL (http) アクセスを無効にして、https のみを許可したいかもしれません。このため、Subversion の <Location> ブロックに別のディレクティブ SSLRequireSSL を追加しなければなりません。

<Location> ブロックの例は以下のようになります。

```
<Location /svn>
  DAV svn
  SVNParentPath D:¥SVN
  SSLRequireSSL
  AuthType Basic
```

```
  AuthName "Subversion repositories"
  AuthUserFile passwd
  #AuthzSVNAccessFile svnaccessfile
  Require valid-user
</Location>
```

3.7.10. 仮想 SSL ホストでのクライアント証明書の使用方法

Nigel Green さんが TortoiseSVN のメーリングリストに投稿した内容です。ありがとうございます!

サーバの構成によっては、2 つ仮想 SSL ホストをひとつのサーバにセットアップする必要があるかも知れません。ひとつはクライアント証明書のいらず、web アクセスが公開されています。二つ目はクライアント証明書が必要な安全なサーバで、Subversion サーバを動作させています。

SSLVerifyClient Optional ディレクティブを、Apache 設定ファイルの サーバごとの セクション (つまり VirtualHost ブロックと Directory ブロックの外側) に追加すると、最初の SSL ハンドシェイクでクライアント証明書を要求するように、強制できます。mod_ssl のバグにより、SSL 接続を再ネゴシエーションする場合、動作していないときと同じように、この時点で証明書が必要になるのは避けられません。

解決策は、Subversion でロックしたい仮想ホストのディレクトリに、以下のディレクティブを追加することです。

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

クライアント証明書を受け取り、検証に成功した場合だけ、このディレクティブは、そのディレクトリへのアクセスを許可します。

まとめると、Apache 設定ファイルの関係する行は、以下のようになります。

```
SSLVerifyClient Optional

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

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

### Virtual host configuration for SUBVERSION
### (requiring a client certificate)
<VirtualHost 127.0.0.1:443>
  <Directory "subversion host root path">
    SSLRequire %{SSL_CLIENT_VERIFY} eq "SUCCESS"
  </Directory>

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

第4章 日常操作ガイド

このドキュメントは 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。だいたい何をしたいか分かっているなら、調べる場所として便利ですが、どのように行うかは全く思い出させてくれません。

Subversion でのバージョン管理の解説が必要なら、Version Control with Subversion [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、現在もまだ作成中です。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 (DUG) 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 の作業は空いた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、以下を前提としています。

- すでに TortoiseSVN をインストールしてあることを前提にします。
- バージョン管理システムになじみがあることを前提にします。
- Subversion の基本を知っていることを前提にします。
- サーバのセットアップを行っているかどうかして、Subversion のリポジトリにアクセスできることを前提にします。

4.1. さぁはじめましょう

4.1.1. アイコンオーバーレイ

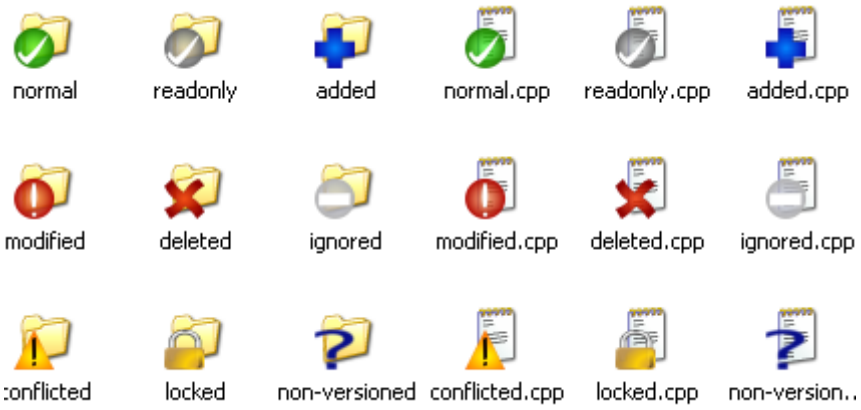


図4.1 エクスプローラのアイコンオーバーレイ表示

TortoiseSVN の目立つ機能に、作業コピーのファイルに現れるアイコンオーバーレイがあります。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。オーバーレイが表現する内容については 「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

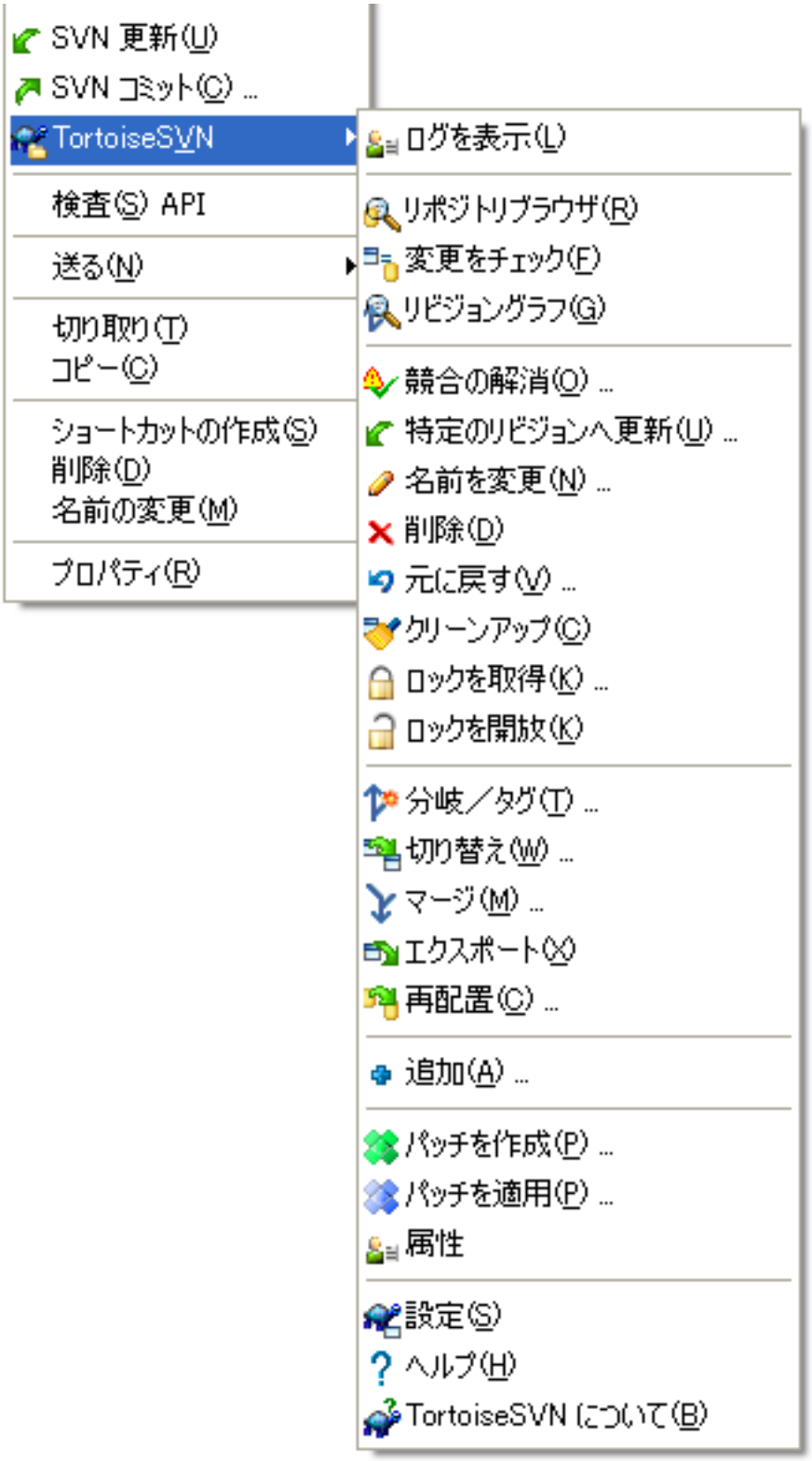


図4.2 バージョン管理下のディレクトリに対するコンテキストメニュー

すべての TortoiseSVN コマンドは、Windows エクスプローラのコンテキストメニューから呼び出されます。ファイルやフォルダを 右クリック すると、ほとんどは直接見えています。ファイルやフォルダであるかどうか、その親フォルダがバージョン管理下にあるかどうかでそのコマンドは有効になります。TortoiseSVN メニューを、エクスプローラのファイルメニューの一部としても見ることができます。

ヒント

ほとんど使用しないようなコマンドは、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には、Shift キーを押したまま、右クリック してください。

いくつかの場合、TortoiseSVN エントリを、複数眼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 バージョン管理フォルダでショートカットする、エクスプローラ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、エクスプローラのファイルメニューに TortoiseSVN のエントリが 3 つ あります。ひとつはフォルダ用、ひとつはショートカットそのもの用、最後にショートカットが指し示すオブジェクト用です。以上を見分けるのに、ファイル、フォルダ、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に関係なく、アイコンの右下にマークがつきます。

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、Microsoft が Vista 用に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

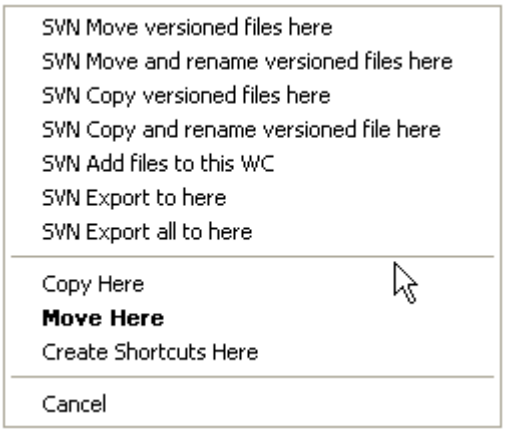


図4.4 バージョン管理下のディレクトリに対する右ドラッグメニュー

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を 右ドラッグ 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 のショートカット出よく知られたものですが、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。行き詰まってしまったら、リフレッシュを兼ねて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 1 キーコマンドかもしれません。例えば……エクスプローラでは、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 email 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……

Ctrl-C
: ……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。

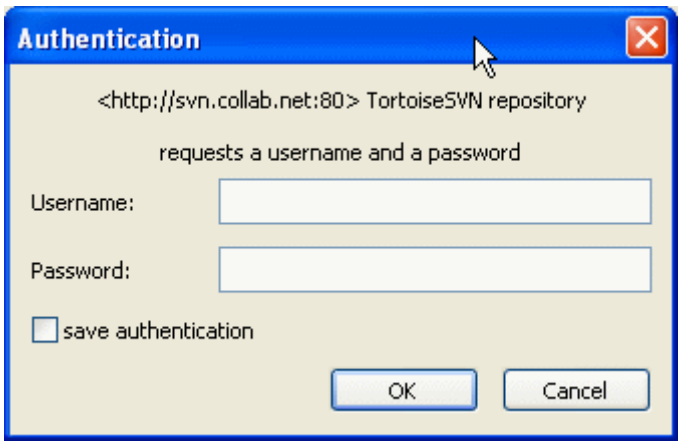


図4.5 認証ダイアログ

ユーザ名とパスワードを入力してください。TortoiseSVN が証明書を Subversion のデフォルトディレクトリ (%APPDATA%¥Subversion¥auth) にある、以下の 3 つのサブディレクトリに格納するかどうかチェックボックスで指定できます。

- svn.simple には BASIC 認証 (ユーザ名/パスワード) の証明書があります。
- svn.ssl.server には SSL サーバ証明書があります。
- svn.username には、(パスワードがいらない) ユーザ名のみの認証用の証明書があります。

全サーバの認証キャッシュをクリアしたければ、TortoiseSVN 設定ダイアログの**保存データ** 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同様に、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クリアします。「保存データの設定」をご覧ください。

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と言う人もいます。そのようにするには、%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します。以下のようになります。

```
@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、windows-help-central.com [http://www.windows-help-central.com/windows-shutdown-script.html] に説明があります。

サーバの認証やアクセスコントロールのセットアップ方法について詳細な情報は、「リポジトリへのアクセス」 を参照してください。

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。しかし画面全体の最大化よりも、縦方向の最大化や、横方向の最大化が便利なことがあります。

4.2. リポジトリへのデータインポート

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、既存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。もっと深いアドバイスが必要であれば、「リポジトリレイアウト」をお読みください。

この節では、ディレクトリ階層をリポジトリへ一度にインポートするよう設計された、Subversion の import コマンドについて説明しています。これは動作しますが、以下のようにいくつか欠点があります。

- 含めるファイルやフォルダを選択する方法が、共通無視設定を除いて存在しません。
- インポートしたフォルダが作業コピーになりません。サーバからファイルをコピーし直すのに、チェックアウトしなければなりません。
- 間違ったフォルダ階層を、リポジトリに簡単にインポートしてしまいます。

このため、import コマンドをまったく使わず、むしろ、「その場でインポート」で説明する、2 段階法に従うのをお勧めします。しかし、あなたは今、基本的な import コマンドがどのように動作するかを説明する場にいます……

リポジトリにプロジェクトをインポートする前に、以下を行ってください。

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 といったコンパイラが生成したファイル、コンパイルしたバイナリ) は削除してください。
2. フォルダやサブフォルダにファイルを構成してください。あとでファイルは削除・移動できますが、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。

今度は Windows エクスプローラのプロジェクトディレクトリ構造のトップレベルフォルダを選択し、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... コマンドを選択し、ダイアログボックスを表示します。

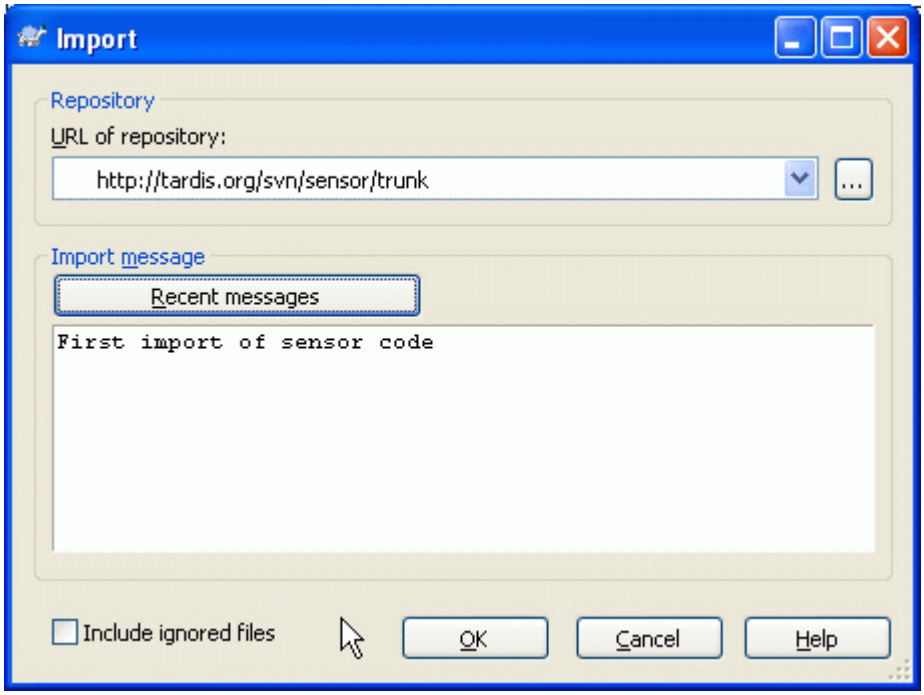


図4.6 インポートダイアログ

このダイアログに、プロジェクトにインポートしたいリポジトリの場所を示す、URL を入力しなければなりません。インポートしようとするローカルフォルダ自体が現れるのではなく、その中身がリポジトリに現れるということに気づくのは重要です。例えば以下の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直接 trunk の中にあるのに驚くかもしれません。URL の一部として、http://mydomain.com/svn/trunk/Widget-X のようにサブディレクトリを指定する必要があります。import コマンドは、リポジトリの中にサブディレクトリがない場合、自動的に作成す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、共通無視パターンにマッチしたファイル・フォルダはインポートされません。この動作を上書きするには、**無視されたファイルを含む** チェックボックスを使用してください。共通無視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。インポートしたフォルダは、バージョン管理下に入らないことに注意してください! 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

既にリポジトリがあると仮定したときに、新しいフォルダ構造を追加する場合、以下の手順にしたがってください。

1. リポジトリブラウザを使用して、リポジトリに直接プロジェクトフォルダを作ってください。
2. インポートしたいフォルダのトップで新しいフォルダをチェックアウトしてください。ローカルフォルダが空でないという警告が出ます。これでバージョン管理下にあるトップレベルフォルダに、バージョン管理外の内容があることになります。
3. バージョン管理下のフォルダ上で TortoiseSVN → 追加... を使い、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、フォルダへの svn:ignore 属性の設定など、必要な変更を加えることができます。
4. トップレベルフォルダ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. 特殊ファイル

時々、バージョン管理下のファイルにユーザごとのデータを含める必要があります。全てのユーザ・開発者が手元の環境をセットアップするときに、修正しなければならないファイルがあるということです。しかしそういったファイルは、ユーザがそれぞれいつでもリポジトリにコミットしてしまえるために、バージョン管理が難しいのです。

そういった場合、テンプレート ファイルを使うのをお勧めします。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。開発者のそのファイルをチェックアウトします。それから、開発者ごとにそのファイルの 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 のビルド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。

ユーザが不安にならないように、TortoiseVars.bat を親フォルダの無視リストに追加してあります。つまり、Subversion の svn:ignore 属性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) 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 エクスプローラの作業コピーを作成したい場所で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。すると以下のダイアログが表示されます。

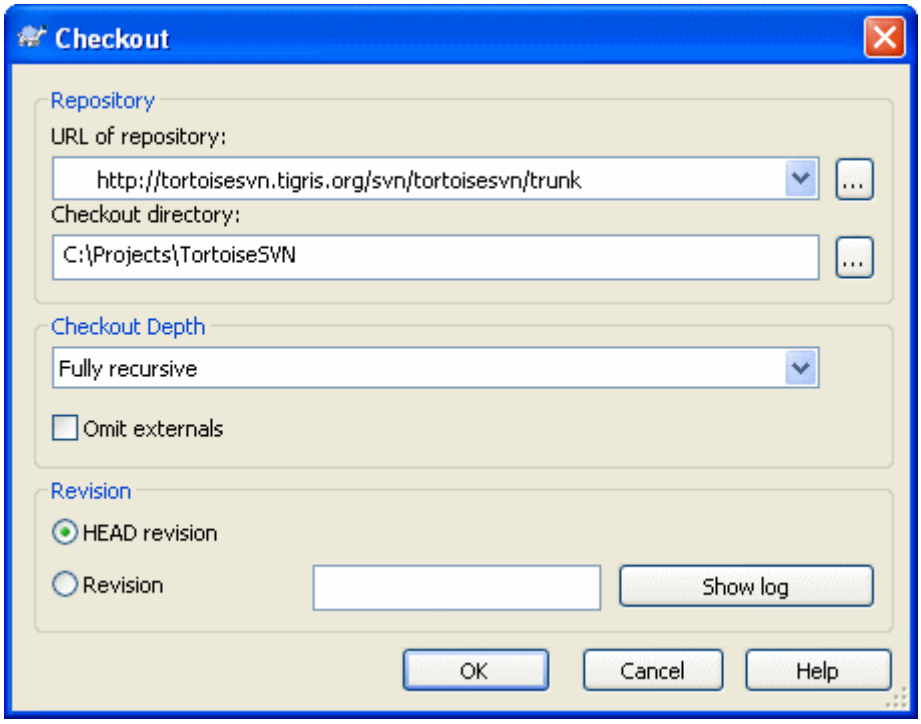


図4.7 チェックアウトダイアログ

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度

チェックアウトの際、子フォルダを再帰する　深度　を選択できます。大きなツリーの一部が必要なだけなら、トップレベルフォルダのみをチェックアウトし、それから選択したフォルダを再帰的に更新します。

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

直接の子階層 (フォルダを含む)
: 指定したディレクトリに対して、すべてのファイルや子フォルダを含めてチェックアウトしますが、子フォルダ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に対して、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度を保持します。このオプションは、チェックアウトダイアログには使用しませんが、深さの設定を持つその他のダイアログでは、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、**特定のリビジョンへ更新** ダイアログでしか使用できません。

わずかな作業コピーをチェックアウトする (チェックアウト深度を **再帰的** 以外にする等) 場合、リポジトリブラウザ (「リポジトリブラウザ」) や変更をチェックダイアログ (「こちらの状態とあちらの状態」) を用いて、サブフォルダを追加で取得できます。

リポジトリブラウザでは、チェックアウトしたフォルダを 右クリック し、TortoiseSVN → **リポジトリブラウザ** でリポジトリブラウザを表示してください。作業コピーに追加するサブフォルダを探し、**項目を特定リビジョンへ更新...** を使用してください。このメニューは、作業コピーに親項目が存在し、かつ選択した項目が存在しない場合にのみ表示します。

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。リポジトリにあるが、チェックアウトされていないファイル・フォルダを、**リモート操作による追加** として、すべてこのダイアログに表示します。作業コピーに追加したいフォルダを 右クリック し、**コンテキストメニュー → 更新** としてください。

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がある大きなツリーがあり、`Project03`, `Project25`, `Project76/SubProj` だけチェックアウトしようとしていると仮定しましょう。以下のような手順となります。

1. 深度を「この項目のみ」として親フォルダを チェックアウトしてください。すると、空のトップレベルフォルダが手に入ります。

2. リポジトリの内容を表示するため、新しいフォルダを選択し、TortoiseSVN → **リポジトリブラウザ** を使用してください。

3. `Project03` を右クリックし、Context menu → **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をすべて取得します。

   `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` フォルダには、これだけですべて配置する `SubProj` 以外に何も内容がないことに注意してください。Subversion は、内容の取得をせずに、中間フォルダのみ作成しました。

作業コピーの深さの変更

一度、作業コピーを特定の深さでチェックアウトしている場合、**コンテキストメニュー → 項目を特定リビジョンへ更新...** を使用して後から深さを変更できます。

旧式サーバの利用

1.5 以前のサーバは、作業コピーの深度に対するリクエストを理解しないため、常にリクエストを効果的に処理できません。それでもコマンドは動作するでしょうが、旧式のサーバは、クライアントが必要でなくフィルタしたままにしているすべてのデータを送ってしまいます。そのため大量のネットワークトラフィックも発生してしまいます。可能であれば、サーバを 1.5 にアップグレードするべきです。

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 にチェックがついていたり、深度を増やしたい場合、作業コピーを更新するのに TortoiseSVN → 更新TortoiseSVN → **特定のリビジョンへ更新...** を実行する必要があります。通常の更新では外部参照を全て取得し、既存の深度を保持てしまいます。

ディレクトリツリーの トランク （trunk） 部分か、それ以下をチェックアウトするのをお勧めします。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

時には .svn ディレクトリを除いてコピーしたいかもしれません。ソースの圧縮した tarball 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. 変更点をリポジトリにコミット

作業コピーへの変更を送信するのは、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 して、サーバや手元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を選択し、TortoiseSVN → コミット... を行ってください。

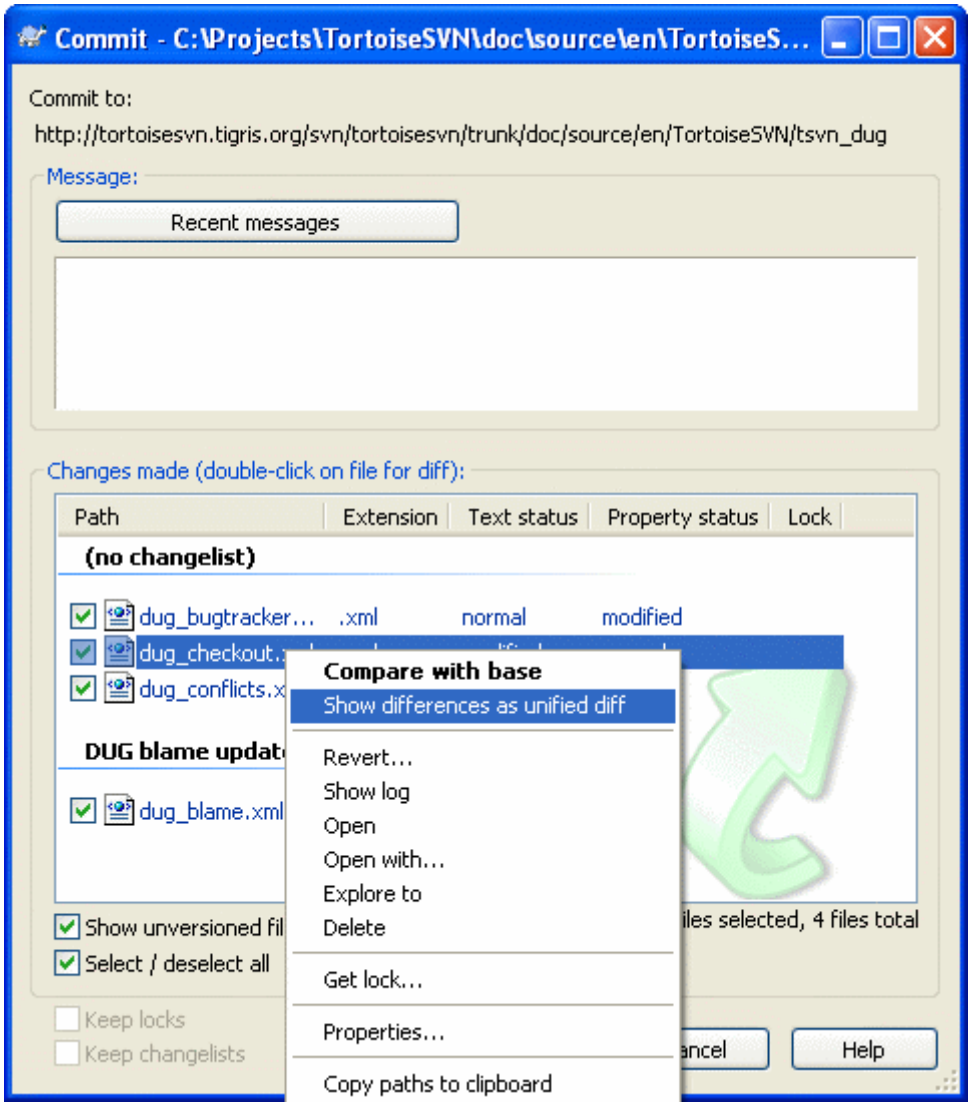


図4.8 コミットダイアログ

コミットダイアログには変更したファイルが、追加・削除・バージョン管理外を含め表示されます。変更し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の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、フォルダをコミットすればそのファイルを探します。フォルダ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。

svn:externals　を使って別のリポジトリから取得したファイルにも変更がある場合、その変更を同時に不可分コミットできません。この場合、警告シンボルがファイルリストにつき、外部参照ファイルは別々にコミットしなければならないとツールチップで教えてくれます。

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 (コンパイラが生成したファイルやエディタのバックアップなど) を表示しすぎだと思う場合は、以下のようないくつかの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 属性を設定したディレクトリにのみ影響します。SVN 属性ダイアログで そのディレクトリの svn:ignore 属性を修正できます。

詳細は「無視するファイルとディレクトリ」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・未選択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。

下のペインに表示された列は、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を表示します。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を表示し、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、次回以降も同じ列見出しになります。

デフォルトでは変更をコミットする際に、保持していたファイルのロックが、コミット完了後自動的に開放されます。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します。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。長々フォルダを変更チェックする、トップレベルフォルダからコミットするのを避けたければ、あるフォルダ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 SVN は自動的に追加します。

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ることがあり、その場合、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。単純に古い名前 (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** を用いて 2 つのファイルが名前変更であると指定してください。

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

頻繁に変更するけれども、コミットしたくないバージョン管理下のファイルがあることがあります。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を更新しただけではコミットする必要はありません。

このように、やっかいなケースを解決するのに、`ignore-on-commit` という変更リストを予約しておきます。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にて自動的にチェックが外されます。それでも変更をコミットしたい場合は、コミットダイアログにて、手で選択しなければなりません。

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のままですが、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えます。`text` にスタイルを適用する場合、`*text*` で太字、`_text_` で下線、`^text^` で斜体を用いてください。

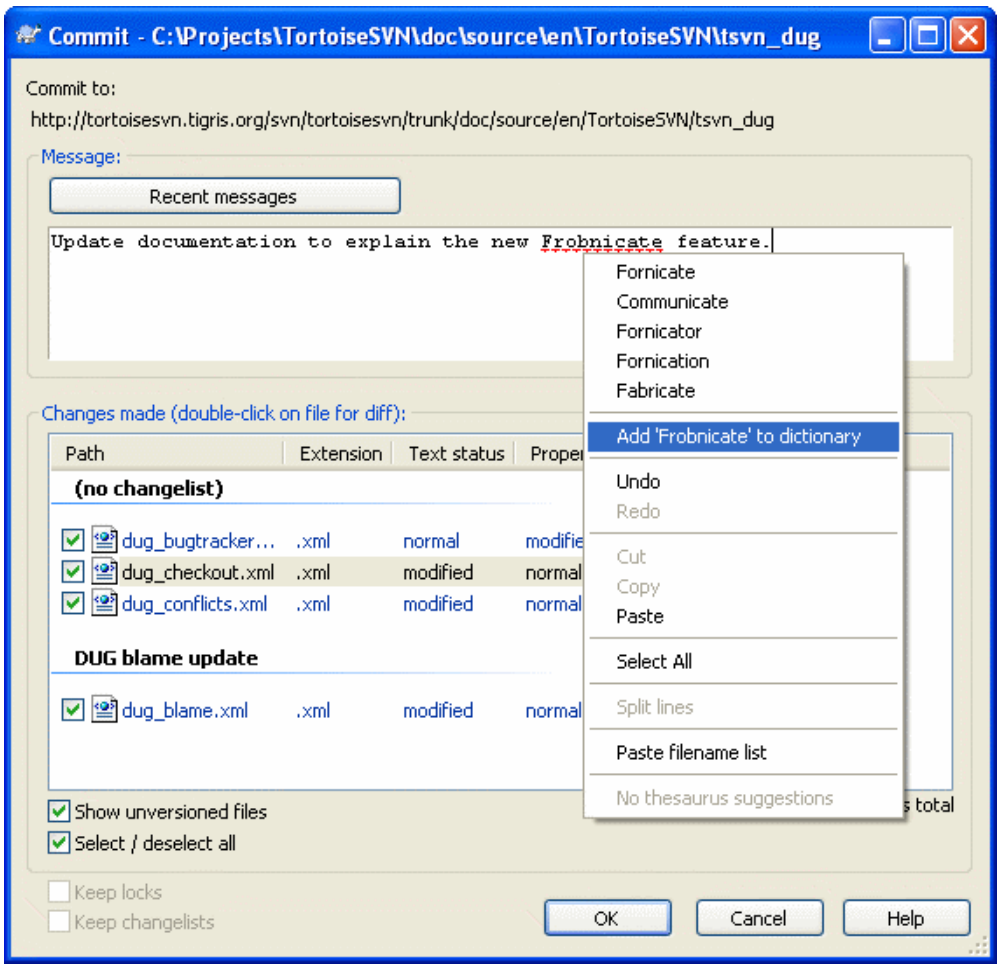


図4.9 コミットダイアログのスペルチェッカ

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。提案する訂正内容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 の技術用語を知っているわけではありません。そのため正しい綴りの単語をエラーとして表示知る可能性があります。ご心配なく。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・関数の自動補完機構も持っています。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。入力している単語が (3 文字入力したり、Ctrl+Space を押して)、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 に渡される正規表現は、TortoiseSVN をインストールした `bin` フォルダで保持されます。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] にある導入や、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。

前回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。

保存したすべてのコミットメッセージを、TortoiseSVNの設定ダイアログの**保存データ** ページで消去したり、**最近のログメッセージ** ダイアログで Delete キーを用いて、特定の保存したログメッセージを消去できます。

チェックしたパスをログメッセージに入れたい場合、エディットコントロールで **コンテキストメニュー → ファイル名のリストを貼付** を使用できます。

ログメッセージにパスを入力する別の方法は、単純にファイルリストにあるファイルを、エディットコントロールにドラッグすることです。

特殊なフォルダ属性

コミットログメッセージのフォーマットを制御したり、スペルチェッカで使用する言語を指定する、特殊なフォルダ属性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

バグ追跡システムが有効なら、Bug-ID / Issue-Nr: テキストボックスで課題を複数設定できます。複数の課題はコンマで区切らなければなりません。そうでなければ、正規表現ベースのバグ追跡サポートを使用して、ログメッセージの一部として課題への参照を追加してください。詳細は「バグ追跡システム / 課題追跡システムとの統合」をご覧ください。

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を表示します。

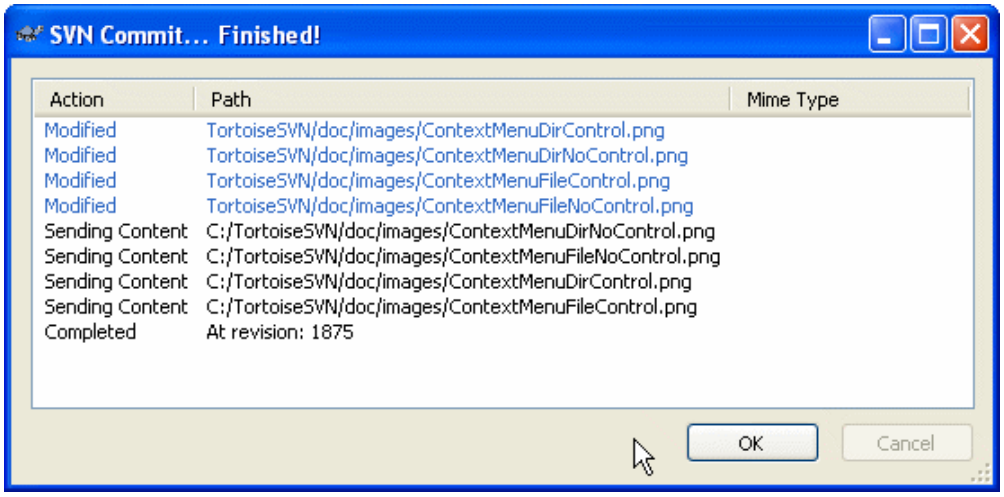


図4.10 コミットの状況を表示している進行ダイアログ

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します。

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 での色設定」をご覧ください。

4.5. 他人の修正に伴う作業コピーの更新

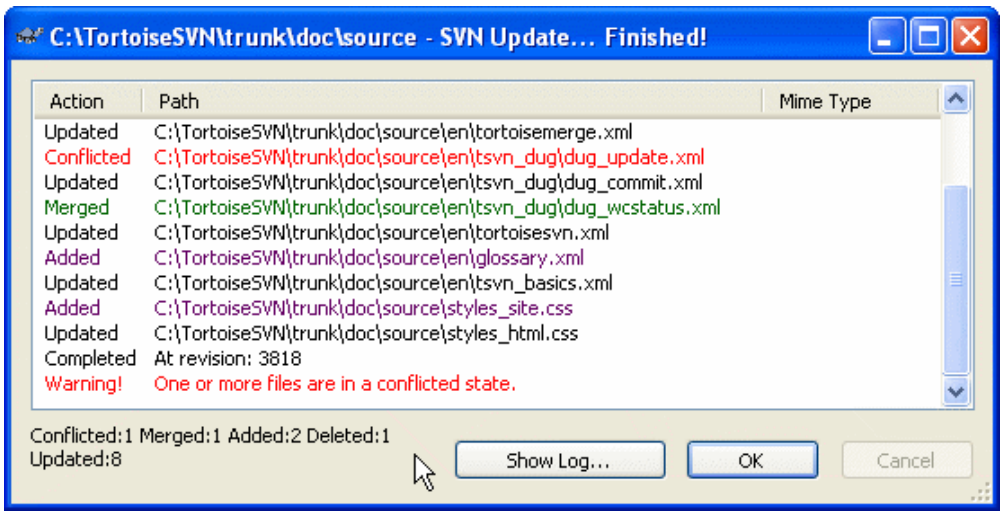


図4.11 更新が完了したときの進行ダイアログ

定期的に、手元の作業コピーに他の人が行った変更を、確実に取り込まなくてはなりません。サーバから手元のコピーへ変更を取り込む手順を、更新 と言います。更新はひとつのファイルに対して行われることも、選択したファイルに対して行われることも、ディレクトリ構造に対して再帰的に行われることもあり得ます。更新するには、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のコンテキストメニューから TortoiseSVN → 更新 を選んでください。更新の進行を表すウィンドウがポップアップし、更新を始めます。他の人が行った変更が、あなたのファイルにマージされますが、あなたが行った変更は保持されています。更新では、リポジトリは影響を受けません。

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しています。

紫
: 作業コピーに追加した新規項目

暗赤色
: 作業コピーから削除された余計な項目と作業コピーに置き換えた紛失項目

緑
: リポジトリからの変更を手元の変更とうまくマージしました。

明赤色
: リポジトリからの更新を手元の更新とマージしたら、競合が発生したので解消しなければなりません。

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ました。

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 での色設定」をご覧ください。

更新中に 競合 (同時に同じファイルの同じ行を変更して、それが一致しなかった場合に起きる) 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、競合を解消するよう外部マージツールを起動できます。

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では、オプションを付けず、作業コピーをリポジトリの最新 (HEAD) リビジョンに更新するだけです。これが最も一般的なケースです。更新プロセスをもっと制御したい場合は、代わりに TortoiseSVN → 特定のリビジョ

ンへ更新... を使用してください。これで作業コピーを最新のものだけでなく、指定したリビジョンに更新できます。作業コピーがリビジョン 100 だったとして、リビジョン 50 にある状態を反映したい場合、単純にリビジョン 50 へ更新します。同じダイアログで、現在のフォルダを更新する際の 深度 を選択できます。これは「チェックアウトの深度」で説明しています。デフォルト値は、既存の深さを変更しないように、**作業コピー** 戸なっています。また、更新時に外部プロジェクト (`svn:externals` で関連づけたプロジェクト) を無視して更新できます。

注意

リビジョンを指定してファイルやフォルダを更新した場合、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、「out of date」エラーが発生します! ファイルへの変更を取り消し、以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ロールバックしてください。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。

特定のリビジョンへ更新 は、履歴中の以前のポイント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、個々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たままにしかねないので、よい考えとは言えません。更新したファイルの名前を変更すると、作業コピーからそのファイルが消えてしまい、見つからなくなる可能性があります。以前のリビジョンにはそんな名前のファイルは存在しないからです。更新によって現れたファイルとの区別が付かないため、通常の、緑のオーバーレイの付いた項目にも注意を払うべきです。

単にそのファイルの古いリビジョンを手元のコピーに欲しければ、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・フォルダ

エクスプローラで複数のファイルやフォルダを選択し、**更新** を選択した場合、そのファイルやフォルダ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持ってきたファイルやフォルダは、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。

既存のローカルファイル

更新していると時々、手元に同じ名前のファイルがあるというメッセージを出して、更新が失敗することがあります。典型的には、Subversion で新しいバージョンのファイルを取得したときに、すでに作業フォルダにバージョン管理外の同じ名前のファイルがある場合に発生します。Subversion がバージョン管理外のファイルを上書きすることはありません。偶然他の開発者が同じファイル名で作成し、コミットしてしまった場合など、あなたの作業した内容が何か含まれているかもしれないからです。

このエラーメッセージがでた場合、簡単な解決法は手元にあるバージョン管理外のファイル名を変更することです。名前を変更したファイルがまだ必要かどうかチェックもできます。

このエラーメッセージがまだ出続ける場合、問題のファイルの一覧を出す代わりに TortoiseSVN → **変更をチェック** をしてみてください。この方法で、一気にその問題に対処できます。

4.6. 競合の解消

リポジトリからあなたのファイルを更新・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。

ツリー競合
: 開発者がファイルやフォルダの移動・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. ファイル競合

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 はプロジェクトについては何も知りませんから、開発者間の競合については解決できません。競合が報告されたら、問題のファイルを開き、<<<<<<< で始まる行を探してください。競合のある箇所は以下のようにマークがつけられています。

```
<<<<<<< filename
 あなたの変更
=======
 リポジトリからマージされたコード
>>>>>>> revision
```

また、競合のあるファイルごとに Subversion は手元のディレクトリにファイルを 3 つ追加します。

filename.ext.mine
: 作業コピーを更新する前に、(競合マーカがついていない) 作業コピーに存在するファイルです。このファイルには最新の変更以外何もありません。

filename.ext.rOLDREV
: 作業コピーを更新する前の BASE リビジョンのファイルです。最新の編集を行う前、チェックアウトした状態のファイルです。

filename.ext.rNEWREV
: 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。

外部マージツール・競合エディタを TortoiseSVN → **競合の編集** で起動できますし、競合の解消に他のエディタを使って手で修正もできます。コードがどうあるべきか決めて、必要な変更をしてから保存してください。

その後、TortoiseSVN → **問題の解消** 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。問題の解消コマンドは、実際には競合を解消しないことに注意してください。変更をコミットできるよう、`filename.ext.mine` ファイルや `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。手元のファイルには変更を加えず (あなたが変更したままを保証)、`filename.ext.r*` ファイルを作ります。自分の変更を破棄し、リポジトリのバージョンにしたい場合は、「元に戻す」コマンドを使うだけです。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、「問題の解消」コマンドを行ってコミットしてください。

親フォルダを右クリックし、TortoiseSVN → **問題の解消...** を選択すると、「問題の解消」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。解消するものを選択してください。

4.6.2. ツリーの競合

開発者がファイルやフォルダを移動・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、その競合を解消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、競合を解消しようと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しますが、競合を整理するにはさらに作業が必要かもしれません。作業中の BASE 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。

4.6.2.1. 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します。

2. 開発者 B は、同時に作業コピーで `Foo.c` を `Bar.c` に移動したり、単にその親フォルダで `Foo.c` を削除しました。

開発者 B の作業コピーを更新した結果、以下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の競合の場合、`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、`Foo.c` に行った変更をマージできます。シンプルな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、何もせずに競合解決マークをつけ、事実上開発者 A の変更を破棄します。

名前変更された `Bar.c` のオリジナルファイルが見つかれば、競合編集ダイアログは変更をマージするか訊いてきます。どこで更新を実行したかによって、ソースファイルが見つからない可能性があります。

4.6.2.2. 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。

またはフォルダ移動の場合...

1. 開発者 A は、親フォルダ `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以下のようになります。

- `Bar.c` へ作業コピーへ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、ツリーの競合があります。

フォルダの競合では以下のようになります。

- `BarFolder` が作業コピーに、通常のフォルダとして追加されます。

・ FooFolder (履歴と共に) 追加されたとマークされ、ツリーの競合があります。

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 の再編成を受け入れ、新しい構造にある対応するファイルに変更をマージするか、開発者 A の変更を取り消し、自分のファイルを保持するかを決めねばなりません。

自分の変更を入れ替えてマージするため、開発者 B はまず、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。ログダイアログを使用するとわかります。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、この手順を自動化する方法はなく、単純にする方法すらないからです。一旦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。この場合、競合エディタダイアログの 削除 ボタンを使用し、掃除と競合解決マークを付けてください。

開発者 A の変更は誤りだと開発者 B が判断した場合、競合エディタダイアログの 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を、解決としてマークますが、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、またログダイアログが役に立ちます。

4.6.2.3. 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、以下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」状態で追加されます。

・ Foo.c は削除マークが付き、ツリーが競合した状態となります。

競合を解決するには、Foo.c がリポジトリで名前変更・移動し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。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、手で競合を解決した後で、競合エディタダイアログのボタンで、ツリーの競合に解決マークを付ける必要があります。

4.6.2.4. 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、作業コピーが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 はツリーの競合として、紛失マークが付きます。

この競合を解決するには、開発者 B が競合エディタダイアログでファイルに解決マークを付け、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、それから解決マークを付けると、あなた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。

4.6.2.5. 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 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、作業コピーが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 の再編成を受け入れ、新しい構造にある対応するファイルに変更をマージするか、開発者 A の変更を取り消し、自分のファイルを保持するかを決めねばなりません。

自分の手元の変更をバラバラにしてマージするために、開発者 B は、競合ファイル Foo.c がリポジトリで、どのファイル名に名前変更・移動されたかを、まず調べ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して行うことができます。競合エディタは、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。そして今回、マージを自動化することも、簡単にマージする方法すらない場合は、変更を手でマージしなければなりません。一度変更を適用してしまえば、競合したパスはもう必要なく、削除できます。この場合、掃除と競合に解決マークをつけるために、競合エディタの 削除 ボタンを使用してください。

開発者 B が、開発者 A の変更は誤りだと判断した場合、競合エディタダイアログの 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を、解決としてマークますが、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。

4.6.2.6. 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、作業コピーが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。競合エディタは、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、手で競合を解決した後で、競合エディタダイアログのボタンで、ツリーの競合に解決マークを付ける必要があります。

4.7. ステータス情報取得

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

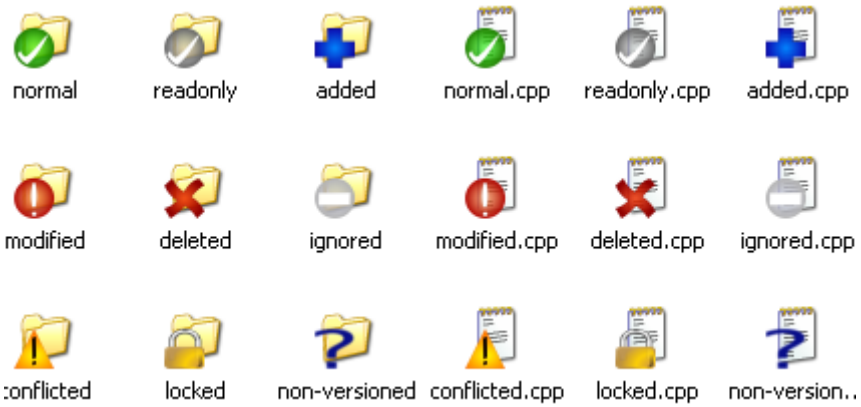


図4.12 エクスプローラのアイコンオーバーレイ表示

今、Subversion リポジトリから作業コピーにチェックアウトしたとすると、Windows エクスプローラ上でファイルのアイコンが変わっていることと思います。これこそ TortoiseSVN の評判がいい理由のひとつです。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 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 修正 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 属性をセットしていると、そのファイルにロックをかけるまで Subversion は読み取り専用にします。この読み取り専用ファイルには、編集前にロックするまで、このオーバーレイを表示します。

ファイルをロックして Subversion 状態が 通常 ならこのアイコンが表示され、他の人がそのフィルへの変更をコミットしても良いのにロックを解放忘れをしないようにしてくれます。

バージョン管理から 削除 されるカレントフォルダ内のファイルやフォルダ、またバージョン管理が紛失したフォルダ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には+サインがつきます。

横棒サインは、ファイルやフォルダが、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で、無視されない場合には、このアイコンを表示します。このオーバーレイはオプションです。

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かなり制限されており、また TortoiseCVS の旧バージョンも使用していると、使用できるオーバーレイスロットが足らなくなります。そのため TortoiseSVN は「Good Citizen (TM)」に挑戦し、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。

現在では、たくさんの Tortoise クライアントが存在する (TortoiseCVS, TortoiseHG, ...) ため、アイコンの限界は現実の問題になっています。これに対処するため、TortoiseSVN プロジェクトは Tortoise クライアントで共通に使用できるように、DLL で読み込む共有アイコンセットを導入しました。もう統合されているかを、各クライアントのプロバイダでチェックしてみてください。:-)

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. Windows エクスプローラの TortoiseSVN 列

アイコンオーバーレイと同様 (またそれ以上) の情報を、Windows エクスプローラの詳細ビューに追加する列で表示できます。

単に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の、その他... を選んでください。「詳細ビュー」 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。お好みに応じて、ドラッグ&ドロップで順番を変えたり大きさを変更してください。

重要

Windows エクスプローラへの列の追加は、Vista では利用できません。Microsoft が全ファイルへのそのような列の追加を、もう許さず、特定のファイルタイプのみにしたからです。

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。

4.7.3. こちらの状態とあちらの状態

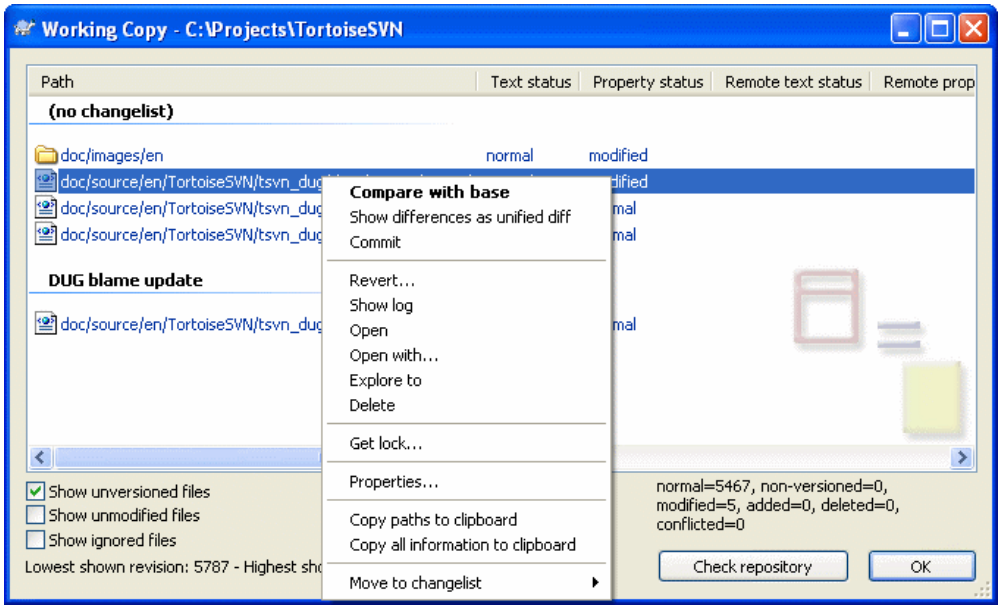


図4.13 変更をチェック

あなたが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で何らかの変更をした全てのファイルを表示し、バージョン管理外のファイルも指定すれば同様に表示します。

リポジトリをチェック をクリックすると、リポジトリの変更点も参照できます。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、作業コピーに対してチェックアウトした深さまでしか、リモートの状態を取得しません。リポジトリ内のファイル・フォルダすべてを、チェックアウトしていなくても見たい場合は、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、状態を色分けで表示します。

青
: ローカルで修正した項目

紫
: 追加した項目。履歴から追加した項目には **テキストの状態**列に + 印がつきます。また、そこにコピーされた項目にはツールチップが現れます。

暗赤色
: 削除・紛失した項目

緑
: ローカル・リポジトリで修正しした項目。更新時にマージされます。更新時に競合する可能性があります。

明赤色
: ローカルで修正されリポジトリでは削除された、ないしリポジトリで修正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
: 更新がなくバージョン管理外の項目

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 での色設定」をご覧ください。

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!

このダイアログのコンテキストメニューから、変更点の差分を取れます。あなた が行った手元の変更をチェックするのには、**コンテキストメニュー → Base と比較** を使用してください。他の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 Diff 形式で差分を表示** を使用してください。

個々のファイルhwの変更を元に戻すこともできます。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、修復するのに 元に戻す ことができます。

バージョン管理外のファイルや無視ファイルは、**コンテキストメニュー → 削除** でゴミ箱に送れます。完全に (ゴミ箱経由なしで) 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。

ファイルの詳細を調べたければ、ここからテキストエディタや IDE のような、別のアプリケーションにドラッグできます。

下のペインに表示された列は、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を表示します。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を表示し、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、次回以降も同じ列見出しになります。

関連のない複数のタスクを一度に処理する場合、変更リストにあるファイルをまとめて扱うこともできます。詳細情報は「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、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。リビジョン範囲は作業コピー全体ではなく、表示されている項目のみに対して表してい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、**未変更ファイルを表示** チェックボックスにチェックを入れなければなりません。

ヒント

作業コピーをフラットに参照 (フォルダ階層のすべてのファイルとフォルダを表示するなど) したければ、**変更をチェック** ダイアログが一番簡単です。**未変更ファイルを表示** 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を表示します。

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ることがあり、その場合、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。単純に古い名前 (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** を用いて 2 つのファイルが名前変更であると指定してください。

4.7.4. 差分表示

時にはどんな変更をしたのかファイルの中を見たくなるでしょう。それには、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 を選択するとできます。これで外部差分ビューアを起動し、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 (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のどちらでも、差分を表示できます。

エクスプローラで比較したい2 つのファイルを選択 (例 Ctrl とマウスを使用) し、TortoiseSVN のコンテキストメニューから 差分 を選択してください。最後にクリックしたファイル (フォーカスのあるもの。例 点線で四角囲われている) を最新として扱います。

4.8. 変更リスト

理想的な世界では、常に一度に 1 つのことしか行わず、作業コピーには論理的にひとかたまりの変更しかありません。OK。では現実に戻りましょう。複数の関係ない仕事を、一度に行わなければならないことが、よくあります。そしてそのときの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かち合ってしまってはいけません。同じファイルに対する異なるタスクがある場合、変更点を分割する方法はありません。

重要

TortoiseSVN の変更リスト機能は、Windows 2000 では提供されていないシェル機能に依存しているため、Windows XP 以降でしか機能しません。申し訳ありませんが、Win2K は今となっては古すぎます。ご容赦ください。

更新リストを様々な場所で目にしますが、最も重要なのはコミットダイアログと更新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、その後で更新チェックダイアログを開きましょう。初めて更新チェックダイアログを開くと、変更のあるファイルをすべて表示します。機能に照らして、更新リストを構成し、ファイルをグループ化するとしたらどうでしょう。

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。最初はチェンジリストがありません。そのため、初めて追加する際には、変更リストを作成することになります。なんのために使用するのか判る名前を、変更リストに付けて、**OK** をクリックしてください。ダイアログが変更され、項目のグループが表示されるようになります。

一度更新リストを作成すると、別の更新リストや Windows エクスプローラから、項目をドラッグ & ドロップできます。エクスプローラからドラッグするのは、ファイルを変更する前に更新リストに追加できるので、便利です。更新チェックダイアログからも同じことができますが、未更新ファイルすべてを表示しなければなりません。

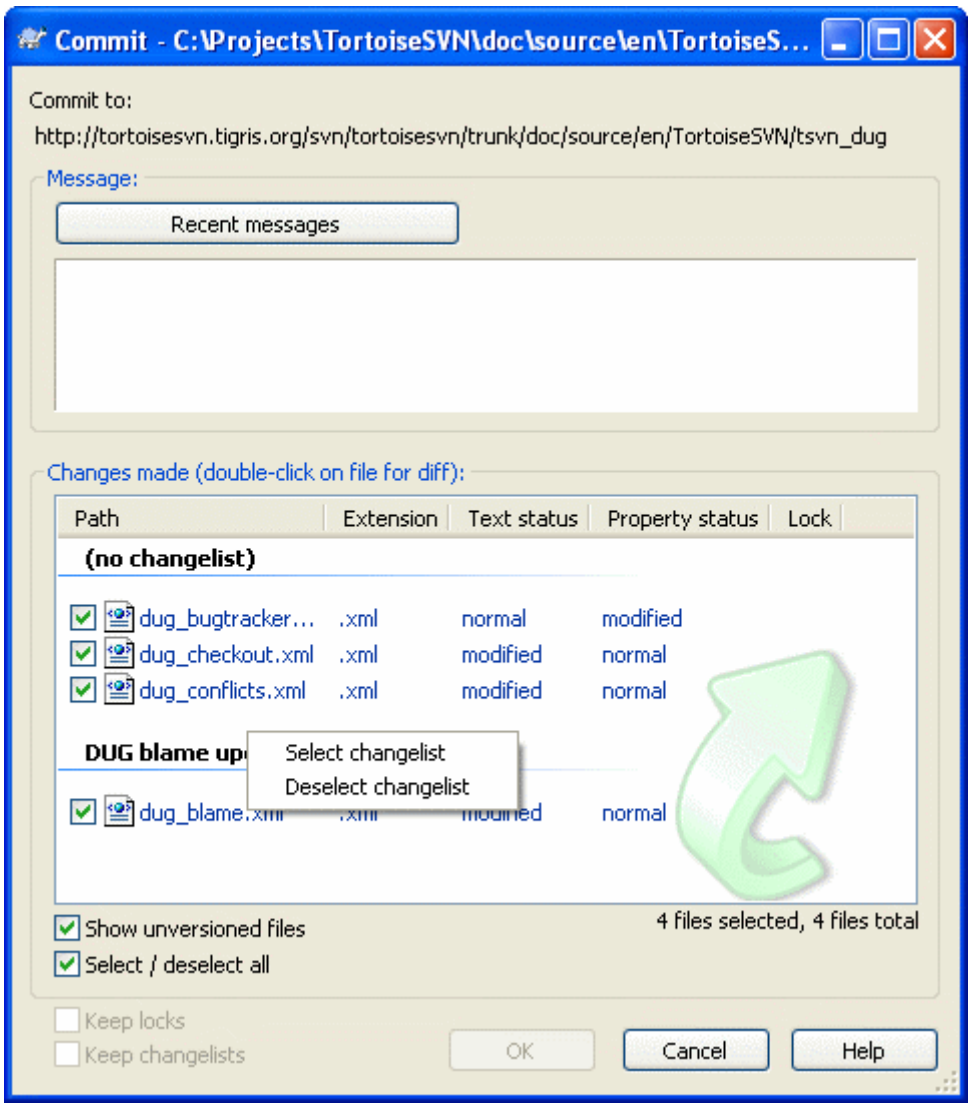


図4.14 変更リスト付きコミットダイアログ

コミットダイアログでは、更新リストでグループ化した、同じファイルを見られます。グルーピングの直接的な表示方法は別としても、コミットするファイルを選択する際に、グループの見出しとしても利用できます。

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。しかし、Vista に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 は自身で使用する更新リストを、`ignore-on-commit` という名前で予約しています。これは、手元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。

更新リストに属するファイルをコミットする際、通常更新リストに入れておく必要がなくなったと仮定できます。そのためデフォルトでは、コミットしたファイルは、更新リストから取り除かれます。更新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にある、**変更リストを保持する** チェックボックスを使用してください。

ヒント

変更リストは完全にローカルクライアントの機能です。変更リストの作成・削除はリポジトリや他の誰かの作業コピーに影響しません。これは単にあなたがファイルを構成する、便利な方法に過ぎません。

4.9. リビジョンログダイアログ

変更・コミットのたびに、変更点をログメッセージに残しておくべきです。これを見れば、あと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は、ログメッセージをすべて取得して表示します。表示は 3 画面に分けられています。

- 上部ペインには、ファイル・フォルダへの変更のコミットごとのリビジョンを表示します。この概要には日付時間、そのリビジョンをコミットした人、ログメッセージの先頭が含まれています。

  青く表示している行は、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。

- 中央ペインには、選択したリビジョンの全ログメッセージを表示します。

- 下部ペインには、選択したリビジョンで変更したファイル・フォルダの一覧を表示します。

しかしそれよりも多くのことを行います。プロジェクトの履歴情報取得できるような、コンテキストメニューを用意しています。

4.9.1. リビジョンログダイアログの起動

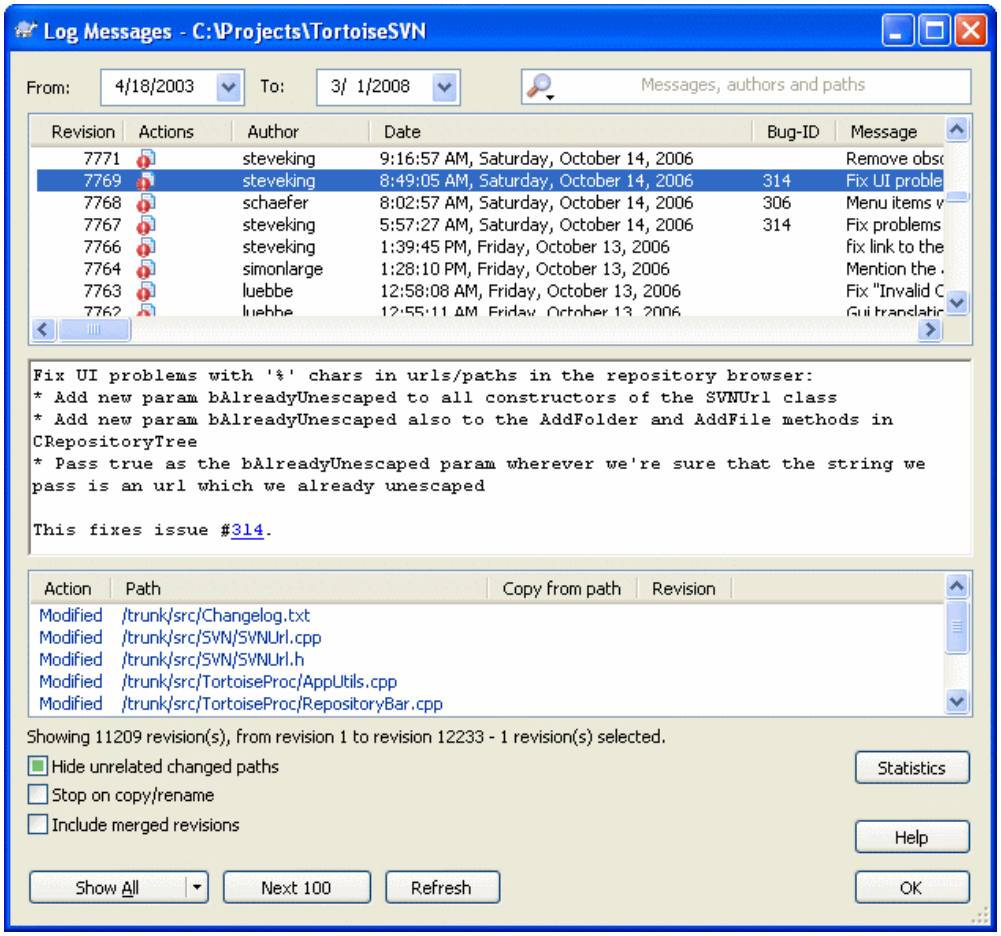


図4.15 リビジョンログダイアログ

いろいろな場所からログダイアログを表示できます。

- 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します。

リポジトリが利用できない場合、「オフラインモード」で説明する、**オフラインにしますか?** ダイアログを目にすることになります。

4.9.2. リビジョンログのアクション

上部ペインには、そのリビジョンで何が行われたかをまとめた **アクション** 列があります。4 つのアイコンが、それぞれ 1 列ずつ表示されます。

ファイルやディレクトリにそのリビジョンで変更がある場合、変更 アイコンが第一列に表示されます。

ファイルやディレクトリがそのリビジョンで追加された場合、追加 アイコンが第二列に表示されます。

ファイルやフォルダがそのリビジョンで削除された場合、削除 アイコンが第三列に表示されます。

ファイルやフォルダがそのリビジョンで置換された場合、置換 アイコンが第四列に表示されます。

4.9.3. 追加情報の取得

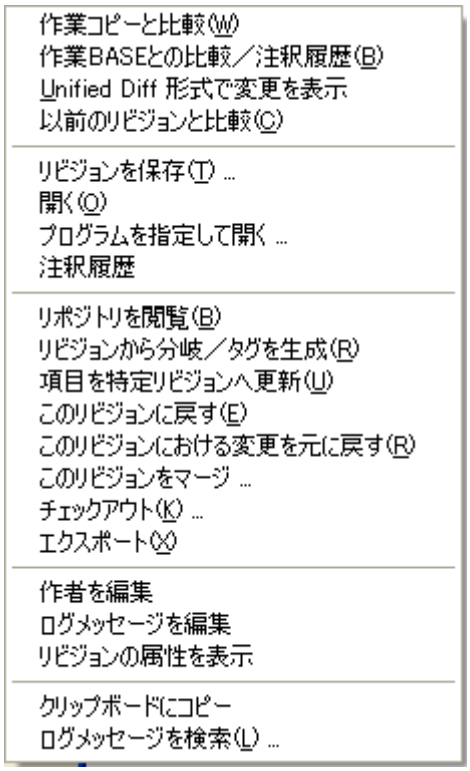


図4.16 リビジョンログダイアログ上部ペインのコンテキストメニュー

ログダイアログの上部ペインには、より詳細な情報にアクセスす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でしか現れないものと、フォルダにしか現れないものがあります。

作業コピーとの比較
: 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。フォルダに対するログダイアログの場合、変更のあったファイルを一覧表示し、個々のファイルそれぞれに対し変更点の検査ができます。

作業中の BASE との比較と注釈
: 選択したリビジョンと作業中の BASE にあるファイルの注釈をとり、差分ツールを使って注釈レポートを比較できます。詳細は、「注釈履歴の差分」をご覧ください (ファイルのみ)。

unified diff で変更表示
: 選択したリビジョンで行った変更点を、Unified-Diff ファイル (GNU patch 形式) で表示します。これには文脈内の数行で変更点のみ表示します。視覚的なファイル比較よりも読むのが大変ですが、すべての変更を一度にコンパクトな形式で確認できます。

前のリビジョンとの比較
: 選択したリビジョンを直前のリビジョンと比較できます。作業コピーとの比較と同じように動作します。フォルダに対するこの操作は、比較するファイルを選択する、「変更されたファイル」ダイアログを、まず表示します。

前のリビジョンとの比較と注釈
: ファイルを選択できるように、「変更されたファイル」ダイアログを表示します。選択したリビジョンと直前のリビジョンの注釈をとり、ビジュアル差分ツールを使って結果を比較できます (フォルダのみ)。

リビジョンを保存...
　選択したリビジョンのファイルを保存し、古いバージョンのファイルを取得します。

開く/プログラムを指定して開く...
　選択したファイルを、ファイルタイプに応じた規定のビューアか選択したプログラムで開きます。(ファイルのみ)

注釈履歴...
　選択したリビジョン以降のファイルの注釈を取得します。(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を検査するため、リポジトリブラウザ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/タグを生成
　選択したリビジョンからブランチ・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場合に便利です。

項目を特定リビジョンへ更新
　選択したリビジョンに作業コピーを更新します。時間をさかのぼり、作業コピーを過去の状態にしたり、リポジトリにさらにコミットが行われており、作業コピーを一度に更新したい場合に便利です。1　ファイルだけでなく、作業コピー全体のディレクトリを更新するのが最善です。そうでなければ作業コピーに矛盾が生じ、変更をコミットできなくなってしまいます。

　以前の変更点を恒久的に取り消したい場合、**このリビジョンに戻す** を代わりに使用してください。

このリビジョンへ元に戻す
　以前のリビジョンに変更を戻します。既にいくつか変更を行っており、本当にリビジョン N へ戻したい場合、あなたの求めるものがこのコマンドです。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この操作はリポジトリに影響を　与えません。これは、ファイル・フォルダを以前のリビジョンに置き換えて、選択したリビジョン以降の変更を　すべて　取り消してしまうことに注意してください。ローカルで変更している場合、このコマンドはその変更を作業コピーにマージします。

　作業コピーが未変更の場合、この操作を実行すると、作業コピーが変更されます。すでに変更している場合、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。

　内部的に起こっていることは、選択し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　Subversion　が実行し、以前のコミットの効果を元に戻すということです。

　この操作後に、元に戻したものを元に戻し、作業コピーを前回の未変更状態に戻す場合、Windows エクスプローラから TortoiseSVN → **元に戻す** を使用し、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・フォルダがあったのか見たいだけであれば、代わりに **特定のリビジョンへ更新** 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元に戻す
　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! そのリビジョンのみの変更を元に戻すということに注意してください。以前のリビジョンで作業コピーのファイル全体を置き換えるわけではありません。他に関係ない変更をおこなってから、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場合、この操作を実行すると、作業コピーが変更されます。すでに変更している場合、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。

　内部的に起こっていることは、ひとつのリビジョンに対して逆マージを Subversion が実行し、以前のコミットの効果を元に戻すということです。

このリビジョンへ元に戻す で説明しているように、元に戻したものを元に戻すことができます。

このリビジョンをマージ...
異なる作業コピーにあるリビジョンを選択し、マージを行います。フォルダ選択ダイアログにより、マージ結果の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。変更し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うまい方法です! 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な機能です。

チェックアウト...
選択したリビジョンを選択したフォルダに、改めて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...
選択したファイルやフォルダを、リビジョンを選んでエクスポートできます。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・ログメッセージの編集
以前行ったコミットのログメッセージや作者を編集できます。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。

リビジョン属性の表示
ログメッセージと作者以外の、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できます。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できます。入力したログメッセージと、Subversion が作成した動作概要 (下部ペインに表示) も検索範囲になります。大文字小文字は区別しません。

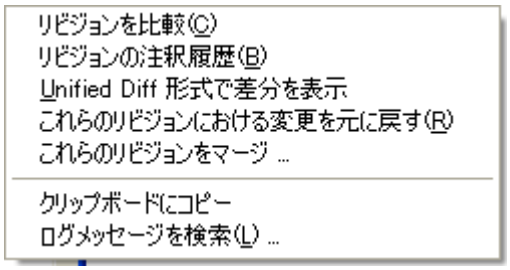


図4.17 2 リビジョン選択時の上部ペインコンテキストメニュー

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変わり、以下の機能だけになります。

リビジョンを比較
選択した 2 つのリビジョンを視覚差分ツールで比較します。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。

これをフォルダに対して行った場合、さらに変更のあったファイルを一覧表示するダイアログを表示し、diff の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をとり、視覚差分ツールを使って比較できます。詳細は、「注釈履歴の差分」 をご覧ください。

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

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リビジョンの範囲を指定してすべての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一度に元に戻す、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。

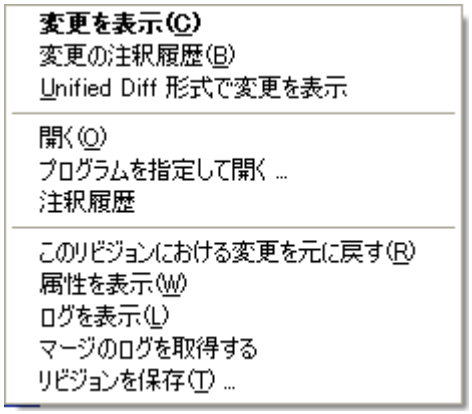


図4.18 ログダイアログ下部ペインのコンテキストメニュー

ログダイアログの下部ペインにもコンテキストメニューがあり、以下を行えます。

変更の表示
: 選択したファイルの選択したリビジョンの差分を表示します。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。

変更の注釈
: 選択したファイルの、選択したリビジョンとその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 diff での表示
: ファイルの変更点を unified diff 形式で表示します。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、ファイルタイプに応じた規定のビューア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元に戻す
　このリビジョンの選択したファイルへの変更を、元に戻します。

属性の表示
　選択した項目の Subversion 属性を表示します。

ログの表示
　選択した 1 ファイルのリビジョンログを表示します。

マージログの取得
　選択した 1 ファイルに対して、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...
　そのファイルがまだ古いバージョンの場合、選択したリビジョンのファイルを保存します。

ヒント

私たちが変更と言ったり差分と言ったりしていることに、お気づきのことと思います。では差分とはなんでしょうか?

Subversion は、2 つの異なるものを表すのにリビジョン番号を使用します。一般的にリビジョンはその時点でのリポジトリの状態を表しますが、そのリビジョンで作成されたチェンジセットを表すのにも使用します。例えば「r1234 で完了」と言ったときには、r1234 でコミットした実装した機能 X の変更と言う意味になります。どの意味で使用しているかをはっきりさせるために、2 つの用語を使い分けています。

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するようになります。Subversion 用語で、これは `diff -r M:N` となります。

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するようになります。Subversion 用語で、これは `diff -r N-1:N` や `diff -c N` となります。

下部ペインには、選択したリビジョンで変更したすべてのファイルを表示しています。そのため、常に 変更 をコンテキストメニューで表示できます。

4.9.4. 詳細なログメッセージの取得

ログダイアログは以下のような理由で、すべての変更を常に表示するわけではありません。

- 大きなリポジトリでは何百何千もの変更があり、それをすべて取得するには非常に長い時間がかかります。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止める**　チェックボックスにチェックがあると、リポジトリのどこか別の場所にコピーされたファイルやフォルダがあった時点でログの表示が停止します。これはブランチ (やタグ) を参照するときに、ブランチのルートで停止するので便利です。また、ブランチのみの変更を素早く表示できます。

  通常、このオプションにチェック付け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、あなたのお好みの通りになります。

マージダイアログからログ参照ダイアログを表示すると、このチェックボックスは、デフォルトでは常にチェックされています。これはマージがブランチの変更を調べるためで、ブランチのルートを越えて戻るようなことは、このインスタンスでは理解できません。

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の名前を変更している所でログの表示が停止するという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** 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、前回使用したオプションを記憶している、マルチファンクションボタンがあります。三角をクリックすると他のオプションを指定できます。

リビジョンの範囲を指定する場合、**範囲を表示 ...** を使用してください。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。

最新リビジョンからリビジョン 1 へ すべての ログメッセージが見たければ、**すべて表示** 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

ログダイアログには、現在の作業コピーのリビジョンではなく、HEAD　からログを表示するため、作業コピーがまだ取得していない内容のログメッセージを表示することが、しばしばあります。これを明確にするために、作業コピーが持つリビジョンに一致するコミットメッセージを太字で表示します。

フォルダのリビジョンを表示する際、強調されたリビジョンは、そのフォルダの近辺にある最大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。これは大きな作業コピーでは、遅い処理になり、クロールが完了するまでログメッセージが表示されません。この機能を無効にしたり、制限したりしたい場合は、「レジストリの設定」　で説明するように、レジストリキー `HKCU¥Software¥TortoiseSVN¥RecursiveLogRev` をセットしてください。

4.9.6. マージ追跡機能

Subversion 1.5 以降では、属性を利用して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、ブランチ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 コミットしたとしても、トランクでのその機能の開発には、1 コミットしかしていないことになってしまいます。

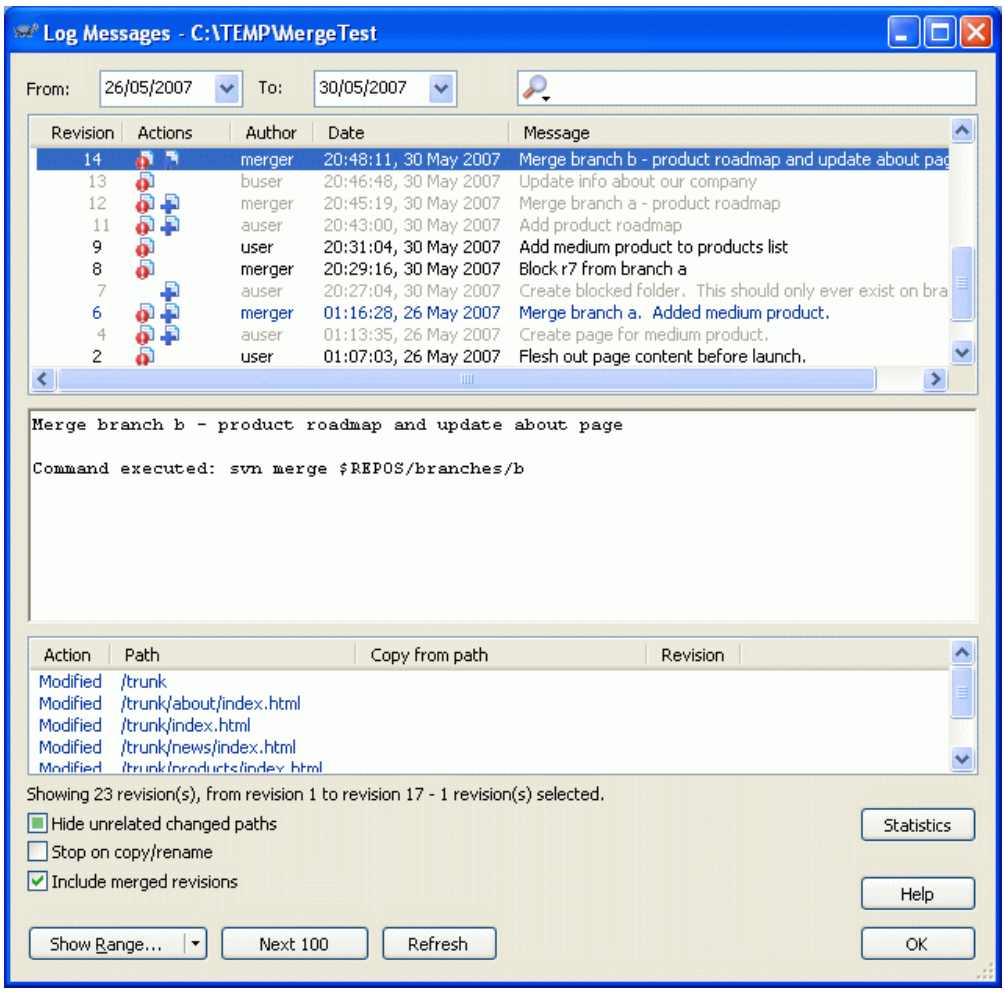


図4.19 マージ追跡リビジョンを表示したログダイアログ

コミットにマージが含まれる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由来のログメッセージの挿入も行います。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、灰色で表示します。

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

リビジョン属性は、各項目の Subversion 属性とはまったく違います。リビジョン属性は、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、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区なるかもしれません。認証のセットアップ方法を忘れてしまったとか……

Subversion では、必要に応じてリビジョン属性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 フックをセットアップしなければなりません。この手順の詳細については、Subversion Book の Hook Scripts [http://svnbook.red-bean.com/en/1.5/svn.reposadmin.create.html#svn.reposadmin.create.hooks] をご覧ください。(訳注 日本語では フックスクリプト [http://subversion.bluegate.org/doc/

ch05s02.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、「サーバ側フックスクリプト」をご覧ください。

一度、必要なフックをサーバにセットアップしてしまえば、任意のリビジョンに対して、作者とログメッセージ (もしくはその他のリビジョン属性) を変更できます。ログダイアログの上部ペインからコンテキストメニューを使用してください。中央ペインのコンテキストメニューを使用しても、ログメッセージを編集できます。

警告

Subversion のリビジョン属性はバージョン管理されないため、そういった属性 (例: `svn:log` コミットメッセージ属性) を変更すると、以前の値を 完全に 上書きしてしまいます。

4.9.8. ログメッセージのフィルタリング

興味あるログメッセージを表示するよう限定するなら、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フィルタコントロールを使用したほうがいいです。開始終了日のコントロールで日付の範囲をしてください。検索ボックスで入力した部分文字列を持つメッセージだけを表示できます。

検索アイコンをクリックして、検索したい情報を選択してください。また正規表現モードも選択できます。通常はシンプルなテキスト検索があれば十分でしょうが、もっと柔軟な検索語句を指定したい場合、正規表現を使用できます。ボックスの上にマウスを持っていくと、正規表現の使い方のヒントをツールチップで表示します。オンラインドキュメントやチュートリアルも http://www.regular-expressions.info/ にあります。フィルタは、フィルタ文字列がログエントリとマッチするかどうかをチェックし、フィルタ文字列と マッチ したエントリのみを表示します。

フィルタ文字列に一致しないログエントリを表示するようフィルタをかけるには、文字列をエクスクラメーションマーク ('!') で始めてください。例えば、フィルタ文字列 `!username` は `username` がコミットしていないエントリのみを表示します。

このフィルタは、すでに取得したメッセージにのみ効果があ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ような動作はしません。

また、下部ペインのパス名を、**変更されたパスで無関係なものは表示しない**　チェックボックスでフィルタできます。関係するパスは、ログを表示するのに使用するパスを含んでいます。フォルダのログを取得する場合、そのフォルダ以下のログを取得することを意味しています。ファイルに対しては、その 1 ファイルのみであることを意味しています。チェックボックスは　3　状態をとります。すべてのパスを表示する、または無関係なものを灰色にする、無関係なものを完全に隠すです。

時には運用手順として、ログメッセージを特定のフォーマットにする必要があるかもしれません。これは変更点を説明するテキストが、上部ペインに表示する短縮サマリに表示されないということです。`tsvn:logsummary`　属性を使用して、上部ペインに表示しているログメッセージの一部を抽出できます。この属性の使い方は、「TortoiseSVN のプロジェクト属性」をご覧ください。

リポジトリブラウザ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　の属性にアクセスできることに依存しているため、チェックアウトした作業コピーを使用するときにだけ、結果を目にするでしょう。リモートから属性を取得するのは遅い操作ですので、作動中のこの機能をリポジトリブラウザから目にすることはないでしょう。

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、その他といった情報です。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

このページでは自分で検討できるように、特定の期間やリビジョン番号で、最小・最大・平均といった数値を提供します。

4.9.9.2. 「作者によるコミット」ページ

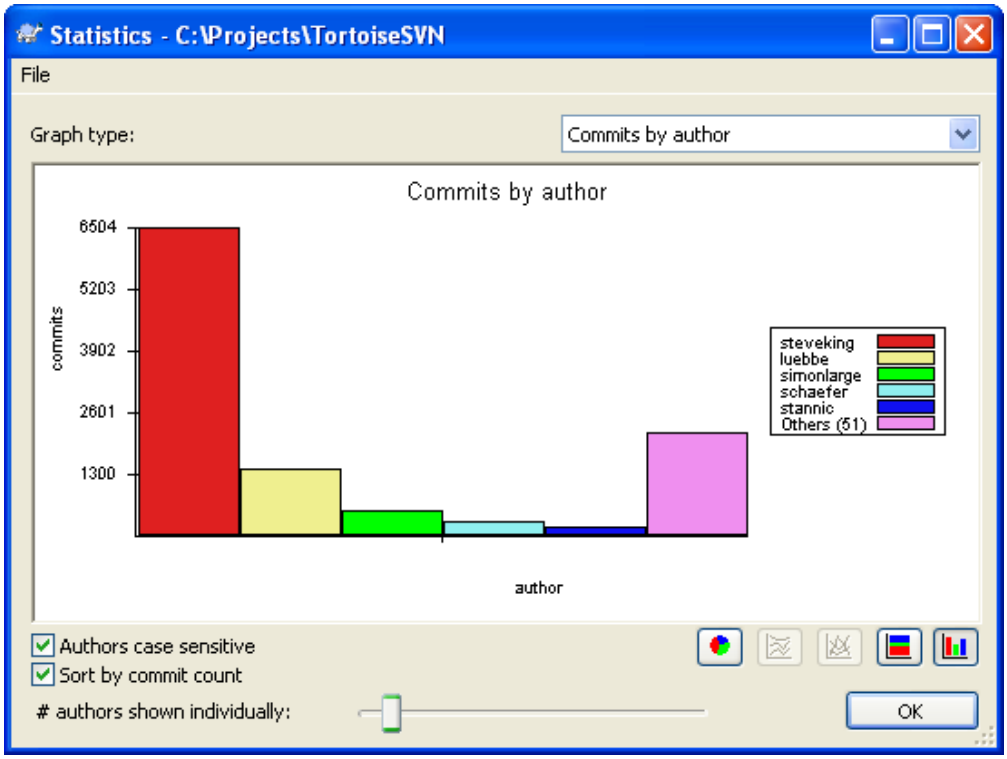


図4.20 「作者によるコミット」ヒストグラム

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

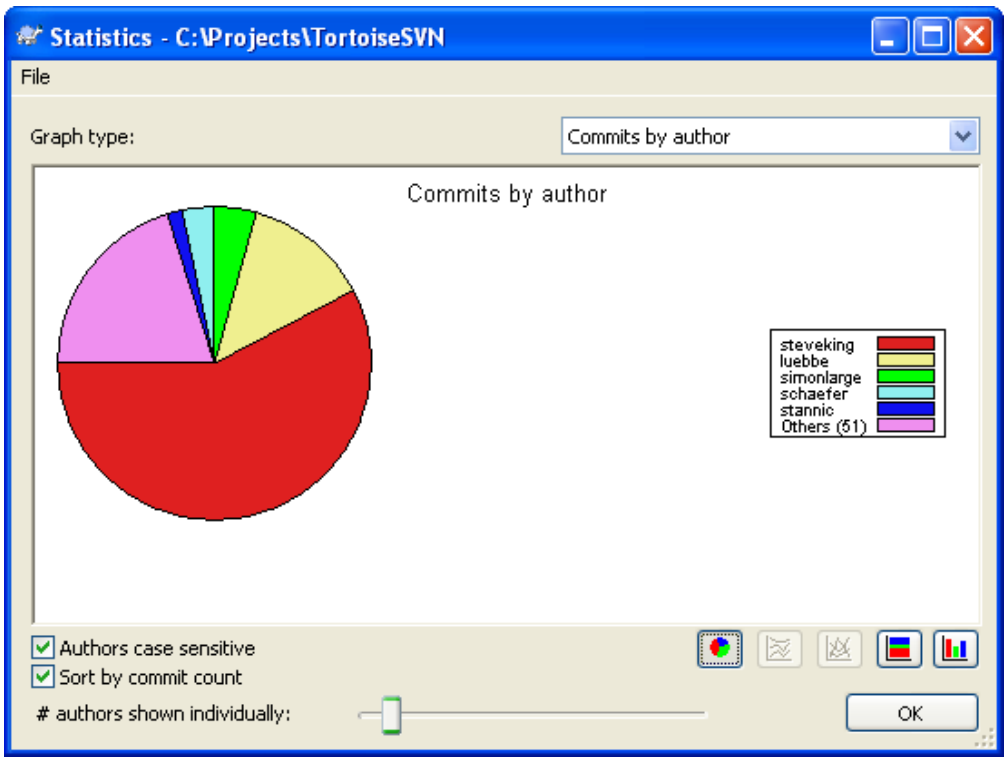


図4.21「作者によるコミット」円グラフ

少数の主要な作者と、多数のこまかな協力者がいるところでは、小さなセグメントが多くなり、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を使用してしきい値 (全コミットにおけるパーセンテージ) を設定し、活動状況がしきい値以下の人を その他 カテゴリにします。

4.9.9.3.「日毎のコミット数」ページ

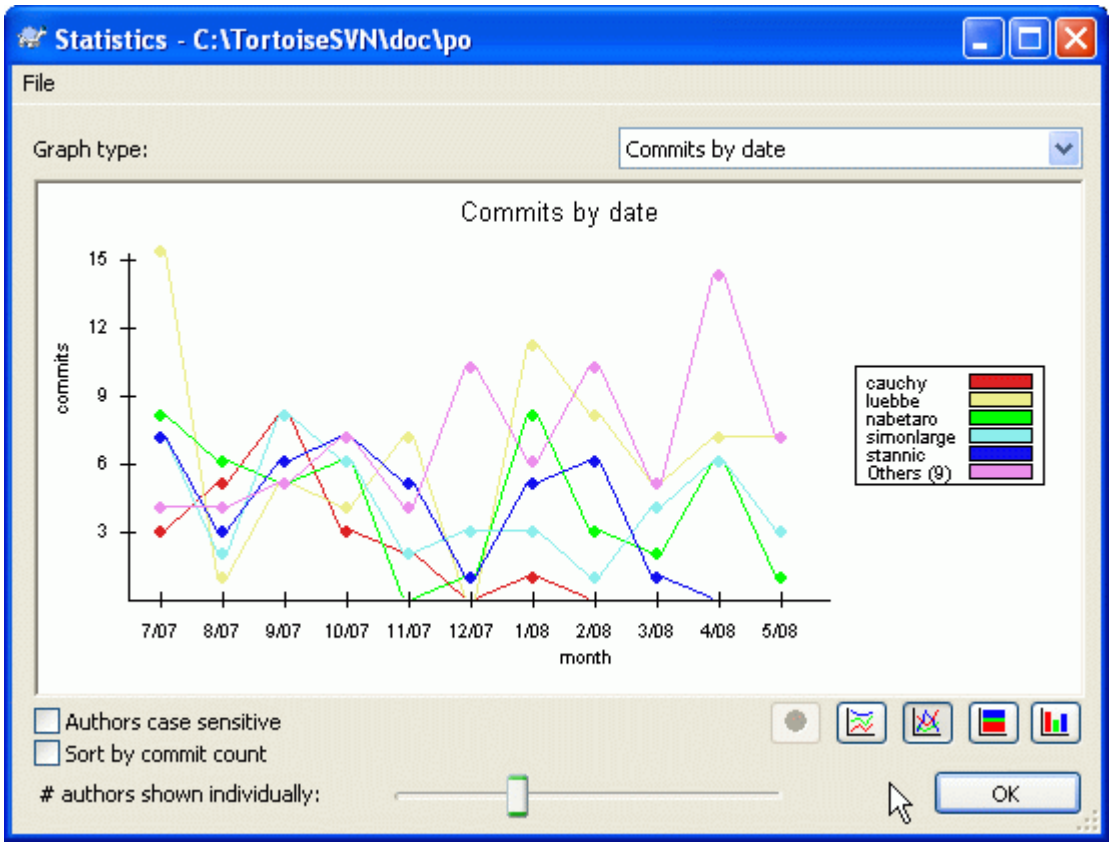


図4.22「日毎のコミット数」グラフ

このページでは、コミット回数 と 作者から、プロジェクトの活動状況をグラフで表示します。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。

複数の作者がいる場合、グラフの線がたくさん表示されます。ここで 2 通りの表示があります。通常 は、作者の活動状況ごとに表示し、積み重ね は、作者の活動状況を積み重ねて表示します。後者では、グラフを読みやすくするため、線が交差するのを避けますが、作者ごとに表示した方が見やすいでしょう。

デフォルトでは、分析する際、ユーザ名の大文字と小文字を区別します。そのため、PeterEgan と PeteRegan というユーザは別人として扱われます。しかし多くの場合、ユーザ名の大文字と小文字を区別しておらず、時々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制御してください。

この統計はログダイアログと同じ期間をカバーするの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

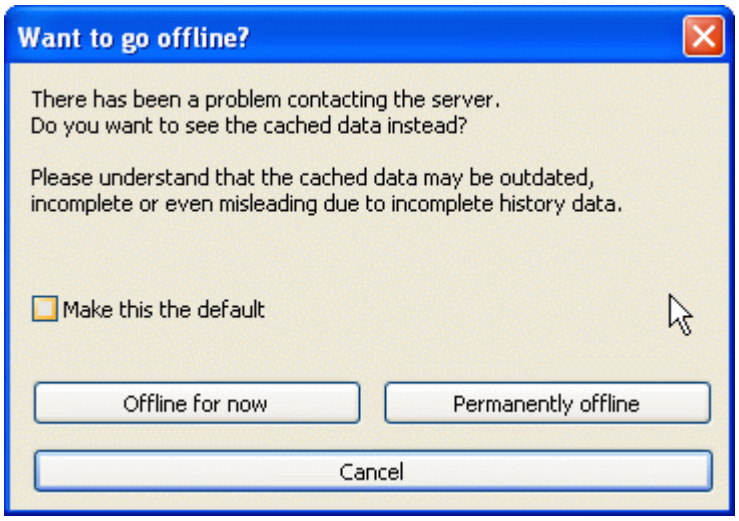


図4.23 オフライン化ダイアログ

サーバに到達せず、ログキャッシュを有効にしている場合、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利用できます。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 種のオプションがあります。

オフラインにする
: 完全に現在の操作をオフラインモードにしますが、ログデータが要求された次の回に、リポジトリに再試行します。

常にオフライン
: リポジトリ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、選択した内容を常に使用します。ここで設定したとしても、TortoiseSVN → Settings 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サーバ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。これは現在表示し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. 差分の参照

プロジェクト開発を進める上でもっとも共通な需要のひとつに、どのように変更したかを確認するということがあります。同じファイルの2つのリビジョン間や、2つの別のファイルの違いを確認したくなるはずです。TortoiseSVN には、テキストファイルの差分を表示する TortoiseMerge という内蔵ツールがあり、画像ファイルの差分を見るのに TortoiseIDiff というツールもあります。もちろん、お好み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

手元の変更
: 作業コピー内の あなた がどんな変更を行ったか確認したければ、エクスプローラのコンテキストメニューでTortoiseSVN → 差分 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たい場合や、(トランクで作業していて) 指定したブランチの変更点を見たい場合、エクスプローラのコンテキストメニューを利用できます。Shift キーを押したままファイルを右クリックし、TortoiseSVN → URL と差分 を選択してください。続くダイアログでは、ローカル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、おそらく2つのタグや、ブランチ・タグとトランクを選択して diff を取るのに、リポジトリブラウザも利用できます。そこでのコンテキストメニューには、リビジョンの比較 があり、これで比較できます。「フォルダの比較」 をご覧ください。

以前のリビジョンからの差分
: 特定のリビジョンと作業コピー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てください。関心のある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見なせる自分の作業コピーとの差分を見る場合、単にファイルを右クリックし、TortoiseSVN → 以前のバージョンと差分 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) 最後にコミットした日時と、作業中の BASE との diff を実行します。これにより、現在の作業コピーにある状態に持っていった最後の変更点を参照できます。作業コピーにあるものよりも新しい変更点は表示されないでしょう。

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、比較したい 2 つのリビジョンを選択して (通常 Ctrl を使用して) ください。それから、コンテキストメニューから リビジョンを比較 を選択してください。

  フォルダ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-Diff 出力 (GNU patch 形式) を使用できます。これには文脈内の数行の変更点しかありません。視覚的なファイル比較よりも読むのが大変ですが、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 Show Differences as Unified-Diff 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・フォルダと URL 間の差分
: 作業コピーの 2 つの異なるファイルや、Subversion リポジトリにあるファイルの差分を確認する場合、エクスプローラで直接両方選択してください。その後、Shift キーを押しながら右クリックし、コンテキストメニューを表示してください。TortoiseSVN → URL と差分 を選択してください。作業コピーのフォルダについても同じことです。TortoiseMerge はパッチファイルと同様に (それぞれを見られる、変更のあったファイルの一覧として) 差分を表示します。

注釈履歴の差分
: 差分だけでなく、作者・リビジョン・変更日時を見たい場合、リビジョンログダイアログより差分と注釈履歴を組み合わせることで行えます。詳細は、「注釈履歴の差分」をご覧ください。

フォルダ間の差分

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の比較を行えます。「外部 Diff/Merge ツール」で利用できるツールをご紹介します。

サードパーティの diff 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 を押して代替ツールを使用できます。その他の diff 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りませんが、コードの意味には変更はありません。以下のオプションでは、比較や差分適用の際の変更点のとり方を管理します。この設定は、**マージ** ダイアログや **注釈履歴** ダイアログに対して、TortoiseMerge の設定と同様に作用します。

改行コードを無視する 改行コードしか差異のない変更は無視します。

空白を比較する インデントや行内の空白の追加・削除を変更点に含みます。

空白の変更を無視する 空白の数や種類しか差異のない変更は無視します。例: インデントの変更やタブからスペースへの変更など。以前何も無かった箇所への空白の追加や、空白の削除は、変更として表示します。

すべて空白を無視する 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、常に差分に含めます。

4.10.3. フォルダの比較

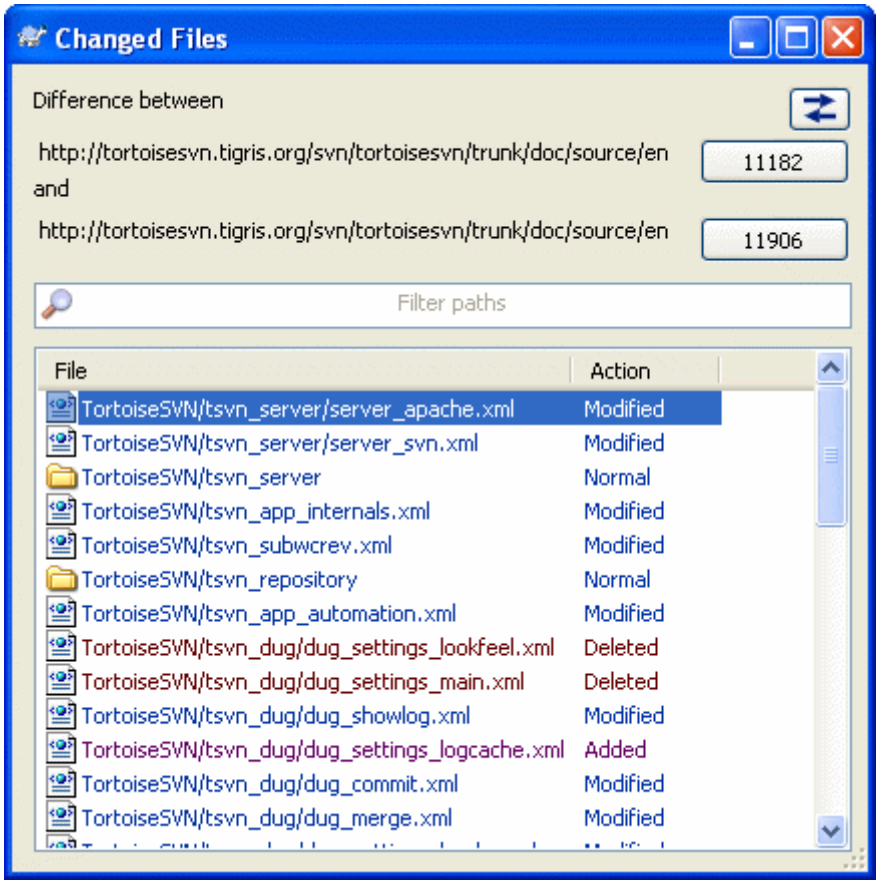


図4.24 リビジョンの比較ダイアログ

リポジトリブラウザにある 2 つのツリーを選択したり、ログダイアログにあるフォルダのの 2 つのリビジョンを選択すると、**コンテキストメニュー → リビジョンを比較** を行えます。

このダイアログは、変更されたすべてのファイルを表示し、コンテキストメニューから個々に比較・注釈履歴取得を行えます。

変更したツリー をエクスポートできます。これは、他の誰かがプロジェクトのツリー構造を必要しているけれども、変更したファイルのみ必要としている場合に便利です。この操作は変更したファイルにのみ有効ですので、興味のあるファイル (たいていすべて) を選択する必要があります。その後、**コンテキストメニュー → 選択をエクスポート...** としてください。変更したツリーを保存する場所を指定することになります。

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...** として、変更したファイルのリストもエクスポートできます。

ファイルの一覧 と 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** を用いて行えます。

最上部のボタンは、比較の方向を変更します。AからBへの変更点を表示でき、希望によりBからAにでも表示できます。

リビジョン番号があるボタンは、リビジョン範囲を変えるのに使用できます。範囲を変える際、2 つのリビジョン間の差異に自動で更新します。

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。シンプルなテキスト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) たくさんありますが、画像ファイルにどんな変更が加えられたかを知るツールも欲していました。ですから TortoiseIDiff を作りました。

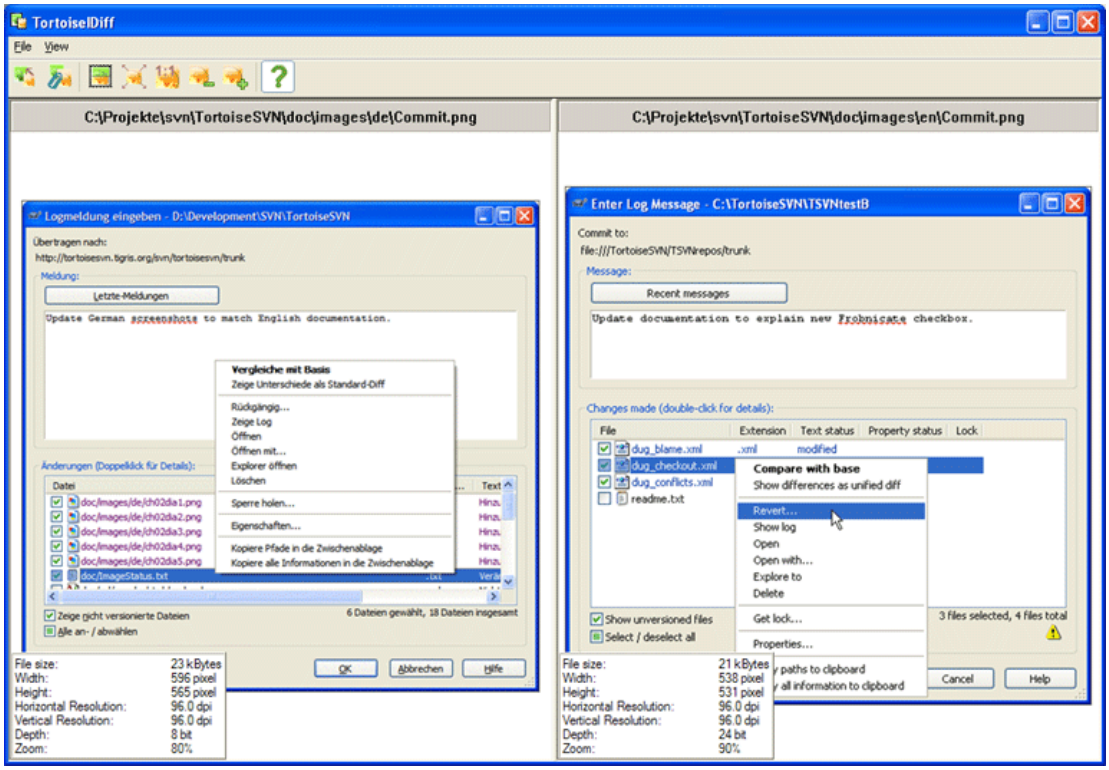


図4.25 画像差分ビューア

いずれかのよくある画像フォーマットで TortoiseSVN → Diffとすると、画像の差分を取る TortoiseIDiff を起動します。デフォルトでは画像を横に並べて表示しますが、表示メニューやツールバーで上下に表示したり、お好みなら画像を重ねて表示し、ライトボックスを使っているかのように見せかけることもできます。

当然、画像をズームイン・ズームアウトしたり、平行移動もできます。単に左ドラッグでも、イメージを平行移動できます。Link images together オプションを選択しておくと、視点変更コントロール (スクロールバー、マウスホイール) は両方の画像をリンクして動かします。

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します。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、画像タイトルバーの上にマウスを持っていくと表示されます。

イメージをオーバーレイしている場合、上部のスライダで、画像の相対的な輝度 (アルファブレンディング) を制御できます。スライダ上のクリックした箇所で透明度を直接制御できますし、スライダをドラッグしてインタラクティブに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で透明度を 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、秒ごとに透明度を自動で交互に切り替えます。複数の小さな変更を見るのに便利でしょう。

ブレンディングではなく差分を確認したい場合もあります。各リビジョンにプリント基板の画像イメージがあり、どのトラックが変更されたかを見たい場合などです。アルファブレンディングモードを無効にすると、各ピクセルの色を XOR して差分を表示します。変更のない部分は真っ白になり、変更点だけ色がつきます。

4.10.5. 外部 Diff/Merge ツール

私たちが提供するツールがニーズに合わなかったら、たくさんあるオープンソース・商用のツールの中から、利用できるものを試してみてください。みんなそれぞれいいところがありますし、この一覧は完全なものではありません。が、検討するかもしれないものを以下に挙げます。

WinMerge
: WinMerge [http://winmerge.sourceforge.net/] は、ディレクトリも扱えるすばらしいオープンソース差分ツールです。

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] から取得してください。

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

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 diff/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 diff 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の両方に対応した、diff/マージ用に便利な商用ツールです。マージ時に 3 方向の比較を行い、関数の順番を変更した場合に使用する同期リンクを持っています。Araxis [http://www.araxis.com/merge/index.html] からダウンロードしてください。

SciTE
: このテキストエディタには unified diff の文法に色が付いていて、読みやすくなっています。Scintilla [http://www.scintilla.org/SciTEDownload.html] からダウンロードしてください。

Notepad2
: Notepad2 は Windows 標準のメモ帳プログラムを置き換えるように設計されていて、Scintilla のオープンソースエディットコントロールを基にしています。unified diff を見るのによいのと同様、ほとんどの作業で Windows のメモ帳よりも便利です。こちら [http://www.flos-freeware.ch/notepad2.html]から自由にダウンロードしてください。

以上のツールを使用するよう TortoiseSVN をセットアップするには「外部プログラムの設定」をご覧ください。

4.11. ファイルやディレクトリの追加

図4.26 バージョン管理外のファイルでのエクスプローラ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 としてください。

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ます。他の開発者にもそのファイルやディレクトリを有効にするには、まず作業コピーをコミットしてください。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を起こしません。

複数の追加

すでにバージョン管理下にあるフォルダで追加コマンドも行えます。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、以下のようにドラッグ&ドロップで操作できます。

1. 追加するファイルを選択します。
2. それを作業コピー内の新しい場所に 右ドラッグ してください。
3. 右マウスボタンを放します。
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する を選択してください。そのファイルを作業コピーにコピーし、バージョン管理に追加します。

作業コピーにあるファイルを、コミットダイアログへ単に左ドラッグ&ドロップしても追加できます。

ファイルやフォルダ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、追加の取り消しができます。

4.12. ファイル・フォルダのコピー・移動・名前変更

既存ファイルを、リポジトリ内の別プロジェクトに持っていかねばならないことがしばしばあり、単純にコピーしたくなると思います。既に述べたように、単純にコピーした上で、そのファイルを追加することもできますが、それまでの履歴を失ってしまいます。また、後でオリジナルファイルにバグ修正を行った場合、Subversion 内で新しいコピーがオリジナルと関連付いている場合に限り、自動的に修正をマージできます。

ファイルやフォルダを、もっとも簡単に作業コピーからコピーする方法は、右ドラッグメニューを使うことです。ファイルやフォルダを、ある作業コピーから他の作業コピーや同じフォルダへ 右ドラッグ すると、マウスのボタンを放すと同時にコンテキストメニューが現れます。

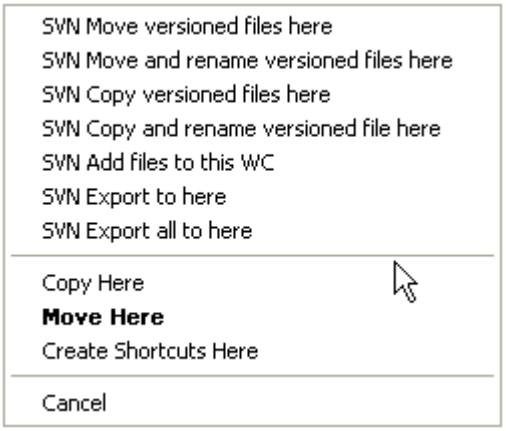


図4.27 バージョン管理下のディレクトリに対する右ドラッグメニュー

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した。必要なら同時に名前の変更もできます。

作業コピーにあるバージョン管理下のファイルや、ふたつの作業コピーの間で、慣れたカット & ペーストでもコピーや移動を行えます。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 を行ってください。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は、TortoiseSVN → 貼り付け を利用し (注意: Windows 標準の 貼り付け ではありません)、新しい作業コピーの場所にコピーや移動できます。

TortoiseSVN → ブランチ/タグ を利用して、ファイルやフォルダ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 → リビジョンから分岐／タグを生成 を利用して、ログダイアログにあるファイルやフォルダ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を用いて、お好みの場所に配置したり、リポジトリから直接作業コピーにコピーできます。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」をご覧ください。

リポジトリ間のコピーはできません

リポジトリ内 のファイルやフォルダはコピーできますが、TortoiseSVN を用いて履歴を保持したまま、リポジトリから別のリポジトリへコピーや移動はできません。リポジトリが同じサーバにあったとし

ても不可能です。できるのは、現在の状態の内容をコピーし、別のリポジトリに新しい内容として追加することだけです。

同じサーバの 2 つの URL が、同じリポジトリか違うリポジトリかがはっきりしない場合、リポジトリブラウザを使用してひとつ URL を開き、そのリポジトリルートがどこにあるか探してください。どちらの場所も、ひとつのリポジトリブラウザウィンドウにある場合、それは同じリポジトリにあります。

4.13. 無視するファイルとディレクトリ

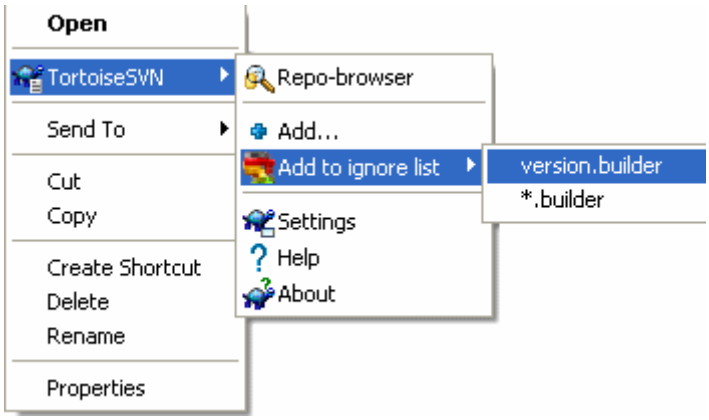


図4.28 バージョン管理外のファイルでのエクスプローラコンテキストメニュー

ほとんどのプロジェクトで、バージョン管理下にあるべきでないファイルやフォルダがあることでしょう。コンパイラが作成したファイル、*.obj, *.lst や、もしかしたら実行ファイルを格納する出力フォルダもそうでしょう。変更をコミットの際、TortoiseSVN はコミットダイアログにバージョン管理外のファイルを表示します。もちろんこの表示を off にできますが、そうすると、新しいソースファイルの追加を忘れるかもしれません。

こういった問題を避ける最良の方法は、プロジェクトの無視リストに、派生ファイルを追加することです。この方法だと、派生ファイルが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外のファイルは現れるままになります。

バージョン管理外のファイルで 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か、同じ拡張子を持つファイルすべてかを選択するサブメニューを表示します。複数のファイルを選択すると、サブメニューを表示せず、指定したファイル・フォルダを追加します。

無視リストから項目を削除したい場合、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。フォルダの svn:ignore 属性に直接アクセスすることもできます。これにより、ファイル名展開にもっと一般的なパターンが指定ができます。属性を直接指定することについては、「プロジェクト設定」 をご覧ください。無視パターンは行ごとに区切らなければならないことをご承知おきください。空白で区切ったパターンは動作しません。

共通無視リスト

ファイルを無視する別の方法は、共通無視リスト に追加することです。共通無視リストはクライアントの属性ということが、大きく違います。これは 全 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。一般的には、可能なら svn:ignore 属性を使用する方がよいです。指定し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適用されるからです。詳細は「一般設定」をご覧ください。

バージョン管理下項目の無視

バージョン管理下のファイルやフォルダは無視できません。これは Subversion の機能です。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、ファイル名展開を使用できます。これは本来 Unix でファイルを指定するのにメタキャラクタをワイルドカードとして使用するテクニックです。以下の文字が特別な意味を持ちます。

*
: 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一文字にマッチします。かっこ内にある「-」で繋がった 2 つの文字は、辞書的にその 2 つの文字の間にある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。

パターンマッチは大文字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、大文字小文字に関わらず `*.tmp` を無視するには、`*.[Tt][Mm][Pp]` といったパターンで指定できます。

展開の公式定義が必要な場合、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] で手に入ります。

共通無視リストにはパスを含めません

パターンにパス情報を含めるべきではありません。パターンマッチングは、単純なファイル名やフォルダ名に対して使用することを想定されています。`CVS` フォルダをすべて無視したい場合は、無視リストに、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` フォルダにある `tmp` フォルダをすべて無視したいけれども、`doc` フォルダにあるものは無視したくない場合は、共通無視リストではなく `svn:ignore` 属性を使用するべきです。この用途に共通無視リストを使用する、信頼性のある方法はありません。

4.14. 削除・移動・名前の変更

CVS と違い、Subversion はファイル・フォルダの名前の変更や移動ができます。そのため TortoiseSVN のサブメニューには、削除や名前の変更のメニューエントリがあります。

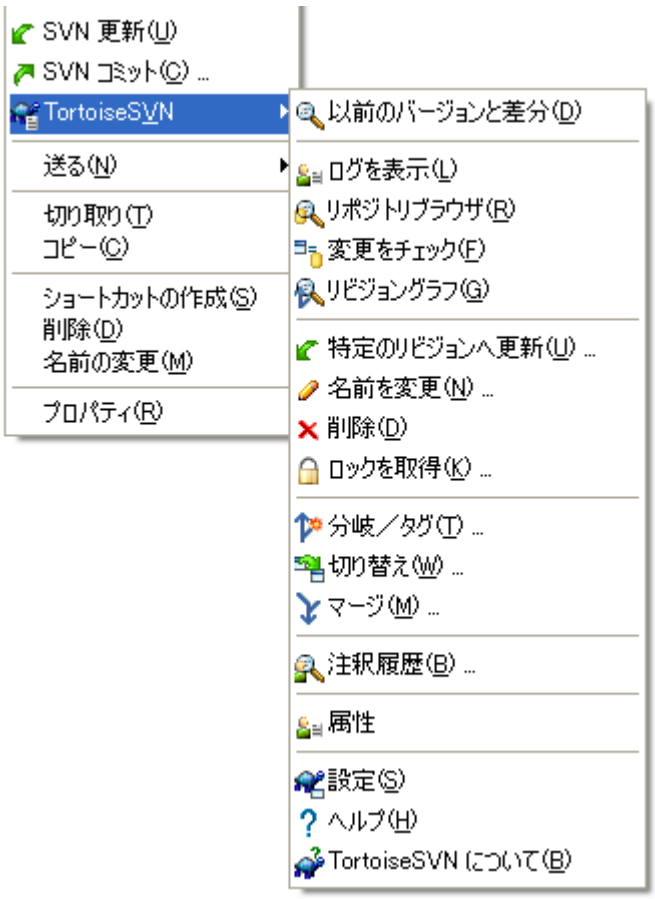


図4.29 バージョン管理下のファイルに対するエクスプローラのコンテキストメニュー

4.14.1. ファイル・フォルダの削除

subversionから、ファイルやフォルダを削除するには、TortoiseSVN → 削除 としてください。

ファイルを TortoiseSVN → 削除 すると、作業コピーからすぐに削除し、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、親フォルダで TortoiseSVN → 元に戻す で元に戻せます。

フォルダを TortoiseSVN → 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、そのフォルダで TortoiseSVN → 元に戻す で元に戻せます。この違いは、Subversion の部分によるもので、TortoiseSVN によるものではありません。

リポジトリから削除したい、でもバージョン管理外のファイル・フォルダとして保持しておきたい場合には、**拡張コンテキストメニュー → 削除 - ローカルファイルを保持** をお使いください。エクスプローラのリストペイン (右ペイン) 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を表示します。

TortoiseSVN コンテキストメニューを使用しないで、ファイルをエクスプローラで削除した場合、コミットダイアログにそのファイルを表示し、コミットの前にバージョン管理から削除するかどうか確認することになります。しかし、作業コピーを更新する場合、Subversion は紛失ファイルを特定し、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必要がある場合は、常に TortoiseSVN → 削除 を使用するようにしてください。そうすると Subversion がファイルを本当に削除したいのかどうか推測する必要がなくなります。

TortoiseSVN コンテキストメニューを使用しないで、フォルダをエクスプローラで削除した場合、作業コピーを破損してしまいコミットできなくなります。作業コピーを更新すると、Subversion はリポジトリから最新のフォルダを取得し、紛失したフォルダと置き換えます。そのため、フォルダを削除する正しい方法は、TortoiseSVN → 削除 を使用することです。

削除したファイルやフォルダの取り消し

削除したファイルやフォルダがあり、その変更をリポジトリにコミットしているなら、通常のTortoiseSVN → **元に戻す** では、元に戻せません。しかし、ファイルやフォルダが完全に失われたわけではありません。ファイルやフォルダを削除したリビジョンがわかっているなら (わからなければログダイアログで探してください) 、リポジトリブラウザを開きそこにリビジョンに切り替えてください。削除したファイル・フォルダを選び、右クリックして **コンテキストメニュー** → **コピー...** を選択してください。コピー先は作業コピーのパスにしてください。

4.14.2. ファイルやフォルダの移動

ファイル・フォルダの名前の変更をその場で行う場合、**コンテキストメニュー** → **名前の変更...** を使用してください。新しい項目名を入力すると完了します。

別のサブフォルダへなど、作業コピー内に移動する場合、右ドラッグ　&　ドロップハンドラを以下のように使用してください。

1. 移動したいファイル・ディレクトリを選択してください。
2. それを作業コピー内の新しい場所に 右ドラッグ してください。
3. 右マウスボタンを放します。
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** を選択してください。

親フォルダでのコミット

名前の変更や移動は、追加に続いて削除として動作するため、名前の変更・移動したファイルの親フォルダに対してコミットしなければなりません。そうでないと、名前の変更・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。名前の変更・移動した部分を除いてコミットしないと、リポジトリの陰に隠れてしまい、共同作業者が更新すると、古いファイルが削除されません。つまり、新旧のファイルの両方がある状態になります。

このとき、フォルダ内のファイルにいずれの変更を加える前に、フォルダの名前の変更をコミットしなければなりません。そうしないと、作業コピーが台無しになってしまいます。

また、リポジトリブラウザを用いて、項目を移動できます。詳細は「リポジトリブラウザ」をご覧ください。

外部参照の SVN 移動はしないでください

`svn:externals` で作成したフォルダに対して、TortoiseSVN **移動** や **名前の変更** コマンドを実行するべきではありません。このアクションは外部参照項目を親リポジトリから削除してしまい、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、移動元と異動先それぞれ親フォルダに対して `svn:externals` 属性を調整してください。

4.14.3. ファイル名の大文字小文字の変換

Windows 上の Subversion で、ファイル名を文字の大小のみの変更はやりにくいです。変更中の短時間でも、両方のファイル名が同時に存在してしまうためです。Windows は、文字の大小を区別しないファイルシステムを使用しており、通常の「名前の変更」コマンドは使用できません。

幸い、ログの履歴を失わずにファイルの名前を変更する、(少なくとも) 2 つの有効な解決法があります。重要なのは Subversion で名前を変更することです。単にエクスプローラで名前を変更すると、作業コピーが壊れてしまいます!

解決法 A) (お勧め)

1. 作業コピーの変更をコミットします。

2. リポジトリブラウザを使用して、ファイル名を UPPERcase から upperCASE へ直接変更します。

3. 作業コピーを更新します。

解決法 B)

1. TortoiseSVN のサブメニューにある名前の変更コマンドで、UPPERcase から UPPERcase_ へ変更します。

2. 変更をコミットします。

3. UPPERcase_ から upperCASE へ名前を変更します。

4. 変更をコミットします。

4.14.4. 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、文字の大小のみが異なる同じ名前のファイル (例: `TEST.TXT` と `test.txt`) がある場合、Windows のクライアントで、親ディレクトリの更新・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 人の人が別々の作業コピーから、同じ名前で文字の大小のみ異なる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、文字の大小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。

この場合どちらかを残し、一方をリポジトリから削除 (または名前変更) するよう決めなければなりません。

2 つのファイルが同じ名前にならないようにする方法

競合するようなチェックインが起きないようにするサーバスクリプトが、http://svn.collab.net/repos/svn/trunk/contrib/hook-scripts/ にあります。

4.14.5. ファイルの名前変更の修復

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を紛失ファイル、新しいファイル名を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りませんから、追跡履歴を失うことになります。

もっと良い方法は、Subversion に実際には名前変更であったと通知することです。これは**コミット** ダイアログと **変更をチェック** ダイアログで行えます。単に古い名前 (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** を使用して、2 つのファイルが名前の変更であったと通知します。

4.14.6. バージョン管理外フォルダの削除

通常、生成した全ファイルといった無視リストは、Subversion が無視します。しかし、もしクリーンビルドを提供するため、無視されるファイルをすべてクリアするとしたらどうでしょうか? 通常、makefile を設定しているでしょうが、makefile のデバッグやビルドシステムを変更する際に、デッキをクリアする方法があると便利です。

TortoiseSVN 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ファイルを削除する...** を使用するという選択肢を提供しています。エクスプローラのリストペイン (右ペイン) で Shift キーを押したまま、フォルダを右クリックすると、拡張コンテキストメニューを表示します。これにより作業コピーにあるバージョン管理外ファイルの一覧ダイアログが現れます。削除するファイルを選択状態・未選択状態にできます。

そのようなファイルを削除した際、ゴミ箱を使用していますので、もし間違ってバージョン管理化にお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

最後に更新したときからファイルに行った変更をすべて元に戻すなら、ファイルを選択してからコンテキストメニューをポップアップするよう 右クリック してください。その後、TortoiseSVN → **元に戻す** コマンドを選択してください。変更し元に戻せるファイルを表示するダイアログがポップアップします。元に戻すファイルを選択し、OK をクリックしてください。

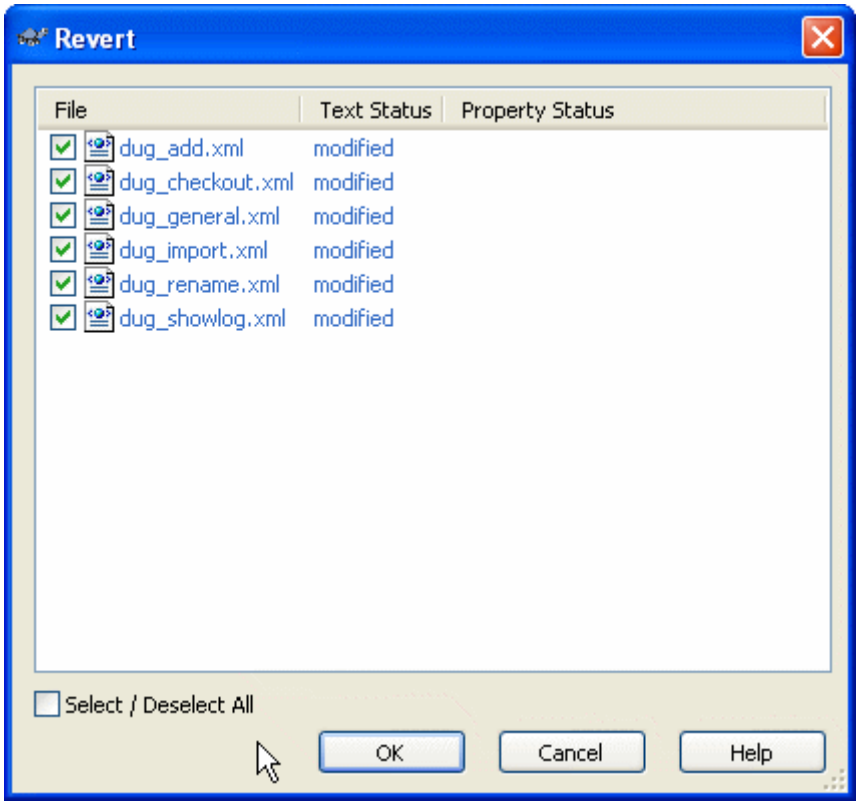


図4.30 元に戻すダイアログ

削除や名前の変更を元に戻したい場合は、削除して消えてしまった親フォルダで右クリックし、「元に戻す」を実行する必要があります。

追加した項目を元に戻す場合、TortoiseSVN → **追加を元に戻す...** のようにコンテキストメニューが現れます。実際には「元に戻す」(revert) なのですが、わかりやすく名前を変更しました。

このダイアログの列は、**変更をチェック** の列と同じ方法でカスタマイズできます。詳細は「こちらの状態とあちらの状態」をご覧ください。

コミットした変更を元に戻す

元に戻す は手元にある変更しか元に戻せません。すでにコミットしてしまった変更は元に戻せないのです。あるバージョンのコミットした変更を元に戻すには、「リビジョンログダイアログ」 にある詳細情報をご覧ください。

「元に戻す」が遅い

変更を元に戻す際、思ったよりも時間がかかっていると思うかも知れません。これは、間違えて元に戻しても、変更を取り出せるように、ファイルの変更した版をごみ箱に送っているからです。しかし、もしごみ箱がいっぱいになると、Windows がファイルを置く場所を探すのに長い時間がかかります。解決法はシンプルです。ごみ箱を空にするか、TortoiseSVN の設定で **元に戻す際にごみ箱を利用する** 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** をフォルダに対して行う必要があります。作業コピーの最上層で行うのがいいでしょう。

クリーンアップには他の便利な側面もあります。ファイルの日付に変更があっても内容に変更がない場合、元のコピーとバイトごとの比較を行わない限り、Subversion は本当に変更があったかどうかわかりません。この状態になったファイルがたくさんあると、状態を取得するのが (ダイアログがたくさん表示されて) 遅くなります。作業コピーでクリーンアップを行うと、「壊れた」タイムスタンプを修正し、状態チェックが早く終わるように戻ります。

コミットタイムスタンプの使用

いくつかの Subversion の以前のリリースでは、**コミット時刻の使用** オプションをチェックしてのチェックアウト時に、タイムスタンプが一致しないというバグがありました。そういった作業コピーをスピードアップするのに、クリーンアップコマンドを使用してください。

4.17. プロジェクト設定

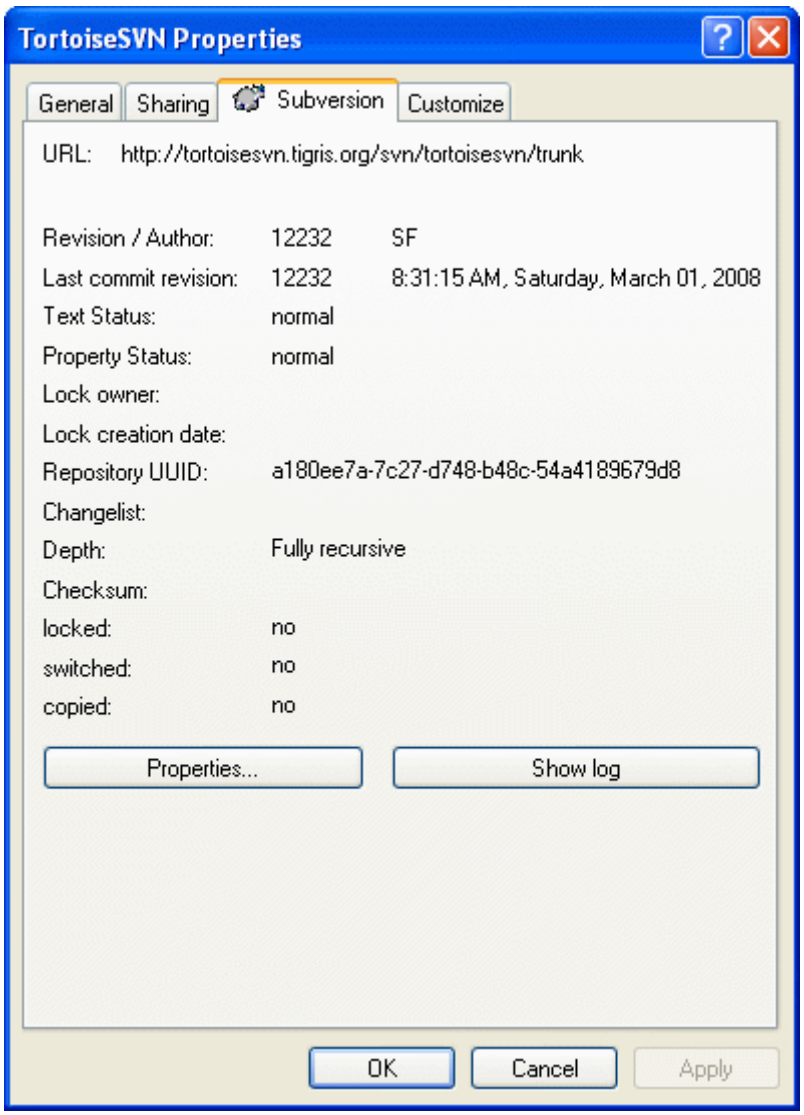


図4.31 エクスプローラプロパティページの Subversion タブ

時にオーバーレイアイコンより詳しい、ファイル・ディレクトリに関する情報が欲しいことがあるでしょう。そういったとき Subversion が提供するすべての情報をエクスプローラのプロパティダイアログから取得することができます。単に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 を選択してください。(注: エクスプローラが提供する通常のプロパティメニューで、TortoiseSVN のサブメニュー内のものではありません!) Subversion 管理下にあるファイル・フォルダの属性ダイアログボックスには、TortoiseSVN が追加した新しい属性ページがあります。ここで、選択したファイル・フォルダに関連する情報を、すべて見ることができます。

4.17.1. Subversion の属性

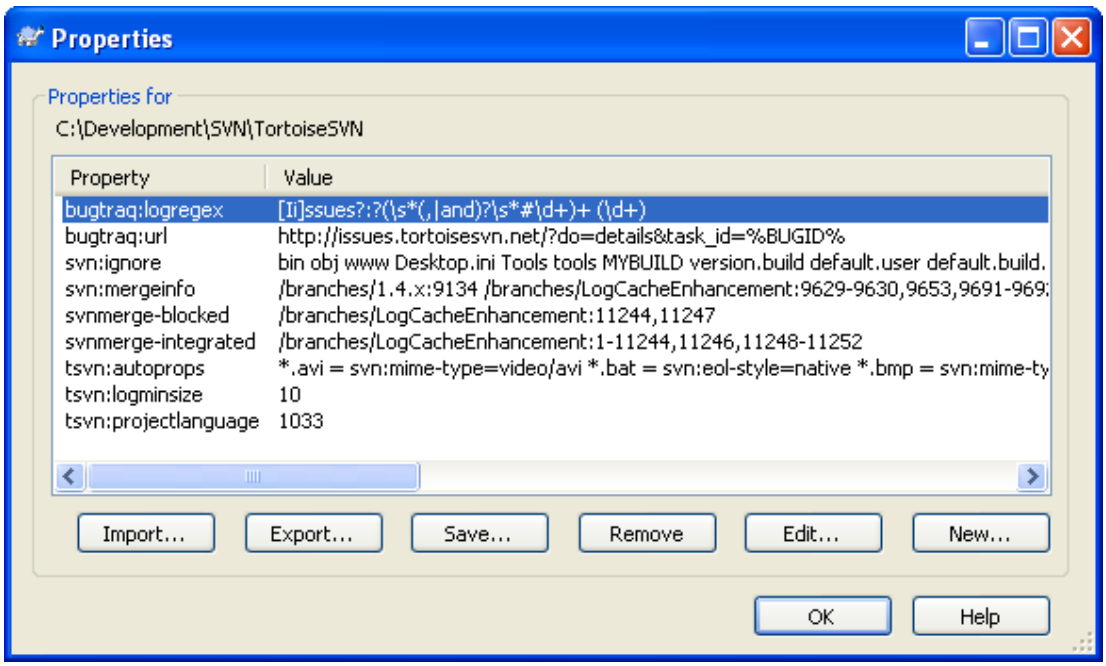


図4.32 Subversion の属性ページ

Windows のプロパティダイアログから、Subversion の属性を確認・設定できますが、さらに TortoiseSVN → 属性 や コンテキストメニュー → 属性 からの TortoiseSVN の状態リストでも確認できます。

自分で定義した属性の他、Subversion で特別な意味のある属性も追加できます。これは `svn:` で始まります。`svn:externals` も、そのような属性です。「外部項目」で外部参照の説明をしていますのでご覧ください。

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。現在キーワードは以下のものをサポートしています。

$Date$
: 最後のコミット日時。作業コピーを更新したときの情報を元にしています。もっと新しい情報を、リポジトリにチェックしにいくようなことはありません。

$Revision$
: 最後のコミットを行ったリビジョン。

$Author$
: 最後のコミットを行った作者

$HeadURL$
: このファイルのリポジトリでのフル URL。

Id
: 前述の 4 つのキーワードの組み合わせたもの。

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/1.5/svn.advanced.props.special.keywords.html] で、キーワードの説明とその有効化・利用法の説明をしています。

Subversion の属性に関する詳細情報は、Special Properties [http://svnbook.red-bean.com/en/1.5/svn.advanced.props.html] をご覧ください。

4.17.1.2. 属性の追加と編集

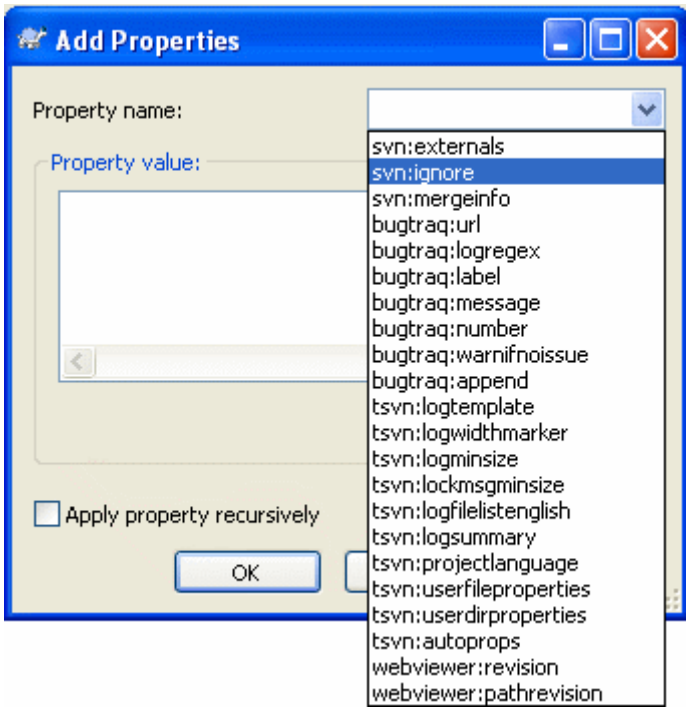


図4.33 属性の追加

新しい属性をセットするには、まず **追加...** をクリックしてください。必要な属性名をコンボボックスから選択・入力し、下のテキストボックスに値を入力してください。無視リストのような複数の値を持つ属性は、複数行で入力してください。OK をクリックして、この属性をリストに追加してください。

一度にたくさんの項目に対して属性を設定する場合は、エクスプローラでファイル/フォルダを選択し、**コンテキストメニュー → 属性** を選択してください。

現在のフォルダ以下にある すべての ファイルやフォルダに対して属性を適用する場合、**再帰** チェックボックスをチェックしてください。

`svn:needs-lock` のようないくつかの属性は、ファイルにしか適用できず、フォルダのドロップダウンリストには属性名が表示されません。その階層内すべてのファイルにそういった属性を適用できますが、属性名は自分で入力しなければなりません。

既存の属性を編集する場合、属性の一覧から編集する属性を選択し、**編集...** をクリックしてください。

既存の属性を削除したい場合、属性の一覧から削除する属性を選択し、**削除** をクリックしてください。

`svn:externals` 属性は、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。

4.17.1.3. 属性のエクスポートとインポート

しばしば、何度も同じ属性のセット (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、属性を簡単にコピーするのに、エクスポート・インポート機能を利用できます。

すでに属性をセットしているファイル・フォルダで、TortoiseSVN → **属性** を使用し、エクスポートしたい属性を選択して **エクスポート...** をクリックしてください。属性の名前と値を保存するファイル名を入力することになります。

属性を適用したいフォルダで、TortoiseSVN → 属性 とし、インポート... をクリックしてください。インポートするファイルを指定することになりますので、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。属性をそのフォルダに、再帰的ではなく追加します。

属性をツリーに対し再帰的に追加する場合、前述の手順の後、属性ダイアログで属性を選び、編集... をクリックしてください。属性を再帰的に適用する チェックボックスをチェックの後、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属性の受け渡し専用で、このファイルを編集する必要はありません。

4.17.1.4. バイナリ属性

TortoiseSVN は、ファイルを使用することでバイナリの属性値を扱えます。バイナリ属性値を読むには、ファイルに 保存... してください。バイナリ値をセットするには、十六進エディタや他の適切なツールを利用して必要な内容を持つファイルを作成し、そのファイルから 開く... としてください。

バイナリ属性はあまり使用されませんが、いくつかのアプリケーションでは便利です。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、属性にサムネイルを格納しておきたいでしょう。

4.17.1.5. 属性の自動設定

ファイルやフォルダをリポジトリに追加した際に、自動的に属性を設定するように 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 設定ファイルを編集して、あなた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、直接編集を行える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、シンプルなテキストファイルです。ここで 2 つの変更をする必要があります。ひとつめは、`miscellany` という見出しのセクションで、`enable-auto-props = yes` という行をアンコメントします。ふたつめは、その下にある、どのファイルタイプに、どの属性を追加するかを定義したセクションを、編集する必要があります。この方法は、一般的な subversion の機能で、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` 属性をフォルダに設定することで、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、作業コピーを更新すると、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、ファイルが追加されたときにのみ、auto-propsを適用することに注意するべきです。すでにバージョン管理下にあるファイルには、auto-props で属性を変更することはありません。

新しいファイルには正しい属性が適用されていると、絶対に間違いないとしたい場合は、必要な属性がセットされていないコミットを拒否するような、pre-commit フックをリポジトリにセットアップするべきです。

属性のコミット

Subversion の属性はバージョン管理されます。属性を変更したり追加した後には、変更をコミットする必要があります。

属性の競合

他のユーザが同じ属性を変更したなどして、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。競合を解消した後にこのファイルを削除してください。

4.17.2. TortoiseSVN のプロジェクト属性

TortoiseSVN は自身が持つ特殊な属性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。この属性が設定されていなかったり、値が 0 の場合、ログメッセージが空でも良くなります。

  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。この属性が設定されていなかったり、値が 0 の場合、ロックメッセージが空でも良くなります。

- tsvn:logwidthmarker はログのフォーマットとして、行の最大長 (通常 80 文字) をそろえる必要があるプロジェクトで使用します。この属性が 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以下のことが起きます。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を配置し、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 はログメッセージを整形するルールを持つプロジェクトで使用します。この属性は複数行の文字列を保持しており、コミットを開始すると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 も使用している場合、テンプレートより長い値を必ずセットするか、そうでなければ保護機構を失うかのどちらかとなります。

- 新しくファイルの追加やインポートした際に、拡張子を元に属性を付加するように、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 つの属性を 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、etc.) 込みで張り付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、各国後で挿入するかを定義できます。この属性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モジュールを使用できます。スペルチェッカをインストールしている場合、この属性でどのスペルチェッカ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。あなたの言語の値は MSDN: Language Identifiers [http://msdn2.microsoft.com/en-us/library/ms776260.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、英語 (アメリカ) は0x0409 や 1033 を入力してください。

- tsvn:logsummary 属性は、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 属性の値は、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。

- 新しい属性を加える際に、コンボボックスから選択するか、任意の属性名を入力できます。プロジェクトでカスタム属性を使用し、その属性をコンボボックスに表示する (属性名の入力ミスを防ぐ) 場合、tsvn:userfileproperties や tsvn:userdirproperties でカスタム属性を作成できます。この属性はフォルダに適用してください。そのフォルダ以下でファイルを作成すると、定義した属性名ごとにカスタム属性を表示します。

tsvn: 属性のいくつかは 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。

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属性を使用します。詳細情報は「バグ追跡システム / 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属性を使用して、いくつかの Web ベースリポジトリブラウザとも統合できます。詳細は「Web ベースリポジトリビューアとの統合」をご覧ください。

フォルダへのプロジェクト属性の設定

以上の 特殊なプロジェクト属性は、動作するシステムの フォルダ に設定しなければなりません。ファイルやフォルダをコミットする際に、属性はフォルダから読み込まれます。属性が見つからない場合、TortoiseSVN はフォルダツリーを上位階層に向かって検索します。これはバージョン管理外フォルダに到達するか、ツリーのルート (例: C:¥) に到達するまで行います。各ユーザが trunk/ からチェックアウトしていて、サブフォルダからチェックアウトしていないことが確認できるのであれば、trunk/ にその属性を設定するだけで十分です。確認できない場合、各サブフォルダに対し再帰的に属性を設定しなければなりません。プロジェクト階層の深い場所の属性設定は、高い階層 (trunk/ に近い場所) の設定を上書きします。

プロジェクト属性 だけ ですが、再帰チェックボックスを使って階層の全サブフォルダに属性をセットできます。このとき全ファイルが対象になるわけではありません。

TortoiseSVN で新しいサブフォルダを作成する際、親フォルダに設定されているプロジェクト属性は、新しいサブフォルダに自動的に追加されます。

注意

TortoiseSVN のプロジェクト属性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。プロジェクト属性は、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。

4.18. 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。全てのユーザが同じレイアウトを保持するには、svn:externals 属性を設定し、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

では /project1 の作業コピーを D:¥dev¥project1 にチェックアウトしましょう。D:¥dev¥project1 フォルダを選択し、右クリック してコンテキストメニューのWindows メニュー → プロパティ+追加...+設定

URL は適切にエスケープされねばならず、そうでなければ動作しません。例えば、前述の 2 番目の例にあるように、各空白を %20 に置換し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。

明確なリビジョン番号の利用

前述のように、外部参照定義のすべてにおいて、明示的なリビジョン番号を使用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、明示的なリビジョン番号を使用するということは、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言った違いに現れます。

古い svn:external の定義

この形式は Subversion 1.5 で導入されました。古い形式で、同じ情報を異なる順番で見ることもできます。新しい形式は、以下で説明する便利な機能を提供するため好まれていますが、古いクライアントでは動作しないでしょう。差異は Subversion Book [http://svnbook.red-bean.com/en/1.5/svn.advanced.externals.html] でご覧になれます。

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をコミットしたときに、外部プロジェクトに行った変更もコミットされてしまいます。

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 属性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。

必要であれば、URL の後にペグリビジョンも指定できます。例えば http://sounds.red-bean.com/repos@19 といった形です。

TortoiseSVN が属性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` 属性を変更しなければなりません。

Subversion 1.6 でのファイル外部項目サポートは不完全

subversion 1.6 では、一度追加してしまうと、`svn:externals` 属性をすべて削除したとしても、外部ファイルを作業コピーから削除できません。ファイルを削除するには、新しい作業コピーをチェックアウトする必要があります。

4.19. ブランチ・タグ付け

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点を隔離することがあります。このラインはブランチとして知られています。ブランチは、コンパイルエラーやバグで開発の本流を混乱させずに、新機能を十分に試すのに使用されます。新機能が十分安定したら、開発ブランチをメインブランチ (トランク) に マージ します。

バージョン管理システムのもう一つの特徴は、特定のリビジョン (リリースリビジョンなど) をマークする能力です。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。このプロセスを タグ付け と呼んでいます。

Subversion には、ブランチ付けやタグ付け用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できます。チープコピーは、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、リポジトリに余分な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

プロジェクトをお勧め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にはとてもシンプルです。

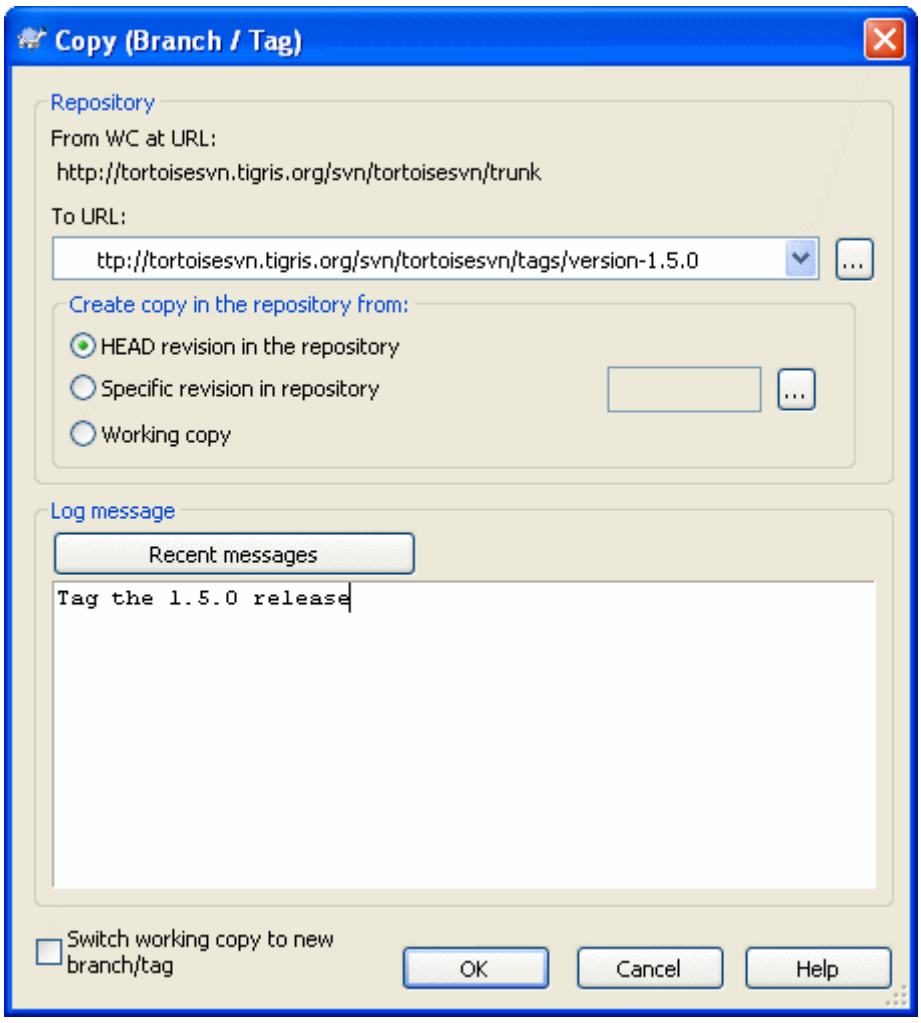


図4.34 ブランチ・タグダイアログ

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

新しいブランチのデフォルトの先 URL は、作業コピーの基準になった元 URL になっています。ブランチ・タグの新しいパスに URL を編集する必要があるでしょう。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、既存のリポジトリ構造を見るのに、リポジトリブラウザを開く右のボタンを押してください。

では、コピー元を選択してください、ここでは 3 種類の選択肢があります。

リポジトリの最新 (HEAD) リビジョン
: 新しいブランチは、リポジトリの最新 (HEAD) リビジョンから直接コピーされます。作業コピーから移されるデータはありませんし、ブランチは非常に高速に作成されます。

リポジトリのリビジョン指定
: 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、リビジョンログを表示する右のボタンを押し、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移されるデータはありませんし、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、手元の作業コピーと同一のコピーになります。作業コピーに対して、以前のリビジョンへ更新したり、手元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。当然そういった複雑なタグの場合、リポジトリに既になければ、作業コピーからリポジトリへ転送が発生するでしょう。

新しく作成したブランチに、自動的に作業コピーを切り替えたい場合、**新しい分岐／タグへ作業コピーを切り替え** チェックボックスを使用してください。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てはいけません。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。

リポジトリに新しいコピーをコミットするのに **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 リポジトリの中 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. チェックアウトするか切り替えるか...

……これは (それほどじゃないですが) 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。:-)

新しく生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。以下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。**最新のリビジョン**　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、手元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かねばならないか、既にコミットしてあるリビジョン (大抵 HEAD) まで、変更を取り消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 エクスプローラでコピーし、**TortoiseSVN → 切り替え...** を使用することで、新しいブランチにすることができます。

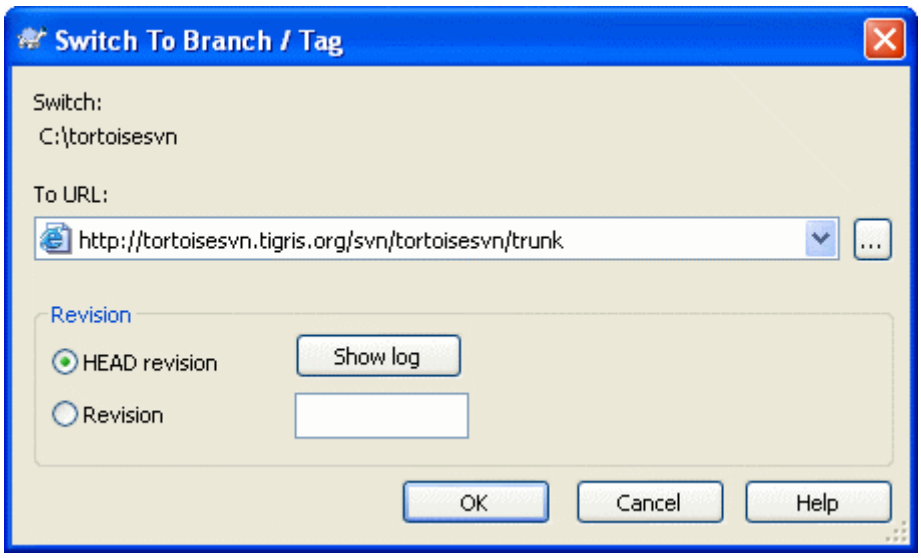


図4.35 切り替えダイアログ

Subversion 自身はタグとブランチを区別しませんが、通常以下のように多少使い分けをします。

- タグは、典型的には、プロジェクトの特定の段階で静的スナップショットを作成するのに使います。通常は開発には使用しません。開発にはブランチを使用します。これが最初に/trunk /branches /tags というリポジトリ構造をお勧めした理由です。タグリビジョンで作業するのは 名案ではありません が、手元のファイルに書き込み保護をかけられませんので、誤って編集してしまうのを止められません。しかし、/tags/ を含むリポジトリパスにコミットしようとしたときに、TortoiseSVN は警告を発します。

- おそらく既にタグ付けしたリリースについて、さらに修正を加える必要があるかもしれません。これを扱う正しい方法は、まずタグから新しいブランチを作成し、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 内でどのように分岐やマージを行っているかを、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うため) 重要です。Subversion book の Branching and Merging [http://svnbook.red-bean.com/en/1.5/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。

一般に変更を加えていない作業コピーへマージするのはいい方法です。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、変更を取り消す必要があるかもしれません。**取り消す** 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

ひとつ以上のリビジョンをブランチに (またはトランクに) 適用したり、その変更点を他のブランチに適用したい時、この方法で行えます。

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、追加チェックを利用します。これはあなた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 に次のことを行うように指示を出しています。「トランクの最新リビジョン [FROM] から ブランチの最新リビジョン [TO] 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ます」 最終結果は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・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/1.5/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

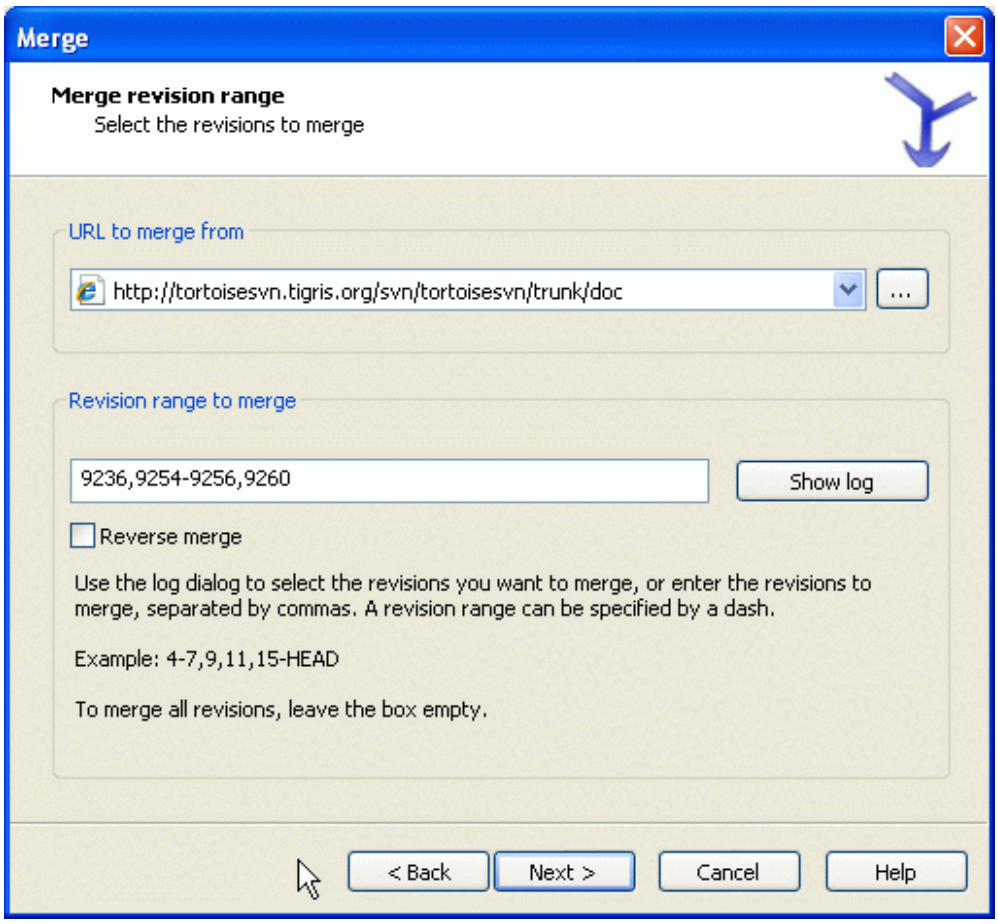


図4.36 マージウィザード - リビジョン範囲の選択

元: フィールドには、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。このブランチから以前マージしていれば、以前に使用した URL の履歴のドロップダウンリストを使うだけです。

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一つ、カ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利用できます。

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、元に戻す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。

他の人が変更をコミットしている場合、HEAD リビジョンを使用するの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 をクリックして「マージオプション」に進んでください。

4.20.2. ブランチの再統合

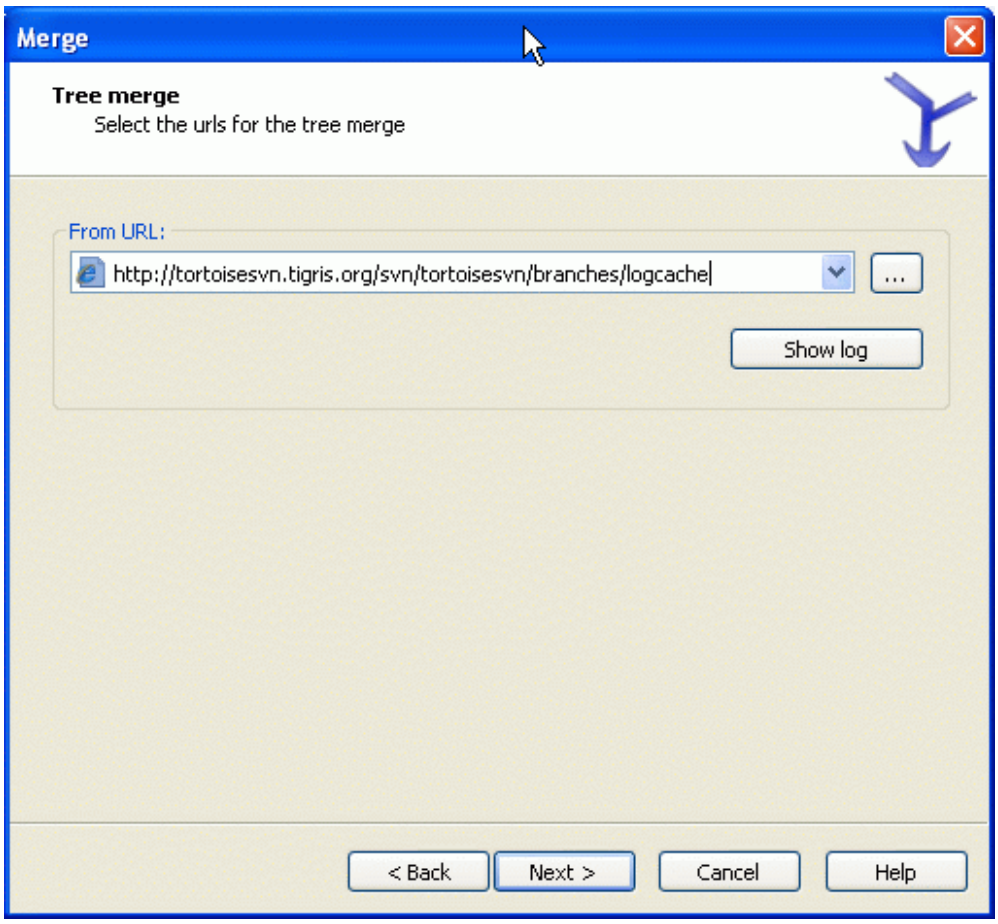


図4.37 マージウイザード - マージの再統合

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 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 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 (まばらなチェックアウトではならない)。ローカルの変更や切り替わった項目、最新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なければなりません (もしくはマージするとマークされなければなりません)。マージす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

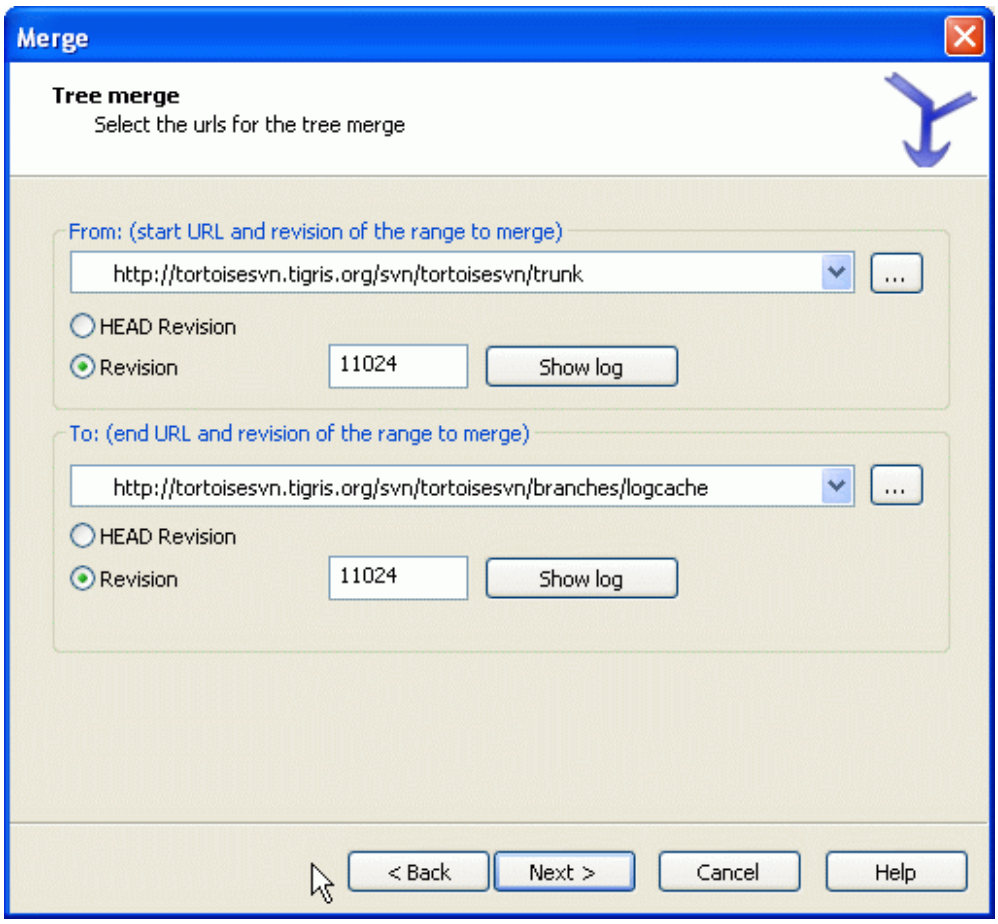


図4.38 マージウィザード - ツリーのマージ

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。

元: フィールドには、トランク の完全なフォルダ 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、リポジトリの参照もできます。

先: フィールドに、機能ブランチの URL を入力してください。

元 リビジョン フィールドと **先 リビジョン** フィールドの両方に、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 リビジョンを使用できます。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使用するだけでしょう。

また、マージする深度も指定できます。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べきかということです。深度については「チェックアウトの深度」で説明しています。デフォルトの深度は **作業コピー** で、既存の深度をそのまま使用します。

大抵、共通の系統が関係する変更点をマージするように、ファイルの履歴を考慮に入れたいと思うでしょう。時には、おそらく関係する (しかしリポジトリにない) ファイルのマージが必要となります。例えば、サードパーティ製ライブラリのバージョン 1 とバージョン 2 を、それぞれ別のディレクトリにインポートしたとします。これは論理的には関係ありますが、Subversion は tarball をインポートしたとしか見えないので、関係があるとはわかりません。この 2 つのツリーの違いをマージしようとするなら、完全に追加してから完全に除去することになります。Subversion が履歴ベース差分ではなくパスベース差分を使用するようにするには、**系統情報の無視** チェックボックスをチェックしてください。このトピックの詳細は、Subversion book の Noticing or Ignoring Ancestry [http://svnbook.red-bean.com/en/1.5/svn.branchmerge.advanced.html#svn.branchmerge.advanced.ancestry] をご覧ください。

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。

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、**マージのみを記録する** 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二つ考えられます。一つは、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、手でコードを修正した後、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。もう一つは、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って、**マージ** ボタンを押すだけになったとします。結果のプレビューを行う場合、**マージのテスト** はマージ操作を行いますが、作業コピーにいずれの変更も加えません。これで実際のマージで変更が入るファイルの一覧を表示し、また競合が発生する範囲も表示します。

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一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。これを理解するには、マージは差分と密接に関連していることを思い出す必要があります。マージ処理は、リポジトリの 2 点間における差異の一覧を生成し、その差異を作業コピーに適用するというように動作します。進行ダイアログは、単純に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、手動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、変更点のマージを繰り返してしまうの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。ただフォルダとフォルダ、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。フォルダを選択してダイアログを開いたらマージするフォルダの　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利用します。

- 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` 属性を格納しませんし、以前のサーバでは、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の詳細は、Subversion の Merge tracking documentation [http://subversion.tigris.org/merge-tracking/index.html] にあります。

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、競合を解消しておきたいでしょう。TortoiseSVN は、マージ競合コールバック 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。

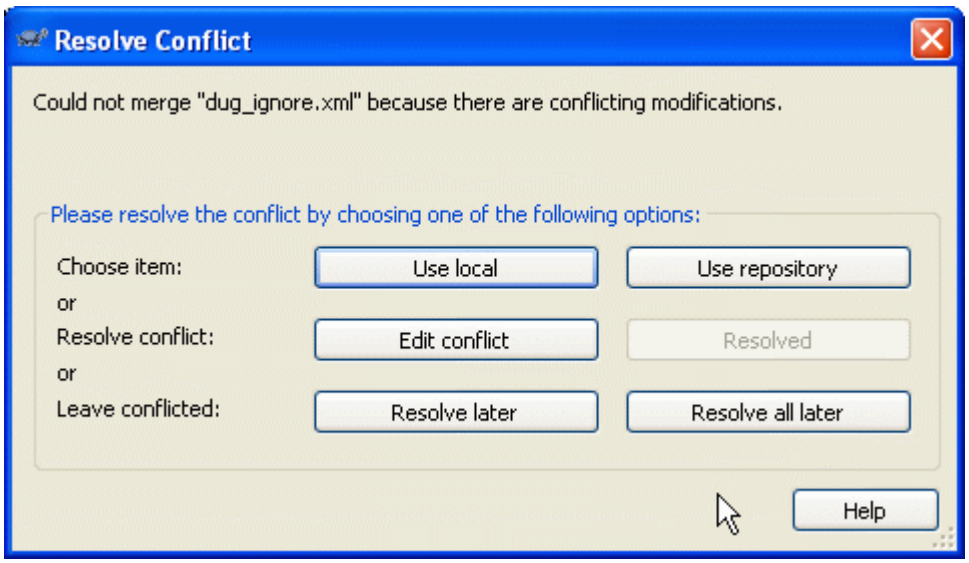


図4.39 マージ競合コールバックダイアログ

マージ中に競合が発生した場合、3 種類の選択肢があります。

1. あなたの行った変更のほうが重要で、リポジトリのものを破棄して手元のバージョンを保持するという判断もあり得ます。もしくはリポジトリのものを支持し、自分のした変更を破棄することもできます。どちらにしろ、マージは一切行いません。どちらかを選ぶことになります。

2. 通常は、競合の内容を調べ、解消することになります。この場合、**競合の編集** を選択してマージツールを起動することになります。結果に満足したら、**問題の解消** をクリックしてください。

3. 最後の選択は、解消を先延ばしにして、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...** を利用できます。

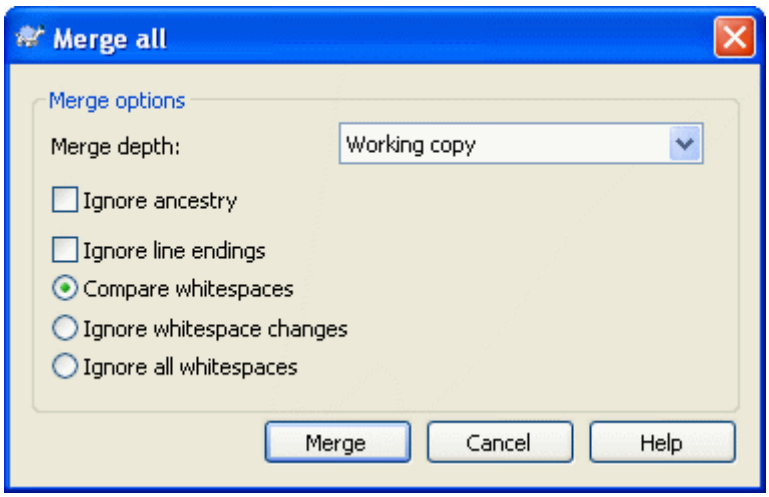


図4.40 マージ再統合ダイアログ

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、マージ追跡を利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・マージ法」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。
- あなたの会社が過去に、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、手元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、あなたしかそのファイルをコミットできなくなります。他のユーザがコミットしようとしても、あなたがロックを解放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。

しかし他のユーザは、あなた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 の属性で `svn:needs-lock` があります。ファイルにこの属性が (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手元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、解放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放することを、ロックの 破壊 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、ロックトークンは手元の作業コピーに作成されます。他の誰かがロックを破壊したりして、食い違いが発生すると、手元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。

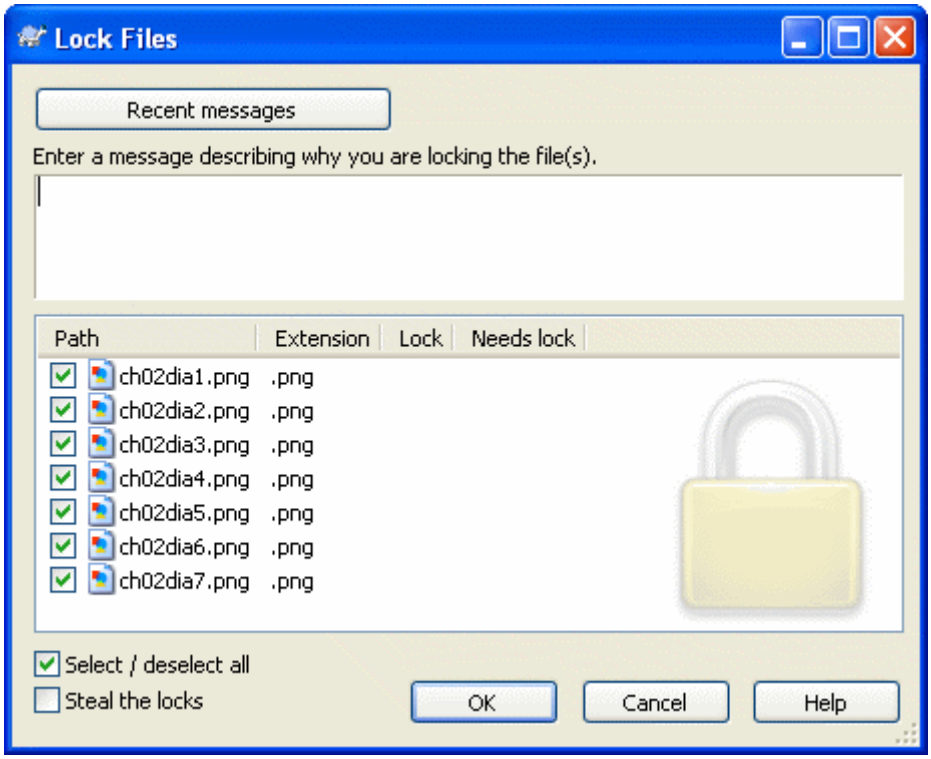


図4.41 ロックダイアログ

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。

フォルダを選択して、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放を忘れないように、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放されます。特定のファイルでロックを解放したくない場合、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。

手動でロックを解放するには、ロックを解除したいファイルを作業コピーで選択し、TortoiseSVN → **ロックを開放** 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、ロックを解放します。フォルダに対してこのコマンドを使用し、再帰的に全ロックの解放を行えます。

4.21.4. ロック状態のチェック

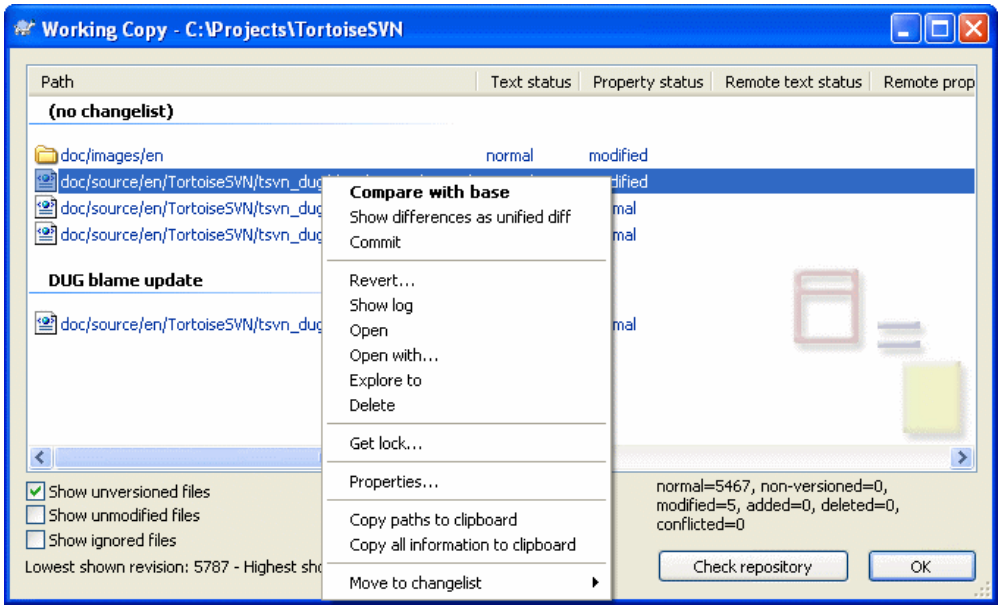


図4.42 変更をチェックダイアログ

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。手元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 (そして誰があなたのロックを破壊したり奪ったかを見るには)、**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、破壊したり横取りしたりするように、ロックを取得したり解放したりもできます。

ロックの破壊・横取りは避ける

ほかの誰かのロックを、断りもなく破壊・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奪った場合、自分がロックを解放すると、自由にあなたのものを上書きできます。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` 属性を設定することです。属性の設定については 「プロジェクト設定」 をご覧ください。この属性を持つファイルは、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をリポジトリに追加した際に、自動的に属性を設定するように Subversion を設定できます。詳細情報は「属性の自動設定」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すべてに通知されます。リポジトリフォルダの `hooks/post-lock.tmpl` に、サンプルフックスクリプトがあります。

フックを利用して、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。

パッチファイルはシンプルな Unified-Diff 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、

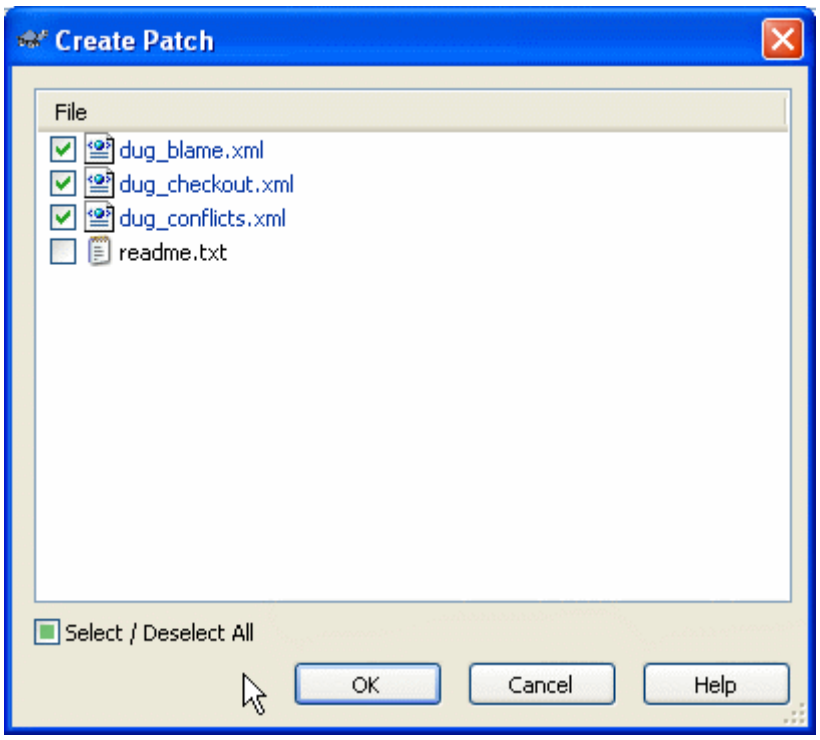


図4.43 パッチ作成ダイアログ

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。

このダイアログの列は、**変更をチェック** の列と同じ方法でカスタマイズできます。詳細は「こちらの状態とあちらの状態」をご覧ください。

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、お好みのファイル名でファイルを保存してください。パッチファイルはお好み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。

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

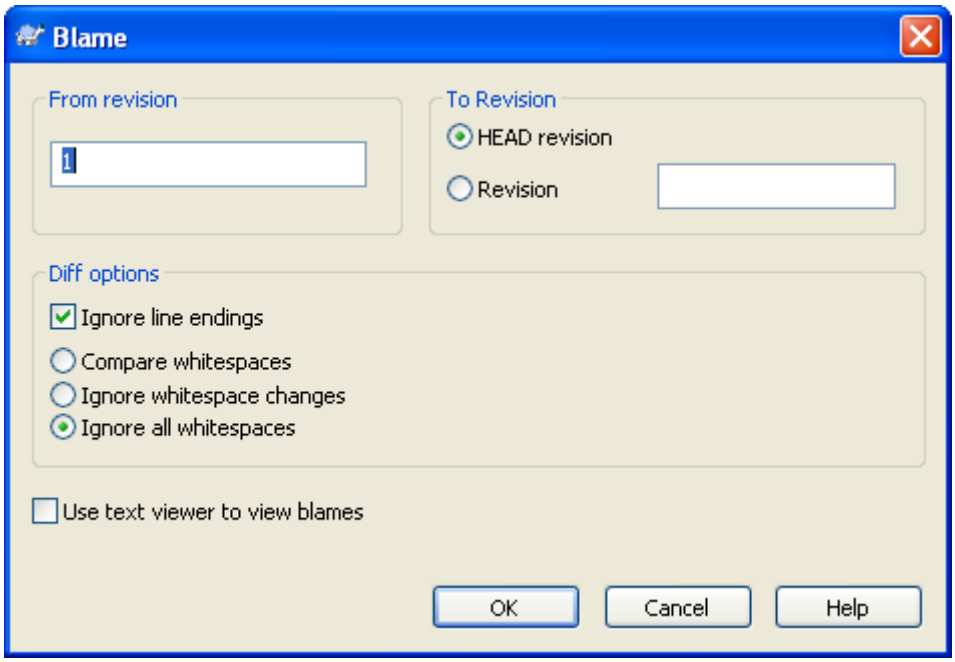


図4.44 注釈ダイアログ

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。

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。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、終了するのに数分かかることに注意してください。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。

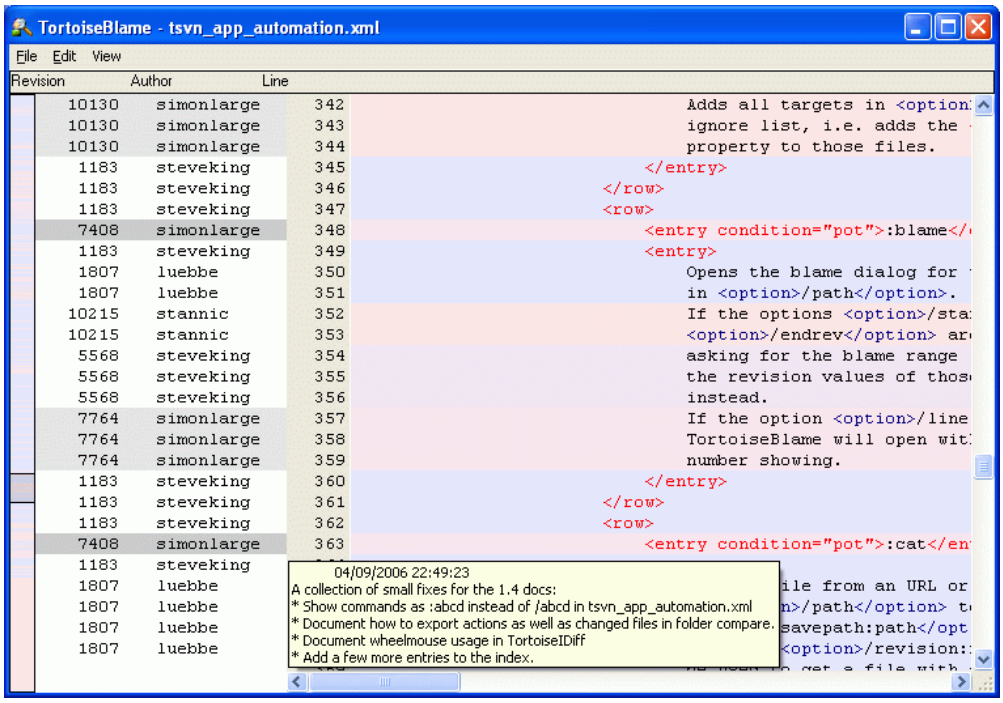


図4.45 TortoiseBlame

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利用可能になります。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。

別のパスからマージした変更がある行でマージ追跡している場合、TortoiseBlame はマージしたところのリビジョンではなく、オリジナルファイルの最終更新のリビジョンと作者を表示します。そのような行は、リビジョンと作者をイタリックで表示します。この方法でマージした行を見ない場合は、**マージ情報を含める** チェックボックスのチェックを外してください。

マージされるパスを確認する場合、**表示 → パスをマージ** を選択してください。

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

注釈レポートの制限に、特定のリビジョンにあるファイルしか見られず、行ごとに最終更新者しか表示できない、というものがあります。時にはどんな変更が行われたのと同じように、誰が変更を行ったかを知りたいこともあるでしょう。ここで必要となるのが、注釈レポートと差分の組み合わせです。

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
: 画面上部のリビジョンを 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、差分ビューアで注釈を比較できます。

変更の注釈履歴
: 上部ペインでリビジョンをひとつ選択し、下部ペイン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、差分ビューアで注釈を比較できます。

作業 BASE 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 → 作業ベースとの比較・注釈履歴を行う** 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中の BASE にあるファイルの注釈データを取得し、差分ビューアで注釈を比較します。

4.24. リポジトリブラウザ

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 があります。エクスプローラ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でリポジトリの構造や状態を確認できます。

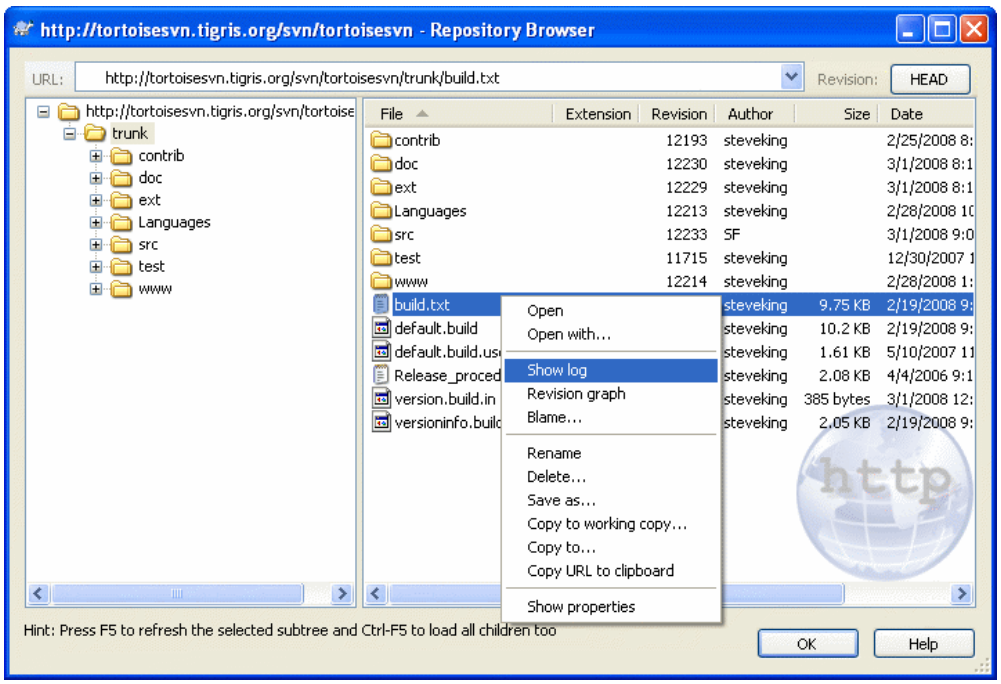


図4.46 リポジトリブラウザ

リポジトリブラウザでは、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は、Windows エクスプローラによく似ており、あなたのコンピュータ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。左ペインでディレクトリツリーを表示し、右ペインで選択したディレクトリの内容を表示します。リポジトリブラウザ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。

Windows エクスプローラのように、右ペインの列見出しをクリックして、並び順をお好みに変更できます。また、エクスプローラのように両方のペインでコンテキストメニューを利用できます。

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、ファイルタイプに応じた規定のビューアか選択したプログラムで開きます。
- あなた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。
- 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。
- ファイルの削除や名前の変更をします。
- 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。
- ファイルの属性を表示・編集します。

フォルダ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。

- そのフォルダのリビジョンログの表示や、そのフォルダ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。
- あなたのハードディスクに、フォルダのバージョン管理外のコピーをエクスポートします。
- あなたのハードディスクに、フォルダ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を作成します。
- リポジトリに直接ファイルやフォルダを追加します。
- フォルダの削除や名前の変更
- 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ォルダをコピーします。
- フォルダの属性の表示・編集
- 比較用にフォルダをマークします。マークしたフォルダ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に対して、unified-diff 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。

右ペインで 2 つの項目を選択する場合、unified-diff 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。

右ペインで複数のフォルダを選択すると、一度にすべてを共通の親フォルダ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。

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をドラッグ & ドロップでも操作できます。エクスプローラからリポジトリブラウザへフォルダをドラッグすると、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

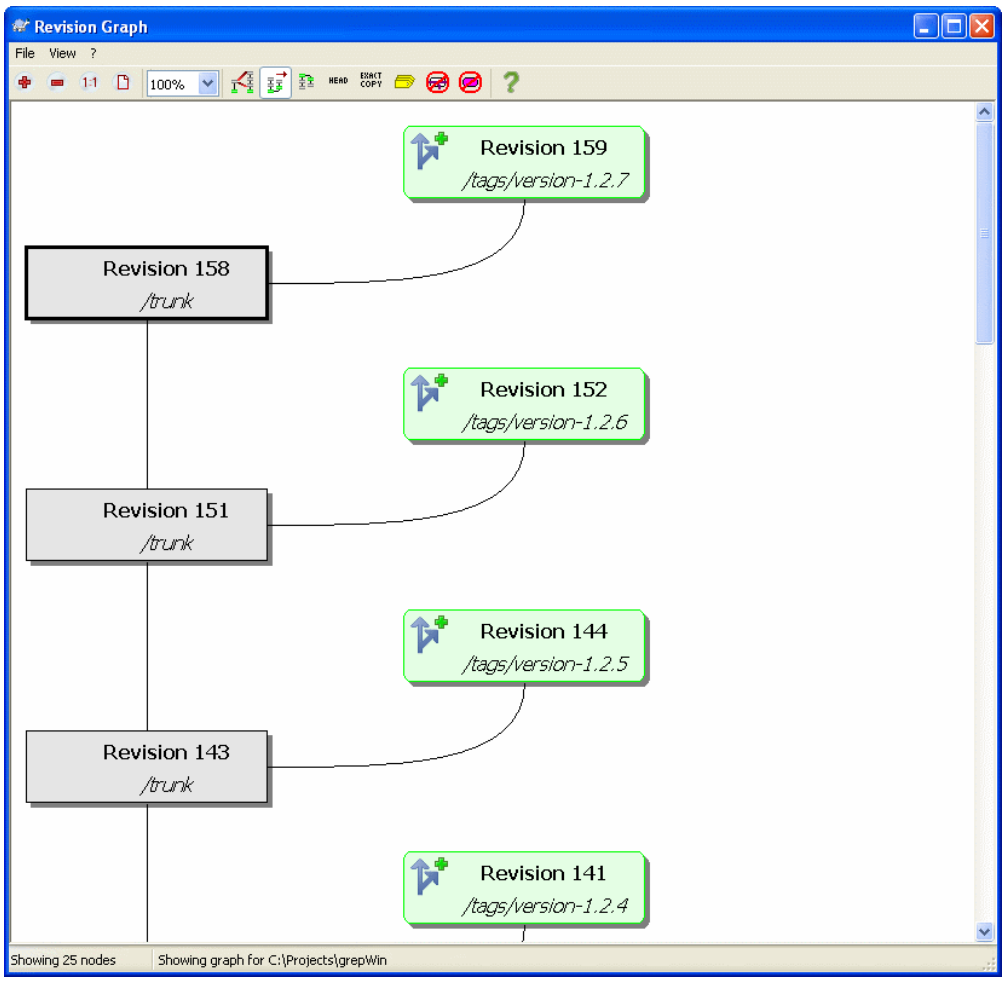


図4.47 リビジョングラフ

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

各リビジョングラフのノードは、あなたが見ているツリーの、何か変更されたリポジトリの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
: 追加された項目や、別のファイル・フォルダをコピーして作成されたものを、角の丸い長方形で表します。デフォルトの色は緑です。タグやブランチは特殊な状態として扱い、TortoiseSVN → 設定 によって異なる影を用います。

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 (角を落とした矩形) で表します。デフォルト色は赤です。

名前を変更した項目
: 名前変更した項目は、八角形でも表します。デフォルトの色は青です。

ブランチティップリビジョン
: グラフは通常表示しているブランチポイントに制限されていますが、各リビジョンの、個々の最新 (HEAD) リビジョンも表示できるとしばしば便利です。**最新のリビジョンを表示する** を選択すると、各最新リビジョンノードが楕円で表示されます。ここで言う最新 (HEAD) は、そのパスでコミットした最終リビジョンを指し、リポジトリの HEAD リビジョンではないことにご注意ください。

作業コピーリビジョン
: 作業コピーのリビジョングラフを呼び出すと、**作業コピーのリビジョンを表示する** を用いてグラフの BASE リビジョンを表示できます。このとき、BASE ノードは太い輪郭で表します。

変更した作業コピー
: 作業コピーのリビジョングラフを呼び出すと、Show WC modifications を用いて、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、通常の矩形で表します。

デフォルトでグラフには、項目の追加・コピー・削除のみ表示することにご注意ください。プロジェクト内のすべてのリビジョンを表示すると、重要なものだけでも非常に大きいグラフを生成してしまいます。本当に、変更が行われたすべてのリビジョンが必要なら、**表示** メニューやツールバーに、このようにする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、ノードを縦方向に、厳密なリビジョン順で配置するため、実行された順番の、視覚的な手がかりがあります。同じ列にあるふたつのノードの順番は明白です。ふたつ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。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、参照として使用することにご注意ください。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、お好み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。

分岐をグループ化
: デフォルトの振る舞い (グループ化無効) では、すべての行をリビジョンごとに厳密に並べ替えます。その結果、長生きなブランチは、変更が少ないにもかかわらず大量の列を占有し、グラフが非常に広くなってしまいます。

  このモードは変更をブランチでグループ化するため、グローバルなリビジョン順はありません。ブランチ上の連続したリビジョンは、連続した線で (大抵) 表します。しかしサブブランチの場合、グラフをスリムにしておくため、古いブランチと同じ列に後から発生したブランチを配置するようにします。その結果、異なるリビジョン由来の変更を同じ行に含む可能性があります。

古いものを先頭に表示する
: 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。

トップでツリーをそろえる
: 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する** 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。

線が交差しないようにする
: グラフのレイアウト上で、線の交差が大量に発生した場合、このオプションで整理してください。これによりレイアウトの列はあまり論理的でない場所に配置する可能性があります (例: 列でなく対角線)。また、描画するのに大きな領域が必要です。

パス名で差分をとる
: 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する
: 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。

最新のリビジョンを表示
: 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。

正確なコピー元の表示
: 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、その時の最新 (HEAD) 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 リビジョンよりも小さい可能性があることにご注意ください。

タグを折りたたむ
: プロジェクトにタグがたくさんある場合、すべてのタグをグラフの独立したノードとして表示するため、多くの領域を消費し、興味深い開発ブランチ構造が目立たなくなってしまいます。同時に、リビジョンを比較するため、タグの内容に簡単にアクセスする必要があるかもしれません。このオプションは、タグのノードを非表示にし、コピーしたノードにツールチップで表示するようにします。タグが作成された場合、元のノードの右側にタグアイコンを表示します。

削除されたパスは表示しない
: リポジトリの HEAD (最新) リビジョンに、すでにもうないパス (削除されたブランチなど) を隠します。

未変更のブランチを隠す
: それぞれのファイルやサブフォルダに対して、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 (最新) 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
: 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。

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。

概要表示
: 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. グラフの利用

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。

リビジョンの日付、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利用できます。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-Diff 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。変更し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うまい方法です! 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

初めて見たユーザは、ユーザ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。

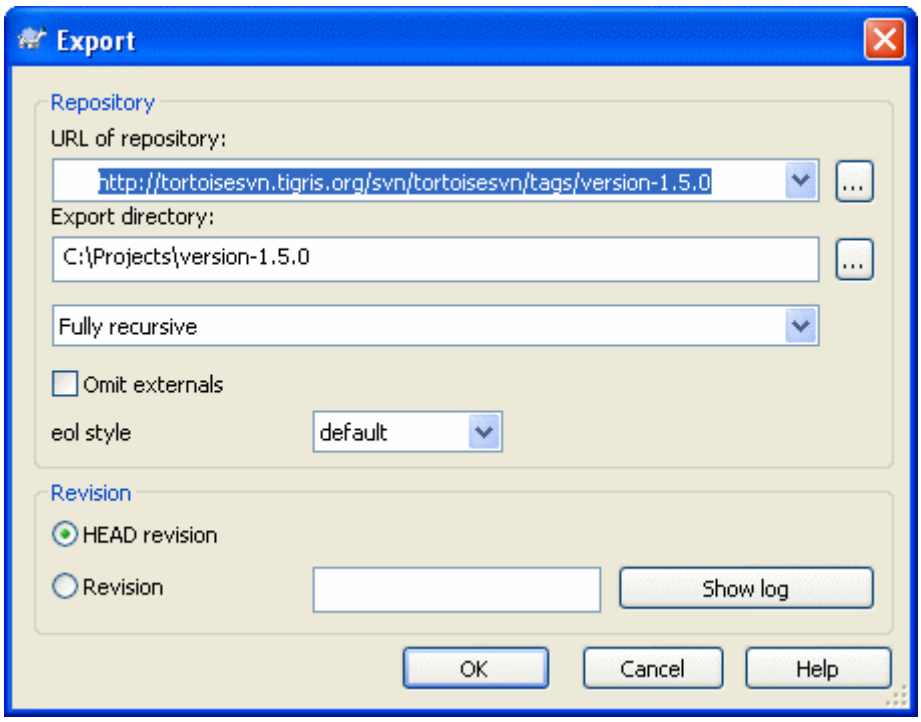


図4.48 URL からエクスポートダイアログ

バージョン管理外のフォルダでこのコマンドを実行すると、TortoiseSVN は選択したフォルダ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、最上位フォルダのみかどうか、外部参照を省略するかどうか、svn:eol-style 属性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 を利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する** や **コンテキストメニュー → SVN ここに全てをエクスポートする** 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に、エクスポートするフォルダと同じ名前が存在するフォルダを目的フォルダにすると、既存の内容を上書きするか、Target (1) のように、自動的に名前を生成して新しいフォルダを作成するかを選択してください。

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 (「リポジトリブラウザ」) を使う必要があります。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からエクスプ

ローラの目的の場所にドラッグするか、リポジトリブラウザ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

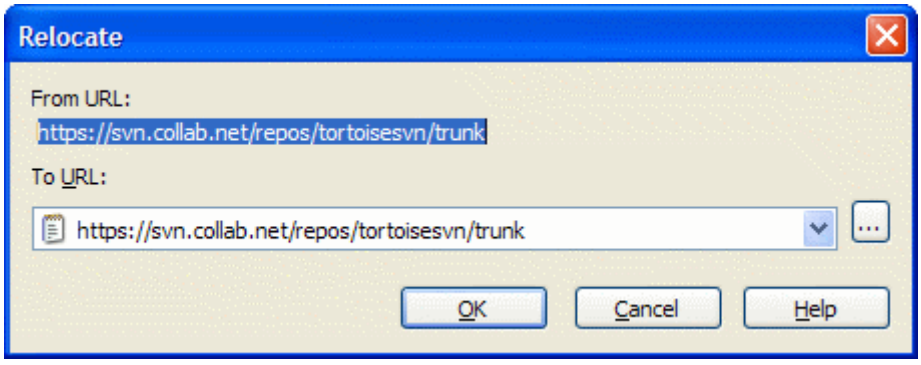


図4.49 再配置ダイアログ

様々な理由でリポジトリの場所 (IP/URL) を変更することがあります。おそらく、コミットできなくなってしまい、でも作業コピーをもう一度チェックアウトしてから変更を新しい作業コピーに行うなんてことはしたくないと、困ってしまうことでしょう。TortoiseSVN → 再配置 はそのためにあります。基本的には大したことはしません。すべての `.svn`フォルダ内の `entries` ファイルを検索し、URL を新しい値に変更するだけです。

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. バグ追跡システム / 課題追跡システムとの統合

変更が特定のバグ ID や課題 ID に関連するのは、ソフトウェア開発では非常に一般的です。バグ追跡システム (課題追跡システム) のユーザは、課題追跡システムの特定の ID と Subversion で行った変更を関連付けたいと思っています。そのため、ほとんどの課題追跡システムは、コミットに関連するバグ ID を探すためにログメッセージを解釈する、pre-commit フックスクリプトを提供しています。これは、pre-commit フックスクリプトが正しく解釈できるよう、適切なログメッセージを書くようユーザに頼っています。これはいくらか誤る傾向があります。

TortoiseSVN は、以下の 2 種類の方法でユーザの補助を行います。

1. ユーザがログメッセージを入力すると、コミットに関連付けられた課題番号を含む、よく定義された行が自動で追加されます。これにより、バグ追跡ツールが正しく解釈できない方法で、ユーザが課題番号を入力してしまう危険を減らせます。

   また、TortoiseSVN は入力したログメッセージの中で、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、ユーザが知ることができます。

2. ユーザ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バグ ID に、課題に言及したページをブラウザで開くリンクを張ります。

4.28.1. ログメッセージの課題番号付与

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:` で始まる属性を使用します。以下のようにフォルダに設定しなければなりません。(「プロジェクト設定」)

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。ひとつは単純な文字列によるもの、もうひとつは 正規表現 によるものです。属性にはどちらの方法も使用できます。

bugtraq:url
: この属性には、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、`%BUGID%` を含まなければなりません。`%BUGID%` は入力した課題番号に置換されます。これにより TortoiseSVN はログダイアログにリンクを生成し、リビジョンログ参照時に、バグ追跡ツールへ直接ジャンプできるようになります。この属性がないと、TortoiseSVN は

課題番号を表示するだけで、リンクを生成しません。例: TortoiseSVN プロジェクトは http://issues.tortoisesvn.net/?do=details&id=%BUGID% 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、あなたの課題追跡システムが、あなたのソースリポジトリと同じドメイン・サーバ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 属性を調整する必要はありません。相対 URL を指定するには以下の 2 つの方法があります。

^/ で始まる場合は、リポジトリルートからの相対 URL であると見なします。例えば、リポジトリが http://tortoisesvn.net/svn/trunk/ にある場合、^/../?do=details&id=%BUGID% を、http://tortoisesvn.net/?do=details&id=%BUGID% と解釈します。

/ で始まる URL は、サーバ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% を、http://tortoisesvn.net/?do=details&id=%BUGID% と解釈します。

bugtraq:warnifnoissue
課題番号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したければ、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 として扱います。

4.28.1.1. テキストボックスへの課題番号

シンプルアプローチでは、TortoiseSVN はユーザに、バグ ID を入力できる独立した入力フィールドを表示します。すると、ユーザ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
この属性は、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。この属性が設定されると、変更をコミットする際に、TortoiseSVN は課題番号を入力するよううながします。これはログメッセージの最後に行を追加するのに使用します。コミットの際に課題番号に置換される、%BUGID% を含まねばなりません。コミットログが課題番号への参照を含むのを確実に行います。課題番号は常に一貫した形式で、特殊なコミットで課題番号を関連付けるよう、バグ追跡ツールが解釈できます。例えば Issue : %BUGID% を使用するはずですが、これは自分のツールに依存します。

bugtraq:append
バグ ID をログメッセージの最後に追加する (true) ときや、ログメッセージの先頭に挿入する (false) ときにこの属性を定義します。有効な値は true/false です。 既存のプロジェクトが壊れないよう、これが定義されていない場合、true として扱います。

bugtraq:label
このテキストは、TortoiseSVN がコミットダイアログの、課題番号を入力するエディットボックスのラベルに使用されます。設定されていないと、Bug-ID / Issue-Nr: が表示されます。このラベルにぴったり合うようにウィンドウのサイズ変更ができませんので、ラベルは 20-25 文字以下になるよう留意してください。

bugtraq:number
true に設定すると、課題番号テキストフィールドには数値しか入力できなくなります。カンマは例外ですので、カンマで区切って複数の数値を入力できます。有効な値は true/false です。 定義されていなければ、true として扱います。

4.28.1.2. 正規表現を利用した課題番号

正規表現でのアプローチでは、TortoiseSVN は独立した入力フィールドを表示しませんが、ユーザが入力したログメッセージの中の、課題追跡システムに関係する部分をマーク付けします。ログメッセージを入力している間、有効です。これはバグ ID が、ログメッセージのどこにあってもかまわないと言うことです! 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

この属性は、正規表現 モードでバグ追跡システムを有効にします。ここには改行で区切っていずれかひとつの正規表現を設定できます。

2 つ正規表現を設定した場合、最初の正規表現を、バグ ID を含む表現を探す、前処理フィルタとして使用します。ふたつ目の正規表現で、はじめの正規表現の結果から、生のバグ ID を抽出します。これにより、お好みでバグIDの一覧と、自然言語表現が利用できます。例えば、いくつかのバグを修正し、「This change resolves issues #23, #24 and #25」といった文字列を入れたとします。

上記のログメッセージの中にある表現を用いてバグ ID を捕捉するには、以下のような (TortoiseSVN プロジェクトで使用している) 正規表現を使用できます。`[Ii]ssues?:?(¥s*(,|and)?¥s*#¥d+)+` と `(¥d+)` です。

最初の正規表現で、周囲ののログメッセージから「issues #23, #24 and #25」を抜き出します。ふたつ目の正規表現は、最初の正規表現の出力から、生の 10 進の数値を抽出します。つまり、バグ 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」が続いてもよく (複数型)、コロンが続いてもよいです。この後には、0 以上の空白が続き、コンマか「and」が続いてもよく、その後また 0 以上の空白があるグループが 1 つ以上あります。最後に必ず「#」と 10進数の数値が来ます。

正規表現が 1 個のみの場合は、むき出しのバグ 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 のような少数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。あなたの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] にある導入や、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。

`bugtraq:message` と `bugtraq:logregex` の両方をセットしている場合、`logregex` を優先します。

ヒント

pre-commit フックスクリプトで解釈していれば、課題追跡システムがない場合でも、ログメッセージ内の課題に言及している箇所を、リンクにするために使用できます!

またリンクが必要なくても、課題番号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:` 属性のいくつかは `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。

フォルダへの属性の設定

システムが動作するのに、上記の属性はフォルダにセットしなければなりません。ファイルやフォルダをコミットする際に、このフォルダから属性が読み込まれます。そこに属性がない場合、TortoiseSVN は、バージョン管理外フォルダか、ツリーのルート (例: `C:¥`) に行き着くまで、フォルダツリーをさかのぼって探します。ユーザがそれぞれ、例えば`trunk/` からのみチェックアウトして、

サブフォルダを含まないことが確かならば、trunk/ にのみセットすれば十分です。確かでなければ、各サブフォルダに再帰的に属性を設定する必要があります。プロジェクト階層の深いところで設定された属性は、高いレベル (trunk/ に近い) のものを上書きします。

tsvn: 属性 だけ ですが、再帰チェックボックスを使って階層の全サブフォルダに属性をセットできます。このとき全ファイルが対象になるわけではありません。

リポジトリブラウザから課題追跡システムの情報を利用できません

課題追跡システムとの統合は、subversion の属性にアクセスできることに依存しているため、チェックアウトした作業コピーを使用するときにだけ、結果を目にするでしょう。リモートから属性を取得するのは遅い操作ですので、作動中のこの機能をリポジトリブラウザから目にすることはないでしょう。

この課題追跡システムとの統合は、TortoiseSVN に限定されていません。ですから、いずれの Subversion クライアントでも使用できます。詳細は、TortoiseSVN のソースリポジトリにある、Issue Tracker Integration Specification [http://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/issuetrackers.txt] の全体をお読みください (リポジトリのアクセスのしかたは「TortoiseSVN は自由!」で説明します)。

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会話ができるような、外部プログラムを統合する COM インターフェースを備えています。通常、今回のコミットで取り組む課題を取り上げられるように、自分に割り当てられた未解決の課題を取得するよう、課題追跡システムに問い合わせを行うのでしょう。

Any such interface is of course highly specific to your issue tracker system, so we cannot provide this part, and describing how to create such a program is beyond the scope of this manual. The interface definition and sample plugins in C# and C++/ATL can be obtained from the contrib folder in the TortoiseSVN repository [http://tortoisesvn.googlecode.com/svn/trunk/contrib/issue-tracker-plugins]. (「TortoiseSVN は自由!」 explains how to access the repository). A summary of the API is also given in 6章IBugtraqProvider インターフェース Another (working) example plugin in C# is Gurtle [http://code.google.com/p/gurtle/] which implements the required COM interface to interact with the Google Code [http://code.google.com/hosting/] issue tracker.

説明のために、あなたのシステム管理者が、あなた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。

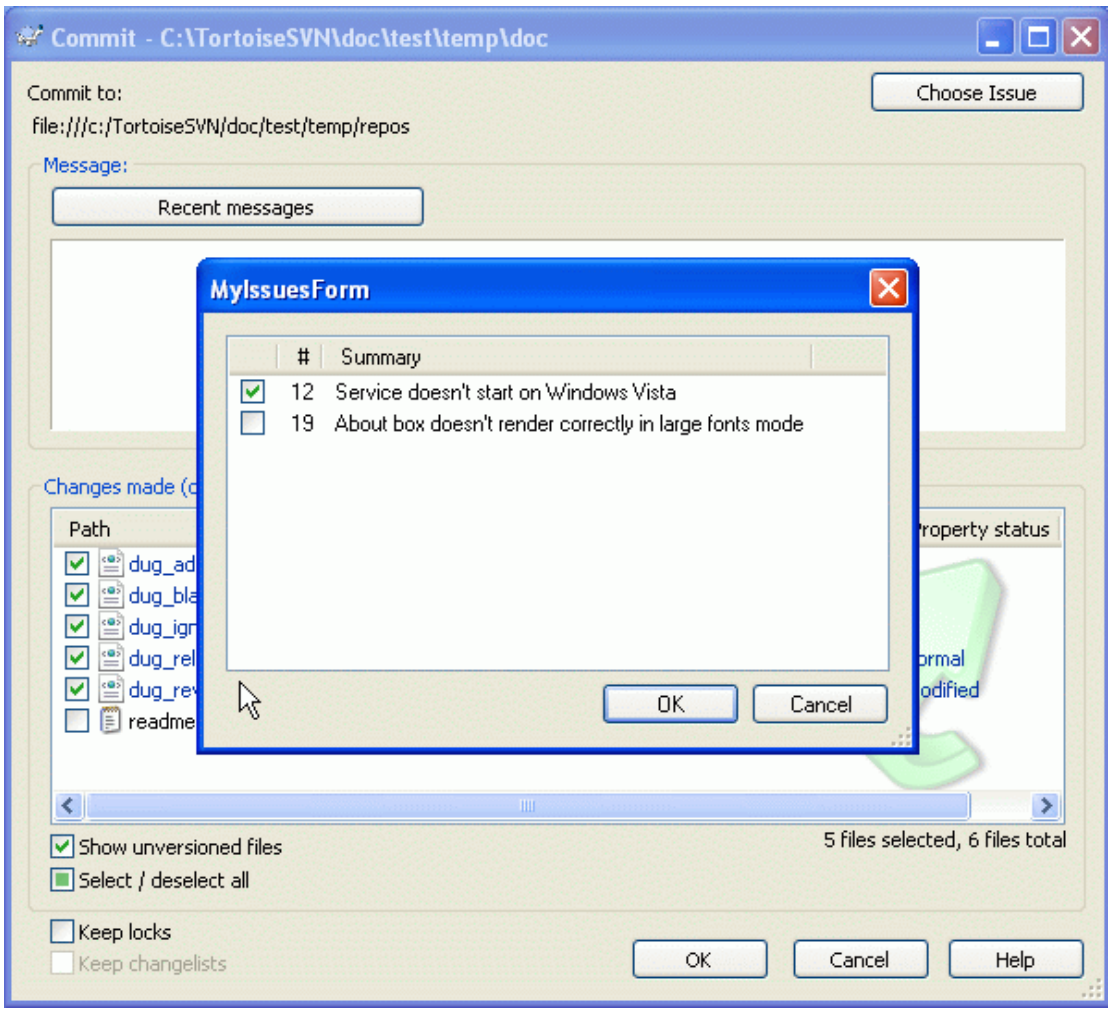


図4.50 課題追跡システムクエリダイアログの例

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 ベースリポジトリビューアとの統合

ViewVC [http://www.viewvc.org/] や WebSVN [http://websvn.tigris.org/] のような、Subversion と共に使用する Web ベースのリポジトリビューアがいくつかあります。TortoiseSVN は、そういったビューアとのリンクを提供します。

TortoiseSVN はお好みのリポジトリビューアと統合できます。そのためには、リンク用に用意されている属性を定義する必要があります。以下のようにフォルダに設定しなければなりません。(「プロジェクト設定」)

webviewer:revision
: 指定したリビジョンの全変更点を表示するリポジトリビューアの URL を、この属性にセットしてください。適切に URI エンコードされていなければならず、%REVISION% を含んでいる必要があります。%REVISION% は問い合わせるリビジョン番号に置換されます。これにより TortoiseSVN は、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で表示** を表示するようになります。

webviewer:pathrevision
: 指定したリビジョン、指定したファイルの変更点を表示するリポジトリビューアの URL を、この属性にセットしてください。適切に URI エンコードされていなければならず、%REVISION% と %PATH% を含んでいる必要があります。%PATH%

は、リポジトリのルートからの取得するパスに置換されます。これにより TortoiseSVN は、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 → リビジョンとパスをウェブビューアで表示** を表示するようになります。例えば、ログダイアログの下部ペインにある `/trunk/src/file` エントリで右クリックした場合、URL の `%PATH%` は `/trunk/src/file` に置換されます。

絶対 URL の代わりに相対 URL も使用できます。これは、あなたのウェブビューアが、あなたのソースリポジトリと同じドメイン・サーバ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` 属性や `webviewer:pathrevision` 属性を調整する必要はありません。フォーマットは `bugtraq:url` 属性と同じです。「バグ追跡システム / 課題追跡システムとの統合」をご覧ください。

フォルダへの属性の設定

システムが動作するのに、上記の属性はフォルダにセットしなければなりません。ファイルやフォルダをコミットする際に、このフォルダから属性が読み込まれます。そこに属性がない場合、TortoiseSVN は、バージョン管理外フォルダか、ツリーのルート (例: `C:¥`) に行き着くまで、フォルダツリーをさかのぼって探します。ユーザがそれぞれ、例えば`trunk/` からのみチェックアウトして、サブフォルダを含まないことが確かならば、`trunk/` にのみセットすれば十分です。確かでなければ、各サブフォルダに再帰的に属性を設定する必要があります。プロジェクト階層の深いところで設定された属性は、高いレベル (`trunk/` に近い) のものを上書きします。

`tsvn:` 属性 だけ ですが、**再帰**チェックボックスを使って階層の全サブフォルダに属性をセットできます。このとき全ファイルが対象になるわけではありません。

リポジトリブラウザからリンクするリポジトリビューアはありません

リポジトリビューアとの統合は、subversion の属性にアクセスできることに依存しているため、チェックアウトした作業コピーを使用するときにだけ、結果を目にすることになります。リモートから属性を取得するのは遅い操作ですので、作動中のこの機能をリポジトリブラウザから目にすることはないでしょう。

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

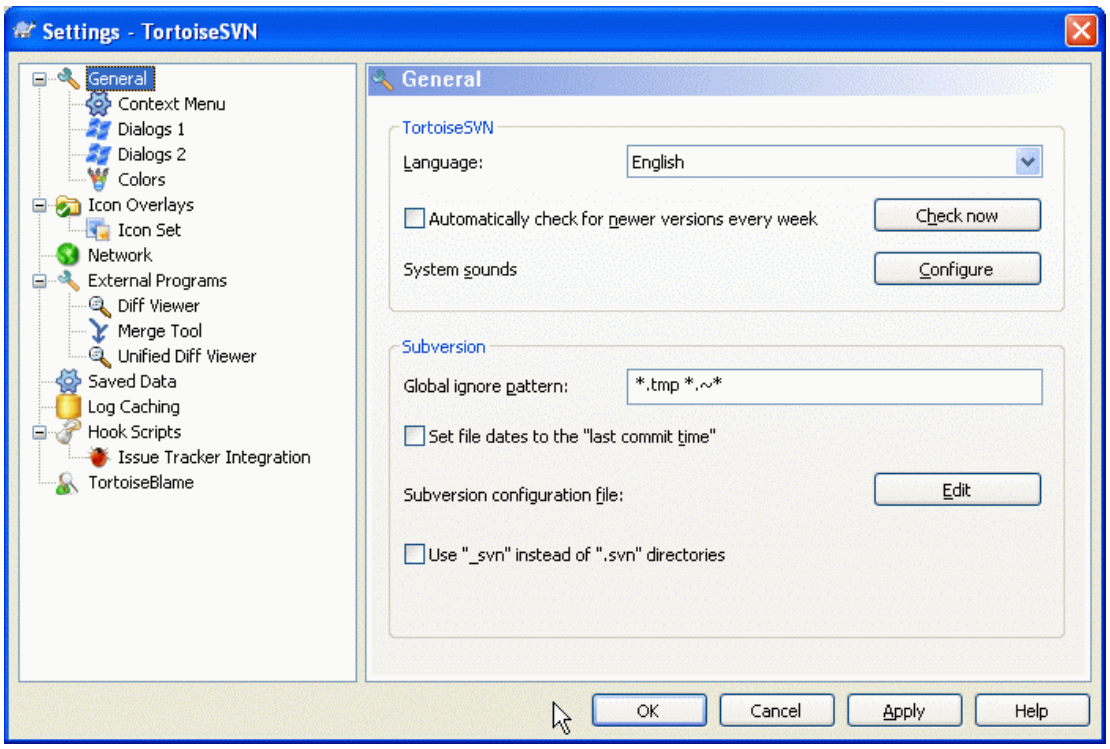


図4.51 設定ダイアログの一般ページ

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
: ユーザインターフェースの言語を選択してください。他に何かありますか?

毎週、最新のバージョンのチェックを自動的に行う
: チェックすると、TortoiseSVN は週に一度、プログラムの新バージョンがないかダウンロードサイトに接続します。すぐに答えが知りたければ、**今すぐチェック** を使用してください。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 にはデフォルトでカスタムサウンドが 3 つ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。

共通無視パターン
: 共通無視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に表示し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します。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が必要ありません。ファイルとディレクトリを区別する方法はないことに注意してください。パターンマッチの文法といった詳細情報は、「無視リストでのパターンマッチ」 をご覧ください。

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

共通無視パターンを設定した Subversion 設定ファイルを使用すると、ここでした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** 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、他のユーザには影響しません。対照的に、バージョン管理で `svn:ignore` 属性を使用し、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「無視するファイルとディレクトリ」をご覧ください。

ファイルの日付に「最後にコミットした日付」を設定する

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、現在の日付にしておくのが最善でしょう。「最後にコミットした日付」 を使用していて、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/1.5/svn.advanced.confarea.html] をご覧ください。Automatic Property Setting [http://svnbook.red-bean.com/en/1.5/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/1.5/svn.advanced.confarea.html#svn.advanced.confarea.windows-registry] を参照してください。

`.svn` ディレクトリの代わりに、`_svn` を使用する

web プロジェクトを使用する際、Subversion が内部情報を格納する `.svn` フォルダを、VS.NET は扱えません。これは Subversion のバグではありません。VS.NET とそこで使用する frontpage 拡張のバグです。この件に関してもっと調べるには「Subversion の作業フォルダ」をご覧ください。

Subversion や TortoiseSVN の振る舞いを変更する場合、これを制御する環境変数を設定するのに、チェックボックスを使用できます。

このオプションを変更すると、既存の作業コピーを新しい管理下のディレクトリに、自動でコンバートできないことに注意してください、スクリプトを使って自分で行うか、単純に新しい作業コピーをチェックアウトする必要があります。

4.30.1.1. コンテキストメニュー設定

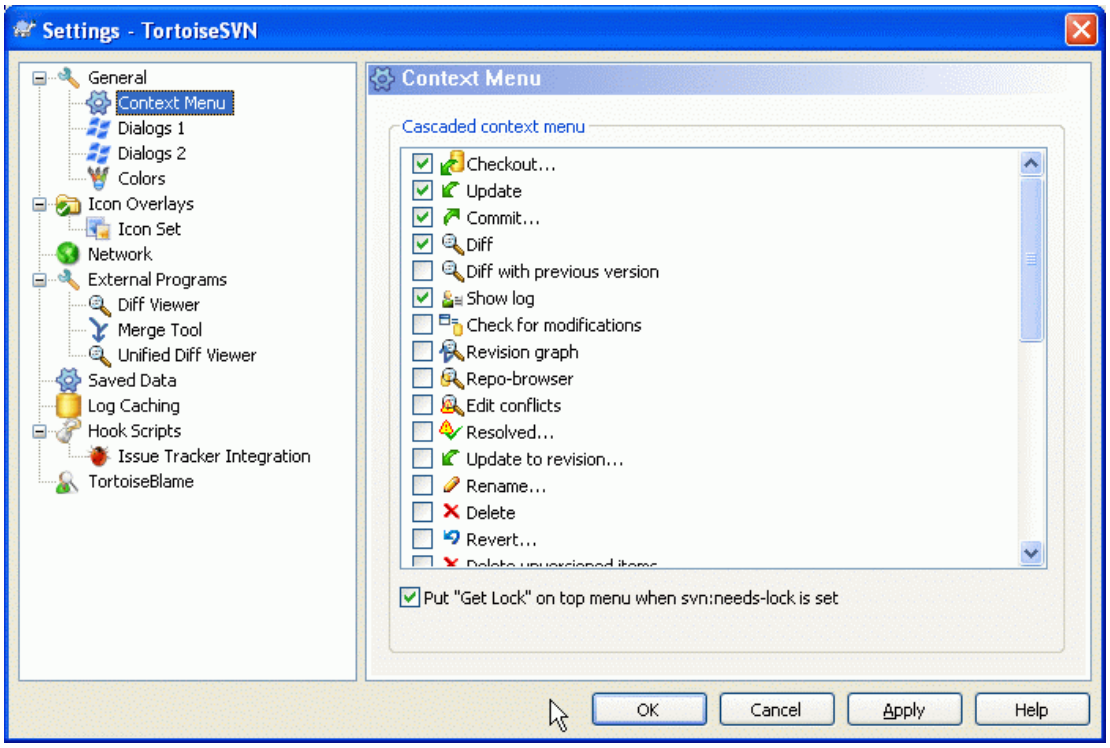


図4.52 設定ダイアログ、コンテキストメニューページ

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` 属性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` 属性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。

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

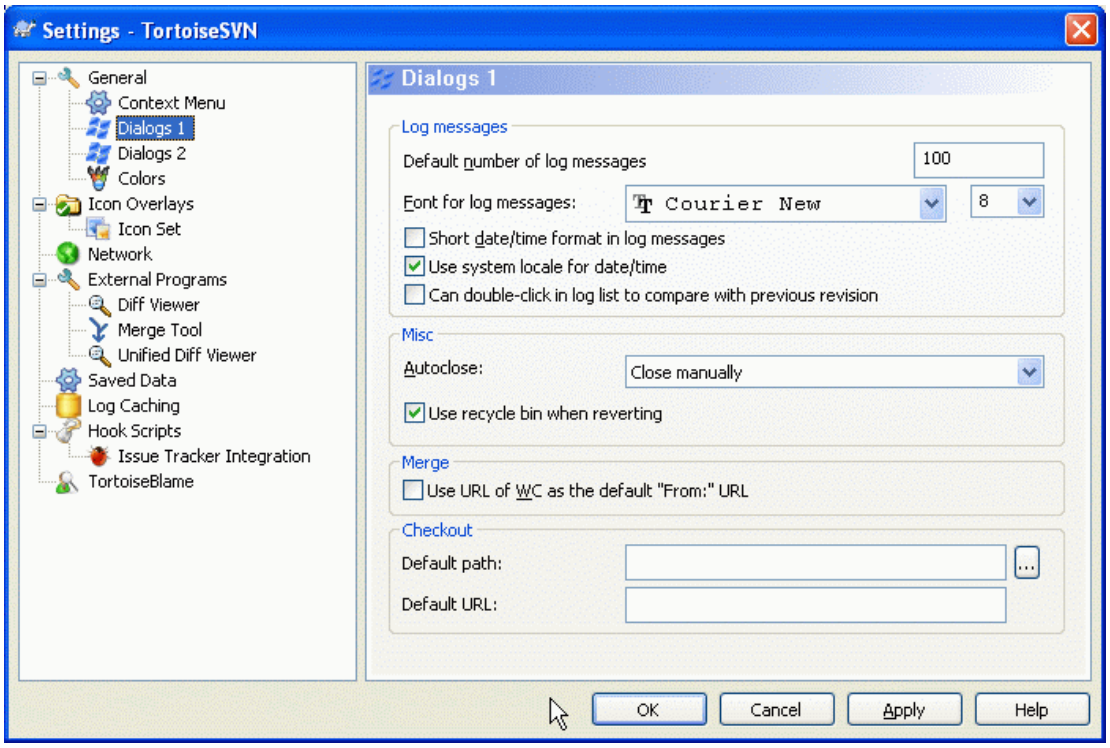


図4.53 設定ダイアログ、ダイアログ 1 ページ

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
: **TortoiseSVN → ログを表示** を選択して最初に TortoiseSVN が趣得するログメッセージの数を制限します。サーバ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** をいつでも使用できます。

ログメッセージ用フォント
: リビジョンログダイアログ中央ペインのログメッセージを表示す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のログメッセージ編集時にも使われます。

ログメッセージに短い日付形式を利用する
: 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、短い形式を利用します。

ログリスト内でダブルクリックすることで以前のリビジョンと比較する
: ログダイアログの上部ペインでリビジョンを比較しているなら、ダブルクリックで実行するため、このオプションを使用できます。差分の取得は長いプロセスになりがちなので、デフォルトでは有効ではありません。また、多くの人は間違ってダブルクリックしたあとで待ちたくないでしょうから、その意味でもデフォルトで有効にしていません。

進行ダイアログ
: TortoiseSVN はアクション中に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 (お勧め) 設定は **手動で閉じる** で、全メッセージを見て、何が起きたのかチェックできます。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、削除が無い場合は自動的に閉じる 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。

手元の操作でマージ、追加、削除が無い場合は自動的に閉じる は、マージ、追加、削除が無い場合は自動的に閉じる のように進行ダイアログを閉じますが、ファイルの追加や、変更の取り消しなど手元の操作の場合のみに作用します。リモート操作では、ダイアログが開いたままになります。

競合がない場合は自動的に閉じる 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。

エラーがない場合は自動的に閉じる は競合があっても常にダイアログを閉じます。ダイアログが開いたままになる条件は、Subversion が処理を継続できないようなエラーの発生です。例えば、サーバ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です。

元に戻す際にゴミ箱を利用する
: ローカルの変更を元に戻す際、あなたの変更は破棄されます。TortoiseSVN は元のコピーに戻す前に、変更したファイルをゴミ箱に送るというセーフティネットを用意しています。ゴミ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として利用する
: 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利用できます。

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

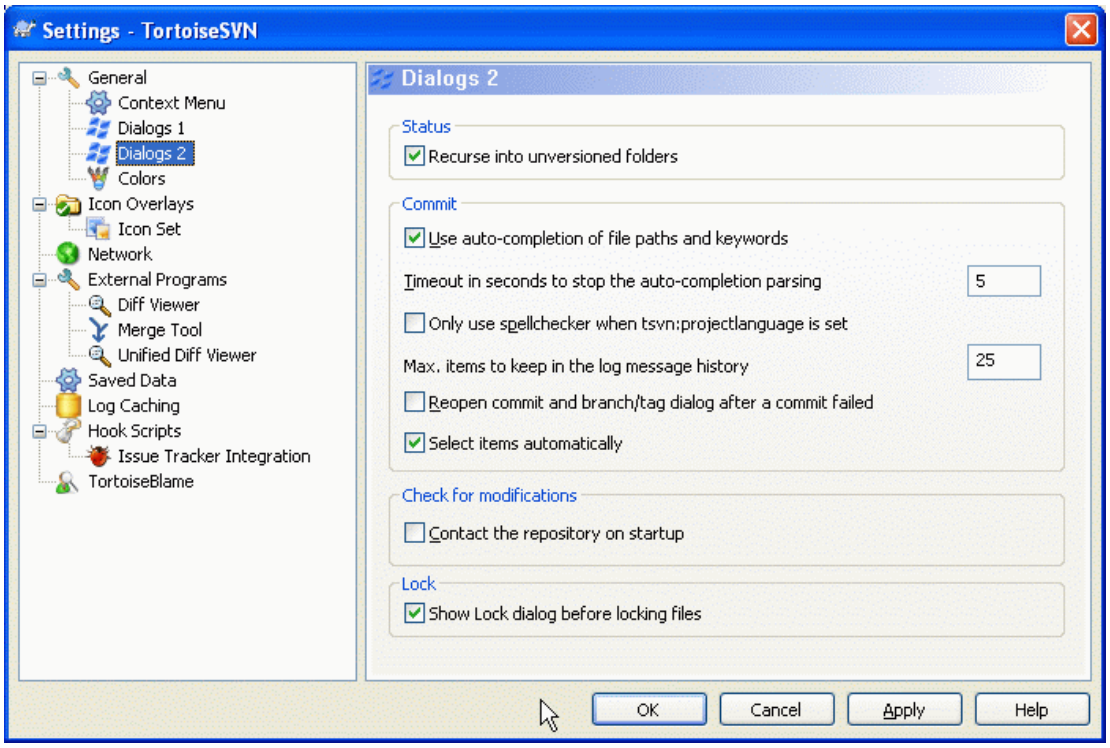


図4.54 設定ダイアログ、ダイアログ 2 ページ

バージョン管理外フォルダの再帰検索

チェックされている (デフォルト状態) と、常にバージョン管理外のフォルダ (とそれ以下のすべてのファイル・フォルダ) を **追加**, **コミット**, **変更をチェック** ダイアログに表示します。チェックしないと、バージョン管理外の先頭しか表示しません。チェックしなければ、上記のダイアログを散らかさずにすみます。この場合、バージョン管理外のフォルダを追加するよう選択すると、再帰的に追加します。

ファイルのパスやキーワードの自動補完を利用する

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を利用する

コミット時にスペルチェッカを使用しない場合、ここにチェックをつけてください。プロジェクト属性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分岐／タグダイアログを再表示する

何らかの理由でコミットが失敗 (作業コピーを更新する必要があったり、pre-commit フックスクリプトがコミットを拒否したり、ネットワークエラーなど) したときに、もう一度できるようにコミットダイアログを開いたままにするのに、このオプションを選択できます。しかし、このオプションが原因で問題が起こる可能性があることを知っておいてください。作業コピーを更新する必要があるという理由で失敗した場合、更新して競合が発生したら、まずこれを解決しなければなりません。

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自分で選択する場合、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、変更をチェックダイアログが作業コピーのチェックをするようになります。また、**リポジトリをチェック** をクリック時にリポジトリへの接続のみ行うようになります。リポジトリを常にチェックしたい場合、この設定で自動的にチェックが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` 属性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 での色設定

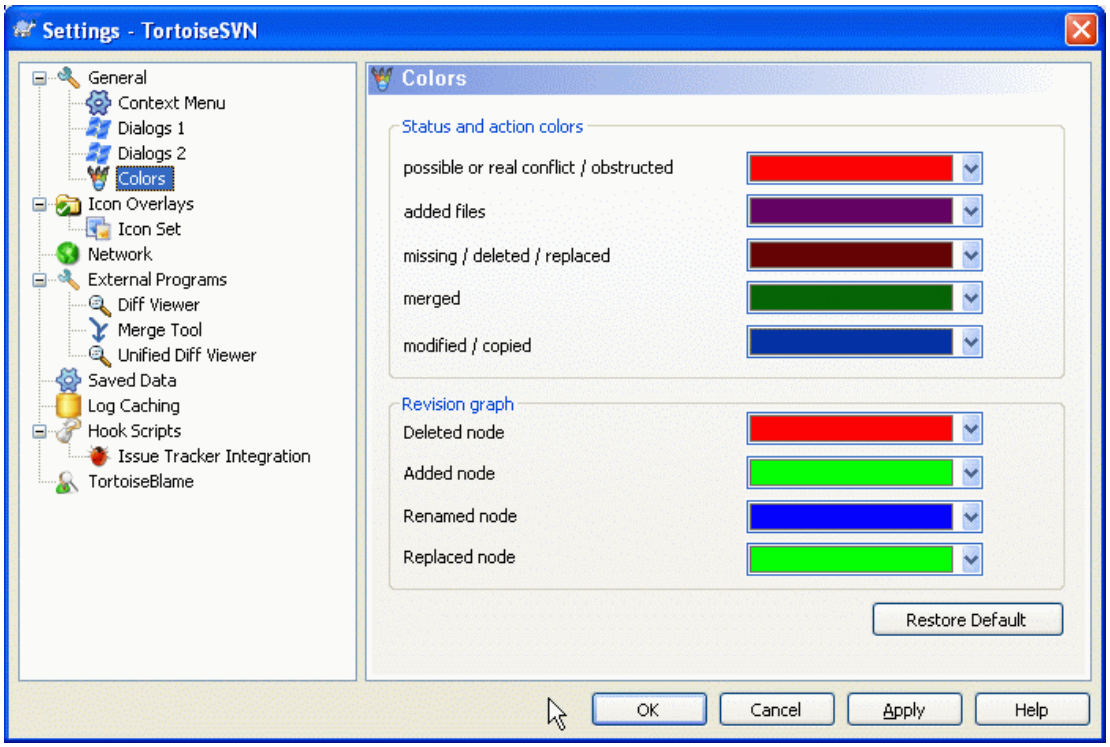


図4.55 設定ダイアログ、色設定ページ

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。

マージ済み
: 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。

追加したノード
: (追加・コピー・移動操作で) リポジトリに追加した項目。

名前を変更したノード
: 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。

4.30.2. リビジョングラフの設定

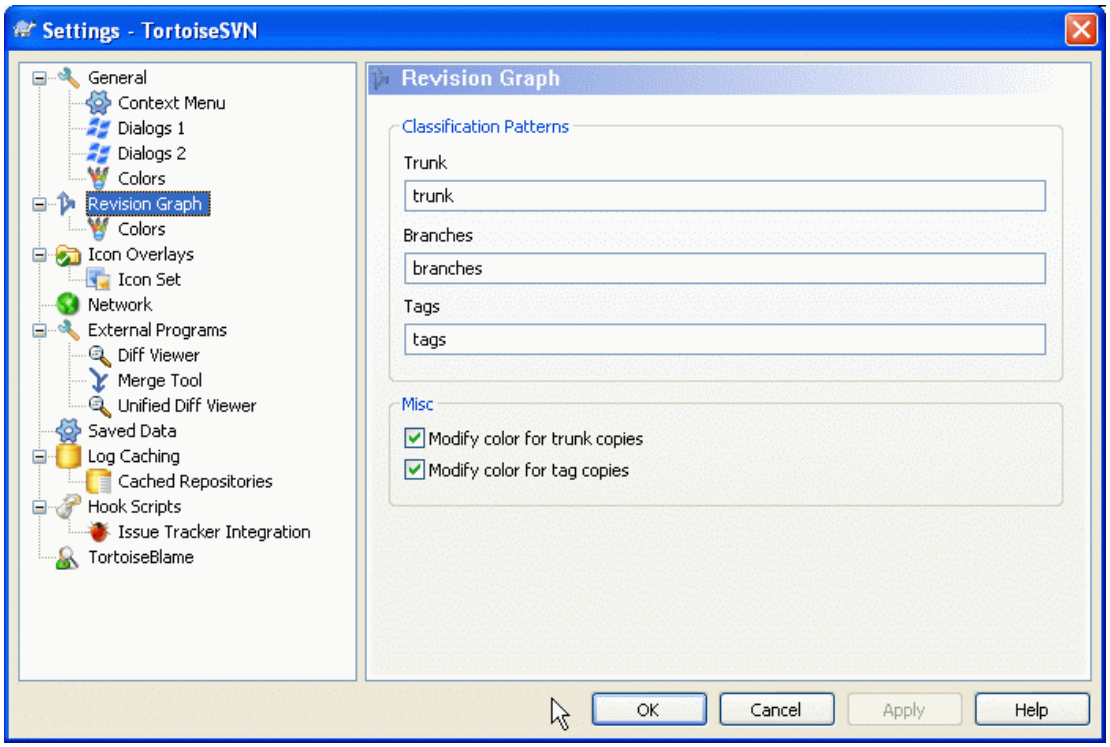


図4.56 設定ダイアログ、リビジョングラフページ

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、もちろんあなたの使い方はことなるかもしれません。

  みっ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。マッチング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

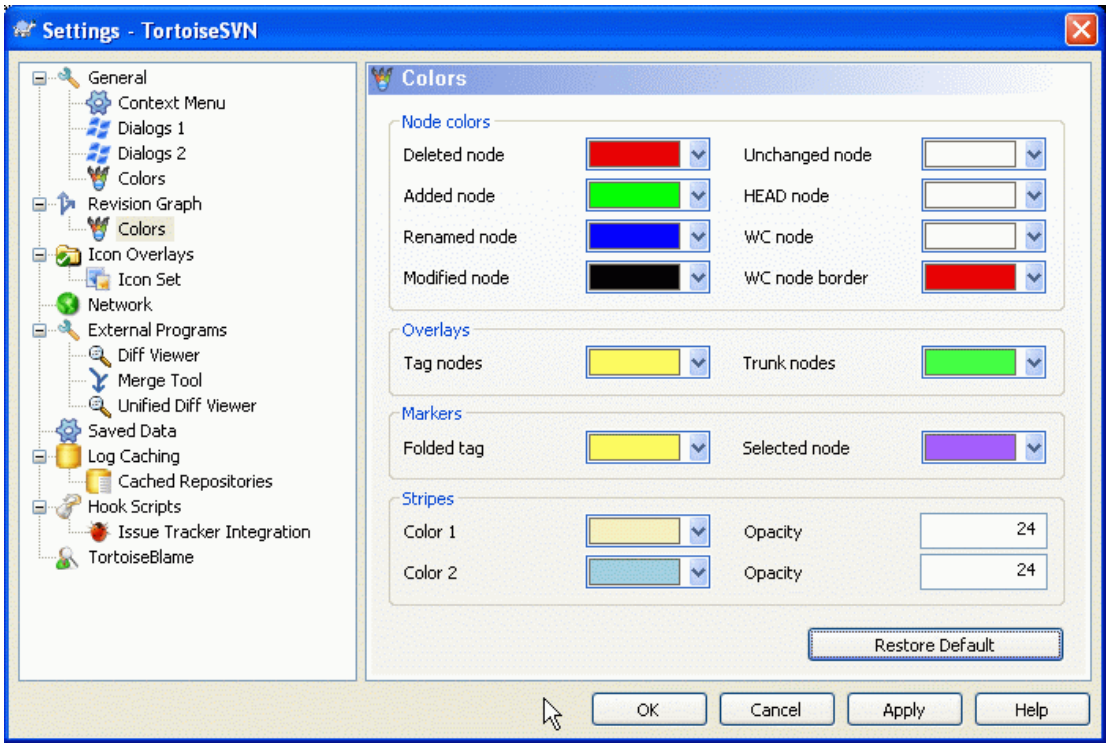


図4.57 設定ダイアログ、リビジョングラフ色設定ページ

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。

HEAD ノード
: リポジトリにある現在の 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。

WC 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

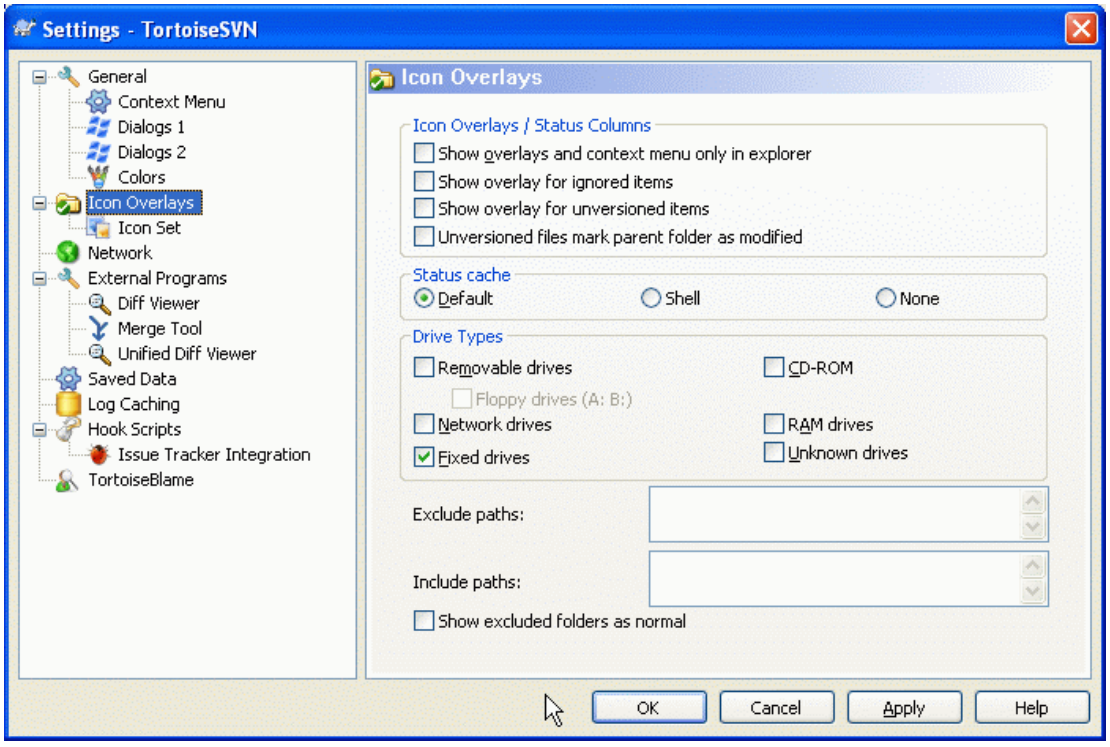


図4.58 設定ダイアログ、アイコンオーバーレイページ

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。

デフォルトでは、オーバレイアイコンやコンテキストメニューは 開く・保存ダイアログでも、Windows エクスプローラのように表示されます。Windows エクスプローラでのみ表示するようにするなら、**エクスプローラ上でのみオーバレイを表示する** をチェックしてください。

無視する項目やバージョン管理外の項目は、通常オーバーレイを表示しません。これらもオーバーレイを表示する場合は、このチェックボックスに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、キャッシュオプション (後述) が デフォルト の場合にのみ有効です。

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サイズをもとに、TortoiseSVN が使用するキャッシュタイプを以下から選択できます。

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、数秒でオーバーレイアイコンは更新されます。

  利点: オーバーレイアイコンは状態を再帰的に表示します。つまり、作業コピーの深いところにあるファイルが更新されると、作業コピーのルートまでの全フォルダにも更新オーバーレイアイコンが付きます。さらにプロセスがシェルに通知を送れるのなら、通常、左のツリービューのオーバーレイも変更します。

  欠点: プロジェクトについて作業をしていないときでも、プロセスが一定に動作しています。さらに、作業コピーのサイズ・数量にも依存しますが、およそ 10-50 MB の RAM を使用します。

シェル
: シェル拡張 dll の内部で直接キャッシュします。別のフォルダに移るたびに、ステータス情報を取得します。

  利点: メモリ消費がとても少なく (およそ 1 MB の RAM) てすみます。また、状態をリアルタイム に表示します。

  欠点: 1 つのフォルダしかキャッシュしないため、オーバーレイアイコンは状態を再帰的に表示しません。大きな作業コピーでは、デフォルトのキャッシュよりも、エクスプローラのフォルダ表示に時間がかかります。また、mime-type 列が利用できません。

なし
: この設定では、TortoiseSVN はエクスプローラで状態をまったく取得しません。このため、ファイルはオーバーレイアイコンを取得できず、バージョン管理されたフォルダは 「通常」のオーバーレイアイコンしか取得しません。他のオーバーレイは表示されず、他の方法では有効だった、追加の列は無効になります。

  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が遅くなることもありません。

  欠点: ファイル・フォルダの状態情報は、エクスプローラに表示されません。作業コピーの変更点を見るには、「変更をチェック」 ダイアログを使用しなければなりません。

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、ネットワーク共有上にある作業コピーに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そのもので特定されるので、特殊なケースのように思えます。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、アイコンオーバーレイや状態列を表示しないよう TortoiseSVN に指示する場合に使用します。変更する予定のないライブラリのみを含む非常に大きな作業コピーがあって、オーバーレイさせる必要がない場合に便利です。以下に例を示します。

f:¥development¥SVN¥Subversion では、指定したフォルダ のみ オーバーレイを無効にします。このフォルダ内の全ファイル・フォルダのオーバーレイはまだ見ることができます。

f:¥development¥SVN¥Subversion* では、f:¥development¥SVN¥Subversion で始まる すべて のファイル・フォルダでオーバーレイが無効になります。このパス以下のいずれのファイル・フォルダでもオーバーレイをさせたくないという意味になります。

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。

この 3 つの設定はどのように影響するのかといった問い合わせが、たまにあります。間違いない回答は以下になります。

```
if（含めるリストにあるパス）
  オーバーレイを表示
if（許可されたドライブの種類に属するパス）AND（除外するリストにないパス）
  オーバーレイを表示
```

含めるリストは 常に オーバーレイを表示します。一方、ドライブの種類でチェックされたドライブは、除外するパスを 除いて、すべて表示されます。

TSVNCache.exe は、検索するパスを制限することもできます。特定のフォルダ以下のみを見たい場合は、全ドライブタイプで無効にし、スキャンしたい特別なフォルダのみを含めてください。

SUBST ドライブの除外

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　が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、それは通常オリジナルパスに対してであるため、オーバーレイを更新できません。これは、subst　パスのオーバーレイはまったく更新されない可能性があるということでもあります。

これに対処する簡単な方法は、オーバーレイ表示からオリジナルのパスを除外し、代わりに subst パスにオーバーレイを表示することです。

時には、作業コピーに含まれる領域を　TSVNCache　の検索・変更監視から除外し、しかしバージョン管理下であることを視覚表示したいことがあります。**'通常' として除外フォルダを表示する** チェックボックスで、そのようにできます。このオプションにより、除外領域 (チェックしないドライブタイプや除外指定した場所) にあるバージョン管理下のフォルダには、緑のチェックマークで、通常でかつ最新であるという表示がつきます。これにより、作業コピーを見ていることをお知らせしますが、フォルダのオーバーレイはおそらく正しくないでしょう。ファイルには、オーバーレイがまったく付きません。オーバーレイは表示されませんが、コンテキストメニューは有効な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

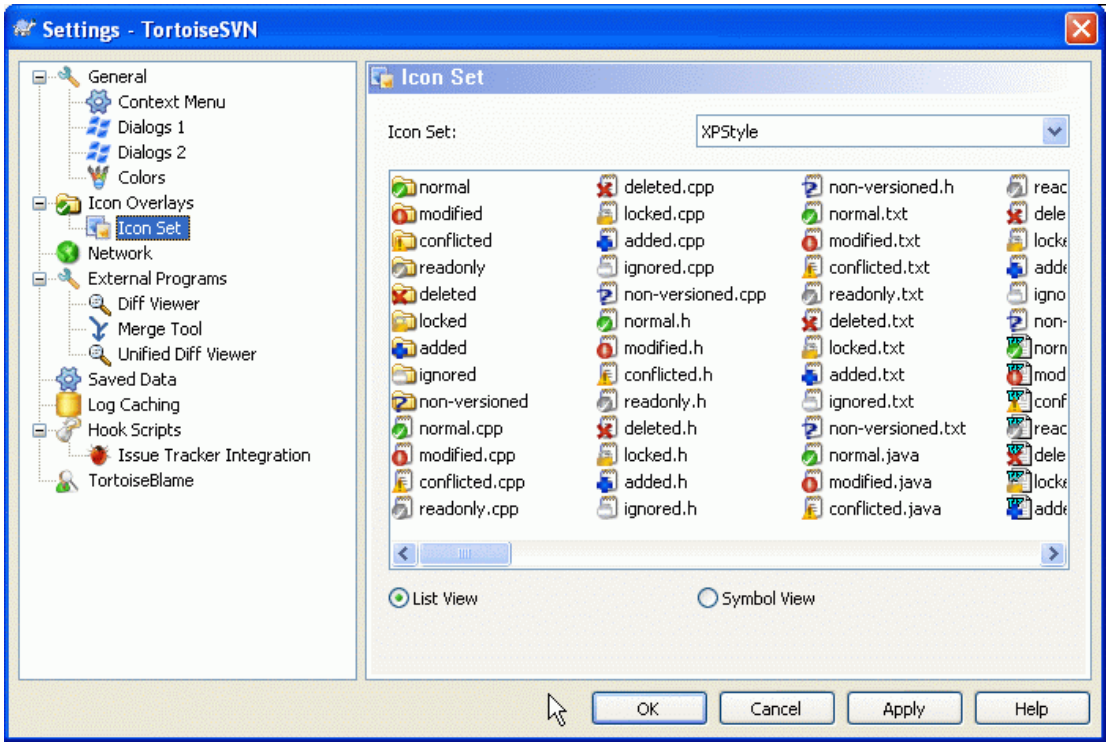


図4.59 設定ダイアログ、アイコン設定ページ

オーバーレイアイコンをお好み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の再起動が必要なことに注意してください。

4.30.4. ネットワーク設定

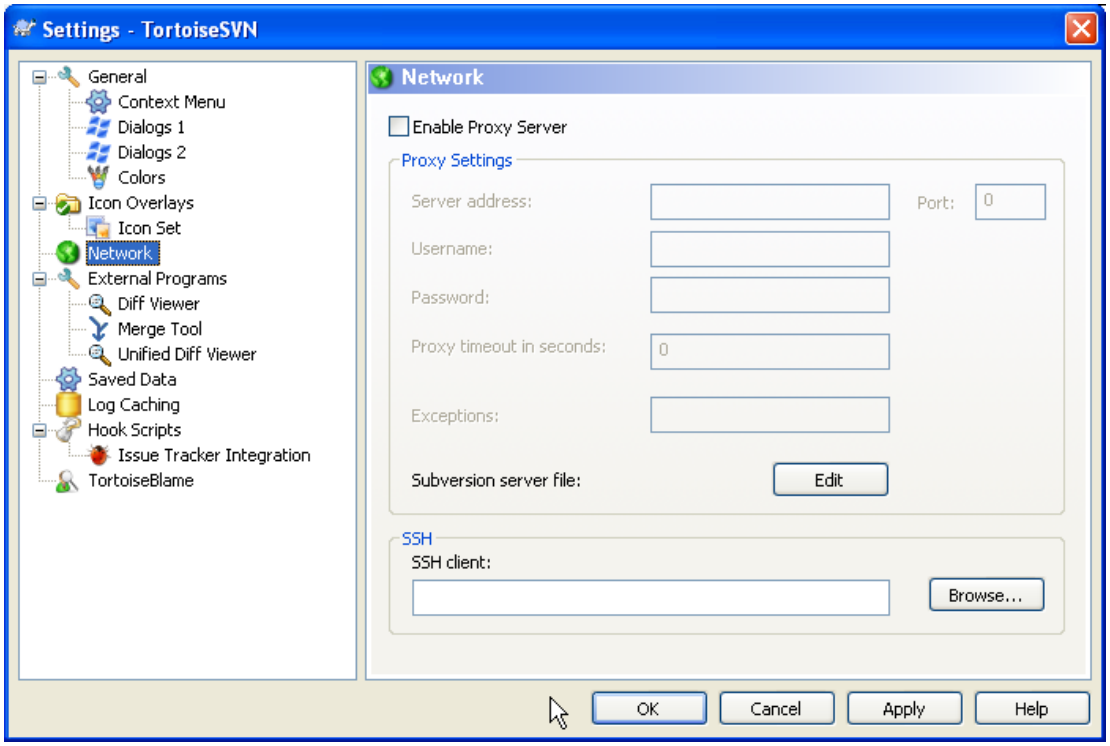


図4.60 設定ダイアログのネットワークページ

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、設定に Subversion servers ファイルを使用する必要があります。直接行う場合 **編集** を使用してください。このファイルの使用法の詳細は、Runtime Configuration Area [http://svnbook.red-bean.com/en/1.5/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で使用できます。

TortoisePlink 自身は、Plink のちょっとした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

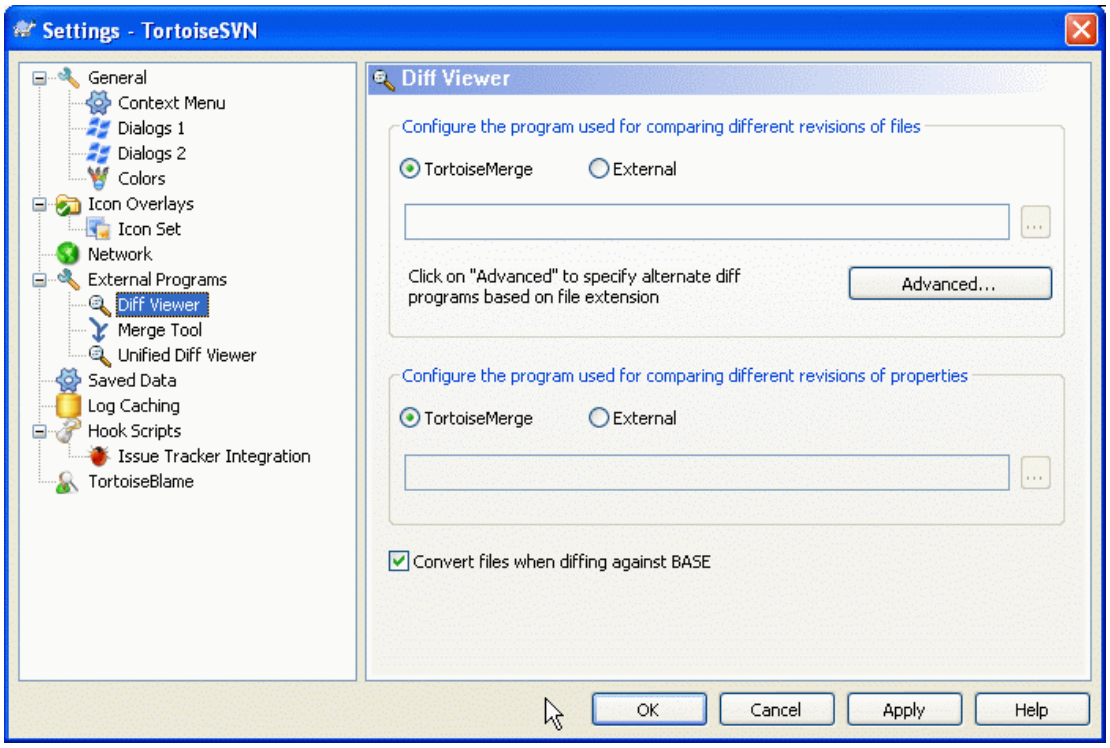


図4.61 設定ファイルの差分ビューアページ

ここでは、TortoiseSVN が使用する 差分/マージプログラムを定義できます。基本設定では TortoiseSVN と一緒にインストールされる TortoiseMerge が使われます。

TortoiseSVN と一緒に使用する外部 差分/マージプログラムの一覧は、「外部 Diff/Merge ツール」をご覧ください。

4.30.5.1. 差分ビューア

外部差分プログラムは、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。外部プログラムは、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を使用します。このパラメータが現れると、適切な値に置換されます。パラメータの順番はお使いの差分ツールに依存しています。

%base
: 変更していないオリジナルのファイル

%bname
: BASE ファイルのウィンドウタイトル

%mine
: 変更を行ったあなたのファイル

%yname
: あなたのファイルのウィンドウタイトル

ウィンドウタイトルは純粋なファイル名ではありません。TortoiseSVN は表示するための名前で扱い、それによって名前を作成します。つまり、リビジョン 123 のファイルから作業コピーのファイルへ差分を取る場合、名前は `filename : revision 123` と `filename : 作業コピー` となります。

以下に例を示します。ExamDiff Pro の場合:

```
C:¥Path-To¥ExamDiff.exe %base %mine --left_display_name:%bname
    --right_display_name:%yname
```

KDiff3 の場合:

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

WinMerge の場合:

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

Araxis の場合:

```
C:¥Path-To¥compare.exe /max /wait /title1:%bname /title2:%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、手元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 の属性用に別の diff ツールを指定することもできます。属性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を使いたくなるかもしれません。

代替 diff 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 → Diff** で優先 diff ツールを使用し、Shift+ **コンテキストメニュー → Diff** で代替 diff 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消に使用します。パラメータ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
: BASE 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
: あなたの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

以下に例を示します。Perforce Merge の場合:

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

KDiff3 の場合:

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

Araxis の場合:

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

WinMerge (2.8 以降) の場合:

```
C:¥Path-To¥WinMerge.exe %merged
```

4.30.5.3. 差分 / マージの高度な設定

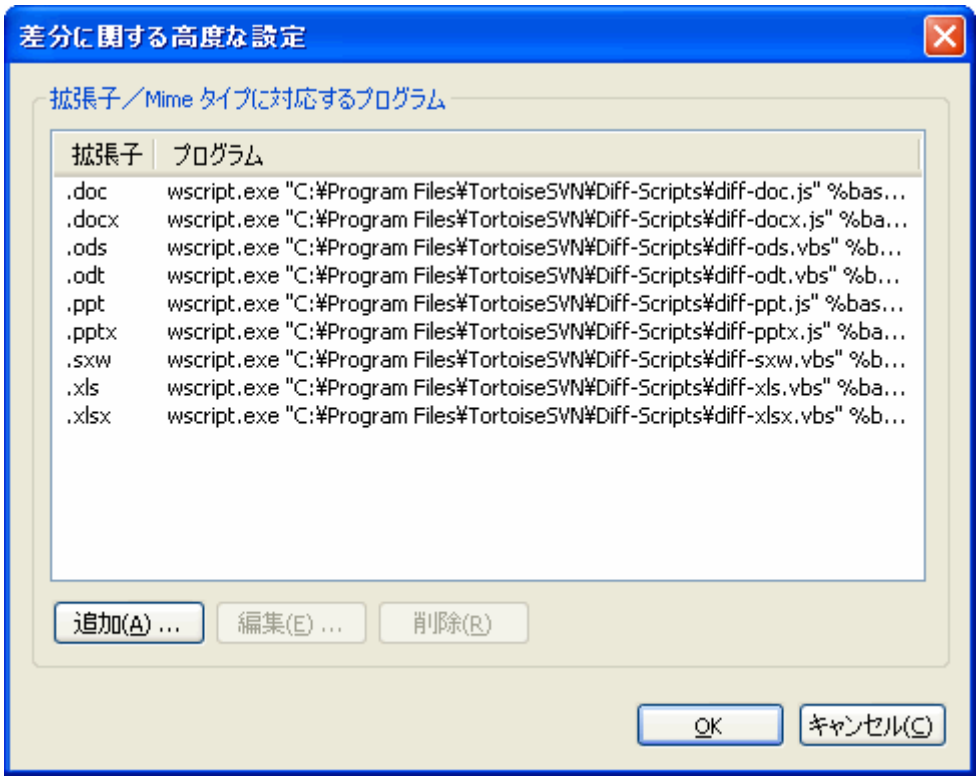


図4.62 設定ダイアログの差分／マージ の高度な設定ダイアログ

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 属性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

unified-diff ファイル (パッチファイル) 用のビューアです。パラメータは必要ありません。**デフォルト** オプションは .diff ファイル、及び .txt ファイルに関連づけがないかどうかをチェックするためのものです。.diff ファイル用のビューアを持っていなければ、おそらくメモ帳 (NotePad) になるでしょう。

オリジナルの Windows メモ帳は 行末が CR-LF でないファイルをうまく扱えません。ほとんどの unified diff ファイルは 行末が LF のため、メモ帳ではうまく表示できません。しかし、無料のメモ帳の代替ソフトウェアをダウンロードできます。Notepad2 [http://www.flos-freeware.ch/notepad2.html] は 行末を正しく表示できるだけでなく、追加・削除した行を、色を変えて表示できます。

4.30.6. 保存データの設定

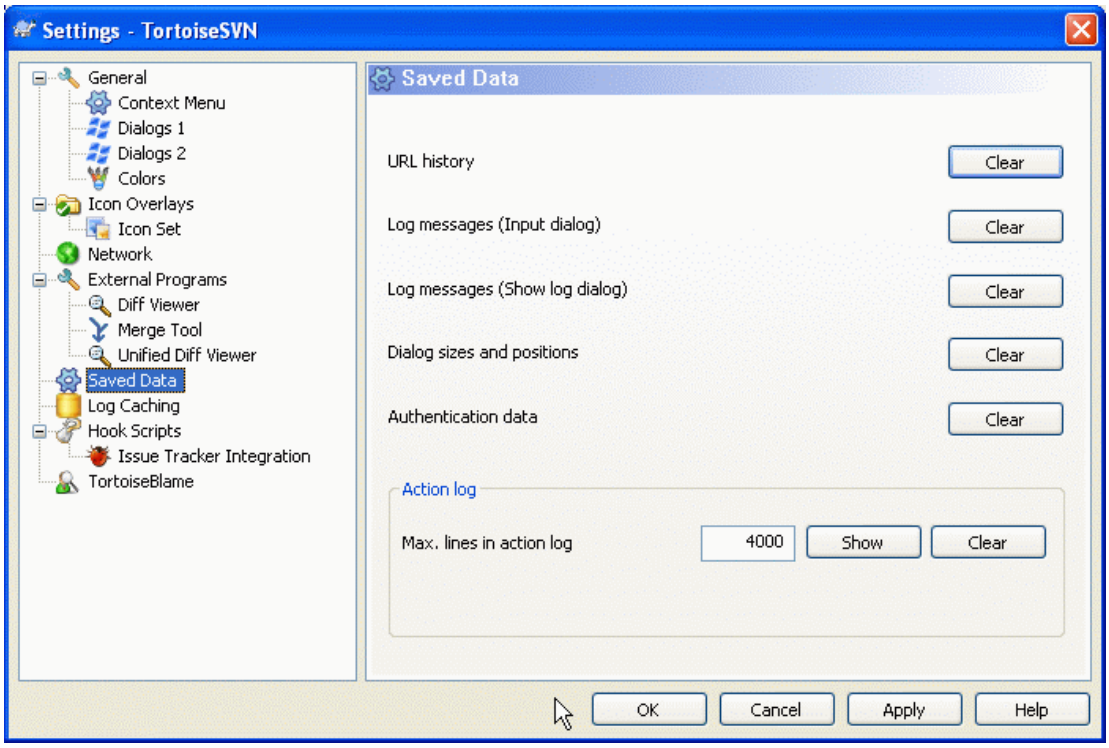


図4.63 設定ダイアログ、保存データページ

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 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 (ログ参照ダイアログ)

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、キャッシュをクリア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 サーバの認証を通れば、ユーザ名とパスワードを毎回入力しなくてもいいように、手元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名でアクセスするのにクリアしたいかもしれません……
ジョンはあなたが彼の PC を使うのを知っていますか?

特定のサーバのみ、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. ログのキャッシュ

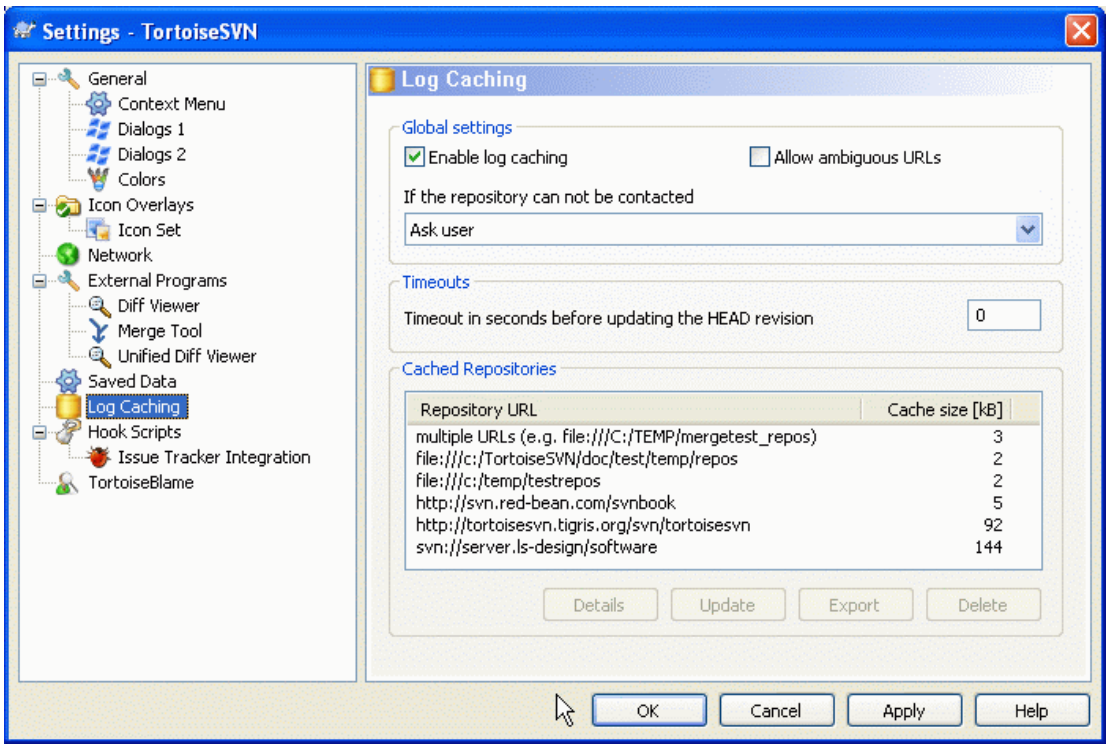


図4.64 設定ダイアログ、ログキャッシュページ

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、リポジトリサーバがダウンしていたりした場合でも、ログキャッシュを利用して、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。

最新の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、サーバに接続します。頻繁にログダイアログを開き、サーバが遅い場合、これによりサーバ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。

小さなキャッシュを削除するためにアクティブでないと判定する日数

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。

削除されたアクティブでないキャッシュの最大サイズ

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。

キャッシュを削除するまでに許容するツール失敗の最大数

時折、キャッシュに何か障害が発生し、それが元でクラッシュ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。それほど安定していない、ナイトリービルドを使用する場合、とにかくキャッシュを保持するため、これを選択できます。

4.30.7.1. キャッシュされたリポジトリ

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

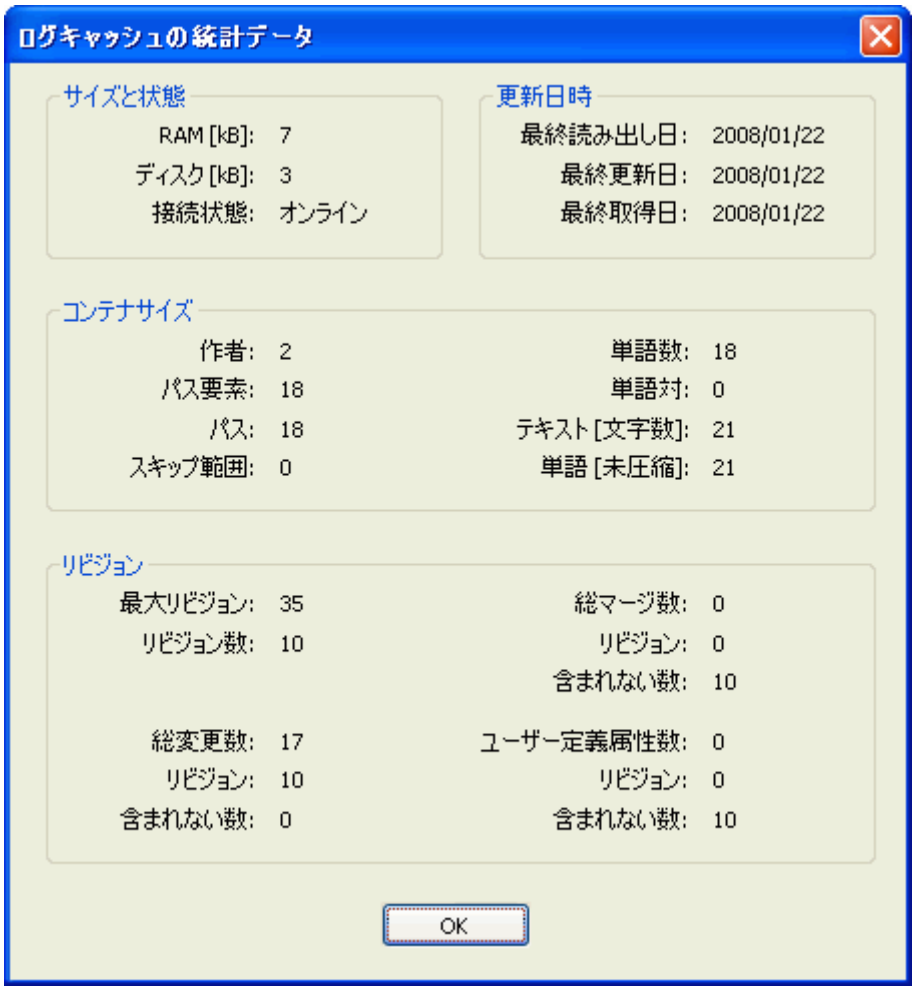


図4.65 設定ダイアログ、ログキャッシュ統計

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時間です。

最終取得日
: サーバから最新リビジョンを最後に取得した時間です。

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。

リビジョンカウント
: 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

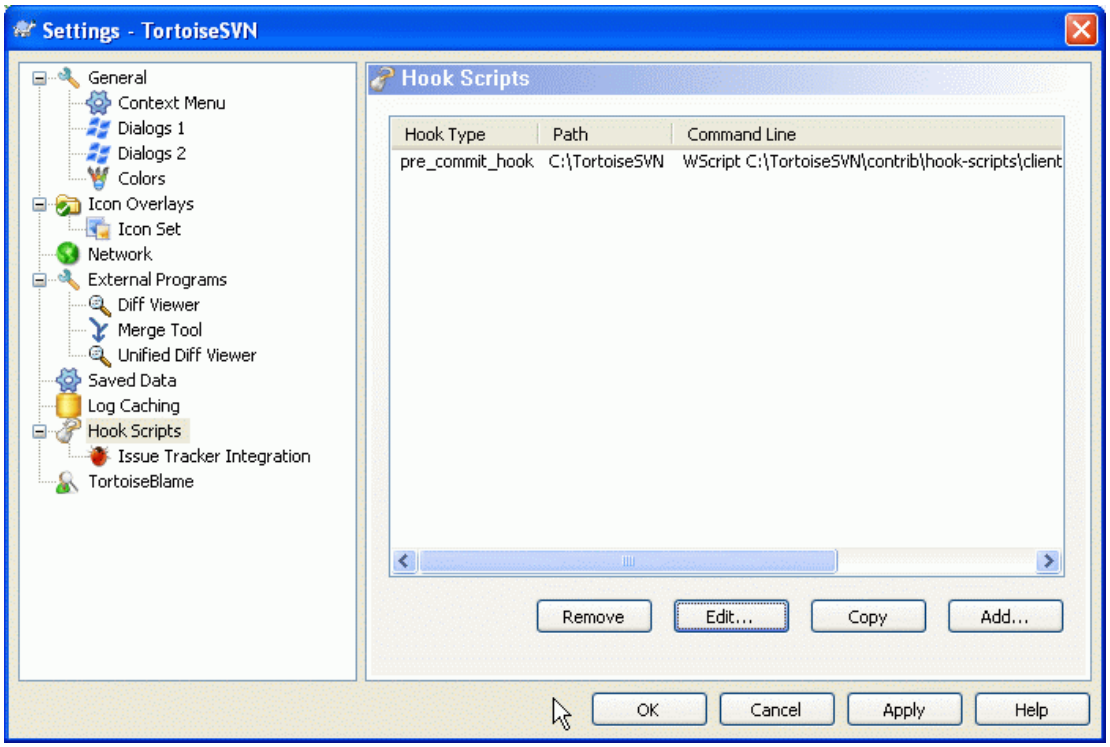


図4.66 設定ダイアログ、フックスクリプトページ

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属性としてではなく、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。

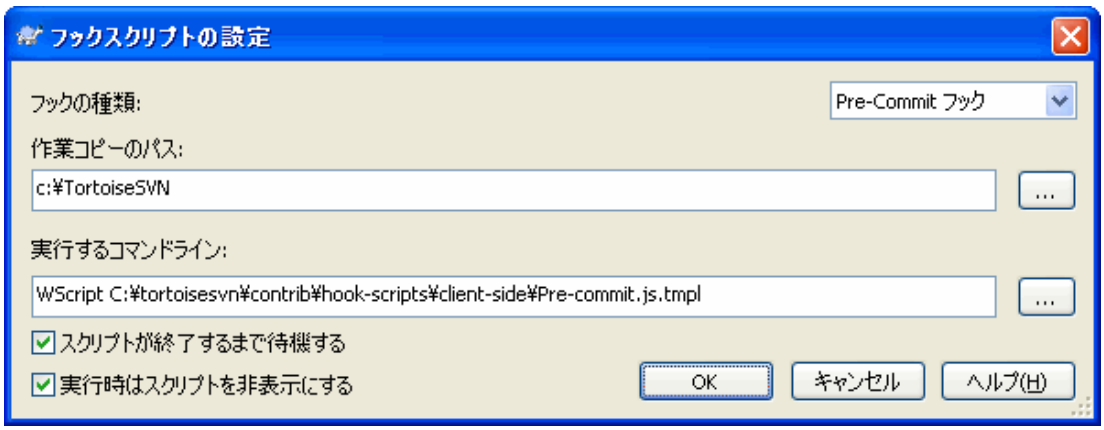


図4.67 設定ダイアログ、フックスクリプトの設定

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
: Subversion の更新を実際に始める前に呼ばれます。

Post-update
: 更新が完了した後で (成功・失敗に関わらず) 呼ばれます。

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 (フックスクリプトや実行ファイルのパスを含む) を指定しなければなりません。バッチファイルや、実行ファイル、その他 Windows に関連付けられたファイル (perl スクリプトなど) を指定できます。

コマンドラインには、TortoiseSVN　が設定するパラメータを含めます。呼ばれるフックによってパラメータは異なります。各フックには以下の順番で渡されるパラメータがあります。

Start-commit
: `PATHMESSAGEFILECWD`

Pre-commit
: `PATHDEPTHMESSAGEFILECWD`

Post-commit
    PATHDEPTHMESSAGEFILEREVISIONERRORCWD

Start-update
    PATHCWD

Pre-update
    PATHDEPTHREVISIONCWD

Post-update
    PATHDEPTHREVISIONERRORCWD

各パラメータ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度です。

    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。

Sample client hook scripts can be found in the `contrib` folder in the TortoiseSVN repository [http://tortoisesvn.googlecode.com/svn/trunk/contrib/hook-scripts]. (「TortoiseSVN は自由!」 explains how to access the repository).

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 プラグインを利用できます。そのようなプラグインの利用を、「課題追跡システムからの情報取得」 で説明しています。あなたのシステム管理者が、すでにあなたが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。

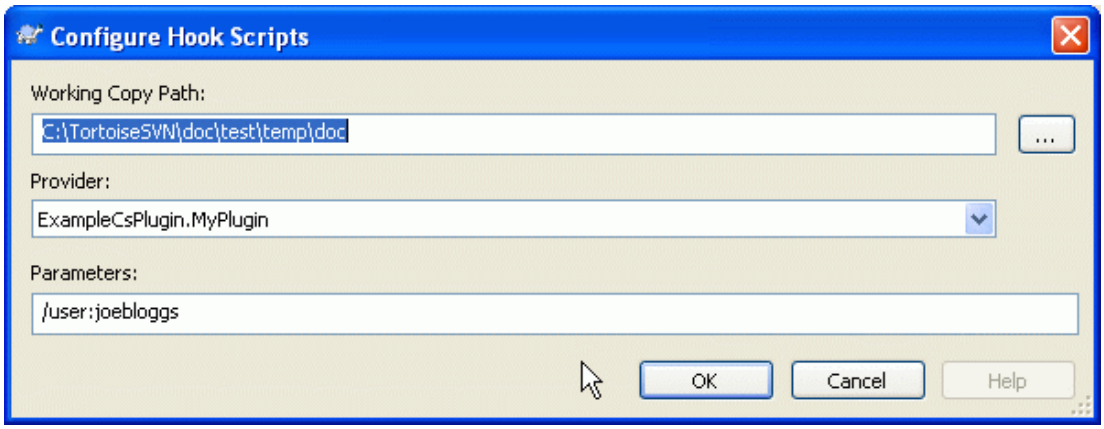


図4.68 設定ダイアログの課題追跡システムとの統合ページ

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の指定ができます。パラメータはプラグインに固有ですが、あなた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名を含めるかも知れません。

すべてのユーザが、あなたのプロジェクトで同じ COM プラグインを使用するようにしたいのであれば、プラグインを `bugtraq:provideruuid` 属性や `bugtraq:providerparams` 属性に指定することもできます。

bugtraq:provideruuid
: このプロパティには、IBugtraqProvider の COM UUID (例、`{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用プロバイダの、Gurtle bugtraq provider [http://code.google.com/p/gurtle/] のUUID です)

bugtraq:providerparams
: この属性は、IBugtraqProvider に渡すパラメータを指定します。

このふたつの属性に何を指定すればよいかは、IBugtraqProvider プラグインのドキュメントをチェックしてください。

4.30.9. TortoiseBlame の設定

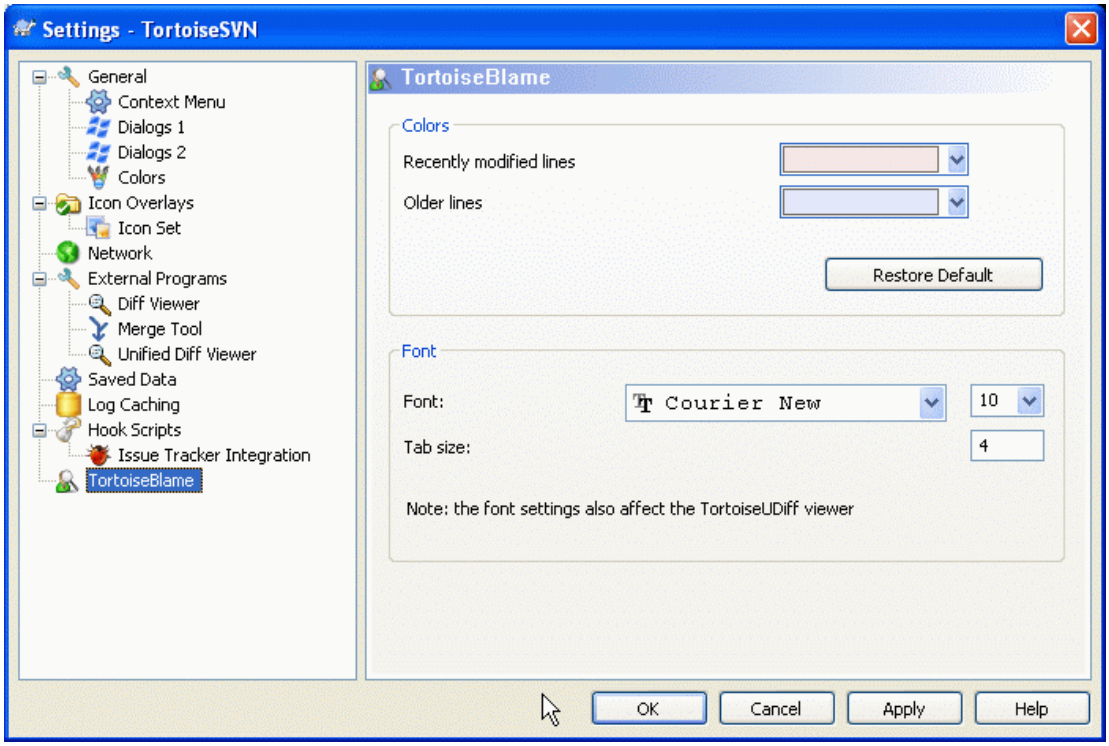


図4.69 設定ダイアログ、TortoiseBlame ページ

TortoiseBlame で使用する設定は、メインのコンテキストメニューから制御できますが、TortoiseBlame 自体では直接できません。

色
: TortoiseBlame は、ファイルにある行の歴史を背景色で表現できます。ここでは、最新と最古の色を指定して両端を設定でき、TortoiseBlame はこの 2 色を補間して、行ごとのリポジトリのリビジョンを表現します。

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。ここで指定したものを、ファイルの内容と、左ペインに表示する作者とリビジョンの両方に使用します。

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. レジストリの設定

まれに使われる設定は、レジストリを直接編集するしか方法はありません。何をしているのか明確にわかっている場合に限り、レジストリの値を編集してください。

設定
: レジストリの `HKCU¥Software¥TortoiseSVN¥ConfigDir` を使用して、Subversion 設定ファイルの場所を異なる場所に指定できます。これは TortoiseSVN のすべての動作に影響を与えます。

キャッシュトレイアイコン
: TSVNCache プログラムのキャッシュトレイアイコンを追加するため、`HKCU¥Software¥TortoiseSVN¥CacheTrayIcon` に `DWORD` キーを値 1 で作成してください。これによりプログラムを作法に則って終了させます。そのため実際には開発者のみに有用です。

デバッグ

シェル拡張から TortoiseProc.exe に渡されたコマンドラインパラメータを表示するには、HKCU¥Software¥TortoiseSVN¥Debug に DWORD のキーを作成し、値を 1 としてください。

コンテキストメニューアイコン

これは Windows エクスプローラ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。TortoiseSVN にシェルのコンテキストメニューのアイコンを表示させたくない場合、HKCU¥Software¥TortoiseSVN¥ShowContextMenuIcons に DWORD のキーを値 0 で作成してください。またアイコンを表示する場合は、この値を 1 としてください。

状態オーバーレイの抑止

他の TortoiseSVN コマンド (更新、コミットなど) を実行している間、エクスプローラの状態オーバーレイアイコンを更新したくない場合、HKCU¥Software¥TortoiseSVN¥BlockStatus に DWORD のキーを作成し、値を 1 としてください。

更新チェック URL

HKCU¥Software¥TortoiseSVN¥UpdateCheckURL には TortoiseSVN の更新があった場合にダウンロードするテキストファイルの URL が格納されています。必要に応じて HKCU を HKLM に変更することもできますが、HKLM の設定を HKCU は上書きしてしまいます。企業の管理者が承認するまで TortoiseSVN の更新をさせないようにするのに便利だと思います。

自動補完リストのファイル名からの拡張子省略

コミットメッセージエディタの自動補完リストには、コミットするファイル名が表示されます。この名前から拡張子を除くには、HKCU¥Software¥TortoiseSVN¥AutocompleteRemovesExtensions に DWORD のキーを作成し、値を 1 としてください。

常にエクスプローラの列を表示

Windows エクスプローラの詳細ビューに表示する、TortoiseSVN が追加する列は、通常、作業コピーのみに有効です。作業コピーのみではなく、どこでもこの列を表示するには、HKCU¥Software¥TortoiseSVN¥ColumnsEveryWhere に DWORD キーを作成し、値を 1 としてください。

マージログの区切り

別のブランチから、あるリビジョンをマージした際、マージ追跡情報が有効だと、そのリビジョン由来のログメッセージをまとめて、コミットログメッセージにします。変更するには、HKCU¥Software¥TortoiseSVN¥MergeLogSeparator に SZ キーを作成し、お好みの文字列を設定してください。

変更の注釈履歴に常時 TortoiseMerge を使用

TortoiseSVN は外部 diff ビューアを使用できます。しかし、ほとんどのそのようなビューアは注釈履歴 (「注釈履歴の差分」) には対応していません。そのため、このような場合に TortoiseMerge にフォールバックしたくなるかと思います。それには、HKCU¥Software¥TortoiseSVN¥DiffBlamesWithTortoiseMerge に DWORD キーを作成し、値を 1 としてください。

ログダイアログでフォルダの現在のバージョンも強調

ログダイアログでは、ファイルのログを表示する際、現在の作業コピーのリビジョンを強調しています。フォルダに対して同じことをするには、作業コピーに対してクロールしなければなりませんが、作業コピーが大きいと、この操作に時間がかかります。この機能を有効にするには、HKCU¥Software¥TortoiseSVN¥RecursiveLogRev に DWORD のレジストリキーを、作成しなければなりません。0 に設定するとこの機能を無効にします (フォルダの強調なし)。1 (デフォルト) に設定すると、状態を再帰的に取得します (作業コピーツリーの最新リビジョンを検索)。2 に設定すると、選択したフォルダ自体のリビジョンをチェックしますが、子項目のチェックはしません。

同名の項目がある場合のチェックアウト失敗

デフォルトでは、インポート直後など、既存のフォルダ構造上に作業コピーをチェックアウトすると、リポジトリの内容と差異がある既存の項目は上書きされず、変更マークが付きます。コミットの際には、このローカルの内容がリポジトリに送信されることになります。ふたりの人が同じファイルを追加した場合、後でコミットした方がオリジナルバージョンを上書きしてしまわないように、既存の内容と差異がある場合は、チェックアウトが失敗するべきだと思う人々もいます。この例のように強制的にチェックアウトが失敗するようにする場合、`HKCU¥Software¥TortoiseSVN¥AllowUnversionedObstruction` に `DWORD` レジストリキーを 値 0 で作成しなければなりません。

4.30.11. Subversion の作業フォルダ

web プロジェクトを使用する際、Subversion が内部情報を格納する `.svn` フォルダを、VS.NET 2003 は扱えません。これは Subversion のバグではありません。VS.NET 2003 とそこで使用する frontpage 拡張のバグです。

> 注: このバグは VS2005 以降で修正されています。

Subversion の 1.3.0 と TortoiseSVN では、環境変数 `SVN_ASP_DOT_NET_HACK` を設定できます。この環境変数が設定されていると、Subversion は `.svn` フォルダではなく `_svn` フォルダを使用します。環境変数が効果を現すにはシェルを再起動しなければなりません。通常 PC を再起動することを意味します。簡単にできるように、設定ダイアログの一般ページにあるチェックボックスで設定できます。「一般設定」をご覧ください。

詳細情報や、その他のこの問題を避ける第一の方法については、私たちの FAQ [http://tortoisesvn.net/aspdotnethack] で関係する記事を調べてください。

4.31. 最終ステップ

寄付!

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、コンピュータの前で過ごす私たちを限りなく応援できます。

私たちは TortoiseSVN の作業中、音楽を聴くのが大好きです。また長時間このプロジェクトに関わっているために、音楽がたくさん必要です。そこで、私たちの好きな CD や DVD の欲しいものリストを用意しました。http://tortoisesvn.tigris.org/donate.html パッチや翻訳を送ってプロジェクトに貢献している人の一覧もご覧ください。

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利用します。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 は作業コピーの全ファイルの Subversion 状態を読み込みます。デフォルトでは外部参照は除外します。検出した最も大きいコミットリビジョン番号と、そのタイムスタンプを記録します。また、作業コピーの手元の変更点か更新リビジョンが混ざっているのも記録します。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のいくつかを解説します。複数のオプションを使いたい場合、シングルグループで指定しなければなりません。例: `-n -m`ではなく `-nm` とします。

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
$WCREV$	作業コピーのもっとも大き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

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 インターフェース

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 インターフェースを使用できます。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ー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);
```

第6章 IBugtraqProvider インターフェース

シンプルな bugtraq: 属性を使用するよりも、より密接に課題管理システムと統合するため、TortoiseSVN は COM プラグインを利用できます。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利用することでユーザとやりとりし、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。

お好みの言語で 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、リポジトリの contrib/issue-tracker-plugins フォルダに、C++/ATL や C# で書かれたサンプルプラグインがあります。このフォルダにはプラグインをビルドするために必要な include ファイルもあります (リポジトリにアクセスする方法は「TortoiseSVN は自由!」で説明しています)。

6.1. IBugtraqProvider インターフェース

TortoiseSVN 1.5 では、IBugtraqProvider インターフェースを実装したプラグインを利用できます。このインターフェースは、プラグインが課題管理システムとやりとりするのに使用する、いくつかのメソッドを提供しています。

```
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、ユーザがプラグインを追加し、設定した設定ダイアログから呼ばれます。parameters 文字列は、課題管理システムの URL やログイン情報などの必要な追加情報を取得するため、プラグインが使用します。プラグインは parameters 文字列を検証し、不正であればエラーダイアログを表示するべきです。hParentWnd パラメータは、プラグインが表示するダイアログで、親ウィンドウとして使われます。parameters 文字列の検証が成功したら、プラグインは TRUE を返さねばなりません。プラグインが FALSE を返すと、作業コピーパスへのプラグインの追加は失敗します。

```
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
);
```

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で使用する文字列 ("Choose issue" や "Select ticket") をここで提供できます。文字列は長すぎないようにしてください。そうでないとボタンに入り切りません。このメソッドがエラー (E_NOTIMPL など) を返すと、デフォルトの文字列をボタンに使用します。

```
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
);
```

これがプラグインのメインメソッドです。TortoiseSVN のコミットダイアログで、ユーザがプラグインボタンをクリックすると、このメソッドが呼ばれます。parameters 文字列は、設定ダイアログで、プラグインの設定をしたときに入力してあった文字列になります。プラグインは通常、課題管理システムの URL やログイン情報、もしくはさらにその他のものを、このパラメータから取得します。commonRoot 文字列には、コミットダイアログが表示するために選択した、すべての項目の上位パスが格納されています。コミットダイアログでユーザが選択した項目すべてのルートパスは含まれていないことにご注意ください。pathList パラメータには、コミットするとユーザが選択したパス (文字列) の配列を格納しています。originalMessage パラメータには、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を格納しています。ユーザがまだテキストを入力していない場合は、空文字列となります。newMessage 返却文字列は、既に入力されている文字列を置き換えて、ログメッセージエディットボックスにコピーされます。プラグインが originalMessage 文字列を変更しない場合、同じ文字列を返さなければなりません。そうしないと、ユーザが入力したテキストが失われてしまいます。

6.2. IBugtraqProvider2 インターフェース

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ースを追加しました。この IBugtraqProvider2 インターフェースは 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
);
```

TortoiseSVN のコミットダイアログで、ユーザがプラグインボタンをクリックすると、このメソッドが呼ばれます。このメソッドは GetCommitMessage() の代わりに呼び出します。ここでも使用するパラメータの説明は、GetCommitMessage のドキュメントを参照してください。commonURL パラメータは、コミットダイアログを表示するために選択した、すべてのファイルの親 URL です。これは基本的に commonRoot パスの URL になります。bugID パラメータは bug-ID フィールド (表示する場合、bugtraq:message 属性とともに設定) の内容が格納されています。返却パラメータ bugIDOut には、メソッドが返却する際の bug-ID フィールドを入れるのに使用します。revPropNames と revPropValues には、コミット時に設定するリビジョン属性の、名前・値の組を配列で格納しています。プラグインは、双方の配列のサイズが同じになるようにしなければなりません! revPropNames にある各属性名は、revPropValues にも、対応する値を持たねばなりません。リビジョン属性が設定されない場合、プラグインは空配列を返さなければなりません。

```
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を閉じ、コミットが始まる前に、確実に呼ばれます。プラグインは、コミットするよう選択したファイル・フォルダやユーザが入力したログメッセージの検証に、このメソッドを利用できます。パラメータは GetCommitMessage2() と同じです。違いがあるのは、commonURL が今度は、すべてのチェックした項目の共通 URL であることと、commonRoot がすべてのチェックした項目のルートパスであることです。返却パラメータ errorMessage

は、TortoiseSVN がユーザに表示するエラーメッセージか、コミットが始まるのであれば空文字列でなければなりません。エラーメッセージが返されると、TortoiseSVN はエラー文字列をダイアログに表示し、誤りを修正できるようにコミットダイアログを表示したままにします。そのためプラグインは、何が問題で、どのように修正するのかを説明するような、エラーメッセージを返すべきです。

```
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成功後に呼ばれます。プラグインは、選択した課題をクローズしたり、課題に対してコミットについての情報を追加したりするのに使用できます。パラメータは GetCommitMessage2 と同じです。

```
HRESULT HasOptions(
  // Whether the provider provides options
  [out, retval] VARIANT_BOOL *ret
);
```

このメソッドは、プラグインの設定を行う設定ダイアログから呼ばれます。プラグインが ShowOptionsDialog で独自の設定ダイアログを提供する場合、ここで必ず TRUE を返します。そうでなければ FALSE を返さなければなりません。

```
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から、HasOptions が TRUE を返したときに表示される "オプション" ボタンを、ユーザがクリックしたときに呼ばれます。ユーザがプラグインの設定を簡単に行えるように、オプションダイ

アログを表示できます。parameters 文字列は、既に設定・入力されたプラグインパラメータを格納しています。newparameters 返却パラメータは、オプションダイアログで集めた情報を、プラグインが構成したパラメータ文字列を格納しなければなりません。paramameters 文字列は、IBugtraqProvider・IBugtraqProvider2 の他のすべてのメソッドに渡されます。

付録A よくある質問 (FAQ)

TortoiseSVN は常に開発中のため、時々完全にはドキュメントを最新に保守できないことがあります。そこで、TortoiseSVN メーリングリスト <dev@tortoisesvn.tigris.org> や <users@tortoisesvn.tigris.org> でされた質問から選んだ内容を、オンライン FAQ [http://tortoisesvn.tigris.org/faq.html] で管理しています。

また私たちは、プロジェクトの 課題追跡システム [http://issues.tortoisesvn.net] を運用しており、To-Do リストや既に修正したバグの情報を公開しています。バグを見つけたり新機能のリクエストをしたいときには、まずここをチェックして、誰かが既にやっていないか確認してください。

回答がない疑問がある場合、質問する最善の場所はメーリングリストです。<users@tortoisesvn.tigris.org> は TortoiseSVN を使用する上で疑問がある場合に使用します。TortoiseSVN を開発する上での手助けが必要なら、<dev@tortoisesvn.tigris.org> での議論に参加するべきです。

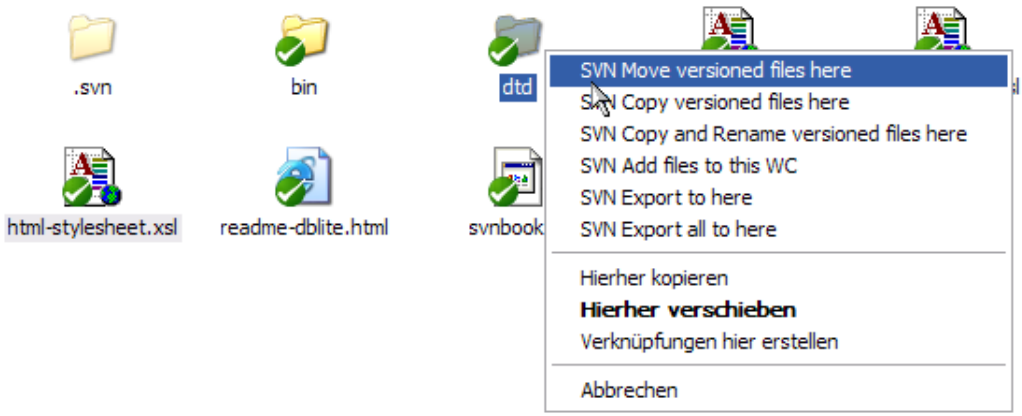
付録B どうしたら……

この付録には TortoiseSVN を使用する上での 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。移動・コピーしたいファイルの上で、単に 右クリック したままにしてください。そのまま新しい場所にドラッグし、マウスボタンを放します。現れたコンテキストメニューで、コンテキストメニュー → ここの Subversion にコピー なり、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コピーする を選択してください。



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に直接アクセスできる必要があります。

B.2.1. サーバ上のフックスクリプト

リポジトリサーバ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。

サーバのリポジトリフォルダには、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、コミットがサーバによって拒否され、ユーザ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. プロジェクト属性

TortoiseSVN は機能を制御するのに属性を使用します。そういった属性のひとつに、tsvn:logminsize 属性があります。

フォルダにこの属性をセットすると、属性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。

TortoiseSVN が属性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。

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。手元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利用

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新しい HEAD にするのに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を選択するべきです。

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するのに **すべて取得** や **次の 100** を使用する必要があるでしょう。

3. 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元に戻す** 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新しい 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利用

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 URL を入力してください。デフォルト 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、最新 (HEAD) リビジョンを利用できます。

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 の利用

TortoiseSVN は決してデータを失わないため、「巻き戻した」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/1.5/svn.reposadmin.maint.html] にあります。(訳注: 日本語訳は Subversion によるバージョン管理 [http://subversion.bluegate.org/doc/ch05s03.html#svn.reposadmin.maint.tk.svndumpfilter] にあります)

B.5. ファイルやフォルダに対して 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 → 最新のリビジョンを比較** 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 → 最新の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 の利用

プロジェクトにあるフォルダに `svn:externals` 属性をセットしてください。この属性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として使用するサブフォルダ名と、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をコミットしたときに、外部プロジェクトに行った変更もコミットされてしまいます。

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。一度フォルダ属性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を取得します。

B.6.2. ネストした作業コピーの利用

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. 相対位置の利用

同じ共通コアコードを複数のプロジェクトで使用するけれど、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 の環境でのみ動作します。また、謎なファイルがどこにあるかチームが知るために、必要なドライブマッピングのドキュメントが必要です。厳密に言えばこの方法は、閉じた開発環境用です。一般的にはお勧めしません。

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ースを使用して、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。

まだコミットしていなければ、TortoiseSVN → 取り消し... で追加を取り消せば済みます。また間違えて追加しないように、そのファイルを無視リストに追加しておくといいでしょう。

ファイルが既にリポジトリに入ってしまっていたら、もう少し手を動かさなければなりません。

1. Shift キーを押しながら拡張コンテキストメニューを表示し、TortoiseSVN → 削除 - ローカルファイルを保持 を使用して、ローカルコピーを消さずにマークしたリポジトリのファイル・フォルダを削除してください。

2. 親フォルダで、TortoiseSVN → コミット としてください。

3. また同じトラブルに見舞われないように、そのファイル・フォルダを無視リストに追加します。

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

もう必要なくなった作業コピーがある場合、きれいに取り除くにはどのようにするのでしょう? 簡単です。単純に Windows エクスプローラで削除してください! 作業コピーはプライベートな存在で、それ自体にすべて含んでいます。

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

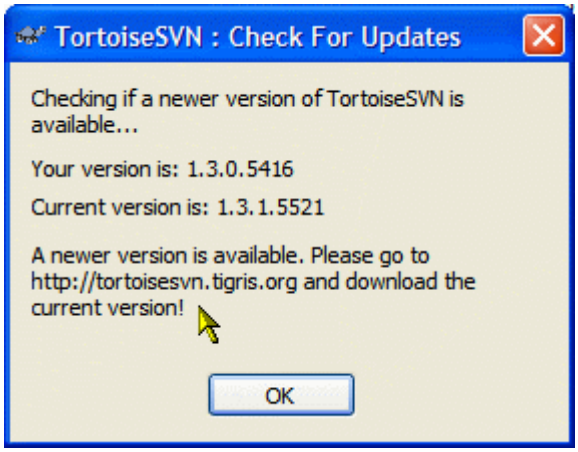
TortoiseSVN インストーラは MSI ファイルとなっています。これはドメインコントローラのグループポリシーに対して MSI ファイルを追加する際に、問題が発生しないことを意味しています。

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 のバージョン 1.3.0 以降では、User Configuration ではなく Computer Configuration のもとで、インストールしなければなりません。これは、このバージョンから使用する、新しい CRT/MFC DLL が ユーザごと ではなく、コンピュータごと にデプロイされるからです。TortoiseSVN を本当にユーザごとにインストールするのを原則としなければならないのなら、まずコンピュータごとに、MFC と CRT パッケージ バージョン 8 を Microsoft からインストールし、TortoiseSVN をユーザごとにインストールしてください。

C.2. 更新チェックのリダイレクト

TortoiseSVN は数日おきに新バージョンが有効でないかチェックします。新バージョンが有効なら、その旨ユーザに伝えるダイアログを表示します。



図C.1 アップグレードダイアログ

自分のドメインにいるたくさんのユーザ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 バージョン 1.4.0以降では、自分のイントラネットサーバにアップグレードチェックをリダイレクトできます。レジストリキー HKCU¥Software¥TortoiseSVN¥UpdateCheckURL (文字列値) に自分のイントラネットにあるテキストファイルを指す URL をセットできます。テキストファイルは以下の形式になります。

```
1.4.1.6000
A new version of TortoiseSVN is available for you to download!
```

```
http://192.168.2.1/downloads/TortoiseSVN-1.4.1.6000-svn-1.4.0.msi
```

ファイルの最初の行はバージョン文字列です。TortoiseSVN インストールパッケージのバージョン文字列とピッタリ一致しなくてはなりません。2 行目は、アップグレードダイアログに表示するカスタムテキストです。ご希望のテキストをなんでも書けます。アップグレードダイアログの大きさには限りがあることだけ注意してください。長すぎる文字列は切り取られてしまいます! 3 行目は、新しいインストールパッケージの URL です。アップグレードダイアログのカスタムメッセージラベルをクリックすると、この URL を開きます。msi ファイルを直接指定する代わりに、Web ページを指定しておくこともできます。Web ページを指定しておくと、URL をデフォルト Web ブラウザで開き、ユーザが見ることになります。msi パッケージを指定しておくと、ブラウザは msi ファイルを保存するかどうか、ユーザに確認してきます。

C.3. SVN_ASP_DOT_NET_HACK 環境変数の設定

バージョン 1.4.0 以降では、TortoiseSVN のインストーラは SVN_ASP_DOT_NET_HACK 環境変数を設定するオプションを、もうユーザに提供しません。このオプションが何をするためのものなのか知っているにもかかわらず、常に すべて インストールするユーザを混乱させ、問題を引き起こすからです。

しかし、このオプションはユーザに対して隠されているだけです。TortoiseSVN のインストーラにこの環境変数を設定するように強制するには、ASPDOTNETHACK 属性を TRUE にしてください。例えば、インストーラ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
```

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がいる場合に、そのコマンドを無効にできます。また、エクスプローラ のコンテキストメニューエントリを隠しますが、他の方法では実行できることに注意してください。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
0x0000000000000000	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

TortoiseSVN の全コマンドはコマンドラインパラメータから制御できるため、バッチスクリプトや、他のプログラム (例: お好みのテキストエディタ) から指定したコマンドやダイアログを起動するのを自動化できます。

重要

TortoiseSVN は GUI クライアントであることを忘れないでください。またこの自動化ガイドは、ユーザの入力を集めるのに TortoiseSVN ダイアログを表示させる方法を説明しています。入力を全くさせないようなスクリプトを書く場合は、公式 Subversion コマンドラインクライアントの方を使用してください。

D.1. TortoiseSVN コマンド

TortoiseSVN GUI プログラムは TortoiseProc.exe で呼び出されます。すべてのコマンドで /command:abcd パラメータを指定できます。abcd にはコマンド名が必要です。ほとんどのコマンドは /path:"some¥path" という形で、少なくともひとつのパスを引数に渡す必要があります。以下の表で、/command:abcd パラメータに渡すコマンドと、/path:"some¥path" パラメータに渡すパスを説明しています。

いくつかのコマンドは対象のパスを複数与えられる (例: コミット時に複数ファイルを指定) ので、/path パラメータには * で区切って複数のパスを与えます。

TortoiseSVN は、シェル拡張とメインプログラムの間で複数の引数を渡すのに、一時ファイルを使用します。TortoiseSVN 1.5.0 以降から、/notempfile パラメータはサポートされなくなり、追加する必要はなくなります。

コミットや更新、その他たくさんのコマンドで使用する進行ダイアログは、通常 OK ボタンを押すまで、コマンドが終了した後も開いたままになります。設定ダイアログにある該当オプションをチェックして、動作を変更できます。しかしそれでは、バッチファイルからコマンドを起動しても、TortoiseSVN コンテキストメニューから起動しても、進行ダイアログは閉じてしまいます。

異なる場所にある設定ファイルを指定するには、/configdir:"path¥to¥config¥directory" パラメータを使用してください。レジストリに設定されているデフォルトパスを上書きします。

永続的な設定を行わずに、コマンド終了時に進行ダイアログを自動で閉じるには、/closeonend パラメータを渡すことで行えます。

- /closeonend:0 自動でダイアログを閉じません。
- /closeonend:1 エラーがなければ自動で閉じます。
- /closeonend:2 エラーや競合がなければ自動で閉じます。
- /closeonend:3 エラー、競合、マージがなければ自動で閉じます。
- /closeonend:4 エラー、競合、マージが手元の操作で起きなければ自動で閉じます。

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、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:abcd の形で使用してください。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	about ダイアログを表示します。コマンドを与えなくても表示します。

表D.1 使用できるコマンドとオプションの一覧

サンプル (一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:"test log message" /closeonend:0

TortoiseProc.exe /command:update /path:"c:¥svn_wc¥" /closeonend:0

TortoiseProc.exe /command:log /path:"c:¥svn_wc¥file1.txt"
                 /startrev:50 /endrev:60 /closeonend:0
```

D.2. TortoiseIDiff 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ー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ース (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 コマンド

E.2.1. チェックアウト

```
svn checkout [-N] [--ignore-externals] [-r rev] URL PATH
```

トップフォルダのみチェックアウトする がチェックされているなら `-N` スイッチを使用してください。

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 について更新します。

ここでのコマンドラインオプションはありません。**特定のリビジョンへ更新** 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] [-N] [--ignore-externals] PATH
```

トップフォルダのみ更新する がチェックされているなら -N スイッチを使用してください。

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する 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、Subversion のコミットをそこで行います。すべてのファイルの選択チェックボックスを初期状態からはずすと、TortoiseSVN は作業コピーに対し再帰的コミットを一度行います。ファイルを選択しない場合、コミットコマンドラインで個々のパスを指定して、非再帰的コミット (-N) を使用しなければなりません。

```
svn commit -m "LogMessage" [-N] [--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 Diff を使用する場合、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 CLI コマンドの出力では比較を行い、結果 をunified-diff 形式で出力します。しかし、TortoiseSVN がこれ

を使用しているわけではありません。TortoiseSVN は TortoiseMerge (や、お好みの diff 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、直接 CLI で相当するものはありません。

TortoiseSVN で、バージョン管理下であるかどうかはともかく、任意の 2 ファイルを比較することもできます。TortoiseSVN は、2 つのファイルを選択した diff プログラムに単に渡し、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する 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

```
svn info URL_of_WC
svn list [-r rev] -v URL
```

リポジトリブラウザ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 diff/merge ツールを起動します。

E.2.11. 解消

```
svn resolved PATH
```

E.2.12. 名前の変更

```
svn rename CURR_PATH NEW_PATH
```

E.2.13. 削除

```
svn delete PATH
```

E.2.14. 取り消し

```
svn status -v PATH
```

第一段階として、作業コピー内で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、Subversion の revert が行われます。ファイル選択チェックボックスをすべてデフォルトのままにした場合、TortoiseSVN は 再帰 (-R) revert を作業コピーに対して行います。ファイルの選択を外すと、revert コマンドラインに選択したパスをすべて、個々に指定しなければなりません。

```
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. ロックの解放

```
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。

- リポジトリの最新 (HEAD) 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 merge で使用されるものを diff 操作で示します。

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からアクセスし、エクスポート先に指定フォルダを使用します。

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 属性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. パッチの作成

```
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 diff 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になっていなければ、パッチの適用はやりにくい作業です。運良く TortoiseMerge を使用できます。TortoiseMerge は Subversion に該当する機能がありません。

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。

修正 オーバーレイアイコンは、ローカルで変更がある**修正**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 オーバーレイアイコンは、`svn:needs-lock` 属性を設定しているファイルに表示されます。Subversion 1.4.0 以降で作成した作業コピー向けに、`svn:needs-lock` 状態は、Subversion がローカルにキャッシュし、このオーバーレイアイコンを表示するかどうか決めるにに使用します。1.4.x 以前の形式の作業コピー向けには、TortoiseSVN は読込専用ファイルにこのオーバーレイアイコンを表示します。Subversion は自動的に作業コピーをアップグレードしますが、`svn:needs-lock` 属性のキャッシュは、そのファイル自体の更新が発生するまで更新されません。

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、**無視** 状態の項目を、共通無視リストにあるか、親フォルダに svn:ignore 属性が設定されているかにかかわらず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 が状態を再帰的に表示する (デフォルト) 際、各フォルダは、自身の状態と、そこにある全ての子ファイルの状態を反映させたオーバーレイアイコンを表示します。ひとつにまとめたオーバーレイアイコンを表示するのに、前述の優先順位を使用します。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。

- 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。
- すでにロードされたオーバーレイが 13 個未満であるときのみ ロック をロードします。スロットが足りなければ 通常 にフォールバックします。
- すでにロードされたオーバーレイが 14 個未満であるときのみ 追加 をロードします。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 SSH を用いた安全な svnserve

本節は、Subversion と TortoiseSVN で svn+ssh プロトコルを使用するよう設定する段階的ガイドです。サーバにログインするのに、すでに認証済み SSH 接続を使用している場合、すでに本節で説明する段階に達しています。さらなる詳細は、Subversion book で見つかるでしょう。まだ SSH を使用しておらず、Subversion を保護するために SSH を使用したい場合は、Subversion のユーザごとに、独立した SSH のユーザアカウントをサーバに用意しなくてもすむ、簡単な方法をガイドします。

この実装では、全 Subversion のユーザ用に、ひとつの SSH ユーザアカウントを作成し、実際の Subversion ユーザで分けるのに、異なる認証鍵を使用します。

この付録では、すでに Subversion のツールをインストールしてあり、このマニュアルですでに述べたようなリポジトリを作成してあると仮定しています。SSH とともに使用する場合、svnserve をサービスやデーモンとして起動するべきでないということに注意してください。

ここの情報の多くは、Marc Logemann が提供しているチュートリアル (www.logemann.org [http://www.logemann.org/2007/03/13/subversion-tortoisesvn-ssh-howto/] にあります) から来ています。Windows サーバに設定するさらなる情報は、Thorsten Müller が提供しています。ありがとうございます!

G.1. Linux サーバの設定

サーバで SSH を有効にする必要があります。また、おそらく OpenSSH を使用していると思います。ほとんどのディストリビューションには、これがすでにインストールされていると思います。以下のように入力して ssh のジョブを探してください。

```
ps xa | grep sshd
```

注意すべき点は、Subversion をソースからビルドする際、./configure に引数を何も与えない場合、Subversion は /usr/local に bin ディレクトリを作成し、そこにバイナリを配置するということです。SSH でのトンネリングモードを使用する場合、ユーザが SSH でログインするには、svnserve プログラムやその他のバイナリを、実行する必要があるということを意識しなければなりません。このため、PATH 変数に/usr/local/bin を含めるか、/usr/sbin ディレクトリやその他の PATH にある共通のディレクトリに、バイナリへのシンボリックリンクを作成するかを行う必要があります。

すべて問題ないか確認するには、SSH で対象ユーザとしてログインし、以下を入力してください。

```
which svnserve
```

このコマンドは、svnserve に到達できるかを確認します。

svn リポジトリにアクセスする際に使用する、新しいユーザを追加してください。

```
useradd -m svnuser
```

必ず、リポジトリへのすべてのアクセス権を与えてください。

G.2. Windows サーバの設定

Cygwin SSH daemon を、http://pigtail.net/LRP/printsrv/cygwin-sshd.html で説明しているようにインストールしてください。

リポジトリにアクセスするのに使用する、Windows のユーザアカウント svnuser を作成してください。必ず、リポジトリへのすべてのアクセス権を与えてください。

まだパスワードファイルがない場合、Cygwin コンソールから以下のように作成できます。

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

G.3. TortoiseSVN と併せて使用する SSH クライアントツール

Windows クライアントで SSH を使用するのに必要なツールを、http://www.chiark.greenend.org.uk/~sgtatham/putty/ から入手してください。download セクションへ行き、Putty, Plink, Pageant, Puttygen を取得してください。

G.4. OpenSSH 証明書の作成

次のステップは、認証用の鍵ペアの作成です。鍵を作成するには 2 通りの方法があります。ひとつはクライアント上で PuTTYgen を用いて鍵を作成してから、公開鍵をサーバに送信し、また秘密鍵を PuTTY で使用する方法です。もうひとつは、OpenSSH ツールの ssh-keygen を用いて鍵ペアを作成し、秘密鍵をクライアントにダウンロードして、秘密を PuTTY スタイルの秘密鍵に変換する方法です。

G.4.1. ssh-keygen を用いた鍵の作成

サーバに root か svnuser でログインして以下を入力してください。

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

実際の (あなたしか知らない) パスフレーズと鍵ファイルに置換してください。ここでは、1024 ビット鍵フレーズの SSH2 DSA 鍵を作成します。以下のように入力すると、

```
ls -l keyfile*
```

keyfile と keyfile.pub の2 つのファイルがあることでしょう。お察しの通り、.pub ファイルは公開鍵で、もう一つは秘密鍵です。

公開鍵を、svnuser のホームディレクトリにある .ssh フォルダに追加してください。

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

生成した秘密鍵を使用するには、putty フォーマットに変換しなければなりません。これは秘密鍵のフォーマットが標準化団体によって指定されていないからです。秘密鍵ファイルを自分のクライアント PC にダウンロードした後、PuTTYgen を起動し、Conversions → Import key を指定してください。サーバから取得した keyfile ファイルを選び、鍵作成時のパスフレーズを使用してください。最後に Save private key をクリックして、keyfile.PPK を保存してください。

G.4.2. PuTTYgen を用いた鍵の作成

公開鍵・秘密鍵のペアを作成するのに PuTTYgen を使用して、保存してください。公開鍵をサーバにコピーし、svnuser のホームディレクトリにある .ssh フォルダに追加してください。

G.5. PuTTY を用いたテスト

接続のテストには、PuTTY を使用します。プログラムを起動し、Session タブで、hostname にサーバの名前か IP アドレスを設定し、プロトコルを SSH、保存するセッションに SvnConnection やその他適切な名前を指定してください。SSH

では、preferred SSH protocol version に 2 を設定し、Auth では先ほど変換した .PPK 秘密鍵ファイルへのフルパスを指定してください。Sessions タブに戻り、Save ボタンを押してください。これで saved sessions のリストに SvnConnection ができます。

Open をクリックして、telnet スタイルのログインプロンプトを表示してください。ユーザ名に svnuser を使用すると、すべてうまくいっていれば、パスワードのプロンプトが表示されずに、直接接続できるはずです。

サーバの /etc/sshd_config を編集する必要があるかも知れません。以下の行を編集し、その後 SSH サービスを再起動してください。

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

G.6. TortoiseSVN と併せた SSH のテスト

ここまで、SSH を用いたログインができるかどうかだけをテストしてきました。今度は、SSH 接続で実際に svnserve を実行できるかを、確認する必要があります。サーバでは、以下のように /home/svnuser/.ssh/authorized_keys を変更して、たくさんの subversion 作者に同じシステムアカウント svnuser を許可します。subversion 作者は同じログイン名ですが、異なる認証鍵を使用することに注意してください。そのため、全ユーザ用に 1 行追加しなければなりません。

注意: これ全部で、非常に長い 1 行です。

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

設定に従って、セットする必要があるいくつかの値があります。

<ReposRootPath> は、リポジトリがあるディレクトリへのパスに置き換えられます。これにより、URL 中にサーバまでのフルパスを指定せずともよくなります。Windows サーバであっても、通常のスラッシュを使用しなければならないことに注意してください (例: c:/svn/reposroot)。以下の例では、repos というリポジトリルート以下のリポジトリフォルダと仮定します。

<author> は、コミット時に格納したい svn author で置き換えます。これはまた、svnserve が svnserve.conf で指定した自身のアクセス権を使用するようにもできます。

<PublicKey> は先ほど生成した公開鍵に置き換えられます。

<Comment> は、お好みのコメントを記述できますが、svn author 名と実名とのマッピングに使うと便利です。

Windows エクスプローラのフォルダで右クリックし、TortoiseSVN → リポジトリブラウザ を選択してください。URL の入力を求められるので、次の形で入力してください。

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

この URL はどのような意味でしょうか? サーバへのリクエストの扱い方を、TortoiseSVN に教えるスキーマ名は、svn+ssh です。スラッシュ 2 つの後、サーバに接続するユーザ名を指定します。この場合は svnuser です。@ の後、PuTTY のセッション名を指定します。このセッション名には、秘密鍵とサーバの IP アドレスや DNS 名をどこから取得するかといった詳細が、すべて含まれています。最後にリポジトリへのパスを指定しなければなりません。authorized_keys に指定したように、サーバのリポジトリルートからの相対パスとなります。

OK をクリックすると、リポジトリの内容を閲覧できるでしょう。こうなれば、TortoiseSVN と連動して、SSH トンネリングで動作したことになります。

デフォルトでは、TortoiseSVN は自身が持つバージョンの Plink を、接続に使用することに注意してください。これは、接続するたびにコンソールウィンドウがポップアップするのを防ぎますが、エラーメッセージを出力する先がないということでもあります。「Unable to write to standard output」というエラーを受け取ったときは、TortoiseSVN のネットワーク設定で、クライアントに Plink を指定するようにしてみてください。これにより Plink が生成した真のエラーメッセージを見られるようになります。

G.7. 様々な SSH の設定

TortoiseSVN で URL を簡単にする方法は、PuTTY セッションの中でユーザを設定することです。このためには、PuTTY で定義済みのセッション SvnConnection を読み込み、Connection タブの Auto login user にユーザ名 (例: svnuser) を設定してください。前述のように、PuTTY のセッションを保存して、TortoiseSVN で以下の URL を試してください。

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

今回、TortoiseSVN が使用する SSH クライアント (TortoisePlink.exe) に、PuTTY のセッション SvnConnection を与えるだけです。このクライアントは、必要なすべての詳細を取得するのに、セッションをチェックします。

執筆時点では、PuTTY は保存された設定のすべてをチェックしていないため、同じサーバ名に対する複数の設定があると、一致した最初のものを使用してしまいます。また、デフォルト設定を編集して保存すると、自動ログインのユーザ名は、保存されません。

多くの人は、自分の鍵を格納するのに、Pageant を使用するのを好みます。PuTTY セッションには、鍵の格納機能があるため、Pageant が常に必要というわけではありません。しかし、異なるサーバへの鍵を保存することを想像してみてください。この場合、接続しようとするサーバにより、PuTTY のセッションを何度も編集を繰り返さねばならないでしょう。PuTTY, Plink, TortoisePlink やその他の PuTTY ベースのツールは、SSH サーバに接続する際、接続を開始するのに Pageant が保持しているすべての秘密鍵をチェックするので、この状況下では Pageant は完全に筋が通っています。

このタスクのためには、単に Pageant を実行し、秘密鍵を追加してください。先ほど PuTTY のセッションに定義した秘密鍵と同じにするべきです。Pageant を秘密鍵の格納に使用する場合、PuTTY セッションから、秘密鍵ファイルへの参照を削除できます。もちろん、他のサーバ向けや他のユーザの鍵を追加できます。

クライアントマシンを再起動するたびに、この手順を繰り返したくなければ、Pageant を Windows の自動起動グループに配置しましょう。Pageant.exe のコマンドライン引数として、鍵の完全パスを追加できます。

SSH サーバに接続する最後の方法は、単純に TortoiseSVN で以下の URL を使用することです。

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

ご覧の通り、保存した PuTTY のセッションを使用せず、IP アドレス (やドメイン名) を接続ターゲットに指定します。ユーザを指定することはできますが、秘密鍵ファイルをどのように見つけるかを尋ねられることでしょう。TortoisePlink.exe は、PuTTY スイートの通常の Plink ツールを改変したものですので、TortoiseSVN はまた、Pageant に格納してあるすべての鍵に対して試行します。

最後の方法を利用する場合、PuTTY にデフォルトユーザ名が設定されていないことを、確認してください。設定されている場合に、PuTTY が接続を切断してしまうというバグ報告を受けています。デフォルトユーザを削除するに

は、HKEY_CURRENT_USER¥Software¥SimonTatham¥Putty¥Sessions¥Default%20Settings¥HostName を単純にクリアしてください。

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ののリビジョンです。BASE リビジョンは、通常 HEAD リビジョンと同じではありません。
BDB	Berkeley DB。十分テストされているリポジトリ用データベースバックエンドですが、ネットワーク共有できません。1.2 以前のリポジトリのデフォルトです。
FSFS	リポジトリ用の Subversion が持つファイルシステムバックエンドです。ネットワーク共有が可能です。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の最新リビジョンです。
Revision	変更セットのコミット時、常に新しい 「リビジョン」 をリポジトリに作成します。各リビジョンは、履歴の決まった場所に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指し示しています。
SVN	Subversion のよく使われる省略表現。 「svnserve」 リポジトリサーバで使われる、Subversion カスタムプロトコルの名前です。
インポート	ひとつのリビジョンで、フォルダ階層のエントリをリポジトリにインポートするコマンドです。
エクスポート	このコマンドは、作業コピーと同様にバージョン管理下のフォルダをコピーしますが、.svn は作成しません。
クリーンアップ	以下 Subversion book から引用します。「作業コピーを再帰的にクリーンアップ (ロックの削除、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 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	手元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 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、手元の作業コピーを作成する Subversion コマンドです。
パッチ	作業コピーにテキストファイルの変更のみある場合、Unified Diff 形式で変更内容の単一ファイルを作成するのに、Subversion の Diff コマンドを使用できます。この形のファイルは、よく 「パッチ」 と言われており、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できます。コミットアクセスできない

人は、権限のあるコミッタが適用するように、変更をパッチファイルにして送ることができます。また、変更に自信がなければ、他の人に見てもらうようにパッチを送れます。

ブランチ

ある点で開発が2つの独立したパスに分岐したことをと表す、リビジョン管理システムの用語です。メインラインを不安定にせずに新機能の開発を行うように、メインの開発ラインからブランチを作成できます。また、今後バグフィックスしか行わない安定版リリースのブランチを作成し、新機能の開発は不安定なトランクで行えます。Subversion のブランチは、「簡易コピー」として実装されています。

マージ

リポジトリに追加された変更を、手元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 のマージツールを用いて、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。

リビジョン属性 (revprop)

ちょうどファイルが属性を持てるように、リポジトリの各リビジョンも属性を持てます。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 revprops が自動的に作成され、それぞれコミット日時、コミッタ、ログメッセージを表しています。これらの属性は編集できますが、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

データを格納し保守する中心部。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のリビジョンの歴史を表します。「履歴」とも表します。

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き、ロックが外れるまでリポジトリにコミット不可の印を付けます。

作業コピー

手元の「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。また通常手元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、リポジトリが移動する場合、作業コピーを「再配置」して、リポジトリの URL を新しい場所に指し示してください。

注意: このコマンドは、作業コピーが同じリポジトリ、同じ場所を指していて、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「リビジョンへの更新」が履歴上の別のポイントへ、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ポイントへ、作業コピーの場所ウィンドウを変更します。違いが少ししかないトランクとブランチの双方に作業する際に、これが特に役に立ちます。その 2 つの間で作業コピーを切り替え、違いのあるファイルのみを転送します。

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に、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。

取り消し
作業コピーを最後に更新したときから、Subversion はそれぞれのファイルの 「当初の」 コピーを手元に保持しています。変更を行い、その変更を取り消したい場合は、「取り消し」コマンドを使用して当初のコピーに戻せます。

属性
ディレクトリやファイルをバージョン管理下に置くのに加えて、Subversion はバージョン管理下の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・ファイルごとの「属性」として参照されます。属性には、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属性がいくつかあります。TortoiseSVN にも `tsvn:logminsize` のような特殊な属性があります。お好みの名前と値を持つ属性の追加もできます。

履歴
ファイルやフォルダのリビジョンの歴史を表します。「ログ」とも表します。

差分
「差分表示 (Show Differences)」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、すべての行に対して、最後に変更したリポジトリのリビジョンと、誰が変更したのかを注釈します。我々の GUI 実装では、TortoiseBlame を呼び出し、リビジョン番号の上にマウスを持っていくと、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手元にマージされる際、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、ファイルを手で修正し競合を解消しなければなりません。

解消
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 (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 「競合の解消」 と言われています。競合したファイルに解消マークを付けると、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

A

C

D

F

G

I

M

N

R

S

T

U

V

W